Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Terminal:

"Visualización de algoritmos de ordenamiento y búsqueda interna con TikZ"

Alumnos:

Bocanegra Rodríguez Julio Alfonso 204200963
Tovar González Yolanda 204360446

Asesor:
Dr. Francisco Javier Zaragoza Martínez

20197

Tabla de contenido

Objetivo general

Implementar un programa con interfaz gráfica que genere el código fuente en lenguaje *TikZ,* y obtener diversas visualizaciones de algoritmos de ordenamiento y búsqueda interna, para facilitar el aprendizaje del comportamiento de los mismos.

Objetivos particulares

1. Proponer los diferentes modos de visualización que se implementarán para cada uno de los algoritmos de ordenamiento y búsqueda interna.

2. Diseñar e implementar el módulo de almacenamiento y recuperación de los algoritmos de ordenamiento y búsqueda interna.

3. Diseñar e implementar el módulo de las especificaciones de los datos de entrada (ingresados manualmente o aleatorios).

4. Diseñar e implementar el módulo que dará formato a la representación gráfica (visualizaciones) de los algoritmos de ordenamiento y búsqueda interna.

5. Elaborar el módulo de datos de salida que generará el código fuente en el lenguaje *TikZ* y el archivo PDF.

6. Diseñar e implementar el módulo de la interfaz gráfica donde el usuario especifique las propiedades de la visualización de los algoritmos de ordenamiento y búsqueda interna.

Antecedentes

En la carrera de Ingeniería en Computación se han estudiado algoritmos de ordenamiento y búsqueda, que desde su aparición han sido vitales para el desarrollo de aplicaciones, el manejo y dominio de la lógica de programación en la resolución problemas.

En la literatura, se menciona el uso en visualizaciones de los algoritmos de búsqueda y ordenamiento, los cuales sólo son tratados de manera teórica [1] [2]. En su libro Sedgewick [3] nos da una idea clara del comportamiento de los algoritmos más conocidos, además se incluye en el texto implementaciones completas de los métodos empleados, junto con descripciones del funcionamiento de los programas en un conjunto coherente de ejemplos. Estos ejemplos dan una idea de lo que se busca en esta propuesta, agregando otras características a las visualizaciones para hacerlas más claras y comprensibles.

Existe una propuesta de Proyecto Terminal [4], que tiene un objetivo parecido al que se pretende en este nuevo proyecto, que es el de implementar una herramienta que facilite el estudio y el aprendizaje de estructuras de tipo árbol y algoritmos de ordenamiento a través de visualizaciones, en ella se propuso realizar animaciones en *OpenGL*. En nuestra propuesta se busca que las visualizaciones sean personalizadas por el usuario las veces que desee.

En Internet también podemos encontrar animaciones de visualizaciones de algoritmos de ordenamiento y búsqueda [5][6], en ellas se ve en cada paso cómo se modifican las posiciones de los elementos. Estas visualizaciones sólo tienen un formato por lo cual no se puede cambiar el modo de visualizar los objetos, por ejemplo los colores o hacer portátil la visualización que es lo que se busca en esta propuesta.

La ventaja de programar gráficos es que permite tener un control absoluto y preciso sobre todos los detalles, por ello se requiere una herramienta que permita realizar gráficos sencillos y de una manera rápida.

En este proyecto se trabajará con *LaTeX*[1] , el cual se considera como un ambiente especial en cuestión de insertar figuras implementadas con comandos simples. Existen varios editores que permiten hacer figuras y generar el código en *LaTeX*, listo para introducirlo en nuestro documento, uno de ellos es *TikZ,* este paquete sirve para crear gráficos para documentos *LaTeX* usando un ambiente más sencillo y comandos especiales para dibujar líneas, curvas, rectángulos, etc.

---

[1] LaTeX es un sistema de composición de textos que está orientado especialmente a la creación de documentos científicos que contengan fórmulas matemáticas, también se utiliza para realizar desde sencillas cartas hasta un libro completo.

Se puede decir que *TikZ* permite crear gráficos sofisticados de una manera bastante intuitiva y fácil. En Internet [8] se encontrarán una gran variedad de ejemplos uno de ellos es:

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{shapes,snakes}
\begin{document}

\pagestyle{empty}

\begin{tikzpicture}[scale=2]
   \tikzstyle{ann} = [draw=none,fill=none,right]
   \matrix[nodes={draw, ultra thick, fill=blue!20},
      row sep=0.3cm,column sep=0.5cm] {
   \node[draw=none,fill=none] {Plain node}; &
   \node[rectangle] {Rectangle}; &
   \node[circle] {Circle};\\
   \node[ellipse] {Ellipse};&
   \node[circle split] {Circle \nodepart{lower} split};&
   \node[forbidden sign,text width=4em, text centered]
            {Forbidden sign};\\
   \node[diamond] {Diamond};&
   \node[cross out] {Cross out};&
   \node[strike out] {Strike out};\\
   \node[regular polygon,regular polygon sides=5] {$n=5$};&
   \node[regular polygon,regular polygon sides=7] {$n=7$};&
   \node[regular polygon,regular polygon sides=9] {$n=9$};&
   \node[ann]{Regular polygon};\\
   \node[star,star points=4] {$p=4$};&
   \node[star,star points=7,star point ratio=0.8] {$p=7$};&
   \node[star,star points=10] {$p=9$};&
   \node[ann]{Star};\\
   };
\end{tikzpicture}
```

El resultado de este código son varias figuras geométricas. (Ver figura 1)

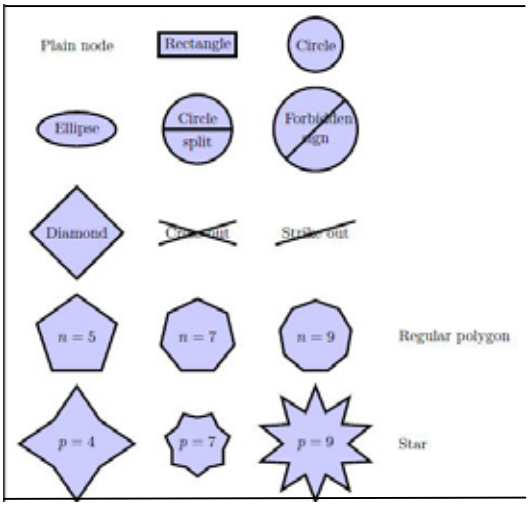

Figura 1. Figuras geométricas realizadas con *TikZ*.

Justificación

Los algoritmos de ordenamiento y búsqueda constituyen un interesante caso de estudio para la resolución de problemas. Los métodos de ordenamiento se clasifican como internos o externos según dónde estén los elementos a ordenar. Los datos en los algoritmos de ordenamiento interno se encuentran en la memoria principal (RAM) y los datos en los algoritmos de ordenamiento externo se encuentran en memoria externa como un disco duro.

Este proyecto se enfocará al desarrollo de una herramienta de software que permita la visualización de algoritmos de ordenamiento y búsqueda interna, para facilitar la comprensión y el aprendizaje de los mismos. Como ejemplo de lo que sería una visualización, tenemos la figura 2, en donde se observa cómo se van ordenando los elementos de diferente tamaño, mostrado en orden descendente, hasta quedar acomodados en orden ascendente.

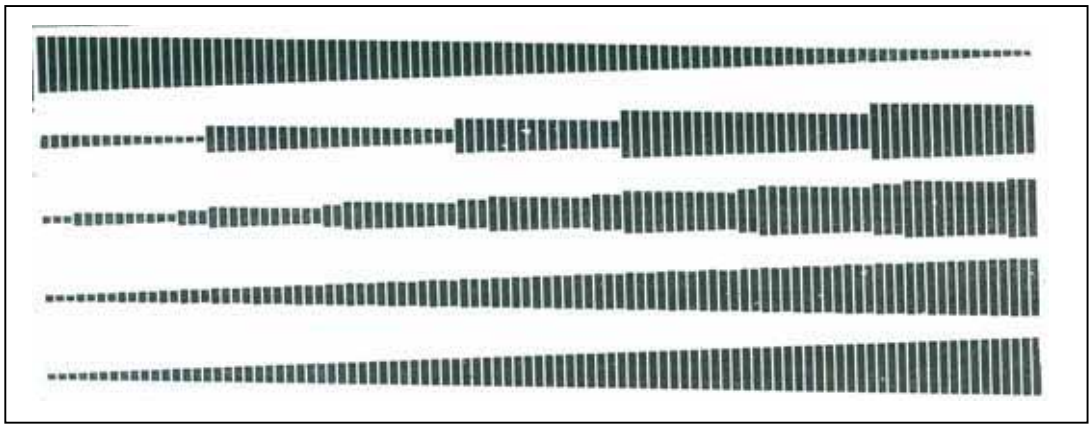

***Figura 2**. Ejemplo de una visualización de un algoritmo de ordenamiento*
Fuente: "Algoritmos en C++" de Sedgewick [3].

Como ejemplo de la visualización de un algoritmo de búsqueda tenemos la figura 3, que nos muestra cómo se lleva a cabo la búsqueda de un elemento en un conjunto previamente ordenado.

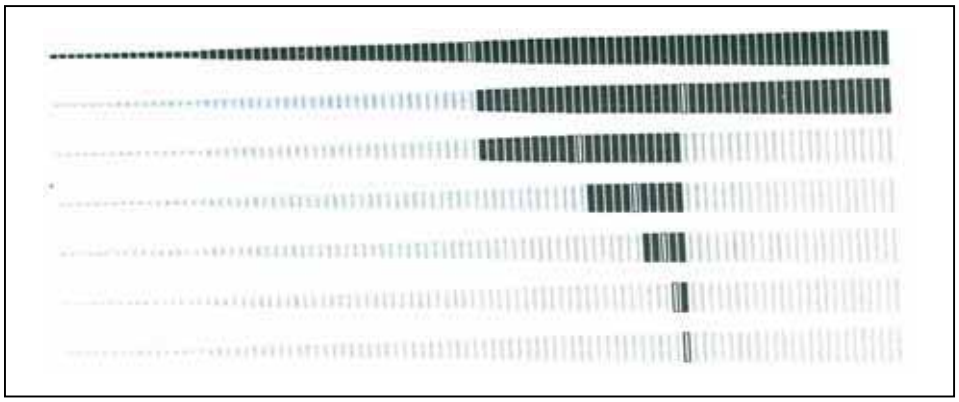

***Figura 3**. Visualización de búsqueda binaria.*
Fuente: "Algoritmos en C++" de Sedgewick [3].

La intención del proyecto es generar visualizaciones como las de las figuras 2 y 3 pero con diferentes formatos, es decir, elegir los colores a utilizar, la representación de los elementos a ordenar o el número de elementos a visualizar, según el algoritmo elegido. Las visualizaciones generadas por la herramienta a desarrollar mostrarán el comportamiento de los algoritmos a cada paso.

El usuario que interactúe con la herramienta de software no necesitará conocimiento alguno de cómo se comportan los algoritmos, ya que la intención es facilitar su aprendizaje. El alumno de la carrera de Ingeniería en Computación ha manejado, implementado y analizado los algoritmos de ordenamiento y búsqueda que aquí se presentarán. Los algoritmos ya estarán implementados en la aplicación, por lo que serán transparentes para el usuario y éste no tiene que preocuparse por ello, lo único que necesita es saber qué algoritmo desea visualizar, y éste le será mostrado en un archivo PDF. Se seleccionó este formato para entregar el resultado final, debido a su versatilidad y facilidad de uso.

La posible continuación de este proyecto puede ser la implementación de visualizaciones de algoritmos de ordenamiento externo. También se podrían realizar visualizaciones de algoritmos en estructuras de datos diferentes como árboles, listas, pilas y colas.

Descripción técnica

La intención de este proyecto es implementar una interfaz gráfica de usuario que genere visualizaciones que muestran el comportamiento de algoritmos de ordenamiento y búsqueda interna.

El proyecto está estructurado en seis módulos: Módulo de Interfaz gráfica, Módulos de datos de entrada, Módulo de formato, Módulo de recuperación, Módulo de almacenamiento y Módulo de datos de salida (figura 4).

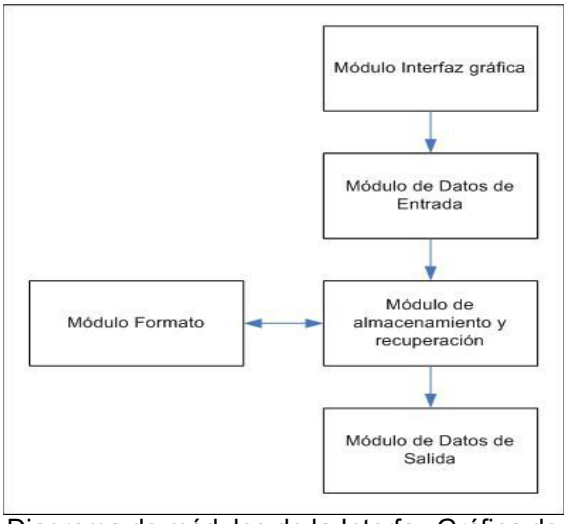

Figura 4. Diagrama de módulos de la Interfaz Gráfica de Usuarios.

El módulo de interfaz gráfica será el encargado de la interacción del software con el usuario. Presentará las opciones con las que se puede personalizar la visualización del algoritmo. Se podrá elegir el algoritmo de ordenamiento o búsqueda de entre una lista ya establecida. Los datos a visualizar en el algoritmo seleccionado podrán ser introducidos de manera aleatoria, en este caso el usuario debe especificar el rango y el número de datos. La otra opción es introducirlos de forma manual, en donde el usuario dará los datos desde teclado.

También se podrá establecer el formato de la visualización donde se especifican las propiedades como son colores, tamaño, formas, etc.

El módulo de datos de entrada tomará las propiedades especificadas en el módulo anterior, preparando la información de tal forma que el módulo de recuperación pueda interpretarla. Otra tarea que realizará es la generación de números aleatorios en el caso de que se haya escogido esta opción.

El módulo de recuperación tendrá comunicación con tres módulos que son el de formato, que dará la presentación a la visualización, con el de almacenamiento para obtener el algoritmo de ordenación o búsqueda según corresponda y por último con el de datos de salida para obtener el PDF de la visualización.

La comunicación con los tres módulos mencionados anteriormente será para llevar a cabo la construcción de un archivo con el código fuente en *TikZ*, que el módulo de salida de datos convertirá en PDF de acuerdo a la definición de las propiedades de la visualización.

En el módulo de formato se dará la apariencia al algoritmo, aquí se tendrá almacenados todos los diseños que se desarrollarán con *TikZ*, colores, tamaño, formas, etc. De acuerdo a las propiedades dadas por el usuario.

El módulo de almacenamiento contendrá los algoritmos de ordenamiento y búsqueda seleccionados, listos sólo para ser extraídos por el módulo de recuperación

Por último tenemos el módulo de datos de salida, él obtendrá un archivo con código fuente en *TikZ*. A este archivo se le añadirá código en *LaTeX* para dar las últimas propiedades como comentarios o texto que aparecerá en la visualización. La última tarea que realiza este módulo es convertir el archivo con código *TikZ* y *LaTeX* para obtener la visualización en un PDF.

La plataforma que se utilizará es Debian 5 GNU/Linux [9], que es una distribución libre del sistema operativo GNU/Linux. Se empleará el entorno de desarrollo integrado NetBeans IDE 6.7.1 [10], para programar en Java los módulos e implementar la interfaz gráfica.

Especificaciones técnicas

Los algoritmos de ordenamiento y búsqueda seleccionados, serán implementados para vectores que fue la estructura de datos que se seleccionó.

Se eligieron un total de 12 algoritmos, 8 de ordenamiento y 4 de búsqueda. La lista de estos algoritmos se muestra en la tabla 1, y fueron elegidos por que son los algoritmos más vistos a lo largo de la carrera de Ingeniería en Computación de la Universidad Autónoma Metropolitana unidad Azcapotzalco.

Algoritmo de ordenamiento	**Algoritmos de Búsqueda**
Selección directa	Secuencial
Inserción directa	Binaria
Intercambio directo	Fibonacci
Mergesort	Interpolación
Quicksort	
Ordenamiento del duende	
Shell sort	
Radix sort	

Tabla 1: Algoritmos de ordenamiento y búsqueda interna.

El número de datos máximo que se podrá ingresar varía según la visualización elegido y puede estar en un rango de 8 a 32. Esta cantidad se estableció por que se desea ver el comportamiento de los algoritmos, no se realizará un análisis, como el de tiempo de ejecución con un número de elementos considerable, así que es una buena cantidad para poder estudiar como se comportan los algoritmos. También se consideró que se puedan observar en la pantalla o en la visualización deseada.

La representación gráfica de cada elemento según sea el caso de cada algoritmo, puede ser elegida por el usuario de entre una lista de diseños que serán desarrollados como parte del proyecto.

La capacidad máxima del proyecto no se alcanzará debido a que no se incluyen muchos algoritmos de ordenamiento y búsqueda.

El proyecto puede considerarse concluido evaluando que cada módulo cumpla con su tarea según la descripción técnica mencionada anteriormente y al final, al integrar cada módulo, se debe de cumplir con el objetivo general del proyecto.

Al final del proyecto se entregará en un CD con toda la documentación, en la que se incluye manual de usuario, manual de instalación, código fuente debidamente comentado, los formatos de datos de entrada y archivos de salida y la documentación del programador donde se incluirán diagramas de flujo, entidad-relación, etc.

Desarrollo

Como primera parte del desarrollo del proyecto se diseñó los modos de visualización para los algoritmos que se propusieron. Los diseños se realizaron en TikZ y son los siguientes:

Visualizaciones para algoritmos de ordenamiento

Círculos. Esta visualización esta conformada por círculos de diferente tamaño. El ordenamiento se realiza de forma ascendente con respecto al radio y se muestra cada iteración que realiza el algoritmo.

A continuación se muestran algunos ejemplos de cómo se comportan los algoritmos de ordenamiento como: selección directa (figura 5), inserción directa (figura 6), intercambio directo (figura 7), ordenamiento del duende (figura 8), shell sort (figura 9), para ésta visualización.

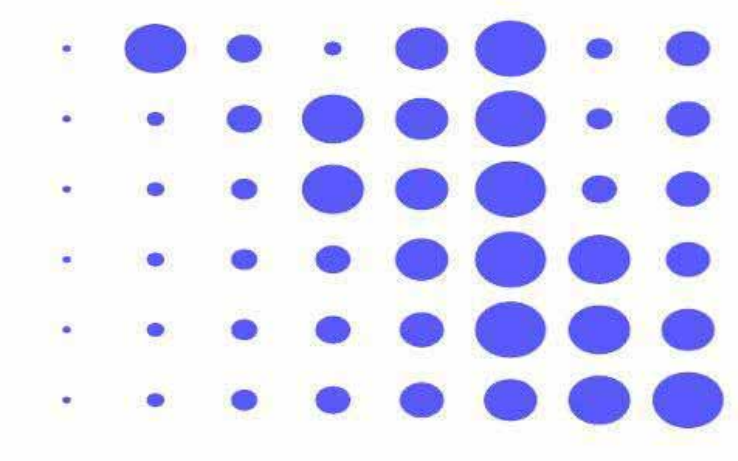

Figura 5. Visualización de círculos para el algoritmo de selección directa.

Figura 6. Visualización de círculos para el algoritmo de inserción directa.

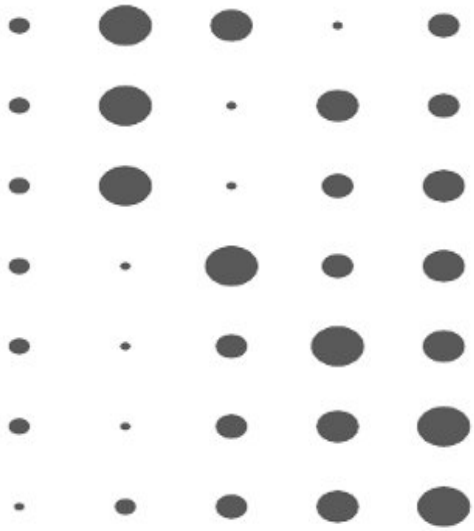

Figura 7. Visualización de círculos para el algoritmo de intercambio directo.

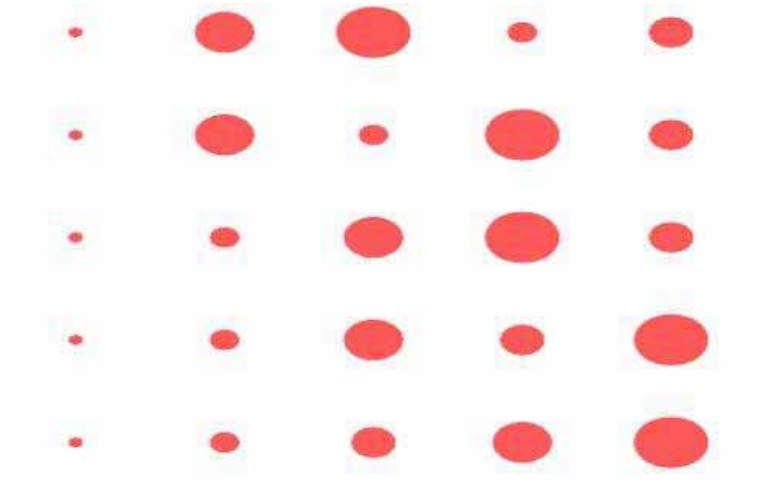

Figura 8. Visualización de círculos para el algoritmo de ordenamiento del duende.

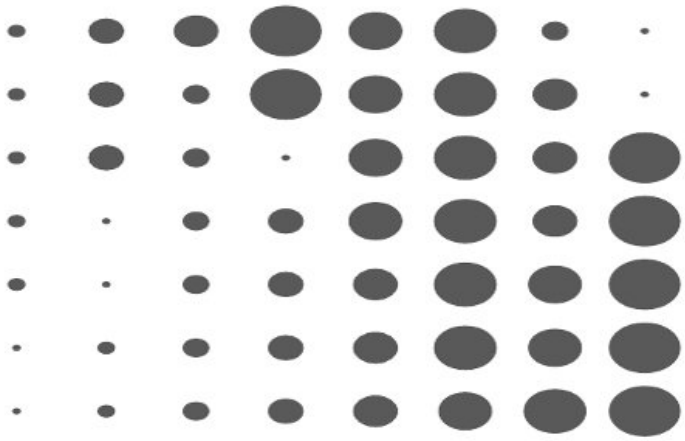

Figura 9. Visualización de círculos para el algoritmo de shell sort.

Pirámide. La visualización está conformada por rectángulos, tomando en cuenta para su ordenamiento el largo de estos, ya que el ancho será constante. Cada rectángulo representa un elemento a ordenar.

Se muestra cada iteración que realiza el algoritmo hasta ordenar los elementos de forma ascendente, obteniendo al final la figura de una pirámide. Se muestra a continuación algunos ejemplos del comportamiento de los algoritmos de ordenamiento para ésta visualización. Ver de la figura 10 a la figura 14.

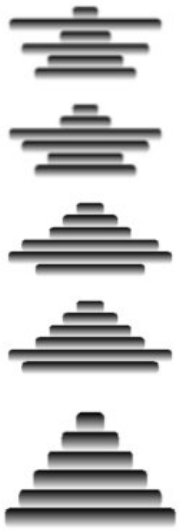

Figura 10. Visualización de pirámide para el algoritmo de selección directa.

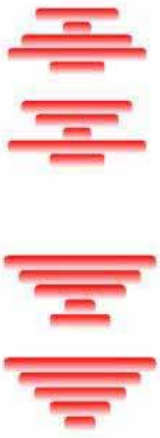

Figura 11. Visualización de pirámide para el algoritmo de inserción directa.

Figura 12. Visualización de pirámide para el algoritmo de intercambio directo.

Figura 13. Visualización de pirámide para el algoritmo de ordenamiento del duende.

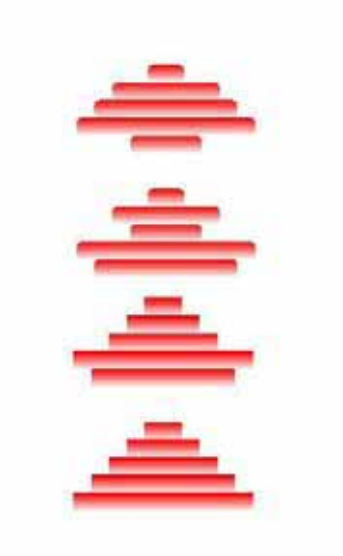

Figura 14. Visualización de pirámide para el algoritmo de shell sort.

Líneas. Esta visualización se lleva a cabo con líneas con diferente porcentaje de degradado. Este porcentaje es la propiedad que se toma para realizar el ordenamiento, al final del algoritmo las líneas se ven ordenadas de las clara a la más oscura. Además se puede observar como se dan los intercambios entre las líneas. Se tienen las siguientes imágenes mostrando el comportamiento de los algoritmos de ordenamiento para ésta visualización. Ver de la figura 15 a la 20.

Figura 15. Visualización de líneas para el algoritmo de selección directa.

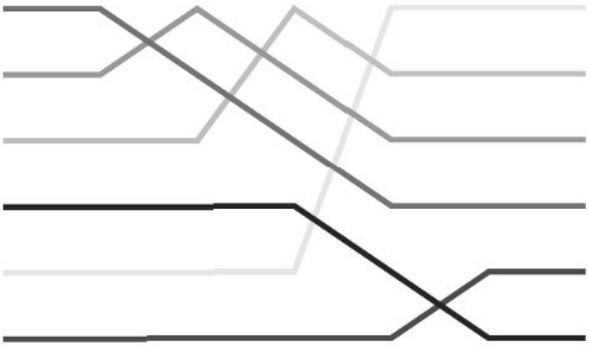

Figura 16. Visualización de líneas para el algoritmo de inserción directa.

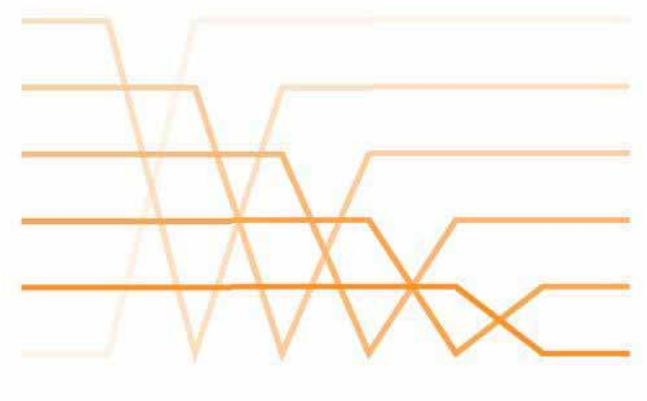

Figura 17. Visualización de líneas para el algoritmo de intercambio directo.

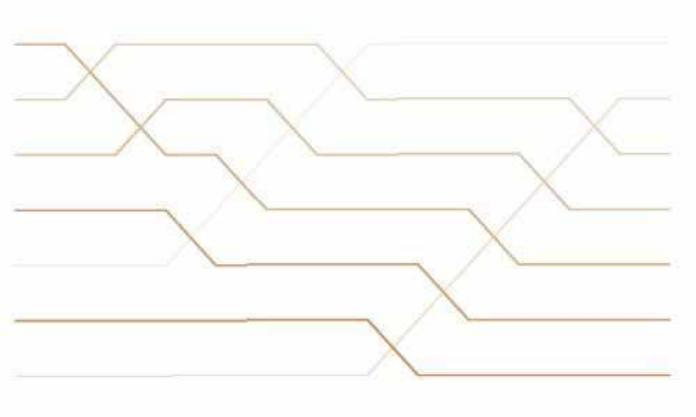

Figura 18. Visualización de líneas para el algoritmo de ordenamiento del duende.

Figura 19. Visualización de líneas para el algoritmo de shell sort.

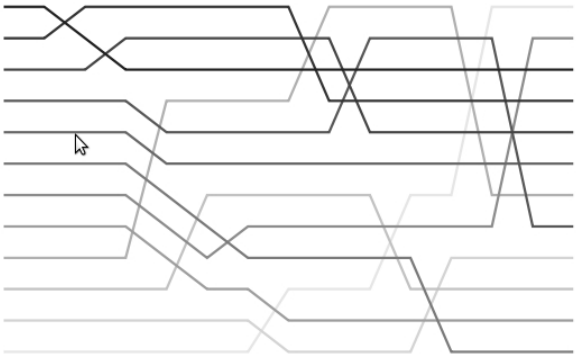

Figura 20. Visualización de líneas para el algoritmo de quicksort.

Estrellas. Para esta visualización se toman las figuras de estrellas pero con diferente número de picos. Al aplicar el algoritmo de ordenamiento las estrellas quedaran ordenadas de menor a mayor de acuerdo al número de picos que tengan. Y se muestra a cada paso como se intercambian de posición los elementos. Algunos ejemplos de cómo se comportan los algoritmos de ordenamiento para esta visualización son: (Ver de la figura 21 a la 25).

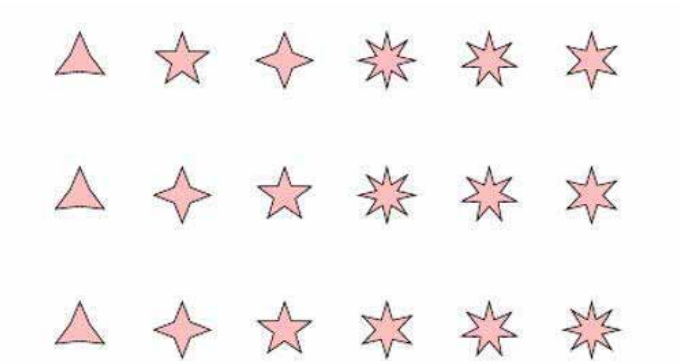

Figura 21. Visualización de estrellas para el algoritmo de selección directa.

Figura 22. Visualización de estrellas para el algoritmo de inserción directa.

Figura 23. Visualización de estrellas para el algoritmo de intercambio directo.

Figura 24. Visualización de estrellas para el algoritmo de ordenamiento del duende.

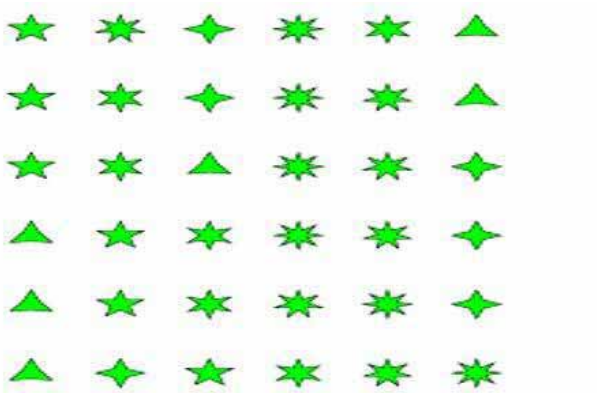

Figura 25. Visualización de estrellas para el algoritmo de shell sort..

Elementos. El diseño de esta visualización esta basado en los elementos químicos de la tabla periódica. El ordenamiento de los elementos se lleva acabo tomando en cuenta su número atómico, quedando al final de la ejecución del algoritmo de menor a mayor.
A continuación se muestra algunos ejemplos de cómo se comporta los algoritmos de ordenamiento para la visualización de elementos. Ver de la figura 26 a la 30.

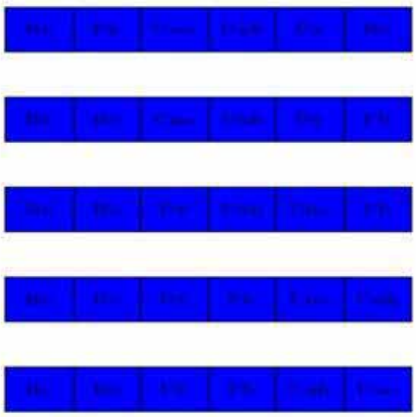

Figura 26. Visualización de elementos para el algoritmo de selección directa.

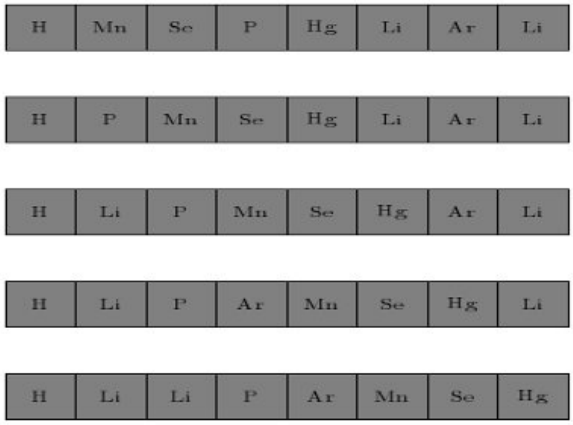

Figura 27. Visualización de elementos para el algoritmo de inserción directa.

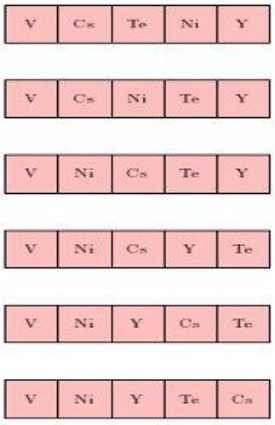

Figura 28. Visualización de elementos para el algoritmo de intercambio directo.

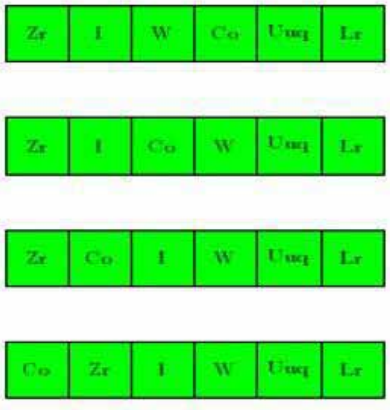

Figura 29. Visualización de elementos para el algoritmo de ordenamiento del duende.

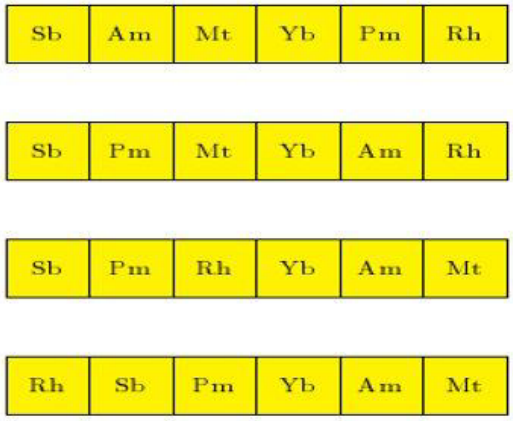

Figura 30. Visualización de elementos para el algoritmo de shell sort..

Letras. Esta visualización consiste en generar aleatoriamente letras del alfabeto, el objetivo es ordenarlas alfabéticamente. Se mostrará cada iteración que vaya realizando el algoritmo. Por ejemplo se tiene cómo quedaría esta visualización para el algoritmo de ordenamiento de megesort (figura 31) y radix sort (figura 32).

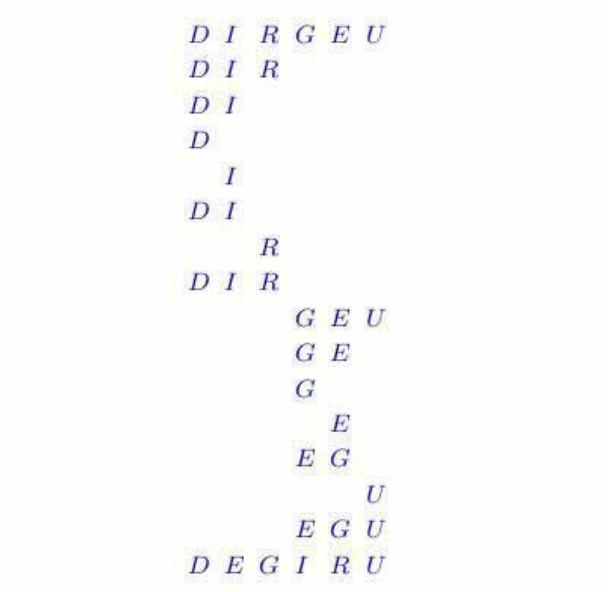

Figura 31. Visualización de letras para el algoritmo de megesort.

Figura 32. Visualización de letras para el algoritmo de radix sort.

Visualizaciones para algoritmos de búsqueda

En estas visualizaciones consideramos que los elementos se encuentran ordenados, por lo que solo lleva a cabo la búsqueda de algún elemento.

Círculos. Esta visualización consta de círculos ordenados de acuerdo a su tamaño. En la parte superior aparecerá el elemento que se desea buscar, este tendrá un color rojo, mientras que los elementos ordenados estarán de un color diferente, el cual el usuario puede elegir. Se mostrará cada iteración que realice el algoritmo, cuando se encuentre el elemento buscado se remarca de color rojo (Ver figura 33 y 34).

Figura 33. Círculos en algoritmos de búsqueda secuencial.

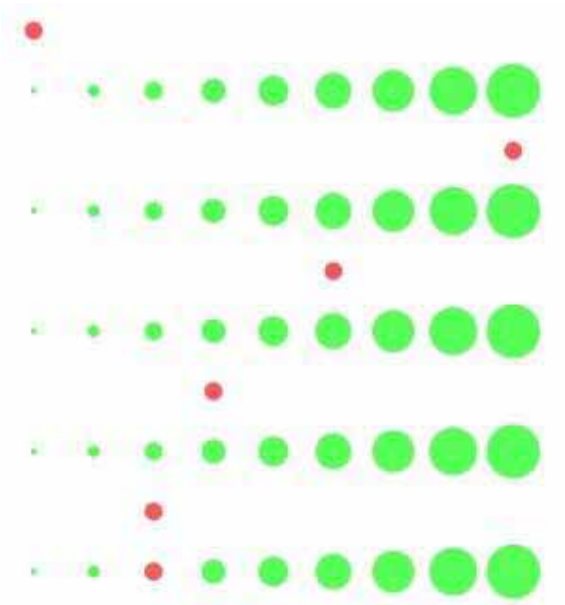

Figura 34. Círculos en algoritmos de búsqueda fibonacci.

Estrellas. En esta visualización las estrellas se encuentran ordenadas de acuerdo al número de picos. Tenemos el ejemplo de búsqueda binaria (figura 35). El elemento que se está buscando aparecerá de un color rojo, mientras que los demás elementos aparecerán de un color diferente, los elementos que se descartan en la búsqueda aparecerán de color gris. Se mostrará todas las iteraciones que realice el algoritmo.

Figura 35. Estrellas en el algoritmo de búsqueda binaria.

Elementos. Esta visualización consiste en tener ordenados elementos químicos ordenados de acuerdo a su número atómico. Se tiene un ejemplo para la búsqueda binaria (figura 36) cuyo elemento buscado se representa con un color rojo mientras que los demás elementos se representan con un color diferente.

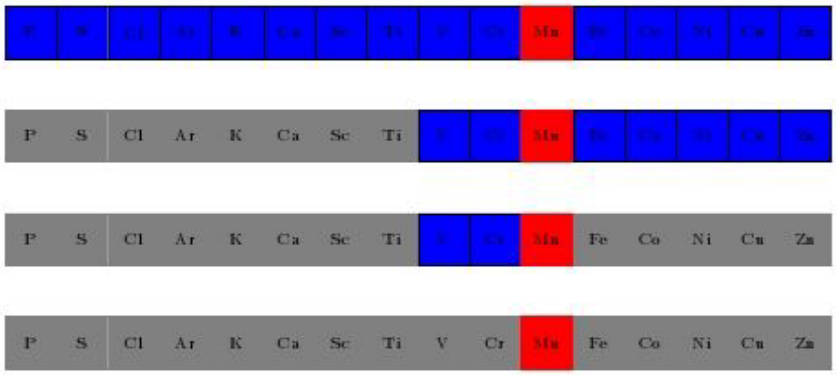

Figura 36. Elementos en el algoritmo de búsqueda binaria.

Letras: Esta visualización consiste en letras ordenadas alfabéticamente, la letra que se va a buscar se encuentra en la parte superior, con un color rojo. En la parte inferior se mostrará los elementos en el arreglo. Se mostrará las iteraciones que realice el algoritmo. Tenemos el ejemplo del algoritmo de interpolación. (figura 37)

Figura 37. Letras en el algoritmo de interpolación.

Módulos

El módulo de interfaz gráfica es el encargado de interactuar con el usuario. Está encargado de mostrar de manera gráfica e intuitiva las propiedades que se le pueden asignar a una visualización. Consta de tres pantallas: una principal, una para el caso que se elijan algoritmos de ordenamiento y otra para el caso de los algoritmos de búsqueda. Una vez elegidas las propiedades de la visualización se le pasan los datos al módulo de datos de entrada (figura 38).

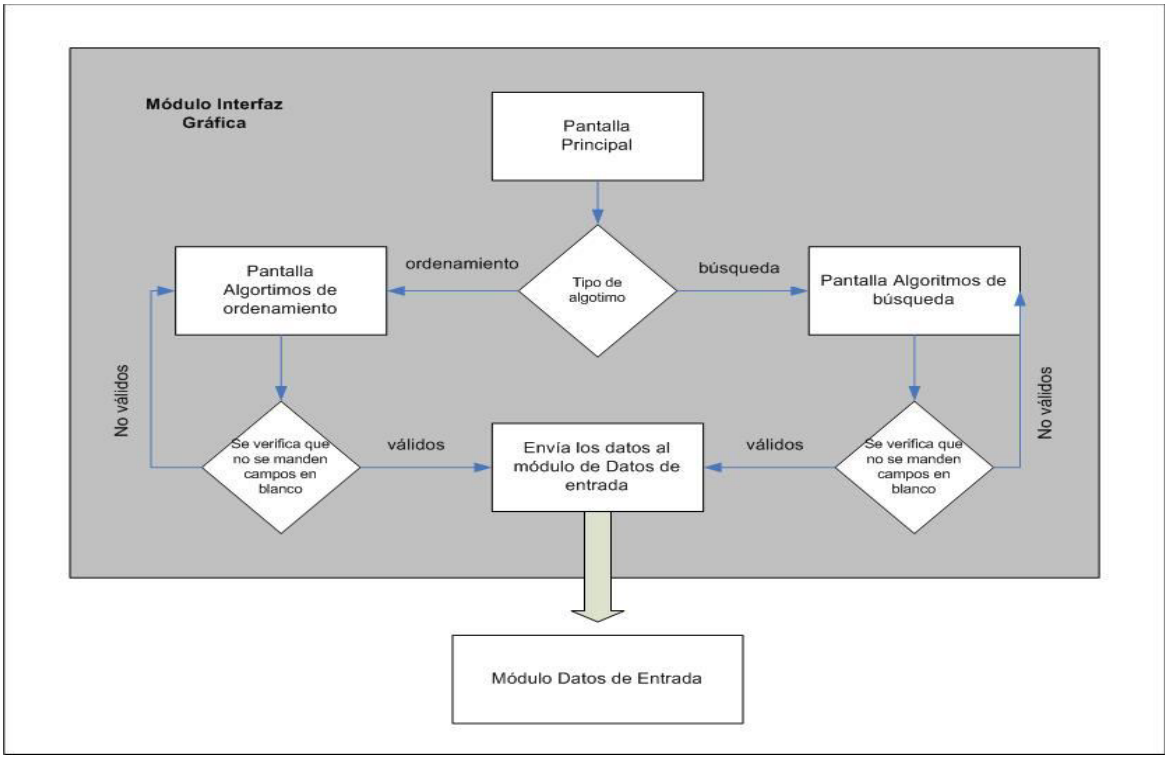

Figura 38. Módulo de Interfaz gráfica.

El módulo de Datos de Entrada recibe los datos introducidos por el usuario y tiene dos funciones. La primera es verificar que los datos sean correctos y la segunda es acomodar los datos en una clase para que en los módulos siguientes se obtenga esta información fácilmente. Una vez terminado estos dos pasos, los datos son mandados al módulo de Almacenamiento y recuperación (figura 39).

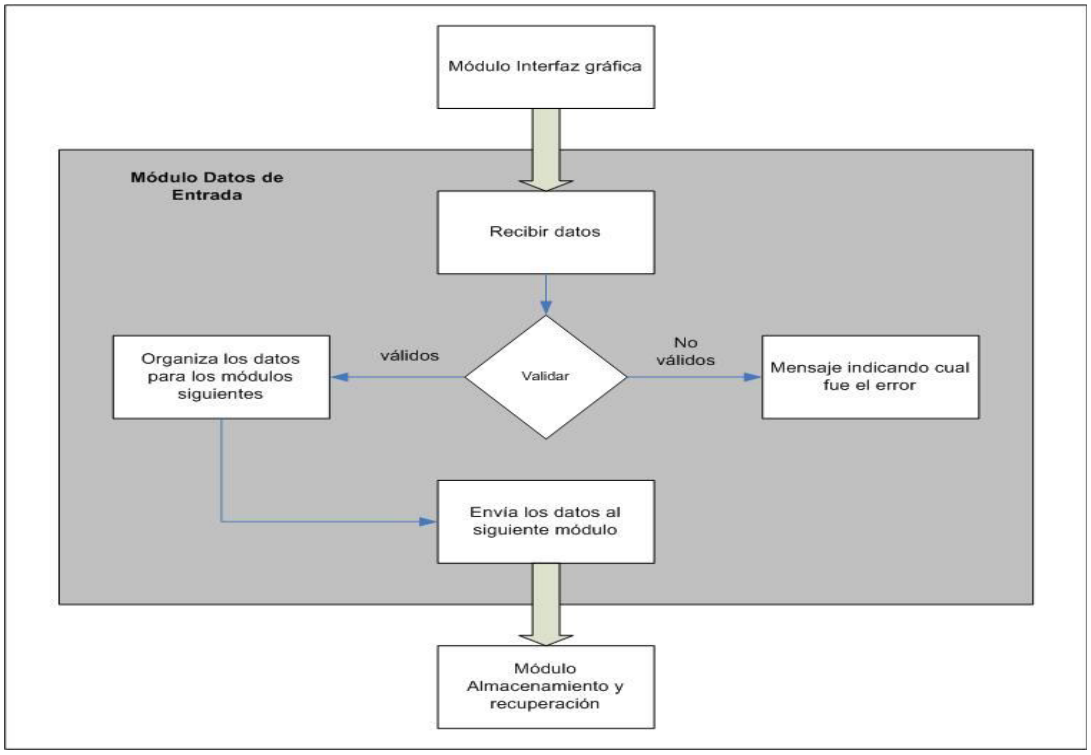

Figura 39. Módulo de Datos de Entrada.

En el módulo de almacenamiento y recuperación se tienen almacenados los algoritmos seleccionados para esta aplicación. Este módulo recibe el nombre del algoritmo del módulo de datos de entrada y se verifica que el algoritmo exista. Si el algoritmo existe, entonces se obtiene de una clase previamente definida y se envía al módulo de formato. Como último pasa de esta parte se agregan las últimas propiedades al algoritmo seleccionado y manda los datos al módulo de datos de salida (figura 40).

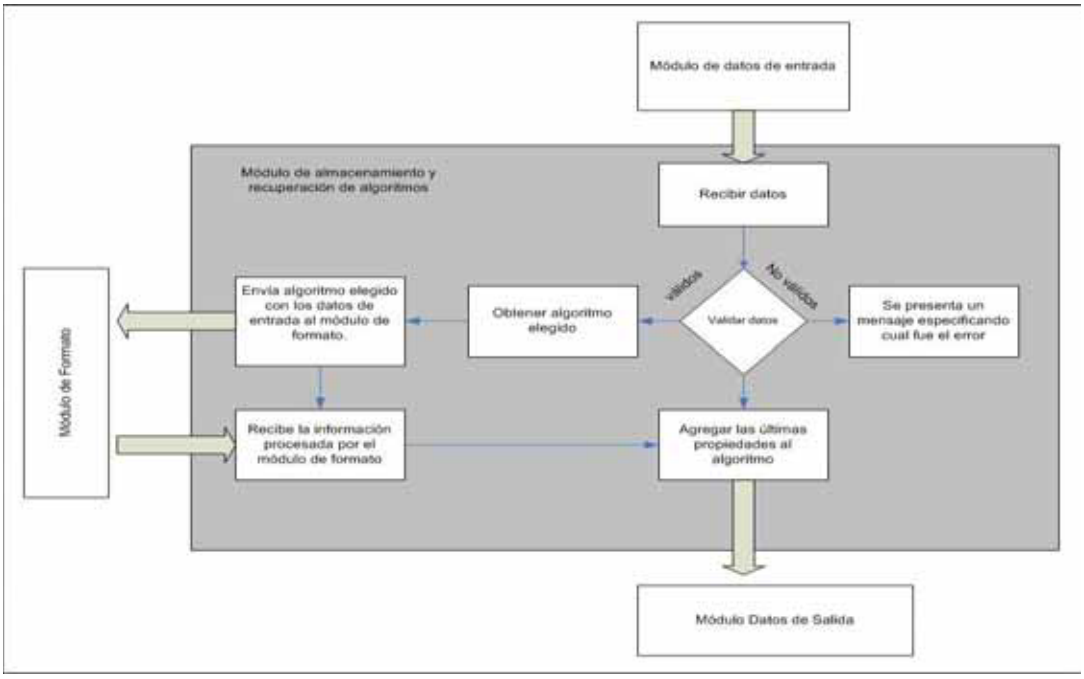

Figura 40. Módulo de almacenamiento y recuperación.

Cuando se llega al módulo de formato entonces ya se han establecido todas las propiedades que tendrá la visualización. Este módulo ejecuta el algoritmo y a cada iteración se va formando la visualización. Una vez terminada la ejecución del algoritmo los datos son regresados al módulo de almacenamiento y recuperación para asignarle las últimas propiedades (figura 41).

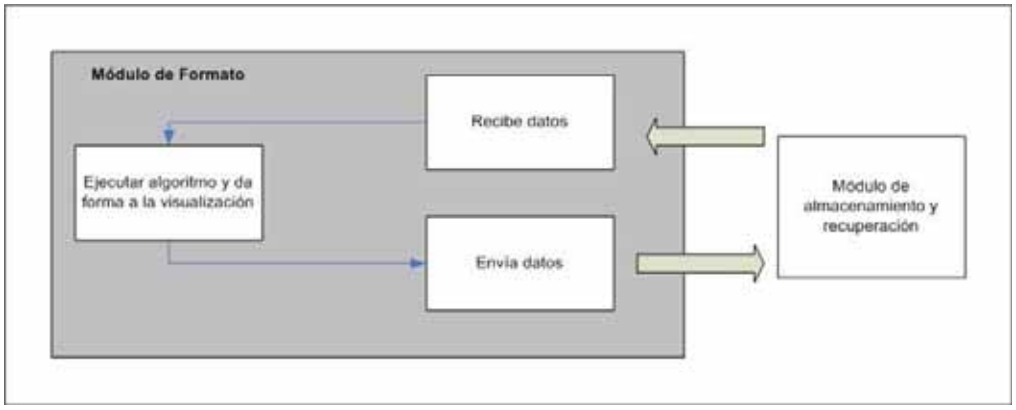

Figura 41. Módulo de formato.

El último módulo por el que pasan los datos es el de Datos de salida. En este punto, los datos están listos para ser guardados en un archivo .tex. Una vez hecho esto, el módulo de salida de datos toma el archivo .tex para generar un documento PDF, en el cual se presenta la visualización del algoritmo (figura 42).

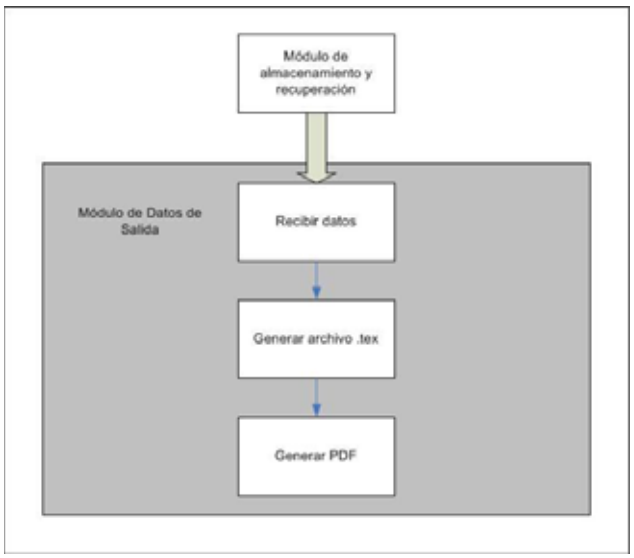

Figura 42. Módulo de Datos de Salida

Clases

Las clases que obtuvimos son:

- Interfaz Principal
- Interfaz Ordenamiento
- Interfaz Búsqueda
- Algoritmo
- Crea Archivo
- Comandos
- Propiedades

En la figura 43 tenemos el diagrama de clases.

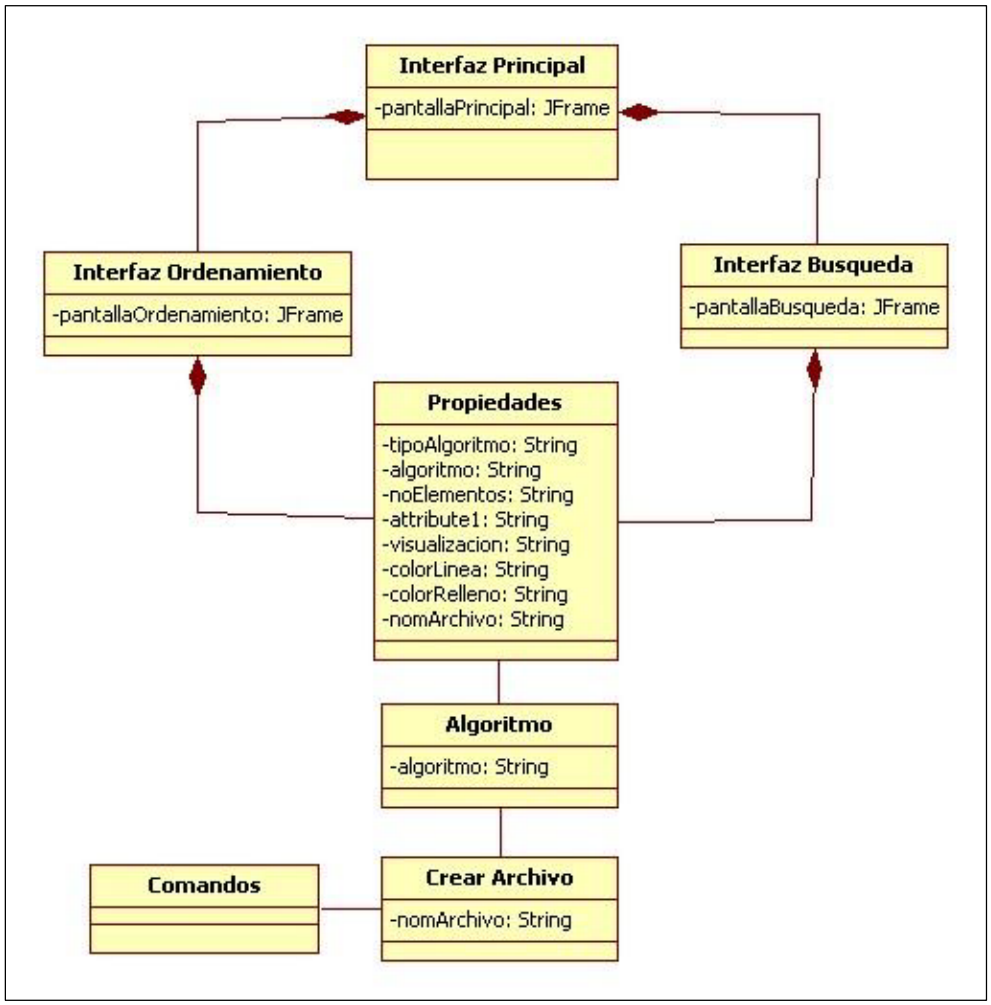

Figura 43. Diagrama de clases

Diccionario de clases

Nombre:		Interfaz principal
Descripción:		Esta clase incluye la interfaz gráfica de lo que es la pantalla principal de la aplicación. Desde aquí el usuario puede elegir el tipo de algoritmo (ordenamiento o búsqueda) y el algoritmo que se desea crear su visualización.
Atributo	**Tipo de datos**	**Visibilidad**
pantallaPrincipal	JFrame	Private

Nombre:		Interfaz Ordenamiento
Descripción:		Esta clase muestra la interfaz gráfica cuando se elige la opción de algoritmos de ordenamiento en la pantalla principal. En esta clase se eligen las propiedades que tendrá el algoritmo.
Atributo	**Tipo de datos**	**Visibilidad**
PantallaOrdenamiento	JFrame	Private

Nombre:		Interfaz Búsqueda
Descripción:		En esta clase se despliega la interfaz gráfica cuando se elige la opción de algoritmos de búsqueda en la pantalla principal. Al igual que la pantalla de ordenamiento, aquí se despliegan las propiedades que llevará la visualización.
Atributo	**Tipo de datos**	**Visibilidad**
PantallaBusqueda	JFrame	Private

Nombre:		Propiedades
Descripción:		Esta clase va almacenando todas las opciones seleccionadas por usuario a través de las pantallas
Atributo	**Tipo de datos**	**Visibilidad**
tipoAlgortimo	String	Private
Algoritmo	String	Private
noElementos	Int	Private
visualizacion	String	Private
colorLinea	String	Private
colorRelleno	String	Private

nomArchivo	String	Private

Nombre:		Algoritmo
Descripción:		En esta clase se tienen los algoritmos previamente guardados. Aquí se efectúa la ejecución del algoritmo para ir dándole forma según los propiedades elegidas por el usuario
Atributo	**Tipo de datos**	**Visibilidad**
Algoritmo	String	Private
Métodos		
Nombre		**Visibilidad**
ejecutaAlgoritmo(String):void		Public

Nombre:		CrearArchivo
Descripción:		En esta clase se crea el archivo final en .tex.
Atributo	**Tipo de datos**	**Visibilidad**
nomArchivo	String	Private
Métodos		
Nombre		**Visibilidad**
abreArchivo(String):void		Public
cerrarArchivo(String):void		Public

Nombre:		Comandos
Descripción:		Esta clase es utilizada para la ejecución de algunas llamadas al sistema para crear el documento PDF,
Atributo	**Tipo de datos**	**Visibilidad**
Métodos		
Nombre		**Visibilidad**
ejecutaComando(String):void		Public

Casos de uso

Para indicar cómo el usuario interactúa con la aplicación, en seguida mostramos el diagrama de casos de uso (figura 44) como la descripción de cada uno.

Casos de uso de la aplicación:

(P-1) Pantalla 1. Seleccionar algoritmo.
(P-2) Pantalla 2. Opciones de ordenamiento.
(P-3) Pantalla 3. Opciones de búsqueda.

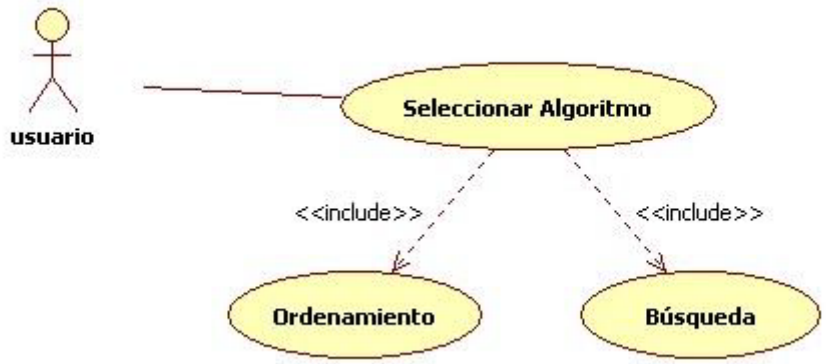

Figura 44. Diagrama de Casos de uso

Caso de uso seleccionar algoritmo

Caso de Uso	Seleccionar algoritmo
Actores	Usuario
Tipo	Inclusión
Propósito	El usuario podrá elegir qué algoritmo desea visualizar, ya sea un algoritmo de ordenamiento o un algoritmo de búsqueda.
Resumen	Se le presentará al usuario una lista de algoritmos de ordenamiento y una lista de algoritmos de búsqueda. El usuario podrá elegir qué visualización desea para la representación del algoritmo escogido.
Pre-condiciones	No hay pre-condiciones.
Post condiciones	Se cambiará a la ventana correspondiente según la elección.
Flujo principal	Se presentará la pantalla (P-2) si se escogió un algoritmo de ordenamiento (S-1) y un tipo de visualización (S-2). En otro caso se presentará la pantalla (P-3) si se escogió un algoritmo de búsqueda (S-3) y una visualización(S-4). Tendrá la opción de salir (S-5).
Subflujo	(S-1) Si se escogió la opción de un algoritmo de ordenamiento se activará una lista con varios algoritmos, donde el usuario podrá elegir uno de estos. (S-2) Se podrá elegir una opción entre una lista de nombres de visualizaciones para el algoritmo escogido. Una vez elegido el algoritmo de ordenamiento y su visualización se presentará la pantalla (P-2). (S-3)Se presentará una lista de algoritmos de búsqueda. (S-4)Se podrá elegir una opción entre una lista de nombres de visualizaciones para el algoritmo de búsqueda. Una vez escogido el algoritmo y la visualización se presentará la pantalla (P-3). (S-5) Se tiene la opción cancelar donde se podrá salir de la aplicación.
Excepciones	(E-1) Sólo se podrá elegir un tipo de algoritmo a la vez ya sea de ordenamiento o búsqueda.

Caso de uso opciones de ordenamiento

Caso de Uso	Opciones de ordenamiento
Actores	Usuario
Tipo	Inclusión
Propósito	Que el usuario pueda personalizar la visualización que más le agrade para la representación del algoritmo de ordenamiento que eligió.
Resumen	El usuario especificará cuántos elementos tendrá la visualización, así como el color de línea y el color del relleno. Por último le dará el nombre al archivo que se generará.
Pre-condiciones	Que se haya escogido un algoritmo de ordenamiento y una visualización.
Post condiciones	Se cambiará a la ventana correspondiente según la elección.
Flujo principal	Al escoger un algoritmo de ordenamiento se presentará la pantalla (P-2), está tendrá las opción de Número de elementos (S-1). Y para personalizar la visualización tendrá las opciones: Color de línea (S.2) y Color de relleno (S-3). El usuario introducirá el nombre del archivo que se generará (S-4). Una vez hecho todo esto podrá elegir la opción de Aceptar para generar el archivo (S-5). O bien Cancelar (S-6).
Subflujo	(S-1) Se podrá elegir cuántos elementos se desea ordenar este tendrá un valor máximo de 32 elementos dependiendo de la visualización que se haya escogido. (S-2) Se podrá elegir entre una lista de colores. (S-3)Se podrá elegir una opción para el color del relleno que tendrá la visualización. (S-4)El usuario introducirá el nombre del archivo que se generará. (S-5)Al elegir Aceptar se generará el archivo con todas las características que se especificaron anteriormente. (S-6)Al elegir cancelar se pasará a la pantalla principal (P-1).
Excepciones	No hay excepciones.

Caso de uso opciones de búsqueda

Caso de Uso	Opciones de búsqueda
Actores	Usuario
Tipo	Inclusión
Propósito	Que el usuario podrá elegir entre varias opciones para personalizar la visualización que se desea generar para el algoritmo de búsqueda.
Resumen	El usuario podrá especificar el número de elementos que contendrá la visualización. Escogerá que elemento desea buscar. Así como seleccionará el color de línea y el color del relleno. Al final le dará el nombre al archivo que se generará.
Pre-condiciones	Que se haya escogido un algoritmo de búsqueda y un tipo de visualización.
Post condiciones	Se cambiará a la ventana correspondiente según la elección.
Flujo principal	Cuando se ha escogido un algoritmo de búsqueda se mostrará la pantalla (P-3). La pantalla (P-3) tendrá que escoger el número de elementos (S-1). El usuario introducirá el elemento a buscar (S-2). Se podrá elegir color de línea (S-3) y el color de relleno (S-4). Se introducirá el nombre del archivo (S-5). Una vez hecho todo esto se podrá elegir Aceptar (S-6) o bien Salir (S-7).
Subflujo	(S-1) Se tiene la opción de elegir cuántos elementos se desea en la visualización, estos estarán dentro del rango de 1 a 32 elementos. (S-2) El usuario introducirá cuál es el elemento que se desea buscar, éste debe estar dentro del rango que se escogió. (S-3) El usuario podrá elegir qué color de línea desea. (S-4) Se podrá elegir dentro de una lista el color de relleno. (S-5) El usuario introducirá el nombre del archivo que desea generar. (S-6) Cuando el usuario haya especificado todas estas características podrá elegir Aceptar para generar el archivo. (S-7) Si el usuario desea salir se presentará posteriormente la pantalla principal (P-1).
Excepciones	No hay excepciones.

Diagramas de robustez

Para validar los casos de uso y asegurarnos que consideramos todos los elementos necesarios para llevar a cabo la funcionalidad del sistema presentamos los diagramas de robustez para los casos de uso: seleccionar algoritmo (figura 45), opción ordenamiento (figura 46) y opción de búsqueda (figura 47).

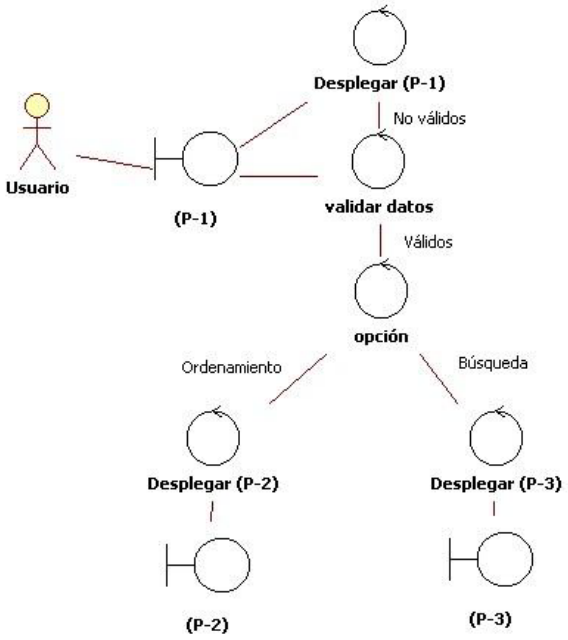

Figura 45. Diagrama de robustez para el caso de uso seleccionar algoritmo.

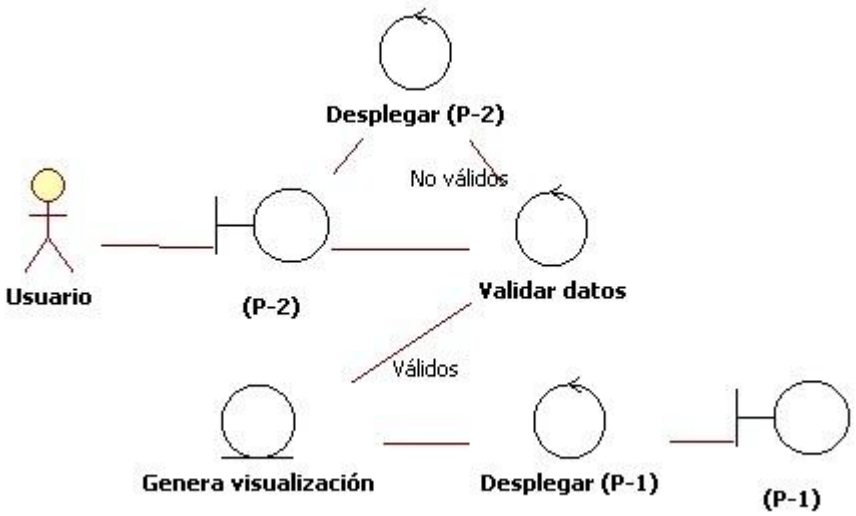

Figura 46. Diagrama de robustez para el caso de uso opción ordenamiento.

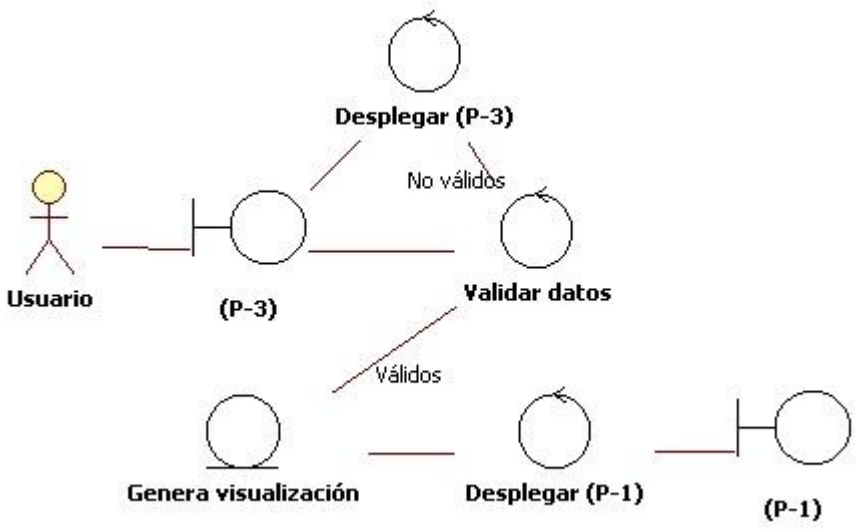

Figura 47. Diagrama de robustez para el caso de uso opción búsqueda.

Archivos incluidos en el proyecto

Los archivos que se entregan junto con este documento son principalmente dos.

El primero es el proyecto generado por el IDE de desarrollo Netbeans llamado ProyectoT. Se entrega de esta forma por si alguien estuviera interesado en modificar el código fuente sea más sencillo hacerlo. Dentro de esta carpeta esta otra llamado “src”, en donde se tiene todo el código fuente dividido en paquetes. Los paquetes son: algoritmos, manejo de archivos, comandos, pantallas y nodos.

En el paquete de algoritmos se incluyen clases en donde se han definido los algoritmos utilizados en este proyecto.

En el paquete de manejo de archivos, se tiene una clase para la creación de un archivo con java. Esta clase se utiliza para crear un archivo .tex.

En el paquete comandos se incluye una clase para hacer llamadas al sistema para la creación de un documento PDF, tomando en cuenta que se está trabajando bajo alguna distribución del sistema operativo GNU/Linux.

El paquete nodos contiene clases auxiliares para la creación de arreglos tipo nodo. Esta clase almacena las características que tendrá la visualización.

El último paquete contiene las pantallas de la aplicación. La clase principal de la aplicación está contenida en este paquete y se llama principal.java.

El segundo archivo es un .jar el cual esta listo para ejecutarse con el comando "java –jar GeneraVisualizaciones.jar".

Requisitos para ejecutar la aplicación

Para poder ejecutar la aplicación correctamente, se necesita instalar PGF (*Portable Graphics Format*) y JDK (*Java Development Kit*), preferentemente de Sun o NetBeans.

Instalar PGF en Debian 5

PGF es un paquete para la creación de gráficos, el cual podemos instalarlo de la siguiente forma:

#aptitude install pgf

Para obtener mayor información se puede consultar las siguientes páginas.

http://packages.debian.org/pgf
http://packages.debian.org/latex-xcolor

Instalar JDK de Sun en Debian 5

Bajar JDK 6, o el que desees, desde la pagina de Sun.
Necesitamos crear una carpeta para la instalación de preferencia en /usr/java

mkdir /usr/java

Le damos permisos sobre la carpeta al usuario (tú usuario).

chown -R nombre_de_tu_usuario /usr/java

Copiamos el archivo bajado (JDK 6 actualización 5) a la carpeta creada.

cp /home/usuario/java/jdk-6u5-linux-i586.bin /usr/java

Le asignamos permisos de ejecución al archivo descargado.

chmod a+x /usr/java/jdk-6u5-linux-i586.bin

Instalamos el Binario

./jdk-6u5-linux-i586.bin

Aceptamos la licencia, y continuamos con la instalación.
Luego damos permisos a nuestro usuario a la carpeta creada por la instalación.

chown -R inforux /usr/java/jdk1.6.0_05

Ahora editamos el Archivo /root/.bashrc para que se ejecute lo comando “javac” y “java” desde cualquier ubicación agregando dos líneas de código con un editor de texto, en esta caso utilizamos vim.

vim /root/.bashrc

Se agregan las siguientes líneas:

export JAVA_HOME=/usr/java/jdk1.6.0_05
export PATH=$JAVA_HOME/bin:$PATH

Y con esto quedara instalado y configurado JDK.

Instalar NetBeans

Descargamos el archivo para instalar NetBeans NetBeans 6.9.1. Una vez descargado le asignamos permiso de ejecución.

chmod +x ./netbeans-6.9-ml-linux.sh

y para instalarlo, solamente lo ejecutamos:

./netbeans-6.9-ml-linux.sh

Posteriormente aparecerá el instalador, tenemos que seguir todos los pasos, después ya tendremos NetBeans instalado en nuestro equipo.

Manual de usuario

El objetivo de esta aplicación es la comprensión del comportamiento de algoritmos de ordenamiento y búsqueda interna, mediante visualizaciones que se generarán en archivos PDF.

Si el usuario desea ver el comportamiento de un algoritmo de ordenamiento, éste elegirá la opción “Ordenamiento” (figura Núm. 48), automáticamente se activará la lista con los nombres de los algoritmos disponibles.

También se activará la lista con los nombres de los Tipos de visualización que se tienen. El usuario podrá observar en la parte inferior de la pantalla una vista previa de las visualizaciones.

Una vez hecho esto se dará Aceptar para pasar a la siguiente pantalla. En otro caso si se desea salir de la aplicación se escogerá la opción “Salir”.

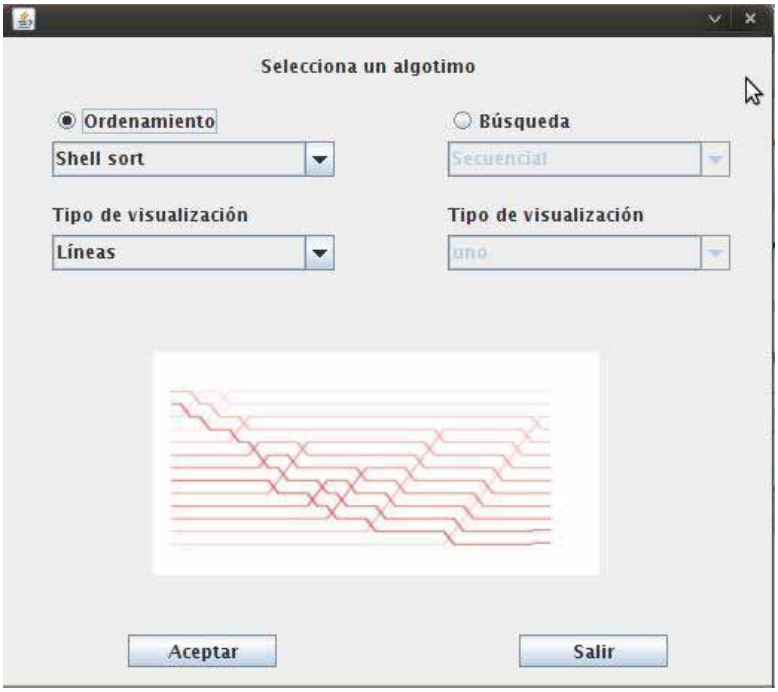

Figura 48. Se selecciona un algoritmo de ordenamiento

Ahora el usuario podrá personalizar la visualización del algoritmo de ordenamiento que desea obtener en la pantalla siguiente (figura 49) donde tendrá las siguientes opciones:

Número de elementos. El usuario podrá escoger cuántos elementos desea ordenar, el rango varía dependiendo de la visualización que se haya escogido, el número mínimo a escoger es de 3 elementos, mientras que el número máximo es de 32 elementos.

Color de línea: El usuario podrá escoger el color de línea que tendrán las figuras en la visualización.

Color de relleno: El usuario podrá escoger el color de relleno que tendrán las figuras en la visualización.

Nombre del archivo: En este espacio el usuario escribirá el nombre del archivo que desee generar.

Una vez especificadas estas características al dar aceptar se generará el archivo PDF. En caso contrario si se escoge Cancelar volverá a la pantalla principal.

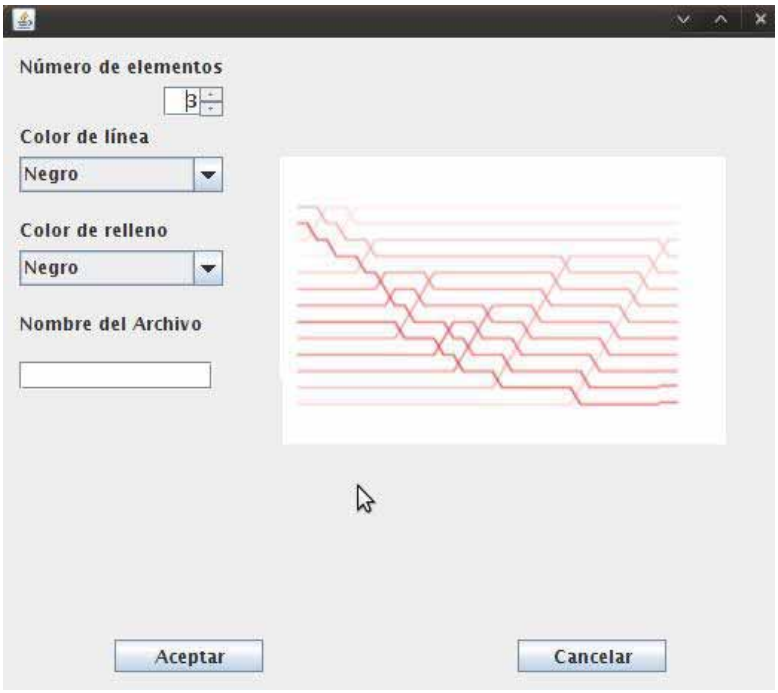

Figura 49. Opciones de ordenamiento

En caso de que el usuario elija un algoritmo de búsqueda (figura 50), de igual forma se activarán dos listas una para escoger que algoritmo se desea y otra para el tipo de visualización. El usuario podrá ver la vista previa en la parte inferior.

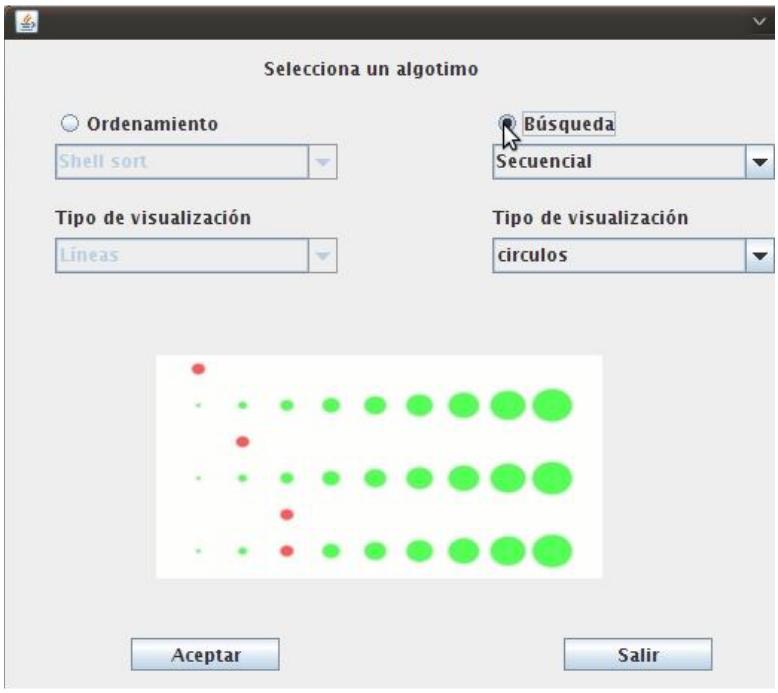

Figura 50. Se selecciona algoritmo de búsqueda.

Cuando ya se haya escogido el algoritmo de búsqueda y el tipo de visualización se pasará a la siguiente pantalla (figura 51) donde se podrá personalizar la visualización que se eligió.

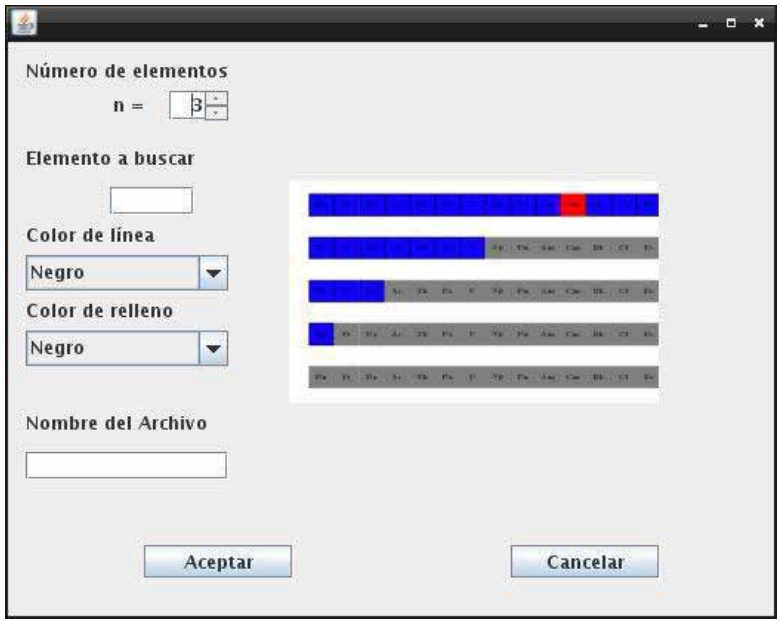

Figura 51. Opciones de búsqueda.

En esta pantalla se tiene la opción Número de elementos, la cual indica cuántos elementos se quieren en la visualización. El rango de estos puede variar dependiendo de la visualización que se haya escogido. El número mínimo es de 3 elementos, mientras que el número máximo es de 32 elementos.

En la siguiente opción el usuario puede introducir qué elemento desea buscar, éste debe de estar dentro del rango que se eligió. Para personalizar la visualización puede cambiar el color de línea y el color del relleno.

Por último se solicita el nombre del archivo. Una vez que se dieron todas estas propiedades, al aceptar se generará el archivo en PDF. En caso contrario si se elige Cancelar se regresará a la pantalla principal.

Conclusiones

Bocanegra Rodríguez Julio

En todas las carreras universitarias de Ciencias e Ingeniería se requiere de un conocimiento básico de algoritmos y de programación. Una persona novata en la programación desea aprender cómo realizar un programa que sea rápido y eficiente.

Una ayuda importante es la utilización de algoritmos de ordenamiento y búsqueda. Nosotros nos enfocamos a los algoritmos de ordenamiento y búsqueda interna, por lo que tratamos de dar una amplia visión de cómo es el comportamiento de dichos algoritmos.

El objetivo de este proyecto terminal es hacer fácil y sencillo, el aprendizaje del comportamiento de los algoritmos de ordenamiento y búsqueda. Por lo que se diseñaron diferentes visualizaciones, para la comprensión de los mismos.

Estas visualizaciones podrán ser personalizadas por el usuario, pues tendrá las opciones de escoger los colores, las formas, según sea de su agrado. El alumno que interactúe con esta aplicación no necesita tener conocimiento alguno de cómo se comportan los algoritmos, ya que la intención es facilitar su aprendizaje.

Tovar González Yolanda

Los estudiantes de Ing. en Computación que van iniciando o bien se van adentrando al mundo de la programación, es importante que conozcan varias técnicas o métodos para realizar programas eficientes.

Una herramienta de gran utilidad son los algoritmos de ordenamiento y búsqueda. Por ello la importancia de este proyecto terminal, cuyo objetivo es obtener visualizaciones, es decir, representaciones gráficas del comportamiento de algoritmos de ordenamiento y búsqueda interna, esto ayudará al alumno a facilitar su aprendizaje.

Estas muestran iteración por iteración los intercambios de los elementos en el arreglo. Se utilizaron diferentes visualizaciones, por ejemplo, usamos figuras,

letras, colores, etc. Estos elementos ayudan a que la comprensión del comportamiento de los algoritmos sea más fácil, puesto que se da una idea más clara, ya que muchos de nosotros somos visuales, aprendemos viendo y observando patrones.

Todo esto con la finalidad de que el alumno pueda comprender el comportamiento de dichos algoritmos y sea capaz de aplicarlos al momento de programar.

Código fuente

La organización de los paquetes se muestra en la figura 52.

Figura 52. Organización de paquetes

Paquete ManejoArchivos

Archivo CrearArchivo.java

```
/*
 * Clase utilizada para el manejo de archivos
                                        */

package ManejoArchivos;


import java.io.FileNotFoundException;
import java.lang.SecurityException;
import java.util.*;

public class CrearArchivo {
      private Formatter salida;
```

```
        // permite al usuario abrir el archivo
          public void abrirArchivo(String nombre)
          {
             try
             {
                salida = new Formatter( nombre );
             } // fin de try
             catch ( SecurityException securityException )
             {
                System.err.println(
                   "No tiene acceso de escritura a este archivo." );
                System.exit( 1 );
             } // fin de catch
             catch ( FileNotFoundException filesNotFoundException )
             {
                System.err.println( "Error al crear el archivo." );
                System.exit( 1 );
             } // fin de catch
          } // fin del metodo abrirArchivo

         public void cerrarArchivo()
          {
             if ( salida != null )
                salida.close();
          } // fin del mÃ©todo cerrarArchivo

        public void escribe(String cadena)
        {
             salida.format(cadena);
             salida.format("\n");
        }
}
```

Paquete ManejoComandos

Archivo Comandos.java

```
 /* Clase utilizada para hacer llamdas al sistema*/

package ManejoComandos;


import java.io.*;

public class Comandos {
      public void ejecutaComando( String command)
      {
      try
      {
      final Process process = Runtime.getRuntime().exec(command);
      new Thread(){
            public void run(){
                  try{
                        InputStream is = process.getInputStream();
                        byte[] buffer = new byte[1024];
                        for(int count = 0; (count = is.read(buffer)) >=
0;){
                              System.out.write(buffer, 0, count);
                        }
                  }
                  catch(Exception e){
                        e.printStackTrace();
                  }
            }
      }.start();
      new Thread(){
            public void run(){
                  try{
                        InputStream is = process.getErrorStream();
                        byte[] buffer = new byte[1024];
                        for(int count = 0; (count = is.read(buffer)) >=
0;){
                              System.err.write(buffer, 0, count);
                        }
                  }
                  catch(Exception e){
                        e.printStackTrace();
                  }
            }
      }.start();

      int returnCode = process.waitFor();
      System.out.println("Return code = " + returnCode);
      }
      catch (Exception e){
            e.printStackTrace();
      }
      }
}
```

Paquete Nodos

Archivo Nodo.java

```
/*
 * Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class Nodo {
        //Coordenadas X y Y
      private double x;
      private double y;
      private int trans;
      private String codigo;
        private String color;
        private double radio;

        //Constructores
      public Nodo(){

      }

        public Nodo(int trans, String codigo)
      {
            x=0.0;
            y=0.0;
            this.trans = trans;
            this.codigo = codigo;
      }

        public Nodo(double x, double y, int trans,String codigo)
      {
            this.x=x;
            this.y=y;
            this.trans=trans;
            this.codigo=codigo;
      }

        public Nodo(double x, double y,String color,double radio){
      this.x = x;
      this.y = y;
      this.color = color;
      this.radio = radio;
      }


        //MÃ©todos get y set para los atributos de la clase
        public void agregaCoordenadas(String cad)
      {
            codigo = codigo + cad;
      }
```

```
public void agregaColor(String color)
{
      codigo = codigo + color + "!" + trans + "] ";
}

public void sumaX(double suma)
{
      x+=suma;
}

public void agregarCodigo(String codigo)
{
      this.codigo += codigo;
}

public void setX(double x) {
      this.x = x;
}
  public void setCoorx(double x)
{
      this.x=x;
}
public double getX() {
      return x;
}
public void setY(double y) {
      this.y = y;
}
  public void setCoory(double y)
{
      this.y=y;
}
public double getY() {
      return y;
}
public void setTrans(int trans) {
      this.trans = trans;
}
public int getTrans() {
      return trans;
}
public void setCodigo(String codigo) {
      this.codigo = codigo;
}
public String getCodigo() {
      return codigo;
}

public void imprime()
{
      System.out.printf("%f %f %d\n",x,y,trans);
}

  public void setColor(String color) {
      this.color = color;
  }
```

```
        public String getColor() {
            return color;
        }
        public void setRadio(double radio) {
            this.radio = radio;
        }
        public double getRadio() {
            return radio;
        }
}
```

Archivo Nodo2.java

```
/*
* Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class Nodo2 {
    //Atributos de la clase
private double x;
private double y;
private String color;
private double largo;
private double ancho;

//Constructor
public Nodo2(double x, double y,double largo,String color){
      this.x = x;
      this.y = y;
      this.color = color;
      this.largo= largo;

}
//MÃ©todos get y set para los atributos
public void setX(double x) {
      this.x = x;
}
public double getX() {
      return x;
}
public void setY(double y) {
      this.y = y;
}
public double getY() {
      return y;
}
public void setColor(String color) {
      this.color = color;
}
public String getColor() {
      return color;
}
public void setLargo(double largo) {
```

```
      this.largo= largo;
}
public double getLargo() {
      return largo;
}
public void setAncho(double ancho) {
      this.ancho= ancho;
}
public double getAncho() {
      return ancho;
}

}
```

Archivo Nodo3.java

```
/*
 * Este nodo es utilizado en las visualizaciones de estrellas
 */

package Nodos;

public class Nodo3 {
    private double x;
      private double y;
      private int picos;

      public Nodo3(double x, double y,int picos){
            this.x = x;
            this.y = y;
            this.picos = picos;
      }

      public void setX(double x) {
            this.x = x;
      }

      public double getX() {
            return x;
      }

      public void setY(double y) {
            this.y = y;
      }

      public double getY() {
            return y;
      }

      public void setPicos(int picos) {
            this.picos = picos;
      }

      public int getPicos() {
            return picos;
      }
```

```
}
```

Archivo Nodo4.java

```
/*
* Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class Nodo4 {
    private double x;
      private double y;
      private int atomico;
      private String simbolo;

      public Nodo4(){
      }

      public Nodo4(double x, double y,int atomico,String simbolo){
            this.x = x;
            this.y = y;
            this.atomico = atomico;
            this.simbolo = simbolo;
      }

      public void setX(double x) {
            this.x = x;
      }

      public double getX() {
            return x;
      }

      public void setY(double y) {
            this.y = y;
      }

      public double getY() {
            return y;
      }

      public void setAtomico(int atomico) {
            this.atomico = atomico;
      }

      public int getAtomico() {
            return atomico;
      }

      public void setSimbolo(String simbolo) {
            this.simbolo = simbolo;
      }
```

```
	public String getSimbolo() {
		return simbolo;
	}

	public void imprime(){
		System.out.println("x: "+x+", y: "+y+", a: "+atomico+", s:
"+simbolo);
	}
}
```

Archivo Nodo5.java

```
/*
* Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class Nodo5 {
    private double x;
    private double y;
    private int atomico;
    private String simbolo;
    private String relleno;
    private String linea;

      public Nodo5(){
      }

      public Nodo5(double x, double y,int atomico,String simbolo,String
linea,String relleno){
            this.x = x;
            this.y = y;
            this.atomico = atomico;
            this.simbolo = simbolo;
            this.linea = linea;
            this.relleno = relleno;
      }

      public String getRelleno(){
            return relleno;
      }

      public void setRelleno(String relleno){
            this.relleno = relleno;
      }

      public String getLinea(){
            return linea;
      }

      public void setLinea(String linea){
```

```
            this.linea = linea;
      }

      public void setX(double x) {
            this.x = x;
      }

      public double getX() {
            return x;
      }

      public void setY(double y) {
            this.y = y;
      }

      public double getY() {
            return y;
      }

      public void setAtomico(int atomico) {
            this.atomico = atomico;
      }

      public int getAtomico() {
            return atomico;
      }

      public void setSimbolo(String simbolo) {
            this.simbolo = simbolo;
      }

      public String getSimbolo() {
            return simbolo;
      }

      public void imprime(){
            System.out.println("x: "+x+", y: "+y+", a: "+atomico+", s:
"+simbolo);
      }
}
```

Archivo Nodo6.java

```
/*
* Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class Nodo6 {
    private double x;
      private double y;
      private int picos;
      private String relleno;
      private String linea;
```

```
	public Nodo6(double x, double y,int picos,String linea,String
relleno){
		this.x = x;
		this.y = y;
		this.picos = picos;
		this.linea = linea;
		this.relleno = relleno;
	}

	public String getRelleno(){
		return relleno;
	}

	public void setRelleno(String relleno){
		this.relleno = relleno;
	}

	public String getLinea(){
		return linea;
	}

	public void setLinea(String linea){
		this.linea = linea;
	}

	public void setX(double x) {
		this.x = x;
	}

	public double getX() {
		return x;
	}

	public void setY(double y) {
		this.y = y;
	}

	public double getY() {
		return y;
	}

	public void setPicos(int picos) {
		this.picos = picos;
	}

	public int getPicos() {
		return picos;
	}
}
```

Archivo Nodo7.java

```
package Nodos;
```

```
public class Nodo7 {
    private double x;
private double y;
private String color;
private int letra;

public Nodo7(double x, double y,String color,int letra){
      this.x = x;
      this.y = y;
      this.color = color;
      this.letra = letra;
      }
public void setX(double x) {
      this.x = x;
}
public double getX() {
      return x;
}
public void setY(double y) {
      this.y = y;
}
public double getY() {
      return y;
}
public void setColor(String color) {
      this.color = color;
}
public String getColor() {
      return color;
}
public void setLetra(int letra) {
      this.letra= letra;
}
public int getLetra() {
      return letra;
}

public void imprime(){
      System.out.printf("%f,%f,%d,%s\n",x,y,letra,color);
      }
}
```

Archivo Nodo8.java

```
package Nodos;


public class Nodo8 {
    private double x;
private double y;
private String color;
private int radio;
```

```
public Nodo8(double x, double y,String color,int radio){
      this.x = x;
      this.y = y;
      this.color = color;
      this.radio = radio;
      }
public void setX(double x) {
      this.x = x;
}
public double getX() {
      return x;
}
public void setY(double y) {
      this.y = y;
}
public double getY() {
      return y;
}
public void setColor(String color) {
      this.color = color;
}
public String getColor() {
      return color;
}
public void setRadio(int radio) {
      this.radio = radio;
}
public int getRadio() {
      return radio;
}

public void imprime(){
      System.out.printf("%f,%f,%d,%s\n",x,y,radio,color);
      }
}
```

Archivo NodoElementos.java

```
/*
 * Nodo utilizado para crear arreglos con las propiedades
 * de la visualizaciÃ³n
 */

package Nodos;

public class NodoElementos {
    private int atomico;
      private String simbolo;

      public NodoElementos(String simbolo,int atomico){
            this.atomico = atomico;
            this.simbolo = simbolo;
      }

      public void setAtomico(int atomico) {
```

```
            this.atomico = atomico;
      }

      public int getAtomico() {
            return atomico;
      }

      public void setSimbolo(String simbolo) {
            this.simbolo = simbolo;
      }

      public String getSimbolo() {
            return simbolo;
      }
}
```

Paquete Pantallas

Archivo Busqueda.java

```
/*
 *Pantalla para asignar propiedades a los algoritos de bÃºsqueda
 *
 */
package Pantallas;

import Algoritmos.*;
import ManejoComandos.Comandos;
import javax.swing.JOptionPane;

public class Busqueda extends javax.swing.JFrame {
    private javax.swing.JFrame padre;
    private Propiedades p;
    /** Creates new form Ordenamiento */
    public Busqueda(javax.swing.JFrame parent, boolean modal, Propiedades
prop) {
        p = prop; //Propiedades del algortimo de bÃºsqueda
        padre=parent; //Referencia a la pantalla principal
        initComponents(); //Dibujamos la pantalla
        setLocation(250,180); //LocalizaciÃ³n de la ventana en pantalla
        updateLabel(p.getVisualizacion()); //Muestra imagen en la ventana
    }

    /** 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">//GEN-
BEGIN:initComponents
    private void initComponents() {

        labelElementos = new javax.swing.JLabel();
        spinnerRango = new javax.swing.JSpinner();
```

```
        labelColorLinea = new javax.swing.JLabel();
        comboColorLinea = new javax.swing.JComboBox();
        labelRelleno = new javax.swing.JLabel();
        comboColorRelleno = new javax.swing.JComboBox();
        labelNombreArchivo = new javax.swing.JLabel();
        textNombreArchivo = new javax.swing.JTextField();
        botonAceptar = new javax.swing.JButton();
        botonCancelar = new javax.swing.JButton();
        labelNumeroBuscar = new javax.swing.JLabel();
        textElemento = new javax.swing.JTextField();
        imagenLabel = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        labelElementos.setText("NÃ°mero de elementos");

        spinnerRango.setModel(new javax.swing.SpinnerNumberModel(3, 3,
16, 1));

        labelColorLinea.setText("Color de lÃnea");

        comboColorLinea.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "Negro", "Amarillo", "Anaranjado", "Azul", "Blanco", "CafÃ©",
"Rosa", "Verde" }));

        labelRelleno.setText("Color de relleno");

        comboColorRelleno.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Negro", "Amarillo",
"Anaranjado", "Azul", "Blanco", "CafÃ©", "Rosa", "Verde" }));

        labelNombreArchivo.setText("Nombre del Archivo");

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

        botonCancelar.setText("Cancelar");
        botonCancelar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                botonCancelarActionPerformed(evt);
            }
        });

        labelNumeroBuscar.setText("Elemento a buscar");

        imagenLabel.setText("imagen");

        jLabel1.setText("n = ");
```

```
        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.TR
AILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(spinnerRango,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(labelElementos))
                .addContainerGap(369, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addComponent(textNombreArchivo,
javax.swing.GroupLayout.DEFAULT_SIZE, 137, Short.MAX_VALUE)
                            .addComponent(comboColorRelleno, 0, 137,
Short.MAX_VALUE)
                            .addComponent(comboColorLinea, 0, 137,
Short.MAX_VALUE)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(labelColorLinea)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 50,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(labelRelleno)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 37,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(labelNombreArchivo)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
```

```
                                    .addComponent(textElemento,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(labelNumeroBuscar))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(41, 41, 41)
                        .addComponent(imagenLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(92, 92, 92)
                        .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 148,
Short.MAX_VALUE)
                        .addComponent(botonCancelar,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(78, 78, 78))
        );

        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {comboColorLinea, comboColorRelleno});

        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {botonAceptar, botonCancelar});

        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(labelElementos)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(spinnerRango,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(labelNumeroBuscar)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(textElemento,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(labelColorLinea)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(comboColorLinea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(labelRelleno)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(comboColorRelleno,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(31, 31, 31)
                        .addComponent(labelNombreArchivo)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(textNombreArchivo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(imagenLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 196,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(44, 44, 44)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(botonCancelar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(26, 26, 26))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    //MÃ©todo que cambia el nombre del color de espaÃ±ol a inglÃ©s
    public String convierteColor(String color){
         if( color == "Negro" )
            return "black";
         else if( color == "Amarillo" )
            return "yellow";
         else if( color == "Anaranjado" )
            return "orange";
         else if( color == "Azul" )
            return "blue";
```

```
          else if( color == "Blanco" )
             return "white";
          else if( color == "CafÃ©" )
             return "brown";
          else if( color == "Rosa" )
             return "pink";
          else
             return "green";
    }

    //Cuando se da clic en el botÃ³n cancelar, regresa a la pantalla
principal
    private void botonCancelarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonCancelarActionPerformed
        // TODO add your handling code here:
        setVisible(false);
        padre.setVisible(true);
    }//GEN-LAST:event_botonCancelarActionPerformed

    //Se crea el objeto segÃºn el algortimo seleccionado
    void binariaElementos(int n,int elemento){
        String nomArchivo = textNombreArchivo.getText().toString();
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        BinariaEl binaria = new BinariaEl();
        binaria.creaVector(n, elemento-1, linea, relleno);
        binaria.binaria(nomArchivo+".tex", elemento-1);
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    void binariaEstrellas(int n,int elemento){
        String nomArchivo = textNombreArchivo.getText().toString();
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        BinariaE binaria = new BinariaE();
        binaria.creaVector(n, elemento+2, linea, relleno);
        binaria.binaria(nomArchivo+".tex", elemento+2);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    void secuencialCirculos(int n,int elemento){
        String nomArchivo = textNombreArchivo.getText().toString();
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
```

```
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        SecuencialC sec = new SecuencialC();
        sec.creaVector(n, relleno, elemento);
        sec.secuencial(nomArchivo+".tex");
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    void fibonacciCirculos(int n,int elemento){
        String nomArchivo = textNombreArchivo.getText().toString();
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        Fibonacci fib = new Fibonacci();
        fib.creaVector(n, relleno, elemento);
        fib.fibonacci(nomArchivo+".tex");
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    void interpolacionLetras(int n,int elemento){
        String nomArchivo = textNombreArchivo.getText().toString();
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        Interpolacion in = new Interpolacion();
        in.creaVector(n);
        in.interpolacion(nomArchivo+".tex",relleno ,elemento);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Cuando se da clic en el botÃ³n aceptar se vÃ¡lidan los datos
    private void botonAceptarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonAceptarActionPerformed
        String evento = evt.getActionCommand();
        String texto = textNombreArchivo.getText(); //Se obtiene el
nombre del archivo
        String elemento = textElemento.getText(); // Se obtiene el
elemento a buscar
        int n = Integer.parseInt(spinnerRango.getValue().toString());
//Rango de los datos
        elemento.trim(); //Se quitan los espacios en blanco
        if ("".equals(texto)){ //Se verifica que no este vacÃa la opciÃ³n
de elemento a buscar
            String mensaje = "Introduce el nombre del Archivo";
```

```
                JOptionPane.showMessageDialog(null, mensaje);
        }
        else if (elemento.length() == 0){
            String mensaje = "Introduce el elemento a buscar";
                JOptionPane.showMessageDialog(null, mensaje);
        }
        else if (elemento.length() > 2){ //Se verifica que solo se
ingresen nÃ°meros de 2 dÃgitos
            String mensaje = "Introduce un nÃ°mero de dos digÃtos";
                JOptionPane.showMessageDialog(null, mensaje);
        }
        else if (elemento.length() < 3){
            int numero=Integer.parseInt(elemento); //Se obtinene el rango
            if( numero > n ){ //Se verifica que el elemento a buscar este
dentro del rengo
            String mensaje = "El elemento a buscar tiene que estar entre
1 y n";
                JOptionPane.showMessageDialog(null, mensaje);
            }
            else{//SÃ todo es correcto entonces se manda a llamar al
mÃ©todo correspondiente
                if(p.getAlgortimo().equals("Binaria")){
                   if(p.getVisualizacion().equals("Estrella"))
                         binariaEstrellas(n,numero);
                   else if(p.getVisualizacion().equals("Elemento"))
                               binariaElementos(n,numero);
                }
                else if(p.getAlgortimo().equals("Secuencial")){
                   if(p.getVisualizacion().equals("CÃrculo"))
                         secuencialCirculos(n,numero);
                }
                else if(p.getAlgortimo().equals("InterpolaciÃ³n")){
                   if(p.getVisualizacion().equals("Letra"))
                         interpolacionLetras(n,numero);
                }
                else if(p.getAlgortimo().equals("Fibonacci")){
                   if(p.getVisualizacion().equals("CÃrculo"))
                         fibonacciCirculos(n,numero);
                }
                setVisible(false);
                padre.setVisible(true);
            }
        }
    }//GEN-LAST:event_botonAceptarActionPerformed

//MÃ©todo que crea la imagen mostrada en pantalla
    protected boolean updateLabel(String name) {
        javax.swing.ImageIcon icon = createImageIcon("../imagenes/"+name
+ ".gif");
        imagenLabel.setIcon(icon);
        imagenLabel.setToolTipText("A drawing of a " +
name.toLowerCase());
        if (icon != null) {
            imagenLabel.setText(null);
            return true;
        } else {
```

```
            imagenLabel.setText("VisualizaciÃ³n no disponible para
algortimo elegido");
            return false;
        }
    }

     /** Regresa un ImageIcon, o null si la ruta no es vÃ¡lida. */
    protected static javax.swing.ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = Main.class.getResource(path);
        if (imgURL != null) {
            return new javax.swing.ImageIcon(imgURL);
        } else {
            //System.err.println("Couldn't find file: " + path);
            return null;
        }
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                //new Busqueda().setVisible(true);
                Busqueda dialog = new Busqueda(new javax.swing.JFrame(),
true, new Propiedades());
                dialog.addWindowListener(new
java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent
e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton botonAceptar;
    private javax.swing.JButton botonCancelar;
    private javax.swing.JComboBox comboColorLinea;
    private javax.swing.JComboBox comboColorRelleno;
    private javax.swing.JLabel imagenLabel;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel labelColorLinea;
    private javax.swing.JLabel labelElementos;
    private javax.swing.JLabel labelNombreArchivo;
    private javax.swing.JLabel labelNumeroBuscar;
    private javax.swing.JLabel labelRelleno;
    private javax.swing.JSpinner spinnerRango;
    private javax.swing.JTextField textElemento;
    private javax.swing.JTextField textNombreArchivo;
    // End of variables declaration//GEN-END:variables

}
```

Archivo Main.java

```
package Pantallas;

import javax.swing.JOptionPane;


public class Main extends javax.swing.JFrame {
    //Propiedades para la visualizacion
    private String visualizacion;
    private String algoritmo;
    private String opcion;
    private Propiedades p;

    public Main() {
        p = new Propiedades();//Se crea un objeto para guardar las
propiedades
        p.setNoElementos(32); //Valores iniciales
        visualizacion = "LÃneas";
        opcion = "Ordenamiento";
        algoritmo = "Shell sort";
        initComponents();//Se dibuja la pantalla
        setLocation(250,180); //localizaciÃ³n de la ventana en pantalla
    }

    /** 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">//GEN-
BEGIN:initComponents
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        tituloSelecciona = new javax.swing.JLabel();
        radioOrdenmamiento = new javax.swing.JRadioButton();
        radioBusqueda = new javax.swing.JRadioButton();
        comboOrdenamiento = new javax.swing.JComboBox();
        comboBusqueda = new javax.swing.JComboBox();
        visualizacionOrdena = new javax.swing.JComboBox();
        visualizacionBusca = new javax.swing.JComboBox();
        tipoVisualizacion1 = new javax.swing.JLabel();
        tipoVisualizacion2 = new javax.swing.JLabel();
        imagenLabel = new javax.swing.JLabel();
        botonAceptar = new javax.swing.JButton();
        botonSalir = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
```

```
            }
        });

        tituloSelecciona.setText("Selecciona un algotimo");

        buttonGroup1.add(radioOrdenmamiento);
        radioOrdenmamiento.setSelected(true);
        radioOrdenmamiento.setText("Ordenamiento");
        radioOrdenmamiento.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                radioOrdenmamientoActionPerformed(evt);
            }
        });

        buttonGroup1.add(radioBusqueda);
        radioBusqueda.setText("BÃºsqueda");
        radioBusqueda.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                radioBusquedaActionPerformed(evt);
            }
        });

        comboOrdenamiento.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Shell sort", "SelecciÃ³n
directa", "InserciÃ³n directa", "Intercambio directo", "Mergesort",
"Quicksort", "Ordenamiento del duende", "Radix sort" }));
        comboOrdenamiento.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                comboOrdenamientoActionPerformed(evt);
            }
        });

        comboBusqueda.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "Secuencial", "Binaria", "Fibonacci", "InterpolaciÃ³n" }));
        comboBusqueda.setEnabled(false);
        comboBusqueda.setFocusCycleRoot(true);
        comboBusqueda.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                comboBusquedaActionPerformed(evt);
            }
        });

        visualizacionOrdena.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "LÃneas", "Degradado",
"Figuras", "PirÃ¡mide", "Estrellas", "Elementos", "Letras" }));
        visualizacionOrdena.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                visualizacionOrdenaActionPerformed(evt);
            }
        });
```

```
        visualizacionBusca.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Estrella", "Elemento",
"Letra", "CÃrculo" }));
        visualizacionBusca.setEnabled(false);
        visualizacionBusca.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                visualizacionBuscaActionPerformed(evt);
            }
        });

        tipoVisualizacion1.setText("Tipo de visualizaciÃ³n");

        tipoVisualizacion2.setText("Tipo de visualizaciÃ³n");

        imagenLabel.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/LÃneas.gif")));
// NOI18N

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

        botonSalir.setText("Salir");
        botonSalir.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                botonSalirActionPerformed(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(175, 175, 175)
                .addComponent(tituloSelecciona)
                .addContainerGap(199, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addGap(34, 34, 34)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addComponent(tipoVisualizacion1)
                            .addComponent(visualizacionOrdena,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequentialGroup()
                                .addGap(51, 51, 51)
                                .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 76,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(visualizacionBusca,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(tipoVisualizacion2))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                .addComponent(botonSalir,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(42, 42, 42))))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addComponent(comboOrdenamiento, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(radioOrdenmamiento,
javax.swing.GroupLayout.DEFAULT_SIZE, 264, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                            .addComponent(radioBusqueda)
                            .addComponent(comboBusqueda, 0, 109,
Short.MAX_VALUE))))
                .addGap(28, 28, 28))
            .addGroup(layout.createSequentialGroup()
                .addGap(102, 102, 102)
                .addComponent(imagenLabel)
                .addContainerGap(116, Short.MAX_VALUE))
        );

        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {comboBusqueda, comboOrdenamiento,
visualizacionBusca, visualizacionOrdena});

        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {botonAceptar, botonSalir});
```

```
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(tituloSelecciona)
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
                    .addComponent(radioBusqueda)
                    .addComponent(radioOrdenmamiento))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(comboBusqueda,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(comboOrdenamiento,
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.LE
ADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(tipoVisualizacion1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(visualizacionOrdena,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(tipoVisualizacion2)
                        .addGap(6, 6, 6)
                        .addComponent(visualizacionBusca,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 55,
Short.MAX_VALUE)
                .addComponent(imagenLabel)
                .addGap(40, 40, 40)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(botonSalir,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {botonAceptar, botonSalir});

        pack();
    }// </editor-fold>//GEN-END:initComponents

/*
 * Damos las funciones al Radio BotÃ³n de BÃºsqueda
 */
    private void radioBusquedaActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_radioBusquedaActionPerformed
        comboBusqueda.setEnabled(true); //Activamos el comboBox de
BÃºsqueda
        comboOrdenamiento.setEnabled(false); //Desactivamos el comboBox
de Ordenamiento
        visualizacionBusca.setEnabled(true); //Activamos el comboBox
visualizacion BÃºsqueda
        visualizacionOrdena.setEnabled(false); //Desactivamos el comboBox
visualizacion Ordenamiento
        opcion=evt.getActionCommand();
        String evento = evt.getActionCommand();
        visualizacion = (String)visualizacionBusca.getSelectedItem();
        algoritmo = (String)comboBusqueda.getSelectedItem();
        p.setNoElementos(16);

        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida

        /*if(algoritmo.equals("Fibonacci")){
            if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else*/ if(visualizacion.equals("Letra") ){
          if(algoritmo.equals("InterpolaciÃ³n")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("CÃrculo")){
          if(algoritmo.equals("Secuencial") ||
algoritmo.equals("Fibonacci")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("Estrella")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
```

```
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("Elemento")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else{
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }
    }//GEN-LAST:event_radioBusquedaActionPerformed

/*
 * Damos las funciones al Radio BotÃ³n de Ordenamiento
 */
    private void
radioOrdenmamientoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_radioOrdenmamientoActionPerformed
        //Ponemos activados los comboBox de Ordenaiento y desactivamos
los de BÃºsqueda
        comboBusqueda.setEnabled(false);
        comboOrdenamiento.setEnabled(true);
        visualizacionBusca.setEnabled(false);
        visualizacionOrdena.setEnabled(true);
        opcion=evt.getActionCommand();
        //NOmbre de la visualizaciÃ³n y del algoritmo
        visualizacion = (String)visualizacionOrdena.getSelectedItem();
        algoritmo = (String)comboOrdenamiento.getSelectedItem();

        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida
        /*if(algoritmo.equals("Mergesort") ){
            if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else*/ if(visualizacion.equals("Letras")){
          if(algoritmo.equals("Radix sort") ||
algoritmo.equals("Mergesort")){
              p.setNoElementos(26);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("LÃneas")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende") ||
```

```
                algoritmo.equals("Quicksort") ){
              p.setNoElementos(32);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Figuras") ||
visualizacion.equals("PirÃ¡mide")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(8);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Estrellas") ||
visualizacion.equals("Elementos")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(16);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
         else{
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }
    }//GEN-LAST:event_radioOrdenmamientoActionPerformed

    //Al dar clic sobre el combo de los algortimos de ordenamienti
    private void
comboOrdenamientoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_comboOrdenamientoActionPerformed
        String evento = evt.getActionCommand();
        algoritmo = (String)comboOrdenamiento.getSelectedItem();
        visualizacion = (String)visualizacionOrdena.getSelectedItem();
        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida
        if(visualizacion.equals("Letras") ){
            System.out.println(algoritmo+"  "+visualizacion);
          if(algoritmo.equals("Radix sort") ||
algoritmo.equals("Mergesort")){
              p.setNoElementos(26);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
```

```
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("LÃneas")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende" ) ||
                algoritmo.equals("Quicksort") ){
              p.setNoElementos(32);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
            else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Figuras") ||
visualizacion.equals("PirÃ¡mide")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(8);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Estrellas") ||
visualizacion.equals("Elementos")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(16);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
         else {
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }

    }//GEN-LAST:event_comboOrdenamientoActionPerformed

    private void
visualizacionOrdenaActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_visualizacionOrdenaActionPerformed
        //Obtenemos el valos selecciona del comboBox que no da el nombre
```

```
        // de la visualizaciÃ³n
        String evento = evt.getActionCommand();
        visualizacion = (String)visualizacionOrdena.getSelectedItem();
        algoritmo = (String)comboOrdenamiento.getSelectedItem();
        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida
        /*if(algoritmo.equals("Mergesort")){
            if(!updateLabel("ninguna"))
                visualizacion = null;
        }*/
        if(visualizacion.equals("Letras")){
          if(algoritmo.equals("Radix sort") ||
algoritmo.equals("Mergesort")){
              p.setNoElementos(26);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("LÃneas")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")
                || algoritmo.equals("Quicksort") ){
              p.setNoElementos(32);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Figuras") ||
visualizacion.equals("PirÃ¡mide")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo") ||
algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(8);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else if(visualizacion.equals("Estrellas") ||
visualizacion.equals("Elementos")){
          if(algoritmo.equals("Shell sort") ||
algoritmo.equals("SelecciÃ³n directa") || algoritmo.equals("InserciÃ³n
directa") ||
                algoritmo.equals("Intercambio directo")
||algoritmo.equals("Ordenamiento del duende")){
              p.setNoElementos(16);
              if(!updateLabel(visualizacion))
                    visualizacion = null;
```

```
          }
          else if(!updateLabel("ninguna"))
                visualizacion = null;
        }
         else{
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }

    }//GEN-LAST:event_visualizacionOrdenaActionPerformed

    private void botonSalirActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonSalirActionPerformed
        System.exit(0);
    }//GEN-LAST:event_botonSalirActionPerformed

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
        System.exit(0);
    }//GEN-LAST:event_formWindowClosing

    private void
visualizacionBuscaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_visualizacionBuscaActionPerformed
        //Obtenemos el valos selecciona del comboBox que no da el nombre
        // de la visualizaciÃ³n
        String evento = evt.getActionCommand();
        visualizacion = (String)visualizacionBusca.getSelectedItem();
        algoritmo = (String)comboBusqueda.getSelectedItem();
        p.setNoElementos(16);
        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida
        /*if(algoritmo.equals("Fibonacci")){
            if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else*/ if(visualizacion.equals("Letra") ){
          if(algoritmo.equals("InterpolaciÃ³n")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("CÃrculo")){
          if(algoritmo.equals("Secuencial") ||
algoritmo.equals("Fibonacci")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;

        }
        else if(visualizacion.equals("Estrella")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
```

```
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("Elemento")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else{
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }
    }//GEN-LAST:event_visualizacionBuscaActionPerformed

    private void botonAceptarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonAceptarActionPerformed
        //Verificamos que todas las opciones sean vÃ¡lidas
        if(opcion.equalsIgnoreCase("Ordenamiento")){ //Algortimos de
ordenamiento
            if(visualizacion == null){
                String mensaje = "Tienes que elegir un algortimo"+
                                 " con una visualizacion disponible";
                JOptionPane.showMessageDialog(null, mensaje);
            }
            else{
                p.setAlgortimo(algoritmo);
                p.setVisualizacion(visualizacion);
                setVisible(false);
                Ordenamiento ordena = new Ordenamiento(this,true,p);
                ordena.setVisible(true);
            }
        }
        else{ //Algortimos de bÃºsqueda
            if(visualizacion == null){
                String mensaje = "Tienes que elegir un algortimo"+
                                 " con una visualizacion disponible";
                JOptionPane.showMessageDialog(null, mensaje);
            }
            else{
                p.setAlgortimo(algoritmo);
                p.setVisualizacion(visualizacion);
                setVisible(false);
                Busqueda busca = new Busqueda(this,true,p);
                busca.setVisible(true);
            }
        }
    }//GEN-LAST:event_botonAceptarActionPerformed

    //Elegir algortimo de bÃºsqueda
    private void comboBusquedaActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_comboBusquedaActionPerformed
        String evento = evt.getActionCommand();
        visualizacion = (String)visualizacionBusca.getSelectedItem();
```

```
        algoritmo = (String)comboBusqueda.getSelectedItem();
        p.setNoElementos(16);
        //Se carga la imagen segÃºn el algoritmo y visualizaciÃ³n elegida
        /*if(algoritmo.equals("Fibonacci")){
            if(!updateLabel("ninguna"))
                visualizacion = null;
        }
        else*/ if(visualizacion.equals("Letra") ){
          if(algoritmo.equals("InterpolaciÃ³n")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("CÃrculo")){
          if(algoritmo.equals("Secuencial") ||
algoritmo.equals("Fibonacci")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("Estrella")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else if(visualizacion.equals("Elemento")){
          if(algoritmo.equals("Binaria")){
              if(!updateLabel(visualizacion))
                    visualizacion = null;
          }
          else if(!updateLabel("ninguna"))
             visualizacion = null;
        }
        else{
            if(!updateLabel("ninguna"))
             visualizacion = null;
         }
    }//GEN-LAST:event_comboBusquedaActionPerformed


    //Actualiza la imagen
    protected boolean updateLabel(String name) {
        name = name+".gif";
        javax.swing.ImageIcon icon =
createImageIcon("../imagenes/"+name/* + ".gif"*/);
        imagenLabel.setIcon(icon);
        imagenLabel.setToolTipText("A drawing of a " +
name.toLowerCase());
        if (icon != null) {
            imagenLabel.setText(null);
```

```
            return true;
        } else {
            imagenLabel.setText("VisualizaciÃ³n no disponible para
algortimo elegido");
            return false;
        }
    }

     /** Regresa un ImageIcon, o null si la ruta no es vÃ¡lida. */
    protected static javax.swing.ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = Main.class.getResource(path);
        if (imgURL != null) {
            return new javax.swing.ImageIcon(imgURL);
        } else {
            //System.err.println("Couldn't find file: " + path);
            return null;
        }
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton botonAceptar;
    private javax.swing.JButton botonSalir;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JComboBox comboBusqueda;
    private javax.swing.JComboBox comboOrdenamiento;
    private javax.swing.JLabel imagenLabel;
    private javax.swing.JRadioButton radioBusqueda;
    private javax.swing.JRadioButton radioOrdenmamiento;
    private javax.swing.JLabel tipoVisualizacion1;
    private javax.swing.JLabel tipoVisualizacion2;
    private javax.swing.JLabel tituloSelecciona;
    private javax.swing.JComboBox visualizacionBusca;
    private javax.swing.JComboBox visualizacionOrdena;
    // End of variables declaration//GEN-END:variables

}
```

Archivo Ordenamiento.java

```
/*
 * Pantalla para asignar las propiedades a los algortimos
 * de ordenamiento
 */

/*
```

```
 * Ordenamiento.java
 */

package Pantallas;

import Algoritmos.*;
import ManejoComandos.Comandos;
import javax.swing.JOptionPane;


public class Ordenamiento extends javax.swing.JFrame {
    private javax.swing.JFrame padre;
    private Propiedades p;

    /** Creates new form Ordenamiento */
    public Ordenamiento(javax.swing.JFrame parent, boolean modal,
Propiedades prop) {
        //super(parent, modal);
        p = prop; //propiedades del algortimo elegido
        padre=parent;
        initComponents(p.getNoElementos());//Se dibuja la ventana
        updateLabel(p.getVisualizacion()); //Se carga la imagen en
pantalla
        setLocation(250,180);//posiciÃ³n de la ventana en pantalla
    }

    //MÃ©todo para crear imagen
    protected boolean updateLabel(String name) {
        javax.swing.ImageIcon icon = createImageIcon("../imagenes/"+name
+ ".gif");
        imagenLabel.setIcon(icon);
        imagenLabel.setToolTipText("A drawing of a " +
name.toLowerCase());
        if (icon != null) {
            imagenLabel.setText(null);
            return true;
        } else {
            imagenLabel.setText("VisualizaciÃ³n no disponible para
algortimo elegido");
            return false;
        }
    }

     /** Regresa un ImageIcon, o null si la ruta no es vÃ¡lida. */
    protected static javax.swing.ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = Main.class.getResource(path);
        if (imgURL != null) {
            return new javax.swing.ImageIcon(imgURL);
        } else {
            //System.err.println("Couldn't find file: " + path);
            return null;
        }
    }
    /** 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">//GEN-
BEGIN:initComponents
    private void initComponents(int n) {

        labelElementos = new javax.swing.JLabel();
        spinnerRango = new javax.swing.JSpinner();
        labelColorLinea = new javax.swing.JLabel();
        comboColorLinea = new javax.swing.JComboBox();
        labelRelleno = new javax.swing.JLabel();
        comboColorRelleno = new javax.swing.JComboBox();
        labelNombreArchivo = new javax.swing.JLabel();
        textNombreArchivo = new javax.swing.JTextField();
        botonAceptar = new javax.swing.JButton();
        botonCancelar = new javax.swing.JButton();
        imagenLabel = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        labelElementos.setText("NÃ°mero de elementos");

        spinnerRango.setModel(new javax.swing.SpinnerNumberModel(3, 3, n,
1));

        labelColorLinea.setText("Color de lÃnea");

        comboColorLinea.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "Negro", "Amarillo", "Anaranjado", "Azul", "Blanco", "CafÃ©",
"Gris", "Rojo", "Rosa", "Verde" }));
        comboColorLinea.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                comboColorLineaActionPerformed(evt);
            }
        });

        labelRelleno.setText("Color de relleno");

        comboColorRelleno.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Negro", "Amarillo",
"Anaranjado", "Azul", "Blanco", "CafÃ©", "Gris", "Rojo", "Rosa", "Verde"
}));

        labelNombreArchivo.setText("Nombre del Archivo");

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

        botonCancelar.setText("Cancelar");
```

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

        imagenLabel.setText("imagen");

        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.LE
ADING, false)
                    .addComponent(spinnerRango,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(labelElementos,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(labelColorLinea)
                    .addComponent(comboColorLinea, 0, 137,
Short.MAX_VALUE)
                    .addComponent(labelRelleno)
                    .addComponent(comboColorRelleno, 0, 137,
Short.MAX_VALUE)
                    .addComponent(labelNombreArchivo)
                    .addComponent(textNombreArchivo))
                .addGap(38, 38, 38)
                .addComponent(imagenLabel)
                .addContainerGap(285, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(76, 76, 76)
                .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 164,
Short.MAX_VALUE)
                .addComponent(botonCancelar,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(78, 78, 78))
        );

        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {comboColorLinea, comboColorRelleno});
```

```
        layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {botonAceptar, botonCancelar});

        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(labelElementos)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(spinnerRango,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(labelColorLinea)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(imagenLabel)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(comboColorLinea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(labelRelleno)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(comboColorRelleno,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(104, 104, 104)
                        .addComponent(labelNombreArchivo)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(textNombreArchivo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 83,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                            .addComponent(botonCancelar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(botonAceptar,
javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE))))
```

```
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents
    //SÃ se da clic en el botÃ³n Cancelar se regresa a la pantalla
principal
    private void botonCancelarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonCancelarActionPerformed
        // TODO add your handling code here:
        setVisible(false);
        padre.setVisible(true);
    }//GEN-LAST:event_botonCancelarActionPerformed

//MÃ©todo que cambia el nombre del color de espaÃ±ol a inglÃ©s
    public String convierteColor(String color){
         if( color == "Negro" )
            return "black";
         else if( color == "Amarillo" )
            return "yellow";
         else if( color == "Anaranjado" )
            return "orange";
         else if( color == "Azul" )
            return "blue";
         else if( color == "Blanco" )
            return "white";
         else if( color == "CafÃ©" )
            return "brown";
         else if( color == "Gris" )
            return "gray";
         else if( color == "Rojo" )
            return "red";
         else if( color == "Rosa" )
            return "pink";
         else
            return "green";
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void GnomeLineas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        GnomeL gnome = new GnomeL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        gnome.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
      gnome.ejecutaGnome(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void ShellLineas(){
```

```
        String nomArchivo = textNombreArchivo.getText().toString();
        ShellL shell = new ShellL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        shell.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
      shell.ejecutaShell(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void intercambioLineas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        IntercambioL intercambio = new IntercambioL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        intercambio.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
      intercambio.ejecutaIntercambio(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void insercionLineas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        InsercionL insercion = new InsercionL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        insercion.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
      insercion.ejecutaInsercion(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void seleccionLineas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        SeleccionL seleccion = new SeleccionL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        seleccion.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
```

```
      seleccion.ejecutaSeleccion(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void quickLineas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        QuickL q = new QuickL();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        q.creaVector(n);
        String color = (String)comboColorLinea.getSelectedItem();
        color = convierteColor(color);
      q.ejecutaQuick(nomArchivo+".tex", color, n);
      Comandos e = new Comandos();
      e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void burbujaFiguras(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String color = (String)comboColorRelleno.getSelectedItem();
        color = convierteColor(color);
        BurbujaF burbuja= new BurbujaF();
      burbuja.creaVector(n,color);
      burbuja.intercambioDirecto(nomArchivo+".tex");
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void burbujaPiramide(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String color = (String)comboColorRelleno.getSelectedItem();
        color = convierteColor(color);
        BurbujaP burbuja= new BurbujaP();
      burbuja.creaVector(n,color);
      burbuja.intercambioDirecto(nomArchivo+".tex");
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
     public void insercionFiguras(){
        String nomArchivo = textNombreArchivo.getText().toString();
```

```
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    InsercionF insercion= new InsercionF();
  insercion.creaVector(n,color);
  insercion.insercionDirecta(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

 //Se crea el objeto segÃºn el algortimo seleccionado
public void insercionPiramide(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    InsercionP insercion= new InsercionP();
  insercion.creaVector(n,color);
  insercion.insercionDirecta(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void seleccionFiguras(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    SeleccionF seleccion= new SeleccionF();
  seleccion.creaVector(n,color);
  seleccion.seleccionDirecta(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void gnomeFiguras(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    GnomeF gnome= new GnomeF();
  gnome.creaVector(n,color);
  gnome.algoritmoGnome(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void shellFiguras(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
```

```
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    ShellF shell= new ShellF();
  shell.creaVector(n,color);
  shell.algoritmoShell(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void seleccionPiramide(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    SeleccionP seleccion= new SeleccionP();
  seleccion.creaVector(n,color);
  seleccion.seleccionDirecta(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void gnomePiramide(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    GnomeP gnome= new GnomeP();
  gnome.creaVector(n,color);
  gnome.algoritmoGnome(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void shellPiramide(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String color = (String)comboColorRelleno.getSelectedItem();
    color = convierteColor(color);
    ShellP gnome= new ShellP();
  gnome.creaVector(n,color);
  gnome.algoritmoShell(nomArchivo+".tex");
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃºn el algortimo seleccionado
public void shellEstrellas(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String relleno = (String)comboColorRelleno.getSelectedItem();
```

```
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        ShellE shell= new ShellE();
      shell.creaVector(n);
      shell.shell(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void gnomeEstrellas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        GnomeE gnome= new GnomeE();
      gnome.creaVector(n);
      gnome.gnome(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void burbujaEstrellas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        BurbujaE burbuja= new BurbujaE();
      burbuja.creaVector(n);
      burbuja.intercambioDirecto(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void insercionEstrellas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
```

```
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        InsercionE insercion= new InsercionE();
      insercion.creaVector(n);
      insercion.insercionDirecta(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void seleccionEstrellas(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        SeleccionE seleccion= new SeleccionE();
      seleccion.creaVector(n);
      seleccion.seleccionDirecta(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void shellElementos(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        ShellEl shell= new ShellEl();
      shell.creaVector(n);
      shell.shell(nomArchivo+".tex",linea,relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void gnomeElementos(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
```

```
    GnomeEl gnome= new GnomeEl();
  gnome.creaVector(n);
  gnome.gnome(nomArchivo+".tex",linea,relleno);
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃ°n el algortimo seleccionado
public void burbujaElementos(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String relleno = (String)comboColorRelleno.getSelectedItem();
    String linea = (String)comboColorLinea.getSelectedItem();
    relleno = convierteColor(relleno);
    linea = convierteColor(linea);
    BurbujaEl burbuja= new BurbujaEl();
  burbuja.creaVector(n);
  burbuja.intercambioDirecto(nomArchivo+".tex",linea,relleno);
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃ°n el algortimo seleccionado
public void insercionElementos(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String relleno = (String)comboColorRelleno.getSelectedItem();
    String linea = (String)comboColorLinea.getSelectedItem();
    relleno = convierteColor(relleno);
    linea = convierteColor(linea);
    InsercionEl insercion= new InsercionEl();
  insercion.creaVector(n);
  insercion.insercion(nomArchivo+".tex",linea,relleno);
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃ°n el algortimo seleccionado
public void seleccionElementos(){
    String nomArchivo = textNombreArchivo.getText().toString();
    int n = Integer.parseInt(spinnerRango.getValue().toString());
    String relleno = (String)comboColorRelleno.getSelectedItem();
    String linea = (String)comboColorLinea.getSelectedItem();
    relleno = convierteColor(relleno);
    linea = convierteColor(linea);
    SeleccionEl seleccion= new SeleccionEl();
  seleccion.creaVector(n);
  seleccion.seleccion(nomArchivo+".tex",linea,relleno);
    Comandos e = new Comandos();
    e.ejecutaComando("pdflatex "+nomArchivo+".tex");
    e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
}

//Se crea el objeto segÃ°n el algortimo seleccionado
```

```
    public void mergeLetras(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        Mergesort merge= new Mergesort();
      merge.creaVector(n,relleno);
      merge.crearArchivo(nomArchivo+".tex");
        Comandos e = new Comandos();
        e.ejecutaComando("pdflatex "+nomArchivo+".tex");
        e.ejecutaComando("rm "+nomArchivo+".log "+nomArchivo+".aux");
    }

    //Se crea el objeto segÃºn el algortimo seleccionado
    public void radixLetras(){
        String nomArchivo = textNombreArchivo.getText().toString();
        int n = Integer.parseInt(spinnerRango.getValue().toString());
        String relleno = (String)comboColorRelleno.getSelectedItem();
        String linea = (String)comboColorLinea.getSelectedItem();
        relleno = convierteColor(relleno);
        linea = convierteColor(linea);
        Radix radix= new Radix();
      radix.creaVector(n);
      radix.radix(nomArchivo+".tex",relleno);
        Comandos e = new Comandos();
        e.ejecutaComando("latex "+nomArchivo+".tex");
      e.ejecutaComando("dvips -Ppdf  -t landscape "+nomArchivo+".dvi");
      e.ejecutaComando("ps2pdf "+nomArchivo+".ps");
      e.ejecutaComando("rm "+nomArchivo+".dvi "+nomArchivo+".log
"+nomArchivo+".ps "+nomArchivo+".aux");
    }

    //SÃ se selecciona el botÃ³n aceptar se hacen validaciones
    private void botonAceptarActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_botonAceptarActionPerformed
        String texto = textNombreArchivo.getText();
        if ("".equals(texto)){ //SÃ la cadena esta vacÃa de muestra un
mensaje
            String mensaje = "Introduce el nombre del Archivo";
                JOptionPane.showMessageDialog(null, mensaje);
        }
        else{   //SegÃºn el algoritmo y visualizaciÃ³n se llama al
mÃ©todo caorrespondiente
            if( p.getAlgortimo() == "Shell sort" ){
                if( p.getVisualizacion() == "LÃneas" )
                    ShellLineas();
                else if( p.getVisualizacion() == "Figuras" )
                    shellFiguras();
                else if( p.getVisualizacion() == "PirÃ¡mide" )
                    shellPiramide();
                else if( p.getVisualizacion() == "Estrellas" )
                    shellEstrellas();
                else if( p.getVisualizacion() == "Elementos" )
                    shellElementos();
            }
```

```
else if( p.getAlgortimo() == "Ordenamiento del duende" ){
    if( p.getVisualizacion() == "LÃneas" )
        GnomeLineas();
    else if( p.getVisualizacion() == "Figuras" )
        gnomeFiguras();
    else if( p.getVisualizacion() == "PirÃ¡mide" )
        gnomePiramide();
    else if( p.getVisualizacion() == "Estrellas" )
        gnomeEstrellas();
    else if( p.getVisualizacion() == "Elementos" )
        gnomeElementos();
}
else if( p.getAlgortimo() == "Intercambio directo" ){
    if( p.getVisualizacion() == "LÃneas" )
        intercambioLineas();
    else if( p.getVisualizacion() == "Figuras" )
        burbujaFiguras();
    else if( p.getVisualizacion() == "PirÃ¡mide" )
        burbujaPiramide();
    else if( p.getVisualizacion() == "Estrellas" )
        burbujaEstrellas();
    else if( p.getVisualizacion() == "Elementos" )
        burbujaElementos();
}
else if( p.getAlgortimo() == "InserciÃ³n directa" ){
    if( p.getVisualizacion() == "LÃneas" )
        insercionLineas();
    else if( p.getVisualizacion() == "Figuras" )
        insercionFiguras();
    else if( p.getVisualizacion() == "PirÃ¡mide" )
        insercionPiramide();
    else if( p.getVisualizacion() == "Estrellas" )
        insercionEstrellas();
    else if( p.getVisualizacion() == "Elementos" )
        insercionElementos();
}
else if( p.getAlgortimo() == "SelecciÃ³n directa" ){
    if( p.getVisualizacion() == "LÃneas" )
        insercionLineas();
    else if( p.getVisualizacion() == "Figuras" )
        seleccionFiguras();
    else if( p.getVisualizacion() == "PirÃ¡mide" )
        seleccionPiramide();
    else if( p.getVisualizacion() == "Estrellas" )
        seleccionEstrellas();
    else if( p.getVisualizacion() == "Elementos" )
        seleccionElementos();
}
else if( p.getAlgortimo() == "Quicksort" ){
    if( p.getVisualizacion() == "LÃneas" )
        quickLineas();
}
else if( p.getAlgortimo() == "Radix sort" ){
    if( p.getVisualizacion() == "Letras" )
        radixLetras();
}
else if( p.getAlgortimo() == "Mergesort" ){
```

```
                if( p.getVisualizacion() == "Letras" )
                    mergeLetras();
            }
            setVisible(false); //SÃ todo saliÃ³ bien se cierra la ventana
y se muestra
            padre.setVisible(true); // la ventana principal
        }
    }//GEN-LAST:event_botonAceptarActionPerformed

    private void
comboColorLineaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_comboColorLineaActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_comboColorLineaActionPerformed

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                //new Ordenamiento(new javax.swing.JFrame(),
true).setVisible(true);
                Ordenamiento dialog = new Ordenamiento(new
javax.swing.JFrame(), true, new Propiedades());
                dialog.addWindowListener(new
java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent
e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton botonAceptar;
    private javax.swing.JButton botonCancelar;
    private javax.swing.JComboBox comboColorLinea;
    private javax.swing.JComboBox comboColorRelleno;
    private javax.swing.JLabel imagenLabel;
    private javax.swing.JLabel labelColorLinea;
    private javax.swing.JLabel labelElementos;
    private javax.swing.JLabel labelNombreArchivo;
    private javax.swing.JLabel labelRelleno;
    private javax.swing.JSpinner spinnerRango;
    private javax.swing.JTextField textNombreArchivo;
    // End of variables declaration//GEN-END:variables

}
```

Archivo Propiedades.java

```
/*
 * Clase auxiliar para guardar las propiedades de los algoritmos
 */

package Pantallas;


public class Propiedades {
    private String visualizacion;
    private String algortimo;
    private int noElementos;

    public int getNoElementos() {
        return noElementos;
    }

    public void setNoElementos(int noElementos) {
        this.noElementos = noElementos;
    }

    public String getAlgortimo() {
        return algortimo;
    }

    public void setAlgortimo(String algortimo) {
        this.algortimo = algortimo;
    }

    public String getVisualizacion() {
        return visualizacion;
    }

    public void setVisualizacion(String visualizacion) {
        this.visualizacion = visualizacion;
    }

}
```

Paquete Algoritmos

BinariaE.java

```
/*
 * Clase para el algortimo de búsqueda binaria y la visualización de
 * estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo6;
```

```
public class BinariaE {
   Nodo6 vector[]; //Se crea un arreglo de Nodos6

        public void binaria(String nombreArchivo,int dato)
        {
                String codigo,encabezado,pie;
           //se crea el archivo
              CrearArchivo a =new CrearArchivo();
              a.abrirArchivo(nombreArchivo);
                 double y=0.0;
           //encabezado para el archivo .tex

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                   String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);
 for(int j=0;j<vector.length;j++){
        vector[j].setY(y);
        codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
        a.escribe(codigo);
}
y -= 2.0;
        int mitad,izq,der;
        izq=0;
        der=vector.length-1;
while(izq<=der){//el elemento que esta ala izquierda debe de ser menor que el de la derecha
   mitad=(izq+der)/2;//calculamos el elemento central del arreglo hasta encontrar el valor
buscado
   if(dato>vector[mitad].getPicos()){//si el elemento buscado es mayor que el centro
entonces buscamos en la segunda mitad
         for(int j=0;j<=mitad;j++){
                vector[j].setLinea("gray");
                vector[j].setRelleno("gray");
        }
```

```
          izq=mitad+1;//el primer valor del arreglo pasa hacer la mitad mas 1
          for(int j=0;j<vector.length;j++){
                    vector[j].setY(y);
                    codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
                              a.escribe(codigo);
          }
          y -= 2.0;
    }
    else if(dato<vector[mitad].getPicos()){//si el elemento buscado buscado es menor que la
mitad entonces buscamos en la primera
             for(int j=mitad;j<=der;j++){
                    vector[j].setLinea("gray");
                    vector[j].setRelleno("gray");
          }
             der=mitad-1;
          for(int j=0;j<vector.length;j++){
                    vector[j].setY(y);
                    codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
                              a.escribe(codigo);
          }
          y -= 2.0;
    }
    else
          break;
}

          a.escribe(pie);
       a.cerrarArchivo();
          }//fin del metodo


       //Método que crea el arreglo de tipo Nodo
 public void creaVector( int num,int dato,String linea,String relleno ){
          int i;
       double pos=0.0;

          vector=new Nodo6[num];
          for( i = 0; i < num; i++, pos += 1.5 ){
                    if((i+3)==dato)
                              vector[i]=new Nodo6(pos,0.0,i+3,"red","red");
                    else
```

```
                    vector[i]=new Nodo6(pos,0.0,i+3,linea,relleno);
        }
 }
}
```

BinariaEl.java

```
/*
 * Clase para el algortimo de búsqueda binaria y la visualización de
 * estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo6;


public class BinariaE {
   Nodo6 vector[]; //Se crea un arreglo de Nodos6

        public void binaria(String nombreArchivo,int dato)
        {
                String codigo,encabezado,pie;
           //se crea el archivo
              CrearArchivo a =new CrearArchivo();
              a.abrirArchivo(nombreArchivo);
                 double y=0.0;
           //encabezado para el archivo .tex

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);
 for(int j=0;j<vector.length;j++){
        vector[j].setY(y);
        codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
```

```
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
        a.escribe(codigo);
}
y -= 2.0;
        int mitad,izq,der;
        izq=0;
        der=vector.length-1;
while(izq<=der){//el elemento que esta ala izquierda debe de ser menor que el de la derecha
   mitad=(izq+der)/2;//calculamos el elemento central del arreglo hasta encontrar el valor
buscado
   if(dato>vector[mitad].getPicos()){//si el elemento buscado es mayor que el centro
entonces buscamos en la segunda mitad
         for(int j=0;j<=mitad;j++){
                vector[j].setLinea("gray");
                vector[j].setRelleno("gray");
        }
         izq=mitad+1;//el primer valor del arreglo pasa hacer la mitad mas 1
        for(int j=0;j<vector.length;j++){
                vector[j].setY(y);
                codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
                        a.escribe(codigo);
        }
        y -= 2.0;
   }
   else if(dato<vector[mitad].getPicos()){//si el elemento buscado buscado es menor que la
mitad entonces buscamos en la primera
          for(int j=mitad;j<=der;j++){
                vector[j].setLinea("gray");
                vector[j].setRelleno("gray");
        }
          der=mitad-1;
        for(int j=0;j<vector.length;j++){
                vector[j].setY(y);
                codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),vector[j].getLinea(),vector[j].g
etRelleno());
                        a.escribe(codigo);
        }
        y -= 2.0;
   }
   else
        break;
```

```
}

        a.escribe(pie);
      a.cerrarArchivo();
        }//fin del metodo


      //Método que crea el arreglo de tipo Nodo
 public void creaVector( int num,int dato,String linea,String relleno ){
        int i;
      double pos=0.0;

        vector=new Nodo6[num];
        for( i = 0; i < num; i++, pos += 1.5 ){
                if((i+3)==dato)
                        vector[i]=new Nodo6(pos,0.0,i+3,"red","red");
                else
                        vector[i]=new Nodo6(pos,0.0,i+3,linea,relleno);
        }
 }
}
```

BurbujaE.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo3;
import java.util.Random;

public class BurbujaE {
   Nodo3 vector[];

        public void intercambioDirecto(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
              CrearArchivo a =new CrearArchivo();
              a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
```

```
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                    String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

          pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);

       int t = vector.length;
       for (int i = 1; i < t; i++) {
          for ( int k = t- 1; k >= i; k--) {
             if(vector[k].getPicos() < vector[k-1].getPicos()){
                contador++;
                int aux = vector[k].getPicos();
                vector[k].setPicos(vector[k-1].getPicos());
                vector[k-1].setPicos(aux);
                      for(int j=0;j<t;j++){
                            vector[j].setY(y);
                            codigo=String.format("%cnode at (%.2f,%.2f) [star,star
points=%d,star point ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),colorL,colorR);
                            a.escribe(codigo);
                      }
                y=y-2;
                if (contador==8){
                            System.out.println(y);
                   a.escribe(newpage);
                   contador=0;
                            y=0.0;
                     }
            }//fin if
          }// fin 2 for
       }//fin 1 for


          a.escribe(pie);
       a.cerrarArchivo();
          }//fin del metodo

public void creaVector( int num ){
          int vectors[] = new int[num];
          int d, j, i, aux;
       double pos=0.0;
```

```
        Random aleatorio=new Random();
        for( i = 0, j = 3; i < num; i++, j++ )
           vectors[i]=j;

        for( i = 0, j = 1; i < num; i++, j++ ){
           d = aleatorio.nextInt(num-1);
           aux=vectors[d];
              vectors[d]=vectors[num-j];
           vectors[num-j]=aux;
        }

           vector=new Nodo3[num];
           for( i = 0; i < num; i++, pos += 1.5 )
                    vector[i]=new Nodo3(pos,4.0,vectors[i]);
}
}
```

BurbujaEl.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo4;
import Nodos.NodoElementos;
import java.util.Random;

public class BurbujaEl {
   Nodo4 vector[];
   NodoElementos elementos[];

        public void intercambioDirecto(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
             CrearArchivo a =new CrearArchivo();
             a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
```

```
            a.escribe(encabezado);

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,92,92);

          pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,92,92);

       int t = vector.length;
       for (int i = 1; i < t; i++) {
          for ( int k = t- 1; k >= i; k--) {
             if(vector[k].getAtomico() < vector[k-1].getAtomico()){
                contador++;
                Nodo4 aux = new Nodo4();
                        aux = vector[k];
                   double auxX = vector[k].getX();
                vector[k]=vector[k-1];
                vector[k-1]=aux;
                   System.out.print("Aux-->");
                   aux.imprime();
                   System.out.print("k-->");
                   vector[k].imprime();
                   System.out.print("k-1-->");
                   vector[k-1].imprime();


                      vector[k-1].setX(vector[k].getX());
                   System.out.print("k-1-->");
                   vector[k-1].imprime();
                      vector[k].setX(auxX);
                            System.out.print("k-->");
                   vector[k].imprime();
                      for(int j=0;j<t;j++){
                            vector[j].setY(y);
                            codigo=String.format("%cnode at (%.2f,%.2f) [rectangle,minimum
height=1cm,minimum width=1cm,draw=%s,fill=%s]
{%s};",92,vector[j].getX(),vector[j].getY(),colorL,colorR,vector[j].getSimbolo());
                            a.escribe(codigo);
                         }
                y=y-2;
                if (contador==8){
                            System.out.println(y);
                   a.escribe(newpage);
                    contador=0;
                            y=0.0;
                     }
```

```
            }//fin if
          }// fin 2 for
        }//fin 1 for


           a.escribe(pie);
        a.cerrarArchivo();
           }//fin del metodo

public void creaElementos(){
 elementos=new NodoElementos[118];
elementos[0]=new NodoElementos("H",1);
elementos[1]=new NodoElementos("He",2);
elementos[2]=new NodoElementos("Li",3);
elementos[3]=new NodoElementos("Be",4);
elementos[4]=new NodoElementos("B",5);
elementos[5]=new NodoElementos("C",6);
elementos[6]=new NodoElementos("N",7);
elementos[7]=new NodoElementos("O",8);
elementos[8]=new NodoElementos("F",9);
elementos[9]=new NodoElementos("Ne",10);
elementos[10]=new NodoElementos("Na",11);
elementos[11]=new NodoElementos("Mg",12);
elementos[12]=new NodoElementos("Al",13);
elementos[13]=new NodoElementos("Si",14);
elementos[14]=new NodoElementos("P",15);
elementos[15]=new NodoElementos("S",16);
elementos[16]=new NodoElementos("Cl",17);
elementos[17]=new NodoElementos("Ar",18);
elementos[18]=new NodoElementos("K",19);
elementos[19]=new NodoElementos("Ca",20);
elementos[20]=new NodoElementos("Sc",21);
elementos[21]=new NodoElementos("Ti",22);
elementos[22]=new NodoElementos("V",23);
elementos[23]=new NodoElementos("Cr",24);
elementos[24]=new NodoElementos("Mn",25);
elementos[25]=new NodoElementos("Fe",26);
elementos[26]=new NodoElementos("Co",27);
elementos[27]=new NodoElementos("Ni",28);
elementos[28]=new NodoElementos("Cu",29);
elementos[29]=new NodoElementos("Zn",30);
elementos[30]=new NodoElementos("Ga",31);
elementos[31]=new NodoElementos("Ge",32);
elementos[32]=new NodoElementos("As",33);
elementos[33]=new NodoElementos("Se",34);
elementos[34]=new NodoElementos("Br",35);
elementos[35]=new NodoElementos("Kr",36);
```

```
elementos[36]=new NodoElementos("Rb",37);
elementos[37]=new NodoElementos("Sr",38);
elementos[38]=new NodoElementos("Y",39);
elementos[39]=new NodoElementos("Zr",40);
elementos[40]=new NodoElementos("Nb",41);
elementos[41]=new NodoElementos("Mo",42);
elementos[42]=new NodoElementos("Tc",43);
elementos[43]=new NodoElementos("Ru",44);
elementos[44]=new NodoElementos("Rh",45);
elementos[45]=new NodoElementos("Pd",46);
elementos[46]=new NodoElementos("Ag",47);
elementos[47]=new NodoElementos("Cd",48);
elementos[48]=new NodoElementos("In",49);
elementos[49]=new NodoElementos("Sn",50);
elementos[50]=new NodoElementos("Sb",51);
elementos[51]=new NodoElementos("Te",52);
elementos[52]=new NodoElementos("I",53);
elementos[53]=new NodoElementos("Xe",54);
elementos[54]=new NodoElementos("Cs",55);
elementos[55]=new NodoElementos("Ba",56);
elementos[56]=new NodoElementos("La",57);
elementos[57]=new NodoElementos("Ce",58);
elementos[58]=new NodoElementos("Pr",59);
elementos[59]=new NodoElementos("Nd",60);
elementos[60]=new NodoElementos("Pm",61);
elementos[61]=new NodoElementos("Sm",62);
elementos[62]=new NodoElementos("Eu",63);
elementos[63]=new NodoElementos("Gd",64);
elementos[64]=new NodoElementos("Tb",65);
elementos[65]=new NodoElementos("Dy",66);
elementos[66]=new NodoElementos("Ho",67);
elementos[67]=new NodoElementos("Er",68);
elementos[68]=new NodoElementos("Tm",69);
elementos[69]=new NodoElementos("Yb",70);
elementos[70]=new NodoElementos("Lu",71);
elementos[71]=new NodoElementos("Hf",72);
elementos[72]=new NodoElementos("Ta",73);
elementos[73]=new NodoElementos("W",74);
elementos[74]=new NodoElementos("Re",75);
elementos[75]=new NodoElementos("Os",76);
elementos[76]=new NodoElementos("Ir",77);
elementos[77]=new NodoElementos("Pt",78);
elementos[78]=new NodoElementos("Au",79);
elementos[79]=new NodoElementos("Hg",80);
elementos[80]=new NodoElementos("Tl",81);
elementos[81]=new NodoElementos("Pb",82);
elementos[82]=new NodoElementos("Bi",83);
```

```
elementos[83]=new NodoElementos("Po",84);
elementos[84]=new NodoElementos("At",85);
elementos[85]=new NodoElementos("Rn",86);
elementos[86]=new NodoElementos("Fr",87);
elementos[87]=new NodoElementos("Ra",88);
elementos[88]=new NodoElementos("Ac",89);
elementos[89]=new NodoElementos("Th",90);
elementos[90]=new NodoElementos("Pa",91);
elementos[91]=new NodoElementos("U",92);
elementos[92]=new NodoElementos("Np",93);
elementos[93]=new NodoElementos("Pu",94);
elementos[94]=new NodoElementos("Am",95);
elementos[95]=new NodoElementos("Cm",96);
elementos[96]=new NodoElementos("Bk",97);
elementos[97]=new NodoElementos("Cf",98);
elementos[98]=new NodoElementos("Es",99);
elementos[99]=new NodoElementos("Fm",100);
elementos[100]=new NodoElementos("Md",101);
elementos[101]=new NodoElementos("No",102);
elementos[102]=new NodoElementos("Lr",103);
elementos[103]=new NodoElementos("Rf",104);
elementos[104]=new NodoElementos("Db",105);
elementos[105]=new NodoElementos("Sg",106);
elementos[106]=new NodoElementos("Bh",107);
elementos[107]=new NodoElementos("Hs",108);
elementos[108]=new NodoElementos("Mt",109);
elementos[109]=new NodoElementos("Ds",110);
elementos[110]=new NodoElementos("Uuu",111);
elementos[111]=new NodoElementos("Uub",112);
elementos[112]=new NodoElementos("Uut",113);
elementos[113]=new NodoElementos("Uuq",114);
elementos[114]=new NodoElementos("Uup",115);
elementos[115]=new NodoElementos("Uuh",116);
elementos[116]=new NodoElementos("Uus",117);
elementos[117]=new NodoElementos("Uuo",118);

}



public void creaVector( int num ){

        int d, j, i;
      double pos=0.0;
        creaElementos();
      Random aleatorio=new Random();
//      for( i = 0, j = 3; i < num; i++, j++ )
```

```
  //       vectors[i]=j;

           vector=new Nodo4[num];
        for( i = 0; i < num; i++, pos += 1.0){
           d = aleatorio.nextInt(118);
              vector[i]=new
Nodo4(pos,0.0,elementos[d].getAtomico(),elementos[d].getSimbolo());
           }
}
}
```

BurbujaF.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;

//import archivos.CreaArchivo;


public class BurbujaF {
   Nodo vector[];

          public void intercambioDirecto(String nombreArchivo)
          {
                  String codigo,encabezado,pie;
                CrearArchivo a =new CrearArchivo();
                a.abrirArchivo(nombreArchivo);
                  int contador=0;
                  double y=0.0;

          encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{center}%cbegin{tikzpicture}\n",92,92,92,92,92);
                a.escribe(encabezado);

                  String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
```

```
        pie=String.format("%cend{tikzpicture}\n%cend{center}%cend{document}",92,92,
92);

      int t = vector.length;
      for (int i = 1; i < t; i++) {
         for ( int k = t- 1; k >= i; k--) {
            if(vector[k].getRadio() < vector[k-1].getRadio()){
               contador++;
               double aux = vector[k].getRadio();
               vector[k].setRadio(vector[k-1].getRadio());
               vector[k-1].setRadio(aux);
                                 for(k=0;k<vector.length;k++){
                                          vector[k].setY(y);
                                          codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getRa
dio());
                                                a.escribe(codigo);
                                                      }
                                    y=y-2;
                                    if (contador==9){
                                          a.escribe(newpage);
                                           contador=0;   }
                                     } //fin if
                                     }// fin 2 for
                                }//fin 1 for


            a.escribe(pie);
         a.cerrarArchivo();
                  }

         public void creaVector(int num,String color){
         int vectors[] = new int[num];
         int d,j=1,i,aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for(i=0;i<num;i++)
         vectors[i]=i+1;

      for(i=0;i<num;i++,j++){
         d = aleatorio.nextInt(num-1);
         aux=vectors[d];
                     vectors[d]=vectors[num-j];
         vectors[num-j]=aux;
      }
```

```
        vector=new Nodo[num];
        for(i=0;i<num;i++,pos+=2.0)
                vector[i]=new Nodo(pos,0.0,color,vectors[i]*0.1);
        }

}
```

BurbujaP.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo2;
import java.util.Random;

public class BurbujaP{
   Nodo2 vector[];

        public void intercambioDirecto(String nombreArchivo)
        {
                String codigo,encabezado,pie,newpage,rec;
                //String nombreArchivo="ejemplo1.tex";
             CrearArchivo a =new CrearArchivo();
             a.abrirArchivo(nombreArchivo);
           double y=0.0,x;
                int contador=0;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}[rounded corners,ultra thick]\n",92,92,92,92);
                a.escribe(encabezado);

        newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}[
rounded corners,ultra thick]",92,92,92);

                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                int t = vector.length;

      for (int i = 1; i < t; i++) {
         for ( int k = t- 1; k >= i; k--) {
            if(vector[k].getLargo() < vector[k-1].getLargo()){
```

```
                contador++;
                      double aux = vector[k].getLargo();
                vector[k].setLargo(vector[k-1].getLargo());
                vector[k-1].setLargo(aux);
                                for(k=0;k<vector.length;k++){
                                        vector[k].setY(y);
                                        x=(vector.length-vector[k].getLargo())/2;
                                        vector[k].setX(x);
                                        codigo=String.format("%cshade[top color=%s]
(%.2f,%.2f) rectangle
+(%.1f,0.5);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getLargo(
));
                                        a.escribe(codigo);
                                    y=y-0.5;
                                                        }
                                y=y-0.5;
                                    rec=String.format("%cshade[top color=white] (0.0,%.2f)
rectangle +(10,0.5);",92,y);
                                        a.escribe(rec);
                                        y=y-0.5;

                                         if (contador==2){
                                    a.escribe(newpage);
                                    contador=0;
                                            y = 0.0;
                                         }
                                } //fin if
                                 }// fin 2 for
                               }//fin 1 for

            a.escribe(pie);
          a.cerrarArchivo();
                  }

          public void creaVector(int num,String color){
                    int vectors[] = new int[num];
                    int d,j=1,i,aux;
                int pos=0;
                Random aleatorio=new Random();
                for(i=0;i<num;i++)
                    vectors[i]=i+1;

          for(i=0;i<num;i++,j++){
                              d = aleatorio.nextInt(num-1);
                              aux=vectors[d];
                              vectors[d]=vectors[num-j];
                              vectors[num-j]=aux;
```

```
                        }

        vector=new Nodo2[num];
        for(i=0;i<num;i++){
                //pos=(30-pos)/2;
                vector[i]=new Nodo2(pos,0.0,vectors[i],color);
                    }
                 }

}
```

Fibonacci.java

```
package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo8;


public class Fibonacci {
   Nodo8 vector[];
   Nodo8 elemento;
        public int fibonacci(String nombreArchivo)
        {
                String codigo,encabezado,pie,newpage;
                String c;
        CrearArchivo a =new CrearArchivo();
        a.abrirArchivo(nombreArchivo);
           double y=0.0;
                int  b=0;

encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbegin{do
cument}\n%cbegin{tikzpicture}\n",92,92,92,92);
                a.escribe(encabezado);
                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);

newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}",92,92,
92);

                codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,elemento.getColor(),elemento.getX(),elemento.getY(),elemento.getRa
dio()*0.1);
                a.escribe(codigo);
                elemento.setY(elemento.getY()-4.0);
```

```
        for(int j=0;j<vector.length;j++){
                codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[j].getColor(),vector[j].getX(),vector[j].getY(),vector[j].getRadio
()*0.1);
              a.escribe(codigo);
              vector[j].setY(vector[j].getY()-4.0);
        }


                int first,index;

        int f1=1, f2=0, mid=2;                  //initialise the first two fibonacci
           int n = vector.length;        //numbers. F1 will be the main
      while (f1<n) {                     //set F1 to a number >= n
           f1=f1+f2;
           f2=f1-f2;
           mid++;
      }
      f2=f1-f2;                          //set F1 to the largest number <=n
      f1=f1-f2;
      mid--;
      first=0;

      while (mid>0) {                     //loop
          index=first+f1;
                elemento.setX(index*2.0);
   codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,elemento.getColor(),elemento.getX(),elemento.getY(),elemento.getRa
dio()*0.1);
                a.escribe(codigo);
                elemento.setY(elemento.getY()-4.0);

        for(int j=0;j<vector.length;j++){
                codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[j].getColor(),vector[j].getX(),vector[j].getY(),vector[j].getRadio
()*0.1);
              a.escribe(codigo);
              vector[j].setY(vector[j].getY()-4.0);
        }

       if (index>=n || elemento.getRadio()<vector[index].getRadio()) {
                mid--;                  //if the number is bigger, move back
                f2=f1-f2;               //to a smaller fibonacci number
                f1=f1-f2;
       }
       else if (elemento.getRadio()==vector[index].getRadio()){        //found: return the
index
```

```
        //          System.out.printf("Si se encontro");
                            elemento.setX(vector[index].getX());
                            elemento.setY(vector[index].getY()+4.0);
                            codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,elemento.getColor(),elemento.getX(),elemento.getY(),elemento.getRa
dio()*0.1);
                     a.escribe(codigo);
                            a.escribe(pie);
                            a.cerrarArchivo();
                   return(index);
        }
        else {
                     first=index;            //if the number is smaller, move to the
                     mid=mid-2;               //second part of the list and
                     f1=f1-f2;               //reduce F1 back two F-numbers
                     f2=f2-f1;
        }
      }

      System.out.printf("no se encontro\n");
      return(-1);
         }

         public void creaVector(int num,String color,int r){
            int i;
         double pos=0.0;
            vector=new Nodo8[num];
               for(i=0;i<num;i++,pos+=2.0)
                            vector[i]=new Nodo8(pos,-2.0,color,i+1);
               elemento=new Nodo8(0.0,0.0,"red",r);
       }
}
```

GnomeE.java

```
/*
 * Clase para el algortimo Gnome y la visualización de
 * estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo3;
import java.util.Random;

public class GnomeE {
```

```
   Nodo3 vector[]; //Arreglo de tipo NOdo3

        public void gnome(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);

       int n = vector.length;
        int i=1;
          for(int j=0;j<n;j++){
                vector[j].setY(y);
                codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),colorL,colorR);
                a.escribe(codigo);
          }
          y=y-2;
       //inica la ejecuciń del algortimo
           while(i < n){
          if (i == 0 || vector[i-1].getPicos() <= vector[i].getPicos())
             i++;
          else{
                contador++;
             int aux = vector[i - 1].getPicos();
             vector[i - 1].setPicos(vector[i].getPicos());
             vector[i].setPicos(aux);
             i --;
             for(int j=0;j<n;j++){
                        vector[j].setY(y);
```

```
                        codigo=String.format("%cnode at (%.2f,%.2f) [star,star
points=%d,star point ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[j].getX(),vector[j].getY(),vector[j].getPicos(),colorL,colorR);
                        a.escribe(codigo);
                    }
                    y=y-2;
                if (contador==8){
                    a.escribe(newpage);
                    contador=0;
                        y=0.0;
                    }
            }
        }

            a.escribe(pie);
        a.cerrarArchivo();
            }//fin del metodo

public void creaVector( int num ){
            int vectors[] = new int[num];
            int d, j, i, aux;
        double pos=0.0;
        Random aleatorio=new Random();
        for( i = 0, j = 3; i < num; i++, j++ )
            vectors[i]=j;

        for( i = 0, j = 1; i < num; i++, j++ ){
            d = aleatorio.nextInt(num-1);
            aux=vectors[d];
                vectors[d]=vectors[num-j];
            vectors[num-j]=aux;
        }

            vector=new Nodo3[num];
            for( i = 0; i < num; i++, pos += 1.5 )
                    vector[i]=new Nodo3(pos,4.0,vectors[i]);
}
}
```

GnomeEl.java

```
/*
 * Clase para el algortimo Gnome y la visualización de
 * elementos
 */

package Algoritmos;
```

```
import ManejoArchivos.CrearArchivo;
import Nodos.Nodo4;
import Nodos.NodoElementos;
import java.util.Random;

public class GnomeE1 {
    Nodo4 vector[];
   NodoElementos elementos[];

        public void gnome(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);

      int n = vector.length;
        int i=1;
        while(i < n){
        if (i == 0 || vector[i-1].getAtomico() <= vector[i].getAtomico())
            i++;
        else{
                contador++;
            int atomico = vector[i - 1].getAtomico();
                String simbolo = vector[i-1].getSimbolo();
            vector[i - 1].setAtomico(vector[i].getAtomico());
                vector[i - 1].setSimbolo(vector[i].getSimbolo());
                vector[i].setAtomico(atomico);
                vector[i].setSimbolo(simbolo);
            i --;
            for(int j=0;j<n;j++){
                        vector[j].setY(y);
```

```
                        codigo=String.format("%cnode at (%.2f,%.2f) [rectangle,minimum
height=1cm,minimum width=1cm,draw=%s,fill=%s]
{%s};",92,vector[j].getX(),vector[j].getY(),colorL,colorR,vector[j].getSimbolo());
                        a.escribe(codigo);
                    }
                    y=y-2;
                if (contador==8){
                    a.escribe(newpage);
                    contador=0;
                        y=0.0;
                    }
            }
        }


           a.escribe(pie);
        a.cerrarArchivo();
           }//fin del metodo

public void creaElementos(){
 elementos=new NodoElementos[118];
elementos[0]=new NodoElementos("H",1);
elementos[1]=new NodoElementos("He",2);
elementos[2]=new NodoElementos("Li",3);
elementos[3]=new NodoElementos("Be",4);
elementos[4]=new NodoElementos("B",5);
elementos[5]=new NodoElementos("C",6);
elementos[6]=new NodoElementos("N",7);
elementos[7]=new NodoElementos("O",8);
elementos[8]=new NodoElementos("F",9);
elementos[9]=new NodoElementos("Ne",10);
elementos[10]=new NodoElementos("Na",11);
elementos[11]=new NodoElementos("Mg",12);
elementos[12]=new NodoElementos("Al",13);
elementos[13]=new NodoElementos("Si",14);
elementos[14]=new NodoElementos("P",15);
elementos[15]=new NodoElementos("S",16);
elementos[16]=new NodoElementos("Cl",17);
elementos[17]=new NodoElementos("Ar",18);
elementos[18]=new NodoElementos("K",19);
elementos[19]=new NodoElementos("Ca",20);
elementos[20]=new NodoElementos("Sc",21);
elementos[21]=new NodoElementos("Ti",22);
elementos[22]=new NodoElementos("V",23);
elementos[23]=new NodoElementos("Cr",24);
elementos[24]=new NodoElementos("Mn",25);
elementos[25]=new NodoElementos("Fe",26);
```

```
elementos[26]=new NodoElementos("Co",27);
elementos[27]=new NodoElementos("Ni",28);
elementos[28]=new NodoElementos("Cu",29);
elementos[29]=new NodoElementos("Zn",30);
elementos[30]=new NodoElementos("Ga",31);
elementos[31]=new NodoElementos("Ge",32);
elementos[32]=new NodoElementos("As",33);
elementos[33]=new NodoElementos("Se",34);
elementos[34]=new NodoElementos("Br",35);
elementos[35]=new NodoElementos("Kr",36);
elementos[36]=new NodoElementos("Rb",37);
elementos[37]=new NodoElementos("Sr",38);
elementos[38]=new NodoElementos("Y",39);
elementos[39]=new NodoElementos("Zr",40);
elementos[40]=new NodoElementos("Nb",41);
elementos[41]=new NodoElementos("Mo",42);
elementos[42]=new NodoElementos("Tc",43);
elementos[43]=new NodoElementos("Ru",44);
elementos[44]=new NodoElementos("Rh",45);
elementos[45]=new NodoElementos("Pd",46);
elementos[46]=new NodoElementos("Ag",47);
elementos[47]=new NodoElementos("Cd",48);
elementos[48]=new NodoElementos("In",49);
elementos[49]=new NodoElementos("Sn",50);
elementos[50]=new NodoElementos("Sb",51);
elementos[51]=new NodoElementos("Te",52);
elementos[52]=new NodoElementos("I",53);
elementos[53]=new NodoElementos("Xe",54);
elementos[54]=new NodoElementos("Cs",55);
elementos[55]=new NodoElementos("Ba",56);
elementos[56]=new NodoElementos("La",57);
elementos[57]=new NodoElementos("Ce",58);
elementos[58]=new NodoElementos("Pr",59);
elementos[59]=new NodoElementos("Nd",60);
elementos[60]=new NodoElementos("Pm",61);
elementos[61]=new NodoElementos("Sm",62);
elementos[62]=new NodoElementos("Eu",63);
elementos[63]=new NodoElementos("Gd",64);
elementos[64]=new NodoElementos("Tb",65);
elementos[65]=new NodoElementos("Dy",66);
elementos[66]=new NodoElementos("Ho",67);
elementos[67]=new NodoElementos("Er",68);
elementos[68]=new NodoElementos("Tm",69);
elementos[69]=new NodoElementos("Yb",70);
elementos[70]=new NodoElementos("Lu",71);
elementos[71]=new NodoElementos("Hf",72);
elementos[72]=new NodoElementos("Ta",73);
```

```
elementos[73]=new NodoElementos("W",74);
elementos[74]=new NodoElementos("Re",75);
elementos[75]=new NodoElementos("Os",76);
elementos[76]=new NodoElementos("Ir",77);
elementos[77]=new NodoElementos("Pt",78);
elementos[78]=new NodoElementos("Au",79);
elementos[79]=new NodoElementos("Hg",80);
elementos[80]=new NodoElementos("Tl",81);
elementos[81]=new NodoElementos("Pb",82);
elementos[82]=new NodoElementos("Bi",83);
elementos[83]=new NodoElementos("Po",84);
elementos[84]=new NodoElementos("At",85);
elementos[85]=new NodoElementos("Rn",86);
elementos[86]=new NodoElementos("Fr",87);
elementos[87]=new NodoElementos("Ra",88);
elementos[88]=new NodoElementos("Ac",89);
elementos[89]=new NodoElementos("Th",90);
elementos[90]=new NodoElementos("Pa",91);
elementos[91]=new NodoElementos("U",92);
elementos[92]=new NodoElementos("Np",93);
elementos[93]=new NodoElementos("Pu",94);
elementos[94]=new NodoElementos("Am",95);
elementos[95]=new NodoElementos("Cm",96);
elementos[96]=new NodoElementos("Bk",97);
elementos[97]=new NodoElementos("Cf",98);
elementos[98]=new NodoElementos("Es",99);
elementos[99]=new NodoElementos("Fm",100);
elementos[100]=new NodoElementos("Md",101);
elementos[101]=new NodoElementos("No",102);
elementos[102]=new NodoElementos("Lr",103);
elementos[103]=new NodoElementos("Rf",104);
elementos[104]=new NodoElementos("Db",105);
elementos[105]=new NodoElementos("Sg",106);
elementos[106]=new NodoElementos("Bh",107);
elementos[107]=new NodoElementos("Hs",108);
elementos[108]=new NodoElementos("Mt",109);
elementos[109]=new NodoElementos("Ds",110);
elementos[110]=new NodoElementos("Uuu",111);
elementos[111]=new NodoElementos("Uub",112);
elementos[112]=new NodoElementos("Uut",113);
elementos[113]=new NodoElementos("Uuq",114);
elementos[114]=new NodoElementos("Uup",115);
elementos[115]=new NodoElementos("Uuh",116);
elementos[116]=new NodoElementos("Uus",117);
elementos[117]=new NodoElementos("Uuo",118);

}
```

```
public void creaVector( int num ){
          int d, j, i;
       double pos=0.0;
          creaElementos();
       Random aleatorio=new Random();
//       for( i = 0, j = 3; i < num; i++, j++ )
  //       vectors[i]=j;

          vector=new Nodo4[num];
       for( i = 0; i < num; i++, pos += 1.0){
          d = aleatorio.nextInt(118);
             vector[i]=new
Nodo4(pos,0.0,elementos[d].getAtomico(),elementos[d].getSimbolo());
          }
 }
}
```

GnomeF.java

```
/*
 *Clase para el algortimo Gnome y la visualización de
 * figuras
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;

public class GnomeF {
   Nodo vector[];

          public void algoritmoGnome(String nombreArchivo)
          {
                String codigo,encabezado,pie;
             CrearArchivo a =new CrearArchivo();
             a.abrirArchivo(nombreArchivo);
            int i=1,contador=0;
            double y=0.0,aux= 0;

encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbegin{do
cument}\n%cbegin{tikzpicture}\n",92,92,92,92);
                   a.escribe(encabezado); //Escribe el encabezado en el archivo
```

```
            String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
                    pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
          while(i < vector.length) //Empieza el algoritmo Gnome
      {
          if (i == 0 || vector[i-1].getRadio() <= vector[i].getRadio())//Hace comparaciones
             i++;
          else
          {
             contador++; //Cuenta las iteraciones
             aux = vector[i-1].getRadio(); //Realiza el itercambio de posiciones
             vector[i-1].setRadio(vector[i].getRadio());
             vector[i].setRadio(aux) ;
             i --;
             for(int k=0;k<vector.length;k++){ //Este ciclo escribe el código por cada
iteración que haga el algoritmo
             vector[k].setY(y);
             codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getRa
dio());
                    a.escribe(codigo);}
             y=y-2;//Modifica las coordenadas en y
            if (contador==9){ //Si se cuentan 9 iteraciones en una página
             a.escribe(newpage);//Se requerirá de una nueva página
             contador=0; //El contador se reiniciará
             y=0;     } //Las coordenadas volverán a empezar de cero.
          }
      } //termina el algoritmo

          a.escribe(pie); // Escribe el pie en el archivo
      a.cerrarArchivo();


                    }


          public void creaVector(int num,String color){ //Crea el vector de números aleatorios
          int vectors[] = new int[num];
          int d,j=1,i,aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for(i=0;i<num;i++)
          vectors[i]=i+1;    //asigna valores al vector

      for(i=0;i<num;i++,j++){   //Este ciclo se encarga de desordenar el vector de tal modo
que no se repitan
```

```
        d = aleatorio.nextInt(num-1);
              aux=vectors[d];
              vectors[d]=vectors[num-j];
           vectors[num-j]=aux;
      }


        vector=new Nodo[num];
        for(i=0;i<num;i++,pos+=2.0) //Se guardan las propiedades en el vector tipo Nodo
                vector[i]=new Nodo(pos,0.0,color,vectors[i]*0.1);
        }
}
```

GnomeL.java

```
/*
 * Clase para el algortimo Gnome y la visualización de
 * líneas
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;

public class GnomeL {
        private Nodo arreglo[];
        private int vector[];
        private double pasos;

        public void creaVector(int n)
         {
                int m,k,i;
                arreglo = new Nodo[n];
                m = (90 / n);
                k=m;
                m = 10;
                for(i=0;i<n;i++,m+=k)
                        arreglo[i] = new Nodo(m,"draw [color = ");
         }

        public void mezclaArreglo(int n)
        {
                int i,aux,m;
                vector = new int[n];
```

```
                Random aleatorios = new Random();
                Nodo temp = new Nodo();
                m=n;
                for(i=0;i<n;i++,m--){
                        aux = aleatorios.nextInt(m);
                        temp = arreglo[m-1];
                        arreglo[m-1] = arreglo[aux];
                        vector[m-1] = arreglo[aux].getTrans();
                        arreglo[aux]= temp;
                }
        }

        public int intercambios(int n)
{
   int i = 1, aux= 0,cuenta=0;
   while(i < n)
   {
      if (i == 0 || vector[i-1] <= vector[i])
         i++;
      else
      {
         aux = vector[i - 1];
         vector[i - 1] = vector[i];
         vector[i] = aux ;
         i --;
         cuenta++;
      }
   }
   return cuenta+2;
}

        public void estableceColor(String color,int n)
         {
                for(int i=0;i<n;i++)
                        arreglo[i].agregaColor(color);
         }

        public void iniciaCoordenadas(int n)
         {
                double aux,x,y = 0.0;
                int i;
                aux = (14.5) / (n-1);
                y =aux;
                for(i=1;i<n;i++, y+=aux)
                        arreglo[i].setY(y*-1.0);
                for(i=0;i<n;i++){
                        x = arreglo[i].getX();
```

```
                        y = arreglo[i].getY();
                        String coor = String.format("(%.2f,%.2f) -- ",x,y);
                        arreglo[i].agregarCodigo(coor);
                        arreglo[i].setX(x+pasos);
                        x = arreglo[i].getX();
                        String coor2 = String.format("(%.2f,%.2f) ",x,y);
                        arreglo[i].agregarCodigo(coor2);
                }
        }

        public void algoritmoGnome(int n)
        {
                int i = 1, z;
                int aux;
                String coordenadas;
                String cadAux;
      while(i < n)
      {
        if (i == 0 || arreglo[i-1].getTrans() <= arreglo[i].getTrans())
          i++;
        else
        {
          aux = arreglo[i - 1].getTrans();
                cadAux = arreglo[i-1].getCodigo();
          arreglo[i - 1].setTrans(arreglo[i].getTrans());
                arreglo[i - 1].setCodigo(arreglo[i].getCodigo());
          arreglo[i].setTrans(aux);
                arreglo[i].setCodigo(cadAux);
          i --;

          for(z=0;z<n;z++){
                                arreglo[z].sumaX(pasos);
                                coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                                arreglo[z].agregarCodigo(coordenadas);
                        }

        }
      }

      for(z=0;z<n;z++){
        arreglo[z].sumaX(pasos);
        coordenadas = String.format("-- (%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
        arreglo[z].agregarCodigo(coordenadas);
       }
        }
```

```
	public void creaArchivo(String nombre,int pto,int n)
	 {
		CrearArchivo archivo = new CrearArchivo();
		archivo.abrirArchivo(nombre);
		String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{slides}\n%cusepackage{tikz}\n%cb
egin{document}\n%cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,pto);
		String pie =
String.format("%cend{tikzpicture}\n%cend{document}",92,92);
		archivo.escribe(encabezado);
		for(int i=0;i<n;i++){
			archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
		}
		archivo.escribe(pie);
		archivo.cerrarArchivo();
	 }

	public void ejecutaGnome(String nombre,String color, int n)
	{
		int i;
		mezclaArreglo(n);
	//	for(i=0;i<n;i++)
	//		arreglo[i].imprime();
		int q = intercambios(n);
		pasos = 24.0 / q;
		estableceColor(color,n);
		iniciaCoordenadas(n);
		algoritmoGnome(n);
		int p;
		if(n >=2 && n<=6)
			p=7;
		else p=3;
		creaArchivo(nombre,p,n);
	}

}//fin class
```

GnomeP.java

```
/*
 * Clase para el algortimo Gnome y la visualización de
 * pirámide
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
```

```
import Nodos.Nodo2;
import java.util.Random;

public class GnomeP {
    Nodo2 vector[];

        public void algoritmoGnome(String nombreArchivo)
        {
                String codigo,encabezado,pie,newpage,rec;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
           double y=0.0,x,aux;
                int contador=0,i=1;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}[rounded corners,ultra thick]\n",92,92,92,92);
                a.escribe(encabezado); //Escribe el encabezado en el archivo

        newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}[
rounded corners,ultra thick]",92,92,92);

                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                int t = vector.length; //t será el tamaño del vector
while(i < vector.length) //Empieza el algoritmo de gnome
      {
        if (i == 0 || vector[i-1].getLargo() <= vector[i].getLargo()) // Empieza  a comparar
           i++;
        else
        {
           contador++; //Cuenta las iteraciones del algoritmo
           aux = vector[i-1].getLargo();// Empieza el intercambio de posiciones
           vector[i-1].setLargo(vector[i].getLargo());
           vector[i].setLargo(aux) ;
           i --;
           for(int k=0;k<vector.length;k++){ //Este ciclo imprime el código por cada
iteración del algoritmo
                vector[k].setY(y);
                     x=(vector.length-vector[k].getLargo())/2;//modifica coordenadas en x
                     vector[k].setX(x);
                     codigo=String.format("%cshade[top color=%s] (%.2f,%.2f) rectangle
+(%.1f,0.5);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getLargo(
));
                     a.escribe(codigo);
                     y=y-0.5;              //Modifica coordenadas en y
                                }
                y=y-0.5;
```

```
                    rec=String.format("%cshade[top color=white] (0.0,%.2f) rectangle
+(10,0.5);",92,y);
                    a.escribe(rec);    //Imprime un espacio por cada iteración
                    y=y-0.5;           //modifica coordenadas en y

                    if (contador==2){ //Si se cuenta dos iteraciones por página
               a.escribe(newpage);//Se requerira de una nueva página
               contador=0;        // El contador de las iteraciones se inicializa en cero
               y=0;           } //Modifica coordenadas en y
         }
      }

          a.escribe(pie); // Escribe el pie en el archivo
         a.cerrarArchivo();
                  }

         public void creaVector(int num,String color){ //Crea vector de números aleatorios
                  int vectors[] = new int[num];
                  int d,j=1,i,aux;
               int pos=0;
               Random aleatorio=new Random();
               for(i=0;i<num;i++)              //Se asignan valores al vector
                  vectors[i]=i+1;

         for(i=0;i<num;i++,j++){           // Se desordena el vector
                           d = aleatorio.nextInt(num-1);
                           aux=vectors[d];
                           vectors[d]=vectors[num-j];
                           vectors[num-j]=aux;
                                }

         vector=new Nodo2[num];
         for(i=0;i<num;i++){ // se guardaran las propiedades en el vector tipo Nodo
                  vector[i]=new Nodo2(pos,0.0,vectors[i],color);
                        }
                   }

}
```

InsercionE.java

```
/*
 * Clase para la opción con el algortimo de
 * Inserción y la visualización de estrellas
 */

package Algoritmos;
```

```
import ManejoArchivos.CrearArchivo;
import Nodos.Nodo3;
import java.util.Random;


public class InsercionE {
   Nodo3 vector[];

 public void insercionDirecta(String nombreArchivo,String colorL,String colorR){
          String codigo,encabezado,pie;
          //String nombreArchivo="ejemplo1.tex";
          CrearArchivo a =new CrearArchivo();
          a.abrirArchivo(nombreArchivo);
       double y=0.0;
          int contador = 0,d=0,i,j;
          encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);

          String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
          pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);

          a.escribe(encabezado);

 int n = vector.length;
//here
for (i=1; i < n; i++){
          j = i;
          int aux = vector[i].getPicos();
      while ( j > 0 && aux < vector[j-1].getPicos() ) {
           vector[j].setPicos(vector[j-1].getPicos());
//         numbers[j] = numbers[j-1];
        j--;
           d=1;
      }
          vector[j].setPicos(aux);
//         numbers[j] = aux;
          if (d==1){

          contador++;
          for(int k=0;k<n;k++){
```

```
                    vector[k].setY(y);
                    codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[k].getX(),vector[k].getY(),vector[k].getPicos(),colorL,colorR);
                    a.escribe(codigo);
          }

                    d=0;
                    y=y-2;
              if (contador==8){
                    a.escribe(newpage);
                    contador=0;
                              y=0.0;
                    }
          }
   }
//here
          a.escribe(pie);
        a.cerrarArchivo();
  }


  public void creaVector( int num ){
          int vectors[] = new int[num];
          int d, j, i, aux;
        double pos=0.0;
        Random aleatorio=new Random();
        for( i = 0, j = 3; i < num; i++, j++ )
          vectors[i]=j;

        for( i = 0, j = 1; i < num; i++, j++ ){
          d = aleatorio.nextInt(num-1);
          aux=vectors[d];
             vectors[d]=vectors[num-j];
          vectors[num-j]=aux;
        }
          vector=new Nodo3[num];
          for( i = 0; i < num; i++, pos += 1.5 )
                    vector[i]=new Nodo3(pos,4.0,vectors[i]);
  }
}
```

InsercionEl.java

```
/*
 * Clase para la opción con el algortimo de
```

```
 * Inserción y la visualización de elementos
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo4;
import Nodos.NodoElementos;
import java.util.Random;


public class InsercionEl {
   Nodo4 vector[];
   NodoElementos elementos[];

        public void insercion(String nombreArchivo,String colorL,String colorR)
        {
              String codigo,encabezado,pie;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
              int contador=0,d=0;
              double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
              a.escribe(encabezado);

                 String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);

int n = vector.length;
int j;

for (int i=1; i < n; i++){
        j = i;
        int atomico = vector[i].getAtomico();
        String simbolo = vector[i].getSimbolo();
   while ( j > 0 && atomico < vector[j-1].getAtomico() ) {
        vector[j].setAtomico(vector[j-1].getAtomico());
        vector[j].setSimbolo(vector[j-1].getSimbolo());
       j--;
```

```
            d=1;
        }
            vector[j].setSimbolo(simbolo);
            vector[j].setAtomico(atomico);
            if (d==1){

            contador++;
            for(int k=0;k<n;k++){
                    vector[k].setY(y);
                    codigo=String.format("%cnode at (%.2f,%.2f) [rectangle,minimum
height=1cm,minimum width=1cm,draw=%s,fill=%s]
{%s};",92,vector[k].getX(),vector[k].getY(),colorL,colorR,vector[k].getSimbolo());
                    a.escribe(codigo);
            }

                    d=0;
                    y=y-2;
                if (contador==8){
                    a.escribe(newpage);
                    contador=0;
                            y=0.0;
                    }
            }
      }
//here


            a.escribe(pie);
          a.cerrarArchivo();
            }//fin del metodo

public void creaElementos(){
 elementos=new NodoElementos[118];
elementos[0]=new NodoElementos("H",1);
elementos[1]=new NodoElementos("He",2);
elementos[2]=new NodoElementos("Li",3);
elementos[3]=new NodoElementos("Be",4);
elementos[4]=new NodoElementos("B",5);
elementos[5]=new NodoElementos("C",6);
elementos[6]=new NodoElementos("N",7);
elementos[7]=new NodoElementos("O",8);
elementos[8]=new NodoElementos("F",9);
elementos[9]=new NodoElementos("Ne",10);
elementos[10]=new NodoElementos("Na",11);
elementos[11]=new NodoElementos("Mg",12);
elementos[12]=new NodoElementos("Al",13);
elementos[13]=new NodoElementos("Si",14);
```

```
elementos[14]=new NodoElementos("P",15);
elementos[15]=new NodoElementos("S",16);
elementos[16]=new NodoElementos("Cl",17);
elementos[17]=new NodoElementos("Ar",18);
elementos[18]=new NodoElementos("K",19);
elementos[19]=new NodoElementos("Ca",20);
elementos[20]=new NodoElementos("Sc",21);
elementos[21]=new NodoElementos("Ti",22);
elementos[22]=new NodoElementos("V",23);
elementos[23]=new NodoElementos("Cr",24);
elementos[24]=new NodoElementos("Mn",25);
elementos[25]=new NodoElementos("Fe",26);
elementos[26]=new NodoElementos("Co",27);
elementos[27]=new NodoElementos("Ni",28);
elementos[28]=new NodoElementos("Cu",29);
elementos[29]=new NodoElementos("Zn",30);
elementos[30]=new NodoElementos("Ga",31);
elementos[31]=new NodoElementos("Ge",32);
elementos[32]=new NodoElementos("As",33);
elementos[33]=new NodoElementos("Se",34);
elementos[34]=new NodoElementos("Br",35);
elementos[35]=new NodoElementos("Kr",36);
elementos[36]=new NodoElementos("Rb",37);
elementos[37]=new NodoElementos("Sr",38);
elementos[38]=new NodoElementos("Y",39);
elementos[39]=new NodoElementos("Zr",40);
elementos[40]=new NodoElementos("Nb",41);
elementos[41]=new NodoElementos("Mo",42);
elementos[42]=new NodoElementos("Tc",43);
elementos[43]=new NodoElementos("Ru",44);
elementos[44]=new NodoElementos("Rh",45);
elementos[45]=new NodoElementos("Pd",46);
elementos[46]=new NodoElementos("Ag",47);
elementos[47]=new NodoElementos("Cd",48);
elementos[48]=new NodoElementos("In",49);
elementos[49]=new NodoElementos("Sn",50);
elementos[50]=new NodoElementos("Sb",51);
elementos[51]=new NodoElementos("Te",52);
elementos[52]=new NodoElementos("I",53);
elementos[53]=new NodoElementos("Xe",54);
elementos[54]=new NodoElementos("Cs",55);
elementos[55]=new NodoElementos("Ba",56);
elementos[56]=new NodoElementos("La",57);
elementos[57]=new NodoElementos("Ce",58);
elementos[58]=new NodoElementos("Pr",59);
elementos[59]=new NodoElementos("Nd",60);
elementos[60]=new NodoElementos("Pm",61);
```

```
elementos[61]=new NodoElementos("Sm",62);
elementos[62]=new NodoElementos("Eu",63);
elementos[63]=new NodoElementos("Gd",64);
elementos[64]=new NodoElementos("Tb",65);
elementos[65]=new NodoElementos("Dy",66);
elementos[66]=new NodoElementos("Ho",67);
elementos[67]=new NodoElementos("Er",68);
elementos[68]=new NodoElementos("Tm",69);
elementos[69]=new NodoElementos("Yb",70);
elementos[70]=new NodoElementos("Lu",71);
elementos[71]=new NodoElementos("Hf",72);
elementos[72]=new NodoElementos("Ta",73);
elementos[73]=new NodoElementos("W",74);
elementos[74]=new NodoElementos("Re",75);
elementos[75]=new NodoElementos("Os",76);
elementos[76]=new NodoElementos("Ir",77);
elementos[77]=new NodoElementos("Pt",78);
elementos[78]=new NodoElementos("Au",79);
elementos[79]=new NodoElementos("Hg",80);
elementos[80]=new NodoElementos("Tl",81);
elementos[81]=new NodoElementos("Pb",82);
elementos[82]=new NodoElementos("Bi",83);
elementos[83]=new NodoElementos("Po",84);
elementos[84]=new NodoElementos("At",85);
elementos[85]=new NodoElementos("Rn",86);
elementos[86]=new NodoElementos("Fr",87);
elementos[87]=new NodoElementos("Ra",88);
elementos[88]=new NodoElementos("Ac",89);
elementos[89]=new NodoElementos("Th",90);
elementos[90]=new NodoElementos("Pa",91);
elementos[91]=new NodoElementos("U",92);
elementos[92]=new NodoElementos("Np",93);
elementos[93]=new NodoElementos("Pu",94);
elementos[94]=new NodoElementos("Am",95);
elementos[95]=new NodoElementos("Cm",96);
elementos[96]=new NodoElementos("Bk",97);
elementos[97]=new NodoElementos("Cf",98);
elementos[98]=new NodoElementos("Es",99);
elementos[99]=new NodoElementos("Fm",100);
elementos[100]=new NodoElementos("Md",101);
elementos[101]=new NodoElementos("No",102);
elementos[102]=new NodoElementos("Lr",103);
elementos[103]=new NodoElementos("Rf",104);
elementos[104]=new NodoElementos("Db",105);
elementos[105]=new NodoElementos("Sg",106);
elementos[106]=new NodoElementos("Bh",107);
elementos[107]=new NodoElementos("Hs",108);
```

```
elementos[108]=new NodoElementos("Mt",109);
elementos[109]=new NodoElementos("Ds",110);
elementos[110]=new NodoElementos("Uuu",111);
elementos[111]=new NodoElementos("Uub",112);
elementos[112]=new NodoElementos("Uut",113);
elementos[113]=new NodoElementos("Uuq",114);
elementos[114]=new NodoElementos("Uup",115);
elementos[115]=new NodoElementos("Uuh",116);
elementos[116]=new NodoElementos("Uus",117);
elementos[117]=new NodoElementos("Uuo",118);

}



public void creaVector( int num ){

          int d, j, i;
       double pos=0.0;
          creaElementos();
       Random aleatorio=new Random();
//       for( i = 0, j = 3; i < num; i++, j++ )
  //       vectors[i]=j;

          vector=new Nodo4[num];
       for( i = 0; i < num; i++, pos += 1.0){
          d = aleatorio.nextInt(118);
             vector[i]=new
Nodo4(pos,0.0,elementos[d].getAtomico(),elementos[d].getSimbolo());
          }
}
}
```

InsercionF.java

```
/*
 * Clase para la opción con el algortimo de
 * Inserción y la visualización de figuras o círculos
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;
```

```
public class InsercionF {
   Nodo vector[];

        public void insercionDirecta(String nombreArchivo)
        {
                String codigo,encabezado,pie;
                //String nombreArchivo="ejemplo1.tex";
              CrearArchivo a =new CrearArchivo();
              a.abrirArchivo(nombreArchivo);
            double y=0.0;

encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbegin{do
cument}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92);
            a.escribe(encabezado);

pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,92,92);
            for (int i = 1; i < vector.length; i++)
                  {
                        double aux= vector[i].getRadio();
                        for (int j = i-1; j>=0 && (vector[j].getRadio()< aux);j--)
                              {
                              vector[j+1].setRadio(vector[j].getRadio());
                              vector[j].setRadio(aux);
                              }
                              for(int k=0;k<vector.length;k++){
                                    //vector[k].imprime();
                                    vector[k].setY(y);
                                    codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getRa
dio());
                                    a.escribe(codigo);
                                          }
                              y=y-2;
             }


            a.escribe(pie);
       a.cerrarArchivo();


                  }


        public void creaVector(int num,String color){
        int vectors[] = new int[num];
```

```
        int d,j=1,i,aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for(i=0;i<num;i++)
        vectors[i]=i+1;

      for(i=0;i<num;i++,j++){
        d = aleatorio.nextInt(num-1);
              aux=vectors[d];
              vectors[d]=vectors[num-j];
           vectors[num-j]=aux;
      }


        vector=new Nodo[num];
        for(i=0;i<num;i++,pos+=2.0)
                vector[i]=new Nodo(pos,0.0,color,vectors[i]*0.1);
        }

}
```

InsercionL.java

```
/*
 * Clase para la opción con el algortimo de
 * Inserción y la visualización de líneas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;


public class InsercionL {
        private Nodo arreglo[];
        private int vector[];
        private double pasos;

        public void creaVector(int n)
         {
                int m,k,i;
                arreglo = new Nodo[n];
                m = (90 / n);
                k=m;
                m = 10;
```

```
                for(i=0;i<n;i++,m+=k)
                        arreglo[i] = new Nodo(m,"draw [color = ");
        }

        public void mezclaArreglo(int n)
        {
                int i,aux,m;
                vector = new int[n];
                Random aleatorios = new Random();
                Nodo temp = new Nodo();
                m=n;
                for(i=0;i<n;i++,m--){
                        aux = aleatorios.nextInt(m);
                        temp = arreglo[m-1];
                        arreglo[m-1] = arreglo[aux];
                        vector[m-1] = arreglo[aux].getTrans();
                        arreglo[aux]= temp;
                }
        }

public int intercambios(int n){
        int i ,j , aux ,cuenta=0, d = 0;
      for (i=1; i < n; i++) {
                j = i;
                aux = vector[i];

          while ( j > 0 && aux < vector[j-1] ) {
        vector[j] = vector[j-1];
        j--;
                d=1;
          }
          vector[j] = aux;
          if (d==1){
                cuenta++;
                d=0;
          }
  }
      return cuenta+2;
}

        public void estableceColor(String color,int n)
        {
                for(int i=0;i<n;i++)
                        arreglo[i].agregaColor(color);
        }

        public void iniciaCoordenadas(int n)
```

```
        {
                double aux,x,y = 0.0;
                int i;
                aux = (14.5) / (n-1);
                y =aux;
                for(i=1;i<n;i++, y+=aux)
                        arreglo[i].setY(y*-1.0);
                for(i=0;i<n;i++){
                        x = arreglo[i].getX();
                        y = arreglo[i].getY();
                        String coor = String.format("(%.2f,%.2f) -- ",x,y);
                        arreglo[i].agregarCodigo(coor);
                        arreglo[i].setX(x+pasos);
                        x = arreglo[i].getX();
                        String coor2 = String.format("(%.2f,%.2f) ",x,y);
                        arreglo[i].agregarCodigo(coor2);
                }
        }

        public void algoritmoInsercion(int n)
        {
                int i=1 ,j ,z,d=0;
                int aux;
                String coordenadas;
                String cadAux;
//here
for (i=1; i < n; i++){
        j = i;
        aux = arreglo[i].getTrans();
        cadAux = arreglo[i].getCodigo();

    while ( j > 0 && aux < arreglo[j-1].getTrans() ) {
         arreglo[j].setTrans(arreglo[j-1].getTrans());
         arreglo[j].setCodigo(arreglo[j-1].getCodigo());
//       numbers[j] = numbers[j-1];
     j--;
         d=1;
    }
        arreglo[j].setTrans(aux);
        arreglo[j].setCodigo(cadAux);
//      numbers[j] = aux;
        if (d==1){
          for(z=0;z<n;z++){
                arreglo[z].sumaX(pasos);
                coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                arreglo[z].agregarCodigo(coordenadas);
```

```
            }
            d=0;
          }
    }
//here
        for(z=0;z<n;z++){
          arreglo[z].sumaX(pasos);
          coordenadas = String.format("-- (%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
          arreglo[z].agregarCodigo(coordenadas);
         }
          }

          public void creaArchivo(String nombre,int pto,int n)
           {
                    CrearArchivo archivo = new CrearArchivo();
                    archivo.abrirArchivo(nombre);
                    String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{slides}\n%cusepackage{tikz}\n%cb
egin{document}\n%cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,pto);
                    String pie =
String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                    archivo.escribe(encabezado);
                    for(int i=0;i<n;i++){
                              archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
                    }
                    archivo.escribe(pie);
                    archivo.cerrarArchivo();
           }

          public void ejecutaInsercion(String nombre,String color, int n)
          {
                    int i;
                    mezclaArreglo(n);
          //        for(i=0;i<n;i++)
          //                  arreglo[i].imprime();
                    int q = intercambios(n);
                    pasos = 24.0 / q;
                    estableceColor(color,n);
                    iniciaCoordenadas(n);
                    algoritmoInsercion(n);
                    int p;
                    if(n >=2 && n<=6)
                              p=7;
                    else p=3;
                    creaArchivo(nombre,p,n);
          }
}
```

InsercionP.java

```
/*
 * Clase para la opción con el algortimo de
 * Inserción y la visualización de pirámide
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo2;
import java.util.Random;

public class InsercionP {
   Nodo2 vector[];

        public void  insercionDirecta(String nombreArchivo)
        {
                String codigo,encabezado,pie,newpage,rec;
                //String nombreArchivo="ejemplo1.tex";
              CrearArchivo a =new CrearArchivo();
              a.abrirArchivo(nombreArchivo);
           double y=0.0,x;
                int contador=0;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}[rounded corners,ultra thick]\n",92,92,92,92);
                a.escribe(encabezado);

        newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}[
rounded corners,ultra thick]",92,92,92);

                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                for (int i = 1; i < vector.length; i++)
                        {
                                double aux= vector[i].getLargo();
                      contador++;

                                for (int j = i-1; j>=0 && (vector[j].getLargo()< aux);j--)
                         {
                                    vector[j+1].setLargo(vector[j].getLargo());
                                    vector[j].setLargo(aux);
```

```
                                    }
                                    for(int k=0;k<vector.length;k++){
                                   //vector[k].imprime();
                                            vector[k].setY(y);
                                 x=(vector.length-vector[k].getLargo())/2;
                                            vector[k].setX(x);
                                            codigo=String.format("%cshade[top color=%s]
(%.2f,%.2f) rectangle
+(%.1f,0.5);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getLargo(
));
                                        a.escribe(codigo);
                                    y=y-0.5;
                                                                }

                 y=y-0.5;
                      rec=String.format("%cshade[top color=white] (0.0,%.2f) rectangle
+(10,0.5);",92,y);
                        a.escribe(rec);
                        y=y-0.5;
                 if (contador==2){
                 a.escribe(newpage);
                 contador=0;     }
                  }

               //System.out.println("%d",contador);
               a.escribe(pie);
          a.cerrarArchivo();
                  }

          public void creaVector(int num,String color){
                  int vectors[] = new int[num];
                  int d,j=1,i,aux;
                int pos=0;
                Random aleatorio=new Random();
                for(i=0;i<num;i++)
                  vectors[i]=i+1;

          for(i=0;i<num;i++,j++){
                            d = aleatorio.nextInt(num-1);
                            aux=vectors[d];
                            vectors[d]=vectors[num-j];
                            vectors[num-j]=aux;
                                 }

          vector=new Nodo2[num];
          for(i=0;i<num;i++){
                  //pos=(30-pos)/2;
```

```
                vector[i]=new Nodo2(pos,0.0,vectors[i],color);
                    }
                 }

}
```

IntercambioL.java

```
/*
 * Clase para la opción con el algortimo de
 * burbuja y la visualización de líneas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;


public class IntercambioL {
        private Nodo arreglo[];
        private int vector[];
        private double pasos;

        public void creaVector(int n)
         {
                int m,k,i;
                arreglo = new Nodo[n];
                m = (90 / n);
                k=m;
                m = 10;
                for(i=0;i<n;i++,m+=k)
                        arreglo[i] = new Nodo(m,"draw [color = ");
         }

        public void mezclaArreglo(int n)
        {
                int i,aux,m;
                vector = new int[n];
                Random aleatorios = new Random();
                Nodo temp = new Nodo();
                m=n;
                for(i=0;i<n;i++,m--){
                        aux = aleatorios.nextInt(m);
                        temp = arreglo[m-1];
```

```
                    arreglo[m-1] = arreglo[aux];
                    vector[m-1] = arreglo[aux].getTrans();
                    arreglo[aux]= temp;
                }
        }

        public int intercambios(int n)
{
    int i,j , temp,cuenta=0;

        for(i = 0; i < n; i++) {
                for(j = i + 1; j < n; j++) {
                        if (vector[i] > vector[j]) {
                                temp = vector[i];
                                vector[i] = vector[j];
                                vector[j] = temp;
                                cuenta++;
                        }
                }
        }
    return cuenta+2;
}

        public void estableceColor(String color,int n)
        {
                for(int i=0;i<n;i++)
                        arreglo[i].agregaColor(color);
        }

        public void iniciaCoordenadas(int n)
        {
                double aux,x,y = 0.0;
                int i;
                aux = (14.5) / (n-1);
                y =aux;
                for(i=1;i<n;i++, y+=aux)
                        arreglo[i].setY(y*-1.0);
                for(i=0;i<n;i++){
                        x = arreglo[i].getX();
                        y = arreglo[i].getY();
                        String coor = String.format("(%.2f,%.2f) -- ",x,y);
                        arreglo[i].agregarCodigo(coor);
                        arreglo[i].setX(x+pasos);
                        x = arreglo[i].getX();
                        String coor2 = String.format("(%.2f,%.2f) ",x,y);
                        arreglo[i].agregarCodigo(coor2);
                }
```

```
        }

        public void algoritmoIntercambio(int n)
        {
                int i=1 ,j ,z;
                int aux;
                String coordenadas;
                String cadAux;
//here

        for(i = 0; i < n; i++) {
                 for(j = i + 1; j < n; j++) {
                        if (arreglo[i].getTrans() > arreglo[j].getTrans()) {
                                aux = arreglo[i].getTrans();
                                cadAux = arreglo[i].getCodigo();
                                arreglo[i].setTrans(arreglo[j].getTrans());
                             arreglo[i].setCodigo(arreglo[j].getCodigo());
                                arreglo[j].setTrans(aux);
                                arreglo[j].setCodigo(cadAux);
                           for(z=0;z<n;z++){
                                arreglo[z].sumaX(pasos);
                                coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                                arreglo[z].agregarCodigo(coordenadas);
                            }

                        }
                 }
         }
//here
       for(z=0;z<n;z++){
          arreglo[z].sumaX(pasos);
          coordenadas = String.format("-- (%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
          arreglo[z].agregarCodigo(coordenadas);
         }
          }

          public void creaArchivo(String nombre,int pto,int n)
           {
                CrearArchivo archivo = new CrearArchivo();
                archivo.abrirArchivo(nombre);
                String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{slides}\n%cusepackage{tikz}\n%cb
egin{document}\n%cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,pto);
                String pie =
String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                archivo.escribe(encabezado);
```

```
                for(int i=0;i<n;i++){
                        archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
                }
                archivo.escribe(pie);
                archivo.cerrarArchivo();
         }

        public void ejecutaIntercambio(String nombre,String color, int n)
        {
                int i;
                mezclaArreglo(n);
        //      for(i=0;i<n;i++)
        //              arreglo[i].imprime();
                int q = intercambios(n);
                pasos = 24.0 / q;
                estableceColor(color,n);
                iniciaCoordenadas(n);
                algoritmoIntercambio(n);
                int p;
                if(n >=2 && n<=6)
                        p=7;
                else p=3;
                creaArchivo(nombre,p,n);
        }

}
```

Interpolacion.java

```
/*
 Clase para el algortimo Interpolacón y la visualización de
 * letras
*/

package Algoritmos;

import ManejoArchivos.CrearArchivo;

public class Interpolacion {
   int vector[];
        public void interpolacion(String nombreArchivo,String color, int elemento) {
                String codigo,encabezado,pie;
        CrearArchivo a =new CrearArchivo();
        a.abrirArchivo(nombreArchivo);
           double y=0.0,x=0.0;
                int j;
```

```
encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cbegin{d
ocument}\n%cbegin{tikzpicture}\n",92,92,92,92);
                a.escribe(encabezado);
                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);


                int low = 0;
                int high = vector.length - 1;
                int mid=0;
                codigo=String.format("%cnode at (%.2f,%.2f) [red]
{%c};",92,x,y,elemento+64);
                        a.escribe(codigo);
                y -= 1.5;
        for(j=0,x=0.0;j<vector.length;j++,x+=1.0){
                        codigo=String.format("%cnode at (%.2f,%.2f) [%s]
{%c};",92,x,y,color,vector[j]+64);
                        a.escribe(codigo);
        }
        y -= 1.5;
      //inicia la ejecución del algortimo de interpolación
        while (vector[low] < elemento && vector[high] >= elemento) {
                mid = low + ((elemento - vector[low]) * (high - low)) / (vector[high] -
vector[low]);
                codigo=String.format("%cnode at (%.2f,%.2f) [red]
{%c};",92,mid*1.0,y,elemento+64);
                        a.escribe(codigo);
                y -= 1.5;
            //Se escriben los cambios en el archivp
                for(j=0,x=0.0;j<vector.length;j++,x+=1.0){
                        codigo=String.format("%cnode at (%.2f,%.2f) [%s]
{%c};",92,x,y,color,vector[j]+64);
                        a.escribe(codigo);
                }
                y -= 1.5;
                if (vector[mid] < elemento)
                        low = mid + 1;
                else if (vector[mid] > elemento)
                        high = mid - 1;
                else
                        break;
        }
                y +=1.5;
            //Se escribe en el archivo las últimas propiedades
                codigo=String.format("%cnode at (%.2f,%.2f) [red]
{%c};",92,mid*1.0,y,elemento+64);
                a.escribe(codigo);
```

```
                a.escribe(pie);
                a.cerrarArchivo();
        }

      //Se inicializa el arreglo
        public void creaVector(int num){
            int i;
            vector = new int[num];
       for(i=0;i<num;i++)
                vector[i]= i+1;
        }
}
```

Mergesort.java

```
package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo7;
import java.util.Random;

public class Mergesort {
   Nodo7 vector[];
   private int nn;
   private CrearArchivo a;
   private double y;
   private int contador;


public void crearArchivo(String nombreArchivo){
        String encabezado,pie,codigo;
        a =new CrearArchivo();
        a.abrirArchivo(nombreArchivo);
        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cus
epackage{anysize}\n%cmarginsize{1cm}{1cm}{1cm}{1cm}\n%cbegin{document}\n%cb
egin{tikzpicture}\n",92,92,92,92,92,92);
        a.escribe(encabezado);
      int p=0,g;
      int t = vector.length-1;
      algoritmoMergesort(p,t);
      for(g=0;g<vector.length;g++){
            vector[g].setY(y);
            codigo=String.format("%ccoordinate [label=right:%ctextcolor{%s}{$%c$}] ()
at
```

```
(%.2f,%.2f);",92,92,vector[g].getColor(),vector[g].getLetra(),vector[g].getX(),vector[g].get
Y());
                a.escribe(codigo);
              }

        pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
        a.escribe(pie);
           a.cerrarArchivo();
}
public void algoritmoMergesort(int l, int r)
{
           String codigo;
        int i,j,k,m,g,h;
        int b[]=new int[nn];


           if (r > l)
           {


      for(g=0;g<vector.length;g++)
                    vector[g].setY(y);

              for(g=l;g<=r;g++){
                 vector[g].setY(y);
                 codigo=String.format("%ccoordinate [label=right:%ctextcolor{%s}{$%c$}] ()
at
(%.2f,%.2f);",92,92,vector[g].getColor(),vector[g].getLetra(),vector[g].getX(),vector[g].get
Y());
                 a.escribe(codigo);
               }
              y=y-1.0;
              contador++;

               m = (r+l) /2;
               algoritmoMergesort(l, m);

              for(g=0;g<vector.length;g++)
                    vector[g].setY(y);

              for(g=l;g<=m;g++){
                 vector[g].setY(y);
                 codigo=String.format("%ccoordinate [label=right:%ctextcolor{%s}{$%c$}] ()
at
(%.2f,%.2f);",92,92,vector[g].getColor(),vector[g].getLetra(),vector[g].getX(),vector[g].get
Y());
                 a.escribe(codigo);
```

```
                }
            y=y-1.0;
            contador++;

            algoritmoMergesort(m+1, r);


          for(g=0;g<vector.length;g++)
                vector[g].setY(y);

          for(g=m+1;g<=r;g++){
             vector[g].setY(y);
             codigo=String.format("%ccoordinate [label=right:%ctextcolor{%s}{$%c$}] ()
at
(%.2f,%.2f);",92,92,vector[g].getColor(),vector[g].getLetra(),vector[g].getX(),vector[g].get
Y());
              a.escribe(codigo);
                        }y=y-1.0;
          contador++;

                for (i= m+1; i > l;i--){
                        b[i-1] = vector[i-1].getLetra();

            }
            for (j= m; j < r;j++){
                        b[r+m-j] = vector[j+1].getLetra();
                         }

            for (k=l ; k <=r; k++){
                        if(b[i] < b[j]){
                           vector[k].setLetra(b[i++]);
                 }
                else
                           vector[k].setLetra(b[j--]);
            }

}



 }



        public void creaVector(int num,String color){
        int vectors[] = new int[num];
        int d,j=1,i,aux;
```

```
        double pos=0.0;
           y=0.0;
           contador=0;
        Random aleatorio=new Random();
        nn=num;
        vector=new Nodo7[num];
           for(i=0;i<num;i++,pos+=1){
                    d = aleatorio.nextInt(25)+1;
                    vector[i]=new Nodo7(pos,0.0,color,d+64);
           }
     }
}
```

QuickL.java

```
/*
 * Clase para el algortimo Quick sort y la visualización de
 * líneas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;


//Se ejecuta el algortimo de quick sort
public class QuickL {
   private Nodo arreglo[];
         private int vector[];
         private double pasos;
         private int cuenta;

        // Se inicializa el vector utilizado
           public void creaVector(int n)
            {
                    int m,k,i;
                    arreglo = new Nodo[n];
                    m = (90 / n);
                    k=m; //Se calcula la separación de los elementos en la hoja
                    m = 10;
                    for(i=0;i<n;i++,m+=k)
                             arreglo[i] = new Nodo(m,"draw [color = ");
            }

        //Se mezcla el arreglo
```

```
        public void mezclaArreglo(int n)
        {
                int i,aux,m;
                vector = new int[n];
                Random aleatorios = new Random();
                Nodo temp = new Nodo();
                m=n;
                for(i=0;i<n;i++,m--){
                        aux = aleatorios.nextInt(m);
                        temp = arreglo[m-1];
                        arreglo[m-1] = arreglo[aux];
                        vector[m-1] = arreglo[aux].getTrans();
                        arreglo[aux]= temp;
                }
        }

      //Modo utilizado para saber cuantos pasos hace el algoritmo
      //y distribuir mejor los elementos en la página
int colocarIntercambios( int b, int t ){
   int i;
   int pivote, valor_pivote;
   int temp;

   pivote = b;
   //Se ejecuta el algortimo de quick sort
   valor_pivote = vector[pivote];
   for (i=b+1; i<=t; i++){
      if (vector[i] < valor_pivote){
            pivote++;
           temp=vector[i];
           vector[i]=vector[pivote];
           vector[pivote]=temp;
      }
   }
   cuenta++;
   //imprime(v,10);
   temp=vector[b];
   vector[b]=vector[pivote];
   vector[pivote]=temp;
   return pivote;
}

//Método utilizado para contar el número de
//intercambios del algoritmo
void intercambiosQuicksort( int b, int t){
    int pivote;
    if(b < t){
```

```
        pivote=colocarIntercambios(b, t);
        intercambiosQuicksort(b, pivote-1);
        intercambiosQuicksort(pivote+1, t);
    }
}

    //Se asigna el color a la visualización
        public void estableceColor(String color,int n)
         {
                for(int i=0;i<n;i++)
                        arreglo[i].agregaColor(color);
         }

        //Inicializa el vector
           public void iniciaCoordenadas(int n)
            {
                double aux,x,y = 0.0;
                int i;
                aux = (14.5) / (n-1);
                y =aux;
                for(i=1;i<n;i++, y+=aux)
                        arreglo[i].setY(y*-1.0);
                for(i=0;i<n;i++){
                        x = arreglo[i].getX();
                        y = arreglo[i].getY();
                        String coor = String.format("(%.2f,%.2f) -- ",x,y);
                        arreglo[i].agregarCodigo(coor);
                        arreglo[i].setX(x+pasos);
                        x = arreglo[i].getX();
                        String coor2 = String.format("(%.2f,%.2f) ",x,y);
                        arreglo[i].agregarCodigo(coor2);
                }
            }


int colocar( int b, int t ){
    int i;
    int pivote, valor_pivote;
    pivote = b;
    valor_pivote = arreglo[pivote].getTrans();
    for (i=b+1; i<=t; i++){
        if (arreglo[i].getTrans() < valor_pivote){
             pivote++;
            int temp=arreglo[i].getTrans();
                String cod = arreglo[i].getCodigo();
                arreglo[i].setTrans(arreglo[pivote].getTrans());
                arreglo[i].setCodigo(arreglo[pivote].getCodigo());
```

```
                    arreglo[pivote].setTrans(temp);
                    arreglo[pivote].setCodigo(cod);
        }
    }
    int aux=arreglo[b].getTrans();
    String auxC = arreglo[b].getCodigo();

    arreglo[b].setTrans(arreglo[pivote].getTrans());
    arreglo[b].setCodigo(arreglo[pivote].getCodigo());

    arreglo[pivote].setTrans(aux);
    arreglo[pivote].setCodigo(auxC);
    for(int z=0;z<arreglo.length;z++){
                    arreglo[z].sumaX(pasos);
                    String coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                    arreglo[z].agregarCodigo(coordenadas);
    }
          //imprime(v,10);
    return pivote;
}

//Se ejecuta el algrtimo quick sort
void algoritmoQuicksort(int b, int t){
     int pivote;
     if(b < t){
        pivote=colocar(b, t);
        algoritmoQuicksort(b, pivote-1);
        algoritmoQuicksort(pivote+1, t);
     }
}

    //Método que crea el archivo .tex una vez ejecutado el algortimo
          public void creaArchivo(String nombre,int pto,int n)
           {
                    CrearArchivo archivo = new CrearArchivo();
                    archivo.abrirArchivo(nombre);
                    String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{article}\n%cusepackage{tikz}\n%c
usepackage{anysize}\n%cmarginsize{2cm}{1cm}{1cm}{1cm}\n%cbegin{document}\n%
cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,92,92,pto);
                    String pie =
String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                    archivo.escribe(encabezado);

              //Se imprimen las propiedades de cada elemento del arreglo tipo Nodo
          for(int z=0;z<arreglo.length;z++){
```

```
            arreglo[z].sumaX(pasos);
            String coordenadas = String.format("--
(%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
            arreglo[z].agregarCodigo(coordenadas);
          }

                    for(int i=0;i<n;i++){
                              archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
                    }

                    archivo.escribe(pie);
                    archivo.cerrarArchivo();
           }

        //Método que controla la ejcución de esta clase
           public void ejecutaQuick(String nombre,String color, int n)
           {
                    int i;
                    mezclaArreglo(n);
                    cuenta = 0;
                    intercambiosQuicksort(0,n-1);
                    pasos = (24.0 / (cuenta+2));
                    estableceColor(color,n);
                    iniciaCoordenadas(n);
                    algoritmoQuicksort(0,n-1);
              //Se asigna el ancho de la línea
                    int p;
                    if(n >=2 && n<=6)
                              p=7;
                    else p=3;
                    creaArchivo(nombre,p,n);
           }
}
```

Radix.java

```
/*
 *Clase para la opción con el algortimo de
 * Radix y la visualización de letras
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import java.util.Random;
```

```
public class Radix {
   int vector[];
        public void radix(String nombreArchivo,String color) {
                String codigo,encabezado,pie;
        CrearArchivo a =new CrearArchivo();
        a.abrirArchivo(nombreArchivo);
           double y=10.0,x;
                int j;

encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cbegin{d
ocument}\n%cbegin{tikzpicture}\n",92,92,92,92);
                a.escribe(encabezado);
                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);


          int[] b = new int[vector.length];
       int[] b_orig = b;
          int bits = 4;

       int rshift = 0;
       for (int mask = ~(-1 << bits); mask != 0; mask <<= bits, rshift += bits) {

          int[] cntarray = new int[1 << bits];

          for (int p = 0; p < vector.length; ++p) {
             int key = (vector[p] & mask) >> rshift;
             ++cntarray[key];
          }


                for (int i = 1; i < cntarray.length; ++i)
                 cntarray[i] += cntarray[i-1];


          for (int p = vector.length-1; p >= 0; --p) {
             int key = (vector[p] & mask) >> rshift;
             --cntarray[key];
             b[cntarray[key]] = vector[p];
          }


          int[] temp = b; b = vector; vector = temp;
                for(j=0,x=0.0;j<vector.length;j++,x+=1.0){
                        codigo=String.format("%cnode at (%.2f,%.2f) [%s]
{%c};",92,x,y,color,vector[j]+64);
                        a.escribe(codigo);
                }
```

```
                y -= 1.5;
          //System.out.println(Arrays.toString(a));
        }


        if (vector == b_orig)
           System.arraycopy(vector, 0, b, 0, vector.length);

                a.escribe(pie);
                a.cerrarArchivo();
          }


          public void creaVector(int num){
             int i;
          double pos=0.0;
             vector = new int[num];
             int d,tmp,m;
             Random al = new Random();
        for(i=0;i<num;i++)
                vector[i]= i+1;
             m=num;
              for(i=0;i<num;i++,m--){
                d = al.nextInt(m);
                tmp=vector[m-1];
                vector[m-1]=vector[d];
                vector[d]=tmp;
              }
          }
}
```

SecuencialC.java

```
/*
 * Clase para la opción con el algortimo
 * búsqueda y la visualización de círculos
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;


public class SecuencialC {
   Nodo vector[];
   Nodo elemento;
```

```
public void secuencial(String nombreArchivo)
{
        String codigo,encabezado,pie,newpage;
        String c;
CrearArchivo a =new CrearArchivo();
a.abrirArchivo(nombreArchivo);
   double y=0.0,x=0.0,xx;
        int j, b=0;

encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbegin{do
cument}\n%cbegin{tikzpicture}\n",92,92,92,92);
        a.escribe(encabezado);
        pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);

newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}",92,92,
92);

        for(int k=0;k<vector.length;k++,x+=2.0,y-=4.0){
     b++;

        c=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,elemento.getColor(),elemento.getX(),elemento.getY(),elemento.getRa
dio()*0.1);
        a.escribe(c);
        elemento.setY(elemento.getY()-4.0);
        elemento.setX(elemento.getX()+2.0);
         for(j=0, xx=0.0; j<vector.length; j++,xx+=2.0){
                codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[j].getColor(),vector[j].getX(),vector[j].getY(),vector[j].getRadio
()*0.1);
                a.escribe(codigo);
                vector[j].setY(vector[j].getY()-4.0);
         }
    if(b==4){
        a.escribe(newpage);
        b=0;
        for(int t=0;t<vector.length;t++)
           vector[t].setY(-2.0);
           elemento.setY(0.0);
    }
              System.out.println(vector[k].getRadio());
        if(elemento.getRadio()==vector[k].getRadio()){
       break;
        }
    }

     elemento.setY(elemento.getY()+2.0);
```

```
                    elemento.setX(elemento.getX()-2.0);
                    c=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,elemento.getColor(),elemento.getX(),elemento.getY(),elemento.getRa
dio()*0.1);
                    a.escribe(c);
                a.escribe(pie);
                    a.cerrarArchivo();
          }


          public void creaVector(int num,String color,int r){
             int i;
          double pos=0.0;
             vector=new Nodo[num];
                for(i=0;i<num;i++,pos+=2.0)
                            vector[i]=new Nodo(pos,-2.0,color,i+1);
                elemento=new Nodo(0.0,0.0,"red",r);
       }
}
```

SeleccionE.java

```
/*
 * Clase para la opción con el algortimo de
 * Selección y la visualización de estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo3;
import java.util.Random;


public class SeleccionE {
    Nodo3 vector[];

 public void seleccionDirecta(String nombreArchivo,String colorL,String colorR){
          String codigo,encabezado,pie;
       CrearArchivo a =new CrearArchivo();
       a.abrirArchivo(nombreArchivo);
          double y=0.0;
          int contador = 0;
          encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
```

```
        a.escribe(encabezado);
        String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);
        for (int i = 0; i < vector.length - 1; i++){
                int min = i;
                for (int j = i + 1; j < vector.length; j++){
                        if (vector[j].getPicos() < vector[min].getPicos()){
                                min = j;
                        }
                }
                if (i != min) {
                        int aux= vector[i].getPicos();
                        vector[i].setPicos(vector[min].getPicos());
                        vector[min].setPicos(aux);
                        for(int k=0;k<vector.length;k++){
                                vector[k].setY(y);
                                codigo=String.format("%cnode at (%.2f,%.2f) [star,star
points=%d,star point ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[k].getX(),vector[k].getY(),vector[k].getPicos(),colorL,colorR);
                                a.escribe(codigo);
                        }
                        y=y-2;
                        if (contador==8){
                        a.escribe(newpage);
                   contador=0;
                                y=0.0;
                        }
                }
        }
        a.escribe(pie);
      a.cerrarArchivo();
 }

 public void creaVector( int num ){
        int vectors[] = new int[num];
        int d, j, i, aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for( i = 0, j = 3; i < num; i++, j++ )
        vectors[i]=j;

      for( i = 0, j = 1; i < num; i++, j++ ){
        d = aleatorio.nextInt(num-1);
```

```
            aux=vectors[d];
               vectors[d]=vectors[num-j];
            vectors[num-j]=aux;
        }

            vector=new Nodo3[num];
            for( i = 0; i < num; i++, pos += 1.5 )
                    vector[i]=new Nodo3(pos,4.0,vectors[i]);
 }
}
```

SeleccionEl.java

```
/*
 * Clase para la opción con el algortimo de
 * selección y la visualización de elementos
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo4;
import Nodos.NodoElementos;
import java.util.Random;


public class SeleccionEl {
   Nodo4 vector[];
   NodoElementos elementos[];

            public void seleccion(String nombreArchivo,String colorL,String colorR)
            {
                    String codigo,encabezado,pie;
                CrearArchivo a =new CrearArchivo();
                a.abrirArchivo(nombreArchivo);
                    int contador=0;
                    double y=0.0;

            encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                    a.escribe(encabezado);

                        String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
```

```
        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,92,92);

      for (int i = 0; i < vector.length - 1; i++){
                int min = i;
                for (int j = i + 1; j < vector.length; j++){
                        if (vector[j].getAtomico() < vector[min].getAtomico()){
                                min = j;
                        }
                }
                if (i != min) {
                         contador++;
                         Nodo4 aux = new Nodo4();
                        aux = vector[i];
                        double auxX = vector[i].getX();
                        vector[i]=vector[min];
                        vector[min]=aux;
                        vector[min].setX(vector[i].getX());
                        vector[i].setX(auxX);
                        for(int k=0;k<vector.length;k++){
                                vector[k].setY(y);
                                codigo=String.format("%cnode at (%.2f,%.2f) [rectangle,minimum height=1cm,minimum width=1cm,draw=%s,fill=%s] {%s};",92,vector[k].getX(),vector[k].getY(),colorL,colorR,vector[k].getSimbolo());
                                a.escribe(codigo);
                        }
                        y=y-2;
                        if (contador==8){
                        a.escribe(newpage);
                    contador=0;
                                y=0.0;
                        }
                }
        }

          a.escribe(pie);
      a.cerrarArchivo();
          }//fin del metodo

public void creaElementos(){
 elementos=new NodoElementos[118];
elementos[0]=new NodoElementos("H",1);
elementos[1]=new NodoElementos("He",2);
elementos[2]=new NodoElementos("Li",3);
elementos[3]=new NodoElementos("Be",4);
elementos[4]=new NodoElementos("B",5);
```

```
elementos[5]=new NodoElementos("C",6);
elementos[6]=new NodoElementos("N",7);
elementos[7]=new NodoElementos("O",8);
elementos[8]=new NodoElementos("F",9);
elementos[9]=new NodoElementos("Ne",10);
elementos[10]=new NodoElementos("Na",11);
elementos[11]=new NodoElementos("Mg",12);
elementos[12]=new NodoElementos("Al",13);
elementos[13]=new NodoElementos("Si",14);
elementos[14]=new NodoElementos("P",15);
elementos[15]=new NodoElementos("S",16);
elementos[16]=new NodoElementos("Cl",17);
elementos[17]=new NodoElementos("Ar",18);
elementos[18]=new NodoElementos("K",19);
elementos[19]=new NodoElementos("Ca",20);
elementos[20]=new NodoElementos("Sc",21);
elementos[21]=new NodoElementos("Ti",22);
elementos[22]=new NodoElementos("V",23);
elementos[23]=new NodoElementos("Cr",24);
elementos[24]=new NodoElementos("Mn",25);
elementos[25]=new NodoElementos("Fe",26);
elementos[26]=new NodoElementos("Co",27);
elementos[27]=new NodoElementos("Ni",28);
elementos[28]=new NodoElementos("Cu",29);
elementos[29]=new NodoElementos("Zn",30);
elementos[30]=new NodoElementos("Ga",31);
elementos[31]=new NodoElementos("Ge",32);
elementos[32]=new NodoElementos("As",33);
elementos[33]=new NodoElementos("Se",34);
elementos[34]=new NodoElementos("Br",35);
elementos[35]=new NodoElementos("Kr",36);
elementos[36]=new NodoElementos("Rb",37);
elementos[37]=new NodoElementos("Sr",38);
elementos[38]=new NodoElementos("Y",39);
elementos[39]=new NodoElementos("Zr",40);
elementos[40]=new NodoElementos("Nb",41);
elementos[41]=new NodoElementos("Mo",42);
elementos[42]=new NodoElementos("Tc",43);
elementos[43]=new NodoElementos("Ru",44);
elementos[44]=new NodoElementos("Rh",45);
elementos[45]=new NodoElementos("Pd",46);
elementos[46]=new NodoElementos("Ag",47);
elementos[47]=new NodoElementos("Cd",48);
elementos[48]=new NodoElementos("In",49);
elementos[49]=new NodoElementos("Sn",50);
elementos[50]=new NodoElementos("Sb",51);
elementos[51]=new NodoElementos("Te",52);
```

```
elementos[52]=new NodoElementos("I",53);
elementos[53]=new NodoElementos("Xe",54);
elementos[54]=new NodoElementos("Cs",55);
elementos[55]=new NodoElementos("Ba",56);
elementos[56]=new NodoElementos("La",57);
elementos[57]=new NodoElementos("Ce",58);
elementos[58]=new NodoElementos("Pr",59);
elementos[59]=new NodoElementos("Nd",60);
elementos[60]=new NodoElementos("Pm",61);
elementos[61]=new NodoElementos("Sm",62);
elementos[62]=new NodoElementos("Eu",63);
elementos[63]=new NodoElementos("Gd",64);
elementos[64]=new NodoElementos("Tb",65);
elementos[65]=new NodoElementos("Dy",66);
elementos[66]=new NodoElementos("Ho",67);
elementos[67]=new NodoElementos("Er",68);
elementos[68]=new NodoElementos("Tm",69);
elementos[69]=new NodoElementos("Yb",70);
elementos[70]=new NodoElementos("Lu",71);
elementos[71]=new NodoElementos("Hf",72);
elementos[72]=new NodoElementos("Ta",73);
elementos[73]=new NodoElementos("W",74);
elementos[74]=new NodoElementos("Re",75);
elementos[75]=new NodoElementos("Os",76);
elementos[76]=new NodoElementos("Ir",77);
elementos[77]=new NodoElementos("Pt",78);
elementos[78]=new NodoElementos("Au",79);
elementos[79]=new NodoElementos("Hg",80);
elementos[80]=new NodoElementos("Tl",81);
elementos[81]=new NodoElementos("Pb",82);
elementos[82]=new NodoElementos("Bi",83);
elementos[83]=new NodoElementos("Po",84);
elementos[84]=new NodoElementos("At",85);
elementos[85]=new NodoElementos("Rn",86);
elementos[86]=new NodoElementos("Fr",87);
elementos[87]=new NodoElementos("Ra",88);
elementos[88]=new NodoElementos("Ac",89);
elementos[89]=new NodoElementos("Th",90);
elementos[90]=new NodoElementos("Pa",91);
elementos[91]=new NodoElementos("U",92);
elementos[92]=new NodoElementos("Np",93);
elementos[93]=new NodoElementos("Pu",94);
elementos[94]=new NodoElementos("Am",95);
elementos[95]=new NodoElementos("Cm",96);
elementos[96]=new NodoElementos("Bk",97);
elementos[97]=new NodoElementos("Cf",98);
elementos[98]=new NodoElementos("Es",99);
```

```
elementos[99]=new NodoElementos("Fm",100);
elementos[100]=new NodoElementos("Md",101);
elementos[101]=new NodoElementos("No",102);
elementos[102]=new NodoElementos("Lr",103);
elementos[103]=new NodoElementos("Rf",104);
elementos[104]=new NodoElementos("Db",105);
elementos[105]=new NodoElementos("Sg",106);
elementos[106]=new NodoElementos("Bh",107);
elementos[107]=new NodoElementos("Hs",108);
elementos[108]=new NodoElementos("Mt",109);
elementos[109]=new NodoElementos("Ds",110);
elementos[110]=new NodoElementos("Uuu",111);
elementos[111]=new NodoElementos("Uub",112);
elementos[112]=new NodoElementos("Uut",113);
elementos[113]=new NodoElementos("Uuq",114);
elementos[114]=new NodoElementos("Uup",115);
elementos[115]=new NodoElementos("Uuh",116);
elementos[116]=new NodoElementos("Uus",117);
elementos[117]=new NodoElementos("Uuo",118);

}



public void creaVector( int num ){
          int d, j, i;
       double pos=0.0;
          creaElementos();
       Random aleatorio=new Random();
//       for( i = 0, j = 3; i < num; i++, j++ )
  //       vectors[i]=j;

          vector=new Nodo4[num];
       for( i = 0; i < num; i++, pos += 1.0){
          d = aleatorio.nextInt(118);
             vector[i]=new
Nodo4(pos,0.0,elementos[d].getAtomico(),elementos[d].getSimbolo());
          }
}
}
```

eleccionF.java

```
/*
 * Clase para la opción con el algortimo de
 * selección y la visualización de figuras
 */
```

```
package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;




public class SeleccionF {
   Nodo vector[];

        public void seleccionDirecta(String nombreArchivo)
        {
                String codigo,encabezado,pie;
                //String nombreArchivo="ejemplo1.tex";
      CrearArchivo a =new CrearArchivo();
      a.abrirArchivo(nombreArchivo);
           double y=0.0;

encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbegin{do
cument}\n%cbegin{center}%cbegin{tikzpicture}\n",92,92,92,92,92);
           a.escribe(encabezado);

pie=String.format("%cend{tikzpicture}\n%cend{center}%cend{document}",92,92,92);
           for (int i = 0; i < vector.length - 1; i++)
                 {
                        int min = i;
                        for (int j = i + 1; j < vector.length; j++)
                        {
                                if (vector[j].getRadio() < vector[min].getRadio())
                                {
                                        min = j;
                                }
                        }
                        if (i != min)
                        {
                                double aux= vector[i].getRadio();
                                vector[i].setRadio(vector[min].getRadio());
                                vector[min].setRadio(aux);
                                for(int k=0;k<vector.length;k++){
                                        //vector[k].imprime();
                                        vector[k].setY(y);
```

```
                                        codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getRa
dio());
                                        a.escribe(codigo);
                                }
                                y=y-2;
                        }
                }
            a.escribe(pie);
        a.cerrarArchivo();
                }

          public void creaVector(int num,String color){
          int vectors[] = new int[num];
          int d,j=1,i,aux;
        double pos=0.0;
        Random aleatorio=new Random();
        for(i=0;i<num;i++)
          vectors[i]=i+1;

        for(i=0;i<num;i++,j++){
          d = aleatorio.nextInt(num-1);
          aux=vectors[d];
                        vectors[d]=vectors[num-j];
          vectors[num-j]=aux;
        }


          vector=new Nodo[num];
          for(i=0;i<num;i++,pos+=2.0)
                    vector[i]=new Nodo(pos,0.0,color,vectors[i]*0.1);
          }

}
```

SeleccionL.java

```
/*
 * Clase para la opción con el algortimo de
 * selección y la visualización de líneas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
```

```
import Nodos.Nodo;
import java.util.Random;


public class SeleccionL {
   private Nodo arreglo[];
        private int vector[];
        private double pasos;

        public void creaVector(int n)
         {
                int m,k,i;
                arreglo = new Nodo[n];
                m = (90 / n);
                k=m;
                m = 10;
                for(i=0;i<n;i++,m+=k)
                        arreglo[i] = new Nodo(m,"draw [color = ");
         }

        public void mezclaArreglo(int n)
        {
                int i,aux,m;
                vector = new int[n];
                Random aleatorios = new Random();
                Nodo temp = new Nodo();
                m=n;
                for(i=0;i<n;i++,m--){
                        aux = aleatorios.nextInt(m);
                        temp = arreglo[m-1];
                        arreglo[m-1] = arreglo[aux];
                        vector[m-1] = arreglo[aux].getTrans();
                        arreglo[aux]= temp;
                }
        }

public int intercambios(int n){
      int i,j , temp,cuenta=0,minimo;
        for(i=0 ; i<n-1 ; i++){
                minimo=i;
                for(j=i+1 ; j<n ; j++){
                        if (vector[minimo] > vector[j])
                                minimo=j;
                }
                if( i != minimo ){
                        temp=vector[minimo];
                        vector[minimo]=vector[i];
```

```
                        vector[i]=temp;
                        cuenta++;
                }
        }
    return cuenta+2;
}

        public void estableceColor(String color,int n)
         {
                for(int i=0;i<n;i++)
                        arreglo[i].agregaColor(color);
         }

        public void iniciaCoordenadas(int n)
         {
                double aux,x,y = 0.0;
                int i;
                aux = (14.5) / (n-1);
                y =aux;
                for(i=1;i<n;i++, y+=aux)
                        arreglo[i].setY(y*-1.0);
                for(i=0;i<n;i++){
                        x = arreglo[i].getX();
                        y = arreglo[i].getY();
                        String coor = String.format("(%.2f,%.2f) -- ",x,y);
                        arreglo[i].agregarCodigo(coor);
                        arreglo[i].setX(x+pasos);
                        x = arreglo[i].getX();
                        String coor2 = String.format("(%.2f,%.2f) ",x,y);
                        arreglo[i].agregarCodigo(coor2);
                }
         }

        public void algoritmoSeleccion(int n)
        {
                int i=1 ,j ,z;
                int aux,minimo;
                String coordenadas;
                String cadAux;
//here
for(i=0 ; i<n-1 ; i++){
          minimo=i;
          for(j=i+1 ; j<n ; j++){
                if (arreglo[minimo].getTrans() > arreglo[j].getTrans())
                        minimo=j;
          }
          if( i != minimo ){
```

```
                    aux = arreglo[minimo].getTrans();
                    cadAux = arreglo[minimo].getCodigo();
                    arreglo[minimo].setTrans(arreglo[i].getTrans());
                    arreglo[minimo].setCodigo(arreglo[i].getCodigo());
                    //x[minimo]=x[i];
                    arreglo[i].setTrans(aux);
                    arreglo[i].setCodigo(cadAux);
                    for(z=0;z<n;z++){
                      arreglo[z].sumaX(pasos);
                      coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                      arreglo[z].agregarCodigo(coordenadas);
                                }
             }
}
//here
       for(z=0;z<n;z++){
          arreglo[z].sumaX(pasos);
          coordenadas = String.format("-- (%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
          arreglo[z].agregarCodigo(coordenadas);
         }
          }

          public void creaArchivo(String nombre,int pto,int n)
           {
                    CrearArchivo archivo = new CrearArchivo();
                    archivo.abrirArchivo(nombre);
                    String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{slides}\n%cusepackage{tikz}\n%cb
egin{document}\n%cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,pto);
                    String pie =
String.format("%cend{tikzpicture}\n%cend{document}",92,92);
                    archivo.escribe(encabezado);
                    for(int i=0;i<n;i++){
                              archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
                    }
                    archivo.escribe(pie);
                    archivo.cerrarArchivo();
           }

          public void ejecutaSeleccion(String nombre,String color, int n)
          {
                    int i;
                    mezclaArreglo(n);
          //        for(i=0;i<n;i++)
          //                  arreglo[i].imprime();
                    int q = intercambios(n);
```

```
                pasos = 24.0 / q;
                estableceColor(color,n);
                iniciaCoordenadas(n);
                algoritmoSeleccion(n);
                int p;
                if(n >=2 && n<=6)
                        p=7;
                else p=3;
                creaArchivo(nombre,p,n);
        }
}
```

SeleccionP.java

```
/*
 * Clase para la opción con el algortimo de
 * selección y la visualización de pirámide
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo2;
import java.util.Random;

public class SeleccionP {
   Nodo2 vector[];

        public void seleccionDirecta(String nombreArchivo)
        {
                String codigo,encabezado,pie,newpage,rec;
                //String nombreArchivo="ejemplo1.tex";
             CrearArchivo a =new CrearArchivo();
             a.abrirArchivo(nombreArchivo);
           double y=0.0,x;
                int contador=0;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}[rounded corners,ultra thick]\n",92,92,92,92);
                a.escribe(encabezado);

        newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}[
rounded corners,ultra thick]",92,92,92);

                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);
             for (int i = 0; i < vector.length - 1; i++)
```

```
		{
			int min = i;
			for (int j = i + 1; j < vector.length; j++)
			{
				if (vector[j].getLargo() < vector[min].getLargo())
				{
					min = j;
				}
			}
			if (i != min)
			{
				double aux= vector[i].getLargo();
				vector[i].setLargo(vector[min].getLargo());
				vector[min].setLargo(aux);
				contador++;

				for(int k=0;k<vector.length;k++){
					//System.out.printf("%f ",vector[k].getLargo());
					vector[k].setY(y);
					x=(vector.length-vector[k].getLargo())/2;
					vector[k].setX(x);
					codigo=String.format("%cshade[top color=%s]
(%.2f,%.2f) rectangle
+(%.1f,0.5);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getLargo(
));
					a.escribe(codigo);


		y=y-0.5;
						}
		//System.out.println( );
		y=y-0.5;

		rec=String.format("%cshade[top color=white] (0.0,%.2f) rectangle
+(10,0.5);",92,y);
		a.escribe(rec);
		y=y-0.5;
			if (contador==2){
		a.escribe(newpage);
		contador=0;}
			}

		}
	   a.escribe(pie);
	  a.cerrarArchivo();
		}
```

```
        public void creaVector(int num,String color){
                int vectors[] = new int[num];
                int d,j=1,i,aux;
            int pos=0;
            Random aleatorio=new Random();
            for(i=0;i<num;i++)
                vectors[i]=i+1;

        for(i=0;i<num;i++,j++){
                        d = aleatorio.nextInt(num-1);
                        aux=vectors[d];
                        vectors[d]=vectors[num-j];
                        vectors[num-j]=aux;
                             }

        vector=new Nodo2[num];
        for(i=0;i<num;i++){
                //pos=(30-pos)/2;
                vector[i]=new Nodo2(pos,0.0,vectors[i],color);
                      }
                 }

}
```

ShellE.java

```
/*
 * Clase para la opción con el algortimo de
 * shell y la visualización de estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo3;
import java.util.Random;
public class ShellE {
   Nodo3 vector[];

        public void shell(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;
```

```
        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{3cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                    String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);
      int size = vector.length;
      int i,j,d=0;

for(int k=0;k<size;k++){
        vector[k].setY(y);
        codigo=String.format("%cnode at (%.2f,%.2f) [star,star points=%d,star point
ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[k].getX(),vector[k].getY(),vector[k].getPicos(),colorL,colorR);
                        a.escribe(codigo);
}
y=y-2;


int incrmnt = size/2;
while (incrmnt > 0){
   for (i=incrmnt; i < size; i++){
     j = i;
     int temp = vector[i].getPicos();
     while ((j >= incrmnt) && (vector[j-incrmnt].getPicos() > temp)){
       vector[j].setPicos(vector[j-incrmnt].getPicos());
      j = j - incrmnt;
        d=1;
     }
     vector[j].setPicos(temp);
        if(d==1){
                contador++;
                for(int k=0;k<size;k++){
                        vector[k].setY(y);
                        codigo=String.format("%cnode at (%.2f,%.2f) [star,star
points=%d,star point ratio=0.2cm,star point height=0.25cm,draw=%s,fill=%s]
{};",92,vector[k].getX(),vector[k].getY(),vector[k].getPicos(),colorL,colorR);
                        a.escribe(codigo);
                    }
            y=y-2;
```

```
            if (contador==8){
                  a.escribe(newpage);
                  contador=0;
                        y=0.0;
                  }
                  d=0;
         }
   }
   incrmnt /= 2;
}

         a.escribe(pie);
      a.cerrarArchivo();
         }//fin del metodo

public void creaVector( int num ){
         int vectors[] = new int[num];
         int d, j, i, aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for( i = 0, j = 3; i < num; i++, j++ )
         vectors[i]=j;

      for( i = 0, j = 1; i < num; i++, j++ ){
         d = aleatorio.nextInt(num-1);
         aux=vectors[d];
            vectors[d]=vectors[num-j];
         vectors[num-j]=aux;
      }

         vector=new Nodo3[num];
         for( i = 0; i < num; i++, pos += 1.5 )
                  vector[i]=new Nodo3(pos,4.0,vectors[i]);
}
}
```

ShellEl.java

```
/*
 * Clase para la opción con el algortimo de
 * shell y la visualización de elementos
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
```

```
import Nodos.Nodo4;
import Nodos.NodoElementos;
import java.util.Random;

public class ShellE1 {
   Nodo4 vector[];
   NodoElementos elementos[];

        public void shell(String nombreArchivo,String colorL,String colorR)
        {
                String codigo,encabezado,pie;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0;

        encabezado=String.format("%cdocumentclass{article}\n%cusepackage{tikz}\n%cu
sepackage{anysize}\n%cusetikzlibrary{shapes,snakes}\n%cmarginsize{1cm}{1cm}{1cm}
{1cm}\n%cbegin{document}\n%cbegin{center}\n%cbegin{tikzpicture}\n",92,92,92,92,92,
92,92,92);
                a.escribe(encabezado);

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);

        pie=String.format("%cend{tikzpicture}\n%cend{center}\n%cend{document}",92,9
2,92);
int j,d=0;
int size = vector.length;
int incrmnt = size/2;
while (incrmnt > 0){
   for (int i=incrmnt; i < size; i++){
     j = i;
     int atomico = vector[i].getAtomico();
     String simbolo = vector[i].getSimbolo();
     while ((j >= incrmnt) && (vector[j-incrmnt].getAtomico() > atomico)){
         vector[j].setAtomico(vector[j-incrmnt].getAtomico());
       vector[j].setSimbolo(vector[j-incrmnt].getSimbolo());
       j = j - incrmnt;
         d=1;
     }
     vector[j].setAtomico(atomico);
     vector[j].setSimbolo(simbolo);
         if(d==1){
                 contador++;
                 for(int k=0;k<size;k++){
```

```
                        vector[k].setY(y);
                        codigo=String.format("%cnode at (%.2f,%.2f) [rectangle,minimum
height=1cm,minimum width=1cm,draw=%s,fill=%s]
{%s};",92,vector[k].getX(),vector[k].getY(),colorL,colorR,vector[k].getSimbolo());
                        a.escribe(codigo);
                    }
            y=y-2;
            if (contador==12){
                a.escribe(newpage);
                contador=0;
                        y=0.0;
                }
                d=0;
        }
    }
    incrmnt /= 2;
}


        a.escribe(pie);
      a.cerrarArchivo();
        }//fin del metodo

public void creaElementos(){
 elementos=new NodoElementos[118];
elementos[0]=new NodoElementos("H",1);
elementos[1]=new NodoElementos("He",2);
elementos[2]=new NodoElementos("Li",3);
elementos[3]=new NodoElementos("Be",4);
elementos[4]=new NodoElementos("B",5);
elementos[5]=new NodoElementos("C",6);
elementos[6]=new NodoElementos("N",7);
elementos[7]=new NodoElementos("O",8);
elementos[8]=new NodoElementos("F",9);
elementos[9]=new NodoElementos("Ne",10);
elementos[10]=new NodoElementos("Na",11);
elementos[11]=new NodoElementos("Mg",12);
elementos[12]=new NodoElementos("Al",13);
elementos[13]=new NodoElementos("Si",14);
elementos[14]=new NodoElementos("P",15);
elementos[15]=new NodoElementos("S",16);
elementos[16]=new NodoElementos("Cl",17);
elementos[17]=new NodoElementos("Ar",18);
elementos[18]=new NodoElementos("K",19);
elementos[19]=new NodoElementos("Ca",20);
elementos[20]=new NodoElementos("Sc",21);
elementos[21]=new NodoElementos("Ti",22);
```

```
elementos[22]=new NodoElementos("V",23);
elementos[23]=new NodoElementos("Cr",24);
elementos[24]=new NodoElementos("Mn",25);
elementos[25]=new NodoElementos("Fe",26);
elementos[26]=new NodoElementos("Co",27);
elementos[27]=new NodoElementos("Ni",28);
elementos[28]=new NodoElementos("Cu",29);
elementos[29]=new NodoElementos("Zn",30);
elementos[30]=new NodoElementos("Ga",31);
elementos[31]=new NodoElementos("Ge",32);
elementos[32]=new NodoElementos("As",33);
elementos[33]=new NodoElementos("Se",34);
elementos[34]=new NodoElementos("Br",35);
elementos[35]=new NodoElementos("Kr",36);
elementos[36]=new NodoElementos("Rb",37);
elementos[37]=new NodoElementos("Sr",38);
elementos[38]=new NodoElementos("Y",39);
elementos[39]=new NodoElementos("Zr",40);
elementos[40]=new NodoElementos("Nb",41);
elementos[41]=new NodoElementos("Mo",42);
elementos[42]=new NodoElementos("Tc",43);
elementos[43]=new NodoElementos("Ru",44);
elementos[44]=new NodoElementos("Rh",45);
elementos[45]=new NodoElementos("Pd",46);
elementos[46]=new NodoElementos("Ag",47);
elementos[47]=new NodoElementos("Cd",48);
elementos[48]=new NodoElementos("In",49);
elementos[49]=new NodoElementos("Sn",50);
elementos[50]=new NodoElementos("Sb",51);
elementos[51]=new NodoElementos("Te",52);
elementos[52]=new NodoElementos("I",53);
elementos[53]=new NodoElementos("Xe",54);
elementos[54]=new NodoElementos("Cs",55);
elementos[55]=new NodoElementos("Ba",56);
elementos[56]=new NodoElementos("La",57);
elementos[57]=new NodoElementos("Ce",58);
elementos[58]=new NodoElementos("Pr",59);
elementos[59]=new NodoElementos("Nd",60);
elementos[60]=new NodoElementos("Pm",61);
elementos[61]=new NodoElementos("Sm",62);
elementos[62]=new NodoElementos("Eu",63);
elementos[63]=new NodoElementos("Gd",64);
elementos[64]=new NodoElementos("Tb",65);
elementos[65]=new NodoElementos("Dy",66);
elementos[66]=new NodoElementos("Ho",67);
elementos[67]=new NodoElementos("Er",68);
elementos[68]=new NodoElementos("Tm",69);
```

```
elementos[69]=new NodoElementos("Yb",70);
elementos[70]=new NodoElementos("Lu",71);
elementos[71]=new NodoElementos("Hf",72);
elementos[72]=new NodoElementos("Ta",73);
elementos[73]=new NodoElementos("W",74);
elementos[74]=new NodoElementos("Re",75);
elementos[75]=new NodoElementos("Os",76);
elementos[76]=new NodoElementos("Ir",77);
elementos[77]=new NodoElementos("Pt",78);
elementos[78]=new NodoElementos("Au",79);
elementos[79]=new NodoElementos("Hg",80);
elementos[80]=new NodoElementos("Tl",81);
elementos[81]=new NodoElementos("Pb",82);
elementos[82]=new NodoElementos("Bi",83);
elementos[83]=new NodoElementos("Po",84);
elementos[84]=new NodoElementos("At",85);
elementos[85]=new NodoElementos("Rn",86);
elementos[86]=new NodoElementos("Fr",87);
elementos[87]=new NodoElementos("Ra",88);
elementos[88]=new NodoElementos("Ac",89);
elementos[89]=new NodoElementos("Th",90);
elementos[90]=new NodoElementos("Pa",91);
elementos[91]=new NodoElementos("U",92);
elementos[92]=new NodoElementos("Np",93);
elementos[93]=new NodoElementos("Pu",94);
elementos[94]=new NodoElementos("Am",95);
elementos[95]=new NodoElementos("Cm",96);
elementos[96]=new NodoElementos("Bk",97);
elementos[97]=new NodoElementos("Cf",98);
elementos[98]=new NodoElementos("Es",99);
elementos[99]=new NodoElementos("Fm",100);
elementos[100]=new NodoElementos("Md",101);
elementos[101]=new NodoElementos("No",102);
elementos[102]=new NodoElementos("Lr",103);
elementos[103]=new NodoElementos("Rf",104);
elementos[104]=new NodoElementos("Db",105);
elementos[105]=new NodoElementos("Sg",106);
elementos[106]=new NodoElementos("Bh",107);
elementos[107]=new NodoElementos("Hs",108);
elementos[108]=new NodoElementos("Mt",109);
elementos[109]=new NodoElementos("Ds",110);
elementos[110]=new NodoElementos("Uuu",111);
elementos[111]=new NodoElementos("Uub",112);
elementos[112]=new NodoElementos("Uut",113);
elementos[113]=new NodoElementos("Uuq",114);
elementos[114]=new NodoElementos("Uup",115);
elementos[115]=new NodoElementos("Uuh",116);
```

```
elementos[116]=new NodoElementos("Uus",117);
elementos[117]=new NodoElementos("Uuo",118);

}



public void creaVector( int num ){
          int d, j, i;
       double pos=0.0;
          creaElementos();
       Random aleatorio=new Random();
//       for( i = 0, j = 3; i < num; i++, j++ )
  //       vectors[i]=j;

          vector=new Nodo4[num];
       for( i = 0; i < num; i++, pos += 1.0){
          d = aleatorio.nextInt(118);
             vector[i]=new
Nodo4(pos,0.0,elementos[d].getAtomico(),elementos[d].getSimbolo());
          }
}
}
```

ShellF.java

```
/*
 * Clase para la opción con el algortimo de
 * shell y la visualización de estrellas
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;


public class ShellF {
    Nodo vector[];

          public void algoritmoShell(String nombreArchivo)
          {
                String codigo,encabezado,pie;
             CrearArchivo a =new CrearArchivo();
             a.abrirArchivo(nombreArchivo);
                int contador=0;
```

```
                double y=0.0;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}\n",92,92,92,92);
                a.escribe(encabezado);//Escribe el encabezado en el archivo

                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);

      int n= vector.length; //n será el tamaño del vector
      int intervalo,i,j,k;
      intervalo=n/2;
      while(intervalo>0){ //Comienza el algoritmo de Shell
      for(i=intervalo;i<n;i++){
            j=i-intervalo;
            while(j>=0)
            {
              k=j+intervalo;
               if(vector[j].getRadio()<=vector[k].getRadio()) // Se realiza las comparaciones
                 j=-1;
               else{
                  double temp;
                  contador++;// se cuentan las iteraciones que realiza el algoritmo
                  temp=vector[j].getRadio(); //Empieza el intercambio de las posiciones
                  vector[j].setRadio(vector[k].getRadio());
                  vector[k].setRadio(temp); //Termina el intercambio de las posiciones
                  j-=intervalo;
                   for(k=0;k<vector.length;k++){ //Este ciclo imprimirá el código por cada
iteración
                                    vector[k].setY(y);
                                    codigo=String.format("%cfill [%s!65] (%.2f,%.2f)
circle(%.2fcm);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getRa
dio());
                                    a.escribe(codigo);
                              }
                        y=y-2; // Se modifican las coordenadas en y
                        if (contador==9){//Si se cuentan 9 iteraciones en una página
                          a.escribe(newpage);//se requerirá de una nueva página
                          contador=0; // El contador se reiniciará en cero
                        y=0;          }// La coordenada en y se pone en cero
                  }


            }
```

```
            }
          intervalo=intervalo/2; //El intervalo se dividirá entre 2

        }
            a.escribe(pie); //Se escribe el pie en el archivo
          a.cerrarArchivo();
          }

          public void creaVector(int num,String color){ //Crea vector de números aleatorios
          int vectors[] = new int[num];                 // de tal manera que no se repitan
          int d,j=1,i,aux;
       double pos=0.0;
       Random aleatorio=new Random();
       for(i=0;i<num;i++)
          vectors[i]=i+1; // Asigna valores al vector

       for(i=0;i<num;i++,j++){ // Este ciclo se encarga de desordenar el vector
          d = aleatorio.nextInt(num-1);
             aux=vectors[d];
                   vectors[d]=vectors[num-j];
             vectors[num-j]=aux;
       }


          vector=new Nodo[num];
          for(i=0;i<num;i++,pos+=2.0)//Este ciclo se encargará de guardar las propiedades en
el vector
                   vector[i]=new Nodo(pos,0.0,color,vectors[i]*0.1);
          }
}
```

ShellL.java

```
/*
 * Clase para la opción con el algortimo de
 * shell y la visualización de líneas
 */

package Algoritmos;


import ManejoArchivos.CrearArchivo;
import Nodos.Nodo;
import java.util.Random;

public class ShellL
{
```

```
 private Nodo arreglo[];
 private int vector[];
 private double paso;


public void shell(int size)
 {
  int i, j, incrmnt,d=0,z;
  double aux,x;
  String coordenadas;
  Nodo temp;
  incrmnt = size/2;
  while (incrmnt > 0)
  {
    for (i=incrmnt; i < size; i++)
    {
     j = i;
     temp = arreglo[i];
     while ((j >= incrmnt) && (arreglo[j-incrmnt].getTrans() > temp.getTrans()))
     {
         //System.out.printf("coordenada %f\n",arreglo[j].getY());
         //aux = arreglo[j].getY();
         aux = temp.getY();
       arreglo[j] = arreglo[j - incrmnt];
         temp.setCoory(arreglo[j].getY());
         arreglo[j].setCoory(aux);
       j = j - incrmnt;
         d=1;
         //System.out.println("aux "+aux+" j "+j);
         //for(z=0;z<size;z++)
         //arreglo[z].imprime();
     }
     arreglo[j] = temp;
         if(d==1){

                        for(z=0;z<size;z++){
                                arreglo[z].sumaX(paso);
                                coordenadas = String.format("-- (%.2f,%.2f)
",arreglo[z].getX(),arreglo[z].getY());
                                arreglo[z].agregaCoordenadas(coordenadas);
                                //arreglo[z].imprime();
                        }
         //              System.out.printf("\n\n");
         }
         d=0;
    }
```

```
   incrmnt /= 2;
  }
  for(z=0;z<size;z++){
          arreglo[z].sumaX(paso);
          coordenadas = String.format("-- (%.2f,%.2f);",arreglo[z].getX(),arreglo[z].getY());
          arreglo[z].agregaCoordenadas(coordenadas);
  }
 }

public int intercambios(int size)
 {
  int i, j, incrmnt, temp,d=0,cuenta=0;

  incrmnt = size/2;
  while (incrmnt > 0)
  {
    for (i=incrmnt; i < size; i++)
    {
     j = i;
     temp = vector[i];
     while ((j >= incrmnt) && (vector[j-incrmnt] > temp))
     {
       vector[j] = vector[j - incrmnt];
      j = j - incrmnt;
          d=1;
     }
     vector[j] = temp;
          if(d==1){
                  cuenta++;
          }
          d=0;
    }
    incrmnt /= 2;
  }
  return cuenta+2;
 }

 public void estableceColor(String color,int n)
 {
          int i;
          double x,y;
          for(i=0;i<n;i++){
                  arreglo[i].agregaColor(color);

          }

 }
```

```
public void creaVector(int n)
{
	int m,k,i;
	double x=0.00,y=0.00;
	arreglo = new Nodo[n];
	m = (90 / n);
	k=m;
	m = 10;
	for(i=0;i<n;i++,m+=k)
		arreglo[i] = new Nodo(x,y,m,"draw [color = ");
}

public void iniciaCoordenadas(int n)
{
	double aux,x,y = 0.0;
	int i;
	aux = (14.5) / (n-1);
	y =aux;
	for(i=1;i<n;i++, y+=aux)
		arreglo[i].setCoory(y*-1.0);
	for(i=0;i<n;i++){
		x = arreglo[i].getX();
		y = arreglo[i].getY();
		//System.out.println(y);
		String coor = String.format("(%.2f,%.2f) -- ",x,y);
		arreglo[i].agregaCoordenadas(coor);
		arreglo[i].setCoorx(x+paso);
		x = arreglo[i].getX();
		String coor2 = String.format("(%.2f,%.2f) ",x,y);
		arreglo[i].agregaCoordenadas(coor2);
	}
}


public void creaArchivo(String nombre,int pto,int n)
{
	CrearArchivo archivo = new CrearArchivo();
	archivo.abrirArchivo(nombre);
	String encabezado =
String.format("%cdocumentclass[landscape,a5papper]{slides}\n%cusepackage{tikz}\n%cb
egin{document}\n%cbegin{tikzpicture}[line width=%dpt]\n",92,92,92,92,pto);
	String pie = String.format("%cend{tikzpicture}\n%cend{document}",92,92);
	archivo.escribe(encabezado);
	for(int i=0;i<n;i++){
		archivo.escribe(String.format("%c%s",92,arreglo[i].getCodigo()));
	}
```

```
        archivo.escribe(pie);
        archivo.cerrarArchivo();
}


public void ejecutaShell(String nombre,String color,int n)
{
        int i,aux,m;
        vector = new int[n];
        Random aleatorios = new Random();
        Nodo temp = new Nodo();
        m=n;
        for(i=0;i<n;i++,m--){
                aux = aleatorios.nextInt(m);
                temp = arreglo[m-1];
                arreglo[m-1] = arreglo[aux];
                vector[m-1] = arreglo[aux].getTrans();
                arreglo[aux]= temp;
        }
        int z = intercambios(n);
        paso = 24.0 / z;
        //for(i=0;i<n;i++)
        //      arreglo[i].imprime();
        //System.out.printf("\n\n");
        /*for(i=0;i<n;i++)
                System.out.printf("%d ",vector[i]);
        System.out.println("");
        for(i=0;i<n;i++)
                System.out.printf("%d ",vector[i]);
        System.out.println("");*/
        estableceColor(color,n);
        iniciaCoordenadas(n);
        //for(i=0;i<n;i++)
        //      arreglo[i].imprime();
//      System.out.println(z);
        shell(n);
        int p;
        if(n==4)
                p=7;
        else if(n==8)
                p=3;
        else if(n==16)
                p=3;
        else if(n==32)
                p=2;
        else p= 3;
        creaArchivo(nombre,p,n);
```

```
        //for(i=0;i<n;i++)
        //        arreglo[i].imprime();
 }


}
```

ShellP.java

```
/*
 * Clase para la opción con el algortimo de
 * shell y la visualización de pirámides
 */

package Algoritmos;

import ManejoArchivos.CrearArchivo;
import Nodos.Nodo2;
import java.util.Random;


public class ShellP {
   Nodo2 vector[];

        public void algoritmoShell(String nombreArchivo)
        {
                String codigo,encabezado,pie,rec;
            CrearArchivo a =new CrearArchivo();
            a.abrirArchivo(nombreArchivo);
                int contador=0;
                double y=0.0,x;

        encabezado=String.format("%cdocumentclass{slides}\n%cusepackage{tikz}\n%cbe
gin{document}\n%cbegin{tikzpicture}[rounded corners,ultra thick]\n",92,92,92,92);
                a.escribe(encabezado);// Escribe el encabezado en el archivo
                String
newpage=String.format("%cend{tikzpicture}\n%cnewpage\n%cbegin{tikzpicture}\n",92,9
2,92);
                pie=String.format("%cend{tikzpicture}\n%cend{document}",92,92);

        int n= vector.length; //n será el tamaño del vector
        int intervalo,i,j,k;
        intervalo=n/2;

        while(intervalo>0){ //empieza algoritmo de Shell
      for(i=intervalo;i<n;i++){
           j=i-intervalo;
```

```
            while(j>=0)
            {
              k=j+intervalo;
               if(vector[j].getLargo()<=vector[k].getLargo())
                 j=-1;
               else{
                  double temp;
                  contador++;//cuenta las iteraciones
                  temp=vector[j].getLargo(); //realiza el intercambio de posiciones
                  vector[j].setLargo(vector[k].getLargo());
                  vector[k].setLargo(temp);
                  j-=intervalo;
                   for(k=0;k<vector.length;k++){//Este ciclo imprime el código por cada
iteración del algoritmo
                                    vector[k].setY(y);
                           x=(vector.length-vector[k].getLargo())/2;//Se modifican coordenadas
en y
                                    vector[k].setX(x);
                                    codigo=String.format("%cshade[top color=%s] (%.2f,%.2f)
rectangle
+(%.1f,0.5);",92,vector[k].getColor(),vector[k].getX(),vector[k].getY(),vector[k].getLargo(
));
                                    a.escribe(codigo);//Escribe el código en el archivo
                              y=y-0.5;//modifica coordenadas en y

                           }
                              y=y-0.5;

                           rec=String.format("%cshade[top color=white] (0.0,%.2f) rectangle
+(10,0.5);",92,y);
                           a.escribe(rec);//Imprime un espacio vacio por cada iteración
                           y=y-0.5;      //modifica coordenadas en y

                          if (contador==2){//si se cuentan dos iteraciones por página
                            a.escribe(newpage);// se requerira una nueva página
                            contador=0; //El contador de iteraciones se reinicia en cero
                          y=0;     }//las coordenadas en y se reinicia en cero
                  }


             }

           }
       intervalo=intervalo/2;

       }
            a.escribe(pie);//Escribe el pie en el archivo
```

```
        a.cerrarArchivo();
                }

        public void creaVector(int num,String color){ //Se crea vector con números
aleatorios
        int vectors[] = new int[num];
        int d,j=1,i,aux;
      double pos=0.0;
      Random aleatorio=new Random();
      for(i=0;i<num;i++)      //Se asignan valores al vector
        vectors[i]=i+1;

      for(i=0;i<num;i++,j++){   //este ciclo se encarga de desordenar el vector
        d = aleatorio.nextInt(num-1);
           aux=vectors[d];
                vectors[d]=vectors[num-j];
           vectors[num-j]=aux;
      }


        vector=new Nodo2[num];
        for(i=0;i<num;i++,pos+=2.0)// Se guardaran las propiedades en el vector tipo Nodo
                vector[i]=new Nodo2(pos,0.0,vectors[i],color);
        }
}
```

Bibliografía:

[1] KNUTH, Donald. “El arte de programar ordenadores. Clasificación y Búsqueda”. Volumen 3. Ed. Reverté, S. A., 1987. pp. 776.

[2] JOYANES, Aguilar Luís. Zahonero Martínez Ignacio. “Programación en C. Metodología, algoritmos y estructuras de datos”. Segunda edición. Ed. Mc Graw Hill. 2005. pp. 719.

[3] SEDGEWICK, Robert. “Algoritmos en C++”. Primera Edición. Ed. Pearson Educación. 1955. pp. 726.

[4] García Arévalo Gustavo y Mares Martínez Juan Carlos, estudiantes de Licenciatura en Ingeniería en Computación. Trimestre: 2009 Invierno “Propuesta de Proyecto Terminal Implementación de una interfaz gráfica interactiva para algoritmos de ordenamiento y estructuras tipo árbol”, Universidad Autónoma Metropolitana Unidad Azcapotzalco.

[5] Sorting Algorithm Animations. Consultada el 21 de octubre de 2009. http://www.sorting-algorithms.com/

[6] Quicksort. Consultada el 21 de octubre del 2009. http://es.wikipedia.org/wiki/Quicksort

[7] TILL, Tantau. “The TikZ and PGF Packages”, Manual para versión 2.00. Institut Theoretische Informatik. http://sourceforge.net/projects/pgf. Consultada el 20 de octubre del 2009.

[8] *Examples TikZ.* Consultada el 20 de octubre de 2009. http://www.texample.net/tikz/examples/

[9] Debian. Consultada el 8 de noviembre del 2009. http://www.debian.org/index.es.html

[10] NetBeans. Consultada el 8 de noviembre del 2009. http://netbeans.org