Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Simulador de los axiomas de Huzita para hacer construcciones con origami

Proyecto Tecnológico

Reporte Final

Trimestre 2015 Invierno

Presentan:

David Gonzalo Flores Navarro
209329851

Fernando Mauricio Arredondo Dana
209301502

Asesor: Alejandro Aguilar Zavoznik

9 de abril de 2015

Declaratoria

Yo, Alejandro Aguilar Zavoznik, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, David Gonzalo Flores Navarro, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Fernando Mauricio Arredondo Dana, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Resumen

Recientemente ha proliferado el estudio del origami desde un punto de vista matemático, particularmente se han encontrado similitudes en el estudio de los dobleces de papel y las construcciones con regla y compás que se han estudiado desde la Grecia antigua. Así como los axiomas de Euclides describen lo que se puede hacer con líneas rectas y círculos, los axiomas de Huzita describen los dobleces permitidos en una hoja de papel, por ejemplo el primer axioma nos indica que podemos hacer un doblez que contenga dos puntos dados.

En el presente reporte se explica como fue el desarrollo del proyecto "Simulador de los axiomas de Huzita para hacer construcciones con origami". En éste desarrollamos un programa que, mediante una animación, simula los dobleces en una hoja de papel al aplicar cada uno de estos axiomas.

Este trabajo cuenta con dos partes importantes que se complementan. Por un lado en el "marco teórico" se presenta la matemática necesaria para hallar las rectas que representan a los dobleces de cada uno de los seis axiomas; por otro lado, en el "desarrollo del proyecto" se describe la interfaz gráfica que muestra de forma animada los dobleces resultantes. Finalmente, en la sección de "análisis y resultados" se describe el programa obtenido.

El resultado final del proyecto es un programa desarrollado en lenguaje de programación Java en el entorno NetBeans del cual se aprovecharon sus clases predefinidas tipo "jFrame" para programar ventanas y así tener la interfaz gráfica necesaria para interactuar con el usuario. Además de esto, se usó un panel en el cual se muestra la simulación de una hoja de papel y cómo ésta se deforma con el uso de cada axioma.

Consideramos que el desarrollo de este simulador tiene relevancia en el sentido de que puede mostrar, virtualmente, la ejecución de los axiomas y con esto el usuario puede trabajar de una manera práctica, ya que le permite trazar líneas y puntos de una forma precisa usando coordenadas en un plano cartesiano, mientras que en una hoja de papel es complicado trazar o elegir los dobleces que se desean hacer.

Índice general

Índice de figuras

Capítulo 1

Introducción

El origami es el arte japonés que consiste en el doblado del papel sin utilizar algún otro objeto a excepción de las manos. Existen varios trabajos en los que se han relacionado el origami y las matemáticas, en particular, el artículo de Huzita [**1**], donde se plantean seis axiomas para hacer construcciones con origami.

1. Dados dos puntos p_1 y p_2, hay un único doblez que pasa a través de ellos.
2. Dados dos puntos p_1 y p_2, hay un único doblez que pone a p_1 sobre p_2.
3. Dadas dos líneas ℓ_1 y ℓ_2, existen dobleces que ponen a ℓ_1 sobre ℓ_2.
4. Dado un punto p_1 y una línea ℓ_1, hay un único doblez perpendicular a ℓ_1 que pasa a través de p_1.
5. Dados dos puntos p_1 y p_2 y una línea ℓ_1, existen dobleces que ponen a p_1 sobre ℓ_1 pasando a través de p_2
6. Dados dos puntos p_1 y p_2 y dos líneas ℓ_1 y ℓ_2, existen dobleces que ponen a p_1 sobre ℓ_1 y a p_2 sobre ℓ_2.

Usando esto, podemos resolver problemas tanto geométricos como algebraicos, por ejemplo, es posible obtener un triángulo equilátero a través de origami usando el procedimiento de la figura 1.

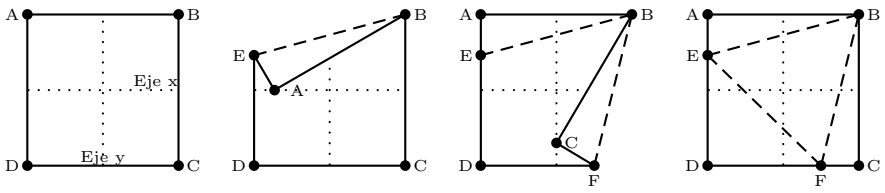


FIGURA 1. Construcción con origami de un triángulo equilátero.

El presente proyecto tiene como objetivo realizar un programa que permita al usuario proporcionar una sucesión de dobleces usando los axiomas de Huzita, los cuales serán mostrados en una animación.

1.1. Antecedentes

En esta sección se mostrarán algunos trabajos relacionados con el proyecto que se realizó.

Referencias internas

- **Algoritmos para plegado de mapas cuadrados [2]:** En este trabajo también se aborda el tema del doblado del papel, incluso el autor realiza simulaciones visuales del doblado tal y como se plantea en el simulador de los axiomas de Huzita, sin embargo las simulaciones de este proyecto son en dos dimensiones, mientras que las del antecedente son de tres, lo cual implica una ventaja para lo que aquí se propone, ya que hay una reducción en los recursos necesarios para las simulaciones. Otra diferencia es cómo se hace el doblado, mientras que en la referencia que aquí se cita se buscan algoritmos de plegados ortogonales, en el proyecto que se realizó el doblado se hace mediante los axiomas de Huzita.
- **Simulador cuerpos rígidos: esfera y terreno accidentado [3]:** En este proyecto se habla de una simulación, sin embrago no es de papel, se plantea la simulación de una esfera y para esto se emplea la física que tendría ésta al moverla, al contrario en el proyecto que aquí se presenta, la física del movimiento no es considerada como tal, solo se considera una animación que emule el doblado del papel.

- **Ambiente de programación visual para graficar geometrías simples mediante transformaciones ortogonales con álgebra geométrica y álgebra matricial** [4]**:** El objetivo de este proyecto es crear un software capaz de hacer gráficas de objetos geométricos, para ello se apoya en el álgebra vectorial y el álgebra geométrica. El proyecto que realizamos también es un software que utiliza el álgebra geométrica para poder graficar, en este caso no figuras geométricas, más bien simular el doblado del papel.

Referencias externas

- **One-, Two- and Multi-Fold Origami Axioms** [5]**:** En este artículo se describen los axiomas que se utilizan para poder hacer construcciones de origami, tal y como se plantea en este proyecto, se usan los puntos y líneas para aplicar los axiomas, además en el artículo se usan ejemplos para la solución de problemas matemáticos utilizando el origami, lo cual representa una posible utilidad del software resultante de la realización de este proyecto.
- **Propuesta metodológica para la enseñanza de las secciones cónicas en el grado décimo de la institución educativa villas de san Ignacio de Bucaramanga** [6]**:** En esta tesis, en particular en la sección de "Cuarta Fase" página 49, se describen los axiomas para la generación de construcciones de origami. Esta referencia es un trabajo orientado a la enseanza, con esto podemos ver una utilidad didáctica del software que resultaría de la realización de este proyecto.

1.2. Justificación

Existen diversos problemas matemáticos que pueden ser resueltos mediante el uso de los axiomas de Huzita, sin embargo no siempre es sencillo realizar los dobleces necesarios. El software que se desarrollado cuenta con herramientas que proporcionan una simulación de estos elementos, con el fin de ayudar al estudio de dichas cuestiones. A la hora de realizar las construcciones, se utilizaron conceptos correspondientes a diversas áreas de las matemáticas, por ejemplo, geometría analítica y álgebra lineal. Uno de los objetivos de este proyecto fue observar cómo interactúan estos elementos con las técnicas de programación que se han aprendido a lo largo de la carrera de Ingeniería en Computación.

Para mostrar una correcta simulación del doblado del papel es necesaria la aplicación de los conocimientos adquiridos relacionados con la graficación, además esto implicará esconder la matemática asociada a los axiomas de Huzita, para que así el usuario solo vea la animación de un doblez.

1.3. Objetivos

Objetivo General.

Desarrollar un sistema que implemente los axiomas de Huzita para hacer construcciones animadas con origami.

Objetivos Específicos.

1. Permitir al usuario emplear cualquiera de los 6 axiomas mediante una interfaz gráfica.
2. Programar la geometría y el álgebra necesaria para la implementación de los axiomas de Huzita.
3. Crear una animación para la visualización gráfica de la simulación del doblado de la hoja de papel.

Capítulo 2

Marco teórico

En este capítulo expondremos las herramientas matemáticas que fueron empleadas para el desarrollo del simulador de los axiomas de Huzita. Entre lo que fue utilizado, se encuentran conceptos de las materias de cálculo, algebra lineal y geometría analítica.

Este trabajo se desarrolla considerando que la simulación de los dobleces será en una hoja que, colocada sobre el plano cartesiano, tiene como vértices los puntos (-100, -100), (100, -100), (100, 100) y (-100, 100). A continuación explicaremos las matemáticas que empleamos para programar la simulación de cada uno de los axiomas.

2.1. Axioma 1

En este axioma hay que hacer el doblez que pasa por dos puntos, p_1 y p_2, dentro de la hoja de papel. Esto lo hacemos usando la ecuación punto-punto de una recta:

$$(y - y_1) = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1),$$

donde $p_1 = (x_1, y_1)$ y $p_2 = (x_2, y_2)$. La ecuación nos da la recta resultante después de hacer un doblez que pase por los dos puntos.

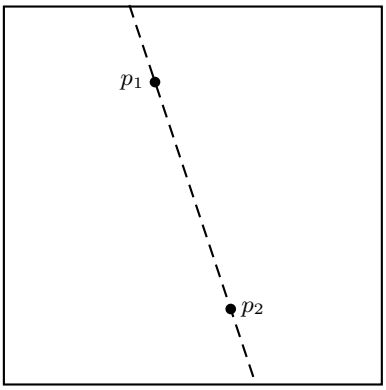


FIGURA 2. Axioma 1, doblez que pasa por p_1 y p_2.

2.2. Axioma 2

El segundo axioma pide doblar la hoja de tal forma que pongamos un punto p_1 sobre otro punto p_2. Sea ℓ_1 la recta que pasa por p_1 y p_2. La recta ℓ_2, que representa al doblez que se usa para simular este axioma, es perpendicular a ℓ_1 y pasa por el punto medio de p_1 y p_2, este último se denotará como p_m.

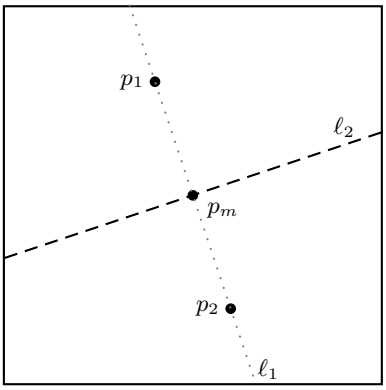


FIGURA 3. Axioma 2, doblez generado al colocar a p_1 sobre p_2.

Sean $p_1 = (x_1, y_1)$ y $p_2 = (x_2, y_2)$. El primer paso para encontrar ℓ_2 es obtener la pendiente de ℓ_1:

$$m_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)},$$

de donde la pendiente de ℓ_2 es:

$$m_2 = -\frac{1}{m_1}.$$

La ecuación de ℓ_2 tiene pendiente m_2 y pasa por el punto p_m, cuyas coordenadas son:

$$p_m = \left(\frac{(x_1 + x_2)}{2}, \frac{(y_1 + y_2)}{2} \right).$$

Con lo anterior, la ecuación de ℓ_2 es:

$$y - \frac{(y_1 + y_2)}{2} = m_2 \left(x - \frac{(x_1 + x_2)}{2} \right).$$

Esta ecuación define la recta que simula el doblado de papel que pone a un punto sobre el otro.

2.3. Axioma 3

Ahora queremos colocar una línea ℓ_1 sobre otra línea ℓ_2. Para obtener la recta que simula el doblez al encimar estas dos líneas, se tienen los siguientes casos:

Si ℓ_1 y ℓ_2 son paralelas entonces la recta que buscamos también debe de ser paralela a estas líneas, por lo que la pendiente m de las tres rectas es la misma.

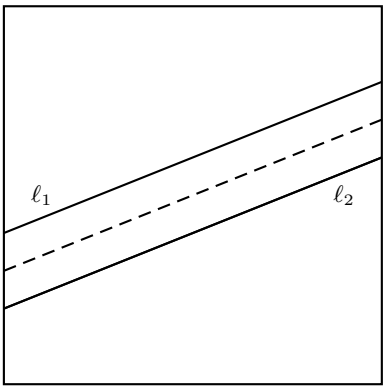


FIGURA 4. Axioma 3, doblez generado al colocar a ℓ_1 sobre ℓ_2.

Para trazar la recta que buscamos, necesitamos que ésta pase justo a la mitad de ℓ_1 y ℓ_2. Tomamos un punto p_1 de ℓ_1, el que es definido como $p_1 = (p_1x, p_1y)$ y luego a p_2 de ℓ_2 , el cual es $p_2 = (p_2x, p_2y)$. Obtenemos el punto medio p_m entre p_1 y p_2:

$$p_m x = \frac{(p_1 x + p_2 x)}{2},$$
$$p_m y = \frac{(p_1 y + p_2 y)}{2},$$
$$p_m = (p_m x, p_m y),$$

Usando el punto p_m y la pendiente m encontramos la ecuación de la recta:

$$(y - p_m y) = m(x - p_m x).$$

En caso de que ℓ_1 y ℓ_2 no sean paralelas se deben de usar los ángulos que forman éstas con respecto al eje de las x.

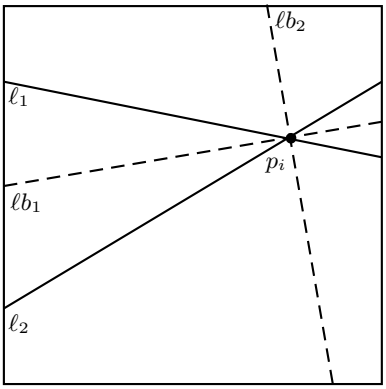


FIGURA 5. Posibles dobleces considerando que ℓ_1 y ℓ_2 no son paralelas.

Para encontrarlos hacemos lo siguiente:

$$\alpha_1 = \arctan(m_1),$$
$$\alpha_2 = \arctan(m_2),$$

De donde m_1 es la pendiente de ℓ_1 y m_2 es la pendiente de ℓ_2. El ángulo es el promedio de α_1 y α_2, el que se obtiene mediante:

$$\alpha_m = \frac{(\alpha_1 + \alpha_2)}{2}.$$

Notemos que hay dos posibles dobleces en este caso: el primero genera una línea representada por la recta ℓb_1, la cual forma un ángulo α_m con respecto a la horizontal. La segunda posibilidad es representada por la recta ℓb_2, donde ℓb_1 y ℓb_2 son perpendiculares. Sea p_i la intersección de ℓ_1 y ℓ_2. Ya teniendo las inclinaciones de ℓb_1 y ℓb_2, y el punto p_i que pasa por las dos rectas, podemos trazarlas, obteniendo así las representaciones de los dobleces resultantes de aplicar el axioma 3 a ℓ_1 y a ℓ_2.

2.4. Axioma 4

Para hacer que un doblez pase por un punto p y sea perpendicular a una línea ℓ se usa este axioma. Si la pendiente de ℓ es m, entonces la pendiente de la recta resultante es:

$$m_d = -\frac{1}{m},$$

como queremos que esta recta pase por p, considerando que $p = (px, py)$, entonces usamos la ecuación punto pendiente:

$$y - py = m_d(x - px),$$

con la que obtenemos el doblez generado al aplicar el axioma 4 sobre p y ℓ.

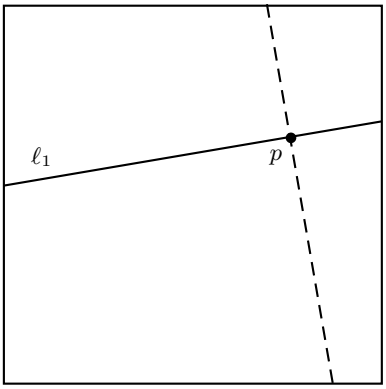


FIGURA 6. Axioma 4, doblez que pasa por p y es perpendicular a ℓ.

2.5. Axioma 5

El axioma dice lo siguiente: "Dados dos puntos p_1 y p_2 y una línea ℓ_1, hay dobleces que ponen a p_1 sobre ℓ_1 pasando a través de p_2." Entonces tenemos una situación como la que se muestra en la figura 7.

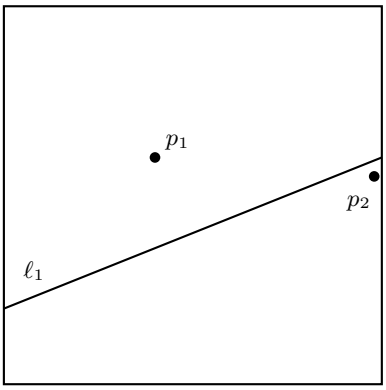


FIGURA 7. Datos necesarios para aplicar el axioma 5.

Teniendo en cuenta lo anterior, p_1 debe de quedar sobre algún punto de ℓ_1 y el doblez que lo permite debe de pasar por p_2. Para simular éste doblez hicimos equivalente el anterior sistema con el que se observa en la figura 8.

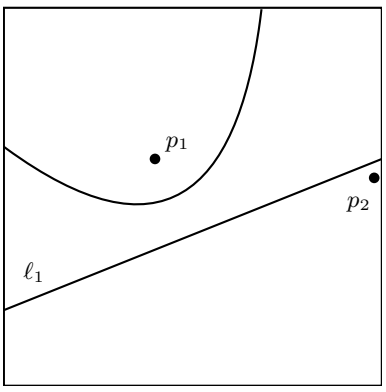


FIGURA 8. Parábola generada con el foco p_1 y con la directriz ℓ_1.

Como podemos observar en la figura, tenemos una parábola generada con el foco p_1 y con la directriz ℓ_1, además de que contamos con un punto p_2. La figura 9 nos muestra como se puede resolver el axioma dando como resultado que p_1 queda sobre ℓ_1 y el doblez pasa por p_2.

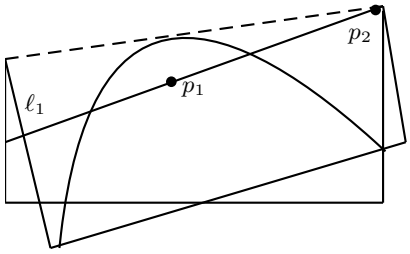


FIGURA 9. Doblez hecho para que p_1 sea colocado sobre ℓ_1 y que pase por p_2.

Para establecer el doblez de la figura 9, se genera una parábola con ℓ_1 como directriz y con p_1 como foco. Esto es conveniente, ya que la distancia del foco a un punto de la parábola es igual a la que hay de ese punto a la directriz, por lo tanto si se hace un doblez sobre una recta tangente a la parábola es seguro que el foco quede sobre algún punto de la directriz.

Al conocer esto, podemos poner a p_1 sobre ℓ_1 para cumplir con el correcto doblez; solo nos falta asegurarnos que éste pase por p_2. Así la línea que buscamos es tangente a la parábola y pasa por p_2.

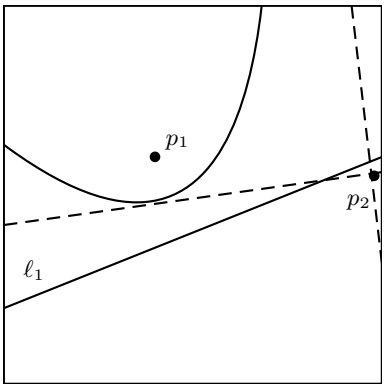


FIGURA 10. Dobleces posibles que son tangentes a la parábola y están sobre $p2$.

Como vemos en la figura 10, para este sistema de puntos y línea en particular, hay dos dobleces posibles que son tangentes a la parábola y están sobre p_2. En general, dependiendo de la posición de los puntos, puede haber una, dos o ninguna línea que cumpla con las condiciones establecidas.
Las matemáticas detrás de la explicación anterior son las siguientes:

Primero obtenemos la ecuación de la parábola generada por p_1 y ℓ_1. Para esto, rotamos todo el sistema un ángulo α de tal forma que ℓ_1 quede horizontal, la ecuación es:

$$(x-a)^2 + b^2 + c^2 = 2(b-c)y,$$

de donde la directriz, al ser horizontal, es $y = c$ y el foco es el punto (a, b). Después reorganizamos todos los términos de tal forma que nos quede la ecuación general de la parábola:

$$y = px^2 + qx + r. \tag{1}$$

Ahora hay que trazar la recta tangente que pase por ésta y por p_2 rotado. La pendiente m de la recta tangente se puede obtener derivando la ecuación (1):

$$m = 2px + q.$$

Para trazar la recta utilizamos la ecuación punto pendiente $y - y_2 = m(x - x_2)$, ya que tenemos el punto p_2 rotado (p_2x', p_2y') y la pendiente m, solo sustituimos valores:

$$y - p_2y' = (2px + q)(x - p2x'),$$

remplazando a y, nos queda la ecuación de la siguiente manera:

$$(px^2 + qx + r) - p_2y' = (2px + q)(x - p_2x').$$

La ecuación anterior está en función de x, esta variable es la coordenada en el plano donde la parábola y la recta tangente que pasa por p_2y' tienen la misma imagen.

Al resolver la ecuación de segundo grado nos damos cuenta de cuantos posibles dobleces cumplen con lo establecido en el axioma 5. Si el discriminante asociado a la ecuación es positivo hay dos dobleces, si es cero solo es uno y si es negativo no hay doblez posible.

Para trazar la recta que buscamos, usamos la ecuación punto-punto con (x, px^2+qx+r) y p_2 rotado. Finalmente, rotamos todo el sistema -α para regresarlo a la posición normal.

2.6. Axioma 6

Dados dos puntos, p_1 y p_2, y dos líneas, ℓ_1 y ℓ_2, buscamos un doblez que coloca a p_1 sobre ℓ_1 y a p_2 sobre ℓ_2. Este axioma es equivalente a encontrar una linea que simultáneamente es tangete a dos parábolas y que se resuelve usando una ecuación de tercer grado que tiene a lo más tres soluciones. Las dos parábolas tienen como foco a p_1 y a p_2 y directrices ℓ_1 y ℓ_2 respectivamente.

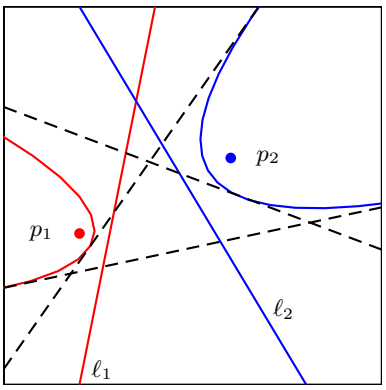


FIGURA 11. Posibles dobleces que colocan a p_1 sobre ℓ_1 y a p_2 sobre ℓ_2.

Para encontrar la ecuación de la parábola con foco p_1 y directriz ℓ_1 rotamos el sistema α radianes de tal manera que ℓ_1 quede paralela al eje de las x. En el sistema ya rotado definimos lo siguiente:

- $p_1 = (a_1, a_2)$
- $p_2 = (b_1, b_2)$
- $\ell_1 : (c_1, c_2) \leftrightarrow (c_3, c_2)$.
- $\ell_2 : (d_1, d_2) \leftrightarrow (d_3, d_4)$.

de donde la ecuación de la recta ℓ_1 es $y = c_2$.
La parábola asociada a p_1 y ℓ_1 es:

$$(x - a_1)^2 + a_2^2 - c_2^2 = 2(a_2 - c_2)y.$$

Para simplificar el uso de la ecuación, ésta queda de la siguiente manera:

$$y = e_1 x^2 + e_2 x + e_3,$$

donde:

- $e_1 = \dfrac{1}{2(a_2 - c_2)}$,
- $e_2 = \dfrac{-2a_1}{2(a_2 - c_2)}$,
- $e_3 = \dfrac{a_1^2 + a_2^2 - c_2^2}{2(a_2 - c_2)}$,

Procedemos a encontrar la ecuación que nos permite definir las rectas que son tangentes a la parábola. Suponemos que un punto de alguna de estas rectas es

$$(x_0, e_1 x_0^2 + e_2 x_0 + e_3),$$

y la pendiente m_1 de la recta tangente a este punto se obtiene con la derivada de la parábola evaluada en x_0:

$$m_1 = 2e_1 x_0 + e_2,$$

por lo que la recta tangente en este punto es:

$$y = m_1(x - x_0) + e_1 x_0^2 + e_2 x + e_3.$$

Sustituyendo a m_1 queda:

$$y = (2e_1 x_0 + e_2)(x - x_0) + e_1 x_0^2 + e_2 x_0 + e_3. \tag{2}$$

Solo un conjunto de estas rectas son las que nos interesa, aquellas que a su vez son tangentes a la parábola generada por la directriz ℓ_2 y el foco p_2. Para esto, primero encontramos la ecuación de la segunda parábola, de forma análoga a como encontramos la ecuación de la primera. Hacemos una segunda rotación al sistema de β radianes, de tal forma que ℓ_2 sea horizontal. La ecuación que nos da las rectas tangentes a esta segunda parábola es:

$$y = m_2(x - x_1) + g_1 x_1^2 + g_2 x_1 + g_3,$$

donde:

$$m_2 = 2g_1 x_1 + g_2.$$

Los valores de g_1, g_2 y g_3 se obtienen de la misma manera que los valores e_1, e_2 y e_3, solo que en lugar de usar la primera usamos la segunda parábola para obtenerlos. El valor de x_1 es la abscisa de un punto de las rectas. Regresamos el sistema al punto en el que ℓ_1 es horizontal, por lo que la ecuación que nos define las rectas tangentes a la segunda parabola queda:

$$y = \frac{(2g_1 x_1 + g_2)(\cos(-\beta)x - x_1) - \operatorname{sen}(-\beta)x + g_1 x_1^2 + g_2 x_1 + g3}{\cos(-\beta) + (2g_1 x_1 + g_2)\operatorname{sen}(-\beta)}.$$

Se cambian variables para facilitar el uso de la ecuación anterior:

$$y = \frac{(k_1 x_1 + k_2)x + k_3 x_1^2 + k_4 x_1 + k_5}{k_6 x_1 + k_7} \tag{3}$$

donde:

- $k_1 = 2g_1 \cos(-\beta)$.
- $k_2 = g_2 \cos(-\beta) - \operatorname{sen}(-\beta)$.
- $k_3 = -2g_1 + g_1 = -g_1$.
- $k_4 = -g_2 + g_2 = 0$.
- $k_5 = g_3$.
- $k_6 = 2g_1 \operatorname{sen}(-\beta)$.
- $k_7 = \cos(-\beta) + g_2 \operatorname{sen}(-\beta)$.

Para encontrar el conjunto de rectas tangentes necesitamos resolver un sistema dado por las ecuaciones (2) y (3). Encontramos las soluciones buscando las rectas que están en las dos familias, esto se obtiene al resolver la siguiente ecuación cúbica

$$0 = w_0 + w_1 x + w_2 x^2 + w_3 x^3,$$

donde:

- $w_0 = k_5 k_7 + \dfrac{k_2^2 - 2e_2 k_2 k_7 + e_2^2 k_7^2}{4e_1} - e_3 k_7^2.$
- $w_1 = k_5 k_6 + k_4 k_7 + \dfrac{2k_1 k_2 - 2e_2 k_1 k_7 - 2e_2 k_2 k_6 + 2e_2^2 k_6 k_7}{4e_1} - 2e_3 k_6 k_7.$
- $w_2 = k_4 k_6 + k_3 k_7 + \dfrac{k_1^2 - 2e_2 k_1 k_6 + e_2^2 k_6^2}{4e_1} - e_3 k_6^2.$
- $w_3 = k_3 k_6.$

Para hallar las raíces del polinomio se hizo uso del método de Cardano. El número de soluciones de la ecuación esta relacionado con su discriminante Δ

$$\Delta = 18 w_3 w_2 w_1 w_0 - 4 w_2^3 w_0 - 4 w_3 w_1^3 - 27 w_3^2 w_0^2.$$

1. Si $\Delta > 0$, hay una raíz real.
2. Si $\Delta = 0$, es posible que haya una única raíz que se repite tres veces, o bien, una raíz simple y una que se repite dos veces.
3. Si $\Delta < 0$, entonces hay tres raíces reales distintas.

Existe la posibilidad de que w_0 sea 0 por lo que hay una probabilidad de que la ecuación cúbica se convierta en una cuadrática.

Cada raíz x_1 está definiendo una recta tangente, para trazar esas rectas sustituimos los x_1 en la ecuación (3)

Por último, para que el sistema que esta rotado α radianes coincida con el sistema original, lo giramos -α radianes.

Capítulo 3

Desarrollo del proyecto

En esta sección explicaremos como hicimos el código que dio como resultado el programa que simula los dobleces resultantes al aplicar los axiomas de Huzita. Éste fue desarrollado en lenguaje Java, bajo el paradigma de orientación a objetos. Además se hizo una interfaz gráfica con la ayuda del entorno de desarrollo integrado NetBeans.

Lo primero que hicimos fue definir clases que abstrajeran los conceptos matemáticos (líneas y puntos) que se usan en el doblado de papel mediante los axiomas de Huzita.

3.1. cPunto y cPuntoDoble

Está definida por dos variables de clase, de tipo entero, x y y que representan la coordenada del punto en el plano. Asociada a ésta, definimos la clase *CpuntoDoble* la cual es análoga a *cPunto*, con la diferencia de que las coordenadas son de tipo doble.

3.2. cLinea

Hace referencia a las líneas sobre un plano cartesiano, esta se define a partir de dos variables de clase *cPunto* ($P1$ y $P2$), las cuales son objetos de tipo *CpuntoDoble*, tal y como una línea en el plano puede ser definida mediante dos puntos.

3.3. listaPuntos y listaLineas

Creamos las clase *listaPuntos* y *listaLineas* que ordenan objetos de tipo *cPunto* y *cLinea*, respectivamente. Estas clases están definidas por el objeto al inicio de la lista y al final de ésta, así como un tamaño *(int)*, que indica cuantos objetos están en la lista. Para acceder a los objetos de forma más rápida, también creamos métodos en cada clase que "llenan" arreglos con los elementos de cada lista.

3.4. panelPrincipal

Ésta hereda propiedades de *jPanel* (clase predefinida en NetBeans para dibujar). En esta clase se programaron varios métodos que dibujan líneas, puntos, y otros objetos (como polígonos) sobre este panel. Los paneles usan métodos de la librería gráfica de Java los cuales sirven para trazar figuras, lo que se hizo fue guiarse por estos de tal manera que respondieran a las necesidades del programa, por ejemplo se creó un método que colocara un cuadrado sobre el panel para así simular la hoja de papel la cual delimita el área de trazado, estableciéndola como un plano cartesiano que se mueve sobre las x de -100 a 100 y sobre las y con los mismos valores; el método a su vez hace que este cuadro se redibuje a lo largo de la ejecución del programa. Otros métodos de la clase reciben objetos previamente establecidos y los traza, como ejemplo tenemos los siguientes dos:

- El que recibe un objeto tipo *cPunto* y lo dibuja en las coordenadas correspondientes, tomando en cuenta que el $(0, 0)$ sería el centro del cuadrado que simula la hoja o bien el centro del mismo panel.
- Y el que recibe un objeto tipo *linea* y la dibuja en su lugar correspondiente dentro de la hoja simulada, incluso es posible pintarla del color que se requiera.

Estos dos métodos son usados en toda la ejecución del programa para dibujar lo que todos los elementos en la lista de puntos y en la lista de líneas. Aparte de los ya mencionados se crearon otros métodos con la finalidad de dibujar figuras que se usan para simular el doblado de la hoja de papel.

3.5. ventanaPrincipal

Se creó la clase *ventanaPrincipal* la cual es la interfaz que emplea el usuario; ya sea para introducir puntos, ejecutar algún axioma, borrar puntos, borrar líneas o crear archivos. Esta clase tiene tres variables de clase que serán las que se modificarán a lo largo de toda la ejecución del programa y sobre las cuales trabaja, estas son objetos de tipo *panelPrincipal*, *listaLineas* y *listaPuntos*. La clase hereda propiedades de *jFrame*, esto con la finalidad de programar botones que estén en esta ventana, la mayoría de estos botones crean objetos de clases que también heredan propiedades de *jFrame* por lo que también generan ventanas, los únicos botones que no despliegan otras ventanas son borrar puntos y borrar líneas.

3.6. ventanaPunto

Al hacer clic sobre el botón "Punto" de la ventana principal, se crea un objeto de la clase *ventanaPunto*. El método constructor de esta clase recibe al panel principal y a la lista de puntos, ya que estos serán modificados por la clase, a su vez el método genera una ventana que contiene dos casillas, correspondientes a las coordenadas x y y, y un botón, el cual al ser presionado crea un objeto *cPunto* con los valores de las casillas, este objeto es introducido a la lista de puntos y es dibujado en el panel principal.

3.7. ventanaAxioma1

En la ventana principal al hacer clic sobre el botón "Axioma1" se genera un objeto de esta clase, la que tiene los siguientes métodos:

- *Constructor*: el cual recibe al panel principal, a la lista de puntos y a la lista de líneas y a su vez genera la ventana que contiene un botón y dos listas desplegables las cuales se llenan con los puntos que hay en la lista de puntos.
- *jComboBox1ActionPerformed*: este método hace que cada vez que se elige un punto de una de las listas desplegables, lo asigna a la variable de clase *punto1* y lo dibuja de un color diferente en el panel principal, para indicar cuál es el punto que se está seleccionando. Adicionalmente, las coordenadas de los puntos se muestran en la etiqueta asociada a la lista desplegable.
- *seleccionaPunto2ActionPerformed*: es análogo al anterior solo que responde a la otra lista desplegable, de forma que el objeto seleccionado es asignado a la variable *punto2*.
- *aplicraConSeleccionadorActionPerformed*: Aquí se programa lo que se ejecuta al presionar el botón de "Aplicar", para esto se usan las variables *punto1* y *punto2*, previamente seleccionadas. El objeto se encarga de la animación y el almacenamiento de la línea de doblado que corresponde a aplicar el axioma1 sobre punto1 y punto2.

3.8. doblado

La clase es utilizada por todas aquellas que ejecutan los axiomas.

- *Constructor*: aquí se usa la línea definida por cuatro valores tipo doble los cuales son recibidos y asignados por este método y corresponden a las coordenadas de dos de los puntos que están sobre dicha recta.
- *hacerDoblado*: genera la línea de doblez, la que se divide en dos partes, en la primera se delimita la línea dentro de la hoja de papel, para esto se establecen dos objetos tipo *cPunto* que corresponden a los dos puntos extremos que están sobre la línea y cada uno de ellos toca un lado del cuadrado que simula la hoja. Los objetos *cPunto* definen entonces la clase *cLinea* que será introducida en la lista de líneas y que se dibuja en el panel. En la segunda parte se hace la animación del doblado de la hoja, es decir la simulación que corresponde a doblar la hoja de papel de tal manera que se marque la línea.

  Para hacer la simulación del doblado, primero se procede a dividir la hoja de papel en dos polígonos delimitados por la línea recibida en el constructor y los bordes de la hoja. Uno de estos dos polígonos será girado para simular el doblado, el ángulo de esta rotación corresponde al que tiene la línea con respecto a alguno de los lados del cuadrado que delimitan la hoja, de tal forma que el polígono quedará "reflejado" a partir de la línea. La animación empieza con el cuadrado de 200×200, después se comienza a deformar una sección de éste, puntualmente la parte del polígono que va a quedar reflejado; este proceso es gradual y se hace mediante el trazado de cuadriláteros o triángulos. A la mitad de este proceso, la región que quedará reflejada habrá desaparecido por completo. Este proceso se vuelve hacer pero de manera inversa simulando el doblez de "regreso" de la hoja de papel.

3.9. ventanaAxioma2

Al presionar el botón "Axioma2", en la ventana principal, se crea un objeto de esta clase en la cual su constructor recibe al panel principal, la lista de puntos y la lista de líneas, además de que este método genera una ventana similar a la de la clase *ventanaAxioma1*; en esta interfaz el usuario selecciona los puntos de las listas desplegables y estos se dibujan en el panel de otro color, después de seleccionarlos y dar clic en "Aceptar" se crea un objeto tipo *cLineaPerpendicularDados2Puntos* el cual usa los objetos *cPunto* que eligió el usuario y devuelve un objeto *cLinea* que es perpendicular a la línea sobre la que están dichos puntos, después se crea un objeto tipo doblado el cual se encarga de hacer la animación del doblez correspondiente a esa línea y de incluirla en la lista de líneas.

3.10. cLineaPerpendicularDados2Puntos

El método constructor de esta clase recibe dos objetos tipo *cPunto*. Otro método de la clase usa esos puntos para generar un objeto cLinea, la cual, si fuera dibujada, es perpendicular a la línea sobre la que están los puntos.

3.11. ventanaAxioma3

El botón "Axioma3" de la ventana principal crea un objeto de esta clase, para esto el constructor usa el panel principal, la lista de líneas y la lista de puntos. Además crea una ventana que contiene dos listas desplegables y un botón, los que le sirven al usuario para seleccionar los objetos dentro de la lista de líneas, al momento de seleccionarlas se dibujan de un color diferente en el panel principal; una vez que se han elegido y se da clic en el botón de "Aplicar", se usan las líneas para generar otra, la cual se trazaría si dobláramos la hoja de tal forma que colocáramos dichas líneas una sobre otra. Dependiendo de cuantas líneas sean las que cumplan, se tienen dos casos, el primero es cuando es solo una, en este caso se crea un objeto *cLinea* que sea equivalente y se usa la clase *doblado* para hacer la animación correspondiente, en caso de que sean dos, se crean entonces dos objetos tipo *lineaDeDobladoSinAnimacion* los cuales generarán a su vez dos objetos tipo *cLinea* respectivamente, estos objetos son usados por la clase seleccionDoblez, la cual crea una interfaz para que el usuario escoja cuál de las dos rectas es la que desea, al momento de elegirla se usa la clase doblez para crear la animación correspondiente al objeto *cLinea* que el usuario escogió.

3.12. lineaDeDobladoSinAnimacion

Cuando se genera un objeto de esta clase se reciben valores tipo doble que definen los puntos que están sobre la línea que se desea obtener, a partir de estos se crea un objeto *cLinea*, delimitada dentro de la hoja de papel, usando la misma metodología de la clase doblado. Es importante notar que no se hace la animación ni se introduce en la lista de líneas.

3.13. seleccionDoblez

Esta clase recibe dos *cLineas* y las traza de forma temporal en el panel sin que sean introducidas en la lista de líneas. Mediante una ventana, se le da a elegir al usuario una de estas dos, cuando lo hace se genera la animación correspondiente a la recta que eligió y la guarda en la lista de líneas.

3.14. ventanaAxioma4

Un objeto de esta clase es creado cuando se selecciona el botón "Axioma4" de la ventana principal. Esta clase tiene las siguientes variables de clase:

- *lp*, tipo *listaPuntos.*
- *h*, tipo *panelPrincipal.*
- *ll*, tipo *listaLineas.*
- *linea*, tipo *cLinea.*
- *punto*, tipo *cPunto.*

Éste axioma genera un doblez perpendicular a la línea y que pasa por el punto, para hacer esto la clase tiene los siguientes métodos:

- *Constructor*: le asigna a las variables de clase *lp*, *h* y *ll*, la lista de puntos, el panel principal y la lista de líneas que son usados a lo largo de la ejecución del programa, además inicializa los componentes de la ventana que emplea el usuario, estos componentes son dos listas desplegables y un botón, los seleccionadores son llenados con los objetos que hay en la lista de puntos y en la lista de líneas respectivamente.
- *seleccionador1ActionPreformed*: lo que se hace cada que se selecciona un objeto del seleccionador de los puntos está programado en este método. Cuando se elige un punto, se accede al panel principal y se cambia de color, además que a la variable de clase punto se le asigna este objeto en particular.
- *seleccionador2ActionPreformed*: es análogo al método anterior, solo que responde al seleccionador de las líneas y asigna la variable de clase cLinea con el objeto que se selecciona, la que se marca con un color diferente.
- *aplicarAxioma4ActionPreformed*: este método es la programación detrás del botón "Aplicar". Con la ecuación punto pendiente se encuentra la línea de doblez, la cual es perpendicular a la línea y pasa por el punto. Recordemos que estas variables de clase son asignadas mediante los dos métodos anteriores.

3.15. ventanaAxioma5

Cuando se presiona el botón "Axioma5" en la ventana principal se crea un objeto de este tipo, las variables de clase que lo conforman son las siguientes:

- *lp*, tipo *listaPuntos.*
- *h*, tipo *panelPrincipal.*
- *ll*, tipo *listaLineas.*
- *linea*, tipo *cLinea.*
- *punto*1, tipo *cPunto.*
- *punto*2, tipo *cPunto.*

Este axioma busca un doblez en la hoja que ponga al *punto1* sobre la línea y que dicho doblez pase por el *punto2*. Para esto se tienen los siguientes métodos:

- *Constructor*: este método asigna a la variables *lp*, *h* y *ll*, la lista de puntos, el panel principal y la lista de líneas que son usadas durante toda la ejecución del programa. También crea la interfaz que utiliza el usuario, ésta es conformada de tres listas desplegables y un botón. Las primeras son llenadas con los objetos que hay en la lista de puntos y en la lista de líneas.
- *comboBoxLineaActionPerformed*: aquí se programa lo que se hace cuando se están seleccionando objetos *cLinea* de la lista desplegable que fue llenada con la lista de líneas. Se le asigna a la variable *linea* el objeto que se esté seleccionando, trazándolo en el panel principal con un color diferente.
- *comboBoxP1ActionPerformed*: este método es similar al anterior, solo que aquí se programa lo que se hace cuando se está seleccionando un objeto *cPunto* de una de las listas desplegables que se llenó con la lista de puntos, el cual se asigna a la variable *punto1.*
- *comboBoxP2ActionPerformed*: este método es análogo al anterior, solo que lo que se selecciona es asignado a *punto2.*
- *aplicarA5ActionPerformed*: Al presionar el botón de "Aplicar", lo que se desencadena es lo que se programa en este método. Para poder generar la linea correspondiente al ejecutar el axioma sobre *punto1*, *punto2* y *linea*, se emplea lo estudiado en el "marco teórico" sección "Axioma5". Se establece la parábola, creándose para ello un objeto *cParabola*, pasando a su constructor la línea (directriz) y *punto1* (foco). Con la ecuación de la parábola y *punto2* se obtiene el sistema de ecuaciones que se necesitan para encontrar el doblez. Para que el objeto de la clase *parabola* nos devuelva la ecuación, se necesitan rotar las rectas y los puntos. Con la parábola ya rotada, se establece el sistema de ecuaciones que se reduce a una ecuación de segundo grado y se procede a resolverla.

3.16. cParabola

Esta clase abstrae el concepto de parabola, para esto tiene las siguientes variables de clase:

- *foco*, tipo *cPunto.*
- *directriz*, tipo *cLinea.*
- *anguloRotacion*, tipo *doble.*

La parábola es definida a partir de $foco$ y *directriz*, la variable *anguloRotacion* indica cuanto ha sido rotada una parábola, en caso de que este valor sea 0 quiere decir que ésta se encuentra en su posición original, en otro caso hay que usar ese valor para rotarla y dejarla en su estado inicial.

La clase cuenta con varios métodos, siendo los siguientes tres los principales:

- *Constructor*: recibe un objeto *cLinea* el cual es asignado a la variable de clase *directriz*, recibe un objeto *cPunto* y se le asigna a *foco*, y por último, si la parábola no es generada a partir de otra que fue rotada debe recibir el valor de 0, para asignárselo a *anguloRotacion*.
- *rotarParabola*: este método regresa un objeto *cParabola* y recibe un valor tipo doble que se asignará a la variable *angulo*, la que se usará para rotar el foco y la directriz. A partir de éstos se crea una nuevo objeto *cParabola*, su constructor recibe el foco rotado y la directriz rotada, además del valor de *angulo* sumado a la variable de *anguloRotacion*. Éste es el objeto que se devuelve al invocar el método.
- *obtenerEcuacion*: el método se encarga de obtener los coeficientes de la ecuación de la parábola en forma general a partir del foco y la directriz y los almacena en un objeto tipo *ecuacionParabola*.

3.17. ecuacionParabola

Este método almacena los tres coeficientes que definen la ecuación general de la parábola, cada uno de estos en una variable de clase, la cual es inicializada en el constructor y devuelta por los métodos *get*.

3.18. resuelveEcuacion2doGrado

Esta clase utiliza los coeficientes de la ecuación general de segundo grado para obtener los valores que resuelven la ecuación. Tiene cinco variables de clase: x_1 y x_2 donde se almacenaran los resultados de la ecuación, además de a, b y c que son los coeficientes. El método principal es resuelve, el cual aplica la fórmula para resolver ecuaciones de segundo grado y guarda el resultado en x_1 y x_2. Los valores de x_1 y x_2 se devuelven, respectivamente, mediante los métodos *get*.

3.19. ventanaAxioma6

Un objeto de esta clase es creado cuando se da clic en el botón "Axioma6" en la ventana principal. Se tienen las siguientes variables de clase:

- *lp*, tipo *listaPuntos*.
- *h*, tipo *panelPrincipal*.
- *ll*, tipo *listaLineas*.
- *linea*1, tipo *cLinea*.
- *linea*2, tipo *cLinea*.
- *punto*1, tipo *cPunto*.
- *punto*2, tipo *cPunto*.

El objetivo del axioma es hacer un doblez que coloque a *punto1* sobre *linea1* y a *punto2* sobre *linea2* al mismo tiempo. Para esto se cuenta con los siguientes métodos.

- *Constructor*: este método recibe la lista de puntos, el panel principal y la lista de líneas asignándolos a las variables *lp*, *h* y *ll* respectivamente. Inicializa los componentes de una interfaz, los cuales incluyen cuatro listas desplegables y un botón. Llena dos de las listas desplegables con la lista de líneas y las dos restantes con la lista de puntos.
- *comboBoxP1ActionPerformed*: este método programa lo que se hace cada vez que es seleccionado un objeto de una de las listas desplegables que se llenó con la lista de puntos, el objeto *cPunto* es asignado a la variable *punto1* y éste es dibujado con un color diferente en el panel principal.
- *comboBoxP2ActionPerformed*: es análogo al método anterior, solo que el objeto que es seleccionado de la otra lista desplegable, es asignado a la variable *punto2*.
- *comboBoxLinea1ActionPerformed*: este método es similar a los dos anteriores, pero trabaja con una de las listas desplegables de líneas. El objeto *cLinea* que es seleccionado es asignado a la variable *linea1* y este es marcado con un color diferente.
- *comboBoxLinea2ActionPerformed*: este método es análogo al anterior, asignando el objeto de la otra lista desplegable de líneas a la variable *linea2*.

- *jButton1ActionPerformed*: este método es invocado cuando se da clic en el botón de "Aplicar". El axioma se resuelve con un sistema de dos parábolas, la explicación exacta de la solución a este axioma se encuentra descrita en el marco teórico. Se crean dos objetos tipo *cParabola*, la primera parábola corresponde al *punto1* (foco) y *linea1* (directriz), al igual que en axioma 5 esta parábola es rotada para colocarla en posición vertical, para así obtener su ecuación. La segunda parábola debe quedar horizontal, por lo que se hace una nueva rotación de los dos puntos y las dos líneas. Con las ecuaciones de estas parábolas se establece el sistema de ecuaciones correspondiente a encontrar las rectas tangentes a las dos parábolas. El sistema de ecuaciones se reduce a una ecuación cúbica, la cual es resuelta por un objeto tipo *claseCardano*, el cual recibe los coeficientes de la ecuación y regresa sus raíces. En caso de que sea una sola raíz se establecerá una línea. En caso de que sean dos las raíces, se generaran dos líneas y al igual que en axioma 5 se usa la clase *lineaDobladoSinAnimacion* para crear los objetos *cLinea*, y se crea un objeto tipo *seleccionDoblez* para que el usuario decida entre las dos líneas. En caso de que haya tres raíces, se usa la clase *seleccionDeTresLineas*, similar a la clase *seleccionDoblez*, pero con tres opciones.

3.20. claseCardano

Esta clase es empleada para resolver una ecuación cúbica, para esto se cuenta con los siguientes métodos:

- *Constructor*: este método recibe cuatro variables de tipo doble, las cuales son equivalentes a los coeficientes de la ecuación general de tercer grado. Estos valores son asignados a cuatro de las variables de clase *(A, B, C, D)*.
- *resuelve*: aquí se usan *A, B, C, D* para aplicarlos a la fórmula matemática conocida como *método de Cardano*, la cual resuelve una ecuación cúbica a partir de su forma general. Este método es análogo al que resuelve la ecuación de segundo grado en la clase *resuelveEcuacion2doGrado* pero en lugar de establecer dos resultados, halla las tres raíces del polinomio y las asigna en las variables de clase *x0, x1, x2* a las cuales se puede acceder mediante los métodos *get*. Estas tres variables dependen del discriminante de la ecuación de tercer grado, el cual es obtenido mediante *A, B, C, D*, si éste es positivo, negativo o cero se hace el procedimiento correspondiente, establecido por Cardano, para obtenerlas.

Capítulo 4

Resultados y su análisis

En este capítulo analizaremos el programa en ejecución, sección por sección, el cual es el resultado final de este proyecto. El programa ya compilado genera un archivo ejecutable con extensión .jar el que se inicializa en cualquier sistema operativo o plataforma que tenga instalado la máquina virtual de Java (JVM). A continuación se mostrará y se describirá cada una de las ventanas resultantes:

4.1. Ventana Principal

Al ejecutar el .jar se mostrará la siguiente ventana (12), la cual llamamos "Ventana Principal":

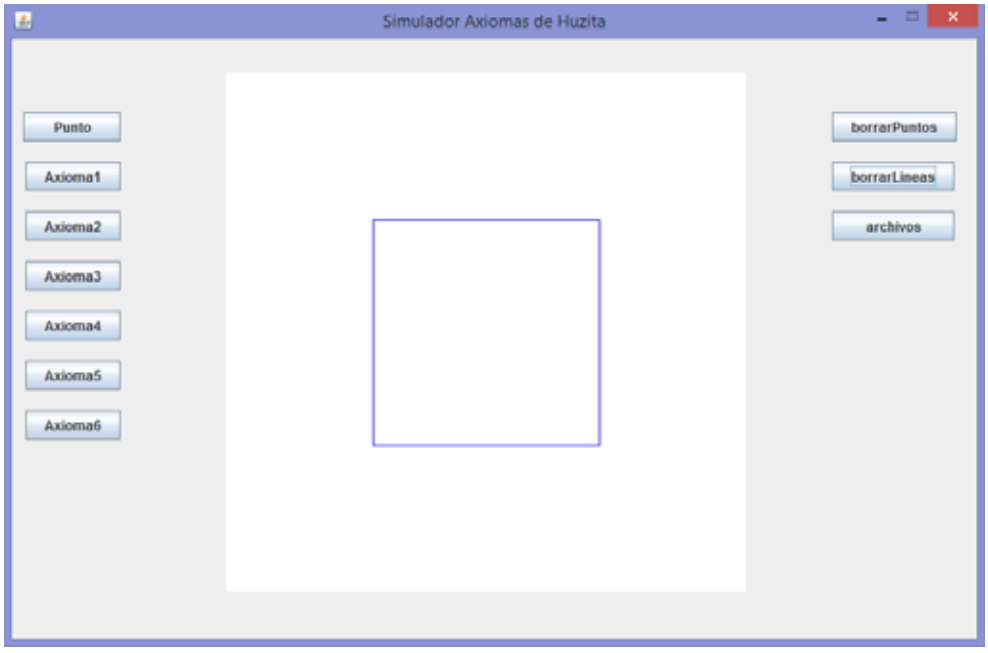


FIGURA 12. Ventana principal, la que contiene a la hoja de papel donde se ejecutan los axiomas.

La ventana anterior contiene un panel al centro y botones en sus extremos. El panel contiene la hoja que será modificada con cada uno de los axiomas, esta se visualiza como un cuadro azul, el cual se deforma mediante las animaciones que simulan los dobleces. Del lado izquierdo de la ventana se cuenta con los botones para la ejecución de cada uno de los axiomas de Huzita y a su vez, se encuentra el botón para poder crear puntos con coordenadas en un intervalo de -100 a 100, para que sean dibujados dentro de la hoja. Ubicados del lado derecho de la "Ventana Principal" se encuentran los botones para poder borrar los puntos y las líneas creadas. Además se tiene el botón "Archivo" el cual permite guardar y recuperar la información ya creada.

4.2. Ventana Punto

En la ventana principal, al hacer click sobre el botón “Punto”, se despliega la interfaz mostrada 13.

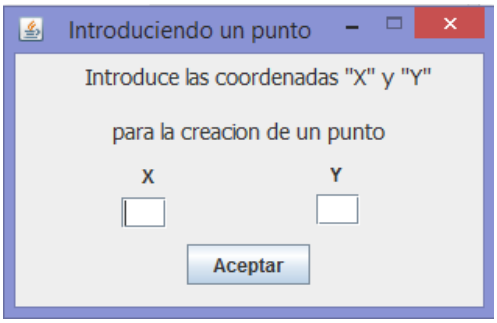


FIGURA 13. Ventana Punto, permite la creación de puntos.

Dicha ventana sirve para introducir las coordenadas del punto a dibujar, los que se introducen en las casillas X y Y respectivamente. Recordemos que el rango de valores permitidos deben ser de -100 a 100, de lo contrario el punto no se dibujará en la hoja.

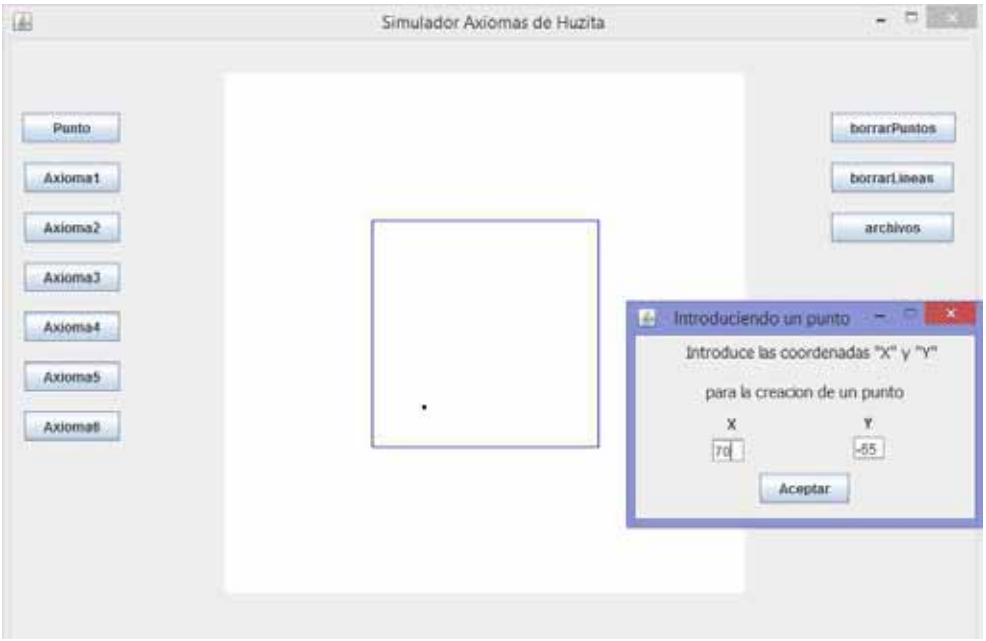


FIGURA 14. Punto dibujado al ser introducidas las coordenadas.

Si así se desea se puede seguir introduciendo cualquier cantidad de puntos hasta que el usuario cierre la ventana.

4.3. Ventana Axioma 1

Al presionar el botón “Axioma1” se despliega la ventana de la figura 15. La figura 16 muestra la ventana antes de aplicar el axioma con dos puntos introducidos, estos se seleccionan usando listas desplegables, en las que se visualizan sus coordenadas y se marca de un color distinto en la hoja. Para poder usar la ventana “Axioma1” es necesario haber introducido al menos dos puntos, de lo contrario esta ventana no ejecutará el axioma programado.

FIGURA 15. Ventana del Axioma 1, seleccionar dos puntos.

Para seleccionar los elementos sobre los cuales se aplica el axioma se usan las listas desplegables de dicha ventana, las cuales muestran los puntos previamente dibujados, al seleccionarlos se visualizan sus coordenadas y se marcan de un color distinto, tal como se muestra en la figura 16.

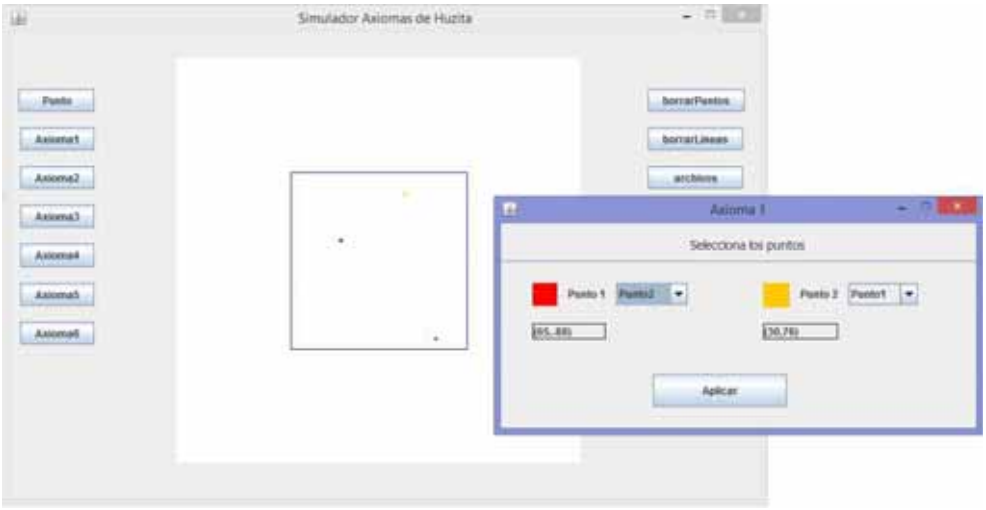
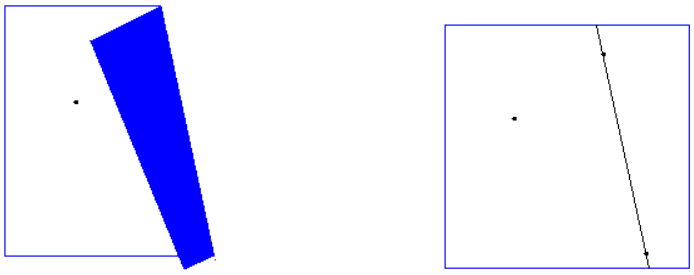


FIGURA 16. Marca de color distinto cada selección.

Al presionar el botón de "Aplicar" se ejecuta el axioma dando como resultado la simulación del doblado de la hoja de papel y después de esta animación queda trazada la línea del doblez:



(a) doblando (b) línea trazada

FIGURA 17. Doblez generado al ser aplicado el axioma.

4.4. Ventana Axioma 2

Para usar esta ventana, los puntos se seleccionan de forma similar a lo que se hace en ventana “Axioma1‘” tal y como se muestra en la figura 18. Recordemos que este axioma debe contar previamente con al menos dos puntos dibujados en la hoja de papel, ya teniendo dichos puntos se tiene que presionar el botón “Axioma2” el cual desplegará la siguiente ventana:

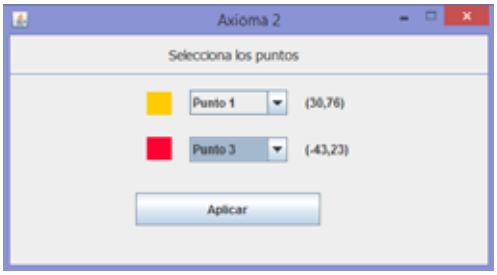


FIGURA 18. Ventana Axioma 2, seleccionar dos puntos.

En la ventana “Axioma2” hay que seleccionar, de las listas desplegables, los puntos sobre los que se ejecutará el axioma. En la hoja donde se encuentran trazados los puntos, el color cambiará dependiendo de cual sea el seleccionado.

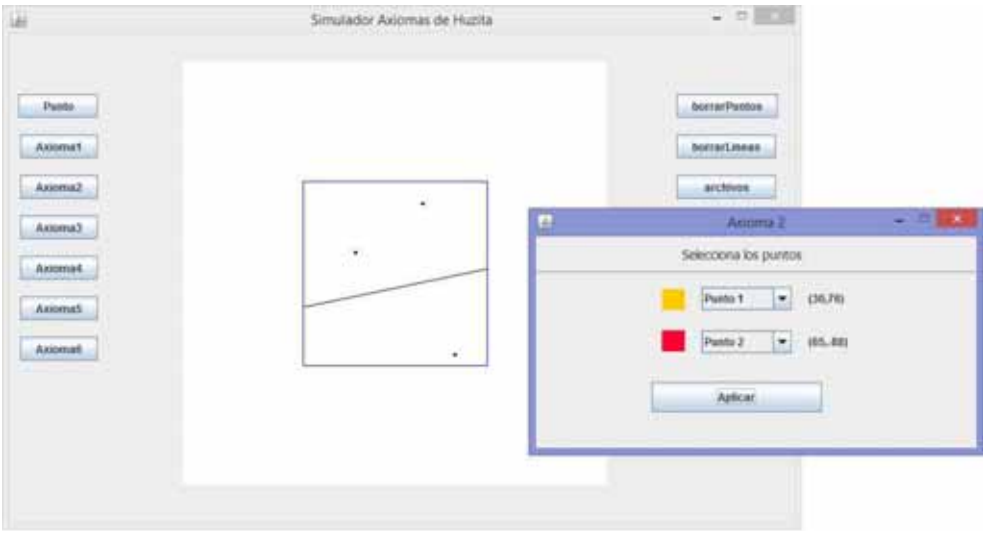
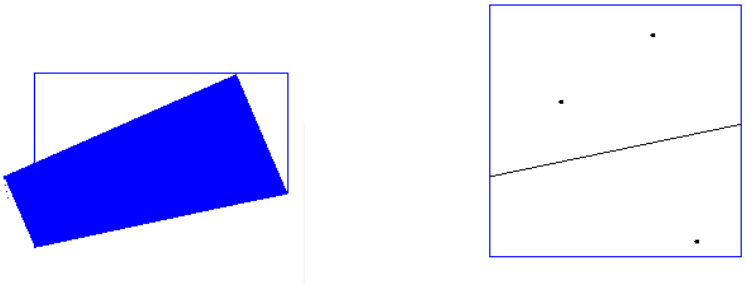


FIGURA 19. Se marcan de color los puntos seleccionados.

Ya teniendo marcados los puntos se procede a aplicar el axioma, el cual queda ejemplificado en la figura 20.



(a) doblando (b) línea trazada

FIGURA 20. Doblez generado al aplicar el axioma.

4.5. Ventana Axioma 3

Para aplicar el siguiente axioma es necesario contar con al menos dos líneas generadas por los otros axiomas. Una vez teniendo las líneas y presionando el boton "Axioma3", se depliega la ventana mostrada en la figura 21.

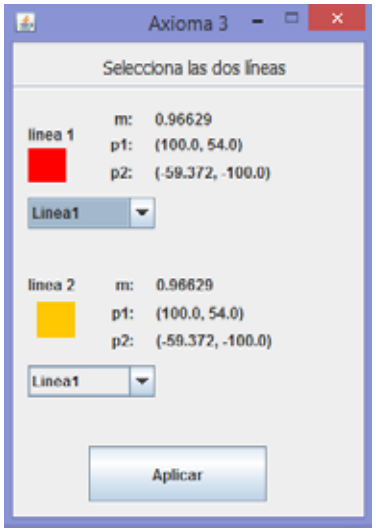


FIGURA 21. Ventana Axioma 3, muestra información acerca de las líneas a seleccionar.

En ésta se eligen las líneas sobre las que se aplica el axioma, esto mediante las listas desplegables. Cada vez que se selecciona una línea aparece su pendiente y las intersecciones de ésta con los bordes de la hoja, además de que se marca con un color. Este axioma tiene dos posibles casos dependiendo de la posición de las líneas seleccionadas. El primer caso se da cuando las dos líneas se intersectan dentro de la hoja de papel, cuando esto sucede, al aplicar el axioma se despliega algo como lo que se observa en la figura 22.

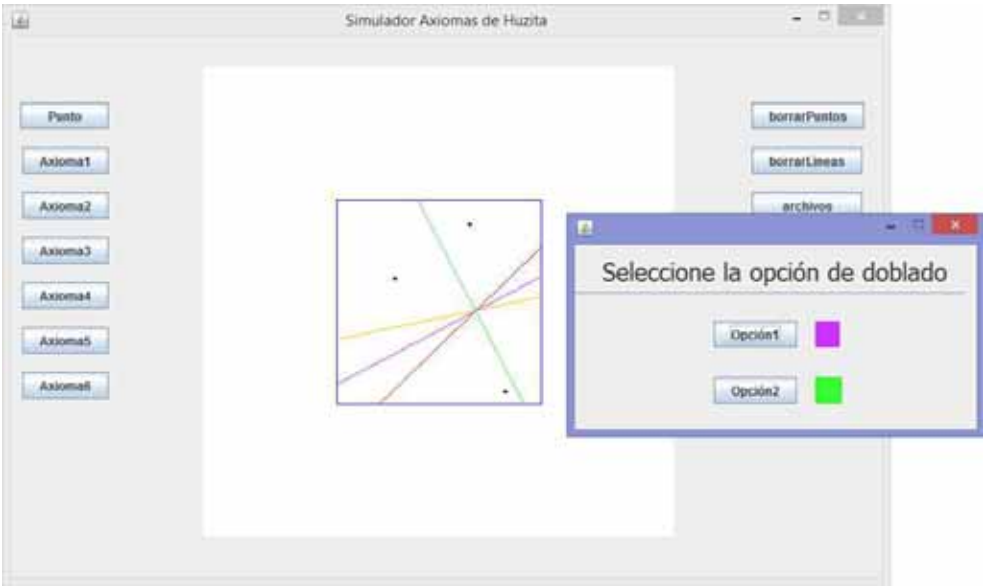
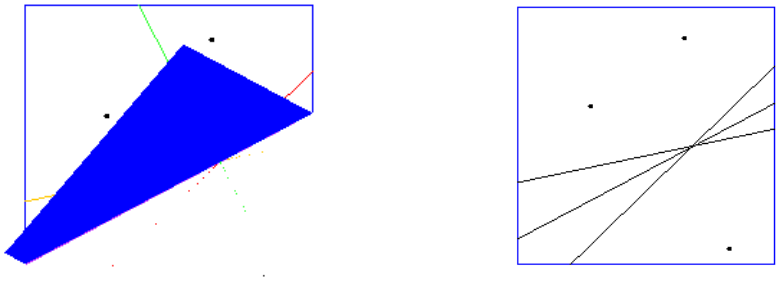


FIGURA 22. Se muestran los posibles dobleces, en caso de no ser líneas paralelas.

En la ventana "Seleccione la opción de doblado" se da a elegir cuál de las dos líneas resultantes se quiere. Cuando se da clic sobre cualquiera de las dos opciones, se hace la animación y se queda trazada la línea resultante de ese doblez, tal y como se ve en la figura 23.



(a) doblando. (b) línea trazada.

FIGURA 23. Doblez generados al aplicar el axioma.

El segundo caso se da cuando las líneas elegidas son paralelas o su intersección esta fuera de la hoja, tal y como se muestra en la siguiente figura 24.

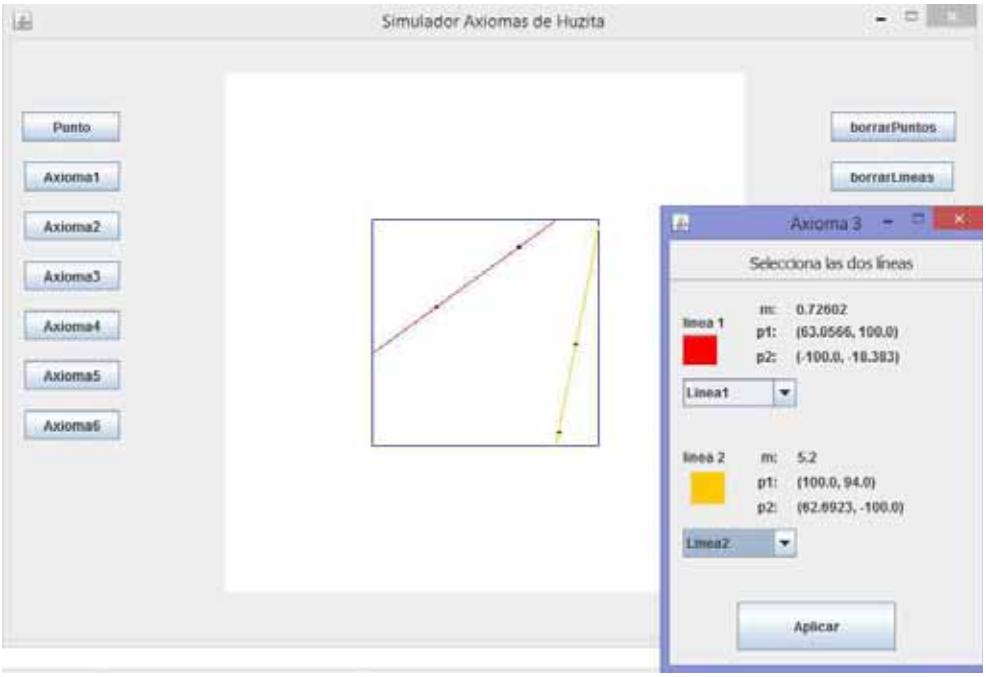
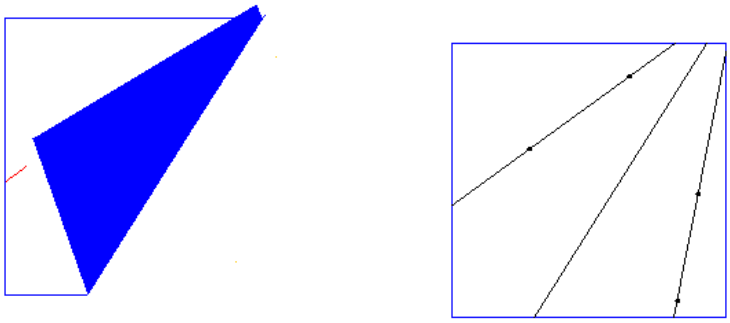


FIGURA 24. Se usan las listas desplegables para seleccionar las líneas.

Al aplicar el axioma, automáticamente se hace la animación del doblez y se traza la línea resultante.



(a) doblando (b) línea trazada

FIGURA 25. Doblez generado inmediatamente después de aplicar el axioma.

4.6. Ventana Axioma 4

Para poder aplicar este axioma es necesario que previamente haya al menos un punto y una línea en la hoja de papel, si ya se cuenta con eso y se presiona el botón “Axioma4” se despliega lo mostrado en la figura 26.

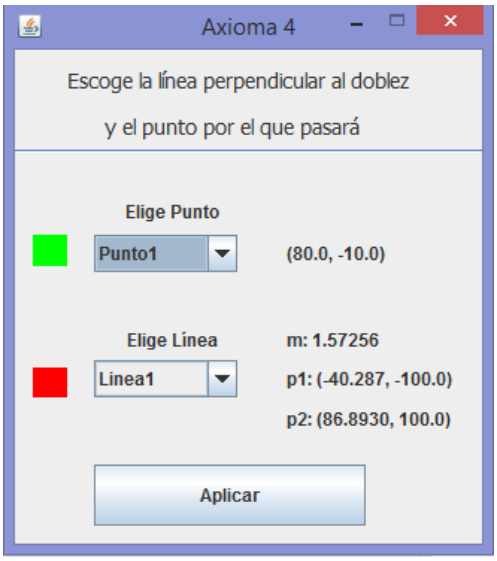


FIGURA 26. Ventana Axioma 4, seleccionar un punto y una línea.

Aquí se usan las listas desplegables para elegir el punto y la línea, cada que se selecciona un punto se colorea y se muestran sus coordenadas, de igual manera, al seleccionar una línea esta se pinta y se visualiza su pendiente y dos puntos pertenecientes a la recta.

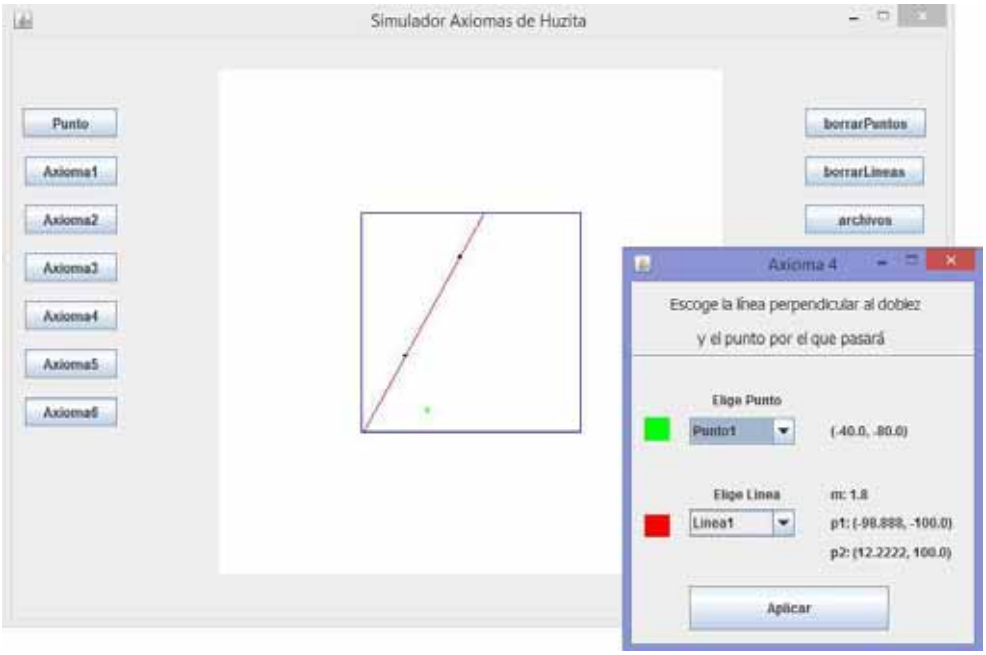
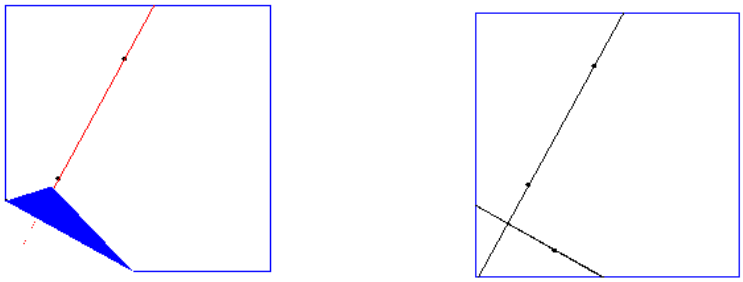


FIGURA 27. Se marcan de color las selecciones realizadas con las listas desplegables.

Después de la selección y de dar clic en “Aplicar”, se hace la animación y queda la línea del doblez ya trazada.



(a) doblando. (b) línea trazada.

FIGURA 28. Doblez generado después de aplicar el axioma.

4.7. Ventana Axioma 5

Cuando se presiona “Axioma5‘”, en la ventana principal, se muestra el cuadro de diálogo mostrado en la figura 29.

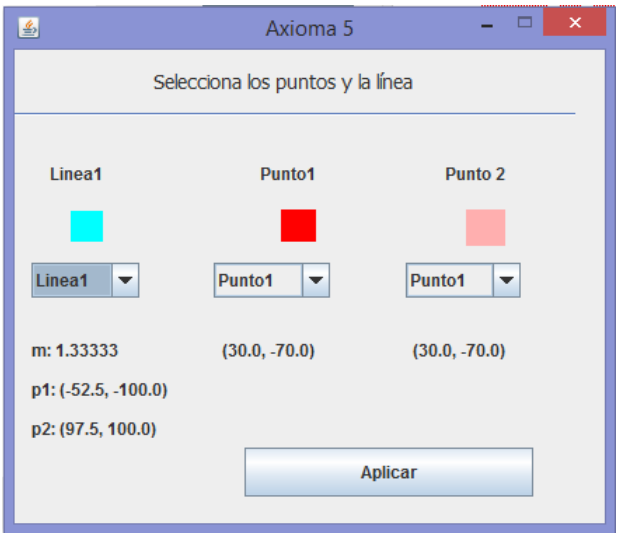


FIGURA 29. Ventana Axioma 5, seleccionar dos puntos y una línea.

El axioma 5 requiere que estén dibujados dos puntos y una línea como se observa en la figura 30.

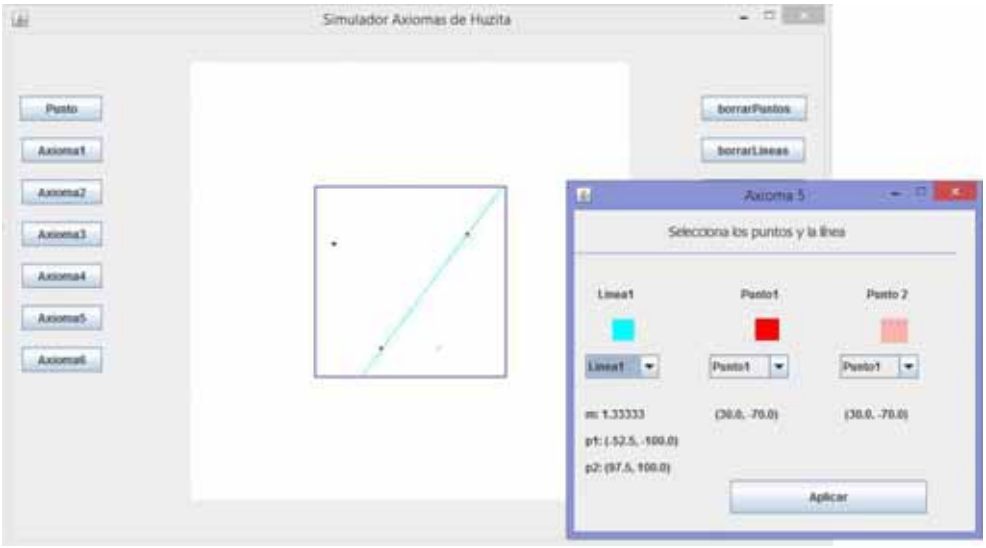


FIGURA 30. Se marcan de color los puntos y líneas seleccionados en las listas desplegables.

Se eligen los elementos con los selecccionadores de tal manera que el punto1 sea el que se va a colocar sobre la línea y el punto 2 es aquel por donde pasará el doblez, se puede apreciar la selección en el panel ya que los puntos y las líneas cambian de color. Una vez que se hayan elegido, se presiona el botón de "Aplicar" para ejecutar el axioma. Pueden existir cero, uno o dos soluciones, en caso de que sea cero, se informará al usuario que no hay doblez. Si hay una ésta se efectuará inmediatamente con su animación correspondiente tal como en los axiomas anteriores. Por último, cuando hay dos posibles dobleces se le desplegará al usuario la ventana que le dará a elegir cuál de los dos se ejecutará, marcándolas en el panel con un color distinto como se muestra en la figura 31.

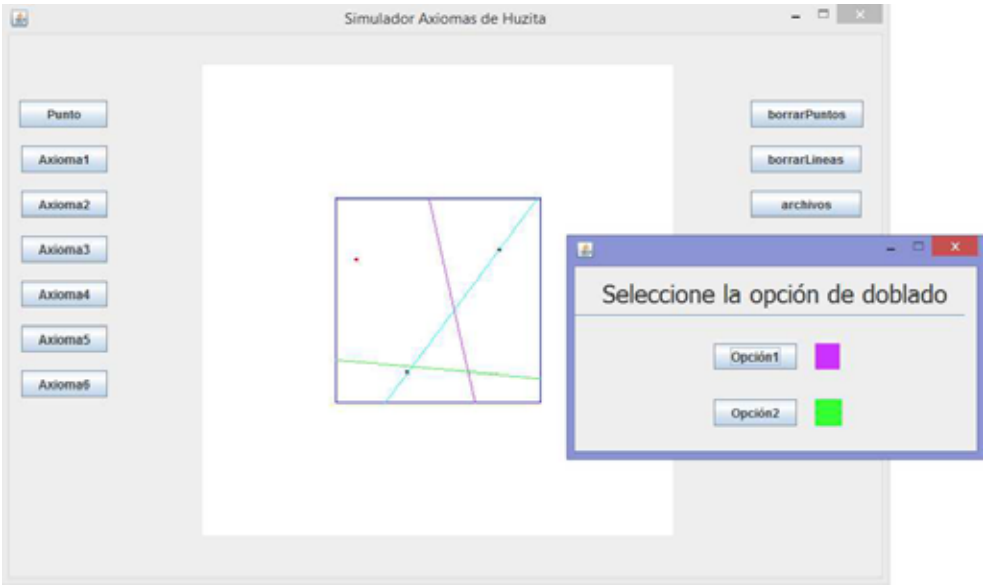
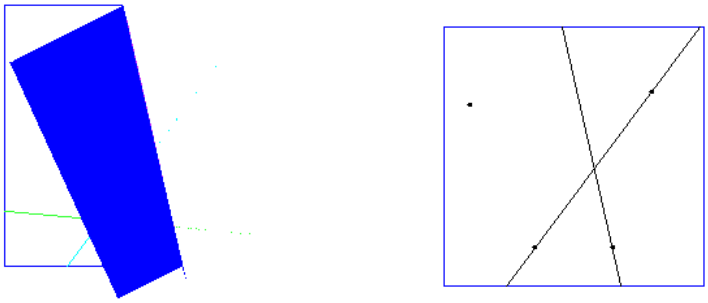


FIGURA 31. Se pueden tener distintas opciones de doblado y seleccionar una.

Al seleccionar la opción, se realiza la animación correspondiente al doblez como se muestra en la figura 32.



(a) doblando. (b) línea trazada.

FIGURA 32. Doblez generado al aplicar el axioma.

4.8. Ventana Axioma 6

Para desplegar la ventana Axioma 6 se presiona el botón con su nombre, ver figura 33.

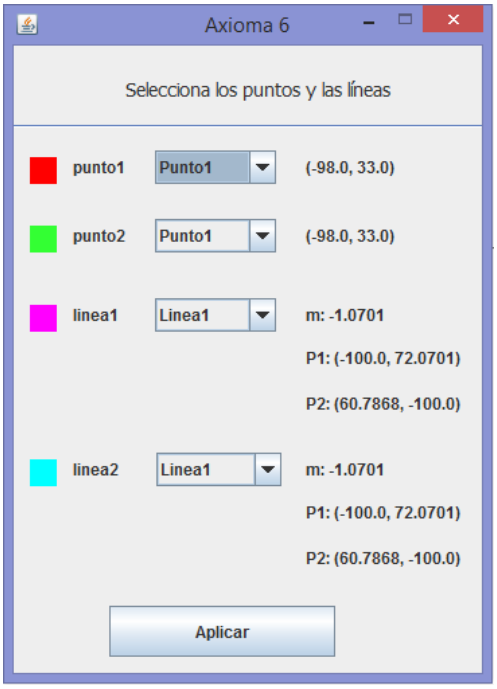


FIGURA 33. Ventana Axioma 6, se debe seleccionar dos puntos y dos líneas.

Es necesario contar con al menos dos líneas y dos puntos para ejecutar al axioma. La línea 1 y el punto 1 generan la primera parábola y de forma análoga la línea 2 y punto 2 generan la otra parábola, para poder establecerlas de esta manera se usan los seleccionadores, los cuales son marcados de colores distintos en el panel, tal y como se muestra en la figura 34.

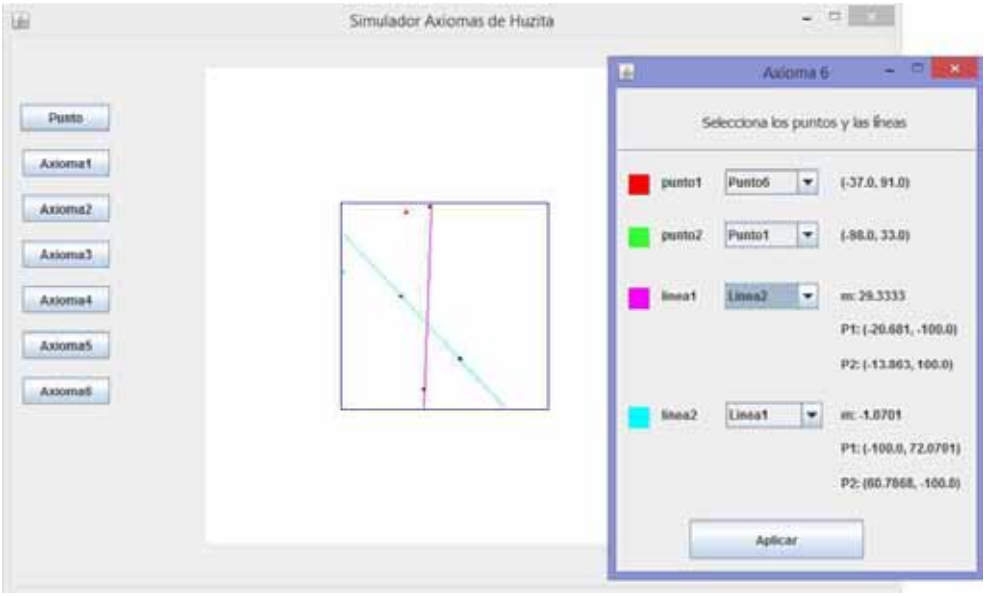


FIGURA 34. Muestra de color los puntos y líneas seleccionados.

Una vez seleccionados, damos click en "Aplicar" lo que despliega una ventana de selección si es que hay más de una solución, como se aprecia en la figura 35. Recordemos que al igual que en "Ventana Axioma 5" puede haber varias soluciones y por lo tanto distintos dobleces.

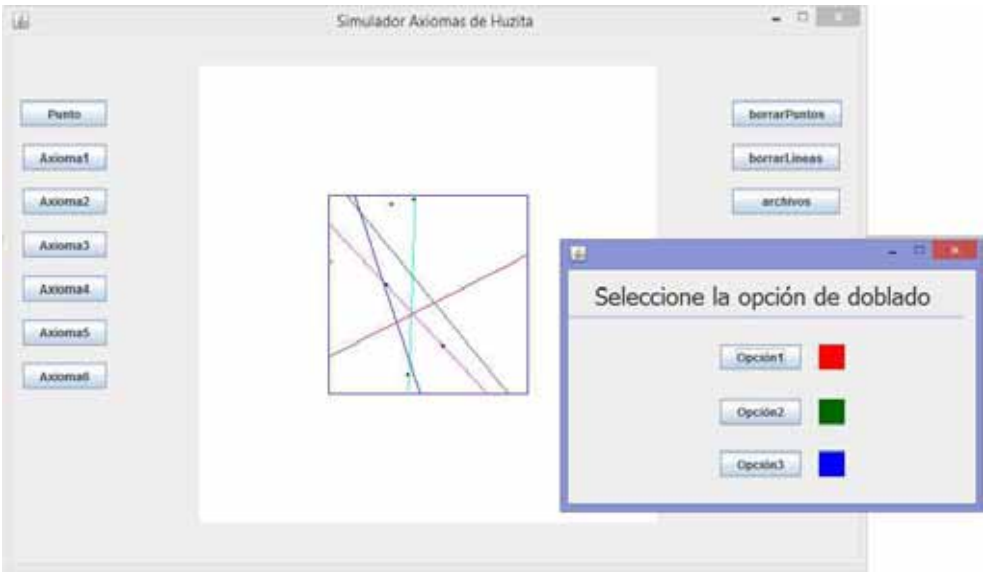
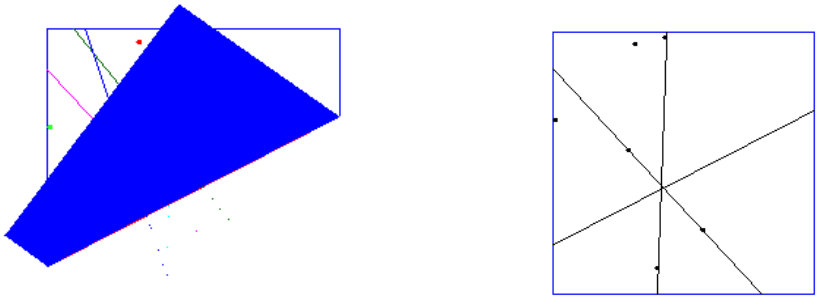


FIGURA 35. Dependiendo de las soluciones encontradas, muestra los posibles dobleces.

Cada uno de los posibles dobleces se muestra de colores diferentes en el panel, de esta manera el usuario puede seleccionar alguno de estos haciendo click en la opción deseada, así se realiza la animación correspondiente al doblez elegido, como se aprecia en la figura 36.



(a) doblando. (b) línea trazada.

FIGURA 36. Doblez generado al seleccionar una de las opciones.

4.9. Ventana archivos

Al presionar el botón "Archivos" se despliega una ventana con la cual se puede interactuar para guardar o recuperar información que el programa pueda usar, véase figura 37.

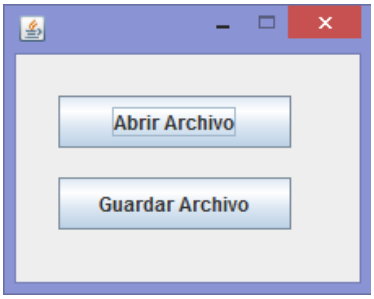
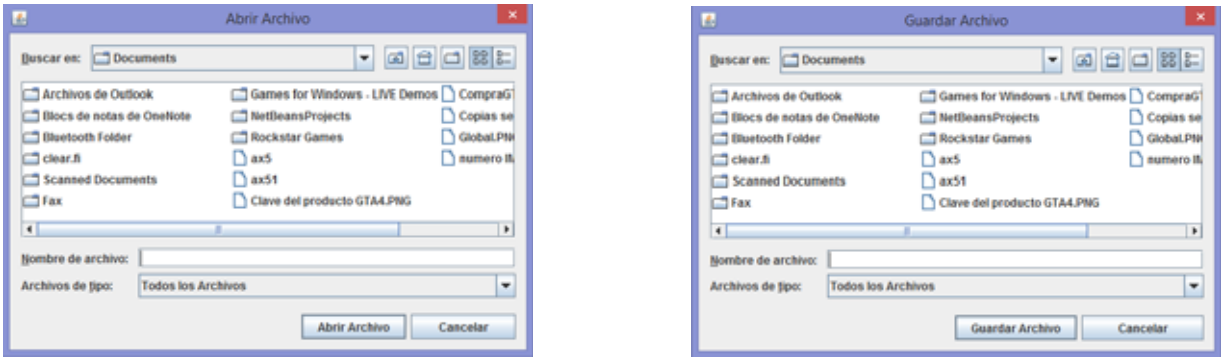


FIGURA 37. Ventana Archivos, puede cargar y guardar los puntos y líneas.

La figura 38 muestra los cuadros de diálogo correspondientes a cada una de las dos acciones.



(a) abrir. (b) guardar.

FIGURA 38. Ventanas para desplazarse por distintos folders en la computadora.

4.10. Análisis

En esta sección se dan a conocer los problemas que surgieron al momento de obtener los resultados expuestos anteriormente. Al plantear el desarrollo, se estableció la realización de una interfaz gráfica con las clases que definirían el cuerpo del programa y que fueron establecidas mediante las matemáticas descritas en el marco teórico.

Uno de los primeros problemas que se presentaron fue en la ventana principal, mas en concreto sobre el panel, esto debido a que dicha clase hereda métodos de *jPanel* de la cual no se tenía conocimiento de acuerdo a su funcionamiento, así que se realizo una investigación de sus métodos, en especial los que trazan los objetos(rectas, polígonos, círculos, etc.) y que serían necesarios para hacer la simulación de la hoja y su doblado. En esta misma clase se tuvo el problema de que cada vez que la ventana principal era modificada, estando el programa en ejecución, no se conservaban los objetos que se habían trazado en el panel, estos ya no eran visibles. La solución a esto fue hacer un método, refresh, que dibujara de nuevo todos los objetos almacenados en la lista de líneas y en la lista de puntos, no importando la acción a realizar sobre la ventana principal.

Cada doblez es representado por una línea recta, para simular el doblado nos interesa encontrar la intersección de ésta con los bordes de la hoja de papel, esta tarea fue complicada debido a que hay muchos casos posibles. Para restringir la línea en la hoja, se diseño e implementó un algoritmo el cual es capaz de decidir el lado de la hoja donde se debe de dejar de dibujar la línea.

Consideramos que el mayor problema con el que nos encontramos fue la manera en que se realizaría la simulación del doblado de la hoja. Esto debido a que la hoja doblada es dividida en dos polígonos, para resolver esto fue necesario el uso de la geometría. A partir de esto se hace la animación desde el estado inicial del papel borrando y dibujando polígonos hasta llegar a la configuración final que simula la hoja doblada. Este proceso se describe en la sección de “Desarrollo”, en la clase doblado.

A lo largo de todo el desarrollo del proyecto hubo una gran cantidad de dificultades relacionadas con el aspecto matemático del problema, como las soluciones por distintos métodos de ecuaciones de segundo y tercer grado. A su vez fue complicado la descripción geométrica del doblez para aspectos tales como la programación de los axiomas y su simulación animada, para esto fue necesario repasar conceptos como ángulos, pendientes, líneas, parábolas, tangentes, etc.

Capítulo 5

Conclusiones

El desarrollo de este proyecto nos dejó entrever la importancia de las matemáticas en las aplicaciones de un sistema funcional. La geometría es la base de este trabajo, nosotros como estudiantes de ingeniería en computación, no usamos sus conceptos de forma regular, ya que nos enfocamos en otro tipo de matemáticas, como las computables. Gracias a la realización de este proyecto pudimos enriquecer este tipo de conocimientos y ver como podemos abstraer problemas reales (como doblar una hoja de papel) y hacer sistemas computables que den soluciones a estos.

A su vez, el proyecto ayudo en reafirmar el conocimiento en programación, por ejemplo, uno de los mayores retos fue simular una hoja de papel y la manera en que se dobla, a esto nos referimos con realizar la adecuada simulación mediante una animación, para esto tuvimos que complementar el conocimiento ya adquirido en materias tales como graficación para enfocarlo en este programa, en particular, al momento de dar la ilusión óptica adecuada para la simulación del doblado de la hoja.

Otro aspecto a considerar, fue el desarrollo del programa en sí, ya que tuvimos que hacer clases que al integrarse dieran como resultado un sistema efectivo, que cumpliera la necesidad del usuario al ejecutar los axiomas, estas clases incluyeron, como ya se mencionó, interfaces gráficas, las cuales se crearon de tal forma que fueran amigables con el usuario. Para la integración de clases e interfaces del sistema, se usaron metodologías en materias aprendidas a lo largo de la carrera como análisis y diseo de sistemas de información. Por otro lado, Estructura de Datos con Orientación a Objetos y Almacenamiento y Recuperación de la Información, nos aportaron las bases necesarias para saber como guardar objetos introducidos por los usuarios y manejarlos en forma de archivos.

Podemos concluir que la realización de este proyecto fue un reto en varios aspectos, tanto de programación como matemáticos. De lo que se tiene certeza, es que se reafirmaron conocimientos aprendidos a lo largo de la carrera y así mismo se adquirieron otros. Cabe mencionar que reconocemos la importancia del uso de otro tipo de matemáticas que no son utilizadas durante la licenciatura, tales como la geometría.

Apéndice A

Código

En las siguientes secciones se encuentra el código referente al programa realizado, éste se encuentra divido conforme a las clases realizadas:

A.1. Main

```
package sahco;
public class SAHCO {

    public static void main(String[] args) {
        ventanaPrincipal v = new ventanaPrincipal();
    }
}
```

A.2. CpuntoDoble

```
package sahco;
import java.lang.Math;

public class CpuntoDoble{ //mtodo identico a cPunto solo que usa valores dobles para mejorar precision
    private double x; //variable de clase que simula la abscisa del punto
    private double y; //ordenada del punto
    private CpuntoDoble siguiente; //variable de clase ordena los cPunto en la lista de Puntos (listaPuntos)
    private CpuntoDoble anterior; //es analoga a siguiente, establece que punto es anterior a este

CpuntoDoble(double X, double Y) { //metodo constructor de, asigna con los valores dobles X y Y a las
     variables de clase
    x=X;
    y=Y;
    siguiente=null;
    anterior=null;
}

public void setX(double X){ //metodo que sirve para asignarle un valor a x
    x=X;
}

public double getX(){ //metodo que regresa el valor de x
    return x;
}

public void setY(double Y){ //metodo que sirve pra asignarle un valor a y
    y=Y;
}
```

```
public double getY(){ //metodo que regresa el valor de y
    return y;
}

public void setSiguiente(CpuntoDoble s){ //metodo que establece que punto va despues dentro de una
     listaPuntosDobles
    siguiente=s;
}

public CpuntoDoble getSiguiente(){ //metodo que te dice cual es el punto que va despues dentro de una
     listaPuntosDobles
    return siguiente;
}

public void setAnterior(CpuntoDoble a){//metodo que establece que punto va antes dentro de una
     listaPuntosDobles
    anterior=a;
}

public CpuntoDoble getAnterior(){ //metodo que te dice cual es el punto que va antes dentro de una
     listaPuntosDobles
    return anterior;
}

public CpuntoDoble rotar(double angulo){ //este metodo regresa el punto rotado
//se multiplica el vector[x y] por la matriz de rotacion[cos(angulo)-sen(angulo)][sen(angulo)cos(angulo)]
    double X=x*Math.cos(angulo)-y*Math.sin(angulo);//se define la nueva coordenada x
    double Y=x*Math.sin(angulo)+y*Math.cos(angulo);//se define la nueva coordenada y
    return new CpuntoDoble(X,Y);//se regresa el punto ya rotado, con las nuevas coordenadas
}

}
```

A.3. ListaPuntosDobles

```
package sahco;
public class ListaPuntosDobles {
    private CpuntoDoble inicio;
    private CpuntoDoble fin;
    private int tamao;
    ListaPuntosDobles(){
        inicio=null;
        fin=null;
         tamao =0;
    }
    public void insertarAlInicio(CpuntoDoble p){
        if( tamao ==0){
            inicio=p;
            fin=p;
             tamao =1;
        }
        else{
            p.setSiguiente(inicio);
            inicio.setAnterior(p);
            inicio=p;
             tamao ++;
```

```
        }
    }

    public int getTamao(){
        return tamao;
    }

    public CpuntoDoble[] regresaLista(){
       CpuntoDoble aux=inicio;
       int i=0;
       CpuntoDoble M[];
       M= new CpuntoDoble[tamao];
       if( tamao ==0)
          return M;
        if(aux==fin){
            M[i]=aux;
        }
        else{
        while(aux!=fin){
            M[i]=aux;
            aux=aux.getSiguiente();
            i++;
        }
        M[i]=aux;
        }
return M;
    }

    public boolean existePunto(double x,double y){

        CpuntoDoble aux=inicio;
        if(aux==null)
            return false;
        if(aux==fin){
            if(aux.getX()==x && aux.getY()==y)
                return true;
            }
        while(aux!=fin){
            if(aux.getX()==x && aux.getY()==y){
                return true;
            }
            else
                aux=aux.getSiguiente();
        }
        if(aux==fin){
            if(aux.getX()==x && aux.getY()==y)
                return true;
            }
        return false;
    }
}
```

A.4. cLinea

```
package sahco;
public class cLinea {
```

```
    private cLinea siguiente;
    private cLinea anterior;
    private CpuntoDoble extremo1;
    private CpuntoDoble extremo2;

    cLinea(CpuntoDoble ex1,CpuntoDoble ex2){

        siguiente=null;
        anterior=null;

        extremo1=ex1;
        extremo2=ex2;
    }

public void setSiguiente(cLinea s){
    siguiente=s;
}

public void setAnterior(cLinea a){
    anterior=a;
}

public cLinea getSiguiente(){
    return siguiente;
}

public cLinea getAnterior(){
    return anterior;
}

public void setP1(CpuntoDoble p){
    extremo1=p;
}

public void setP2(CpuntoDoble p){
    extremo2=p;
}

public CpuntoDoble getP1(){
    return extremo1;
}

public CpuntoDoble getP2(){
    return extremo2;
}

public double ordenadaAlorigen(){
    return (double)(((double)(-this.regresaPendiente().getValor())*(extremo1.getX()))+extremo1.getY());
}

public cPendiente regresaPendiente(){
    if(extremo1.getX()==extremo2.getX()){
        return new cPendiente(true,0.0);
    }
    return new cPendiente(false,(extremo2.getY()-extremo1.getY())/(extremo2.getX()-extremo1.getX()));
    //return new cPendiente(false,(double)((double)(deltaY)/(double)(deltaX)),deltaX,deltaY);
}

public boolean esIgualAlaPendienteDe(cLinea L2){
```

```
    String pendiente1=String.valueOf(this.regresaPendiente().getValor());
    String pendiente2=String.valueOf(L2.regresaPendiente().getValor());
    int i=0;
    int precision=5;
    if(pendiente1.length()<5||pendiente2.length()<5){
        if(pendiente1.length()<=pendiente2.length()){
            precision=pendiente1.length();
        }
        else{
            precision=pendiente2.length();
        }
    }

    while(i<precision){
        if(pendiente1.charAt(i)!=pendiente2.charAt(i))
            break;
        i++;
    }
    if(i!=precision)
        return false;
    else
        return true;

}
}
```

A.5. cLineaPerpendicularDados2Puntos

```
package sahco;
import java.awt.Color;
import java.lang.Math;

public class cLineaPerpendicularDados2Puntos {
    private listaPuntos lp; //variable de clase que guardara la lista de puntos
    private panelPrincipal h; //variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll; //variable de clase que guardara la lista de lineas
    private cPunto punto1Paralelo;
    private cPunto punto2Paralelo;
    cLineaPerpendicularDados2Puntos(cPunto punto1, cPunto punto2,listaPuntos LP, listaLineas
        LL,panelPrincipal H){
    //esta clase contendra los puntos que definiran la linea perpendicular que se genera con las variables
        de entrada
    lp=LP;
    ll=LL;
    h=H;

    int x1=punto1.getX();
    int y1=punto1.getY();
    int x2=punto2.getX();
    int y2=punto2.getY();

    if(x1==x2||y1==y2){
        if(lp.existePunto(x1, y1) && lp.existePunto(x2, y2)){
            if(x1==x2){
                CpuntoDoble PM=this.puntoMedio(x1, y1, x2, y2);
                punto1Paralelo=new cPunto((int)PM.getX(),(int)PM.getY());
                punto2Paralelo=new cPunto(100,(int)PM.getY());
```

```
            }
          else{
            CpuntoDoble PM=this.puntoMedio(x1, y1, x2, y2);
            punto1Paralelo=new cPunto((int)PM.getX(),(int)PM.getY());
            punto2Paralelo=new cPunto( (int)PM.getX(),100);
            }
          }
        else{
            ventanaError v=new ventanaError("Al menos uno de estos puntos no ha sido introducido","Prueba
                introduciendo el punto");
            v.setVisible(true);
          }
       }
    else{
        CpuntoDoble PM=this.puntoMedio(x1, y1, x2, y2);
        double pendientePerpendicular=this.pendientePerpendicular(x1, y1, x2, y2);
        punto1Paralelo=new cPunto((int)PM.getX(),(int)PM.getY());
        punto2Paralelo=new cPunto( 100,(int)this.yPuntoPendienteParalela(PM.getX(), PM.getY(),
            pendientePerpendicular,100));
        }
    }

    public cPunto getParalelo1(){
        return punto1Paralelo;
    }

    public cPunto getParalelo2(){
        return punto2Paralelo;
    }

    public String x1(){
        return Integer.toString(punto1Paralelo.getX());
    }
     public String y1(){
        return Integer.toString(punto1Paralelo.getY());
    }

    public String x2(){
        return Integer.toString(punto2Paralelo.getX());
    }
     public String y2(){
        return Integer.toString(punto2Paralelo.getY());
    }

    public CpuntoDoble puntoMedio(int x1,int y1, int x2, int y2){
        return new CpuntoDoble((x1+x2)/2,(y1+y2)/2);
    }
    public double xPuntoPendienteParalela(double x1,double y1 ,double PP,double Y){
        return (x1+((Y-y1)/PP));
    }
    public double yPuntoPendienteParalela(double x1,double y1 ,double PP,double X){
        return (y1+((X-x1)*PP));
    }
    public double pendientePerpendicular(double x1,double y1, double x2, double y2){
        return -((x2-x1)/(y2-y1));
    }
}
```

A.6. cParabola

```
package sahco;
import java.lang.Math;
public class cParabola {
    private CpuntoDoble foco;
    private cLinea directriz;
    private double anguloRotacion;

    cParabola(CpuntoDoble f, cLinea d, double a){
        foco=f;
        directriz=d;
        anguloRotacion=a;
    }

    public CpuntoDoble getFoco(){
        return foco;
    }

    public cLinea getDirectriz(){
        return directriz;
    }

    public double getAnguloR(){
        return anguloRotacion;
    }

    public cParabola rotarParabola(double angulo){
        CpuntoDoble F=foco.rotar(angulo);
        CpuntoDoble P1=directriz.getP1().rotar(angulo);
        CpuntoDoble P2=directriz.getP2().rotar(angulo);
        cLinea D=new cLinea(P1,P2);
        return new cParabola(F,D,anguloRotacion+angulo);

    }

    public boolean esParabola(){
        if(Math.abs((directriz.regresaPendiente().getValor()*(foco.getX()-directriz.getP1().getX()))-
        (foco.getY()+directriz.getP1().getY()))<0.01)
            return false;
        else
            return true;
    }

    public ecuacionParabola obtenerEcuacion(){
        if(this.esParabola()){
            double P=1.0/(2.0*(foco.getY()-directriz.getP1().getY()));
            double Q=(-2.0)*(foco.getX())/(2.0*(foco.getY()-directriz.getP1().getY()));
            double R=((Math.pow(foco.getX(),2.0))+(Math.pow(foco.getY(),
                 2.0))-(Math.pow(directriz.getP1().getY(),2.0)))/(2.0*(foco.getY()-directriz.getP1().getY()));
            return new ecuacionParabola(P,Q,R);
        }
        else
            return null;
    }
}
```

A.7. cPendiente

```
package sahco;
public class cPendiente {
    private boolean esVertical;
    private double valor;

    cPendiente(boolean b,double v){
        esVertical=b;
        valor=v;
    }

    public double getValor(){
        return valor;
    }

    public boolean getEsVertical(){
        return esVertical;
    }
}
```

A.8. cPunto

```
package sahco; //se dedine que sera parte del programa sahco
public class cPunto { //esta clase abstrae el concepto de un punto en el plano cartesiano
    private int x; //variable de clase que simula la abscisa del punto
    private int y; //ordenada del punto
    private cPunto siguiente; //esta variable de clase sirve para ordenar los cPunto en la lista de Puntos
         (listaPuntos)
    private cPunto anterior; //es analoga a siguiente, establece que punto es anterior a este

cPunto(int X, int Y) { //metodo constructor de la clase cPunto, asigna con los enteros X y Y a las
     variables de clase
    x=X;
    y=Y;
    siguiente=null;
    anterior=null;
}

public void setX(int X){ //metodo que sirve para asignarle un valor a x
    x=X;
}

public int getX(){ //metodo que regresa el valor de x
    return x;
}

public void setY(int Y){ //metodo que sirve pra asignarle un valor a y
    y=Y;
}

public int getY(){ //metodo que regresa el valor de y
    return y;
}
```

```
public void setSiguiente(cPunto s){ //metodo que establece que punto va despues dentro de una listaPuntos
    siguiente=s;
}

public cPunto getSiguiente(){ //metodo que te dice cual es el punto que va despues dentro de una listaPuntos
    return siguiente;
}

public void setAnterior(cPunto a){ //metodo que establece que punto va antes dentro de una listaPuntos
    anterior=a;
}

public cPunto getAnterior(){ //metodo que te dice cual es el punto que va antes dentro de una listaPuntos
    return anterior;
}
}
```

A.9. claseCardano

```
package sahco;
import java.lang.Math;

public class claseCardano {
    private double A;
    private double B;
    private double C;
    private double D;
    private double discriminante;
    private double x1;
    private double x2;
    private double x0;

    claseCardano(double w3,double w2,double w1,double w0){
        A=w3;
        B=w2;
        C=w1;
        D=w0;
    }

    public void resuelve(){
        double a=B/A;
        double b=C/A;
        double c=D/A;
        double z0,z1,z2;
        double p=b-(a*a/3.0);
        double q=((2.0*a*a*a)/27.0)-(a*b/3.0)+c;
        discriminante=(q*q)+((4.0/27.0)*p*p*p);

        System.out.println("Cardano p= "+p);
        System.out.println("Cardano q= "+q);

        if(discriminante==0.0){
            z0=3.0*q/p;
            z1=-3.0*q/(2.0*p);
            z2=z1;
            x0=z0-(a/3.0);
            x1=z1-(a/3.0);
```

```
            x2=z2-(a/3.0);
        }
        else{
            if(discriminante>0.0){
                //double u=Math.pow(((-q+Math.sqrt(discriminante))/(2.0)),(1.0/3.0));
                //double v=Math.pow((-q-Math.sqrt(discriminante))/2.0,1.0/3.0);
                double u=Math.cbrt((-q+Math.sqrt(discriminante))/(2.0));
                double v=Math.cbrt((-q-Math.sqrt(discriminante))/(2.0));
                System.out.println("discriminante:"+discriminante);
                System.out.println("u:"+u);
                System.out.println("v:"+v);
                z0=u+v;
                x0=z0-(a/3.0);
            }
            else{
                z0=formaTrigonometrica(0.0,p,q);
                z1=formaTrigonometrica(1.0,p,q);
                z2=formaTrigonometrica(2.0,p,q);
                x0=z0-(a/3.0);
                x1=z1-(a/3.0);
                x2=z2-(a/3.0);
            }
        }
    }

    public double getDiscriminante(){
        return discriminante;
    }

    public double getX0(){
        return x0;
    }

    public double getX1(){
        return x1;
    }
    public double getX2(){
        return x2;
    }

    private double formaTrigonometrica (double k,double p,double q){
        return 2*Math.sqrt(-p/3.0)*Math.cos(((1.0/3.0)*Math.acos((-q/2.0)*Math.sqrt(27.0/(-p*p*p))))+
        ((Math.PI*k*2.0)/3.0));
    }
}
```

A.10. doblado

```
package sahco;
import java.awt.Color;
import java.lang.Math;

public class doblado { //esta clase contiene los componentes necesarios para realizar el doblado de la hoja
     de papel
    private listaPuntos lp//variable de clase que guarda la lista de puntos
    private panelPrincipal h;//variable de clase que guarda el panel donde se dibujo
    private listaLineas ll;//variable que contendra la lista de lineas
```

```
doblado(listaPuntos LP, listaLineas LL,panelPrincipal H){//metodo constructor del doblado
    lp=LP; //se asigna la lista de puntos
    ll=LL; //se asigna la lista de lineas
    h=H; //se asigna el panel
}

public void hacerDoblado(String tx1,String ty1,String tx2,String ty2,boolean puntoTemporal){//este
     metodo es el
que hace el doblez usando los componentes que se pasaron en el constructor
 //la variable punto temporal es true cuando el doblado se invoca desde cualquier axioma menos el axioma
      1
       int x1=Integer.parseInt(tx1);//se convierte el texto, que introduce el usuario, a entero y se
            guarda la
        x del primer punto
       int y1=Integer.parseInt(ty1);//se convierte el texto a entero y se guarda la y del primer punto
       int x2=Integer.parseInt(tx2);//se convierte el texto a entero y se guarda la x del segundo punto
       int y2=Integer.parseInt(ty2);//se convierte el texto a entero y se guarda la y del segundo punto

       cPunto O1=new cPunto(x1,y1); //primer punto introducido por el usuario, usando los enteros de
            arriba
       cPunto O2=new cPunto(x2,y2);//segundo punto introducido por el usuario

       double dX1=(double)x1; //valor de la x del primer punto introducido por el usuario en tipo doble
       double dY1=(double)y1; //valor de la y del primer punto introducido por el usuario en tipo doble
       double dX2=(double)x2; //valor de la x del segundo punto introducido por el usuario en tipo doble
       double dY2=(double)y2; //valor de la y del segundo punto introducido por el usuario en tipo doble

if(x1==x2||y1==y2){
   if((lp.existePunto(x1, y1) && lp.existePunto(x2, y2))||puntoTemporal){
       if(x1==x2){
           cPunto P1=new cPunto(x1,-100); //primer punto extremo en la hoja de papel de la linea de
                doblez
           cPunto P2=new cPunto(x2,100); //segundo punto extremo en la hoja de papel de la linea de
                doblez
           CpuntoDoble pE1=new CpuntoDoble(dX1,-100.0); //primer punto extremo (doble) en la hoja de
                papel
           de la linea de doblez
           CpuntoDoble pE2=new CpuntoDoble(dX2,100.0); //segundo punto extremo (doble) en la hoja de
                papel
           de la linea de doblez
           cLinea nuevaLinea=new cLinea(pE1,pE2); //se crea la nueva linea en base a los dos puntos
                extremos
           ll.insertarAlInicio(nuevaLinea); //se inserta la nueva linea en la lista de lineas
           int r1=x1-(-100);//distancia de la linea de doblez al margen izquierdo
           int r2=200-r1;//distancia de la linea de doblez al margen derecho
           if(r1>=r2){//el doblez tiene que ser a la izquierda
               for(int i=100;i>x1-r2;i--){ //for que hace el doblez hacia la izquierda
                   if(i>x1){ //cuando el doblez hacia la izquierda va antes de la mitad
                       h.dibujaLineaDeColor(Color.blue, i, 100, i, -100, h.getGraphics());//linea que
                            simula
                        el recorrido de la hoja hacia la izquierda
                       try{
                       Thread.sleep(2); //10
                       }catch(InterruptedException e){}
                       h.dibujaLineaDeColor(Color.white, i, 100, i, -100, h.getGraphics());//se borra la
                            linea para simular la animacion
                   }
```

```
            else{//cuando el doblez hacia la izquierda esta en la mitad o mas alla de la mitad
                h.dibujaTrapecioColor(Color.blue,i, 100, i, -100, x1, 100, x1, -100,
                     h.getGraphics()); //cuadrado que simula la hoja doblandose
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}
            }
        }

        for(int i=x1-r2;i<100;i++){//for que simula el doblez hacia la derecha (de regreso)
            if(i<x1){//cuando el doblez de regreso esta antes de la mitad
                h.dibujaLineaDeColor(Color.blue, -100, 100, -100, -100, h.getGraphics());
                //redibuja MARGEN IZQUIERDO
                int X[]={x1-r2,i,i,x1-r2};
                int Y[]={100,100,-100,-100};
                h.dibujarPoligonoDeColor(Color.white, 4, X, Y, h.getGraphics());
                //cuadrado que simula la hoja doblandose
                h.dibujaLineaDeColor(Color.blue, -100, 100, -100, -100, h.getGraphics());
                 //redibuja mARGEN IZQUIERDO
                h.dibujaLineaDeColor(Color.blue,x1-r2 ,100, i, 100, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,x1-r2 ,-100, i, -100, h.getGraphics());
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}

                }
            else{//cuando el doblez de regreso esta en la mitad o ha rebasado la mitad
                h.dibujaLineaDeColor(Color.blue, i, 100, i, -100, h.getGraphics());
                //linea que simula el recorrido de la hoja hacia la derecha
                h.dibujaLineaDeColor(Color.blue, x1, 100, i,100, h.getGraphics());
                //linea que simula el recorrido de la hoja hacia la derecha
                h.dibujaLineaDeColor(Color.blue, x1, -100, i,-100, h.getGraphics());
                //linea que simula el recorrido de la hoja hacia la derecha
                h.dibujaLinea(x1, 100, x2, 100, h.getGraphics());
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}
                h.dibujaLineaDeColor(Color.white, i, 100, i, -100, h.getGraphics());
                 //se borra la linea para simular la animacion
            }
        }
        h.refresh(h.getGraphics());
        //esta linea redibuja la hoja de papel con el doblez ya marcado
    }
    else{//cuando ellado derecho de dobles es mas grande
       for(int i=-100;i<x1+r1;i++){ //for que simula el doblez hacia la derecha
            if(i<x1){ //cuando el doblez esta antes de la mitad
                h.dibujaLineaDeColor(Color.blue, i, 100, i, -100, h.getGraphics());
                //linea que simula el recorrido de la hoja hacia la derecha

                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}

                h.dibujaLineaDeColor(Color.white, i, 100, i, -100, h.getGraphics());
                 //se borra la linea para simular la animacion
                }
            else{ //cuando el doblez de regreso esta en la mitad o ha rebasado la mitad
```

```
                h.dibujaTrapecioColor(Color.blue,i, 100, i, -100, x1, -100, x1, 100,
                     h.getGraphics()); //cuadrado que simula la hoja doblandose
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}

            }
        }

       for(int i=x1+r1;i>-100;i--){ //for que hace el doblez hacia la izquierda
            if(i>x1){ //cuando el doblez hacia la izquierda va antes de la mitad
                h.dibujaLineaDeColor(Color.blue, 100, 100, 100, -100, h.getGraphics());
                //redibuja mARGEN derecho
                int X[]={x1+r1,i,i,x1+r1};
                int Y[]={100,100,-100,-100};
                h.dibujarPoligonoDeColor(Color.white, 4, X, Y, h.getGraphics());
                //cuadrado que simula la hoja doblandose
                h.dibujaLineaDeColor(Color.blue, 100, 100, 100, -100, h.getGraphics());
                //redibuja mARGEN IZQUIERDO
                h.dibujaLineaDeColor(Color.blue,x1+r1 ,100, i, 100, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,x1+r1 ,-100, i, -100, h.getGraphics());
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}
                h.dibujaLineaDeColor(Color.white, i, 100, i, -100, h.getGraphics());
                //se borra la linea para simular la animacion
            }

            else{ //cuando el doblez hacia la izquierda esta en la mitad o mas alla de la mitad
                 h.dibujaLineaDeColor(Color.blue, i, 100, i, -100, h.getGraphics());
                 //linea que simula el rocorrido de la hoja hacia la izquierda
                h.dibujaLineaDeColor(Color.blue, x1, 100, i,100, h.getGraphics());
                //linea que simula el rocorrido de la hoja hacia la izquierda
                h.dibujaLineaDeColor(Color.blue, x1, -100, i,-100, h.getGraphics());
                //linea que simula el rocorrido de la hoja hacia la izquierda
                h.dibujaLinea(x1, 100, x2, 100, h.getGraphics());
                try{
                Thread.sleep(2); //10
                }catch(InterruptedException e){}
                h.dibujaLineaDeColor(Color.white, i, 100, i, -100, h.getGraphics());
                //se borra la linea para simular la animacion
            }
        }
       h.refresh(h.getGraphics());//esta linea redibuja la hoja de papel con el doblez ya marcado

    }

    }
else{ //cuando la linea de doblez es horizontal
    cPunto P1=new cPunto(-100,y1);
    cPunto P2=new cPunto(100,y1);
    CpuntoDoble pE1=new CpuntoDoble(-100.0,dY1); //primer punto extremo (doble) en la hoja de
        papel de la linea de doblez
    CpuntoDoble pE2=new CpuntoDoble(100.0,dY1); //segundo punto extremo (doble) en la hoja de
        papel de la linea de doblez
    cLinea nuevaLinea=new cLinea(pE1,pE2);//se crea la nueva linea en base a los dos puntos
        extremos

    ll.insertarAlInicio(nuevaLinea); //se inserta la nueva linea en la lista de lineas
```

```
int r1=y1-(-100); //distancia de la linea de doblez al margen superior
int r2=200-r1; //distancia de la linea de doblez al margen inferior
if(r1>=r2){ //el doblez tiene que ser a hacia arriba
    for(int i=100;i>y1-r2;i--){//for que hace el doblez hacia arriba
        if(i>y1){ //cuando el doblez hacia arriba va antes de la mitad
            h.dibujaLineaDeColor(Color.blue, 100, i, -100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia arriba
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}
            h.dibujaLineaDeColor(Color.white, 100, i, -100, i, h.getGraphics());
             //se borra la linea para simular la animacion
        }

        else{ //cuando el doblez hacia arriba esta en la mitad o mas alla de la mitad
            h.dibujaTrapecioColor(Color.blue,100,i,-100, i, -100, y1, 100, y1,
                 h.getGraphics()); //cuadrado que simula la hoja doblandose
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}
        }
    }

    for(int i=y1-r2;i<100;i++){ //for que simula el doblez hacia la abajo(de regreso)
        if(i<y1){ //cuando el doblez de regreso esta antes de la mitad
            h.dibujaLineaDeColor(Color.blue, -100, -100, 100, -100, h.getGraphics());
            //redibuja mARGEN inferior
            int X[]={100,100,-100,-100};
            int Y[]={y1-r2,i,i,y1-r2};
            h.dibujarPoligonoDeColor(Color.white, 4, X, Y, h.getGraphics());
            //cuadrado que simula la hoja doblandose
            h.dibujaLineaDeColor(Color.blue, -100, -100, 100, -100, h.getGraphics());
             //redibuja mARGEN inferior
            h.dibujaLineaDeColor(Color.blue,100,y1-r2 ,100, i, h.getGraphics());
            h.dibujaLineaDeColor(Color.blue,-100,y1-r2, -100 ,i,h.getGraphics());
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}

            }
        else{//cuando el doblez de regreso esta en la mitad o ha rebasado la mitad
            h.dibujaLineaDeColor(Color.blue,100, i, -100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia la arriba
            h.dibujaLineaDeColor(Color.blue, 100,x1, 100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia la arriba
            h.dibujaLineaDeColor(Color.blue,-100, x1, -100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia la arriba
            h.dibujaLinea( 100, y1, -100,y1, h.getGraphics());
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}
            h.dibujaLineaDeColor(Color.white,100, i, -100, i, h.getGraphics());
             //se borra la linea para simular la animacion
        }
    }
    h.refresh(h.getGraphics()); //esta linea redibuja la hoja de papel con el doblez ya
         marcado
}
```

```
else{//cuando el lado de abajo del doblez es mas grande
   for(int i=-100;i<y1+r1;i++){ //for que simula el doblez hacia la arriba
        if(i<y1){//cuando el doblez esta antes de la mitad
            h.dibujaLineaDeColor(Color.blue, 100,i,-100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia arriba

            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}

            h.dibujaLineaDeColor(Color.white, 100, i, -100,i, h.getGraphics());
            //se borra la linea para simular la animacion
            }
        else{ //cuando el doblez de regreso esta en la mitad o ha rebasado la mitad
            h.dibujaTrapecioColor(Color.blue, 100, i, -100,i,-100 ,y1, 100, y1,
                 h.getGraphics()); //cuadrado que simula la hoja doblandose
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}

        }
    }

   for(int i=y1+r1;i>-100;i--){
      //for que hace el doblez hacia abajo
        if(i>y1){ //cuando el doblez hacia la izquierda va antes de la mitad

            int X[]={100,100,-100,-100};
            int Y[]={y1+r1,i,i,y1+r1};
            h.dibujarPoligonoDeColor(Color.white, 4, X, Y, h.getGraphics());
             //cuadrado que simula la hoja doblandose

            h.dibujaLineaDeColor(Color.blue,100,y1+r1 ,100 ,i, h.getGraphics());
            h.dibujaLineaDeColor(Color.blue,-100,y1+r1,-100, i, h.getGraphics());
            try{
            Thread.sleep(2);//10
            }catch(InterruptedException e){}
            h.dibujaLineaDeColor(Color.white,100, i, -100,i, h.getGraphics());
             //se borra la linea para simular la animacion
        }

        else{ //cuando el doblez hacia la izquierda esta en la mitad o mas alla de la mitad
             h.dibujaLineaDeColor(Color.blue,100, i, -100, i, h.getGraphics());
              //linea que simula el recorrido de la hoja hacia la abajo
            h.dibujaLineaDeColor(Color.blue,100, y1, 100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia la abajo
            h.dibujaLineaDeColor(Color.blue, -100,y1,-100, i, h.getGraphics());
            //linea que simula el recorrido de la hoja hacia la abajo
            h.dibujaLinea(x1, 100, x2, 100, h.getGraphics());
            try{
            Thread.sleep(2); //10
            }catch(InterruptedException e){}
            h.dibujaLineaDeColor(Color.white, 100, i, -100,i, h.getGraphics());
             //se borra la linea para simular la animacion
        }
    }
   h.refresh(h.getGraphics()); //esta linea redibuja la hoja de papel con el doblez ya marcado

}
```

```
            }
          }
        else{
            ventanaError v=new ventanaError("Al menos uno de estos puntos no ha sido introducido","Prueba
                introduciendo el punto");
            v.setVisible(true);
          }
       }
    else{
    int xY100=(((100-y1)*(x2-x1)/(y2-y1))+x1);//el valor de "x" cuando "y" vale 100 en la recta del doblez
    int yX100=(((100-x1)*(y2-y1)/(x2-x1))+y1);//el valor de "y" cuando "x" vale 100 en la recta del doblez
    int xYm100=(((-100-y1)*(x2-x1)/(y2-y1))+x1);//el valor de "x" cuando "y" vale -100 en la recta del
        doblez
    int yXm100=(((-100-x1)*(y2-y1)/(x2-x1))+y1);//el valor de "y" cuando "x" vale -100 en la recta del
        doblez
    double xY100D=((((double)100.0-dY1)*(dX2-dX1)/(dY2-dY1))+dX1);
    //el valor de "x" (en doble) cuando "y" vale 100 en la recta del doblez
    double yX100D=((((double)100.0-dX1)*(dY2-dY1)/(dX2-dX1))+dY1);
    //el valor de "y" (en doble) cuando "x" vale 100 en la recta del doblez
    double xYm100D=((((double)-100.0-dY1)*(dX2-dX1)/(dY2-dY1))+dX1);
    //el valor de "x" (en doble) cuando "y" vale -100 en la recta del doblez
    double yXm100D=((((double)-100.0-dX1)*(dY2-dY1)/(dX2-dX1))+dY1);
    //el valor de "y" (en doble) cuando "x" vale -100 en la recta del doblez
    int extremox1;
    int extremoy1;
    int extremox2;
    int extremoy2;

    if((lp.existePunto(x1, y1) && lp.existePunto(x2, y2))||puntoTemporal){
    //solo se entra a este if si existen los dos puntos
        if(xY100<-100){ //este es el caso cuando x<-100 si y=100
            cPunto p1=new cPunto(-100,yXm100);
            //este es el extremo izquierdo del doblez es x=-100 y y=f(-100) f es la funcion de la
                linea entre los puntos dibujados
            CpuntoDoble p1Doble=new CpuntoDoble(-100.0,yXm100D);
            //el mismo punto de encima solo que en valores tipo double
            extremox1=-100+h.getWidth()/2;
            extremoy1=h.getHeight()/2-yXm100;
            if(xYm100>100){
                cPunto p2=new cPunto(100,yX100);
                CpuntoDoble p2Doble=new CpuntoDoble(100.0,yX100D);
                extremox2=100+h.getWidth()/2;
                extremoy2=h.getHeight()/2-yX100;
                //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
                //se crea la nueva linea en base a los dos puntos extremos

                ll.insertarAlInicio(nuevaLinea); //se inserta la nueva linea en la lista de lineas
                double r1=100.0-p1Doble.getY();
                double r2=100.0-p2Doble.getY();
                double s1=200.0-r1;
                double s2=200.0-r2;
                if(r1+r2>s1+s2){
                    double tanTetha=(p1Doble.getY()-p2Doble.getY())/200.0;
                    double tetha=Math.toDegrees(Math.atan(tanTetha));
                    double alpha=90.0-tetha;
                    double betha=(2.0*alpha)-90.0;
                    double cosBetha=Math.cos(Math.toRadians(betha));
```

```
double x0=cosBetha*s1;
int p3x=(int)(p1Doble.getX()+x0);
double senBetha=Math.sin(Math.toRadians(betha));
double y0=senBetha*s1;
int p3y=(int)(p1Doble.getY()+y0);
double x01=cosBetha*s2;
int p4x=(int)(p2Doble.getX()+x01);
double y01=senBetha*s2;
int p4y=(int)(p2Doble.getY()+y01);

int a=p3y-(-100);
int b=p4y-(-100);

for (int Yi=-100;Yi<(-100)+a;Yi++){
    int P=((Yi+100)*100/a);
    int Y2=(P*b/100)-100;
    int Xi=(((Yi-(-100))*(p3x-(-100))/(p3y-(-100)))+(-100));
    int X2=(((Y2-(-100))*(p4x-(100))/(p4y-(-100)))+(100));

    if(Yi<(a/2)-100){

    h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
    h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());


    try{
    Thread.sleep(2); //10
    }catch(InterruptedException e){}

    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
    }
    else{
        if(Yi==(a/2)-100){
            try{
            Thread.sleep(2) ;//2
            }catch(InterruptedException e){}
        }
        else{
            int X[]={p1.getX(),Xi,X2,p2.getX()};
            int Y[]={p1.getY(),Yi,Y2,p2.getY()};
            h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());

            try{
            Thread.sleep(2); //3
            }catch(InterruptedException e){}
            int Xp[]={p2.getX(),X2,100};
            int Yp[]={p2.getY(),Y2,-100};
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());
        }
    }

}

for (int Yi=(-100)+a;Yi>-100;Yi--){
    int P=((Yi+100)*100/a);
    int Y2=(P*b/100)-100;
```

```
int Xi=(((Yi-(-100))*(p3x-(-100))/(p3y-(-100)))+(-100));
int X2=(((Y2-(-100))*(p4x-(100))/(p4y-(-100)))+(100));

if(Yi>(a/2)-100){

       int X[]={p1.getX(),Xi,X2,p4x,p3x};
       int Y[]={p1.getY(),Yi,Y2,p4y,p3y};

       int Xp[]={p1.getX(),p2.getX(),X2,Xi};
       int Yp[]={p1.getY(),p2.getY(),Y2,Yi};

       h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());
       if(Yi>99){
       h.dibujaLineaDeColor(Color.blue, -100,
            100,(((100-(Yi))*(p1.getX()-(Xi))/(p1.getY()-(Yi)))+(Xi)),
       100, h.getGraphics());
       h.dibujaLineaDeColor(Color.BLUE, 100, 100,(((100-(Yi))*(X2-(Xi))/
       (Y2-(Yi)))+(Xi)) , 100, h.getGraphics());
       }
       else{
           h.dibujaLineaDeColor(Color.blue, -100, 100, 100,100,
           h.getGraphics());
       }
       h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
       h.dibujaLineaDeColor(Color.blue, 100, 100,
            100,(((100-(Xi))*(Y2-(Yi))/(X2-(Xi)))+(Yi)),h.getGraphics());
       try{
       Thread.sleep(2);//2
       }catch(InterruptedException e){}
}

else{
    P=((Yi+100)*100/a);
     Y2=(P*b/100)-100;
     Xi=(((Yi-(-100))*(p3x-(-100))/(p3y-(-100)))+(-100));
     X2=(((Y2-(-100))*(p4x-(100))/(p4y-(-100)))+(100));
    if(Yi==(a/2)-100){
       h.getGraphics().drawLine(extremox1,extremoy1,extremox2,extremoy2);
       try{
       Thread.sleep(2); //2
       }catch(InterruptedException e){}
    }
    else{

       h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi,
        h.getGraphics());
       h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2,
        h.getGraphics());
       h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
       try{
       Thread.sleep(2); //2
       }catch(InterruptedException e){}

       h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi,
        h.getGraphics());
       h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),X2,Y2,
        h.getGraphics());
       h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
```

```
            }
        }
    }
    h.refresh(h.getGraphics());
    //esta linea redibuja la hoja de papel con el doblez ya marcado

    *//estas lineas dibujarian como queda el doblez sin animacion
    h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2,
    h.getHeight()/2-p3y);
    h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2,
    h.getHeight()/2-p4y);
    h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,
    p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
     *
     */
}
else{
    double tanTetha=(p1Doble.getY()-p2Doble.getY())/200.0;
//tanTehta=catetoOpuesto(distancia entre puntos extremos)/catetoAdyacente(ancho de hoja)
    double tetha=Math.toDegrees(Math.atan(tanTetha));
    //tetha=arctan(tanTheta) en grados
    double alpha=90.0-tetha; //mitad del angulo de doblado en funcion de la vertical
    double betha=(2.0*alpha)-90.0; //angulo de doblado en funcion de la horizontal
    double cosBetha=Math.cos(Math.toRadians(betha));
    //coseno de betha en grados
    double x0=cosBetha*r1; //catetoAdyacente(x0)=cosenoAngulo(cosBetha)*Hipotenusa(r1)
    int p3x=(int)(p1Doble.getX()-x0); //x del primer punto donde quedara el doblez
    double senBetha=Math.sin(Math.toRadians(betha));
    //seno de betha en grados
    double y0=senBetha*r1; //catetoOpuesto(y0)=senoAngulo(senBetha)*Hipotenusa(r1)
    int p3y=(int)(p1Doble.getY()-y0); //y del primer punto donde quedara el doblez
    double x01=cosBetha*r2; //catetoAdyacente(x01)=cosenoAngulo(cosBetha)*Hipotenusa(r2)
    int p4x=(int)(p2Doble.getX()-x01); //x del segundo punto donde quedara el doblez
    double y01=senBetha*r2; //catetoOpuesto(y01)=senoAngulo(senBetha)*Hipotenusa(r2)
    int p4y=(int)(p2Doble.getY()-y01); //y del segundo punto donde quedara el doblez

    int b=100-p3y;
    //distancia (en y) entre el punto de doblez(p3) con el punto que se dobla(-100,100)
    int a=100-p4y;
    //distancia (en y) entre el punto de doblez(p4) con el punto que se dobla (100,100)

    for (int Yi=100;Yi>p4y;Yi--){
    //for para hacer la animacion de doblado hacia abajo Yi se
    va a mover el doblez de Yi=100 a p4y
        int P=((100-Yi)*100/a);
        //porcentaje del recorrido de Yi ( regla de tres si a=>100% entonces Yi=>? )
        int Y2=100-(P*b/100); //valor del recorrido de Y2 equivalente al
         porcentaje que ha recorrodo Yi( si 100%=>b entonces P=>?)
        int X2=(((Y2-(100))*(p3x-(-100))/(p3y-(100)))+(-100));
        //obteniendo el valor de x sobre la recta entre los puntos p3 y -100,100
        int Xi=(((Yi-(100))*(p4x-(100))/(p4y-(100)))+(100));
        //obteniendo el valor de x sobre la recta entre los puntos p4 y 100,100

        if(Yi>100-(a/2)){ //el doblez hacia abajo antes de la mitad
            //se dibuja un poligono con un los lados siguientes:
            h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2,
             h.getGraphics()); //lado (p1.getx,p1.gety)a(X2,Y2)
            h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi,
            h.getGraphics()); //lado (p2.getx,p2.gety)a(Xi,Yi)
```

```
    h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
     //lado del poligono (Xi,Yi)a(X2,Y2)
    //el otro lado del poligono seria la linea que va de p1 a p2
    try{//aqui se da una pausa para poder ver los cambios gradualmente
    y generar el efecto de movimiento
    Thread.sleep(2); //2
    }catch(InterruptedException e){}
    //despues de la pausa se borra el poligono que se dibujo anteriormente
    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
    }
else{//punto medio del doblez hacia abajo
    if(Yi==100-(a/2)){
        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
         //se dibuja la linea que representa al papel doblado justo a
         la mitad del dolbez try{//aqui se da una pausa para poder ver los cambios
              gradualmente
         y generar el efecto de movimiento
        Thread.sleep(2); //2
        }catch(InterruptedException e){}
        h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
        //se borra la linea de la mitad del doblez
    }
    else{ //aqui continua el doblez hacia abajo depues de la mitad
          //se dibujara un poligono para simular la hoja que se dobla hacia abajo
        int X[]={p1.getX(),X2,Xi,p2.getX()}; //arreglo que contiene las x
         de los puntos del poligono
        int Y[]={p1.getY(),Y2,Yi,p2.getY()}; //arreglo que contiene las y
         de los puntos del poligono
        h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());
        //se dibuja el poligono con los arreglos

        try{ //aqui se da una pausa para poder ver los cambios gradualmente
        y generar el efecto de movimiento
        Thread.sleep(2); //2
        }catch(InterruptedException e){}
        //se dibuja un triangulo para borrar la parte del poligono que
        debe desaparecer despues de cada iteracion

        int Xp[]={p1.getX(),X2,((((100-(((100-((100-(a/2))))*100/a)*b/100))-(100))*
        (p3x-(-100))/(p3y-(100)))+(-100))};//x de los puntos del triangulo, el ultimo
             x equivale a la hoja del lado derecho cuando se esta a la mitad del doblez
        int Yp[]={p1.getY(),Y2,100-(((100-((100-(a/2))))*100/a)*b/100)};
        //"y" de los puntos del triangulo, el ultimo "y" equivale a la hoja del
        lado derecho cuando se esta a la mitad del doblez
        h.dibujarPoligonoDeColor(Color.white,3,Xp,Yp,h.getGraphics());
        //se dibuja el triangulo para borrar
        //en caso de haber puntos del poligono que no se borren se usa el
        siguiente triangulo
        int Xpp[]={p1.getX(),X2,-150};//x de los puntos del triangulo, el
        ultimo x es lejano para borrar todos los puntos faltantes del poligono
        int Ypp[]={p1.getY(),Y2,(-150-p1.getX())*(p2.getY()-p1.getY())/(p2.getX()-
        p1.getX())+p1.getY()};
        //"y" de los puntos del triangulo, el ultimo "y" equivale al "y" sobre la
             linea del doblez con x=-150
        h.dibujarPoligonoDeColor(Color.white,3,Xpp,Ypp,h.getGraphics()); //se dibuja
             el triangulo para borrar
```

```
        }
    }

}

for (int Yi=p4y;Yi<100;Yi++){
//for para hacer el doblez hacia arriba Yi va de p4y a 100
    int P=((100-Yi)*100/a);
    //porcentaje , que porcentaje de "a" equivale la distancia de 100 a Yi
    int Y2=100-(P*b/100); //valor del recorrido de Y2 equivalente al porcentaje
     que ha recorrido Yi( si 100%=>b entonces P=>?)
    int Xi=(((Yi-(100))*(p4x-(100))/(p4y-(100)))+(100));
    //obteniendo el valor de x sobre la recta entre los puntos p4 y 100,100
    int X2=(((Y2-(100))*(p3x-(-100))/(p3y-(100)))+(-100));
    //obteniendo el valor de x sobre la recta entre los puntos p3 y -100,100

    if(Yi<100-(a/2)){ //doblez hacia arriba antes de la mitad

            int Xp[]={p1.getX(),p2.getX(),Xi,X2};
            //valores de x de los puntos del poligono que se dibuje cuando el doblez va
                 hacia arriba
            int Yp[]={p1.getY(),p2.getY(),Yi,Y2};
            //valores de x de los puntos del poligono que se dibuje cuando el doblez va
                 hacia arriba
            h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
            //poligono que simula el doblez hacia arriba

            int X[]={p2.getX(),Xi,X2,p3x,p4x}; //valores de x de los puntos del poligono
                 que borraran el poligono que se dibujo cuando el doblez era hacia abajo
            int Y[]={p2.getY(),Yi,Y2,p3y,p4y}; //valores de y de los puntos del poligono
                 que borraran el poligono que se dibujo cuando el doblez era hacia abajo

            h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());

            if(Yi<-100){ //si el doblez exede el margen inferior, se debe redibujar este
                 margen
            h.dibujaLineaDeColor(Color.blue, 100,
                 -100,(((-100-(Yi))*(p2.getX()-(Xi))/(p2.getY()-(Yi)))+(Xi)) ,-100,
                 h.getGraphics()); //redibujando margen de lado derecho
            h.dibujaLineaDeColor(Color.blue, -100,
                 -100,(((-100-(Yi))*(X2-(Xi))/(Y2-(Yi)))+(Xi)) ,-100, h.getGraphics());
                 //redibujando margen de lado izquierdo
            }
            else{ //cuando se juntan los dos lados
                h.dibujaLineaDeColor(Color.blue, -100, -100, 100,-100,h.getGraphics());
                    //redibujando margen inferior
            }

            h.dibujaLineaDeColor(Color.blue, -100, -100,
                 -100,(((-100-(X2))*(Yi-(Y2))/(Xi-(X2)))+(Y2)),h.getGraphics());
            //linea que redibuja el margen del doblez
            try{ //pausa para efectos de animacion
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
    }

    else{

        if(Yi==100-(a/2)){ //Yi esta en la mitad del doblez hacia arriba
```

```
                        h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                        //se dibuja la linea del doblez
                        try{ //pausa
                        Thread.sleep(2); //2
                        }catch(InterruptedException e){}
                    }
                    else{
//se dibujan las lineas que representan al papel doblandose hacia arriba despues de la mitad
                        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
                        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                        try{
                        Thread.sleep(2);//2
                        }catch(InterruptedException e){}
//se borran las lineas que representan al papel doblandose hacia arriba despues de la mitad
                        h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                        h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
                        h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());

                    }
                }
            }
            h.refresh(h.getGraphics());

            /*//estas lineas dibujarian como que da el doblez, sin animacion
            h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
            h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
            h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,p4x+h.getWidth()/2
                 ,h.getHeight()/2-p4y);*/
        }
    }
    else{

        cPunto p2=new cPunto(xYm100,-100);
         //este es el extremo derecho de la linea de doblez x=f(-100) y y=-100
        CpuntoDoble p2Doble=new CpuntoDoble(xYm100D,-100.0);
        //el mismo punto de arriba pero en valores tpo double
        extremox2=xYm100+h.getWidth()/2; //ajustando el valor de x con el centro del panel
        extremoy2=h.getHeight()/2-(-100); //ajustando el valor de y con el centro del panel
        //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
        cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
        //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

        ll.insertarAlInicio(nuevaLinea); //se inserta la nueva linea en la lista de lineas
        double ty=100.0+p1Doble.getY();
        //distancia de la punta izquierda en "y" a el punto extremo izquierdo en "y"
        double tx=100.0+p2Doble.getX();
        //distancia de la punta izquierda en x a el punto extremo derecho en x
        double tanTetha=tx/ty;
        //tan=CO/CA CO=tx CA=ty la tangente es el angulo de doblado, con respecto a la vertical
        double tetha=Math.toDegrees(Math.atan(tanTetha));
        //angulo en grados
        double alpha=90-(2.0*tetha);//angulo de doblado con respecto a la horizontal
        double cosAlpha=Math.cos(Math.toRadians(alpha));
         //coseno del anugulo de doblado
        double x0=cosAlpha*ty;
        //Cos=CA/H   H=ty  CA=x0 a partir del angulo se obtiene la componente en x del
        punto donde queda la punta del doblez
        int p3x=(int)(p1Doble.getX()+x0); //x de la punta del doblez
```

```
double senAlpha=Math.sin(Math.toRadians(alpha));
//seno del angulo de doblado
double y0=senAlpha*ty;
 //Sen=CO/H   H=ty  CO=y0 a partir del angulo se obtiene la componente en "y" del
  punto donde queda la punta de la hoja doblada
int p3y=(int)(p1Doble.getY()-y0); //"y" de la punta de la hoja ya doblada

/* //estas lineas dibujarian como que da el doblez, sin animacion
h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,extremox2,extremoy2);
*/
    int a=p3y-(-100);
    //"a" es la distancia del punto donde queda la hoja doblada a -100 en el eje Y

    for (int Yi=-100;Yi<p3y;Yi++){
    //for para hacer la animacion de doblado hacia arriba, Yi se movera desde -100
     a py3 sobre el eje Y
       int Xi=(((Yi-(-100))*(p3x-(-100))/(p3y-(-100)))+(-100));
       //el valor de x cuando Yi se va moviendo, este x esta en la recta que pasa
       por el (-100,-100) y el p3
       if(Yi<(a/2)-100){ //cuando Yi se mueve antes de la mitad del doblez
           h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
           //se dibuja la linea que simulara el doblado del lado izquierdo
           h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //se dibuja la linea que simulara el doblado del lado derecho
           try{ //pausa
           Thread.sleep(2); //10
           }catch(InterruptedException e){}
           h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
            //se borra la linea que del doblado en el lado izquierdo
           h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //se borra la linea que del doblado en el lado derecho
       }
       else{ //Yi va mas alla de la mitad del doblez hacia arriba
           if(Yi==(a/2)-100){ //justo a la mitad del doblez hacia arriba

               try{ //pausa
               Thread.sleep(2); //2
               }catch(InterruptedException e){}

           }
           else{
           //Yi se mueve hacia arriba despues de pasar por la linea del doblez (la mitad)
               int X[]={p1.getX(),Xi,p2.getX()};
               //x de los puntos que dibujan el triangulo que simula la hoja de papel doblada
               int Y[]={p1.getY(),Yi,p2.getY()};
               //y de los puntos que dibujan el triangulo que simula la hoja de papel doblada
               h.dibujarPoligonoDeColor(Color.BLUE, 3, X, Y,h.getGraphics());
                //dibujando la hoja doblada

               try{ //pausa
               Thread.sleep(2); //3
               }catch(InterruptedException e){}
               /*int Xp[]={p2.getX(),X2,100};
               int Yp[]={p2.getY(),Y2,-100};
               h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());*/
           }
       }
```

```
}

for (int Yi=(-100)+a;Yi>-100;Yi--){
//for que hace la animacion hacia abajo, Yi va de -100+a a -100 para
cubrir toda la distancia deldoblez a

    int Xi=(((Yi-(-100))*(p3x-(-100))/(p3y-(-100)))+(-100));
    //Xi es el valor de x ,cuando y vale Yi en la recta entre p3x,p3y y -100,-100

    if(Yi>(a/2)-100){
    //entra en las iteraciones donde el doblez va hacia abajo y no ha
     pasado la mitad de la animacion

           int X[]={p1.getX(),Xi,p2.getX(),p3x};
           //Arreglo que guarda las x de los puntos del poligono
           que borra el triangulo que representa la hohja doblada
           int Y[]={p1.getY(),Yi,p2.getY(),p3y};
           //Arreglo que guarda las y de los puntos del poligono que
           borra el triangulo que representa la hohja doblada

           h.dibujarPoligonoDeColor(Color.white,4, X, Y, h.getGraphics());
           //el poligono que borra el triangulo, para simular el doblez

           try{ //pausa
           Thread.sleep(2); //2
           }catch(InterruptedException e){}

    }
    else{ //aqui entra cuando el doblez va mas alla dela mitad

        if(Yi==(a/2)-100){ //exactamente el doblez esta en la mitad
            h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
            //linea del doblez
            try{//pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }
        else{ //entra cuando la animacion va mas alla de la mitad hacia abajo
            h.dibujaLinea(p1.getX(), p1.getY(), p2.getX(), p2.getY(), h.getGraphics());
            //LINEA DEL DOBLEZ
            h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
            //extremo izquierdo de la hoja que se mueve en cada iteracion
            h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //extremo derecho de la hoja que se mueve en cada iteracion

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

            h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
             //se borra extremo izquierdo de la hoja que se mueve en cada iteracion
            h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //se borra extremo derecho de la hoja que se mueve en cada iteracion

            }
    }
}
h.refresh(h.getGraphics());
 //esta linea redibuja la hoja de papel con el doblez ya marcado
```

```
    }
}
else{ //se mete a este else cuando x>=-100 cuando y=100

    if(xY100>100){ //se mete al if cuando x>100 cuando y=100
        cPunto p1=new cPunto(100,yX100); //este es el extremo derecho de la linea de doblez
        CpuntoDoble p1Doble=new CpuntoDoble(100.0,yX100D);
         //el mismo punto de arriba pero en valores tipo double
        extremox1=100+h.getWidth()/2; //ajustando el punto derecho en x alcentro del panel
        extremoy1=h.getHeight()/2-yX100; //ajustando el punto derecho en y alcentro del panel
        if(xYm100<-100){ //si de lado izquierdo x<-100 cuando y=-100
            cPunto p2=new cPunto(-100,yXm100);
            //este es el extremo izquierdo de la linea de doblez
            CpuntoDoble p2Doble=new CpuntoDoble(-100.0,yXm100D);
             //este es el mismo punto de arriba solo que en valores tipo double
            extremox2=-100+h.getWidth()/2; //ajustando el punto izquierdo en x alcentro del panel
            extremoy2=h.getHeight()/2-yXm100;//ajustando el punto izquierdo en y alcentro del panel
            //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
            cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
            //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

            ll.insertarAlInicio(nuevaLinea);
            //se inserta la nueva linea en la lista de lineas
            double r1=100.0-p1Doble.getY();
            // distancia del extremo derecho al punto 100 sobre el eje y
            double r2=100.0-p2Doble.getY();
             // distancia del extremo izquierdo al punto 100 sobre el eje y
            double s1=200.0-r1;
             //distancia del extremo derecho al punto -100 sobre el eje y
            double s2=200.0-r2;
            //distancia del extremo izquierdo al punto -100 sobre el eje y
            if(r1+r2>s1+s2){
             //entra al if cuando la parte de arriba del doblez es mas grande
              y se debe doblar lo de abajo
                double tanTetha=(p1Doble.getY()-p2Doble.getY())/200.0;
                //tangente de tetha=CO/CA CO=distancia entre los extremos sobre el eje "y"
                     CA=distancia horizontal de la hoja
                double tetha=Math.toDegrees(Math.atan(tanTetha));
                //tetha en grados
                double alpha=90.0-tetha;
                //alpha es el angulo con respecto a la vertical
                double betha=(2.0*alpha)-90.0;
                //betha es el angulo de doblez con respecto a la horizontal
                double cosBetha=Math.cos(Math.toRadians(betha));
                //coseno del angulo de doblez
                double x0=cosBetha*s1; //cos=CA/H CA=x0 H=s1
                int p3x=(int)(p1Doble.getX()-x0);
                //este es uno de los puntos donde queda la hoja doblada, la punta derecha en x
                double senBetha=Math.sin(Math.toRadians(betha));
                //seno del angulo de doblez
                double y0=senBetha*s1; //sen=CO/H CO=y0 H=s1
                int p3y=(int)(p1Doble.getY()+y0);
                //este es uno de los puntos donde queda la hoja doblada, la punta derecha en y
                double x01=cosBetha*s2; //cos=CA/H CA=x01 H=s2
                int p4x=(int)(p2Doble.getX()-x01);
                //este es uno de los puntos donde queda la hoja doblada, la punta izquierda en x
                double y01=senBetha*s2; //sen=CO/H CO=y0 H=s2
                int p4y=(int)(p2Doble.getY()+y01);
                 //este es uno de los puntos donde queda la hoja doblada, la punta izquierda en y
```

```
int a=p3y-(-100);
//distancia en el eje Y de la punta derecha del doblado a la punta derecha de
 la hoja de papel
int b=p4y-(-100);
//distancia en el eje X de la punta derecha del doblado a la punta derecha de
 la hoja de papel

for (int Yi=-100;Yi<(-100)+a;Yi++){
 //en este for se hace la animacion de la hoja doblandose hacia arriba
  Yi recorre la distancia "a"
    int P=((Yi+100)*100/a); //Porcentaje del recorrido de Yi sobre "a"
    int Y2=(P*b/100)-100; //el equivalente en "b" del porcentaje recorrido en a
    int Xi=(((Yi-(-100))*(p3x-(100))/(p3y-(-100)))+( 100));
    //obteniendo la x sobre la recta que va de p3 a la punta derecha de la
     hoja sin doblar, dependiendo de canto valga yi
    int X2=(((Y2-(-100))*(p4x-(-100))/(p4y-(-100)))+(-100));
    //obteniendo la x sobre la recta que va de p4 a la punta izquierda de la
     hoja sin doblar, dependiendo de canto valga y2

    if(Yi<(a/2)-100){ //si la animacion del doblez esta antes de la mitad
    //las siguientes lineas dibujan la hoja moviendose hacia arriba en cada iteracion
    h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
    h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());

    try{ //pausa
    Thread.sleep(2);
    }catch(InterruptedException e){}
    //las llineas borran lo dibujado anteriormente para simular el movimeiento
    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
    }
    else{ //pasando la mitad de la animacion del doblez
        if(Yi==(a/2)-100){ //justo en la mitad de la animacion
            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }
        else{ //cuando se pasa el punto medio de la animacion
            int X[]={p1.getX(),Xi,X2,p2.getX()};
             //arreglo con las x de los puntos que dibujaran la hoja doblada
             en cada iteracion
            int Y[]={p1.getY(),Yi,Y2,p2.getY()};
            //arreglo con las y de los puntos que dibujaran la hoja doblada
            en cada iteracion
            h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());
             //se dibuja el trapecio que simula la hoja doblandose en
             cada iteracion

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
            //los siguientes dos arreglos tienen los valores x y y del tiangulo
            que borra la parte del trapecio que debe desaparecer en cada iteracion
            int Xp[]={p2.getX(),X2,-100}; //x de los puntos del trapecio
            int Yp[]={p2.getY(),Y2,-100}; //x de los puntos del trapecio
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());
```

```
            //se borra lo ya dibujado para simular movimiento
        }
    }

}

for (int Yi=(-100)+a;Yi>-100;Yi--){
//este for simula el moviento de la hoja hacia abajo
    int P=((Yi+100)*100/a); //Porcentaje del recorrido de Yi sobre "a"
    int Y2=(P*b/100)-100; //el equivalente en "b" del porcentaje recorrido en a
    int Xi=(((Yi-(-100))*(p3x-(100))/(p3y-(-100)))+(100));
    //obteniendo la x sobre la recta que va de p3 a la punta derecha de la
    hoja sin doblar, dependiendo de cuanto valga yi
    int X2=(((Y2-(-100))*(p4x-(-100))/(p4y-(-100)))+(-100));
    //obteniendo la y sobre la recta que va de p4 a la punta izquierda de la
     hoja sin doblar, dependiendo de cuanto valga y2

    if(Yi>(a/2)-100){
    //cuando va el doblez hacia abajo antes de la mitad del recorrido

        int X[]={p1.getX(),Xi,X2,p4x,p3x};
        //arreglo que guarda los puntos en x del poligono que borra el trapecio en
         cada iteracion para simular el doblado hacia abajo
        int Y[]={p1.getY(),Yi,Y2,p4y,p3y};
        //arreglo que guarda los puntos en y del poligono que borra el trapecio en
        cada iteracion para simular el doblado hacia abajo

        int Xp[]={p1.getX(),p2.getX(),X2,Xi};
         //arreglo que guarda los puntos en x del trapecio que en cada iteracion
         simula el doblado hacia abajo
        int Yp[]={p1.getY(),p2.getY(),Y2,Yi};
         //arreglo que guarda los puntos en y del trapecio que en cada iteracion
         simula el doblado hacia abajo

        h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());
        //el poligono que borra la hoja doblada hacia arriba al mismo tiempo que se
        crea el trapecio que simula el doblez hacia abajo
        if(Yi>99){ //si el doblez exedio el margen superior
        h.dibujaLineaDeColor(Color.blue, 100,
             100,(((100-(Yi))*(p1.getX()-(Xi))/(p1.getY()-(Yi)))+(Xi)) , 100,
             h.getGraphics());
        //se redibuja el margen superior del lado izquierdo
        h.dibujaLineaDeColor(Color.BLUE, -100,
             100,(((100-(Yi))*(X2-(Xi))/(Y2-(Yi)))+(Xi)) ,
         100, h.getGraphics());
         //se redibuja el margen superior del lado derecho
        }
        else{ //se redibuja todo el margen superior, union de los dor margenes del
        if anterior
            h.dibujaLineaDeColor(Color.blue, -100, 100, 100,100,h.getGraphics());
        }
        h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
        //se dibuja el trapecio que simula el doblez hacia abajo de la hoja
        h.dibujaLineaDeColor(Color.blue, -100, 100,
             -100,(((-100-(Xi))*(Y2-(Yi))/(X2-(Xi)))+(Yi)),h.getGraphics());
        //esta linea dibuja el margen izuqerdo cada vez que el doblez va borrando
        el margen
        try{
        Thread.sleep(2); //2
```

```
            }catch(InterruptedException e){}
        }
        else{ //cuando se pasa de la mitad del doblez

            if(Yi==(a/2)-100){ //exactamente a la mitad del doblez
                h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                try{ //pausa
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
            }
            else{
                //se dibujan las lineas que simulan la hoja doblandose hacia abajo
                h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                try{
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
                //se borran las lineas anteriores para dar animacion
                h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
                h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),X2,Y2, h.getGraphics());
                h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
            }
        }
    }
    h.refresh(h.getGraphics()); //esta linea redibuja la hoja de papel con el doblez ya
         marcado

        /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
        h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
        h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
        h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,p4x+h.getWidth()/2
             ,h.getHeight()/2-p4y);
         *
         */
    }
    else{ //cuando la parte de arriba del doblez es mas chica y se debe doblar lo de abajo
        double tanTetha=(p1Doble.getY()-p2Doble.getY())/200.0;
        //tanTehta=catetoOpuesto(distancia entre puntos extremos)/catetoAdyacente(ancho de
             hoja)
        double tetha=Math.toDegrees(Math.atan(tanTetha));
         //tetha=arctan(tanTheta) en grados
        double alpha=90.0-tetha;
        //mitad del angulo de doblado en funcion de la vertical
        double betha=(2.0*alpha)-90.0;
        //angulo de doblado en funcion de la horizontal
        double cosBetha=Math.cos(Math.toRadians(betha));
        //coseno de betha en grados
        double x0=cosBetha*r1;
        //catetoAdyacente(x0)=cosenoAngulo(cosBetha)*Hipotenusa(r1)
        int p3x=(int)(p1Doble.getX()+x0); //"x" del punto donde quedara doblada la punta
             derecha
        double senBetha=Math.sin(Math.toRadians(betha));
        //sen del angulo betha en grados
        double y0=senBetha*r1; //catetoOpoesto(y0)=senoAngulo(senBetha)*Hipotenusa(r1)
        int p3y=(int)(p1Doble.getY()-y0);
         //"y" del punto donde quedara doblada la punta derecha
        double x01=cosBetha*r2;//catetoAdyacente(x01)=cosenoAngulo(cosBetha)*Hipotenusa(r2)
        int p4x=(int)(p2Doble.getX()+x01);
```

```
//"x" del punto donde quedara doblada la punta izquierda
double y01=senBetha*r2;//catetoOpuesto(y01)=senoAngulo(senBetha)*Hipotenusa(r2)
int p4y=(int)(p2Doble.getY()-y01);
//"y" del punto donde quedara doblada la punta izquierda

int b=100-p3y; //distancia (en y) entre el punto de doblez derecho(p3) con
el punto que se dobla(100,100)
int a=100-p4y; //distancia (en y) entre el punto de doblez izquierdo(p4) con
el punto que se dobla (-100,100)

for (int Yi=100;Yi>p4y;Yi--){ //for para hacer la animacion de doblado hacia
abajo Yi se va a mover el doblez de Yi=100 a p4y
int P=((100-Yi)*100/a); //porcentaje del recorrido de Yi ( regla de tres si
 a=>100% entonces Yi=>? )
int Y2=100-(P*b/100); //valor del recorrido de Y2 equivalente al porcentaje
que ha recorrodo Yi( si 100%=>b entonces P=>?)
int X2=(((Y2-(100))*(p3x-(100))/(p3y-(100)))+(100));
//obteniendo el valor de x sobre la recta entre los puntos p3 y 100,100
int Xi=(((Yi-(100))*(p4x-(-100))/(p4y-(100)))+(-100));
//obteniendo el valor de x sobre la recta entre los puntos p4 y -100,100

if(Yi>100-(a/2)){ //el doblez hacia abajo antes de la mitad
//se dibuja un poligono con un los lados siguientes:
h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
 //lado (p1.getx,p1.gety)a(X2,Y2)
h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
 //lado (p2.getx,p2.gety)a(Xi,Yi)
h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
//lado del poligono (Xi,Yi)a(X2,Y2)
//el otro lado del poligono seria la linea que va de p1 a p2
try{//aqui se da una pausa para poder ver los cambios gradualmente y
generar el efecto de movimiento
Thread.sleep(2); //2
}catch(InterruptedException e){}
//despues de la pausa se borra el poligono que se dibujo anteriormente
h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
}
else{ //punto medio del doblez hacia abajo
    if(Yi==100-(a/2)){
        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
        //se dibuja la linea que representa al papel doblado justo a la mitad del
            dolbez
        try{
        //aqui se da una pausa para poder ver los cambios gradualmente y generar
        el efecto de movimiento
        Thread.sleep(2); //2
        }catch(InterruptedException e){}
        h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
         //se borra la linea de la mitad del doblez
    }
    else{ //aqui continua el doblez hacia abajo depues de la mitad
        //se dibujara un poligono para simular la hoja que se dobla hacia abajo
        int X[]={p1.getX(),X2,Xi,p2.getX()};
        //arreglo que contiene las x de los puntos del poligono
        int Y[]={p1.getY(),Y2,Yi,p2.getY()};
        //arreglo que contiene las y de los puntos del poligono
        h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());
```

```
                //se dibuja el poligono con los arreglos

                try{ //aqui se da una pausa para poder ver los cambios gradualmente y generar
                    el efecto de movimiento
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
                //se dibuja un triangulo para borrar la parte del poligono que debe
                    desaparecer despues de cada iteracion

                int Xp[]={p1.getX(),X2,((((100-(((100-((100-(a/2))))*100/a)*b/100))-(100))*
                (p3x-(-100))/(p3y-(100)))+(-100))};
                //x de los puntos del triangulo, el ultimo x equivale a la hoja del lado
                derecho cuando se esta a la mitad del doblez
                int Yp[]={p1.getY(),Y2,100-(((100-((100-(a/2))))*100/a)*b/100)};
                //"y" de los puntos del triangulo, el ultimo "y" equivale a la hoja
                del lado derecho cuando se esta a la mitad del doblez
                h.dibujarPoligonoDeColor(Color.white,3,Xp,Yp,h.getGraphics());
                //se dibuja el triangulo para borrar
                //en caso de haber puntos del poligono que no se borren se usa el
                siguiente triangulo
                int Xpp[]={p1.getX(),X2,-150}; //x de los puntos del triangulo,
                el ultimo x es lejano para borrar todos los puntos faltantes del poligono
                int
                    Ypp[]={p1.getY(),Y2,(-150-p1.getX())*(p2.getY()-p1.getY())/(p2.getX()-p1.getX())+
                p1.getY()};
                //"y" de los puntos del triangulo, el ultimo "y" equivale al "y" sobre la linea
                 del doblez con x=-150
                h.dibujarPoligonoDeColor(Color.white,3,Xpp,Ypp,h.getGraphics());
                //se dibuja el triangulo para borrar

            }
        }

    }

    for (int Yi=p4y;Yi<100;Yi++){
    //for para hacer el doblez hacia arriba Yi va de p4y a 100
        int P=((100-Yi)*100/a);
        //porcentaje , que porcentaje de "a" equivale la distancia de 100 a Yi
        int Y2=100-(P*b/100);
        //valor del recorrido de Y2 equivalente al porcentaje que ha recorrido
        Yi( si 100%=>b entonces P=>?)
        int Xi=(((Yi-(100))*(p4x-(-100))/(p4y-(100)))+(-100));
        //obteniendo el valor de x sobre la recta entre los puntos p4 y -100,100
        int X2=(((Y2-(100))*(p3x-(100))/(p3y-(100)))+(100));
        //obteniendo el valor de x sobre la recta entre los puntos p3 y 100,100

        if(Yi<100-(a/2)){ //doblez hacia arriba antes de la mitad

                int Xp[]={p1.getX(),p2.getX(),Xi,X2};
                 //valores de x de los puntos del poligono que se dibuje cuando el
                 doblez va hacia arriba
                int Yp[]={p1.getY(),p2.getY(),Yi,Y2};
                //valores de x de los puntos del poligono que se dibuje cuando el
                doblez va hacia arriba
                h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
                //poligono que simula el doblez hacia arriba

                int X[]={p2.getX(),Xi,X2,p3x,p4x};
```

```
            //valores de x de los puntos del poligono que borraran el trapecio
             que se dibujo cuando el doblez era hacia abajo
            int Y[]={p2.getY(),Yi,Y2,p3y,p4y};
            //valores de y de los puntos del poligono que borraran el trapecio
            que se dibujo cuando el doblez era hacia abajo

            h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());



            if(Yi<-100){
            //si el doblez exede el margen inferior, se debe redibujar este margen
            h.dibujaLineaDeColor(Color.blue, 100,
                 -100,(((-100-(Yi))*(p1.getX()-(Xi))/(p1.getY()-(Yi)))+(Xi)) ,-100,
             h.getGraphics()); //redibujando margen de lado derecho
            h.dibujaLineaDeColor(Color.blue, -100, -100,(((-100-(Yi))*(X2-(Xi))/
            (Y2-(Yi)))+(Xi)) ,-100, h.getGraphics());
            //redibujando margen de lado izquierdo
            }
            else{ //cuando se juntan los dos lados
                h.dibujaLineaDeColor(Color.blue, -100, -100, 100,-100,h.getGraphics());
                     //redibujando margen inferior
            }

            h.dibujaLineaDeColor(Color.blue, 100, -100,
                 100,(((100-(X2))*(Yi-(Y2))/(Xi-(X2)))+(Y2)),h.getGraphics());
            //linea que redibuja el margen del doblez
            try{ //pausa para efectos de animacion
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }

        else{

            if(Yi==100-(a/2)){ //Yi esta en la mitad del doblez hacia arriba
                h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                //se dibuja la linea del doblez
                try{ //pausa
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
            }
            else{
                //se dibujan las lineas que representan al papel doblandose hacia
                 arriba despues de la mitad
                h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
                h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                try{
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
                //se borran las lineas que representan al papel doblandose hacia
                arriba despues de la mitad
                h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),X2,Y2, h.getGraphics());
                h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());

            }
        }
}
```

```
    h.refresh(h.getGraphics());

        /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
        h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
        h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
        h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,p4x+h.getWidth()/2
             ,h.getHeight()/2-p4y);
         *
         */
    }
}
else{ //si -100>x>100 cuando y=-100
    cPunto p2=new cPunto(xYm100,-100);//este es el punto izquierdo
    CpuntoDoble p2Doble=new CpuntoDoble(xYm100D,-100.0);
    //el mismo punto solo que en valores tipo double
    extremox2=xYm100+h.getWidth()/2; //ajustando el punto al centro del panel
    extremoy2=h.getHeight()/2-(-100); //ajustnado el punto al centro del panel
    //h.getGraphics().drawLine(extremox1,extremoy1 ,extremox2 ,extremoy2);
    cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
    //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

    ll.insertarAlInicio(nuevaLinea);
    //se inserta la nueva linea en la lista de lineas
    double ty=100.0+p1Doble.getY();
    //uno de los lados del triangulo que quedara doblado (Hipotenusa para conseguir
    angulo de doblez)
    double tx=100.0-p2Doble.getX();
     //el otro lado del triangulo que quedara doblado, se usa para conseguir el
     angulo de doblez con respecto a la vertical
    double tanTetha=tx/ty; //Tangente del angulo de doblez con respecto a la vertical
    double tetha=Math.toDegrees(Math.atan(tanTetha));//angulo tetha en grados
    double alpha=90-(2.0*tetha); //angulo de doblado con respecto a la horizontal
    double cosAlpha=Math.cos(Math.toRadians(alpha));//coseno del angulo de doblado
    double x0=cosAlpha*ty; //cosAngulo= CA(x0)/H(ty)
    int p3x=(int)(p1Doble.getX()-x0);//x de la punta ya doblada
    double senAlpha=Math.sin(Math.toRadians(alpha));
    //seno del angulo de doblez en grados
    double y0=senAlpha*ty;//senAngulo=CO(y0)/H(ty)
    int p3y=(int)(p1Doble.getY()-y0);//y de la punta ya doblada

    int a=p3y-(-100);
    //"a" es la distancia del punto donde queda la hoja doblada a -100 en el eje Y

    for (int Yi=-100;Yi<p3y;Yi++){
    //for para hacer la animacion de doblado hacia arriba, Yi se movera desde
    -100 a py3 sobre el eje Y
        int Xi=(((Yi-(-100))*(p3x-(100))/(p3y-(-100)))+(100));
        //el valor de x cuando Yi se va moviendo, este x esta en la recta que
         pasa por el (100,-100) y el p3
        if(Yi<(a/2)-100){ //cuando Yi se mueve antes de la mitad del doblez
        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
        //se dibuja la linea que simulara el doblado del lado izquierdo
        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
        //se dibuja la linea que simulara el doblado del lado derecho
        try{//pausa
        Thread.sleep(2); //10
        }catch(InterruptedException e){}
        h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
        //se borra la linea que del doblado en el lado izquierdo
```

```
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
    //se borra la linea que del doblado en el lado derecho
    }
    else{ //Yi va mas alla de la mitad del doblez hacia arriba
        if(Yi==(a/2)-100){ //justo a la mitad del doblez hacia arriba

            try{//pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

        }
        else{//Yi se mueve hacia arriba despues de pasar por la linea del doblez (la mitad)
            int X[]={p1.getX(),Xi,p2.getX()};
            //x de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            int Y[]={p1.getY(),Yi,p2.getY()};
             //y de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            h.dibujarPoligonoDeColor(Color.BLUE, 3, X, Y,h.getGraphics());
            //dibujando la hoja doblada

            try{//pausa
            Thread.sleep(2); //3
            }catch(InterruptedException e){}
            /*int Xp[]={p2.getX(),X2,100};
            int Yp[]={p2.getY(),Y2,-100};
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());*/
        }
    }

}

for (int Yi=(-100)+a;Yi>-100;Yi--){
//for que hace la animacion hacia abajo, Yi va de -100+a a -100 para cubrir toda
la distancia deldoblez a

    int Xi=(((Yi-(-100))*(p3x-(100))/(p3y-(-100)))+(100));
    //Xi es el valor de x ,cuando y vale Yi en la recta entre p3x,p3y y 100,-100

    if(Yi>(a/2)-100){
    //entra en las iteraciones donde el doblez va hacia abajo y no ha pasado la
    mitad de la animacion

            int X[]={p1.getX(),Xi,p2.getX(),p3x};
            //Arreglo que guarda las x de los puntos del poligono que borra el
            triangulo que representa la hoja doblada
            int Y[]={p1.getY(),Yi,p2.getY(),p3y};
             //Arreglo que guarda las y de los puntos del poligono que borra el
             triangulo que representa la hoja doblada

            h.dibujarPoligonoDeColor(Color.white,4, X, Y, h.getGraphics());
             //el poligono que borra el triangulo, para simular el doblez

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

    }

    else{ //aqui entra cuando el doblez va mas alla dela mitad
```

```
                    if(Yi==(a/2)-100){ //exactamente el doblez esta en la mitad
                        h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                        //linea del doblez
                        try{ //pausa
                        Thread.sleep(2); //2
                        }catch(InterruptedException e){}
                    }
                    else{ //entra cuando la animacion va mas alla de la mitad hacia abajo
                        h.dibujaLinea(p1.getX(), p1.getY(), p2.getX(), p2.getY(), h.getGraphics());
                        //LINEA DEL DOBLEZ
                        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
                        //extremo izquierdo de la hoja que se mueve en cada iteracion
                        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                        //extremo derecho de la hoja que se mueve en cada iteracion

                        try{ //pausa
                        Thread.sleep(2); //2
                        }catch(InterruptedException e){}

                        h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
                        //se borra extremo izquierdo de la hoja que se mueve en cada iteracion
                        h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                        //se borra extremo derecho de la hoja que se mueve en cada iteracion

                        }
                }
            }
            h.refresh(h.getGraphics());//esta linea redibuja la hoja de papel con el doblez ya marcado

            /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
            h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
            h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,extremox2,extremoy2);
            */

        }
    }
    else{ // -100<x<100 cuando y=100
        cPunto p1=new cPunto(xY100,100); //se establece el primer punto extremo
        CpuntoDoble p1Doble=new CpuntoDoble(xY100D,100.0); //el mismo punto de arriba solo
        que en valores double
        extremox1=xY100+h.getWidth()/2; //ajustando el punto a la mitad del panel
        extremoy1=h.getHeight()/2-100; //ajustando el punto a la mitad del panel
        if(xYm100<-100){ //se mete a heste if xuando y=-100 y x<-100
            cPunto p2=new cPunto(-100,yXm100);
            //este es el otro extremo del doblez, el izquierdo entonces p1 es el derecho
            CpuntoDoble p2Doble=new CpuntoDoble(-100.0,yXm100D);
            //mismo punto de arriba solo que en valores double
            extremox2=-100+h.getWidth()/2; //ajustando al centro del panel
            extremoy2=h.getHeight()/2-yXm100; //ajustando al centro del panel
            //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
            cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
            //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

            ll.insertarAlInicio(nuevaLinea);
            //se inserta la nueva linea en la lista de lineas

            double ty=100.0-p2Doble.getY();
            //uno de los lados del triangulo que quedara doblado (Hipotenusa
             para conseguir angulo de doblez)
```

```
double tx=100.0+p1Doble.getX();
//el otro lado del triangulo que quedara doblado, se usa para conseguir el angulo de
     doblez con respecto a la vertical
double tanTetha=tx/ty; //arco tangente del angulo de doblado con respecto a la vertical
double tetha=Math.toDegrees(Math.atan(tanTetha));
//tangente en grados del angulo de doblado con respecto a la vertical
double alpha=90-(2.0*tetha); //angulo de doblado con respecto a la horizontal
double cosAlpha=Math.cos(Math.toRadians(alpha));
//coseno del angulo de doblado
double x0=cosAlpha*ty; //cosAngulo= CA(x0)/H(ty)
int p3x=(int)(p2Doble.getX()+x0);
//componente en x del punto donde quedara la punta izquierda superior doblada
double senAlpha=Math.sin(Math.toRadians(alpha));
//seno del angulo de doblado en grados
double y0=senAlpha*ty; //senAngulo=CO(y0)/H(ty)
int p3y=(int)(p2Doble.getY()+y0);
//componente en y del punto donde quedara la punta ya doblada

int a=100-p3y;
//"a" es la distancia del punto donde queda la hoja doblada a 100 en el eje Y

for (int Yi=100;Yi>p3y;Yi--){
//for para hacer la animacion de doblado hacia abajo, Yi se movera desde 100 a py3
sobre el eje Y
    int Xi=(((Yi-(100))*(p3x-(-100))/(p3y-(100)))+(-100));
    //el valor de x cuando Yi se va moviendo, este x esta en la recta que
    pasa por el (-100,100) y el p3
    if(Yi>100-(a/2)){ //cuando Yi se mueve antes de la mitad del doblez
    h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    //se dibuja la linea que simulara el doblado del lado izquierdo
    h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
    //se dibuja la linea que simulara el doblado del lado derecho
    try{ //pausa
    Thread.sleep(2) ;//10
    }catch(InterruptedException e){}
    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    //se borra la linea que del doblado en el lado izquierdo
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
    //se borra la linea que del doblado en el lado derecho
    }
    else{ //Yi va mas alla de la mitad del doblez hacia abajo
        if(Yi==(a/2)-100){//justo a la mitad del doblez hacia abajo

            try{ //pausa
            Thread.sleep(2) ;//2
            }catch(InterruptedException e){}

        }
        else{
        //Yi se mueve hacia abajo despues de pasar por la linea del doblez (la mitad)
            int X[]={p1.getX(),Xi,p2.getX()};
             //x de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            int Y[]={p1.getY(),Yi,p2.getY()};
             //y de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            h.dibujarPoligonoDeColor(Color.BLUE, 3, X, Y,h.getGraphics());
             //dibujando la hoja doblada

            try{ //pausa
            Thread.sleep(2); //3
```

```
            }catch(InterruptedException e){}
            /*int Xp[]={p2.getX(),X2,100};
            int Yp[]={p2.getY(),Y2,-100};
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());*/
        }
    }

}

for (int Yi=p3y;Yi<100;Yi++){
//for que hace la animacion hacia arriba, Yi va de p3y a 100 para cubrir
toda la distancia deldoblez a

    int Xi=(((Yi-(100))*(p3x-(-100))/(p3y-(100)))+(-100));
    //Xi es el valor de x ,cuando y vale Yi en la recta entre p3x,p3y y -100,100

    if(Yi<100-(a/2)){
    //entra en las iteraciones donde el doblez va hacia arriba y no ha pasado
    la mitad de la animacion

            int X[]={p1.getX(),Xi,p2.getX(),p3x};
            //Arreglo que guarda las x de los puntos del poligono que borra el
             triangulo que representa la hoja doblada
            int Y[]={p1.getY(),Yi,p2.getY(),p3y};
            //Arreglo que guarda las y de los puntos del poligono que borra el
            triangulo que representa la hoja doblada

            h.dibujarPoligonoDeColor(Color.white,4, X, Y, h.getGraphics());
            //el poligono que borra el triangulo, para simular el doblez

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

    }

    else{ //aqui entra cuando el doblez va mas alla dela mitad

        if(Yi==(a/2)-100){ //exactamente el doblez esta en la mitad
            h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
            //linea del doblez
            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }
        else{ //entra cuando la animacion va mas alla de la mitad hacia abajo
            h.dibujaLinea(p1.getX(), p1.getY(), p2.getX(), p2.getY(), h.getGraphics());
             //LINEA DEL DOBLEZ
            h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
            //extremo izquierdo de la hoja que se mueve en cada iteracion
            h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
             //extremo derecho de la hoja que se mueve en cada iteracion

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

            h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
             //se borra extremo izquierdo de la hoja que se mueve en cada iteracion
```

```
                h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
                //se borra extremo derecho de la hoja que se mueve en cada iteracion

                }
        }
    }
    h.refresh(h.getGraphics());
    //esta linea redibuja la hoja de papel con el doblez ya marcado
}
else{//este es el caso cuando y=100 y -100<x<100 ademas de que cuando y=-100 x>100
    if(xYm100>100){//entra cuando x>100 y y=-100
        cPunto p2=new cPunto(100,yX100);
         //se establece el segundo punto extremo (el punto derecho), siendo p1 el punto
              izquierdo
        CpuntoDoble p2Doble=new CpuntoDoble(100.0,yX100D);
         //mismo punto de arriba solo que en valores tipo double
        extremox2=100+h.getWidth()/2; //ajustando a la mitad del panel
        extremoy2=h.getHeight()/2-yX100; //ajustando a la mitad del panel
        //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
        cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
        //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

        ll.insertarAlInicio(nuevaLinea);
         //se inserta la nueva linea en la lista de lineas

        double ty=100.0-p2Doble.getY();
        //lado del triangulo que representa el doblez de la hoja
        double tx=100.0-p1Doble.getX();
        //el otro lado del triangulo que representa el doblez de la hoja
        double tanTetha=ty/tx; //tangente de la mitad del angulo de doblado con respecto a la
             horizontal
        double tetha=Math.toDegrees(Math.atan(tanTetha));
        //mitad del angulo de doblado con respecto a la horizontal en grados
        double alpha=90-(2.0*tetha); //angulo de dolado con respecto a la vertical
        double senAlpha=Math.sin(Math.toRadians(alpha));
        //seno del angulo de doblado en grados
        double x0=senAlpha*tx; // sen=CO/H CO=x0 H=tx
        int p3x=(int)(p1Doble.getX()+x0);
         //x del punto donde queda punta de la esquina derecha ya doblada
        double cosAlpha=Math.cos(Math.toRadians(alpha));
        //Coseno del anguo e doblez en graDOS
        double y0=cosAlpha*tx; // cos=CA/H CA=y0 H=tx
        int p3y=(int)(p1Doble.getY()-y0);
        //y del punto donde queda punta de la esquina derecha ya doblada

        int a=100-p3y;
        //"a" es la distancia del punto donde queda la hoja doblada a 100 en el eje Y

    for (int Yi=100;Yi>p3y;Yi--){
    //for para hacer la animacion de doblado hacia abajo, Yi se movera desde 100 a py3
    sobre el eje Y
        int Xi=(((Yi-(100))*(p3x-(100))/(p3y-(100)))+(100));
        //el valor de x cuando Yi se va moviendo, este x esta en la recta que pasa por
        el (100,100) y el p3
        if(Yi>100-(a/2)){//cuando Yi se mueve antes de la mitad del doblez
        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
         //se dibuja la linea que simulara el doblado del lado izquierdo
        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
        //se dibuja la linea que simulara el doblado del lado derecho
```

```
    try{ //pausa
    Thread.sleep(2); //10
    }catch(InterruptedException e){}
    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
    //se borra la linea que del doblado en el lado izquierdo
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
    //se borra la linea que del doblado en el lado derecho
    }
    else{ //Yi va mas alla de la mitad del doblez hacia abajo
        if(Yi==(a/2)-100){ //justo a la mitad del doblez hacia abajo

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

        }
        else{ //Yi se mueve hacia abajo despues de pasar por la linea del doblez (la mitad)
            int X[]={p1.getX(),Xi,p2.getX()};
            //x de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            int Y[]={p1.getY(),Yi,p2.getY()};
            //y de los puntos que dibujan el triangulo que simula la hoja de papel doblada
            h.dibujarPoligonoDeColor(Color.BLUE, 3, X, Y,h.getGraphics());
             //dibujando la hoja doblada

            try{//pausa
            Thread.sleep(2); //3
            }catch(InterruptedException e){}
            /*int Xp[]={p2.getX(),X2,100};
            int Yp[]={p2.getY(),Y2,-100};
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());*/
        }
    }
}

for (int Yi=p3y;Yi<100;Yi++){
//for que hace la animacion hacia arriba, Yi va de p3y a 100 para cubrir toda la
     distancia deldoblez a

    int Xi=(((Yi-(100))*(p3x-(100))/(p3y-(100)))+(100));
    //Xi es el valor de x ,cuando y vale Yi en la recta entre p3x,p3y y 100,100

    if(Yi<100-(a/2)){
    //entra en las iteraciones donde el doblez va hacia arriba y no ha pasado
    la mitad de la animacion

            int X[]={p1.getX(),Xi,p2.getX(),p3x};
             //Arreglo que guarda las x de los puntos del poligono que borra el
             triangulo que representa la hoja doblada
            int Y[]={p1.getY(),Yi,p2.getY(),p3y};
            //Arreglo que guarda las y de los puntos del poligono que borra el
            triangulo que representa la hoja doblada

            h.dibujarPoligonoDeColor(Color.white,4, X, Y, h.getGraphics());
            //el poligono que borra el triangulo, para simular el doblez

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
```

```
    }
    else{ //aqui entra cuando el doblez va mas alla dela mitad

        if(Yi==(a/2)-100){//exactamente el doblez esta en la mitad
            h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
            //linea del doblez
            try{//pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }
        else{ //entra cuando la animacion va mas alla de la mitad hacia abajo
            h.dibujaLinea(p1.getX(), p1.getY(), p2.getX(), p2.getY(), h.getGraphics());
            //LINEA DEL DOBLEZ
            h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
            //extremo izquierdo de la hoja que se mueve en cada iteracion
            h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //extremo derecho de la hoja que se mueve en cada iteracion

            try{ //pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}

            h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi, h.getGraphics());
            //se borra extremo izquierdo de la hoja que se mueve en cada iteracion
            h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi, h.getGraphics());
            //se borra extremo derecho de la hoja que se mueve en cada iteracion
            }
    }
}
h.refresh(h.getGraphics()); //esta linea redibuja la hoja de papel con el doblez ya
     marcado

    /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
    h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
    h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,extremox2,extremoy2);
    */
}
else{// este es el cuaso cuando y=100 -100<x<100 y ademas de que y=-100 y -100<x<100

    if(xYm100<xY100){
    //cuando el punto extremo izquierdo esta en y=-100 y el derecho cuando y=100
        p1Doble=new CpuntoDoble(xYm100D,-100.0);
        //punto extremo izquierdo, en valores tipo double
        p1= new cPunto(xYm100,-100);//el mismo punto de arriba solo que en valores enteros
        CpuntoDoble p2Doble= new CpuntoDoble(xY100D,100.0);
        //punto extremo derecho, en valores tipo double
        cPunto p2= new cPunto(xY100,100);
         //el mismo punto de arriba solo que en valores enteros
        extremox1=xYm100+h.getWidth()/2;
         //ajustando el punto izquierdo, en x, al centro del panel
        extremoy1=h.getHeight()/2+100;
        //ajustando el punto izquierdo, en y, al centro del panel
        extremox2=xY100+h.getWidth()/2;
         //ajustando el punto derecho, en x, al centro del panel
        extremoy2=h.getHeight()/2-100;
        //ajustando el punto derecho, en y, al centro del panel
        //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
        cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
         //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos
```

```
ll.insertarAlInicio(nuevaLinea);
 //se inserta la nueva linea en la lista de lineas
//se forman dos trapecios divididos por la linea del doblez
double r1=100.0+p1Doble.getX(); //lado inferior del del trapecio izquierdo
double r2=100.0+p2Doble.getX(); //lado superior del del trapecio izquierdo
double s1=200.0-r1; //lado inferior del del trapecio derecho
double s2=200.0-r2; //lado superior del del trapecio derechio

if(r1+r2>s1+s2){ //cuando el trapecio izquierdo es mas grande que el derecho
    double tanTetha=(p2Doble.getX()-p1Doble.getX())/200.0; //tan=CO/CA
         CO=(p2Doble.getX()-p1Doble.getX()) CA=ancho de la hoja
    double tetha=Math.toDegrees(Math.atan(tanTetha));
     //90-tehta es igual a la mitad del angulo de doblado con respecto a la
          horizontal
    double alpha=90.0-tetha; //mitad del angulo de doblado con respecto a la
         horizontal
    double betha=(2.0*alpha)-90.0; //angulo de doblado con respecto a la vertical
    double cosBetha=Math.cos(Math.toRadians(betha));
    //coseno del angulo de doblado en grados
    double y0=cosBetha*s1; //cos(angulo de doblado)= CA/H CA=y0 H=s1
    int p3y=(int)(p1Doble.getY()+y0);//"y" del punto donde queda la punta derecha
         ya doblada
    double senBetha=Math.sin(Math.toRadians(betha)); //coseno del angulo de
         doblado en grados
    double x0=senBetha*s1; //sen(angulo de doblado)= CO/H CO=x0 H=s1
    int p3x=(int)(p1Doble.getX()-x0); //"x" del punto donde queda la punta derecha
         ya doblada
    double y01=cosBetha*s2;//cos(angulo de doblado)= CA/H CA=y01 H=s2
    int p4y=(int)(p2Doble.getY()+y01);//"y" del punto donde queda la punta
    izquierda ya doblada
    double x01=senBetha*s2; //sen(angulo de doblado)= CO/H CO=x01 H=s2
    int p4x=(int)(p2Doble.getX()-x01);
     //"x" del punto donde queda la punta izquierda ya doblada

    int a=100-p3x;
    //distancia (en y) entre el punto de doblez izquierdo(p3) con el punto
    que se dobla (100,-100)
    int b=100-p4x;
     //distancia (en y) entre el punto de doblez derecho(p4) con el punto
     que se dobla(100,100)

    for (int Xi=100;Xi>p3x;Xi--){//for para hacer la animacion de doblado
    hacia la izquierda Xi se va a mover de Xi=100 a p3x
        int P=((100-Xi)*100/a); //porcentaje del recorrido de Xi ( regla de
         tres si a=100% entonces Xi=P? )
        int X2=100-(P*b/100); //valor del recorrido de X2 equivalente al
        porcentaje que ha recorrodo Xi( si 100%=b entonces P=>X2?)
        int Y2=(((X2-(100))*(p4y-(100))/(p4x-(100)))+(100)); //obteniendo el
        valor de "y" sobre la recta entre los puntos p4 y 100,100
        int Yi=(((Xi-(100))*(p3y-(-100))/(p3x-(100)))+(-100)); //obteniendo
        el valor de "y" sobre la recta entre los puntos p3 y 100,-100

        if(Xi>100-(a/2)){ //el doblez hacia abajo antes de la mitad
        //se dibuja un poligono con un los lados siguientes:
        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi,
        h.getGraphics());//lado (p1.getx,p1.gety)a(X2,Y2)
        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2,
         h.getGraphics());//lado (p2.getx,p2.gety)a(Xi,Yi)
```

```
        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
        //lado del poligono (Xi,Yi)a(X2,Y2)
        //el otro lado del poligono seria la linea que va de p1 a p2
        try{ //aqui se da una pausa para poder ver los cambios gradualmente
        y generar el efecto de movimiento
        Thread.sleep(2); //2
        }catch(InterruptedException e){}
        //despues de la pausa se borra el poligono que se dibujo anteriormente
        h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi,
             h.getGraphics());
        h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),X2,Y2,
             h.getGraphics());
        h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
        }
        else{//punto medio del doblez hacia la izquierda
            if(Xi==100-(a/2)){
                h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                 //se dibuja la linea que representa al papel doblado justo a la
                  mitad del dolbez
                try{
                //aqui se da una pausa para poder ver los cambios gradualmente
                y generar el efecto de movimiento
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
                h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
                //se borra la linea de la mitad del doblez
            }
            else{ //aqui continua el doblez hacia la izquierda depues de la
            mitad
                //se dibujara un poligono para simular la hoja que se dobla
                hacia la izquierda
                int X[]={p1.getX(),Xi,X2,p2.getX()};
                //arreglo que contiene las x de los puntos del poligono
                int Y[]={p1.getY(),Yi,Y2,p2.getY()};
                 //arreglo que contiene las y de los puntos del poligono
                h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());
                 //se dibuja el poligono con los arreglos

                try{ //aqui se da una pausa para poder ver los cambios gradualmente
                y generar el efecto de movimiento
                Thread.sleep(2); //2
                }catch(InterruptedException e){}
                //se dibuja un triangulo para borrar la parte del poligono que
                 debe desaparecer despues de cada iteracion

                int Xp[]={p2.getX(),X2,(100-((((100-(100-(a/2)))*100/a))*b/100))};
                //x de los puntos del triangulo, el ultimo x equivale a la hoja
                doblada en la parte superior cuando se esta a la mitad del doblez
                int
                     Yp[]={p2.getY(),Y2,(p2.getY()-p1.getY())*((100-((((100-(100-(a/2)))*
                100/a))*b/100))-p1.getX())/(p2.getX()-p1.getX())+(p1.getY())};
                //"y" de los puntos del triangulo, el ultimo "y" equivale a la hoja
                doblada en la parte superior cuando se esta a la mitad del doblez
                h.dibujarPoligonoDeColor(Color.white,3,Xp,Yp,h.getGraphics());
                //se dibuja el triangulo para borrar
            }
        }

    }
```

```
for (int Xi=p3x;Xi<100;Xi++){
//for para hacer el doblez hacia la derecha Xi va de p3x a 100
    int P=((100-Xi)*100/a);
     //porcentaje , que porcentaje de "a" equivale la distancia de 100 a Xi
    int X2=100-(P*b/100);
    //valor del recorrido de X2 equivalente al porcentaje que ha recorrido Xi
     si 100%=b entonces P=X2?
    int Y2=(((X2-(100))*(p4y-(100))/(p4x-(100)))+(100));
    //obteniendo el valor de "y" sobre la recta entre los puntos p4 y 100,100
    int Yi=((Xi-(100))*(p3y-(-100))/(p3x-(100)))+(-100);
    //obteniendo el valor de "y" sobre la recta entre los puntos p3 y 100,-100

    if(Xi<100-(a/2)){ //doblez hacia la derecha antes de la mitad

        int Xp[]={p1.getX(),p2.getX(),X2,Xi};
        //valores de x de los puntos del poligono que se dibuje cuando el doblez
         va hacia la derecha
        int Yp[]={p1.getY(),p2.getY(),Y2,Yi};
        //valores de x de los puntos del poligono que se dibuje cuando el doblez
        va hacia la derecha
        h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
        //poligono que simula el doblez hacia la derecha

        int X[]={p1.getX(),p3x,p4x,X2,Xi};
        //valores de x de los puntos del poligono que borraran el trapecio que
         se dibujo cuando el doblez era hacia la izquierda
        int Y[]={p1.getY(),p3y,p4y,Y2,Yi};
        //valores de y de los puntos del poligono que borraran el trapecio que
        se dibujo cuando el doblez era hacia la izquierda

        h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());

        if(Xi<-100){
        //si el doblez exede el margen izquierdo, se debe redibujar este margen
            h.dibujaLineaDeColor(Color.blue, -100,
                 -100,-100,(((-100-(Xi))*(p1.getY()-(Yi))/(p1.getX()-(Xi)))+(Yi)),
                 h.getGraphics());
            //redibujando margen izquierdo en la parte de abajo
            h.dibujaLineaDeColor(Color.blue, -100,
                 100,-100,(((-100-(Xi))*(Y2-(Yi))/(X2-(Xi)))+(Yi)),
                 h.getGraphics());
            //redibujando margen izquierdo en la parte de arriba
            }
        else{//cuando se juntan los dos lados, arriba y abajo
            h.dibujaLineaDeColor(Color.blue, -100, -100,
                 -100,100,h.getGraphics());//redibujando margen izquierdo
        }

        h.dibujaLineaDeColor(Color.blue, -100,
             100,((100-Yi)*(X2-Xi)/(Y2-Yi))+Xi,100,h.getGraphics());
        //linea que redibuja el margen superior
        try{//pausa para efectos de animacion
        Thread.sleep(2);//2
        }catch(InterruptedException e){}
    }

    else{
```

```
                if(Yi==100-(a/2)){//Yi esta en la mitad del doblez hacia la derecha
                    h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                    //se dibuja la linea del doblez
                    try{//pausa
                    Thread.sleep(2);//2
                    }catch(InterruptedException e){}

                }
                else{
                    //se dibujan las lineas que representan al papel doblandose hacia
                     la derecha despues de la mitad
                    h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2,
                         h.getGraphics());
                    h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi,
                         h.getGraphics());
                    h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                    try{
                    Thread.sleep(2);//2
                    }catch(InterruptedException e){}
                    //se borran las lineas que representan al papel doblandose hacia
                     la derecha despues de la mitad
                    h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),X2,Y2,
                         h.getGraphics());
                    h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi,
                         h.getGraphics());
                    h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
                }
            }
        }
        h.refresh(h.getGraphics());

        /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
        h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2 ,
        h.getHeight()/2-p3y);
        h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2 ,
        h.getHeight()/2-p4y);
        h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,
        p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
         */

        }
    else{
        //cuando el trapecio derecho es mas grande que el izquierdo
        double tanTetha=(p2Doble.getX()-p1Doble.getX())/200.0;
        //tan=CO/CA CO=p2x-p1x CA=ancho de la hoja
        double tetha=Math.toDegrees(Math.atan(tanTetha));
        //90-este Angulo=mitad del angulo de doblado on respecto a la horizontal
        double alpha=90.0-tetha;
        //mitad del angulo de doblado con respecto a la horizontal
        double betha=(2.0*alpha)-90.0;//angulo de doblado
        double cosBetha=Math.cos(Math.toRadians(betha));
        //coseno del angulo de doblado
        double y0=cosBetha*r1;//cos=CA/H CA=y0 H=r1
        int p3y=(int)(p1Doble.getY()-y0);
        //"y" del punto donde queda doblada la punta inferior izquierda
        double senBetha=Math.sin(Math.toRadians(betha));
        //seno del angulo de doblado
        double x0=senBetha*r1;//sen=CO/H CO=x0 H=r1
        int p3x=(int)(p1Doble.getX()+x0);
```

```
//"x" del punto donde queda doblada la punta inferior izquierda
double y01=cosBetha*r2;//cos=CA/H CA=y01 H=r2
int p4y=(int)(p2Doble.getY()-y01);
//"y" del punto donde queda doblada la punta superior izquierda
double x01=senBetha*r2;//sen=CO/H CO=x01 H=r2
int p4x=(int)(p2Doble.getX()+x01);
//"x" del punto donde queda doblada la punta superior izquierda

int b=p3x-(-100);//distancia en "x" de la punta izquierda inferior
 sin adoblar a la ya doblada , de px1 a px3
int a=p4x-(-100);//distancia en "x" de la punta izquierda superior
 sin adoblar a la ya doblada, de px2 a px4

for (int Xi=-100;Xi<(-100)+a;Xi++){//for que hace el doblado hacia
la derecha, recorriendo todo a desde px2 a px4
    int P=((Xi+100)*100/a);//porcentade que ha recorrido Xi dento de a
    int X2=(P*b/100)-100;//recorrido equivalente al porcentaje que
    lleva Xi, solo dentro de b
    int Yi=(((Xi-(-100))*(p4y-(100))/(p4x-(-100)))+(100));
    //"y" correspondiente a Xi dentro de la recta que une p2 y p4
    int Y2=(((X2-(-100))*(p3y-(-100))/(p3x-(-100)))+(-100));
    //"y" correspondiente a X2 dentro de la recta que une p1 y p3

    if(Xi<(a/2)-100){
    //durante el doblez hacia la deracha antes de la mitad de este
        //las siguentes lineas dibujan la hoja retrayendose hacia la
         derecha en cada iteracion
        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi,
        h.getGraphics());
        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2,
         h.getGraphics());
        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());


        try{//pasua para efectos de animacion
        Thread.sleep(2);//10
        }catch(InterruptedException e){}
        //las siguentes borran la hoja en cada iteracion para simular
         movimiento
        h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi,
         h.getGraphics());
        h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),X2,Y2,
         h.getGraphics());
        h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
    }
    else{ //cuando el doblez esta sobrepasando la mitad de su recorrido
    a la derecha
        if(Xi==(a/2)-100){//justo a la mitad del recorrido
            try{//pausa
            Thread.sleep(2); //2
            }catch(InterruptedException e){}
        }
        else{//el doblez ha superado la mitad del recorrido hacia la derecha
            int X[]={p1.getX(),X2,Xi,p2.getX()}; //"x" de los puntos que
             dibujaran el poligono que simula la hoja doblada
            int Y[]={p1.getY(),Y2,Yi,p2.getY()}; //"y" de los puntos que
            dibujaran el poligono que simula la hoja doblada
            h.dibujarPoligonoDeColor(Color.BLUE, 4, X,
                 Y,h.getGraphics());//poligono(trapecio) que simula del doblez
```

```
            de la hoja

            try{//pausa
            Thread.sleep(2);//3
            }catch(InterruptedException e){}

            int Xp[]={p1.getX(),X2,(((((a/2)-100)+100)*100/a)*b/100)-100};
            //x del triangulo que borra parte del trapecio que desaparece en
             cada iteracion
            int Yp[]={p1.getY(),Y2,(p2.getY()-p1.getY())*(((((((a/2)-100)+100)*
            100/a)*b/100)-100)-p1.getX())/(p2.getX()-p1.getX())+(p1.getY())};
            //y del triangulo que borra parte del trapecio que desaparece en
             cada iteracion
            h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());
            //tiangulo que borra parte del trapecio que debe desaparecer en
            cada iteracion
            }
        }
}

for (int Xi=(-100)+a;Xi>-100;Xi--){
//for que hace la animacion de doblado
hacia la izquierda, de regreso Xi recorre toda la distancia "a"
    int P=((Xi+100)*100/a);
    //porcentade que ha recorrido Xi dento de a
    int X2=(P*b/100)-100;
    //recorrido equivalente al porcentaje que lleva Xi, solo dentro de b
    int Yi=(((Xi-(-100))*(p4y-(100))/(p4x-(-100)))+(100));
    //"y" correspondiente a Xi dentro de la recta que une p2 y p4
    int Y2=(((X2-(-100))*(p3y-(-100))/(p3x-(-100)))+(-100));
    //"y" correspondiente a X2 dentro de la recta que une p1 y p3

    if(Xi>(a/2)-100){ //antes de la mitad del recorrido a la izquierda

        int X[]={X2,p3x,p4x,p2.getX(),Xi};
        //"x" de los puntos del poligono que borra el trapecio dibujado
         cuando el doblez iba a la derecha
        int Y[]={Y2,p3y,p4y,p2.getY(),Yi};
        //"y" de los puntos del poligono que borra el trapecio dibujado
         cuando el doblez iba a la derecha

        int Xp[]={p1.getX(),p2.getX(),Xi,X2};
        //"x" de los puntos del trapecio que simula del doblez de la hoja
        hacia la izquierda
        int Yp[]={p1.getY(),p2.getY(),Yi,Y2};
        //"y" de los puntos del trapecio que simula del doblez de la hoja
        hacia la izquierda

        h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());
        //poligono que borra la hoja doblada en cada iteracion
        if(Xi>99){//si el doblez rebasa el margen izquierdo , se debe redibujar
            h.dibujaLineaDeColor(Color.blue, 100,
                 100,100,(Y2-Yi)*(100-Xi)/(X2-Xi)+(Yi), h.getGraphics());
            //redibujando margen izquierdo en la parte de arriba
            h.dibujaLineaDeColor(Color.blue, 100,
                 -100,100,(p2.getY()-Yi)*(100-Xi)/(p2.getX()-Xi)+(Yi) ,
            h.getGraphics());
            //redibujando margen izquierdo en la parte de abajo
        }
```

```
                    else{//cuando se juntan
                        h.dibujaLineaDeColor(Color.blue,100,-100,100,100,h.getGraphics());
                            //redibujando margen derecho
                    }

                    h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
                     //dibujando hoja que simula el doblez hacia la izquierda

                    h.dibujaLineaDeColor(Color.blue, 100,
                         -100,((-100)-Yi)*(X2-Xi)/(Y2-Yi)+(Xi),-100,h.getGraphics());
                    //redibujando margen inferior
                    try{
                    Thread.sleep(2);//2
                    }catch(InterruptedException e){}
                }

                else{ //cuando el doblez esta pasando despues de la mitad de su recorrido

                    if(Yi==(a/2)-100){
                    //el recorrido hacia la izquierda esta exactamente a la mitad
                        h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
                        //liena de doblez
                        try{//pausa
                        Thread.sleep(2);//2
                        }catch(InterruptedException e){}
                    }
                    else{//despues de la mitad del recorrido hacia la izquierda
                        //las siguentes lineas dibujan al papel desplazandose a la
                        izquierda en cada iteracion
                        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2,
                            h.getGraphics());
                        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi,
                            h.getGraphics());
                        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                        try{
                        Thread.sleep(2); //2
                        }catch(InterruptedException e){}
                        //para simular movimiento se borran las lineas antes dibujadas
                        h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),X2,Y2,
                            h.getGraphics());
                        h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi,
                            h.getGraphics());
                        h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
                    }
                }
            }
            h.refresh(h.getGraphics());
            //esta linea redibuja la hoja de papel con el doblez ya marcado
            /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
            h.getGraphics().drawLine(p1.getX()+h.getWidth()/2,h.getHeight()/2-p1.getY(),
            p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
            h.getGraphics().drawLine(p2.getX()+h.getWidth()/2,h.getHeight()/2-p2.getY(),
            p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
            h.getGraphics().drawLine(p3x+h.getWidth()/2
                 ,h.getHeight()/2-p3y,p4x+h.getWidth()/2
             ,h.getHeight()/2-p4y);
            */
        }
```

```
}
else{
//cuando el punto extremo derecho esta en y=-100 y el izquierdo cuando y=100
    p1Doble=new CpuntoDoble(xY100D,100.0);
    //extremo izquierdo de la linea de doblez
    p1= new cPunto(xY100,100); //mismo punto izquierdo solo que en valores enteros
    CpuntoDoble p2Doble= new CpuntoDoble(xYm100D,-100.0);
    //extremo derecho de la linea de doblez
    cPunto p2= new cPunto(xYm100,-100);
    //mismo punto dercho solo que en valores enteros
    extremox1=xY100+h.getWidth()/2; //ajustando el punto a la mitad del panel
    extremoy1=h.getHeight()/2-100; //ajustando el punto a la mitad del panel
    extremox2=xYm100+h.getWidth()/2; //ajustando el punto a la mitad del panel
    extremoy2=h.getHeight()/2+100; //ajustando el punto a la mitad del panel
    //h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
    cLinea nuevaLinea=new cLinea(p1Doble,p2Doble);
    //se crea la nueva linea(linea de doblez) en base a los dos puntos extremos

    ll.insertarAlInicio(nuevaLinea);
    //se inserta la nueva linea en la lista de lineas
    //se divide la hoja en dos trapecios
    double r1=100.0+p1Doble.getX(); //lado superior del trapecio izquierdo
    double r2=100.0+p2Doble.getX(); //lado inferior del trapecio izquierdo
    double s1=200.0-r1;//lado superior del trapecio derecho
    double s2=200.0-r2;//lado inferior del trapecio derecho

    if(r1+r2>s1+s2){//cuando el trapecio izquierdo es mas grande
        double tanTetha=(p2Doble.getX()-p1Doble.getX())/200.0;
        //tan=CO/CA  CO=distancia en x entre los puntos extermos CA=largo de la hoja
        double tetha=Math.toDegrees(Math.atan(tanTetha));
        //90-tetha=mitad angulo de doblado con respecto a la horizontal
        double alpha=90.0-tetha;
        //mitad del angulo de doblado con respecto a la horizontal
        double betha=(2.0*alpha)-90.0; //angulo de doblado
        double cosBetha=Math.cos(Math.toRadians(betha));
         //coseno del angulo de doblado en grados
        double y0=cosBetha*s1; //cos=CA/H CA=y0 H=s1
        int p3y=(int)(p1Doble.getY()-y0);
        // "y" del punto donde queda doblada la esquina superior izquierda
        double senBetha=Math.sin(Math.toRadians(betha));
        //seno del angulo de doblado en grados
        double x0=senBetha*s1; // sen=CO/H CO=x0 H=s1
        int p3x=(int)(p1Doble.getX()-x0);
        //"x" del punto donde queda doblada la esquina superior izquierda
        double y01=cosBetha*s2; //cos=CA/H CA=y01 H=s2
        int p4y=(int)(p2Doble.getY()-y01);
        // "y" del punto donde queda doblada la esquina inferior izquierda
        double x01=senBetha*s2;// sen=CO/H CO=x01 H=s2
        int p4x=(int)(p2Doble.getX()-x01);
        // "x" del punto donde queda doblada la esquina inferior izquierda

        int a=100-p3x;
        //distancia (en y) entre el punto de doblez izquierdo(p3) con el
        punto que se dobla (100,-100)
        int b=100-p4x;
        //distancia (en y) entre el punto de doblez derecho(p4) con el
        punto que se dobla(100,100)

        for (int Xi=100;Xi>p3x;Xi--){
```

```
//for para hacer la animacion de doblado hacia la izquierda Xi se
va a mover de Xi=100 a p3x
    int P=((100-Xi)*100/a);
    //porcentaje del recorrido de Xi ( regla de tres si a=100%
    entonces Xi=P? )
    int X2=100-(P*b/100);
    //valor del recorrido de X2 equivalente al porcentaje que ha
    recorrodo Xi( si 100%=b entonces P=>X2?)
    int Y2=(((X2-(100))*(p4y-(-100))/(p4x-(100)))+(-100));
    //obteniendo el valor de "y" sobre la recta entre los puntos
    p4 y 100,100
    int Yi=(((Xi-(100))*(p3y-(100))/(p3x-(100)))+(100));
    //obteniendo el valor de "y" sobre la recta entre los puntos
     p3 y 100,-100

    if(Xi>100-(a/2)){//el doblez hacia abajo antes de la mitad
    //se dibuja un poligono con un los lados siguientes:
    h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi,
     h.getGraphics());//lado (p1.getx,p1.gety)a(X2,Y2)
    h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2,
     h.getGraphics());//lado (p2.getx,p2.gety)a(Xi,Yi)
    h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
    //lado del poligono (Xi,Yi)a(X2,Y2)
    //el otro lado del poligono seria la linea que va de p1 a p2
    try{
    //aqui se da una pausa para poder ver los cambios gradualmente y
     generar el efecto de movimiento
    Thread.sleep(2);//2
    }catch(InterruptedException e){}
    //despues de la pausa se borra el poligono que se dibujo
    anteriormente
    h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),Xi,Yi,
     h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),X2,Y2,
    h.getGraphics());
    h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
    }
    else{ //punto medio del doblez hacia la izquierda
        if(Xi==100-(a/2)){
            h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2,
            h.getGraphics());//se dibuja la linea que representa al papel
            doblado justo a la mitad del dolbez
            try{ //aqui se da una pausa para poder ver los cambios
            gradualmente y generar el efecto de movimiento
            Thread.sleep(2);//2
            }catch(InterruptedException e){}
            h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
            //se borra la linea de la mitad del doblez
        }
        else{ //aqui continua el doblez hacia la izquierda despues de la
        mitad de este
            //se dibujara un poligono para simular la hoja que se dobla
             hacia la izquierda
            int X[]={p1.getX(),Xi,X2,p2.getX()};
            //arreglo que contiene las x de los puntos del poligono
            int Y[]={p1.getY(),Yi,Y2,p2.getY()};
             //arreglo que contiene las y de los puntos del poligono
            h.dibujarPoligonoDeColor(Color.BLUE, 4, X, Y,h.getGraphics());
            //se dibuja el poligono con los arreglos
```

```
            try{//aqui se da una pausa para poder ver los cambios
            gradualmente y generar el efecto de movimiento
            Thread.sleep(2);//2
            }catch(InterruptedException e){}
            //se dibuja un triangulo para borrar la parte del poligono
            que debe desaparecer despues de cada iteracion

            int Xp[]={p2.getX(),X2,(100-((((100-(100-(a/2)))*100/a))*b/100))};
             //x de los puntos del triangulo, el ultimo x equivale a la hoja
             doblada en la parte superior cuando se esta a la mitad del doblez
            int
                 Yp[]={p2.getY(),Y2,(p2.getY()-p1.getY())*((100-((((100-(100-(a/2)))*
            100/a))*b/100))-p1.getX())/(p2.getX()-p1.getX())+(p1.getY())};//"y"
                 de
            los puntos del triangulo, el ultimo "y" equivale a la hoja doblada
                 en
            la parte superior cuando se esta a la mitad del doblez
            h.dibujarPoligonoDeColor(Color.white,3,Xp,Yp,h.getGraphics());
            //se dibuja el triangulo para borrar
        }
    }

}

for (int Xi=p3x;Xi<100;Xi++){ //for para hacer el doblez hacia la derecha Xi
va de p3x a 100
    int P=((100-Xi)*100/a);
    //que porcentaje de "a" equivale la distancia de 100 a Xi
    int X2=100-(P*b/100);
     //valor del recorrido de X2 equivalente al porcentaje que ha recorrido
     Xi( si 100%=b entonces P=X2?)
    int Y2=(((X2-(100))*(p4y-(-100))/(p4x-(100)))+(-100));
    //obteniendo el valor de "y" sobre la recta entre los puntos p4 y 100,100
    int Yi=((Xi-(100))*(p3y-(100))/(p3x-(100)))+(100);
    //obteniendo el valor de "y" sobre la recta entre los puntos p3 y 100,-100

    if(Xi<100-(a/2)){ //doblez hacia la derecha antes de la mitad

        int Xp[]={p1.getX(),p2.getX(),X2,Xi};
         //valores de x de los puntos del poligono que se dibuje cuando el
         doblez va hacia la derecha
        int Yp[]={p1.getY(),p2.getY(),Y2,Yi};
        //valores de x de los puntos del poligono que se dibuje cuando el
         doblez va hacia la derecha
        h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
        //poligono que simula el doblez hacia la derecha

        int X[]={p1.getX(),p3x,p4x,X2,Xi};
        //valores de x de los puntos del poligono que borraran el trapecio
        que se dibujo cuando el doblez era hacia la izquierda
        int Y[]={p1.getY(),p3y,p4y,Y2,Yi};
         //valores de y de los puntos del poligono que borraran el trapecio
         que se dibujo cuando el doblez era hacia la izquierda

        h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());

        if(Xi<-100){
        //si el doblez exede el margen izquierdo, se debe redibujar este
```

```
                margen
                   h.dibujaLineaDeColor(Color.blue, -100,
                        100,-100,(((-100-(Xi))*(p1.getY()-(Yi))/(p1.getX()-(Xi)))+(Yi)),
                        h.getGraphics());
                   //redibujando margen izquierdo en la parte de abajo
                   h.dibujaLineaDeColor(Color.blue, -100,
                        -100,-100,(((-100-(Xi))*(Y2-(Yi))/(X2-(Xi)))+(Yi)),
                    h.getGraphics());
                   //redibujando margen izquierdo en la parte de arriba
                   }
               else{//cuando se juntan los dos lados, arriba y abajo
                   h.dibujaLineaDeColor(Color.blue, -100, -100, -100,100,
                   h.getGraphics()); //redibujando margen izquierdo
               }

               h.dibujaLineaDeColor(Color.blue, -100,
                    -100,((-100-Yi)*(X2-Xi)/(Y2-Yi))+Xi,-100,h.getGraphics());
               //linea que redibuja el margen inferior
               try{//pausa para efectos de animacion
               Thread.sleep(2);//2
               }catch(InterruptedException e){}
           }

           else{

               if(Yi==100-(a/2)){ //Yi esta en la mitad del doblez hacia la
                derecha
                   h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,
                   extremoy2);//se dibuja la linea del doblez
                   try{//pausa
                   Thread.sleep(2); //2
                   }catch(InterruptedException e){}

               }
               else{
                   //se dibujan las lineas que representan al papel doblandose
                    hacia la derecha despues de la mitad
                   h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),X2,Y2,
                        h.getGraphics());
                   h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),Xi,Yi,
                        h.getGraphics());
                   h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
                   try{
                   Thread.sleep(2);//2
                   }catch(InterruptedException e){}
                   //se borran las lineas que representan al papel doblandose
                   hacia la derecha despues de la mitad
                   h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),X2,Y2,
                        h.getGraphics());
                   h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),Xi,Yi,
                        h.getGraphics());
                   h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
               }
           }
       }
       h.refresh(h.getGraphics());

       /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
       h.getGraphics().drawLine(extremox1,extremoy1,p3x+h.getWidth()/2
```

```
    ,h.getHeight()/2-p3y);
    h.getGraphics().drawLine(extremox2,extremoy2,p4x+h.getWidth()/2
    ,h.getHeight()/2-p4y);
    h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y
    ,p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
     *
     */
    }
else{//cuando el trapecio derecho es mas grande
    double tanTetha=(p2Doble.getX()-p1Doble.getX())/200.0;
    //tan=CO/CA CO=p2x-p1x CA=ancho de la hoja
    double tetha=Math.toDegrees(Math.atan(tanTetha));
    //90-este Angulo=mitad del angulo de doblado on respecto a la horizontal
    double alpha=90.0-tetha; //mitad del angulo de doblado con
    respecto a la horizontal
    double betha=(2.0*alpha)-90.0;
    //angulo de doblado
    double cosBetha=Math.cos(Math.toRadians(betha));
     //coseno del angulo de doblado
    double y0=cosBetha*r1;//cos=CA/H CA=y0 H=r1
    int p3y=(int)(p1Doble.getY()+y0);
    //"y" del punto donde queda doblada la punta superior izquierda
    double senBetha=Math.sin(Math.toRadians(betha));
    //seno del angulo de doblado
    double x0=senBetha*r1;//sen=CO/H CO=x0 H=r1
    int p3x=(int)(p1Doble.getX()+x0);//"x" del punto donde queda doblada la
    punta superior izquierda
    double y01=cosBetha*r2;//cos=CA/H CA=y01 H=r2
    int p4y=(int)(p2Doble.getY()+y01);//"y" del punto donde queda doblada la
     punta inferior izquierda
    double x01=senBetha*r2;//sen=CO/H CO=x01 H=r2
    int p4x=(int)(p2Doble.getX()+x01);//"x" del punto donde queda doblada la
     punta inferior izquierda

    int b=p3x-(-100);//distancia en "x" de la punta izquierda inferior sin
    adoblar a la ya doblada , de px1 a px3
    int a=p4x-(-100);//distancia en "x" de la punta izquierda superior sin
     adoblar a la ya doblada, de px2 a px4

    for (int Xi=-100;Xi<(-100)+a;Xi++){ //for que hace el doblado hacia la
     derecha, recorriendo todo a desde px2 a px4
       int P=((Xi+100)*100/a); //porcentade que ha recorrido Xi dento de a
       int X2=(P*b/100)-100; //recorrido equivalente al porcentaje que lleva
       Xi, solo dentro de b
       int Yi=(((Xi-(-100))*(p4y-(-100))/(p4x-(-100)))+(-100));
       //"y" correspondiente a Xi dentro de la recta que une p2 y p4
       int Y2=(((X2-(-100))*(p3y-(100))/(p3x-(-100)))+(100));
       //"y" correspondiente a X2 dentro de la recta que une p1 y p3

       if(Xi<(a/2)-100){
       //durante el doblez hacia la deracha antes de la mitad de este
           //las siguentes lineas dibujan la hoja retrayendose hacia la
           derecha en cada iteracion
           h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi,
           h.getGraphics());
           h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2,
            h.getGraphics());
           h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
```

```
            try{//pasua para efectos de animacion
            Thread.sleep(2);//10
            }catch(InterruptedException e){}
            //las siguentes borran la hoja en cada iteracion para simular
            movimiento
            h.dibujaLineaDeColor(Color.WHITE,p2.getX(),p2.getY(),Xi,Yi,
             h.getGraphics());
            h.dibujaLineaDeColor(Color.WHITE,p1.getX(),p1.getY(),X2,Y2,
            h.getGraphics());
            h.dibujaLineaDeColor(Color.WHITE,Xi,Yi,X2,Y2, h.getGraphics());
        }
        else{//cuando el doblez esta sobrepasando la mitad de su
        recorrido a la derecha
            if(Xi==(a/2)-100){//justo a la mitad del recorrido
                try{//pausa
                Thread.sleep(2);//2
                }catch(InterruptedException e){}
            }
            else{//el doblez ha superado la mitad del recorrido hacia
             la derecha
                int X[]={p1.getX(),X2,Xi,p2.getX()};//"x" de los puntos
                 que dibujaran el poligono que simula la hoja doblada
                int Y[]={p1.getY(),Y2,Yi,p2.getY()};//"y" de los puntos
                que dibujaran el poligono que simula la hoja doblada
                h.dibujarPoligonoDeColor(Color.BLUE, 4, X,
                     Y,h.getGraphics());//poligono(trapecio) que simula del
                doblez de la hoja

                try{//pausa
                Thread.sleep(2);//3
                }catch(InterruptedException e){}

                int Xp[]={p1.getX(),X2,(((((a/2)-100)+100)*100/a)*b/100)-
                100};//x del triangulo que borra parte del trapecio que
                desaparece en cada iteracion
                int Yp[]={p1.getY(),Y2,(p2.getY()-p1.getY())*(((((((a/2)-100)+
                100)*100/a)*b/100)-100)-p1.getX())/(p2.getX()-p1.getX())+(p1.getY())};
                //y del triangulo que borra parte del trapecio que desaparece
                 en cada iteracion
                h.dibujarPoligonoDeColor(Color.white,3, Xp, Yp,h.getGraphics());
                //tiangulo que borra parte del trapecio que debe desaparecer en
                 cada iteracion
                }
            }
    }
    for (int Xi=(-100)+a;Xi>-100;Xi--){
    //for que hace la animacion de doblado hacia la izquierda, de regreso Xi
    recorre toda la distancia "a"
        int P=((Xi+100)*100/a);//porcentade que ha recorrido Xi dento de a
        int X2=(P*b/100)-100;
        //recorrido equivalente al porcentaje que lleva Xi, solo dentro de b
        int Yi=(((Xi-(-100))*(p4y-(-100))/(p4x-(-100)))+(-100));
        //"y" correspondiente a Xi dentro de la recta que une p2 y p4
        int Y2=(((X2-(-100))*(p3y-(100))/(p3x-(-100)))+(100));
        //"y" correspondiente a X2 dentro de la recta que une p1 y p3

        if(Xi>(a/2)-100){//antes de la mitad del recorrido a la izquierda
            int X[]={X2,p3x,p4x,p2.getX(),Xi};
            //"x" de los puntos del poligono que borra el trapecio dibujado
```

```
     cuando el doblez iba a la derecha
    int Y[]={Y2,p3y,p4y,p2.getY(),Yi};
    //"y" de los puntos del poligono que borra el trapecio dibujado
     cuando el doblez iba a la derecha

    int Xp[]={p1.getX(),p2.getX(),Xi,X2};
    //"x" de los puntos del trapecio que simula del doblez de la
    hoja hacia la izquierda
    int Yp[]={p1.getY(),p2.getY(),Yi,Y2};//"y" de los puntos del
    trapecio que simula del doblez de la hoja hacia la izquierda

    h.dibujarPoligonoDeColor(Color.white,5, X, Y, h.getGraphics());
    //poligono que borra la hoja doblada en cada iteracion
    if(Xi>99){//si el doblez rebasa el margen izquierdo , se debe
    redibujar
        h.dibujaLineaDeColor(Color.blue, 100,
             100,100,(Y2-Yi)*(100-Xi)/(X2-Xi)+(Yi), h.getGraphics());
        //redibujando margen izquierdo en la parte de arriba
        h.dibujaLineaDeColor(Color.blue, 100,
             -100,100,(p2.getY()-Yi)*(100-Xi)/(p2.getX()-Xi)+(Yi)
             ,h.getGraphics());//redibujando margen izquierdo en la
         parte de abajo
    }
    else{//cuando se juntan
        h.dibujaLineaDeColor(Color.blue,100,-100,100,100,h.getGraphics());
        //redibujando margen derecho
    }
    h.dibujarPoligonoDeColor(Color.blue,4,Xp,Yp,h.getGraphics());
    //dibujando hoja que simula el doblez hacia la izquierda

    h.dibujaLineaDeColor(Color.blue, 100,
         100,((100)-Yi)*(X2-Xi)/(Y2-Yi)+(Xi),100,h.getGraphics());
    //redibujando margen superior
    try{
    Thread.sleep(2);//2
    }catch(InterruptedException e){}
}
else{//cuando el doblez esta pasando despues de la mitad de su recorrido

    if(Yi==(a/2)-100){
    //el recorrido hacia la izquierda esta exactamente a la mitad
        h.getGraphics().drawLine(extremox1,extremoy1,extremox2 ,extremoy2);
        //liena de doblez
        try{//pausa
        Thread.sleep(2); //2
        }catch(InterruptedException e){}
    }
    else{//despues de la mitad del recorrido hacia la izquierda
        //las siguentes lineas dibujan al papel desplazandose a la
         izquierda en cada iteracion
        h.dibujaLineaDeColor(Color.blue,p1.getX(),p1.getY(),X2,Y2,
             h.getGraphics());
        h.dibujaLineaDeColor(Color.blue,p2.getX(),p2.getY(),Xi,Yi,
             h.getGraphics());
        h.dibujaLineaDeColor(Color.blue,Xi,Yi,X2,Y2, h.getGraphics());
        try{
        Thread.sleep(2);//2
        }catch(InterruptedException e){}
        //para simular movimiento se borran las lineas antes dibujadas
```

```
                                    h.dibujaLineaDeColor(Color.white,p1.getX(),p1.getY(),X2,Y2,
                                        h.getGraphics());
                                    h.dibujaLineaDeColor(Color.white,p2.getX(),p2.getY(),Xi,Yi,
                                        h.getGraphics());
                                    h.dibujaLineaDeColor(Color.white,Xi,Yi,X2,Y2, h.getGraphics());
                                }
                            }
                        }
                        h.refresh(h.getGraphics());
                        //esta linea redibuja la hoja de papel con el doblez ya marcado

                        /*Las siguientes lineas dibujan como quedaria el doblez sin animacion
                        h.getGraphics().drawLine(p1.getX()+h.getWidth()/2,h.getHeight()/2-p1.getY(),
                        p3x+h.getWidth()/2 ,h.getHeight()/2-p3y);
                        h.getGraphics().drawLine(p2.getX()+h.getWidth()/2,h.getHeight()/2-p2.getY(),
                        p4x+h.getWidth()/2 ,h.getHeight()/2-p4y);
                        h.getGraphics().drawLine(p3x+h.getWidth()/2 ,h.getHeight()/2-p3y,p4x+
                        h.getWidth()/2 ,h.getHeight()/2-p4y);
                         *
                         */
                    }
                }
            }
        }
    }
    else{
        ventanaError v=new ventanaError("Al menos uno de estos puntos no ha sido introducido",
        "Prueba introduciendo el punto");
        v.setVisible(true);
    }
  }
    }
}
```

A.11. ecuacionParabola

```
package sahco;
public class ecuacionParabola {
    private double p;
    private double q;
    private double r;

    ecuacionParabola(double P,double Q, double R){
        p=P;
        q=Q;
        r=R;
    }

    public double getP(){
        return p;
    }

    public double getQ(){
        return q;
    }
```

```
    public double getR(){
        return r;
    }
}
```

A.12. gestionA

```
package sahco;
import java.io.*;
public class gestionA {
    FileInputStream entrada;
    FileOutputStream salida;
    gestionA(){
    }

    public String abrirATexto(File archivo){
        String contenido="";
        try{
            entrada= new FileInputStream(archivo);
            int ascii;
            while((ascii=entrada.read())!=-1){
            char caracter=(char)ascii;
            contenido+= caracter;
            }
        }catch (Exception e){
        }
        return contenido;
    }

    public String guardarArchivo(File archivo, String contenido){
        String respuesta=null;
        try{
            salida = new FileOutputStream(archivo);
            byte[] byteText=contenido.getBytes();
            salida.write(byteText);
            respuesta= "se guardo el archivo";
        }catch (Exception e){

        }
        return respuesta;
    }
}
```

A.13. lineaDobladoSinAnimacion

```
package sahco;
import java.awt.Color;

public class lineaDobladoSinAnimacion {
    private listaPuntos lp;//variable de clase que guarda la lista de puntos
    private panelPrincipal h;
    //variable de clase que guarda los parametros del panel principal
    private listaLineas ll;//variable de clase que guarda la lista de plineas
```

```
private cPunto p1;
//variable de calse que guarda el primer punto de la linea de doblez
private cPunto p2;
//variable de calse que guarda el segundo punto de la linea de doblez

private CpuntoDoble p1D;
//variable de clase que guarda el primer punto , con valores tipo doble
private CpuntoDoble p2D;
//variable de clase que guarda el segundo punto , con valores tipo doble

lineaDobladoSinAnimacion(listaPuntos LP, listaLineas LL,panelPrincipal H){
//constructor de la clase
    lp=LP;//se asigna la lista de puntos
    ll=LL;//se asigna la lista de lineas
    h=H;//se asigna el panel
}

public cLinea trazarLineaDeDoblado(double x1,double y1,double x2,double y2){
//metodo que devuelve una linea de doblado a partir de dos puntos
    if(x1==x2||y1==y2){//cuando son rectas de 90 y 0 grados
        if(x1==x2){//si es una recta vertical
            return new cLinea(new CpuntoDoble(x1,(double)-100.0),new CpuntoDoble(x1,
            (double)100.0));
        }
        else{//recta horizontal
            return new cLinea(new CpuntoDoble((double)-100.0,y1),new CpuntoDoble(
            (double)100.0,y1));
        }
    }
    else{//cuando son rectas inclinadas
        int xY100=(int)(((100.0-y1)*(x2-x1)/(y2-y1))+x1);
        int yX100=(int)(((100.0-x1)*(y2-y1)/(x2-x1))+y1);
        int xYm100=(int)(((-100.0-y1)*(x2-x1)/(y2-y1))+x1);
        int yXm100=(int)(((-100.0-x1)*(y2-y1)/(x2-x1))+y1);

        double xY100D=((((double)100.0-y1)*(x2-x1)/(y2-y1))+x1);//el valor de "x"
        (en doble) cuando "y" vale 100 en la recta del doblez
        double yX100D=((((double)100.0-x1)*(y2-y1)/(x2-x1))+y1);//el valor de "y"
         (en doble) cuando "x" vale 100 en la recta del doblez
        double xYm100D=((((double)-100.0-y1)*(x2-x1)/(y2-y1))+x1);//el valor de "x"
        (en doble) cuando "y" vale -100 en la recta del doblez
        double yXm100D=((((double)-100.0-x1)*(y2-y1)/(x2-x1))+y1);//el valor de "y"
         (en doble) cuando "x" vale -100 en la recta del doblez

        if(xY100<-100){//este es el caso cuando x<-100 si y=100
            p1=new cPunto(-100,yXm100);//este es el extremo izquierdo del doblez es
             x=-100 y y=f(-100) f es la funcion de la linea entre los puntos dibujados
            p1D=new CpuntoDoble((double)-100.0,yXm100D);//mismo punto de arriba solo
             que en valores tipo doble
            if(xYm100>100){
                p2=new cPunto(100,yX100); //extremo derecho de la linea x=100 y "y"
                 cuando x vale 100 en la linea de doblez
                p2D=new CpuntoDoble((double)100.0,yX100D);//mismo punto de arriba solo
                 que en valores tipo doble
            }
            else{
            p2=new cPunto(xYm100,-100);//este es el extremo derecho de la linea de
            doblez x=f(-100) y y=-100
```

```
    p2D=new CpuntoDoble(xYm100D,(double)-100.0);//mismo punto de arriba solo
     que en valores tipo doble
    }
}
else{//se mete a este else cuando x>=-100 cuando y=100

if(xY100>100){//se mete al if cuando x>100 cuando y=100
    p1=new cPunto(100,yX100);//este es el extremo derecho de la linea de doblez
    p1D=new CpuntoDoble((double)100.0,yX100D);//mismo punto de arriba solo
    que en valores tipo doble
    if(xYm100<-100){//si de lado izquerdo x<-100 cuando y=-100
        p2=new cPunto(-100,yXm100);//este es el extremo izquierdo de la linea
         de doblez
        p2D=new CpuntoDoble((double)-100.0,yXm100D);//mismo punto de arriba
         solo que en valores tipo doble
    }
    else{
        p2=new cPunto(xYm100,-100);//este es el punto izquierdo
        p2D=new CpuntoDoble(xYm100D,(double)-100.0);//mismo punto de arriba
        solo que en valores tipo doble
    }
}
else{// -100<x<100 cuando y=100
    p1=new cPunto(xY100,100);//se establece el primer punto extremo
    p1D=new CpuntoDoble(xY100D,(double)100.0);//mismo punto de arriba solo
    que en valores tipo doble
    if(xYm100<-100){//se mete a heste if cuando y=-100 y x<-100
        p2=new cPunto(-100,yXm100);//este es el otro extremo del doblez,
        el izquierdo entonces p1 es el derecho
        p2D=new CpuntoDoble((double)-100.0,yXm100D);//mismo punto de arriba
        solo que en valores tipo doble
        }
    else{//este es el caso cuando y=100 y -100<x<100 ademas de que cuando
    y=-100 x>100
        if(xYm100>100){//entra cuando x>100 y y=-100
            p2=new cPunto(100,yX100);//se establece el segundo punto extremo
            (el punto derecho), siendo p1 el punto izquierdo
            p2D=new CpuntoDoble((double)100.0,yX100D);//mismo punto de arriba
            solo que en valores tipo doble
        }
        else{// este es el cuaso cuando y=100 -100<x<100 y ademas de que
        y=-100 y -100<x<100
            if(xYm100<xY100){//cuando el punto extremo izquierdo esta en
            y=-100 y el derecho cuando y=100
                p1= new cPunto(xYm100,-100);//EL PUNTO EXTREMO INFERIOR
                e izquierdo
                p1D=new CpuntoDoble(xYm100D,(double)-100.0);//mismo punto
                de arriba solo que en valores tipo doble
                p2= new cPunto(xY100,100); //EL PUNTO EXTREMO SUPERIOR
                y derecho
                p2D=new CpuntoDoble(xY100D,(double)100.0);//mismo punto de
                arriba solo que en valores tipo doble
            }
            else{//cuando el punto extremo derecho esta en y=-100 y el
             izquierdo cuando y=100
                p1= new cPunto(xY100,100);//EL PUNTO EXTREMO SUPERIOR e
                izquierdo
                p1D=new CpuntoDoble(xY100D,(double)100.0);//mismo punto
                de arriba solo que en valores tipo doble
```

```
                        p2= new cPunto(xYm100,-100);//EL PUNTO EXTREMO INFERIOR y
                        derecho
                        p2D=new CpuntoDoble(xYm100D,(double)-100.0);//mismo punto de
                         arriba solo que en valores tipo doble
                        }
                    }
                }
            }
        }

        }
        cLinea nuevaLinea=new cLinea(p1D,p2D);
        return nuevaLinea;
    }
}
```

A.14. listaLineas

```
package sahco;
public class listaLineas {
    private cLinea inicio;
    private cLinea fin;
    private int tamao;

listaLineas(){
    inicio=null;
    fin=null;
     tamao =0;
}

public int getTamao(){
        return tamao;
    }

public void insertarAlInicio(cLinea l){
        if(tamao ==0){
            inicio=l;
            fin=l;
             tamao =1;
        }
        else{
            l.setSiguiente(inicio);
            inicio.setAnterior(l);
            inicio=l;
             tamao ++;
        }
    }

public cLinea borrarAlInicio(){
    if(tamao ==0){
        return null;
    }
    else{
        cLinea aux=inicio;
        if(tamao ==1){
            inicio=null;
            fin=null;
```

```
            tamao =0;
        }
        else{
            inicio=aux.getSiguiente();
            inicio.setAnterior(null);
            tamao -=1;
        }
        return aux;
    }
}

public cLinea[] regresaLista(){
       cLinea aux=inicio;
       int i=0;
       cLinea M[];
       M= new cLinea[tamao];
       if( tamao ==0)
          return M;
        if(aux==fin){
            M[i]=aux;
        }
        else{
        while(aux!=fin){
            M[i]=aux;
            aux=aux.getSiguiente();
            i++;
        }
        M[i]=aux;
        }
        return M;
    }

   public boolean existeLinea(cLinea l){
       cLinea lAux=inicio;
       if(lAux==null)
          return false;
       if(lAux==fin)
          return false;
       while(lAux!=fin){
           if(lAux.getP1().getX()==l.getP1().getX()){
               if(lAux.getP1().getY()==l.getP1().getY())
                    return true;
               else{
               if(lAux.getP1().getX()==l.getP2().getX()){
                   if(lAux.getP1().getY()==l.getP2().getY())
                       return true;
                 }
               }
           }
           else{
               if(lAux.getP2().getX()==l.getP1().getX()){
                   if(lAux.getP2().getY()==l.getP1().getY())
                       return true;
                   else{
                       if(lAux.getP2().getX()==l.getP2().getX()){
                           if(lAux.getP2().getY()==l.getP2().getY())
                               return true;
                       }
                   }
```

```
            }
        }
        lAux=lAux.getSiguiente();
    }
    if(lAux==fin){
        if(lAux.getP1().getX()==l.getP1().getX()){
            if(lAux.getP1().getY()==l.getP1().getY())
                return true;
            else{
            if(lAux.getP1().getX()==l.getP2().getX()){
                if(lAux.getP1().getY()==l.getP2().getY())
                    return true;
              }

            }
        }
        else{
            if(lAux.getP2().getX()==l.getP1().getX()){
                if(lAux.getP2().getY()==l.getP1().getY())
                    return true;
                else{
                    if(lAux.getP2().getX()==l.getP2().getX()){
                        if(lAux.getP2().getY()==l.getP2().getY())
                            return true;

                    }
                }
            }
        }
    }
    return false;
  }
}
```

A.15. listaPuntos

```
package sahco;
public class listaPuntos {
//esta clase sirve para ordenar los puntos en una estructura de tipo lista
    private cPunto inicio;
    //variable de clase que indica el primer punto de la lista de puntos
    private cPunto fin;
    //variable de clase que indica el ultimo punto de la lista de puntos
    private int tamao;
    //variable de calse que indica de que tamao es la lista de puntos

    listaPuntos(){//metodo constructor de la clase , inicializa las variables de clase
        inicio=null;//una lista vacia no tiene punto inicial
        fin=null;//ni final
         tamao =0;//es de tamao cero
    }
    public void insertarAlInicio(cPunto p){
    //este metodo pone a un punto p al inicio de la lista
        if( tamao ==0){//cuando la lista esta vacia
            inicio=p;//el punto p se convierte en el primer punto en la lusta
            fin=p;//y tambien en el ultimo ya que solo habra un elemento en la lista
             tamao =1;//ahora la lista tiene un elemento , por lo que su tamao es uno
```

```
    }
    else{//cuando la lista tiene al menos un elemento
        p.setSiguiente(inicio);
        //como p va a ir al inicio ahora, el viejo inicio ira despues de p
        inicio.setAnterior(p);//y el predecesor del viejo inicio sera p
        inicio=p;//ahora el nuevo inicio es p
         tamao ++;
        //la lista ahora tiene un nuevo elemento , por eso se incrementa en uno
    }
}

public int getTamao(){//este metodo regresa el tamao de la lista
    return tamao;
}

public cPunto[] regresaLista(){//este metodo llena un arreglo con los puntos de la lista
   cPunto aux=inicio;//aux apunta al inicio
   int i=0;//servira para definir la posicion del arreglo
   cPunto M[];
   //el arreglo que contendra los puntos que hay en la lista y que se regresa
    al final del metodo
   M= new cPunto[tamao];//se crea el arreglo con el mismo tamao que tiene la lista
   if( tamao ==0)//cuando la lista esta vacia
      return M;//se resgresa el arreglo vacio
    if(aux==fin){//cuando hay un solo elemento en la lista
        M[i]=aux;//el unico punto en el arreglo estara en la pocision 0
    }
    else{//cuando el arreglo tiene mas de un elemento
    while(aux!=fin){//este while recorre toda la lista del inicio hasta el punto final
        M[i]=aux;//se insertan los puntos en el arreglo segun su pocision
        aux=aux.getSiguiente();//se avanza al siguiente punto en la lista
        i++;//se incrementa la posicion
    }
    M[i]=aux;
    //cuando para el while falta introducir el punto final al arreglo, aqui se hace esto
    }
    return M;//se regresa el arreglo cono los puntos introducidos
}

public boolean existePunto(int x,int y){
//este metodo recibe una coordenada y determina si hay un punto en la lista de puntos
con los mismo valores

    cPunto aux=inicio;//aux apunta al inicio de la lista
    if(aux==null)//cuando la lista esta vacia
        return false;
        //regresa que no , ya que no existe un punto que sea igual al que se busca
    if(aux==fin){//si solo hay un punto en la lista
        if(aux.getX()==x && aux.getY()==y)
        //solo se comparan los valores de X y Y de ese unico punto
            return true;//si son iguales se regresa que si
        }
    while(aux!=fin){
    //si hay mas de un punto en la lista, este while recorre toda la lista
        if(aux.getX()==x && aux.getY()==y){
        //se comparan los valores X y Y con cada punto en la lista
            return true;
            //en caso de que sean iguales si existe el punto en la lista y se regresa "si"
        }
```

```
            else //en caso de que no se procede a comparar el siguiente valor en la lista
                aux=aux.getSiguiente();//aux va recooriendo todos los puntos en la lista
        }
        if(aux==fin){//cuando se llega al ultimo punro de la lista
            if(aux.getX()==x && aux.getY()==y)//se compara ese ultimo punto
                return true;//si es igual regresa "si"
            }
        return false;//una vez que se recorrio toda la lista de puntos y no se
        encontro el punto busxcado se regresa que "no"
    }
}
```

A.16. panelPrincipal

```
package sahco;
import java.awt.Color;
import java.awt.Graphics;

public class panelPrincipal extends javax.swing.JPanel{
    private int puntoMedioX;
    private int puntoMedioY;
    private int hojaPapel[][];
    private listaPuntos listaDePuntos;
    private listaLineas listaDeLineas;

    public panelPrincipal(int width, int height,listaPuntos L, listaLineas LL){
        puntoMedioX=(width/2);
        puntoMedioY=(height/2);
        this.setBackground(Color.WHITE);
        this.setBounds(puntoMedioX-240,puntoMedioY-250,460,460);
        hojaPapel=new int[5][5];

        hojaPapel[1][1]=this.getWidth()/2-100;
        hojaPapel[1][2]=this.getHeight()/2-100;
        hojaPapel[1][3]=this.getWidth()/2+100;
        hojaPapel[1][4]=this.getHeight()/2-100;
        hojaPapel[2][1]=this.getWidth()/2-100;
        hojaPapel[2][2]=this.getHeight()/2-100;
        hojaPapel[2][3]=this.getWidth()/2-100;
        hojaPapel[2][4]=this.getHeight()/2+100;
        hojaPapel[3][1]=this.getWidth()/2-100;
        hojaPapel[3][2]=this.getHeight()/2+100;
        hojaPapel[3][3]=this.getWidth()/2+100;
        hojaPapel[3][4]=this.getHeight()/2+100;
        hojaPapel[4][1]=this.getWidth()/2+100;
        hojaPapel[4][2]=this.getHeight()/2+100;
        hojaPapel[4][3]=this.getWidth()/2+100;
        hojaPapel[4][4]=this.getHeight()/2-100;

        listaDePuntos=L;

        listaDeLineas=LL;
    }

    @Override
    public void paint(Graphics g){
        super.paint(g);
```

```
    g.setColor(Color.BLUE);

    g.drawLine(hojaPapel[1][1],hojaPapel[1][2],hojaPapel[1][3],hojaPapel[1][4]);
    g.drawLine(hojaPapel[2][1],hojaPapel[2][2],hojaPapel[2][3],hojaPapel[2][4]);
    g.drawLine(hojaPapel[3][1],hojaPapel[3][2],hojaPapel[3][3],hojaPapel[3][4]);
    g.drawLine(hojaPapel[4][1],hojaPapel[4][2],hojaPapel[4][3],hojaPapel[4][4]);

    cPunto m[];
    m=listaDePuntos.regresaLista();
    g.setColor(Color.black);
    if(m.length!=0){
    for(int i=0;i<listaDePuntos.getTamao();i++){
        g.fillOval(this.getWidth()/2-2+m[i].getX(),this.getHeight()/2-2-m[i].getY(),4,4);
        }
    }

    cLinea l[];
    l=listaDeLineas.regresaLista();
    if(l.length!=0){
         for(int i=0; i<listaDeLineas.getTamao();i++){
            g.drawLine((int)l[i].getP1().getX()+this.getWidth()/2,this.getHeight()/2-
            (int)l[i].getP1().getY(),(int)l[i].getP2().getX()+this.getWidth()/2,
            this.getHeight()/2- (int)l[i].getP2().getY());
        }
    }
}

public void dibujaLinea(int px1,int py1,int px2, int py2,Graphics g){
    g.drawLine(px1+this.getWidth()/2,this.getHeight()/2-py1,px2+this.getWidth()/2,
    this.getHeight()/2-py2);
}

public void dibujaLineaDeColor(Color c,int px1,int py1,int px2, int py2,Graphics g){
    g.setColor(c);
    g.drawLine(px1+this.getWidth()/2,this.getHeight()/2-py1,px2+this.getWidth()/2,
    this.getHeight()/2-py2);
}

public void dibujaDadaLinea(Color c, cLinea l,Graphics g){
    g.setColor(c);
    g.drawLine((int)l.getP1().getX()+this.getWidth()/2,this.getHeight()/2-
    (int)l.getP1().getY(),(int)l.getP2().getX()+this.getWidth()/2,this.getHeight()/2-
    (int)l.getP2().getY());
}

public void dibujaTrapecioColor(Color c,int px1,int py1,int px2, int py2,int px3,int py3,
int px4, int py4,Graphics g){
    g.setColor(c);
    int[] X= new int[4];
    X[0]=px1+this.getWidth()/2;
    X[1]=px2+this.getWidth()/2;
    X[2]=px3+this.getWidth()/2;
    X[3]=px4+this.getWidth()/2;
    int[] Y= new int[4];
    Y[0]=this.getHeight()/2-py1;
    Y[1]=this.getHeight()/2-py2;
    Y[2]=this.getHeight()/2-py3;
    Y[3]=this.getHeight()/2-py4;
    g.fillPolygon(X, Y, 4);
```

```
    }
    public void dibujarPoligonoDeColor(Color c,int puntos ,int[] X,int[] Y,Graphics g){
        for(int i=0;i<puntos;i++){
            X[i]=X[i]+this.getWidth()/2;
            Y[i]=this.getHeight()/2-Y[i];
        }
        g.setColor(c);
        g.fillPolygon(X,Y,puntos);
    }

    public void dibujaPuntoDeColor(Color c,cPunto p,Graphics g){
        g.setColor(c);
        g.fillOval(this.getWidth()/2-2+p.getX(),this.getHeight()/2-2-p.getY(),4,4);

    }

    public void refresh(Graphics g){
        this.paint(g);
    }

    public void setListaPuntos(listaPuntos lp){
        listaDePuntos=lp;
    }

    public void setListaLineas(listaLineas ll){
        listaDeLineas=ll;
    }
}
```

A.17. resuelveEcuacion2doGrado

```
package sahco;
public class resuelveEcuacion2doGrado {
    private double a;
    private double b;
    private double c;
    private double s1;
    private double s2;


    resuelveEcuacion2doGrado(double A,double B, double C){
        a=A;
        b=B;
        c=C;
    }

    public void resuelve(){
        s1=(-b+Math.sqrt((b*b)-4.0*a*c))/(2.0*a);
        s2=(-b-Math.sqrt((b*b)-4.0*a*c))/(2.0*a);
    }

    public double getDiscriminante(){
        return (b*b)-4.0*a*c;
    }
    public double getS1(){
        return s1;
```

```
    }

    public double getS2(){
        return s2;
    }
}
```

A.18. seleccionDeTresLineas

```
package sahco;
public class seleccionDeTresLineas extends javax.swing.JFrame {
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    cLinea L1;
    cLinea L2;
    cLinea L3;
    double x1,y1,x2,y2,X1,Y1,X2,Y2,xx1,yy1,xx2,yy2;

    public seleccionDeTresLineas(panelPrincipal H,listaLineas LL,listaPuntos LP,cLinea l1,
    cLinea l2,cLinea l3) {
        x1=l1.getP1().getX();
        y1=l1.getP1().getY();
        x2=l1.getP2().getX();
        y2=l1.getP2().getY();

        X1=l2.getP1().getX();
        Y1=l2.getP1().getY();
        X2=l2.getP2().getX();
        Y2=l2.getP2().getY();

        xx1=l3.getP1().getX();
        yy1=l3.getP1().getY();
        xx2=l3.getP2().getX();
        yy2=l3.getP2().getY();

        L1=l1;
        L2=l2;
        L3=l3;

        h=H;
        ll=LL;
        lp=LP;
        initComponents();
        h.dibujaDadaLinea(this.jPanel1.getBackground(), l1,h.getGraphics());
        h.dibujaDadaLinea(this.jPanel2.getBackground(), l2,h.getGraphics());
        h.dibujaDadaLinea(this.jPanel3.getBackground(), l3,h.getGraphics());

        addWindowListener(new java.awt.event.WindowAdapter() {
        //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
            @Override
            public void windowClosing(java.awt.event.WindowEvent evt) {
            //cerrando la ventana
                h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
            }
        });//aqui se acaba la parte del codigo que borra lo que este demas en el panel
        antes de cerrar la ventana
```

```
}


@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton2 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    jSeparator1 = new javax.swing.JSeparator();
    jButton3 = new javax.swing.JButton();
    jPanel3 = new javax.swing.JPanel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setResizable(false);

    jButton2.setText("Opcin2");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    jPanel1.setBackground(new java.awt.Color(255, 0, 0));

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 26, Short.MAX_VALUE)
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 23, Short.MAX_VALUE)
    );

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 24));
    jLabel1.setText("Seleccione la opcin de doblado");

    jButton1.setText("Opcin1");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jPanel2.setBackground(new java.awt.Color(0, 102, 0));

    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 26, Short.MAX_VALUE)
    );
    jPanel2Layout.setVerticalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
    .addGap(0, 23, Short.MAX_VALUE)
);

jButton3.setText("Opcin3");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jPanel3.setBackground(new java.awt.Color(0, 0, 255));

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 26, Short.MAX_VALUE)
);
jPanel3Layout.setVerticalGroup(
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(27, 27, 27)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 346,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 390,
            Short.MAX_VALUE))
        .addContainerGap())
    .addGroup(layout.createSequentialGroup()
        .addGap(153, 153, 153)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jButton3)
                .addGap(18, 18, 18)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addComponent(jButton2)
                .addGap(18, 18, 18)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addComponent(jButton1)
                .addGap(18, 18, 18)
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        .addGap(132, 132, 132))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
```

```
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1))
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                 false)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(27, 27, 27)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton3))
                .addGap(27, 27, 27))
        );

        pack();
    }// </editor-fold>

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)X1),Integer.toString((int)Y1),Integer.toString((int)X2),
    Integer.toString((int)Y2), true);
    ll.borrarAlInicio();
    ll.insertarAlInicio(L2);
    this.dispose();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)x1),Integer.toString((int)y1),Integer.toString((int)x2),
    Integer.toString((int)y2), true);
    ll.borrarAlInicio();
    ll.insertarAlInicio(L1);
    this.dispose();
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)xx1),Integer.toString((int)yy1),Integer.toString((int)xx2),
    Integer.toString((int)yy2), true);
    ll.borrarAlInicio();
    ll.insertarAlInicio(L3);
    this.dispose();
}

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
```

```
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JSeparator jSeparator1;
    // End of variables declaration
}
```

A.19. seleccionDoblez

```
package sahco;
import java.awt.Color;

public class seleccionDoblez extends javax.swing.JFrame {
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    double x1,y1,x2,y2,X1,Y1,X2,Y2;
    cLinea L1;
    cLinea L2;

    seleccionDoblez(panelPrincipal H,listaLineas LL,listaPuntos LP,cLinea l1,cLinea l2) {
        x1=l1.getP1().getX();
        y1=l1.getP1().getY();
        x2=l1.getP2().getX();
        y2=l1.getP2().getY();

        X1=l2.getP1().getX();
        Y1=l2.getP1().getY();
        X2=l2.getP2().getX();
        Y2=l2.getP2().getY();

        L1=l1;
        L2=l2;

        h=H;
        ll=LL;
        lp=LP;
        initComponents();
        h.dibujaDadaLinea(this.jPanel1.getBackground(), l1,h.getGraphics());
        h.dibujaDadaLinea(this.jPanel2.getBackground(), l2,h.getGraphics());

        this.setLocation(800,330);//se coloca la posicion de la ventana
        addWindowListener(new java.awt.event.WindowAdapter() {
        //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
            @Override
            public void windowClosing(java.awt.event.WindowEvent evt) {
            //cerrando la ventana
                h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
            }
        });//aqui se acaba la parte del codigo que borra lo que este demas en el panel antes
         de cerrar la ventana
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
jLabel1 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jPanel1 = new javax.swing.JPanel();
jPanel2 = new javax.swing.JPanel();
jSeparator1 = new javax.swing.JSeparator();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setResizable(false);

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 24));
jLabel1.setText("Seleccione la opcin de doblado");

jButton1.setText("Opcin1");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Opcin2");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jPanel1.setBackground(new java.awt.Color(204, 51, 255));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 24, Short.MAX_VALUE)
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

jPanel2.setBackground(new java.awt.Color(51, 255, 51));

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 26, Short.MAX_VALUE)
);
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(27, 27, 27)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 346,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 373,
                    Short.MAX_VALUE))
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap(136, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                false)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
                     layout.createSequentialGroup()
                        .addComponent(jButton2)
                        .addGap(18, 18, 18)
                        .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
                    layout.createSequentialGroup()
                        .addComponent(jButton1)
                        .addGap(18, 18, 18)
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                             javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(132, 132, 132))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1))
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton2))
                .addGap(25, 25, 25))
        );

        pack();
    }// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)x1),Integer.toString((int)y1),
    Integer.toString((int)x2),Integer.toString((int)y2), true);
    ll.borrarAlInicio();
    ll.insertarAlInicio(L1);
    this.dispose();
```

```
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)X1),Integer.toString((int)Y1),
    Integer.toString((int)X2),Integer.toString((int)Y2), true);
    ll.borrarAlInicio();
    ll.insertarAlInicio(L2);
    this.dispose();
}

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JSeparator jSeparator1;
    // End of variables declaration
}
```

A.20. ventanaError

```
package sahco;
public class ventanaError extends javax.swing.JFrame {

    /** Creates new form ventanaError */
    public ventanaError(String s, String s2) {
        initComponents();
        setLocationRelativeTo(null);
        jLabel1.setText(s);
        jLabel2.setText(s2);
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setResizable(false);

        jLabel1.setText("jLabel1");

        jButton1.setText("jButton1");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel2.setText("jLabel2");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
```

```
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 503,
                         Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 503,
                         Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(224, 224, 224)
                        .addComponent(jButton1)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(36, 36, 36)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap(28, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

    /**
     * @param args the command line arguments
     */

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    // End of variables declaration
}
```

A.21. ventanaPrincipal

```
package sahco;
public class ventanaPrincipal extends javax.swing.JFrame {
    private panelPrincipal p;
    private listaPuntos listaP;
    private listaLineas listaL;
```

```
/** Creates new form ventanaPrincipal */
public ventanaPrincipal() {
    initComponents();
    setLocationRelativeTo(null);
    //this.setExtendedState(MAXIMIZED_BOTH);
    this.setVisible(true);
    listaP = new listaPuntos();
    listaL= new listaLineas();
    p = new panelPrincipal(this.getWidth(),this.getHeight(),listaP,listaL);
    this.add(p);
    p.refresh(p.getGraphics());
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jButton5 = new javax.swing.JButton();
    jButton6 = new javax.swing.JButton();
    borrarPuntos = new javax.swing.JButton();
    borrarLineas = new javax.swing.JButton();
    jButton7 = new javax.swing.JButton();
    archivos = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Simulador Axiomas de Huzita");
    setBounds(new java.awt.Rectangle(0, 0, 800, 1000));
    setResizable(false);

    jButton1.setText("Punto");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jButton2.setText("Axioma1");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    jButton3.setText("Axioma2");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });
```

```
jButton4.setText("Axioma3");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jButton5.setText("Axioma4");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jButton6.setText("Axioma5");
jButton6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton6ActionPerformed(evt);
    }
});

borrarPuntos.setText("borrarPuntos");
borrarPuntos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        borrarPuntosActionPerformed(evt);
    }
});

borrarLineas.setText("borrarLineas");
borrarLineas.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        borrarLineasActionPerformed(evt);
    }
});

jButton7.setText("Axioma6");
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});

archivos.setText("archivos");
archivos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        archivosActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jButton7)
        .addContainerGap(742, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                 false)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(10, 10, 10)
                        .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(layout.createParallelGroup
                    (javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jButton3)
                        .addComponent(jButton4)
                        .addComponent(jButton5)
                        .addComponent(jButton6))))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 629,
             Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(borrarPuntos)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                 false)
                    .addComponent(archivos, javax.swing.GroupLayout.Alignment.LEADING,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
                    .addComponent(borrarLineas, javax.swing.GroupLayout.Alignment.LEADING,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)))
            .addGap(18, 18, 18))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(65, 65, 65)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton1)
                .addComponent(borrarPuntos))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton2)
                .addComponent(borrarLineas))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton3)
                .addComponent(archivos))
            .addGap(18, 18, 18)
            .addComponent(jButton4)
            .addGap(18, 18, 18)
            .addComponent(jButton5)
            .addGap(18, 18, 18)
            .addComponent(jButton6)
            .addGap(18, 18, 18)
            .addComponent(jButton7)
            .addContainerGap(176, Short.MAX_VALUE))
    );
```

```
        pack();
    }// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaPunto vp=new ventanaPunto(p,listaP);
vp.setVisible(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma1 VA=new ventanaAxioma1(listaP,listaL, p);
VA.setVisible(true);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma2 VA2=new ventanaAxioma2(listaP,listaL, p);
VA2.setVisible(true);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma3 VA3=new ventanaAxioma3(listaP,listaL, p);
VA3.setVisible(true);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma4 VA4=new ventanaAxioma4(listaP,listaL, p);
VA4.setVisible(true);
}

private void borrarPuntosActionPerformed(java.awt.event.ActionEvent evt) {
listaP=new listaPuntos();
p.setListaPuntos(listaP);
p.refresh(p.getGraphics());
}

private void borrarLineasActionPerformed(java.awt.event.ActionEvent evt) {
listaL=new listaLineas();
p.setListaLineas(listaL);
p.refresh(p.getGraphics());
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma5 VA5=new ventanaAxioma5(listaP, p,listaL);
VA5.setVisible(true);
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
ventanaAxioma6 VA6=new ventanaAxioma6(listaP, p,listaL);
VA6.setVisible(true);
}

private void archivosActionPerformed(java.awt.event.ActionEvent evt) {
cargarArchivo cA=new cargarArchivo(listaP, p,listaL);
cA.setVisible(true);
}

    /**
     * @param args the command line arguments
     */
```

```
    // Variables declaration - do not modify
    private javax.swing.JButton archivos;
    private javax.swing.JButton borrarLineas;
    private javax.swing.JButton borrarPuntos;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    // End of variables declaration
}
```

A.22. ventanaPunto

```
package sahco;
public class ventanaPunto extends javax.swing.JFrame {
    panelPrincipal p;
    int x;
    int y;
    listaPuntos lp;
    /** Creates new form ventanaPunto */
    public ventanaPunto(panelPrincipal P, listaPuntos lista) {
        p=P;
        initComponents();
        this.setLocation(900,330);
        lp=lista;

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        xText = new javax.swing.JTextField();
        botonAceptar = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        yText = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Introduciendo un punto");
        setResizable(false);

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel1.setText("Introduce las coordenadas \"X\" y \"Y\" ");
```

```
botonAceptar.setText("Aceptar");
botonAceptar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonAceptarActionPerformed(evt);
    }
});

jLabel2.setText("X");

jLabel3.setText("Y");

jLabel4.setFont(new java.awt.Font("Tahoma", 0, 14));
jLabel4.setText("para la creacion de un punto");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap(54, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(layout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(13, 13, 13)
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                         91, Short.MAX_VALUE)
                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
                        27, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(8, 8, 8))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(xText, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        136, Short.MAX_VALUE))))
            .addComponent(jLabel4))
        .addGap(63, 63, 63))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(37, Short.MAX_VALUE)
        .addComponent(jLabel1)
        .addGap(29, 29, 29))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(185, Short.MAX_VALUE)
        .addComponent(yText, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(79, 79, 79))
    .addGroup(layout.createSequentialGroup()
        .addGap(110, 110, 110)
        .addComponent(botonAceptar)
        .addContainerGap(111, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
            javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(xText, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
                             javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
                             javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(yText, javax.swing.GroupLayout.PREFERRED_SIZE,
                             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(botonAceptar)))
                .addContainerGap(14, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

private void botonAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    x=Integer.parseInt(xText.getText());
    y=Integer.parseInt(yText.getText());
    if(x<-100||x>100||y<-100||y>100){
        ventanaError v=new ventanaError("Este punto no se puede dibujar,
         el punto no esta en el rango -100 a 100", "Favor de introducir un punto entre -100 a 100");
        v.setVisible(true);
        this.dispose();
    }
    else{
        cPunto nuevo = new cPunto(x,y);
        lp.insertarAlInicio(nuevo);
        p.getGraphics().fillOval(x+p.getWidth()/2-2,p.getHeight()/2-y-2, 4, 4);
        cPunto m[];
        m=lp.regresaLista();
        for(int clear = 0; clear < 10; clear++)
        {
            System.out.println("\b") ;
        }
        for(int i=0;i<lp.getTamao();i++){
            System.out.println("P"+(i+1)+": ("+m[i].getX()+","+m[i].getY()+")");
        }
    }
}


    // Variables declaration - do not modify
    private javax.swing.JButton botonAceptar;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JTextField xText;
    private javax.swing.JTextField yText;
    // End of variables declaration
```

```
}
```

A.23. cargarArchivo

```
package sahco;
import javax.swing.*;
import java.io.*;
public class cargarArchivo extends javax.swing.JFrame {
    JFileChooser seleccionado= new JFileChooser();
    File archivo;
    gestionA gestion;

    private listaPuntos lp;
    private panelPrincipal h;
    private listaLineas ll;
    public cargarArchivo(listaPuntos LP, panelPrincipal H, listaLineas LL) {
        gestion=new gestionA();
        lp=LP;
        h=H;
        ll=LL;
        initComponents();
    this.setLocation(900,370);
    }

    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnAbrirArchivo = new javax.swing.JButton();
        btnGuardar = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setResizable(false);

        btnAbrirArchivo.setText("Abrir Archivo");
        btnAbrirArchivo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnAbrirArchivoActionPerformed(evt);
            }
        });

        btnGuardar.setText("Guardar Archivo");
        btnGuardar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnGuardarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(btnGuardar, javax.swing.GroupLayout.PREFERRED_SIZE, 143,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addComponent(btnAbrirArchivo, javax.swing.GroupLayout.PREFERRED_SIZE, 143,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(32, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(25, 25, 25)
                .addComponent(btnAbrirArchivo, javax.swing.GroupLayout.PREFERRED_SIZE, 32,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(btnGuardar, javax.swing.GroupLayout.PREFERRED_SIZE, 32,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(22, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

private void btnAbrirArchivoActionPerformed(java.awt.event.ActionEvent evt) {
if(seleccionado.showDialog(this,"Abrir Archivo")==JFileChooser.APPROVE_OPTION){
    this.dispose();
    archivo=seleccionado.getSelectedFile();
    String contenido=gestion.abrirATexto(archivo);
    System.out.println(contenido);
    int contador=0;

    while(true){
        String x="";
        String y="";
        char c=contenido.charAt(contador);
        if(c=='@')
            break;
        while(true){
            c=contenido.charAt(contador);
            if(c==',')
                break;
            x+=c;
            contador++;
        }
        contador=contador+1;
        while(true){
            c=contenido.charAt(contador);
            if(c=='+')
                break;
             y+=c;
            contador++;
        }
        lp.insertarAlInicio(new cPunto(Integer.parseInt(x),Integer.parseInt(y)));
        System.out.println(x+","+y);
        contador++;
    }
    contador+=1;
    while(true){
        CpuntoDoble P1=null;
        CpuntoDoble P2=null;
        char c=contenido.charAt(contador);
       if(c==';')
            break;
```

```
    String x="";
    String y="";
    c=contenido.charAt(contador);
    if(c=='@')
        break;
    while(true){
        c=contenido.charAt(contador);
        if(c==',')
            break;
        x+=c;
        contador++;
    }
    contador=contador+1;
    while(true){
        c=contenido.charAt(contador);
        if(c=='@')
            break;
         y+=c;
        contador++;
    }
    P1=new CpuntoDoble(Double.parseDouble(x),Double.parseDouble(y));
    System.out.println(x+","+y);
    contador++;



    x="";
    y="";
    c=contenido.charAt(contador);
    if(c=='@')
        break;
    while(true){
        c=contenido.charAt(contador);
        if(c==',')
            break;
        x+=c;
        contador++;
    }
    contador=contador+1;
    while(true){
        c=contenido.charAt(contador);
        if(c=='@')
            break;
         y+=c;
        contador++;
    }
    P2=new CpuntoDoble(Double.parseDouble(x),Double.parseDouble(y));
    System.out.println("P2"+x+","+y);
    contador++;

ll.insertarAlInicio(new cLinea(P1,P2));
}

System.out.println(contenido);

System.out.println(lp.getTamao());

System.out.println(lp.regresaLista()[1].getX()+" "+lp.regresaLista()[1].getY());
```

```
    h.refresh(h.getGraphics());
}
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
if(seleccionado.showDialog(this,"Guardar Archivo")==JFileChooser.APPROVE_OPTION){
    this.dispose();
    archivo=seleccionado.getSelectedFile();
    String archivoTexto="";
    for(int i=0;i<lp.getTamao();i++){
        archivoTexto+=lp.regresaLista()[i].getX()+","+lp.regresaLista()[i].getY();
        archivoTexto+="+";
    }
    archivoTexto+="@";
    for(int i=0;i<ll.getTamao();i++){
        archivoTexto+=ll.regresaLista()[i].getP1().getX()+","+ll.regresaLista()[i].
        getP1().getY()+"@";
        archivoTexto+=ll.regresaLista()[i].getP2().getX()+","+ll.regresaLista()[i].
        getP2().getY()+"@";
    }
    archivoTexto+=";";
    gestion.guardarArchivo(archivo, archivoTexto);
}
}

    // Variables declaration - do not modify
    private javax.swing.JButton btnAbrirArchivo;
    private javax.swing.JButton btnGuardar;
    // End of variables declaration
}
```

A.24. ventanaAxioma1

```
package sahco;
import java.awt.Color;

public class ventanaAxioma1 extends javax.swing.JFrame {
//esta clase es la ventana que pide dos puntos para realizar el axioma 1 sonre estos
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas
    private cPunto punto1;//variable que contiene punto de la linea de doblado
    private cPunto punto2;//el otro punto de la linea de doblado

    public ventanaAxioma1(listaPuntos LP, listaLineas LL,panelPrincipal H) {
    //este es el constructor de un objeto de la clase netanaAxioma1
        lp=LP;//se asigna la lista de puntos con lo que se le pasa al constructor cuando se
         crea un nuevo onbjeto tipo ventanaAxioma1
        ll=LL;//se asigna el panel con lo que se le pasa al constructor cuando se crea un
        nuevo onbjeto tipo ventanaAxioma1
        h=H;//se asigna la lista de lineas con lo que se le pasa al constructor cuando se
        crea un nuevo objeto tipo ventanaAxioma1
        initComponents();//se inicializan los componenetes graficos de la ventana
        for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador del 1er
         punto(jComboBox1)
            this.jComboBox1.addItem("Punto"+(i+1));//cada uno de los puntos
```

```
    }//este seleccionador sirve para escoger puntos, al igual que el de abajo
    for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador del 1er
     punto(seleccionaPunto2)
        this.seleccionaPunto2.addItem("Punto"+(i+1));//cada uno de los puntos
    }
    this.setLocation(800,330);//se coloca la posicion de la ventana

    addWindowListener(new java.awt.event.WindowAdapter() {
    //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
        @Override
        public void windowClosing(java.awt.event.WindowEvent evt) {
        //cerrando la ventana
            h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
        }
    });//aqui se acaba la parte del codigo que borra lo que este demas en el panel antes
     de cerrar la ventana
}


@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jComboBox1 = new javax.swing.JComboBox();
    aplicraConSeleccionador = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    jPanel1 = new javax.swing.JPanel();
    seleccionaPunto2 = new javax.swing.JComboBox();
    etiqueta1 = new javax.swing.JLabel();
    etiqueta2 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Axioma 1");
    setResizable(false);

    jLabel6.setText("Punto 1");

    jLabel7.setText("Punto 2");

    jLabel8.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabel8.setText("Selecciona los puntos ");

    jComboBox1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });

    aplicraConSeleccionador.setText("Aplicar");
    aplicraConSeleccionador.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            aplicraConSeleccionadorActionPerformed(evt);
        }
    });
```

```
jPanel2.setBackground(new java.awt.Color(255, 0, 0));

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 28, Short.MAX_VALUE)
);
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 26, Short.MAX_VALUE)
);

jPanel1.setBackground(new java.awt.Color(255, 200, 0));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 30, Short.MAX_VALUE)
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 28, Short.MAX_VALUE)
);

seleccionaPunto2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seleccionaPunto2ActionPerformed(evt);
    }
});

etiqueta1.setText("jLabel1");
etiqueta1.setBorder(javax.swing.BorderFactory.createLineBorder
(new java.awt.Color(0, 0, 0)));

etiqueta2.setText("jLabel1");
etiqueta2.setBorder(javax.swing.BorderFactory.createLineBorder
(new java.awt.Color(0, 0, 0)));

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(36, 36, 36)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
         false)
            .addComponent(etiqueta1, javax.swing.GroupLayout.Alignment.LEADING,
                 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
             Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
            layout.createSequentialGroup()
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel6)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 81,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(86, 86, 86)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
        false)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel7))
            .addComponent(etiqueta2, javax.swing.GroupLayout.DEFAULT_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(seleccionaPunto2, javax.swing.GroupLayout.PREFERRED_SIZE,
        79, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(66, Short.MAX_VALUE))
    .addGroup(layout.createSequentialGroup()
        .addGap(173, 173, 173)
        .addComponent(aplicraConSeleccionador, javax.swing.GroupLayout.PREFERRED_SIZE,
         152, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(171, Short.MAX_VALUE))
    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 496,
     Short.MAX_VALUE)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(171, Short.MAX_VALUE)
        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 167,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(158, 158, 158))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(14, 14, 14)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createParallelGroup
            (javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel6))
            .addGroup(layout.createParallelGroup
            (javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel7)
                .addComponent(seleccionaPunto2, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
            PREFERRED_SIZE))
```

```
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.LEADING)
                    .addComponent(etiqueta1)
                    .addComponent(etiqueta2))
                .addGap(40, 40, 40)
                .addComponent(aplicraConSeleccionador, javax.swing.GroupLayout.
                PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(24, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
//aqui se define lo que pasa cuando se usa el seleccionador del punto 1(jComboBox)
    h.refresh(h.getGraphics());//esta linea redibuja el panel
    if(punto2!=null)//si ya se selecciono el punto 2
        h.dibujaPuntoDeColor(this.jPanel1.getBackground(), punto2,h.getGraphics());
        //dibuja el punto 2
    punto1=this.darPuntoApartirDeString(this.jComboBox1.getSelectedItem().toString());
    this.etiqueta1.setText("("+punto1.getX()+","+punto1.getY()+")");
    h.dibujaPuntoDeColor(Color.red, punto1,h.getGraphics());
    //dibuja el punto 1 en rojo
}

private void aplicraConSeleccionadorActionPerformed(java.awt.event.ActionEvent evt) {
    doblado D=new doblado(lp,ll,h);
    //se crea un objeto tipo doblado que se encargara de crear la linea de doblez y
    la animacion correspondiente
    D.hacerDoblado(Integer.toString(punto1.getX()),Integer.toString(punto1.getY()),
    Integer.toString(punto2.getX()),Integer.toString(punto2.getY()),false);
    h.dibujaPuntoDeColor(Color.black, punto1,h.getGraphics());//dibuja el punto 1 en negro
    h.dibujaPuntoDeColor(Color.black, punto2,h.getGraphics());//dibuja el punto 2 en negro
    punto1=null;
    punto2=null;
}

private void seleccionaPunto2ActionPerformed(java.awt.event.ActionEvent evt) {
//aqui se define lo que pasa cuando se usa el seleccionador del punto 2(seleccionaPunto2)
    h.refresh(h.getGraphics());//esta linea redibuja el panel
    if(punto1!=null)//si ya se selecciono el punto 1
        h.dibujaPuntoDeColor(Color.red, punto1,h.getGraphics());//dibuja el punto 1
    punto2=this.darPuntoApartirDeString(this.seleccionaPunto2.getSelectedItem().toString());
    this.etiqueta2.setText("("+punto2.getX()+","+punto2.getY()+")");
    h.dibujaPuntoDeColor(this.jPanel1.getBackground(), punto2,h.getGraphics());
    //dibuja el punto 2
}
public cPunto darPuntoApartirDeString(String sPunto){
        cPunto[] lista=lp.regresaLista();
        return lista[Integer.parseInt(sPunto.substring(5,6))-1];

}
    // Variables declaration - do not modify
    private javax.swing.JButton aplicraConSeleccionador;
    private javax.swing.JLabel etiqueta1;
    private javax.swing.JLabel etiqueta2;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel6;
```

```
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JComboBox seleccionaPunto2;
    // End of variables declaration
}
```

A.25. ventanaAxioma2

```
package sahco;
import java.awt.Color;
import java.lang.Math;

public class ventanaAxioma2 extends javax.swing.JFrame {
    //esta clase crea la ventana que pide los dos puntos necesarios para aplicar el axioma 2
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas
    private cPunto punto1;//variable que contiene punto de la linea de doblado
    private cPunto punto2;//el otro punto de la linea de doblado
    public ventanaAxioma2(listaPuntos LP, listaLineas LL,panelPrincipal H) {
        initComponents();//se inicializan los componenetes graficos de la ventana
        this.setLocation(800,330);//se pone la posicion de la ventana
        lp=LP;
        ll=LL;
        h=H;
        for(int i=0;i<lp.getTamao();i++){
            this.seleccionadorDePuntos.addItem("Punto "+(i+1));
        }
        for(int i=0;i<lp.getTamao();i++){
            this.seleccionarPunto2.addItem("Punto "+(i+1));
        }

        addWindowListener(new java.awt.event.WindowAdapter() {
        //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
            @Override
            public void windowClosing(java.awt.event.WindowEvent evt) {
            //cerrando la ventana
                h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
            }
        });//aqui se acaba la parte del codigo que borra lo que este demas en el panel
        antes de cerrar la ventana
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jLabel8 = new javax.swing.JLabel();
        seleccionadorDePuntos = new javax.swing.JComboBox();
        panelP1 = new javax.swing.JPanel();
        labelP1 = new javax.swing.JLabel();
        labelP2 = new javax.swing.JLabel();
        panelP2 = new javax.swing.JPanel();
```

```
seleccionarPunto2 = new javax.swing.JComboBox();
jSeparator1 = new javax.swing.JSeparator();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Axioma 2");
setResizable(false);

jButton1.setText("Aplicar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel8.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel8.setText("Selecciona los puntos ");

seleccionadorDePuntos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seleccionadorDePuntosActionPerformed(evt);
    }
});

panelP1.setBackground(new java.awt.Color(255, 204, 0));

javax.swing.GroupLayout panelP1Layout = new javax.swing.GroupLayout(panelP1);
panelP1.setLayout(panelP1Layout);
panelP1Layout.setHorizontalGroup(
    panelP1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 25, Short.MAX_VALUE)
);
panelP1Layout.setVerticalGroup(
    panelP1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

labelP1.setText("eligiendo punto");

labelP2.setText("punto elegido");

panelP2.setBackground(new java.awt.Color(255, 0, 51));

javax.swing.GroupLayout panelP2Layout = new javax.swing.GroupLayout(panelP2);
panelP2.setLayout(panelP2Layout);
panelP2Layout.setHorizontalGroup(
    panelP2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 25, Short.MAX_VALUE)
);
panelP2Layout.setVerticalGroup(
    panelP2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

seleccionarPunto2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seleccionarPunto2ActionPerformed(evt);
    }
});
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(137, 137, 137)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.TRAILING)
                    .addComponent(panelP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
                    .addComponent(panelP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                    PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING, false)
                    .addComponent(seleccionarPunto2, 0, javax.swing.GroupLayout.
                    DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(seleccionadorDePuntos, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 98, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.LEADING)
                    .addComponent(labelP1, javax.swing.GroupLayout.PREFERRED_SIZE, 171,
                         javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(labelP2, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
                         javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGroup(layout.createSequentialGroup()
                .addGap(126, 126, 126)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 185,
                     javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap())
    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 469, Short.MAX_VALUE)
    .addGroup(layout.createSequentialGroup()
        .addGap(159, 159, 159)
        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 150,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(160, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addGap(7, 7, 7)
        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(7, 7, 7)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(seleccionadorDePuntos, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(labelP1, javax.swing.GroupLayout.DEFAULT_SIZE, 22,
                 Short.MAX_VALUE))
```

```
                .addComponent(panelP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(22, 22, 22)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(panelP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(1, 1, 1))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(seleccionarPunto2, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(labelP2, javax.swing.GroupLayout.DEFAULT_SIZE, 24, Short.MAX_VALUE)))
            .addGap(31, 31, 31)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
                 javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(39, 39, 39))
        );

        pack();
    }// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //esto es lo que se hace cuando se presiona el boton de aceptar
    cLineaPerpendicularDados2Puntos lineaP=new cLineaPerpendicularDados2Puntos(punto1,punto2,lp,ll,h);
    //se crea la linea perpendicular, la linea de doblez
    doblado D=new doblado(lp,ll,h);
    //se crea un objeto tipo doblado que se encargara de introducir as la lista la linea de
     doblez y la animacion correspondiente
    D.hacerDoblado(lineaP.x1(),lineaP.y1(),lineaP.x2(),lineaP.y2(),true);
    //se invoca el metodo de hacer doblado dentro de la calse doblado
}

private void seleccionadorDePuntosActionPerformed(java.awt.event.ActionEvent evt) {
h.refresh(h.getGraphics());
    if(punto2!=null){
        h.dibujaPuntoDeColor(panelP2.getBackground(), punto2, h.getGraphics());
    }
punto1=lp.regresaLista()[Integer.parseInt(this.seleccionadorDePuntos.getSelectedItem().
toString().substring(6))-1];
h.dibujaPuntoDeColor(panelP1.getBackground(), punto1, h.getGraphics());
labelP1.setText("("+punto1.getX()+","+punto1.getY()+")");
}

private void seleccionarPunto2ActionPerformed(java.awt.event.ActionEvent evt) {
h.refresh(h.getGraphics());
    if(punto1!=null){
       h.dibujaPuntoDeColor(panelP1.getBackground(), punto1, h.getGraphics());
    }
punto2=lp.regresaLista()[Integer.parseInt(this.seleccionarPunto2.getSelectedItem().
toString().substring(6))-1];
h.dibujaPuntoDeColor(panelP2.getBackground(), punto2, h.getGraphics());
labelP2.setText("("+punto2.getX()+","+punto2.getY()+")");
}


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JSeparator jSeparator1;
```

```
    private javax.swing.JLabel labelP1;
    private javax.swing.JLabel labelP2;
    private javax.swing.JPanel panelP1;
    private javax.swing.JPanel panelP2;
    private javax.swing.JComboBox seleccionadorDePuntos;
    private javax.swing.JComboBox seleccionarPunto2;
    // End of variables declaration
}
```

A.26. ventanaAxioma3

```
package sahco;
import java.lang.Math;
import java.awt.Color;

public class ventanaAxioma3 extends javax.swing.JFrame {
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas

    private cLinea linea1;
    private cLinea linea2;

    private double X1,Y1,X2,Y2;

    private int xI;
    private int yI;



    public ventanaAxioma3(listaPuntos LP, listaLineas LL,panelPrincipal H) {
        lp=LP;
        ll=LL;
        h=H;
        initComponents();
        for(int i=0;i<ll.getTamao();i++){
            this.jComboBox1.addItem("Linea"+(i+1));
        }
        for(int i=0;i<ll.getTamao();i++){
            this.seleccionLinea2.addItem("Linea"+(i+1));
        }
        this.setLocation(830,305);//se pone la posicion de la ventana

        addWindowListener(new java.awt.event.WindowAdapter() {
        //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
            @Override
            public void windowClosing(java.awt.event.WindowEvent evt) {
            //cerrando la ventana
                h.refresh(h.getGraphics());
                //borrando lo que esta demas en el panel
            }
        });//aqui se acaba la parte del codigo que borra lo que este demas en el panel antes
         de cerrar la ventana

    }

    @SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jComboBox1 = new javax.swing.JComboBox();
    jButton3 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jPanel2 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    seleccionLinea2 = new javax.swing.JComboBox();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    textoP2L2 = new javax.swing.JLabel();
    textoP1L2 = new javax.swing.JLabel();
    textoPendiente2 = new javax.swing.JLabel();
    textoP1L1 = new javax.swing.JLabel();
    textoPendiente1 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    textoP2L1 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Axioma 3");
    setResizable(false);

    jComboBox1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });

    jButton3.setText("Aplicar");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabel1.setText("Selecciona las dos lneas");

    jPanel1.setBackground(new java.awt.Color(255, 200, 0));

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 30, Short.MAX_VALUE)
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 28, Short.MAX_VALUE)
    );

    jPanel2.setBackground(new java.awt.Color(255, 0, 0));
```

```
javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 30, Short.MAX_VALUE)
);
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 26, Short.MAX_VALUE)
);

jLabel2.setText("linea 2");

jLabel3.setText("linea 1");

seleccionLinea2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seleccionLinea2ActionPerformed(evt);
    }
});

jLabel4.setText("m:");

jLabel5.setText("p1:");

jLabel6.setText("p2:");

textoP2L2.setText("jLabel4");

textoP1L2.setText("jLabel4");

textoPendiente2.setText("jLabel4");

textoP1L1.setText("jLabel4");

textoPendiente1.setText("jLabel4");

jLabel7.setText("p2:");

textoP2L1.setText("jLabel4");

jLabel8.setText("p1:");

jLabel9.setText("m:");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.TRAILING)
```

```
                            .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING,
                                 javax.swing.GroupLayout.DEFAULT_SIZE, 41, Short.MAX_VALUE)
                            .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.LEADING,
                                 javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
                                DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(21, 21, 21)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.TRAILING)
                            .addComponent(jLabel8)
                            .addComponent(jLabel9)
                            .addComponent(jLabel7))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout
                        .Alignment.LEADING)
                            .addComponent(textoP2L1, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
                                 Short.MAX_VALUE)
                            .addComponent(textoP1L1, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
                                 Short.MAX_VALUE)
                            .addComponent(textoPendiente1, javax.swing.GroupLayout.DEFAULT_SIZE,
                             150, Short.MAX_VALUE)))
                    .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                         javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.TRAILING)
                            .addComponent(jLabel2)
                            .addComponent(jPanel1, javax.swing.GroupLayout.
                            PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                                 javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(29, 29, 29)
                        .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel5)
                            .addComponent(jLabel4)
                            .addComponent(jLabel6))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(textoP2L2)
                            .addComponent(textoP1L2, javax.swing.GroupLayout.DEFAULT_SIZE,
                             152, Short.MAX_VALUE)
                            .addComponent(textoPendiente2, javax.swing.GroupLayout.DEFAULT_SIZE,
                             152, Short.MAX_VALUE))))
                .addContainerGap())
            .addComponent(seleccionLinea2, javax.swing.GroupLayout.PREFERRED_SIZE,
             100, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup()
        .addGap(60, 60, 60)
        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 140,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(66, Short.MAX_VALUE))
    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 266,
     Short.MAX_VALUE)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(60, Short.MAX_VALUE)
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 154,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(52, 52, 52))
);
```

```
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(11, 11, 11)
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(8, 8, 8)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel3)
                .addGap(4, 4, 4)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createParallelGroup(
                javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(textoPendiente1)
                    .addComponent(jLabel9))
                .addGroup(layout.createSequentialGroup()
                    .addGap(20, 20, 20)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.BASELINE)
                        .addComponent(textoP1L1, javax.swing.GroupLayout.PREFERRED_SIZE,
                        14, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel8))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(layout.createParallelGroup(javax.
                    swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(textoP2L1)
                        .addComponent(jLabel7)))))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(36, 36, 36)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(textoPendiente2)
                    .addComponent(jLabel4))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(textoP1L2)
                    .addComponent(jLabel5))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(textoP2L2)
                    .addComponent(jLabel6)))
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(seleccionLinea2, javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
```

```
                39, Short.MAX_VALUE)
                .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());//esta linea redibuja el panel
     if(linea2!=null)
        h.dibujaDadaLinea(this.jPanel1.getBackground(), linea2, h.getGraphics());
    linea1=this.darLineaApartirDeString(this.jComboBox1.getSelectedItem().toString());

    this.textoPendiente1.setText(""+valorTrunco(linea1.regresaPendiente().getValor()));
    this.textoP1L1.setText("("+valorTrunco(linea1.getP1().getX())+", "+
    valorTrunco(linea1.getP1().getY())+")");
    this.textoP2L1.setText("("+valorTrunco(linea1.getP2().getX())+", "+
    valorTrunco(linea1.getP2().getY())+")");
    h.dibujaDadaLinea(this.jPanel2.getBackground(), linea1, h.getGraphics());
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    boolean doblar=true;

    if(linea1.esIgualAlaPendienteDe(linea2)){//cuando son paralelas

                X1=(linea1.getP1().getX()+linea2.getP2().getX())/2.0;
                Y1=(linea1.getP1().getY()+linea2.getP2().getY())/2.0;
                X2=100.0;
                Y2=(linea1.regresaPendiente().getValor()*X2)+linea1.ordenadaAlorigen();



    }//cuando son paralelas Cierra
    else{//cuando no son paralelas o no caen en el caso de la cruz que atraviesa el punto 0,0
        /*if(linea1.regresaPendiente().getDeltaX()==0||linea2.regresaPendiente().getDeltaX()==0){
            if(linea1.getDeltaX()==0){

            }
        }*/

        doblar=false;
        CpuntoDoble inter=this.interseccion(linea1, linea2);
        if(inter.getX()<100&&inter.getX()>-100&&inter.getY()<100&&inter.getY()>-100){
            X1=inter.getX();
            Y1=inter.getY();
            double nuevaRectaAngulo=(this.anguloEnRadianesDeRecta(linea1.regresaPendiente())+
            this.anguloEnRadianesDeRecta(linea2.regresaPendiente()))/2.0;
            if(nuevaRectaAngulo==0.0){
                 seleccionDoblez SD= new seleccionDoblez(h,ll,lp,
                 new cLinea(new CpuntoDoble(X1,-100.0),new CpuntoDoble(X1,100.0)),new cLinea(new
                      CpuntoDoble(-100.0,Y1),
                 new CpuntoDoble(100.0,Y1)));
                 SD.setVisible(true);
                 //seleccionDoblez SD= new seleccionDoblez(h,ll,lp,X1,-100,X1,100,-100,Y1,100,Y1);
            }
            else{
```

```
        //X2=100.0;
        X2=1000.0+X1;
        Y2=((double)(Math.tan(nuevaRectaAngulo)*1000.0)+Y1);
        //Y2=((X2-X1)*((linea1.regresaPendiente().getValor()+
        linea2.regresaPendiente().getValor())/2.0))+Y1;
        lineaDobladoSinAnimacion LSA1=new lineaDobladoSinAnimacion(lp,ll,h);
        lineaDobladoSinAnimacion LSA2=new lineaDobladoSinAnimacion(lp,ll,h);
        cLinea lineaDoblez1=LSA1.trazarLineaDeDoblado(X1, Y1, X2, Y2);
        double XP=1000.0+X1;
        double YP=((double)(Math.tan(nuevaRectaAngulo+(Math.PI/2))*1000.0)+Y1);
        //double XP=100.0;
        //double YP=(XP-X1)*(-1.0/((linea1.regresaPendiente().getValor()+
        linea2.regresaPendiente().getValor())/2.0))+Y1;
        cLinea lineaDoblez2=LSA2.trazarLineaDeDoblado(X1, Y1,XP ,YP );
        seleccionDoblez SD= new seleccionDoblez(h,ll,lp,lineaDoblez1,lineaDoblez2);
        SD.setVisible(true);
    }


}
else{
       doblar=true;
       X1=inter.getX();
       Y1=inter.getY();
       double nuevaRectaAngulo=(this.anguloEnRadianesDeRecta(linea1.regresaPendiente())+
       this.anguloEnRadianesDeRecta(linea2.regresaPendiente()))/2.0;
       if(nuevaRectaAngulo==0.0){
           cLinea LD1=new cLinea(new CpuntoDoble((double)X1,(double)-100.0),new
                CpuntoDoble((double)X1,(double)100.0));
           cLinea LD2=new cLinea(new CpuntoDoble((double)-100.0,(double)Y1),new
                CpuntoDoble((double)100.0,(double)Y1));
           if(this.laLineaEstaDentroDeLaHoja(LD1)){
               X1=LD1.getP1().getX();
               Y1=LD1.getP1().getY();
               X2=LD1.getP2().getX();
               Y2=LD1.getP2().getY();
           }
           else{
               X1=LD2.getP1().getX();
               Y1=LD2.getP1().getY();
               X2=LD2.getP2().getX();
               Y2=LD2.getP2().getY();
           }
       }
       else{
           X2=1000.0+X1;
           Y2=((double)(Math.tan(nuevaRectaAngulo)*1000.0)+Y1);
           lineaDobladoSinAnimacion LSA1=new lineaDobladoSinAnimacion(lp,ll,h);
           lineaDobladoSinAnimacion LSA2=new lineaDobladoSinAnimacion(lp,ll,h);
           cLinea lineaDoblez1=LSA1.trazarLineaDeDoblado(X1, Y1, X2, Y2);
           double XP=1000.0+X1;
           double YP=((double)(Math.tan(nuevaRectaAngulo+(Math.PI/2))*1000.0)+Y1);
           cLinea lineaDoblez2=LSA2.trazarLineaDeDoblado(X1, Y1,XP ,YP );
           if(this.laLineaEstaDentroDeLaHoja(lineaDoblez1)){
               if(this.laLineaEstaDentroDeLaHoja(lineaDoblez2)){
                   doblar=false;
                   seleccionDoblez SD= new seleccionDoblez(h,ll,lp,lineaDoblez1,lineaDoblez2);
                   SD.setVisible(true);
               }
```

```
                    X1=lineaDoblez1.getP1().getX();
                    Y1=lineaDoblez1.getP1().getY();
                    X2=lineaDoblez1.getP2().getX();
                    Y2=lineaDoblez1.getP2().getY();
                    /*if(this.laLineaEstaDentroDeLaHoja(lineaDoblez2)){
                        doblar=false;
                        seleccionDoblez SD= new seleccionDoblez(h,ll,lp,lineaDoblez1,lineaDoblez2,
                        X2-X1,Y2-Y1,XP-X1,YP-Y1);
                        SD.setVisible(true);
                    }*/
                }
                else{
                    X1=lineaDoblez2.getP1().getX();
                    Y1=lineaDoblez2.getP1().getY();
                    X2=lineaDoblez2.getP2().getX();
                    Y2=lineaDoblez2.getP2().getY();
                }
            }

        }
    }

    if(doblar){
        doblado D=new doblado(lp,ll,h);
        D.hacerDoblado(Integer.toString((int)X1),Integer.toString((int)Y1),
        Integer.toString((int)X2),Integer.toString((int)Y2),true);
        ll.borrarAlInicio();
        lineaDobladoSinAnimacion lda=new lineaDobladoSinAnimacion(lp,ll,h);
        ll.insertarAlInicio(lda.trazarLineaDeDoblado(X1, Y1, X2, Y2));
    }

    this.dispose();
}

private void seleccionLinea2ActionPerformed(java.awt.event.ActionEvent evt) {
h.refresh(h.getGraphics());
if(linea1!=null)
    h.dibujaDadaLinea(this.jPanel2.getBackground(), linea1, h.getGraphics());
linea2=this.darLineaApartirDeString(this.seleccionLinea2.getSelectedItem().toString());
this.textoPendiente2.setText(""+valorTrunco(linea2.regresaPendiente().getValor()));
this.textoP1L2.setText("("+valorTrunco(linea2.getP1().getX())+", "+
valorTrunco(linea2.getP1().getY())+")");
this.textoP2L2.setText("("+valorTrunco(linea2.getP2().getX())+", "+
valorTrunco(linea2.getP2().getY())+")");
h.dibujaDadaLinea(this.jPanel1.getBackground(), linea2,h.getGraphics());
}

public cLinea darLineaApartirDeString(String sLinea){
        cLinea[] lista=ll.regresaLista();
        return lista[Integer.parseInt(sLinea.substring(5,6))-1];

}

public CpuntoDoble interseccion(cLinea L1,cLinea L2){
    double X;
    double Y;
    if(L1.regresaPendiente().getEsVertical()||L2.regresaPendiente().getEsVertical()){
        if(L1.regresaPendiente().getEsVertical()){
            X=L1.getP1().getX();
```

```
            Y=(L2.regresaPendiente().getValor()*X)-L2.ordenadaAlorigen();
            //Y=((((double)(L2.getDeltaY())/(double)(L2.getDeltaX()))*(((double)X-
            (double)L2.getP1().getX()))+((double)L2.getP1().getY()));
        }
        else{
            X=L2.getP1().getX();
            Y=(L1.regresaPendiente().getValor()*X)-L1.ordenadaAlorigen();
            //Y=((((double)(L1.getDeltaY())/(double)(L1.getDeltaX()))*(((double)X-
            (double)L1.getP1().getX()))+((double)L1.getP1().getY()));
        }
    }
    else{
        X=(((-L1.ordenadaAlorigen())+(L2.ordenadaAlorigen()))
                /(L1.regresaPendiente().getValor()-L2.regresaPendiente().getValor()));
        Y=((double)(((double)(L1.regresaPendiente().getValor()*X))+L1.ordenadaAlorigen()));
    }
    return new CpuntoDoble(X,Y);
}


public double anguloEnRadianesDeRecta(cPendiente M){
    if(M.getEsVertical()){
        return Math.PI/2;
    }
    else{
        return Math.atan(M.getValor());
    }
}

public boolean laLineaEstaDentroDeLaHoja(cLinea l){
    if(l.getP1().getX()<=100&&l.getP1().getY()<=100&&l.getP2().getX()<=100&&l.getP2
    ().getY()<=100){
        return true;
    }
    else{
        return false;
    }
}

public String valorTrunco(double vOriginal){
    String V=Double.toString(vOriginal);
    int t=V.length();
    if(t>=7)
        return V.substring(0,7);
    else
        return V;
}

    // Variables declaration - do not modify
    private javax.swing.JButton jButton3;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
```

```
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JComboBox seleccionLinea2;
    private javax.swing.JLabel textoP1L1;
    private javax.swing.JLabel textoP1L2;
    private javax.swing.JLabel textoP2L1;
    private javax.swing.JLabel textoP2L2;
    private javax.swing.JLabel textoPendiente1;
    private javax.swing.JLabel textoPendiente2;
    // End of variables declaration
}
```

A.27. ventanaAxioma4

```
package sahco;
import java.awt.Color;

public class ventanaAxioma4 extends javax.swing.JFrame {
    private listaPuntos lp;//variable de clase que guardara la lista de puntos
    private panelPrincipal h;//variable de clase que contendra el panel en el que se dibuja
    private listaLineas ll;//variable de clase que guardara la lista de lineas
    private cPunto punto;
    private cLinea linea;

    private int x1,y1;
    double x2,y2;

    public ventanaAxioma4(listaPuntos LP, listaLineas LL,panelPrincipal H) {
        lp=LP;
        //se asigna la lista de puntos con lo que se le pasa al constructor cuando
        se crea un nuevo onbjeto tipo ventanaAxioma4
        ll=LL;//se asigna el panel con lo que se le pasa al constructor cuando se
        crea un nuevo onbjeto tipo ventanaAxioma4
        h=H;//se asigna la lista de lineas con lo que se le pasa al constructor cuando
        se crea un nuevo objeto tipo ventanaAxioma4
        initComponents();
        for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador(combo box)
            this.seleccionador1.addItem("Punto"+(i+1));//cada uno de los puntos
        }//este seleccionador sirve para escoger los puntos
        for(int i=0;i<ll.getTamao();i++){//se ponen las lineas en el seleccionador2
            this.seleccionador2.addItem("Linea"+(i+1));//cada una de las lineas
        }
        this.setLocation(830,305);//se pone la posicion de la ventana

        addWindowListener(new java.awt.event.WindowAdapter() {
        //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
            @Override
            public void windowClosing(java.awt.event.WindowEvent evt) {
            //cerrando la ventana
                h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
            }
        });//aqui se acaba la parte del codigo que borra lo que este demas en el panel
         antes de cerrar la ventana
    }
```

```
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    seleccionador1 = new javax.swing.JComboBox();
    seleccionador2 = new javax.swing.JComboBox();
    aplicarAxioma4 = new javax.swing.JButton();
    panelPunto = new javax.swing.JPanel();
    panelLinea = new javax.swing.JPanel();
    eligePunto = new javax.swing.JLabel();
    pendiente = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    P1 = new javax.swing.JLabel();
    P2 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Axioma 4");
    setResizable(false);

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabel1.setText("Escoge la lnea perpendicular al doblez");

    seleccionador1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            seleccionador1ActionPerformed(evt);
        }
    });

    seleccionador2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            seleccionador2ActionPerformed(evt);
        }
    });

    aplicarAxioma4.setText("Aplicar");
    aplicarAxioma4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            aplicarAxioma4ActionPerformed(evt);
        }
    });

    panelPunto.setBackground(new java.awt.Color(0, 255, 0));

    javax.swing.GroupLayout panelPuntoLayout = new javax.swing.GroupLayout(panelPunto);
    panelPunto.setLayout(panelPuntoLayout);
    panelPuntoLayout.setHorizontalGroup(
        panelPuntoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 23, Short.MAX_VALUE)
    );
    panelPuntoLayout.setVerticalGroup(
        panelPuntoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 21, Short.MAX_VALUE)
    );

    panelLinea.setBackground(new java.awt.Color(255, 0, 0));
```

```
javax.swing.GroupLayout panelLineaLayout = new javax.swing.GroupLayout(panelLinea);
panelLinea.setLayout(panelLineaLayout);
panelLineaLayout.setHorizontalGroup(
    panelLineaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);
panelLineaLayout.setVerticalGroup(
    panelLineaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);

eligePunto.setText("Punto");

pendiente.setText("pendiente");

jLabel3.setText("Elige Punto");

jLabel4.setText("Elige Lnea");

P1.setText("P1");

P2.setText("P2");

jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
jLabel2.setText("y el punto por el que pasar");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(74, 74, 74)
                .addComponent(jLabel3))
            .addGroup(layout.createSequentialGroup()
                .addGap(75, 75, 75)
                .addComponent(jLabel4))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(panelPunto, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(panelLinea, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(aplicarAxioma4, javax.swing.GroupLayout.PREFERRED_SIZE, 182,
                         javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(seleccionador2, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
                                 Short.MAX_VALUE)
                            .addComponent(seleccionador1, 0, 96, Short.MAX_VALUE))
                        .addGap(33, 33, 33)
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING)
```

```
                        .addComponent(P1)
                        .addComponent(eligePunto)
                        .addComponent(P2)
                        .addComponent(pendiente)))))
            .addGroup(layout.createSequentialGroup()
                .addGap(35, 35, 35)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 262,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(60, 60, 60)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 186,
                    javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(16, Short.MAX_VALUE))
    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 313, Short.MAX_VALUE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 15,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jLabel3)
        .addGap(8, 8, 8)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(seleccionador1, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(eligePunto))
            .addComponent(panelPunto, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(layout.createSequentialGroup()
                .addGap(37, 37, 37)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(pendiente))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(seleccionador2)
                    .addComponent(P1)))
            .addGroup(layout.createSequentialGroup()
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(panelLinea, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(P2)
        .addGap(23, 23, 23)
        .addComponent(aplicarAxioma4, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
);

pack();
```

```
    }// </editor-fold>

private void seleccionador1ActionPerformed(java.awt.event.ActionEvent evt) {
//aqui se define lo que pasa cuando se usa el seleccionador de puntos
    h.refresh(h.getGraphics());//esta linea redibuja el panel
    if(linea!=null)//si ya hay una linea fijada, esta se pinta de rojo
        h.dibujaDadaLinea(panelLinea.getBackground(), linea, h.getGraphics());
    punto=darPuntoApartirDeString(seleccionador1.getSelectedItem().toString());
    //se obtiene el punto que se esta eligiendo en el seleccionador, mediante
    el metodo que ingresa a la lista de puntos
    eligePunto.setText("("+valorTrunco(punto.getX())+", "+valorTrunco(punto.getY())+")");
    h.dibujaPuntoDeColor(panelPunto.getBackground(), punto,h.getGraphics());
    //dibuja el punto que se esta eligiendo en el seleccionador
}

private void seleccionador2ActionPerformed(java.awt.event.ActionEvent evt) {
h.refresh(h.getGraphics());//esta linea redibuja el panel
    if(punto!=null)//cuando ya se eligio el punto
        h.dibujaPuntoDeColor(panelPunto.getBackground(), punto,h.getGraphics());
        //dibuja el punto ,ya fijado, en rojo
    linea=darLineaApartirDeString(seleccionador2.getSelectedItem().toString());
    //se escoge la linea del seleccionador
    pendiente.setText("m: "+valorTrunco(linea.regresaPendiente().getValor()));
    P1.setText("p1: ("+valorTrunco(linea.getP1().getX())+", "+
    valorTrunco(linea.getP1().getY())+")");
    P2.setText("p2: ("+valorTrunco(linea.getP2().getX())+", "+
    valorTrunco(linea.getP2().getY())+")");
    h.dibujaDadaLinea(panelLinea.getBackground(), linea,h.getGraphics());
}

private void aplicarAxioma4ActionPerformed(java.awt.event.ActionEvent evt) {
//esto es lo que sucede cunado se presiona el boton de aplicar
    //se buscan los puntos pra hacer el doblez mediante la ecuaci\'on punto pendiente
    //el punto es el elegido por el usuario y se guarda en x1 y y1,
     coordenadas del nuevo doblez

    double nuevaPendiente=-1.0/linea.regresaPendiente().getValor();
    //valor de la nueva pendiente en tipo doble
    x1=punto.getX();//x del punto elegido
    double x1Doble=(double)x1;//valor de x del punto elegido en doble
    y1=punto.getY();//y del punto elegido
    double y1Doble=(double)y1;//valor de y del punto elegido en doble
    x2=1000.0;//punto arbitrario
    y2=((nuevaPendiente*(1000.0-x1Doble))+y1Doble);
    //usando la ecuaci\'on punto pendiente se obtiene la y correspondiente a la x = 1000
    doblado D=new doblado(lp,ll,h);//se crea un objeto tipo doblado
    D.hacerDoblado(Integer.toString(x1),Integer.toString(y1),Integer.toString((int)x2),
    Integer.toString((int)y2),true);//se invoca el metodo que hace la animacion de
    doblado e inserta la nueva linea a la lista de lineas
    ll.borrarAlInicio();
    lineaDobladoSinAnimacion lda=new lineaDobladoSinAnimacion(lp,ll,h);
        ll.insertarAlInicio(lda.trazarLineaDeDoblado(x1, y1, x2, y2));

    this.dispose();//cierra la ventana
}

public cPunto darPuntoApartirDeString(String sPunto){
//este metodo regresa un punto de la lista de puntos, a partir de su nombre (sPunto)
       cPunto[] lista=lp.regresaLista();//el arrreglo contendra la lista de puntos
```

```
        return lista[Integer.parseInt(sPunto.substring(5,6))-1];
        //dependiendo de cual punto se quiera (sPunto) este se regresa

}
public cLinea darLineaApartirDeString(String sLinea){
//este metodo regresa una linea de la lista de lineas, a partir de su nombre (sLinea)
        cLinea[] lista=ll.regresaLista();//arreglo que contiene la lista de lineas
        return lista[Integer.parseInt(sLinea.substring(5,6))-1];
        //se regresa la linea seleccionada (sLinea)

}

public String valorTrunco(double vOriginal){
    String V=Double.toString(vOriginal);
    int t=V.length();
    if(t>=7)
        return V.substring(0,7);
    else
        return V;
}

    // Variables declaration - do not modify
    private javax.swing.JLabel P1;
    private javax.swing.JLabel P2;
    private javax.swing.JButton aplicarAxioma4;
    private javax.swing.JLabel eligePunto;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JPanel panelLinea;
    private javax.swing.JPanel panelPunto;
    private javax.swing.JLabel pendiente;
    private javax.swing.JComboBox seleccionador1;
    private javax.swing.JComboBox seleccionador2;
    // End of variables declaration
}
```

A.28. ventanaAxioma5

```
package sahco;
import java.awt.Color;

public class ventanaAxioma5 extends javax.swing.JFrame {
    private listaPuntos lp;
    private panelPrincipal h;
    private listaLineas ll;
    private cLinea linea;
    private cPunto punto1;
    private cPunto punto2;

    public ventanaAxioma5(listaPuntos LP, panelPrincipal H, listaLineas LL) {
        lp=LP;
        h=H;
        ll=LL;
        initComponents();
```

```
    for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador(combo box)
        comboBoxP1.addItem("Punto"+(i+1));//cada uno de los puntos
    }//este seleccionador sirve para escoger los puntos

    for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador(combo box)
        comboBoxP2.addItem("Punto"+(i+1));//cada uno de los puntos
    }//este seleccionador sirve para escoger los puntos

    for(int i=0;i<ll.getTamao();i++){//se ponen las lineas en el seleccionador
        comboBoxLinea.addItem("Linea"+(i+1));//cada una de las lineas
    }
    this.setLocation(830,305);

    addWindowListener(new java.awt.event.WindowAdapter() {
    //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
        @Override
        public void windowClosing(java.awt.event.WindowEvent evt) {
        //cerrando la ventana
            h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
        }
    });//aqui se acaba la parte del codigo que borra lo que este demas en el panel
    antes de cerrar la ventana
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel2 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    comboBoxLinea = new javax.swing.JComboBox();
    jLabel3 = new javax.swing.JLabel();
    comboBoxP1 = new javax.swing.JComboBox();
    jLabel4 = new javax.swing.JLabel();
    comboBoxP2 = new javax.swing.JComboBox();
    aplicarA5 = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel();
    jPanel2 = new javax.swing.JPanel();
    jPanel3 = new javax.swing.JPanel();
    Pendiente = new javax.swing.JLabel();
    P1 = new javax.swing.JLabel();
    P2 = new javax.swing.JLabel();
    Punto1 = new javax.swing.JLabel();
    Punto2 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Axioma 5");
    setResizable(false);

    jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
    jLabel2.setText("Selecciona los puntos y la lnea");

    comboBoxLinea.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            comboBoxLineaActionPerformed(evt);
        }
    });
```

```
jLabel3.setText("Linea1");

comboBoxP1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxP1ActionPerformed(evt);
    }
});

jLabel4.setText("Punto1");

comboBoxP2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxP2ActionPerformed(evt);
    }
});

aplicarA5.setText("Aplicar");
aplicarA5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aplicarA5ActionPerformed(evt);
    }
});

jPanel1.setBackground(new java.awt.Color(0, 255, 255));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 22, Short.MAX_VALUE)
);

jPanel2.setBackground(new java.awt.Color(255, 0, 0));

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 25, Short.MAX_VALUE)
);
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 23, Short.MAX_VALUE)
);

jPanel3.setBackground(new java.awt.Color(255, 175, 175));

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 28, Short.MAX_VALUE)
);
jPanel3Layout.setVerticalGroup(
```

```
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 26, Short.MAX_VALUE)
);

Pendiente.setText("Pendiente");

P1.setText("P1");

P2.setText("P2");

Punto1.setText("Punto1");

Punto2.setText("Punto2");

jLabel5.setText("Punto 2");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup
        (javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup
                (javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                    layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(Pendiente)
                            .addComponent(comboBoxLinea,
                            javax.swing.GroupLayout.PREFERRED_SIZE, 77,
                                 javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 43,
                             Short.MAX_VALUE)
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                 layout.createSequentialGroup()
                                .addComponent(comboBoxP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                                83, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap
                                (javax.swing.LayoutStyle.ComponentPlacement.RELATED))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                 layout.createSequentialGroup()
                                .addComponent(Punto1)
                                .addGap(11, 11, 11)))
                        .addGroup(layout.createParallelGroup
                        (javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addPreferredGap
                                (javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(Punto2)
                                .addGap(20, 20, 20))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(54, 54, 54)
```

```
                                .addComponent(comboBoxP2,
                                 javax.swing.GroupLayout.PREFERRED_SIZE, 82,
                                      javax.swing.GroupLayout.PREFERRED_SIZE))))
                        .addGroup(layout.createSequentialGroup()
                            .addGap(13, 13, 13)
                            .addGroup(layout.createParallelGroup
                            (javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                                     javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel3))
                            .addGap(112, 112, 112)
                            .addGroup(layout.createParallelGroup
                            (javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel4)
                                .addComponent(jPanel2,
                                 javax.swing.GroupLayout.PREFERRED_SIZE,
                                 javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(layout.createParallelGroup
                            (javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(layout.createSequentialGroup()
                                    .addGap(92, 92, 92)
                                    .addComponent(jLabel5))
                                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                     layout.createSequentialGroup()
                                    .addGap(101, 101, 101)
                                    .addComponent(jPanel3,
                                    javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE,
                                         javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addGap(20, 20, 20))
                        .addComponent(aplicarA5, javax.swing.GroupLayout.Alignment.TRAILING,
                             javax.swing.GroupLayout.PREFERRED_SIZE, 206,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(51, 51, 51))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(P1, javax.swing.GroupLayout.DEFAULT_SIZE, 19,
                     Short.MAX_VALUE)
                    .addGap(371, 371, 371))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(P2)
                    .addContainerGap(388, Short.MAX_VALUE))))
        .addGroup(layout.createSequentialGroup()
            .addGap(99, 99, 99)
            .addComponent(jLabel2)
            .addContainerGap(126, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 400,
                 javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(14, 14, 14)
            .addComponent(jLabel2)
            .addGap(14, 14, 14)
```

```
.addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE,
     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addGroup(layout.createSequentialGroup()
        .addGap(33, 33, 33)
        .addGroup(layout.createParallelGroup
        (javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel4))
        .addGap(18, 18, 18))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel3)
        .addGap(18, 18, 18)))
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createParallelGroup
    (javax.swing.GroupLayout.Alignment.TRAILING)
        .addGroup(layout.createSequentialGroup()
            .addGap(1, 1, 1)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(9, 9, 9))
        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
        layout.createSequentialGroup()
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
    .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(6, 6, 6)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(
        javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(comboBoxLinea)
            .addComponent(comboBoxP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE,
             javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(29, 29, 29))
    .addGroup(layout.createSequentialGroup()
        .addComponent(comboBoxP1, javax.swing.GroupLayout.PREFERRED_SIZE,
             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(Pendiente, javax.swing.GroupLayout.DEFAULT_SIZE,
         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(Punto2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(Punto1, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(P1, javax.swing.GroupLayout.DEFAULT_SIZE, 14, Short.MAX_VALUE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(P2, javax.swing.GroupLayout.DEFAULT_SIZE, 14, Short.MAX_VALUE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(aplicarA5, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(19, 19, 19))
        );

        pack();
    }// </editor-fold>

private void comboBoxLineaActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(punto1!=null)
        h.dibujaPuntoDeColor(Color.red,punto1,h.getGraphics());
    if(punto2!=null)
        h.dibujaPuntoDeColor(Color.PINK, punto2,h.getGraphics());
    linea=darLineaApartirDeString(comboBoxLinea.getSelectedItem().toString());
    Pendiente.setText("m: "+valorTrunco(linea.regresaPendiente().getValor()));
    P1.setText("p1: ("+valorTrunco(linea.getP1().getX())+", "+
    valorTrunco(linea.getP1().getY())+")");
    P2.setText("p2: ("+valorTrunco(linea.getP2().getX())+", "+
    valorTrunco(linea.getP2().getY())+")");
    h.dibujaDadaLinea(Color.CYAN, linea, h.getGraphics());
}

private void comboBoxP1ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(linea!=null)
        h.dibujaDadaLinea(Color.CYAN, linea,h.getGraphics());
    if(punto2!=null)
        h.dibujaPuntoDeColor(Color.PINK, punto2,h.getGraphics());
    punto1=darPuntoApartirDeString(comboBoxP1.getSelectedItem().toString());
    Punto1.setText("("+valorTrunco(punto1.getX())+", "+valorTrunco(punto1.getY())+")");
    h.dibujaPuntoDeColor(Color.red, punto1,h.getGraphics());
}

private void comboBoxP2ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(linea!=null)
        h.dibujaDadaLinea(Color.CYAN, linea,h.getGraphics());
    if(punto1!=null)
        h.dibujaPuntoDeColor(Color.red, punto1,h.getGraphics());
    punto2=darPuntoApartirDeString(comboBoxP2.getSelectedItem().toString());
    Punto2.setText("("+valorTrunco(punto2.getX())+", "+valorTrunco(punto2.getY())+")");
    h.dibujaPuntoDeColor(Color.PINK, punto2,h.getGraphics());
}

private void aplicarA5ActionPerformed(java.awt.event.ActionEvent evt) {
    CpuntoDoble P1=new CpuntoDoble(punto1.getX(),punto1.getY());
    CpuntoDoble P2=new CpuntoDoble(punto2.getX(),punto2.getY());
    cParabola parabola= new cParabola(P1,linea,0.0);
    double anguloRotacion=-(this.anguloEnRadianesDeRecta(linea.regresaPendiente()));
    if(parabola.esParabola()){
        P2=P2.rotar(anguloRotacion);
        cParabola parabolaRotada=parabola.rotarParabola(anguloRotacion);
        double p=parabolaRotada.obtenerEcuacion().getP();
        double q=parabolaRotada.obtenerEcuacion().getQ();
        double r=parabolaRotada.obtenerEcuacion().getR();
        double u=P2.getX();
        double w=P2.getY();
        double a=p;
```

```
double b=-2.0*p*u;
double c=(-q*u)+w-r;
resuelveEcuacion2doGrado ecuaci\'on=new resuelveEcuacion2doGrado(a,b,c);
ecuacion.resuelve();
if(ecuacion.getDiscriminante()>0){
    double x0=ecuacion.getS1();
    double y0=p*x0*x0+q*x0+r;
    double x0p=1000.0+x0;
    double y0p=((2.0*p*x0+q)*1000.0)+y0;

    double x1=ecuacion.getS2();
    double y1=p*x1*x1+q*x1+r;
    double x1p=1000.0+x1;
    double y1p=((2.0*p*x1+q)*1000.0)+y1;

    CpuntoDoble Punto1=new CpuntoDoble(x0,y0).rotar(-anguloRotacion);
    CpuntoDoble Punto2=new CpuntoDoble(x0p,y0p).rotar(-anguloRotacion);

    CpuntoDoble nuevoP1=new CpuntoDoble(x1,y1).rotar(-anguloRotacion);
    CpuntoDoble nuevoP2=new CpuntoDoble(x1p,y1p).rotar(-anguloRotacion);

    x0=Punto1.getX();
    y0=Punto1.getY();
    x0p=Punto2.getX();
    y0p=Punto2.getY();

    x1=nuevoP1.getX();
    y1=nuevoP1.getY();
    x1p=nuevoP2.getX();
    y1p=nuevoP2.getY();

    lineaDobladoSinAnimacion LSA1=new lineaDobladoSinAnimacion(lp,ll,h);
    lineaDobladoSinAnimacion LSA2=new lineaDobladoSinAnimacion(lp,ll,h);
    cLinea lineaDoblez1=LSA1.trazarLineaDeDoblado(x0, y0,x0p, y0p);
    cLinea lineaDoblez2=LSA2.trazarLineaDeDoblado(x1, y1,x1p ,y1p );
    System.out.println("linea1p1x="+lineaDoblez1.getP1().getX());
    System.out.println("linea1p1y="+lineaDoblez1.getP1().getY());
    System.out.println("linea1p2x="+lineaDoblez1.getP2().getX());
    System.out.println("linea1p2y="+lineaDoblez1.getP2().getY());
    System.out.println("pendiente l1="+lineaDoblez1.regresaPendiente().getValor());

    System.out.println("linea2p1x="+lineaDoblez2.getP1().getX());
    System.out.println("linea2p1y="+lineaDoblez2.getP1().getY());
    System.out.println("linea2p2x="+lineaDoblez2.getP2().getX());
    System.out.println("linea2p2y="+lineaDoblez2.getP2().getY());
    System.out.println("pendiente l2="+lineaDoblez2.regresaPendiente().getValor());
    seleccionDoblez SD= new seleccionDoblez(h,ll,lp,lineaDoblez1,lineaDoblez2);
    SD.setVisible(true);
    this.dispose();

}
else{
    if(ecuacion.getDiscriminante()==0){
        double x0=ecuacion.getS1();
        double y0=x0*x0+q*x0+r;
        double x0p=1000.0+x0;
        double y0p=((2.0*p*x0+q)*1000.0)+y0;

        CpuntoDoble Punto1=new CpuntoDoble(x0,y0).rotar(-anguloRotacion);
```

```
            CpuntoDoble Punto2=new CpuntoDoble(x0p,y0p).rotar(-anguloRotacion);

            x0=Punto1.getX();
            y0=Punto1.getY();
            x0p=Punto2.getX();
            y0p=Punto2.getY();

            doblado D=new doblado(lp,ll,h);
            D.hacerDoblado(Integer.toString((int)x0),Integer.toString((int)y0),
            Integer.toString((int)x0p),Integer.toString((int)y0p),true);
            ll.borrarAlInicio();
            lineaDobladoSinAnimacion lda=new lineaDobladoSinAnimacion(lp,ll,h);
            ll.insertarAlInicio(lda.trazarLineaDeDoblado(x0,y0,x0p,y0p));
        }
        else{
            ventanaError v=new ventanaError("El axioma no se puede aplicar",
            "Prueba seleccionando otros puntos u otra linea");
            v.setVisible(true);
        }
        this.dispose();
    }
    System.out.println("p="+p);
    System.out.println("q="+q);
    System.out.println("r="+r);
    System.out.println("u="+u);
    System.out.println("w="+w);
    System.out.println("a="+a);
    System.out.println("b="+b);
    System.out.println("c="+c);
    System.out.println("ecuacion solucion x1="+ecuacion.getS1());
    System.out.println("ecuacion solucion x2="+ecuacion.getS2());
    System.out.println("angulo Rotacion="+anguloRotacion);


}
else{
    double x1=punto2.getX();
    double y1=punto2.getY();
    double x2=x1+1000.0;
    double y2=y1+(((-1.0)/(linea.regresaPendiente().getValor()))*1000.0);
    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)x1),Integer.toString((int)y1),
    Integer.toString((int)x2),Integer.toString((int)y2),true);
    ll.borrarAlInicio();
    lineaDobladoSinAnimacion LSA1=new lineaDobladoSinAnimacion(lp,ll,h);
    ll.insertarAlInicio(LSA1.trazarLineaDeDoblado(x1, y1, x2, y2));
    this.dispose();
}

}

    public cPunto darPuntoApartirDeString(String sPunto){
    //este metodo regresa un punto de la lista de puntos, a partir de su nombre (sPunto)
        cPunto[] lista=lp.regresaLista();//el arrreglo contendra la lista de puntos
        return lista[Integer.parseInt(sPunto.substring(5,6))-1];
        //dependiendo de cual punto se quiera (sPunto) este se regresa

    }
```

```
    public cLinea darLineaApartirDeString(String sLinea){
    //este metodo regresa una linea de la lista de lineas, a partir de su nombre (sLinea)
        cLinea[] lista=ll.regresaLista();//arreglo que contiene la lista de lineas
        return lista[Integer.parseInt(sLinea.substring(5,6))-1];
        //se regresa la linea seleccionada (sLinea)

    }

    public double anguloEnRadianesDeRecta(cPendiente M){
    if(M.getEsVertical()){
        return Math.PI/2;
    }
    else{
        return Math.atan(M.getValor());
    }
}

 public String valorTrunco(double vOriginal){
    String V=Double.toString(vOriginal);
    int t=V.length();
    if(t>=7)
        return V.substring(0,7);
    else
        return V;
}


    // Variables declaration - do not modify
    private javax.swing.JLabel P1;
    private javax.swing.JLabel P2;
    private javax.swing.JLabel Pendiente;
    private javax.swing.JLabel Punto1;
    private javax.swing.JLabel Punto2;
    private javax.swing.JButton aplicarA5;
    private javax.swing.JComboBox comboBoxLinea;
    private javax.swing.JComboBox comboBoxP1;
    private javax.swing.JComboBox comboBoxP2;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JSeparator jSeparator1;
    // End of variables declaration
}
```

A.29. ventanaAxioma6

```
package sahco;
import java.awt.Color;
import java.lang.Math;

public class ventanaAxioma6 extends javax.swing.JFrame {

    private listaPuntos lp;
```

```
private panelPrincipal h;
private listaLineas ll;

private cLinea linea1;
private cLinea linea2;
private cPunto punto1;
private cPunto punto2;

public ventanaAxioma6(listaPuntos LP, panelPrincipal H, listaLineas LL) {
    lp=LP;
    h=H;
    ll=LL;
    initComponents();
    for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador(combo box)
        comboBoxP1.addItem("Punto"+(i+1));//cada uno de los puntos
    }//este seleccionador sirve para escoger los puntos

    for(int i=0;i<lp.getTamao();i++){//se ponen los puntos en el seleccionador(combo box)
        comboBoxP2.addItem("Punto"+(i+1));//cada uno de los puntos
    }//este seleccionador sirve para escoger los puntos

    for(int i=0;i<ll.getTamao();i++){//se ponen las lineas en el seleccionador
        comboBoxLinea1.addItem("Linea"+(i+1));//cada una de las lineas
    }
    for(int i=0;i<ll.getTamao();i++){//se ponen las lineas en el seleccionador
        comboBoxLinea2.addItem("Linea"+(i+1));//cada una de las lineas
    }
    this.setLocation(800,130);
    addWindowListener(new java.awt.event.WindowAdapter() {
    //este metodo borrara lo que se deba cuando se presione el tache(X) de cerrar
        @Override
        public void windowClosing(java.awt.event.WindowEvent evt) {
        //cerrando la ventana
            h.refresh(h.getGraphics());//borrando lo que esta demas en el panel
        }
    });//aqui se acaba la parte del codigo que borra lo que este demas en el panel
     antes de cerrar la ventana
}
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jSeparator1 = new javax.swing.JSeparator();
    jLabel2 = new javax.swing.JLabel();
    comboBoxP1 = new javax.swing.JComboBox();
    comboBoxP2 = new javax.swing.JComboBox();
    jLabel3 = new javax.swing.JLabel();
    comboBoxLinea1 = new javax.swing.JComboBox();
    jLabel4 = new javax.swing.JLabel();
    comboBoxLinea2 = new javax.swing.JComboBox();
    jLabel5 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    panelP1 = new javax.swing.JPanel();
    panelP2 = new javax.swing.JPanel();
    panelL1 = new javax.swing.JPanel();
    panelL2 = new javax.swing.JPanel();
    textoP1 = new javax.swing.JLabel();
    textoP2 = new javax.swing.JLabel();
    textoPendienteL1 = new javax.swing.JLabel();
    textoP1L1 = new javax.swing.JLabel();
```

```
textoP2L1 = new javax.swing.JLabel();
textoPendiente2 = new javax.swing.JLabel();
textoP1L2 = new javax.swing.JLabel();
textoP2L2 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Axioma 6");
setResizable(false);

jLabel2.setText("punto1");

comboBoxP1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxP1ActionPerformed(evt);
    }
});

comboBoxP2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxP2ActionPerformed(evt);
    }
});

jLabel3.setText("punto2");

comboBoxLinea1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxLinea1ActionPerformed(evt);
    }
});

jLabel4.setText("linea1");

comboBoxLinea2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxLinea2ActionPerformed(evt);
    }
});

jLabel5.setText("linea2");

jButton1.setText("Aplicar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

panelP1.setBackground(new java.awt.Color(255, 0, 0));

javax.swing.GroupLayout panelP1Layout = new javax.swing.GroupLayout(panelP1);
panelP1.setLayout(panelP1Layout);
panelP1Layout.setHorizontalGroup(
    panelP1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);
panelP1Layout.setVerticalGroup(
    panelP1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
    .addGap(0, 20, Short.MAX_VALUE)
);

panelP2.setBackground(new java.awt.Color(51, 255, 51));

javax.swing.GroupLayout panelP2Layout = new javax.swing.GroupLayout(panelP2);
panelP2.setLayout(panelP2Layout);
panelP2Layout.setHorizontalGroup(
    panelP2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);
panelP2Layout.setVerticalGroup(
    panelP2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);

panelL1.setBackground(new java.awt.Color(255, 0, 255));

javax.swing.GroupLayout panelL1Layout = new javax.swing.GroupLayout(panelL1);
panelL1.setLayout(panelL1Layout);
panelL1Layout.setHorizontalGroup(
    panelL1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);
panelL1Layout.setVerticalGroup(
    panelL1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);

panelL2.setBackground(new java.awt.Color(0, 255, 255));

javax.swing.GroupLayout panelL2Layout = new javax.swing.GroupLayout(panelL2);
panelL2.setLayout(panelL2Layout);
panelL2Layout.setHorizontalGroup(
    panelL2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);
panelL2Layout.setVerticalGroup(
    panelL2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 20, Short.MAX_VALUE)
);

textoP1.setText("jLabel6");

textoP2.setText("jLabel6");

textoPendienteL1.setText("jLabel6");

textoP1L1.setText("jLabel6");

textoP2L1.setText("jLabel6");

textoPendiente2.setText("jLabel6");

textoP1L2.setText("jLabel6");

textoP2L2.setText("jLabel6");

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
```

```
jLabel1.setText("Selecciona los puntos y las lneas");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(panelL2, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel5)
                .addGap(28, 28, 28)
                .addComponent(comboBoxLinea2, 0, 93, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(panelL1, javax.swing.GroupLayout.PREFERRED_SIZE,
                                     javax.swing.GroupLayout.DEFAULT_SIZE,
                                 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                RELATED)
                                .addComponent(jLabel4))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(panelP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                                     javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                RELATED)
                                .addComponent(jLabel3))))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(12, 12, 12)
                        .addComponent(panelP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                             javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabel2)))
                .addGap(22, 22, 22)
                .addGroup(layout.createParallelGroup
                (javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(comboBoxP1, 0,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(comboBoxLinea1, 0,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(comboBoxP2, 0, 90, Short.MAX_VALUE))))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(textoP1L1, javax.swing.GroupLayout.DEFAULT_SIZE, 119,
             Short.MAX_VALUE)
            .addComponent(textoPendienteL1, javax.swing.GroupLayout.DEFAULT_SIZE,
            119, Short.MAX_VALUE)
            .addComponent(textoP2L1, javax.swing.GroupLayout.DEFAULT_SIZE, 119,
             Short.MAX_VALUE)
```

```
            .addComponent(textoPendiente2, javax.swing.GroupLayout.DEFAULT_SIZE,
            119, Short.MAX_VALUE)
            .addComponent(textoP1L2, javax.swing.GroupLayout.DEFAULT_SIZE, 119,
             Short.MAX_VALUE)
            .addComponent(textoP2L2, javax.swing.GroupLayout.DEFAULT_SIZE, 119,
            Short.MAX_VALUE)
            .addComponent(textoP2, javax.swing.GroupLayout.DEFAULT_SIZE, 119, Short.MAX_VALUE)
            .addComponent(textoP1, javax.swing.GroupLayout.DEFAULT_SIZE, 119, Short.MAX_VALUE))
        .addContainerGap())
    .addGroup(layout.createSequentialGroup()
        .addGap(71, 71, 71)
        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 168,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(83, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap(57, Short.MAX_VALUE)
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 213,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(52, 52, 52))
    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(24, 24, 24)
        .addComponent(jLabel1)
        .addGap(18, 18, 18)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 13,
             javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(panelP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(comboBoxP1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(textoP1)))
        .addGap(26, 26, 26)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(panelP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3)
                .addComponent(comboBoxP2, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(textoP2)))
        .addGap(34, 34, 34)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(panelL1, javax.swing.GroupLayout.PREFERRED_SIZE,
                 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(comboBoxLinea1, javax.swing.GroupLayout.PREFERRED_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(textoPendienteL1)))
        .addGap(11, 11, 11)
        .addComponent(textoP1L1)
        .addGap(18, 18, 18)
```

```
                .addComponent(textoP2L1)
                .addGap(29, 29, 29)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(panelL2, javax.swing.GroupLayout.PREFERRED_SIZE,
                         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(comboBoxLinea2, javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel5)
                        .addComponent(textoPendiente2)))
                .addGap(11, 11, 11)
                .addComponent(textoP1L2)
                .addGap(18, 18, 18)
                .addComponent(textoP2L2)
                .addGap(28, 28, 28)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
                     javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

private void comboBoxP1ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(linea1!=null)
        h.dibujaDadaLinea(panelL1.getBackground(), linea1,h.getGraphics());
    if(linea2!=null)
        h.dibujaDadaLinea(panelL2.getBackground(), linea2,h.getGraphics());
    if(punto2!=null)
        h.dibujaPuntoDeColor(panelP2.getBackground(), punto2,h.getGraphics());
    punto1=darPuntoApartirDeString(comboBoxP1.getSelectedItem().toString());
    textoP1.setText("("+valorTrunco(punto1.getX())+", "+valorTrunco(punto1.getY())+")");
    h.dibujaPuntoDeColor(panelP1.getBackground(), punto1,h.getGraphics());
}

private void comboBoxP2ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(linea1!=null)
        h.dibujaDadaLinea(panelL1.getBackground(), linea1,h.getGraphics());
    if(linea2!=null)
        h.dibujaDadaLinea(panelL2.getBackground(), linea2,h.getGraphics());
    if(punto1!=null)
        h.dibujaPuntoDeColor(panelP1.getBackground(), punto1,h.getGraphics());
    punto2=darPuntoApartirDeString(comboBoxP2.getSelectedItem().toString());
    textoP2.setText("("+valorTrunco(punto2.getX())+", "+valorTrunco(punto2.getY())+")");
    h.dibujaPuntoDeColor(panelP2.getBackground(), punto2,h.getGraphics());
}

private void comboBoxLinea1ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(punto1!=null)
        h.dibujaPuntoDeColor(panelP1.getBackground(), punto1,h.getGraphics());
    if(punto2!=null)
        h.dibujaPuntoDeColor(panelP2.getBackground(), punto2,h.getGraphics());
    if(linea2!=null)
        h.dibujaDadaLinea(panelL2.getBackground(), linea2,h.getGraphics());
    linea1=darLineaApartirDeString(comboBoxLinea1.getSelectedItem().toString());
    textoPendienteL1.setText("m: "+valorTrunco(linea1.regresaPendiente().getValor()));
```

```
    textoP1L1.setText("P1: ("+valorTrunco(linea1.getP1().getX())+", "+
    valorTrunco(linea1.getP1().getY())+")");
    textoP2L1.setText("P2: ("+valorTrunco(linea1.getP2().getX())+", "+
    valorTrunco(linea1.getP2().getY())+")");
    h.dibujaDadaLinea(panelL1.getBackground(), linea1, h.getGraphics());
}

private void comboBoxLinea2ActionPerformed(java.awt.event.ActionEvent evt) {
    h.refresh(h.getGraphics());
    if(punto1!=null)
        h.dibujaPuntoDeColor(panelP1.getBackground(), punto1,h.getGraphics());
    if(punto2!=null)
        h.dibujaPuntoDeColor(panelP2.getBackground(), punto2,h.getGraphics());
    if(linea1!=null)
        h.dibujaDadaLinea(panelL1.getBackground(), linea1,h.getGraphics());
    linea2=darLineaApartirDeString(comboBoxLinea2.getSelectedItem().toString());
    textoPendiente2.setText("m: "+valorTrunco(linea2.regresaPendiente().getValor()));
    textoP1L2.setText("P1: ("+valorTrunco(linea2.getP1().getX())+", "+
    valorTrunco(linea2.getP1().getY())+")");
    textoP2L2.setText("P2: ("+valorTrunco(linea2.getP2().getX())+", "+
    valorTrunco(linea2.getP2().getY())+")");
    h.dibujaDadaLinea(panelL2.getBackground(), linea2, h.getGraphics());
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    CpuntoDoble P1=new CpuntoDoble(punto1.getX(),punto1.getY());
    CpuntoDoble P2=new CpuntoDoble(punto2.getX(),punto2.getY());
    CpuntoDoble C1=linea1.getP1();
    CpuntoDoble C2=linea1.getP2();
    CpuntoDoble D1=linea2.getP1();
    CpuntoDoble D2=linea2.getP2();
    double alpha=-(this.anguloEnRadianesDeRecta(linea1.regresaPendiente()));
    //angulo que deja horizontal la linea 1
    //procedemos a rotar todo el sistema alpha grados
    P1=P1.rotar(alpha);
    double a1=P1.getX();
    double a2=P1.getY();
    P2=P2.rotar(alpha);
    double b1=P2.getX();
    double b2=P2.getY();
    C1=C1.rotar(alpha);
    double c1=C1.getX();
    double c2=C1.getY();
    C2=C2.rotar(alpha);
    double c3=C2.getX();
    double c4=C2.getY();
    D1=D1.rotar(alpha);
    double d1=D1.getX();
    double d2=D1.getY();
    D2=D2.rotar(alpha);
    double d3=D2.getX();
    double d4=D2.getY();
    System.out.println("d1 "+d1);
    System.out.println("d2 "+d2);
    System.out.println("d3 "+d3);
    System.out.println("d4 "+d4);
    //encontramos los valores de los coeficientes de la parabola asociada a L1 y a P1
    double e1=1.0/(2.0*(a2-c2));
    double e2=(-2.0*a1)/(2.0*(a2-c2));
```

```
double e3=((a1*a1)+(a2*a2)-(c2*c2))/(2.0*(a2-c2));
double beta=-Math.atan((d4-d2)/(d3-d1));//angulo que deja horizontal la linea 1
System.out.println("e1 "+e1);
System.out.println("e2 "+e2);
System.out.println("e3 "+e3);
System.out.println("alpha "+alpha);
//procedemos a rotar por segunda vez para dejar vertical la parabola
asociada a L2:(D1,D2) y a P2

cParabola parabola2= new cParabola(P2,new cLinea(D1,D2),0.0);
parabola2=parabola2.rotarParabola(beta);
double g1=parabola2.obtenerEcuacion().getP();
double g2=parabola2.obtenerEcuacion().getQ();
double g3=parabola2.obtenerEcuacion().getR();
System.out.println("g1 "+g1);
System.out.println("g2 "+g2);
System.out.println("g3 "+g3);
System.out.println("beta "+beta);
//ahora tenemos dos familias de rectas tangentes a las par\'abolas
definidas por los coeficientes "e" y "g"
//como la segunda familia depende de beta, lo reescribimos asi:
double k1=2.0*g1*Math.cos(beta);
double k2=(g2*Math.cos(beta))-(Math.sin(beta));
double k3=-g1;
double k4=0;
double k5=g3;
double k6=2.0*g1*Math.sin(beta);
double k7=Math.cos(beta)+(g2*Math.sin(beta));

System.out.println("k1"+k1);
System.out.println("k2 "+k2);
System.out.println("k3 "+k3);
System.out.println("k4 "+k4);
System.out.println("k5"+k5);
System.out.println("k6 "+k6);
System.out.println("k7 "+k7);

//el sistema de ecuaciones nos queda en funcion de las variables anteriores
double w0=(k5*k7)+(((k2*k2)-(2.0*e2*k2*k7)+(e2*e2*k7*k7))/(4.0*e1))-
(e3*k7*k7);
double w1=(k5*k6)+(k4*k7)+(((2.0*k1*k2)-(2.0*e2*k1*k7)-(2.0*e2*k2*k6)+
(2.0*e2*e2*k6*k7))/(4.0*e1))-(2.0*e3*k6*k7);
double w2=(k4*k6)+(k3*k7)+(((k1*k1)-(2.0*e2*k1*k6)+(e2*e2*k6*k6))/
(4.0*e1))-(e3*k6*k6);
double w3=k3*k6;
System.out.println("w0 "+w0);
System.out.println("w1 "+w1);
System.out.println("w2 "+w2);
System.out.println("w3 "+w3);

double h0,h1,h2;
if(w3!=0.0){
    claseCardano ecuacion=new claseCardano(w3,w2,w1,w0);
    ecuacion.resuelve();
    h0=ecuacion.getX0();
    h1=ecuacion.getX1();
    h2=ecuacion.getX2();
    System.out.println("h0 "+h0);
    System.out.println("h1 "+h1);
```

```
System.out.println("h2 "+h2);
if(ecuacion.getDiscriminante()>0.0){
    double X1=0.0;
    double Y1=(((k1*h0+k2)*X1)+(k3*h0*h0)+(k4*h0)+(k5))/((k6*h0)+k7);
    CpuntoDoble Np1= new CpuntoDoble(X1,Y1);
    double X2=100.0;
    double Y2=(((k1*h0+k2)*X2)+(k3*h0*h0)+(k4*h0)+(k5))/((k6*h0)+k7);
    CpuntoDoble Np2= new CpuntoDoble(X2,Y2);

    Np1=Np1.rotar(-alpha);
    Np2=Np2.rotar(-alpha);
    System.out.println();
    System.out.println("P1="+Np1.getX()+","+Np1.getY());
    System.out.println("P2="+Np2.getX()+","+Np2.getY());
    cLinea Nl1=new cLinea(Np1,Np2);
    doblado D=new doblado(lp,ll,h);
    D.hacerDoblado(Integer.toString((int)Nl1.getP1().getX()),
    Integer.toString((int)Nl1.getP1().getY()),Integer.toString((int)Nl1.getP2().getX()),
    Integer.toString((int)Nl1.getP2().getY()), true);
    ll.borrarAlInicio();
    lineaDobladoSinAnimacion lda=new lineaDobladoSinAnimacion(lp,ll,h);
    ll.insertarAlInicio(lda.trazarLineaDeDoblado
    (Nl1.getP1().getX(),Nl1.getP1().getY(),Nl1.getP2().getX(),Nl1.getP2().getY()));




}
else{
    double X1=0.0;
    double Y1=(((k1*h0+k2)*X1)+(k3*h0*h0)+(k4*h0)+(k5))/((k6*h0)+k7);
    CpuntoDoble Np1= new CpuntoDoble(X1,Y1);
    double X2=100.0;
    double Y2=(((k1*h0+k2)*X2)+(k3*h0*h0)+(k4*h0)+(k5))/((k6*h0)+k7);
    CpuntoDoble Np2= new CpuntoDoble(X2,Y2);

    Np1=Np1.rotar(-alpha);
    Np2=Np2.rotar(-alpha);

    System.out.println("P1="+Np1.getX()+","+Np1.getY());
    System.out.println("P2="+Np2.getX()+","+Np2.getY());

    double X3=0.0;
    double Y3=(((k1*h1+k2)*X3)+(k3*h1*h1)+(k4*h1)+(k5))/((k6*h1)+k7);
    CpuntoDoble Np3= new CpuntoDoble(X3,Y3);
    double X4=100.0;
    double Y4=(((k1*h1+k2)*X4)+(k3*h1*h1)+(k4*h1)+(k5))/((k6*h1)+k7);
    CpuntoDoble Np4= new CpuntoDoble(X4,Y4);

    Np3=Np3.rotar(-alpha);
    Np4=Np4.rotar(-alpha);



    double X5=0.0;
    double Y5=(((k1*h2+k2)*X5)+(k3*h2*h2)+(k4*h2)+(k5))/((k6*h2)+k7);
    CpuntoDoble Np5= new CpuntoDoble(X5,Y5);
    double X6=100.0;
```

```
            double Y6=(((k1*h2+k2)*X6)+(k3*h2*h2)+(k4*h2)+(k5))/((k6*h2)+k7);
            CpuntoDoble Np6= new CpuntoDoble(X6,Y6);

            Np5=Np5.rotar(-alpha);
            Np6=Np6.rotar(-alpha);

            lineaDobladoSinAnimacion LSA1=new lineaDobladoSinAnimacion(lp,ll,h);
            lineaDobladoSinAnimacion LSA2=new lineaDobladoSinAnimacion(lp,ll,h);
            lineaDobladoSinAnimacion LSA3=new lineaDobladoSinAnimacion(lp,ll,h);
            cLinea lineaDoblez1=LSA1.trazarLineaDeDoblado(Np1.getX(),Np1.getY(),
            Np2.getX(),Np2.getY());
            cLinea lineaDoblez2=LSA2.trazarLineaDeDoblado(Np3.getX(),Np3.getY(),
            Np4.getX(),Np4.getY());
            cLinea lineaDoblez3=LSA3.trazarLineaDeDoblado(Np5.getX(),Np5.getY(),
            Np6.getX(),Np6.getY());
            seleccionDeTresLineas SD= new seleccionDeTresLineas(h,ll,lp,
            lineaDoblez1,lineaDoblez2,lineaDoblez3);
            SD.setVisible(true);
        }
    }
}

    public cPunto darPuntoApartirDeString(String sPunto){
    //este metodo regresa un punto de la lista de puntos, a partir de su nombre (sPunto)
        cPunto[] lista=lp.regresaLista();//el arrreglo contendra la lista de puntos
        return lista[Integer.parseInt(sPunto.substring(5,6))-1];
        //dependiendo de cual punto se quiera (sPunto) este se regresa

    }

    public cLinea darLineaApartirDeString(String sLinea){
    //este metodo regresa una linea de la lista de lineas, a partir de su nombre (sLinea)
        cLinea[] lista=ll.regresaLista();//arreglo que contiene la lista de lineas
        return lista[Integer.parseInt(sLinea.substring(5,6))-1];
         //se regresa la linea seleccionada (sLinea)

    }

    public double anguloEnRadianesDeRecta(cPendiente M){
    if(M.getEsVertical()){
        return Math.PI/2;
    }
    else{
        return Math.atan(M.getValor());
    }
    }

    public String valorTrunco(double vOriginal){
    String V=Double.toString(vOriginal);
    int t=V.length();
    if(t>=7)
        return V.substring(0,7);
    else
        return V;
}

    // Variables declaration - do not modify
    private javax.swing.JComboBox comboBoxLinea1;
    private javax.swing.JComboBox comboBoxLinea2;
```

```
    private javax.swing.JComboBox comboBoxP1;
    private javax.swing.JComboBox comboBoxP2;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JPanel panelL1;
    private javax.swing.JPanel panelL2;
    private javax.swing.JPanel panelP1;
    private javax.swing.JPanel panelP2;
    private javax.swing.JLabel textoP1;
    private javax.swing.JLabel textoP1L1;
    private javax.swing.JLabel textoP1L2;
    private javax.swing.JLabel textoP2;
    private javax.swing.JLabel textoP2L1;
    private javax.swing.JLabel textoP2L2;
    private javax.swing.JLabel textoPendiente2;
    private javax.swing.JLabel textoPendienteL1;
    // End of variables declaration
}
```

Apéndice B

Manual de usuario

A continuación explicaremos la forma de emplear el programa SAHCO, el cual se puede ejecutar en cualquier dispositivo que tenga instalado la máquina virtual de Java (JVM). Se puede visitar el link (https://www.java.com/es/) para obtener la última versión. El archivo que se debe de ejecutar se encuentra en formato .jar y basta con dar doble click, en él, para iniciarlo.

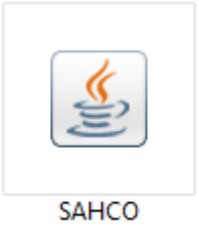


Figura 39. Icono del programa.

Se mostrará inmediatamente la ventana de la figura 40.

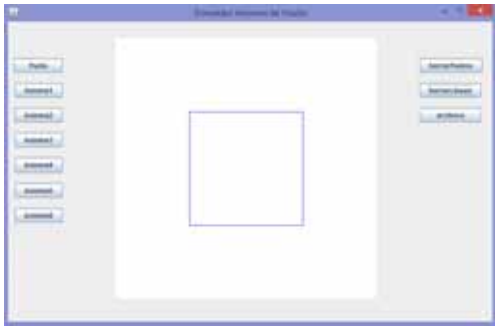


Figura 40. Ventana principal.

Al inicio del programa, se deben de introducir los puntos con los cuales se desea trabajar, estos puntos pueden tener coordenadas desde -100 a 100 tanto en el eje x como en el eje y, se toma la coordenada (0,0) como el centro de la hoja.

Para crear un punto basta con presionar el botón de nombre “Punto”, se desplegará una ventana adicional en la cual se deben introducir las coordenadas deseadas, una vez hecho esto basta con presionar “Aceptar” y el punto aparecerá en la hoja como se muestra en la figura 41.

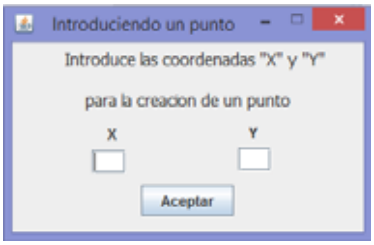


Figura 41. Ventana para crear un punto.

La ventana no se cerrará, gracias a esto se pueden introducir n cantidad de puntos, cuando ya no se desee crear puntos solo se debe cerrar la ventana “Punto” haciendo click en el botón “X” que se encuentra en la esquina superior derecha.

Una vez creados los puntos se puede proceder a aplicar cualquier axioma que se desee, basta con hacer click en el botón que tiene el nombre del axioma, el programa puede aplicar los seis axiomas de Huzita a los puntos y líneas creados, para esto se cuenta con seis botones los cuales hacen referencia a cada uno de ellos.

La ventana principal cuenta con los botones “borrarLineas” y “borrarPuntos” los cuales sirven para poder borrar todos los puntos y todas las líneas previamente creadas, para esto basta con hacer click dichos botones y las líneas y puntos serán borrados, respectivamente, sin la necesidad de volver a ejecutar el programa. A su vez, si se desea o se necesita cerrar el programa pero se quiere seguir trabajando con las líneas y puntos ya creados previamente, se cuenta con la opción para ello, esto nos permite guardar en un archivo de texto plano todos los puntos y líneas que se encuentren en la hoja, basta con hacer click en el botón

"Archivo" y luego en "Guardar Archivo", una ventana se desplegará dando a escoger la ubicación para el archivo que se creará, así como el nombre que llevará. Una vez que ya se tiene el archivo guardado, se puede cargar en el programa haciendo click nuevamente en el botón "Archivo" pero seleccionando "Abrir Archivo", una ventana se desplegará dejando desplazarse por distintas ubicaciones para poder buscar el archivo y abrirlo en el programa.

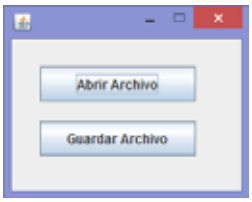


FIGURA 42. Archivos, se puede guardar o cargar el archivo.

A continuación se mostrarán las ventanas que se despliegan, y el uso de cada una de ellas, al escoger cada uno de los axiomas.

Axioma1: Al presionar este botón se desplegará una ventana en la cual hay que seleccionar dos puntos previamente dibujados, de una lista desplegable, para el aplicar el axioma.

FIGURA 43. Axioma 1, seleccionar dos puntos.

Al irse desplazando por la listas desplegables, los puntos se irán marcando de color para identificar que punto corresponde en la hoja. Una vez seleccionados los puntos se debe dar click en "Aplicar", se cerrará la ventana y se podrá visualizar la animación del doblez en la hoja, una vez finalizado el movimiento, se obtendrá una línea resultante de haber aplicado el axioma.

Axioma2: Para el funcionamiento de éste axioma, de la misma forma que en el axioma anterior, se requiere de dos puntos. Se debe dar click en el botón "Axioma2" se abrirá una ventana en la cual se deben seleccionar dos puntos y después de esto hay que presionar "Aplicar". Se cerrará la ventana y se podrá ver la animación del doblez al aplicar dicho axioma.

FIGURA 44. Axioma 2, seleccionar dos puntos.

Axioma3: Para poder aplicar el axioma es necesario contar con al menos dos líneas generadas por cualquiera de los axiomas anteriores. Una vez teniendo esto y presionando el boton "Axioma3" se despliega la ventana mostrada en la figura 45.

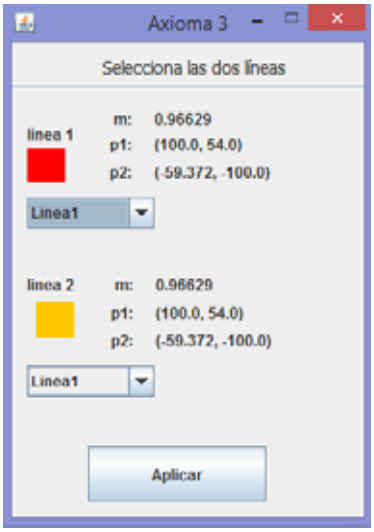


FIGURA 45. Axioma 3, seleccionar dos líneas.

Dicha ventana pide elegir dos líneas sobre las cuales aplicar el axioma, para esto, basta con desplazarse a través de las listas desplegables, se irán marcando de colores las líneas seleccionadas para una fácil identificación, así también, se muestra información acerca de la línea, como su pendiente y los puntos extremos por los cuales está conformada. Al estar seguros de que líneas son las que se desean, hay que presionar el botón "Aplicar".

Este axioma tiene dos posibles casos dependiendo de la posicion de las lneas seleccionadas. El primer caso se da cuando las dos l´neas se intersectan dentro de la hoja de papel, cuando esto sucede, al aplicar el axioma se cerrará la ventana y otra se abrirá, en

ésta habrá que seleccionar que doblez se desea realizar, mostrará dos opciones de distintos colores y a su vez los posibles dobleces estarán marcados en la hoja.

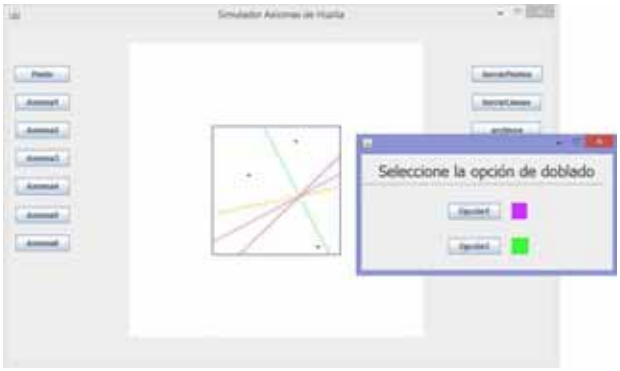


FIGURA 46. Seleccionar posibles dobleces.

Basta con dar click en el botón correspondiente al doblez deseado, la ventana se cerrará y la animación se podrá visualizar.

El segundo caso se da cuando las líneas elegidas son paralelas o su intersección esta fuera de la hoja, al seleccionar las líneas que cumplen con estas condiciones y presionar "Aplicar" en la ventana "Axioma3", simplemente se cerrará la ventana y se visualizará la animación del doblez.

Axioma4: Para poder aplicar este axioma es necesario que previamente haya al menos un punto y una línea en la hoja, si ya se cuenta con eso y se presiona el botón "Axioma4", se pedirá seleccionar un punto y una línea de las listas desplegables.

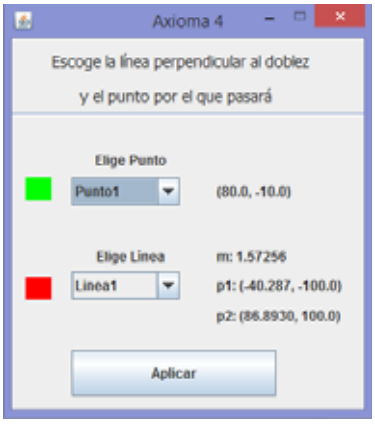


FIGURA 47. Axioma 4, seleccionar un punto y una línea.

Al irse desplazando por las listas desplegables, se marcarán de una color, en la hoja de papel, el punto y línea seleccionados, a su vez, se mostrará información acerca de la línea y punto, como por ejemplo, las coordenadas del punto así como la pendiente y puntos de la línea. Se debe presionar "Aplicar" para poder ver la animación del doblado, al finalizar la animación quedarán los dobleces marcados.

Axioma5: El axioma requiere que se tengan dibujados en la hoja dos puntos y una línea, de igual forma que en los anteriores axiomas, se irán marcando de color los puntos y líneas en la hoja, por los cuales se está desplazando en las listas desplegables de selección que se encuentran en la ventana.

FIGURA 48. Axioma 5, seleccionar dos puntos y una línea.

Una vez que se hayan elegido, se presiona el botón "Aplicar" para ejecutar el axioma, pueden existir cero, uno o dos dobleces que resuelven el axioma, en caso de que sea cero se informará al usuario, con una ventana de error, que no hay doblez que solucione el axioma. Si hay un doblez este se efectuará inmediatamente con su animación correspondiente, tal como en los axiomas anteriores. Por último, cuando hay dos posibles dobleces se desplegará una ventana, en la hay que elegir cual de los dos posibles dobleces se deben ejecutar, las opciones se marcarán de distintos colores en el panel.

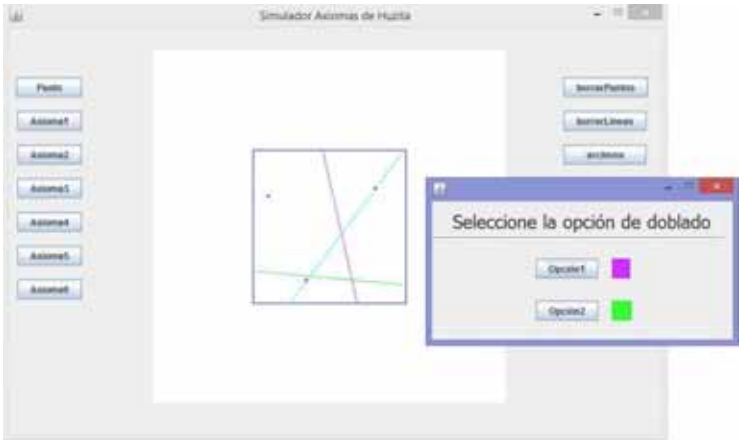


FIGURA 49. Escoger el doblado a realizar si hay mas de una posible solución.

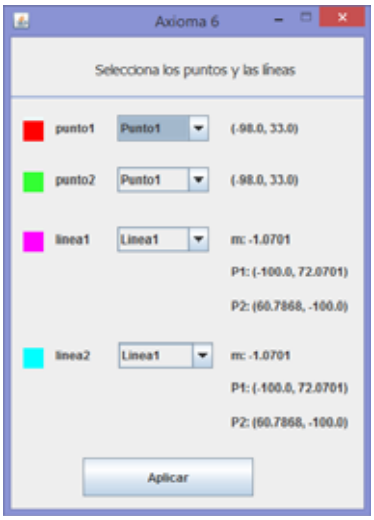


FIGURA 50. Axioma 6, seleccionar dos puntos y dos líneas.

Axioma6: Para aplicar este axioma es necesario contar con al menos dos líneas y dos puntos dibujados en la hoja, de la misma forma que en los anteriores, se deben seleccionar mediante listas desplegables, y en la hoja se irán visualizando de distinto colores. A su vez, se muestra información acerca de las líneas y puntos seleccionados, como las coordenadas de los puntos y los puntos y pendientes de las líneas.

Una vez seleccionados, damos click en "Aplicar" lo cual desplegará una ventana de selección si es que hay tres soluciones, recordemos que al igual que en el axioma 5 puede haber varias soluciones y por lo tanto distintos posibles dobleces. Cada uno de los posibles dobleces se muestra de colores diferentes en el panel, de esta manera el usuario puede seleccionar alguno al solo hacer click en la opción deseada.

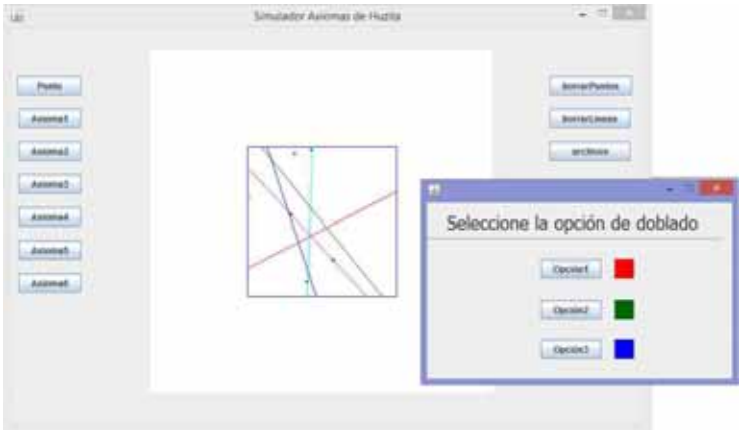


FIGURA 51. Escoger la forma en que se realizará el doblez.

Bibliografía

[1] H. Huzita, "The algebra of paper folding", Proceedings of the First International Meeting of Origami Science and Technology, Humiaki Huzita, ed., 215222, 1989.

[2] A.D. Ramos León, "Algoritmos para plegado de mapas cuadrados", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2011.

[3] J.C. Rangel Reyes, "Simulador cuerpos rígidos: esfera y terreno accidentado", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2011.

[4] J. García Gregorio y C. Cuéllar Medina, "Ambiente de programación visual para graficar geometrías simples mediante transformaciones ortogonales con álgebra geométrica y álgebra matricial", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2009.

[5] R. C. Alperin y R. J. Lang, "One-, Two-, and Multi-Fold Origami Axioms", International Meeting of Origami in Science, Mathematics and Education. R. Lang, ed., AK Peters, Natick, MA, 371393, 2009.

[6] W. E. Calderón Gualdrón, "Propuesta metodológica para la enseanza de las secciones cónicas en el grado décimo de la institución educativa villas de san Ignacio de Bucaramanga", Tesis presentada como requisito parcial para optar al título de Magister en Enseanza de las Ciencias Exactas y Naturales, Universidad Nacional de Colombia Facultad de Ciencias, Medellín, Colombia, 2013.

[7] Aguilar-Zavoznik, A. "Regla y compás vs origami", Carta Informativa de la SMM, No. 68, 2013.

[8] Alperin, R. C., "A Mathematical Theory of Origami Constructions and Numbers", New York Journal of Mathematics, No. 6, 119-133, 2000.

[9] Gertschläger, R., "Euclidean Constructions and the Geometry of Origami", Mathematics Magazine, Vol. 68, No. 5, 1995.