Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Software para la organización del horario del alumno por medio de la coloración de grafos

Alumno: *Yáñez Castillo José Alberto*

Matrícula: 206300036

Asesora: *M. en C. Rafaela Blanca Silva López*

Número económico: 17114

Trimestre: Invierno del 2011

Índice

I. Introducción 4

II. Objetivo general 4

III. Objetivos particulares 4

IV. Descripción del problema 5

V. Diseño 6

1. Definición del requerimiento del usuario. 6

2. Requerimientos funcionales. 6

3. Modelo de casos de uso. 7

4. Identificación de actores. 7

5. Documentación de actores. 7

6. Casos de uso. 7

7. Documentación de casos de uso. 8

7.1 Caso de uso 1: Ingresar materias.- 8

7.2 Caso de uso 2: Corregir materias ingresadas.- 10

7.3 Caso de uso 3: Generar propuesta de horario.- 11

7.4 Caso de uso 4: Mostar horario.- 12

7.5 Caso de uso 5: Desplegar en pantalla.- 13

7.6 Caso de uso 6: Guardar en archivo.- 13

8. Manejo de clases. 13

VI. Análisis de clases 15

1. Diagrama de clases. 15

2. Diccionario de clases. 15

VII. Diseño 16

VIII. Clases finales 19

1. Dependencias de paquetes para la aplicación de Selección 19

2. Dependencia de paquetes para la aplicación de organización 19

3. Clases para la aplicación Selecciona 20

4. Clases de la aplicación Organiza 21

IX. Desarrollo 24

1. Descripción 24

2. Selección de materias 24

3. Implementación de la interfaz grafica para el componente selección de materias..... 25
4. Organización de horarios 25
Entrada de datos 25
5. Ordenar por día y hora..... 27
6. Crear la propuesta de horario 27
7. Mostrar los resultados 28
8. Interfaz gráfica..... 28
X. Documentación código fuente..... 29
XI. Elaboración del manual de usuario 29
XII. Conclusiones 29
Anexo A. Manual de usuario 31
Ejecución de selecciona.jar desde Windows 35
Bibliografía 51

Índice de Figuras

Figura 1. Diagrama de casos de uso del SOHA-CG 8
Figura 2 Ingresar materias 9
Figura 3 Corregir materias 10
Figura 4 Generar propuesta 11
Figura 5 Mostar horario 12
Figura 6 Diagrama de clases 15
Figura 7 Diagrama de secuencia del caso de uso: Ingresar materias 16
Figura 8 Diagrama de secuencia del caso de uso: Corregir materias ingresadas 17
Figura 9 Diagrama de secuencia del caso de uso: Generar propuesta de horario 17
Figura 10 Diagrama de secuencia del caso de uso: Mostar horario 18
Figura 11 Diagrama de secuencia del caso de uso: Desplegar en pantalla y Guardar en archivo 18
Figura 12 Dependencia de paquetes 19
Figura 13 Dependencia de paquetes 19
Figura 14 Clases par Selecciona 20
Figura 15 Interfaz Selecciona 20
Figura 16 Dependencias entre clases de Selecciona 21
Figura 17 Clases de organiza 21
Figura 18 Interfaz Organiza 22
Figura 19 Dependencia de clases en Organiza 23
Figura 20 Pantalla principal de Selección de materias 25
Figura 21 Ubicación de la clase Manejo_Archivo_Horarios.java 26
Figura 22 Ubicación de la clase Oredena_dia_hora.java 27
Figura 23 Ubicación de la clase Propuesta_horario.java 28
Figura 24 Ubicación de la clase Imprime_horario.java 28

I. Introducción

El propósito de este documento es el de proporcionar una descripción general del trabajo realizado en el "Software para la organización de horario del alumno por medio de la coloración de grafos", así como describir algunas funciones del software y el uso que pudiera darle el usuario a este.

El SOHA-CG[1], es un proyecto que funcionará de forma independiente, es decir, no es parte de ningún sistema mayor y no depende de ningún otro proyecto.

Con el SOHA-CG se pretende facilitarle al alumno la selección de materias a cursar y su organización dentro de un horario para la próxima reinscripción de materias.

La función básica de este software es la de seleccionar y ordenar las materias que un alumno pudiera cursar durante el trimestre.

II. Objetivo general

Realizar un software que genere el horario escolar de un alumno por medio de un algoritmo secuencial de coloración de grafos.

III. Objetivos particulares

1. Diseñar el software para la organización de horario del alumno por medio de la coloración de grafos.
    a. Diseñar y modelar el sistema con base en el modelo UML[2].
2. Implementar el algoritmo secuencial de la coloración de grafos.
3. Implementar el sistema para la organización de horarios del alumno por medio de la coloración de grafos en lenguaje de programación JAVA.
4. Realizar pruebas al sistema, asegurando que el producto final funcione adecuadamente.
5. Documentar el sistema realizado.
6. Construir un manual de usuario para el proyecto realizado.

[1] SOHA-CG son las iníciales de Software para la Organización de Horario del Alumno por medio de la Coloración de Grafos

[2] Unified Modeling Language (Lenguaje Unificado de Modelado): Lenguaje gráfico para especificar, construir y documentar un sistema.

IV. Descripción del problema

Dentro de la UAM-A[3] en la División de Ciencias Básicas e Ingeniería existen diez licenciaturas, cada una cuenta con su plan de estudios. Por ser una lista grande de licenciaturas el software a realizar solo contemplará el plan de estudios para los alumnos de la Licenciatura de Ingeniería en Computación.

Cada trimestre en la UAM-A se sigue un procedimiento para que los alumnos reinscriban las materias que cursarán. Los pasos que debe seguir el alumno son:

- Ingresar a su kardex, esto lo hace por medio de la página web del modulo de información escolar de alumnos de licenciatura.
- El alumno consulta la programación de UEA[4]. Revisa los horarios publicados para las UEA que él desea cursar el próximo trimestre.
- El alumno selecciona las materias a cursar, las ingresa en una pre solicitud que se encuentra dentro de la misma página web.
- Se programan días para que los alumnos inscriban las materias que capturaron en su pre solicitud.

En cuanto a los pasos anteriores no existen problemas. Es una forma sencilla de llevar un control sobre la demanda de UEA. El alumno puede asegurar un lugar dentro de los grupos en que se impartirán dichas UEA si realizo una buena organización de su horario.

Las materias que selecciono el alumno tiene ciertas propiedades como: grupo, cupo y el horario en que se imparten. El horario es una propiedad importante ya que los horarios de distintas materias se pueden empalmar ocasionando un problema para el alumno. Para entender el problema pongamos un ejemplo: el alumno desea tomar la materia A y la materia B, pero ambas materias son impartidas a la misma hora. Es entonces en este punto donde surge el empalme y el problema de organizar el horario. Aparte se deben considerar situaciones como:

- No se debe sobrepasar el límite de créditos permitidos en el trimestre
- Si el alumno es de tiempo completo, debe inscribir un mínimo de créditos
- La seriación de ciertas materias debe estar cubierta para poder cursar otras materias

[3] UAM-A: Universidad Autónoma Metropolitana unidad Azcapotzalco
[4] UEA: Unidad de Enseñanza Aprendizaje, así se le nombra a las materias dentro de la UAM-A

V. Diseño

1. Definición del requerimiento del usuario.

El SOHA-CG debe elaborar automáticamente una propuesta del horario de un alumno en base al plan de estudio y las materias aprobadas por el alumno.

2. Requerimientos funcionales.

- Es deseado que el software pueda ejecutarse desde un sistema operativo Windows o Linux.
- La seriación de las materias para la carrera de ingeniería en computación debe ser conocida por el SOHA-CG.
- Al ser ejecutado el software este requiere saber que materias han sido aprobadas por el alumno.
- La forma en que el usuario ingresan las materias (datos de entrada para el software) puede ser: introduciendo las claves de cada materia una por una o bien ingresar todo el conjunto de materias desde un archivo de texto plano, este deberá tener un formato específico.
- Una vez que se validaron las materias aprobadas por el alumno, el software seleccionará las materias que el alumno puede cursar.
- El formato que deberá tener el horario (datos de salida que el software entrega) debe ser interpretado fácilmente por el usuario.

Hardware:

- Se pretende que el hardware necesario para ejecutar la aplicación sea mínimo.

Software:

- Para ejecutar la aplicación el usuario debe tener instalada la maquina virtual de JAVA.

Eficiencia:

- Es posible que el sistema no encuentre el horario más adecuado, debido a la poca oferta de materias ofrecidas durante el trimestre en curso.
- El tiempo para que el software obtenga una solución adecuada es desconocido.

Documentación:

El software debe tener un manual de usuario con los siguientes puntos:

- Ejecutar la aplicación en Windows o Linux
- La forma en que se deben ingresar los datos de entrada.
- Como interpretar la salida y los avisos de errores que se muestren

3. Modelo de casos de uso.

En la primera etapa del proyecto estos son los casos de uso y documentación de casos de uso contemplados.

4. Identificación de actores.

Un actor es un usuario final que interactúa con el SOHA-CG. Un usuario final es un alumno de la carrera de Ingeniería en Computación. Ver tabla 1.

5. Documentación de actores.

Actor	**Alumno/Usuario**
Casos de uso	Generar propuesta de horario, Ingresar materias, Corregir materias ingresadas, Mostrar horario, Desplegar en pantalla o Guardar en archivo.
Tipo	Primario/principal
Descripción	Es un estudiante de la carrera de ingeniería en computación de la UAM que desea una propuesta de horario. Es el encargado de iniciar el sistema, introducir las materias y también recibe la salida del sistema.

Tabla 1. Actor Alumno/Usuario

6. Casos de uso.

Los casos de uso sirven para representar las necesidades de los usuarios y requerimientos. En la figura 1 se muestran los posibles casos de uso para el sistema.

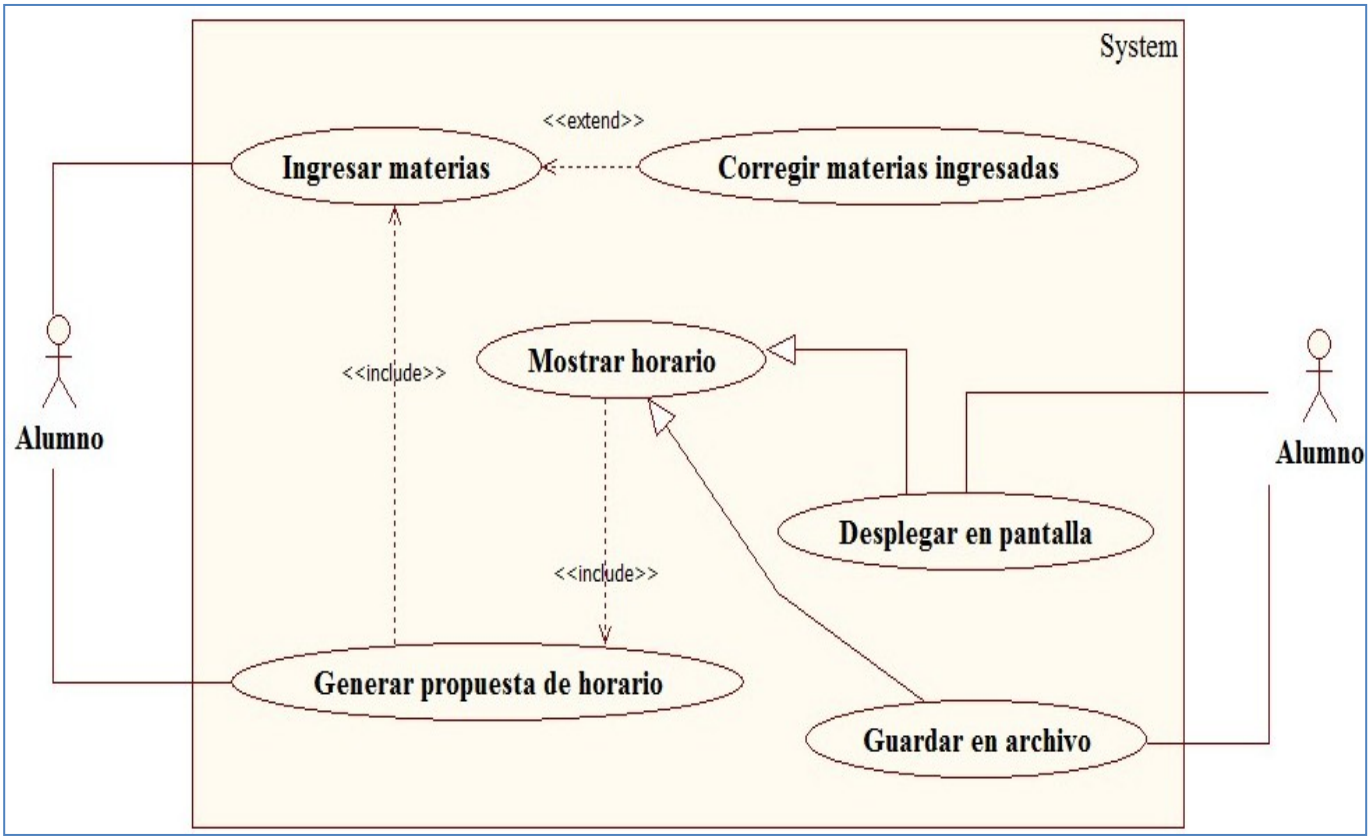

Figura 1. Diagrama de casos de uso del SOHA-CG

7. *Documentación de casos de uso.*

Se presentan las tablas de los distintos casos de uso representados en la figua1 y el diagrama de actividad correspondiente a algunos casos de uso.

7.1 Caso de uso 1: Ingresar materias.-

Caso de uso 1	**Ingresar materias**
Actores	Alumno
Tipo	Inclusión
Resumen	El primer paso para que el sistema genere una propuesta de horario es que el actor ingrese las materias aprobadas.
Pre condiciones	El usuario inicio el sistema.
Post condiciones	El usuario ingreso al sistema las materias que tiene aprobadas. No hay error en estás.
Flujo	Se inicia el sistema y se le pide al usuario que ingrese las materias que

principal	tiene cursadas y aprobadas hasta el momento. Si el usuario desea corregir alguna materia que introdujo mal puede elegir "Corregir materias ingresadas" (S-1). Si se llevo con éxito el usuario elige "Generar propuesta de horario" (S-2).
Subflujos	S-1: El sistema le permite al usuario corregir las materias que ingreso erróneamente al sistema ejecutando el caso de uso "Corregir materias ingresadas". S-2: Se valida que las materias introducidas por el usuario se encuentran dentro del plan de estudios (E-1) y se ejecuta el caso de uso "Generar propuesta de horario".
Excepciones	E-1: si las materias ingresadas por el usuario no existen en el plan de estudios se le pide al usuario que ingrese nuevamente las materias.

Tabla 2. Caso de uso 1, Ingresar materias

Diagrama de actividad para el caso de uso: Ingresar materias. Figura 2

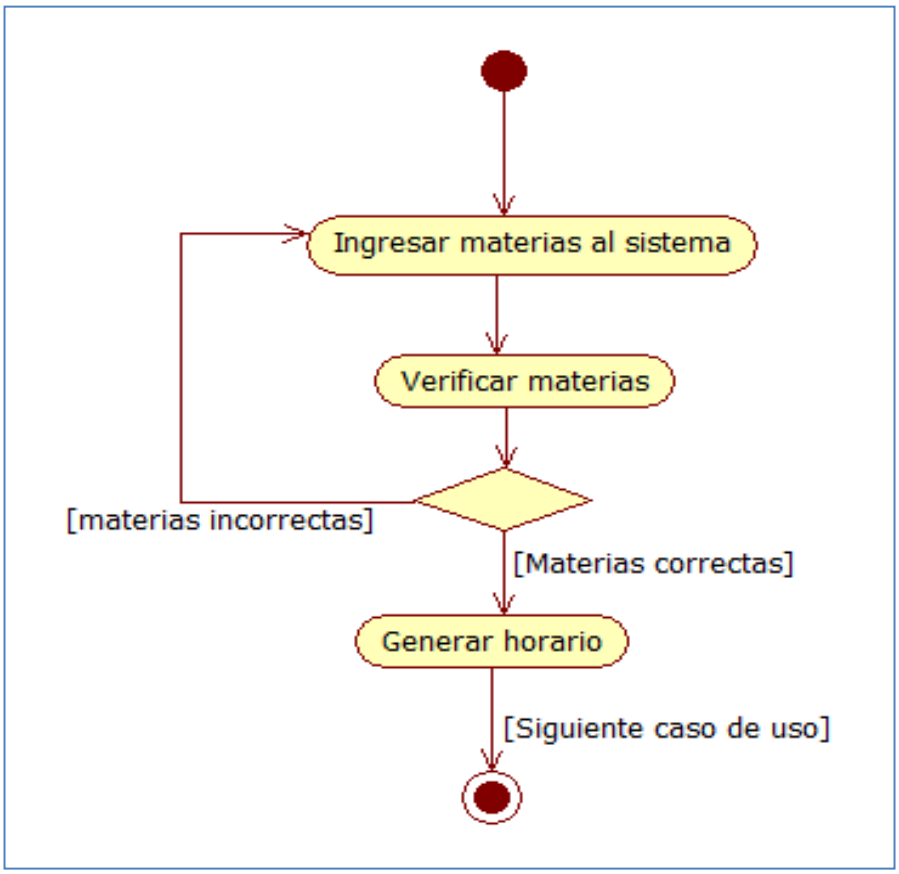

Figura 2 Ingresar materias

7.2 Caso de uso 2: Corregir materias ingresadas.-

Caso de uso 2	Corregir materias ingresadas
Actores	Alumno
Tipo	Extendido
Resumen	Si se requiere corregir las materias ingresadas al sistema o hay un error en ellas se procede a la corrección de las materias.
Pre condiciones	El actor ingreso las materias que tiene aprobadas.
Post condiciones	Se debe generar la propuesta de horario pues las materias ya fueron corregidas.
Flujo principal	El usuario ha decidido corregir las materias o bien el sistema le ha pedido que corrija las materias introducidas por lo tanto se vuelven a "Ingresar las materias" (S-1).
Subflujos	S-1: Se ejecuta nuevamente el caso de uso "Ingresar materias"
Excepciones	Después de ejecutarse este caso de uso es poco probable que existan errores.

Tabla 3. Caso de uso3, corregir materias ingresadas

Diagrama de actividad para el caso de uso: Corregir materias ingresadas Figura 3

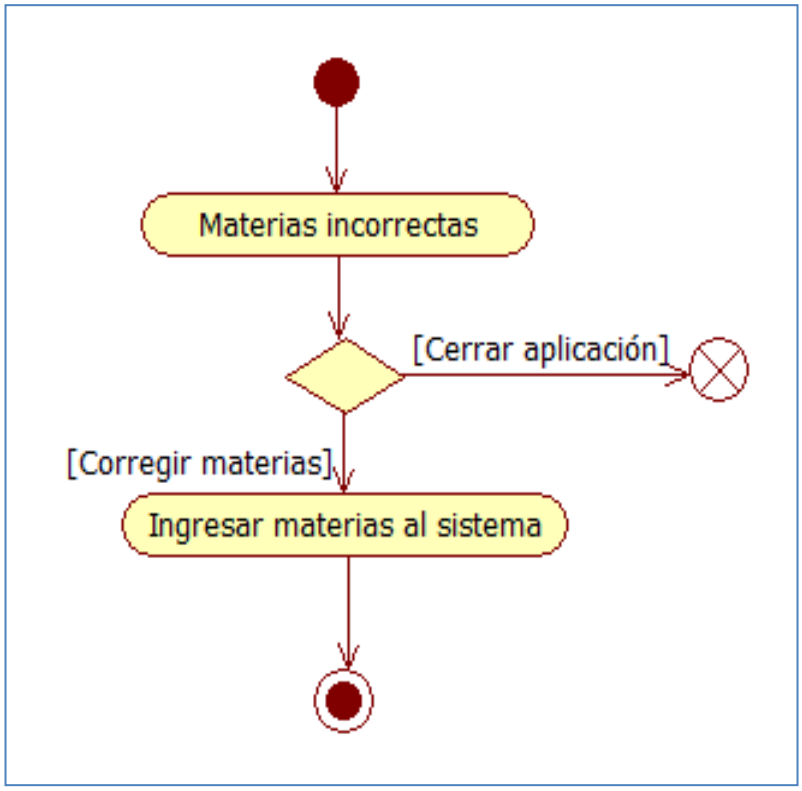

Figura 3 Corregir materias

7.3 Caso de uso 3: Generar propuesta de horario.-

Caso de uso 3	Generar propuesta de horario
Actores	Alumno
Tipo	Básico/Inclusión
Resumen	Al iniciar este caso de uso el sistema está listo para obtener una posible solución al horario del alumno.
Pre condiciones	El actor ingreso las materias que tiene aprobadas, correctamente.
Post condiciones	El sistema realiza una propuesta de horario y el usuario puede visualizar el horario.
Flujo principal	El sistema inicia la selección de materias que el alumno puede cursar, por medio de un algoritmo se realiza la propuesta de horario. Si el sistema encuentra la solución se le presenta al usuario la opción "Mostar Horario"(S-1), de lo contrario se presenta un mensajes (S-2).
Subflujos	S-1: Se ha encontrado una propuesta de horario para el alumno, se procede a ejecutar el caso de uso "Mostrar horario". S-2: No se ha podido encontrar la mejor propuesta de horario (E-1) y se le avisa por medio de un mensaje al usuario.
Excepciones	E-1: Las razones pueden ser diversas.

Tabla 4. Caso de uso 3, Generar propuesta de horario

Diagrama de actividad para el caso de uso: Generar propuesta de horario Figura 4

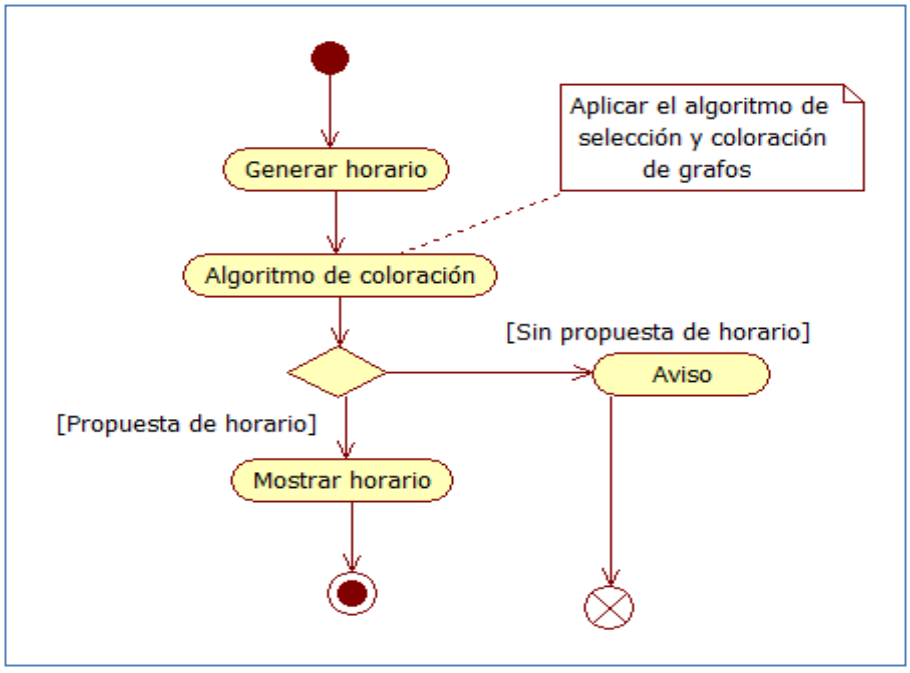

Figura 4 Generar propuesta

7.4 Caso de uso 4: Mostar horario.-

Caso de uso 4	Mostar horario
Actores	Alumno
Tipo	Básico
Resumen	El sistema le muestra al usuario el horario con los resultados obtenidos.
Pre condiciones	El algoritmo aplicado encontró una solución satisfactoria al problema.
Post condiciones	El usuario obtiene una propuesta de horario.
Flujo principal	Se le muestra al usuario dos opciones, la primera "Desplegar en pantalla" (S-1) y la segunda "Guardar en archivo" (S-2).
Subflujos	S-1: Si el usuario elige la opción "Desplegar en pantalla" se ejecuta este caso de uso. S-2: El usuario ha elegido la segunda opción "Desplegar en pantalla" por lo tanto se ejecuta este caso de uso.
Excepciones	

Tabla 5. Caso de uso 4, Mostrar horario

Diagrama de actividad para el caso de uso: Mostar horario Figura 5

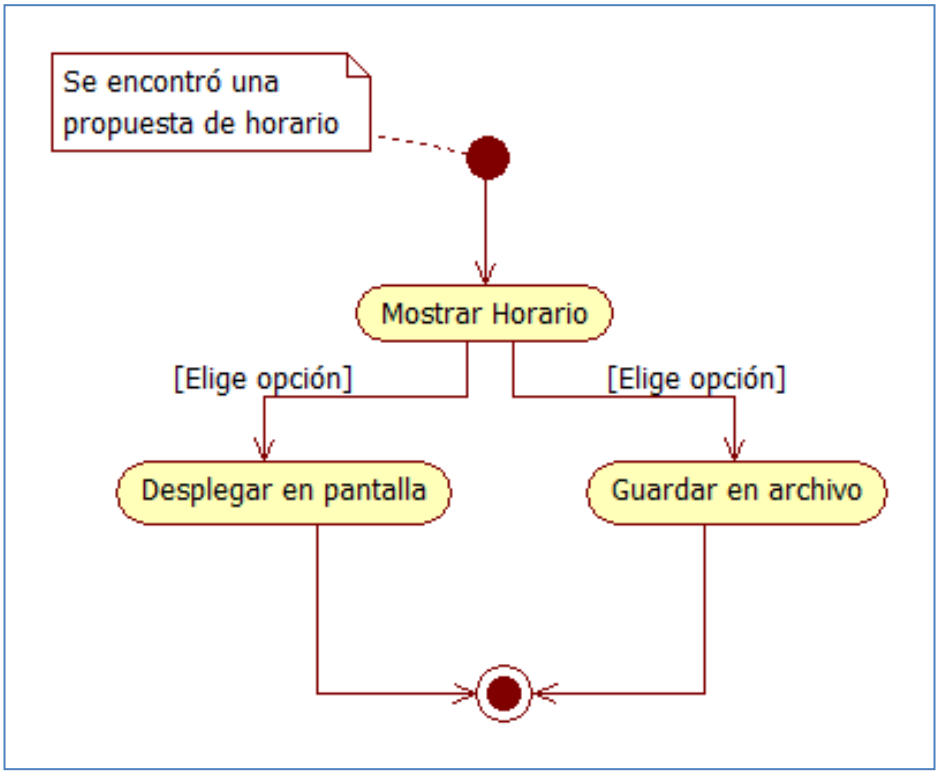

Figura 5 Mostar horario

Los últimos dos casos de uso no se les crea un diagrama de actividad puesto que no se consideran necesarios.

7.5 Caso de uso 5: Desplegar en pantalla.-

Caso de uso 5	**Desplegar en pantalla**
Actores	Alumno
Tipo	Generalización
Resumen	El sistema le muestra al usuario el horario con los resultados obtenidos.
Pre condiciones	El usuario eligió la opción "Mostar horario" mediante el despliegue en pantalla.
Post condiciones	El usuario puede ver en pantalla la propuesta de horario.
Flujo principal	Este es el último de los casos de uso, el sistema puede ser cerrado por el usuario.

Tabla 6. Caso de uso 5, Desplegar en pantalla

7.6 Caso de uso 6: Guardar en archivo.-

Caso de uso 6	**Guardar en archivo**
Actores	Alumno
Tipo	Generalización
Resumen	El sistema guarda el horario en un archivo con los resultados obtenidos.
Pre condiciones	El usuario eligió la opción "Mostar horario" y desea que se guarde en un archivo para su posterior consulta.
Post condiciones	El usuario obtiene un archivo con la propuesta de horario generada.
Flujo principal	Este es el último de los casos de uso, el sistema puede ser cerrado por el usuario.

Tabla 7. Caso de uso 6, Guardar en archivo

8. Manejo de clases.

Se seleccionan los sustantivos que se encuentran en la redacción "Descripción del problema de organización del horario". Los sustantivos son los posibles candidatos a transformarse en clases. Todos son considerados en primer instancia como clases candidatas. Ver tabla 8.

Clases Candidatas			
Alumno Créditos Cupo Grupo	Horario Ingeniería Ing. Computación Tiempo completo	Licenciatura Materias Mínimo de créditos Plan de estudios	Presolicitud Trimestre Seriación UEA

Tabla 8 Clases candidatas

De las clases candidatas se seleccionan solo las clases que se consideran importantes para implementar en el sistema. Ver tabla 9:

Clases elegidas
Alumnos **Horario** **Plan de estudios** **UEA**

Tabla 9 Clases elegidas

Entre los atributos se consideran:

Atributos	
Créditos Cupo Grupo Materias	Mínimo de créditos Trimestre Seriación

Tabla 10 Atributos

Identificación de atributos:

Alumno: nombre(s), apellido paterno, apellido materno, matricula, créditos a cursar, créditos cursados.

Horario: numero de materias a cursar, número de créditos inscritos, trimestre, materias inscritas.

Plan de estudios: materias del plan de estudio, materias cursadas, materias no cursadas, créditos totales del plan de estudio, créditos cursados por el alumno.

UEA: nombre, clave, créditos, horario, seriación.

Algunos atributos que no aparecían en la tabla 10 fueron agregados en la identificación de atributos. La razón es que, se consideran de utilidad para las clases candidatas.

VI. Análisis de clases

1. Diagrama de clases.

Se presenta un diagrama con las clases seleccionadas y los atributos. Se han agregado algunos atributos que se consideran importantes para la implementación del sistema. Algunas clases pueden eliminarse o renombrarse más adelante. También pueden agregarse clases que sean necesarias para el diseño del sistema. Ver figura 6.

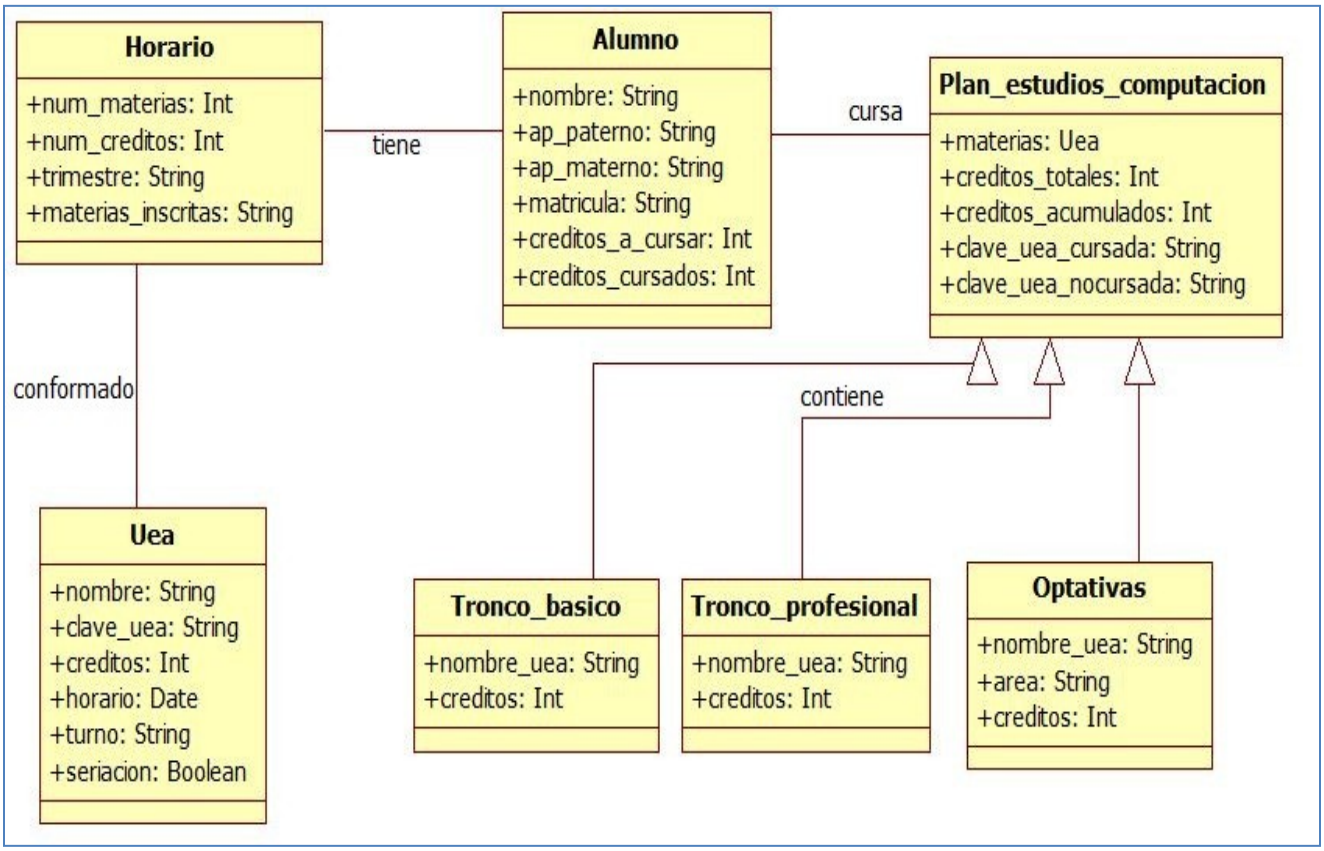

Figura 6 Diagrama de clases

2. Diccionario de clases.

Alumno: guarda los datos personales del alumno, también los datos académicos como son matricula, créditos que lleva cursados y los restantes para terminar el plan de estudios.

Horario: contiene los datos correspondientes como: nombre, numero y créditos que suman las UEA que el alumno inscribe.

Uea: cada materia tiene datos importantes como: nombre, clave, horario, créditos que aporta y su seriación. Todos esos datos son de gran utilidad para planear un horario.

Plan de estudios computación: es necesario conocer las materias que están dentro del plan de estudios de la carrera de computación junto con los créditos totales que se requieren para completar el plan de estudios, está clase utiliza un tipo de dato Uea porque se necesitan saber las materias que están dentro del plan de estudios.

Tronco básico, Tronco profesional y Optativas: estas tres últimas son subclases útiles para dividir plan de estudios, de esta manera será más fácil administrar los créditos totales, cursados y no cursados del plan de estudio.

VII. Diseño

Para realizar el diseño de los casos de uso nos auxiliaremos de diagramas de secuencia, estos son útiles para modelar la interacción entre los casos de uso y las clases propuestas. De esta forma podremos saber si a través de las clases propuestas se pueden crear objetos que ayuden a construir los casos de uso.

Diagrama de secuencia, caso de uso 1: Ingresar materias

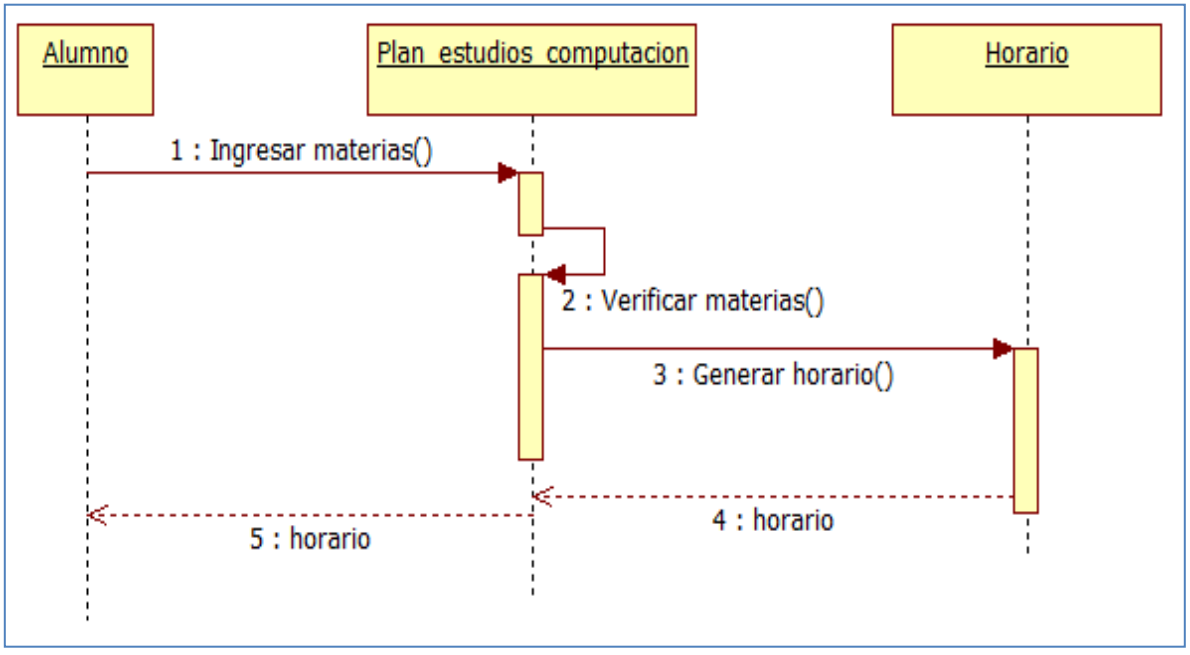

Figura 7 Diagrama de secuencia del caso de uso: Ingresar materias

Diagrama de secuencia, caso de uso 2: Corregir materias ingresadas

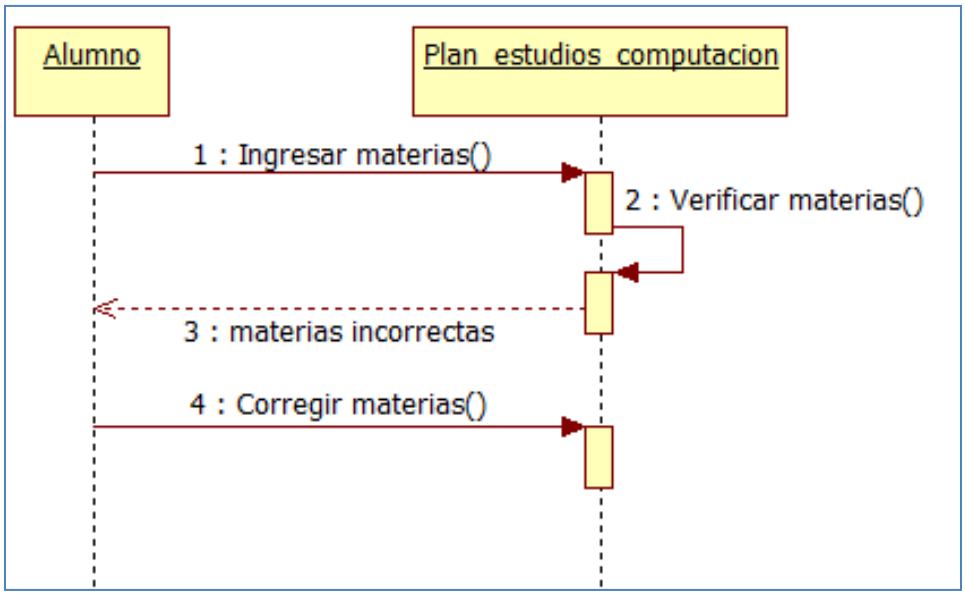

Figura 8 Diagrama de secuencia del caso de uso: Corregir materias ingresadas

Diagrama de secuencia, caso de uso 3: Generar propuesta de horario

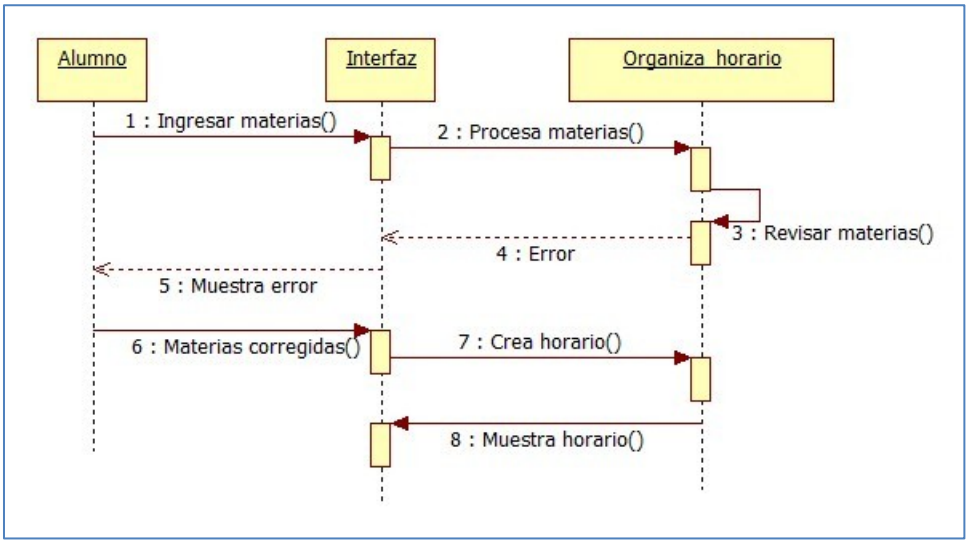

Figura 9 Diagrama de secuencia del caso de uso: Generar propuesta de horario

Diagrama de secuencia, caso de uso 4: Mostar horario

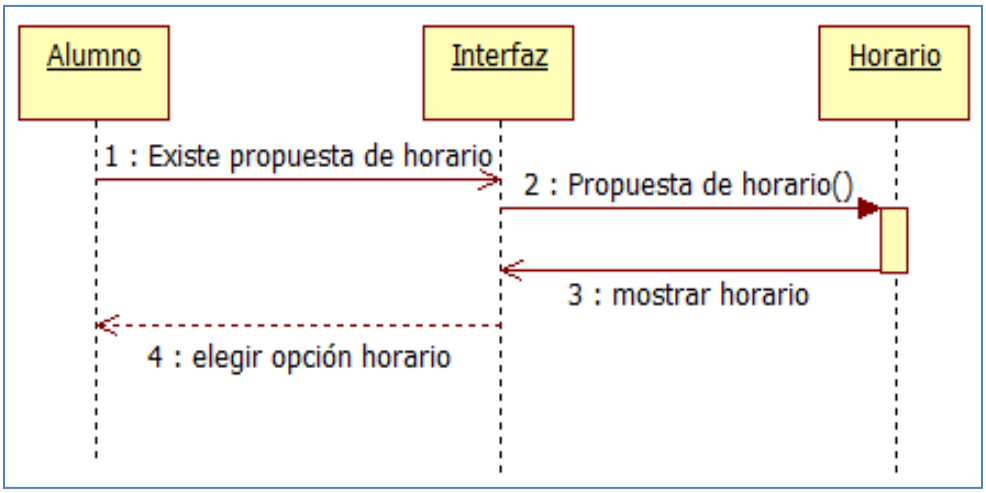

Figura 10 Diagrama de secuencia del caso de uso: Mostar horario

Diagrama de secuencia, caso de uso 5 y 6: Desplegar en pantalla y Guardar en archivo

El siguiente diagrama de secuencia maneja ambos casos de uso pues solo difieren en un paso y no se considera necesario repetir otro diagrama.

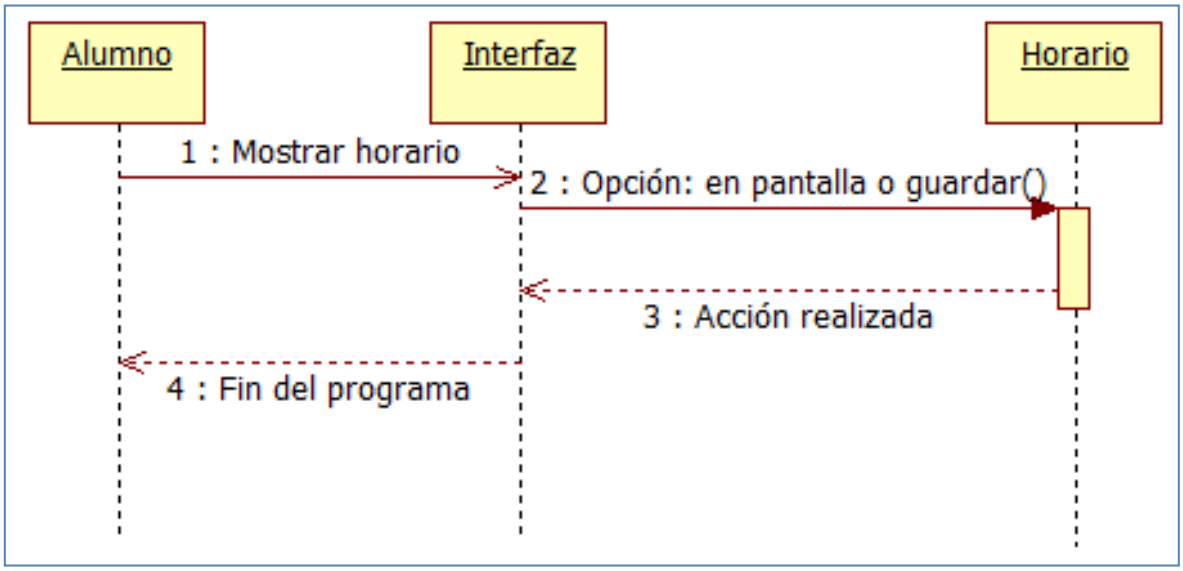

Figura 11 Diagrama de secuencia del caso de uso: Desplegar en pantalla y Guardar en archivo

VIII. Clases finales

Como se menciono anteriormente algunas de las clases podrían eliminarse o cambiarse. Después de programar las aplicaciones se ha aplicado ingeniería inversa para obtener las clases que conforman al sistema. Así como las dependencias de los paquetes.

1. *Dependencias de paquetes para la aplicación de Selección*

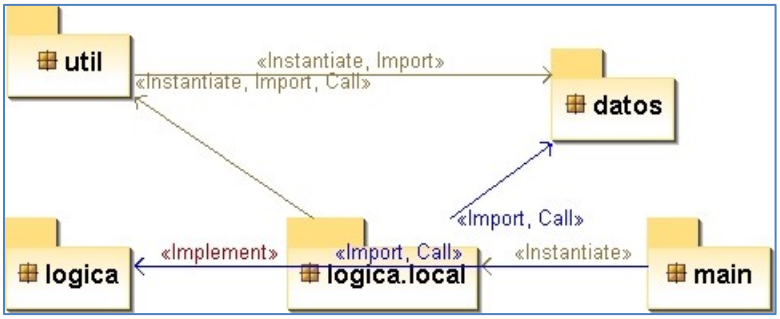

Figura 12 Dependencia de paquetes

2. *Dependencia de paquetes para la aplicación de organización*

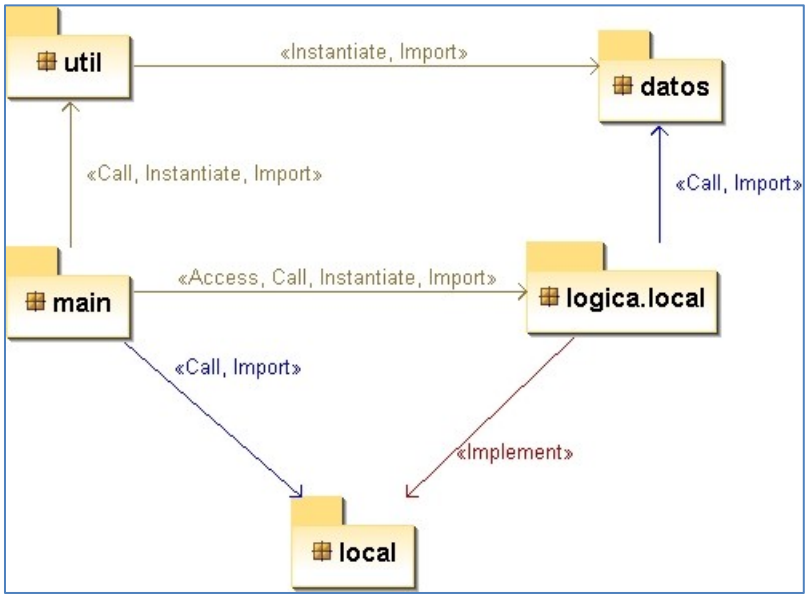

Figura 13 Dependencia de paquetes

3. *Clases para la aplicación Selecciona*

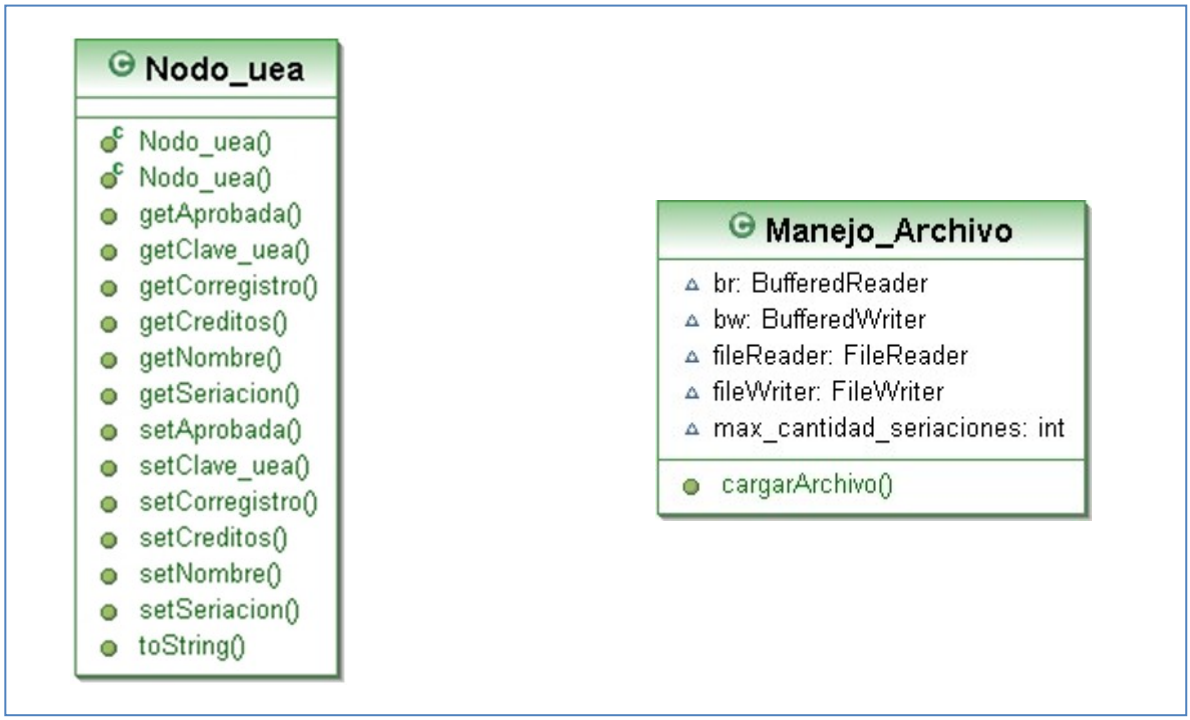

Figura 14 Clases par Selecciona

Figura 15 Interfaz Selecciona

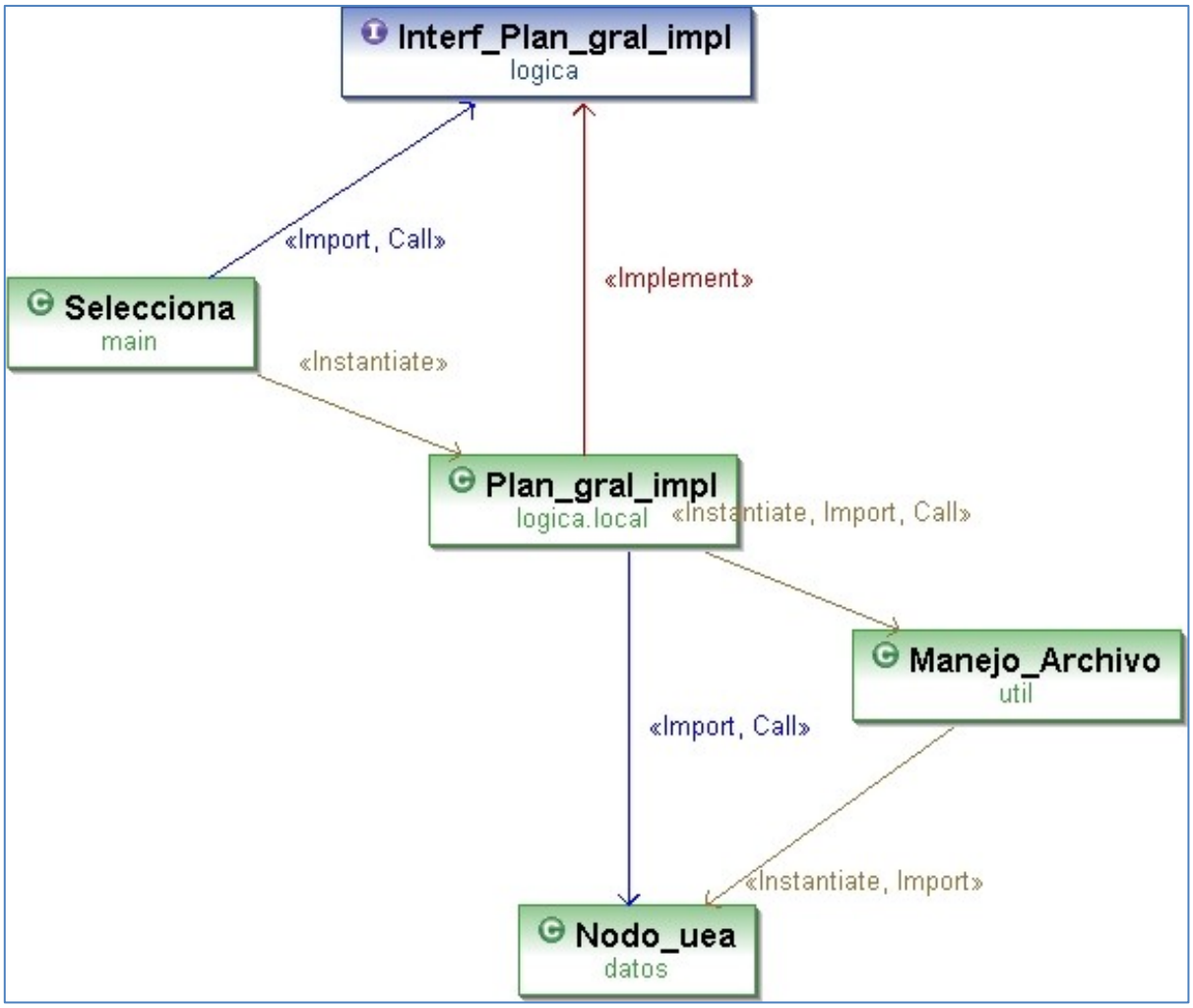

Figura 16 Dependencias entre clases de Selecciona

4. ***Clases de la aplicación Organiza***

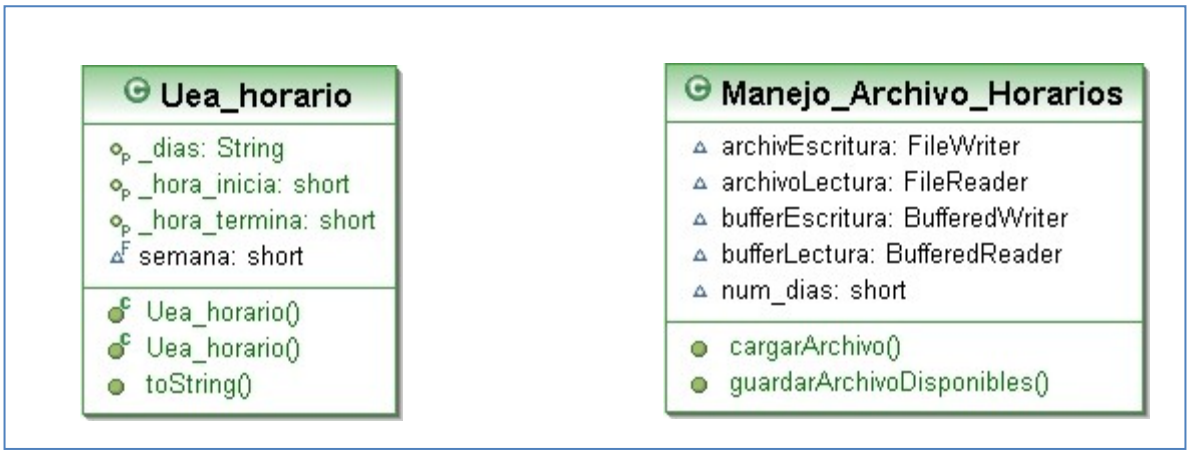

Figura 17 Clases de organiza

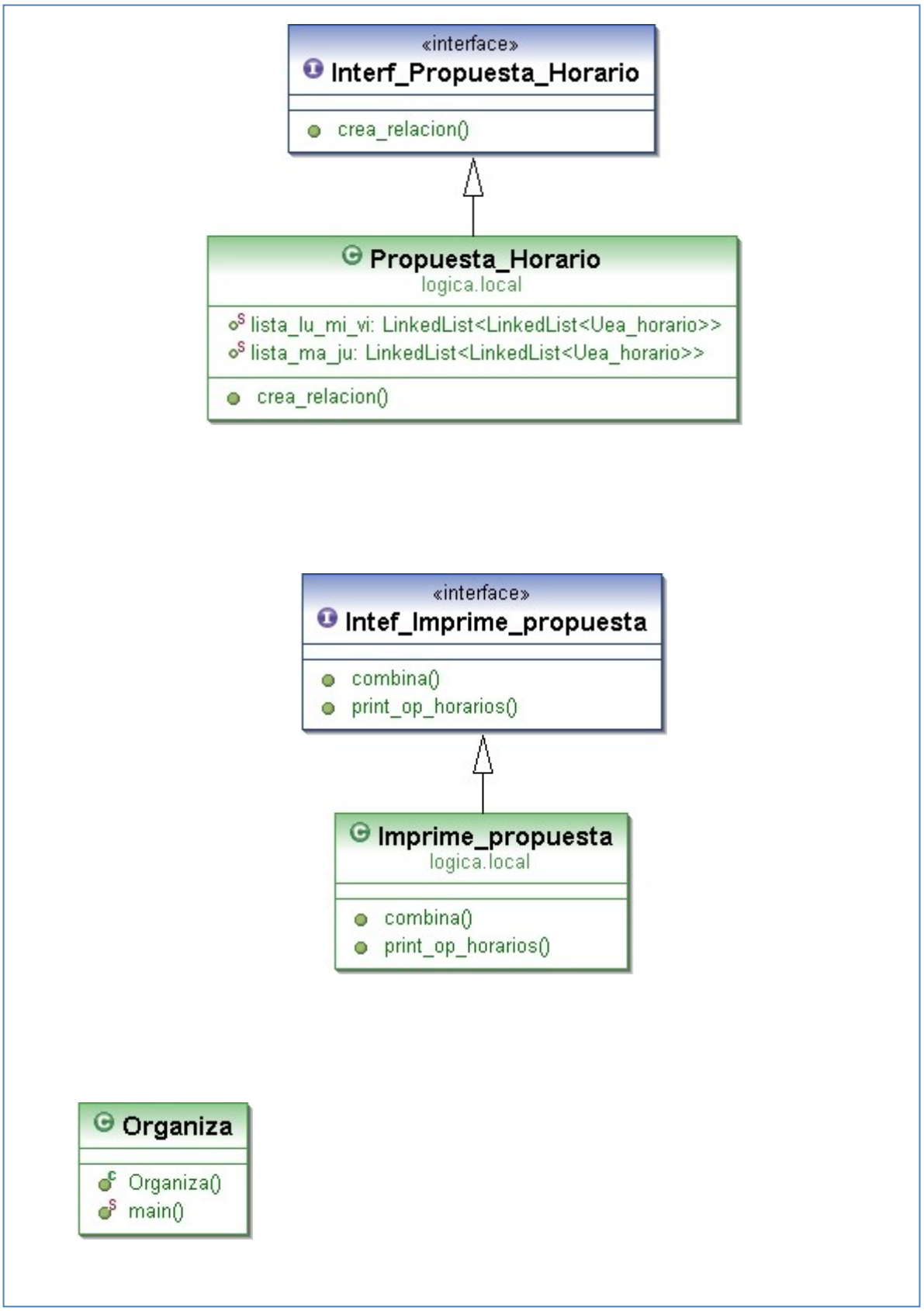

Figura 18 Interfaz Organiza

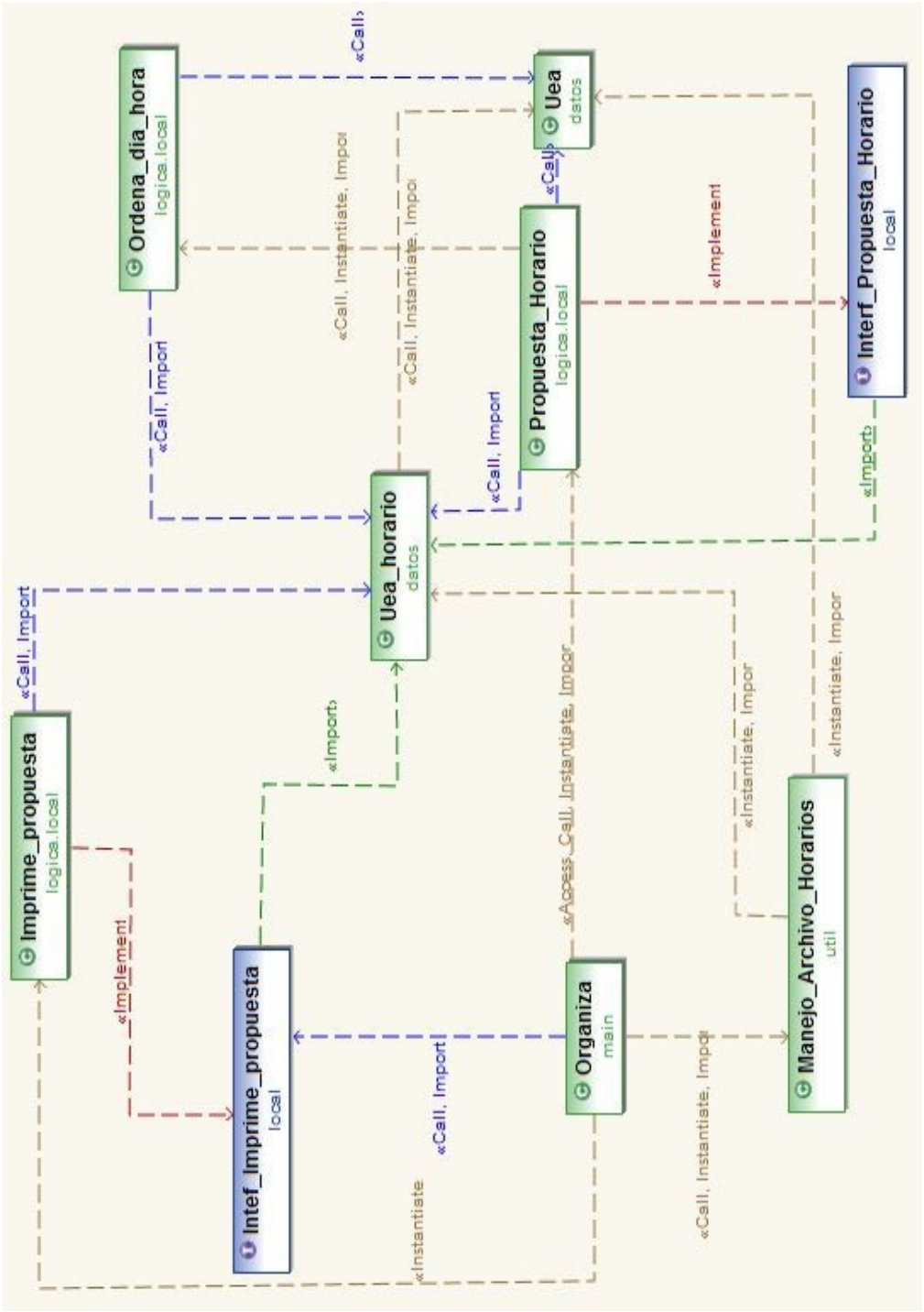

Figura 19 Dependencia de clases en Organiza

IX. Desarrollo

1. Descripción

El proyecto comprende cuatro módulos, el último fue realizado durante el proyecto terminal dos.

Para el ultimo, "Modulo de implementación" se hizo un cambio en cuanto al orden de programar los componentes, esto es, se programo primero el componente "Selección de materias" en lugar del algoritmo de coloración de grafos, ya que la entrada para el componente organización de horarios es saber cuáles son las materias disponibles que puede cursar el alumno. Todo lo demás fue construido de acuerdo con lo programado en el calendario.

Existe un cambio importante en el sistema, el algoritmo de coloración de grafos propuesto para resolver el problema fue sustituido. ¿Por qué cambiar el algoritmo?, el punto de este proyecto es resolver el problema de organizar el horario para un alumno y darle opciones. Por lo tanto, el problema de organización de horarios tiene restricciones fuertes que no pueden ser cubiertas por el algoritmo de coloración, una de ellas era encontrar relaciones que permitan crear un grafo bipartito y sin este, no se puede crear un grafo de todas las materias para su coloración.

Para resolver el problema se busco otra alternativa, crear relaciones entre materias que se ofrecen el mismo día y que el horario en el cual se imparte sea el más próximo a la materia anterior. Una vez realizado lo anterior, se crea un grafo para cada día de la semana, se debe encontrar el camino más largo entre todas las opciones de horarios que se tienen y almacenarla. Los caminos más largos para cada día son buenos candidatos. Por último se deben crear combinaciones con las opciones que se tienen y presentárselas al usuario.

2. Selección de materias

Este componente fue programado en su totalidad durante el proyecto terminal uno.

Durante el proyecto terminal dos, se le creó una interfaz gráfica muy sencilla pero que facilita la entrada de datos para el usuario. Ahora no se necesita crear un archivo de texto con las materias que aprobó el alumno. Solo se deben seleccionar las claves de las uea's que tienen aprobadas y el sistema le muestra las materias que puede cursar el siguiente trimestre.

La pantalla principal para seleccionar las claves de las materias es la siguiente:

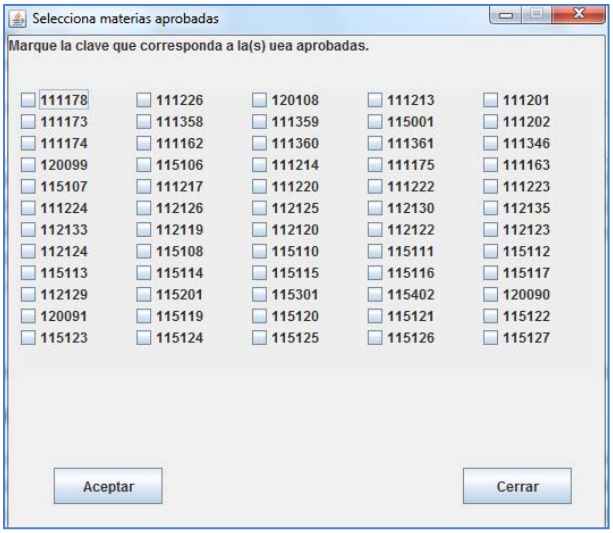

Figura 20 Pantalla principal de Selección de materias

En el manual de usuario se especificará más a fondo de cómo el usuario debe utilizarlo.

La implementación el componente *Selección de materias* se encuentra descrita en el reporte del proyecto terminal uno.

3. Implementación de la interfaz grafica para el componente selección de materias

Para crear la interfaz se hace uso de Java Swing. Se crea la clase *Seleeciona.java* que es una extensión de la clase *JFrame* esto nos facilita la creación de la interfaz. Aquí no se comenta a fondo los métodos y componentes de esa clase, puesto que se encuentra dentro el código documentado.

4. Organización de horarios

Entrada de datos

La salida que se obtiene del componente *Selección de materias* es la que se utilizará para crear las posibles opciones de horario. A esta salida se le debe dar un formato específico para que el programa pueda obtener una solución.

Supongamos que se obtiene la siguiente salida del componente *Selección de materias:*

111173 Física I
Creditos: 9

111202 Cálculo Diferencial e Integral II
Creditos: 12

Para la salida anterior se le da el siguiente formato:

111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:8:00	fin:9:30
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:11:00	fin:12:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:11:00	fin:12:30

Se puede observar que la uea con clave: 111202 tiene dos opciones de horario.
Cada línea está conformada por seis campos, de izquierda a derecha estos son: clave de la uea, nombre de la uea, créditos, días en que se imparte, inicio y fin del horario de la materia.

- **Clave**: Este campo debe colocarse los seis dígitos correspondientes a la clave de la uea.
- **Nombre**: Antes de colocar el nombre de la uea se debe anteponer los dos caracteres "*n:*" (una *n* y dos puntos). Esto para que el programa pueda identificar de que campo se trata.
- **Créditos**: La etiqueta que se antepone es "*cred:*" enseguida puede colocar el valor de créditos que tiene esta uea.
- **Días**: Se colocan las primeras dos letras para cada día en que se imparte la uea. Ejemplo si la uea se imparte lunes, miércoles y viernes. Entonces la etiqueta *días:* va seguida de *Lu-*-Mi-*-Vi*, donde "-*-" es indispensable para separar los días y con el asterisco indicar que ese día no se imparte la materia.
- **Inicio**: La hora en que comienza a impartirse la uea. Colocar la etiqueta "*inicio:*" y después la hora.
- **Fin**: Se debe colocar la hora en la que termina la clase para cierta uea. Siempre se debe colocar la etiqueta "*fin:*" antes de colocar la hora. El formato de las horas, tanto para la hora de inicio como de fin es de 24 hrs.

El archivo de los horarios disponibles de las uea's a cursar se debe guardar con la extensión "*.uea*", la única razón por la que se manejo de esta forma es para que el usuario detecte el archivo de prueba que contiene el sistema. . Hasta este punto el usuario lo único que debe hacer es crear el archivo de datos de entrada para poder entregárselo al sistema y este genere las opciones de horario.

Para poder separar los campos del archivo de entrada se crea el método ***private*** *Uea_horario separarDatos(String linea)*, dentro de la clase *Manejo_Archivo_Horarios* que se encuentra en el paquete *útil*. Ver figura 21.

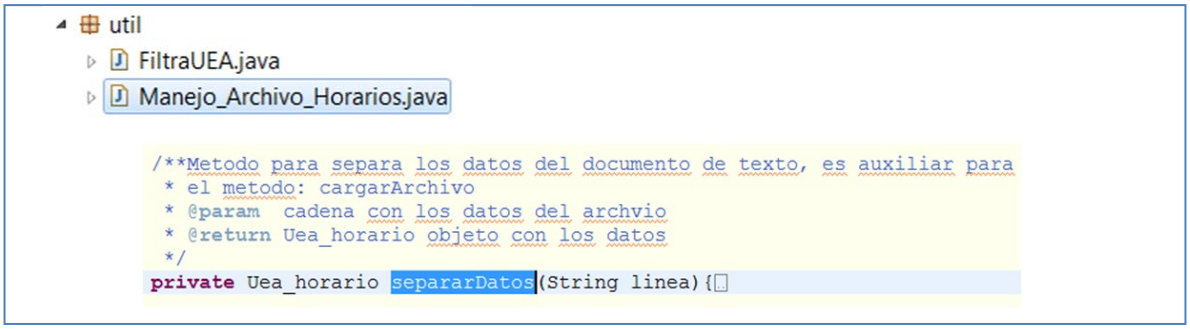

Figura 21 Ubicación de la clase Manejo_Archivo_Horarios.java

El método anterior trabaja en base al manejo de cadenas. Cada línea es separada en los atributos correspondientes para el objeto de tipo *Uea_horario.java*. Todos los objetos creados son almacenados dentro de una estructura de datos, específicamente una lista ligada. La elección de una lista ligada se debe a que podemos guardar y buscar los elementos de forma relativamente rápida y porque no se maneja una gran cantidad de ellos.

5. Ordenar por día y hora

Hasta este punto solo se tiene una lista de objetos de tipo *Uea_horario* almacenados dentro de una estructura de datos. El primer paso a realizar es ordenarlos por día y enseguida por hora, de esta forma se lleva un control sobre datos ordenados y se crean listas independientes para cada día de la semana. Obteniendo cinco listas como resultado.

La clase implementada para realizar el procesamiento descrito anteriormente lleva por nombre: *Ordena_dia_hora.java* está dentro del paquete *lógica.local*. Ver figura 22.

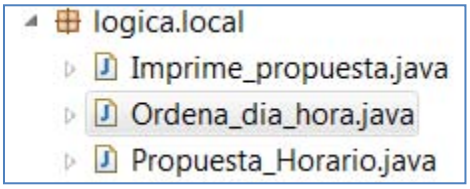

Figura 22 Ubicación de la clase Oredena_dia_hora.java

El ordenamiento por día se realiza de la siguiente forma, cada objeto tienen el atributo días y de esta forma solo se separan solo preguntando a que día pertenece la uea.

Para la ordenación por horas se utilizo el algoritmo de ordenamiento por burbuja, se eligió este porque su implementación es sencilla y es eficiente en este problema.

6. Crear la propuesta de horario

La clase más importante dentro del desarrollo es *Propuesta_Horario.java* esta es encargada de procesar las cinco listas que contiene ordenadas las materias por horario. La función que debe realizar esta clase es relacionar las materias que se pueden cursar el mismo día y casi enseguida una de otra en el horario, esta relación se realiza por parejas. En caso de que solo fuera una materia la que existe ese día, no hay ninguna a relación a crear e inmediatamente es candidata para ser cursada. En cuanto se terminan de crear las relaciones, el algoritmo debe encontrar la cadena más larga de materias, cuando hablamos de la cadena más larga nos referimos a que cada pareja de materias es enlazadas con el horario más aproximo para ese día. La cadena de materias considerada como la más óptima es la más larga, porque

contiene el mayor número de asignaturas que se pueden cursar. En caso de que hubiese más de una cadena optima del mismo tamaño, es considera como otra solución al problema y se guarda como candidato.

La ubicación de la clase *Propuesta_Horario.java* se encuentra dentro del paquete *lógica.local*. Ver ilustración 4. El código se encuentra documentado, ahí puede ver la descripción de cada función.

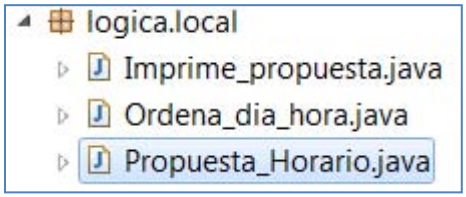

Figura 23 Ubicación de la clase Propuesta_horario.java

7. *Mostrar los resultados*

Una vez finalizada la búsqueda de las cadenas para los cinco días de la semana, el siguiente paso es crear combinaciones de los horarios para cada día. Se realiza a través de un método llamado *combina()* dentro de la clase *Imprime_propuesta.java.*en cuanto son obtenidas las posibles combinaciones los resultados obtenidos deben ser mostrados al usuario de una forma que los pueda comprender. *Imprime_propuesta.java* se encarga de dar el formato adecuado a los horarios obtenidos.

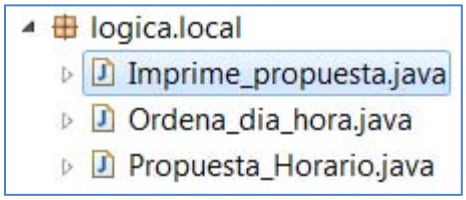

Figura 24 Ubicación de la clase Imprime_horario.java

8. *Interfaz gráfica*

El desarrollo de este proyecto no tenía contemplado implementar una interfaz gráfica que le permitiría al usuario interactuar con la aplicación. Aún con un tiempo ajustado se intento crear una interfaz básica que permita ejecutar de forma sencilla la aplicación. Utilizando la biblioteca gráfica de Java, mejor conocida como Java Swing se realizo la interfaz para la aplicación. Requirió de tiempo y trabajo puesto que no se tienen conocimientos amplios con Swing.

X. Documentación código fuente

Java cuenta con la herramienta Javadoc, a través de esta se genero la documentación de nuestro código fuente. El formato que se obtiene en Javadoc es HTML. Esta documentación no se incluye en este reporte, pero puede ser consultada por separado. Dentro de la carpetas documentación de código fuente.

XI. Elaboración del manual de usuario

Para que el usuario pueda interactuar adecuadamente con las aplicaciones es necesario contar con un manual de usuario. La elaboración del manual se trato de hacer lo más sencilla posible, no utilizando términos tan complejos para facilitar su consulta. Se ilustro con capturas de pantalla para hacerlo más amigable. También puede consultarse independientemente en la carpeta de manual.

XII. Conclusiones

En el seminario del proyecto terminal se propuso resolver el problema de organizar el horario de un alumno por medio de la coloración de grafos, a continuación se describe cuales fueron algunos de los aspectos más importantes durante este trabajo y el resultado.

El primer objetivo consistía en diseñar y modelar el sistema, en este punto se debe hacer la corrección que no es un sistema, es una aplicación que permitirá obtener opciones de horario para un alumno de la carrera de ingeniería en computación. En base al análisis se observo que primero habría que programar el componente que le permitiera al usuario saber cuáles son las materias que puede cursar el siguiente trimestre. La aplicación de selección de materias no requiere de un algoritmo difícil de realizar pero requirió de tiempo para diseñarlo. La elaboración de estas tareas fue parte del primer proyecto terminal y solo hubo el pequeño cambio de reprogramar algunas actividades.

Con el análisis hecho en el primer proyecto sale a relucir que las restricciones del problema son fuertes y resolver el problema por medio de la coloración se volvía difícil. Se me complicaba encontrar alguna forma de crear las relaciones bipartitas, que son la base para crear el grafo al cual debía aplicarse el algoritmo de coloración. Dentro de algunas investigaciones que realice, muchos han planteado una solución pero las restricciones cambian. Una de ellas es que las materias no tienen un horario preestablecido, así que solo se resuelve el que no existan empalmes y todos los alumnos puedan cursar sus materias. En nuestro problema es diferente, los horarios ya están establecidos, “es lo que hay”. Algunos realizan una combinación de algoritmos genéticos y coloración para resolver problemas de horarios.

Si se permitiera que el programa estableciera los horarios desde un principio, se facilitaría la solución. Para no congelar el proyecto, se opto por buscar otra solución utilizando

algoritmo de fuerza bruta. Es probable que no sea el mejor método pero se obtienen una solución al problema.

Al final se opto por dividir la aplicación en dos, una de ellas se encarga de ofrecerle al alumno las materias que puede cursar el siguiente trimestre y la segunda en base a los horarios establecidos ofrece opciones de horarios para el usuario.

Se cumplió el objetivo de realizar un organizador de horario pero no contaba con una interfaz sencilla que le permitiera al usuario interactuar con la aplicación. El último par de semanas se asigno un poco de tiempo para crear una pequeña interfaz. Que facilitaría la ejecución de la aplicación en un sistema con Windows o Linux.

La elaboración del manual y el reporte es un tanto breve, pero se trataron de cubrir los puntos más importantes dentro del desarrollo del proyecto.

El proyecto da para ir más allá que el simple hecho de mostrar opciones de horario a un alumno, si se continúa trabajando en este problema se puede pensar en un sistema que obtenga directamente los datos académicos del alumno desde la base de datos y con ellos saber cuáles son las materias que puede inscribir el alumno. Ofreciendo horarios donde la mayoría de los alumnos no tengan problemas para armar el propio.

Anexo A. Manual de usuario

Universidad Autónoma Metropolitana

Unidad Azcapotzalco

Proyecto Terminal de Ingeniería en Computación

Manual de usuario

Aplicación para la organización de horarios

Autor: Yáñez Castillo José Alberto

Trimestre: invierno 2011

Introducción

Este documento brinda ayuda en el uso de la aplicación. El software necesario para ejecutar la aplicación y muestra un ejemplo de cómo funciona el sistema.

Instalación de software para ejecutar la aplicación

Windows

Cuando la aplicación es ejecutada en el sistema operativo Windows debe asegurarse de tener instalado el software de Java, también llamado *máquina virtual Java*.

¿Cómo saber si su sistema tiene instalado Java? Si no está seguro de esto, diríjase a la dirección: http://www.java.com/es/download/ ahí se encontrará con la siguiente ventana (ilustración 1).

Ilustración 1. Instalar máquina virtual Java

Al descargar el software Java siga las instrucciones que se le presentan para su instalación, no debería tener ninguna complicación para su instalación. Si tiene problemas consulte la ayuda de Java.

Linux

En el sistema operativo Linux también se requiere tener instalada la maquina virtual de java. Para instalarla, diríjase a la dirección: http://www.java.com/es/download/ , de clic en el botón de descarga y elija el paquete que cumpla con los requisitos para su equipo. Ver ilustración 2.

Ilustración 2. Paquetes Java para Linux

Los usuarios de este sistema suelen ser mucho más experimentados, entonces no tendrán problemas para instalarla.

Ejecución de la aplicación

La aplicación para organizar el horario está conformada por dos pequeños aplicaciones una de ellos es ***selecciona.jar*** y el segundo es ***organiza.jar***.

Ejecución de selecciona.jar

La primera aplicación ***selecciona.*****jar** presenta una interfaz donde se le muestra al usuario un conjunto de casillas de verificación, cada una está asociada a una clave de uea. El usuario debe seleccionar las que ya ha cursado para que la aplicación muestre las materias que puede cursar. Una vez que el usuario selecciono las claves que tiene aprobadas, puede dar clic en el botón **Aceptar** para obtener el resultado o bien **Cerrar** para terminar el programa.

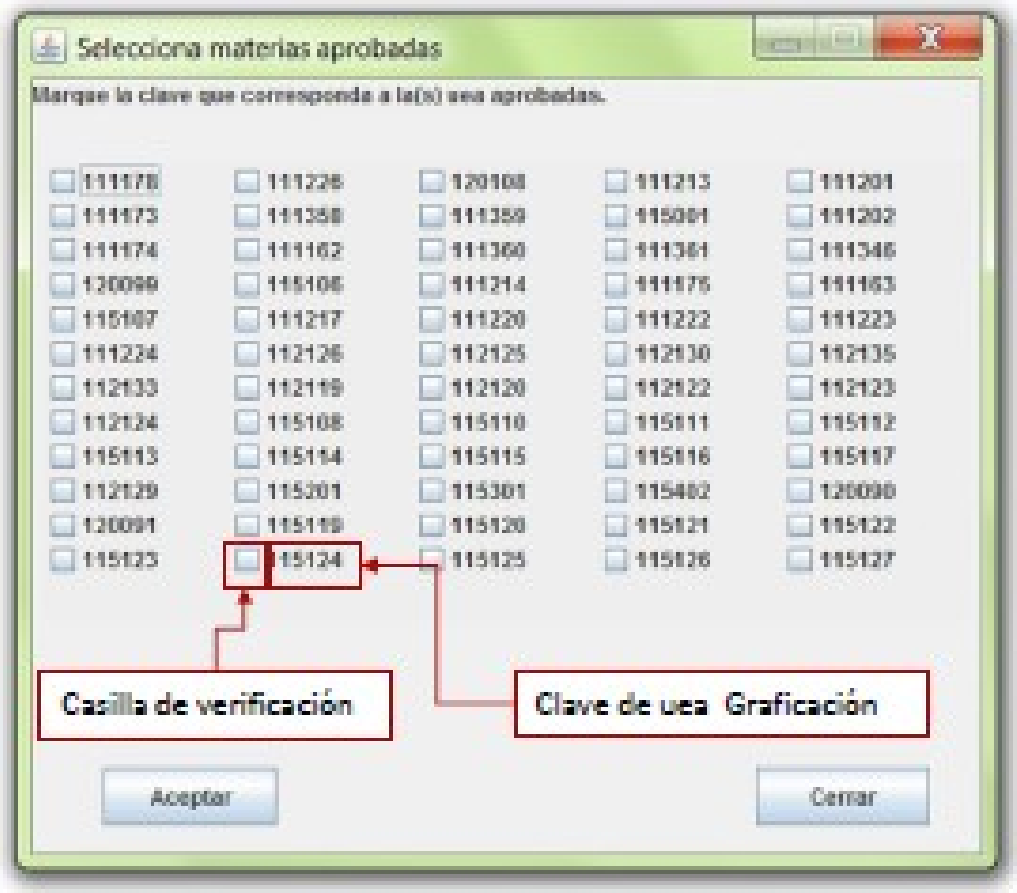

Ilustración 3 Ventana inicial de *selecciona.jar*

Ejecución de selecciona.jar desde Windows

En Windows la ejecución se realiza de forma normal sí instalo correctamente el software para ejecutar la aplicación.

1) Localice el archivo **selecciona.jar**, este debe estar donde usted lo guardo. Para ejecutarlo de doble clic sobre **selecciona.jar**. Ilustración 4.

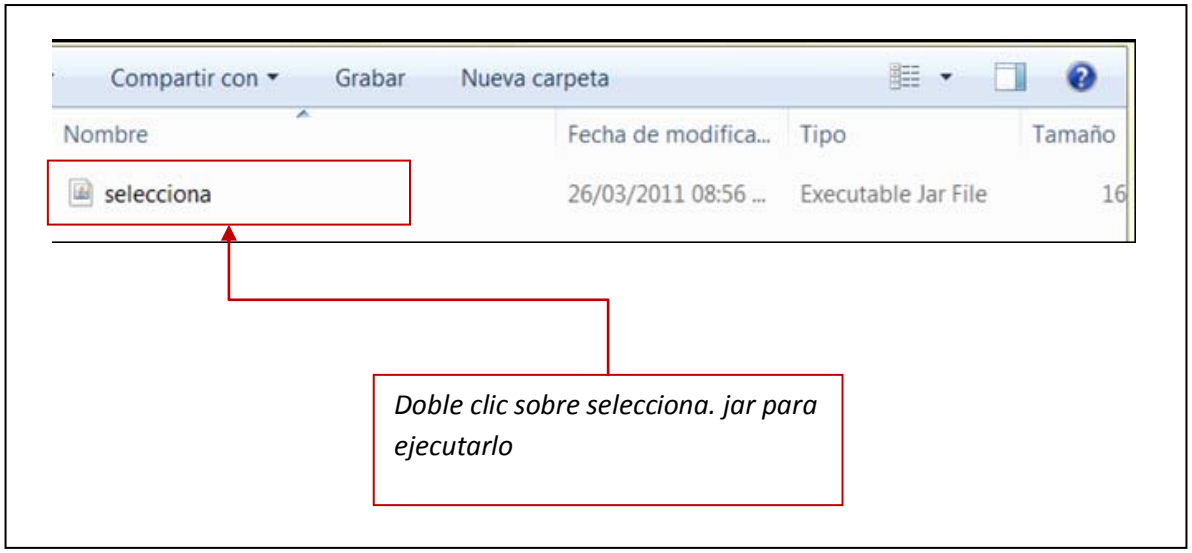

Ilustración 4. Ubicación del archivo *selecciona.jar*

2) Seleccione las claves de las uea's que tiene aprobadas. Ilustración 5.

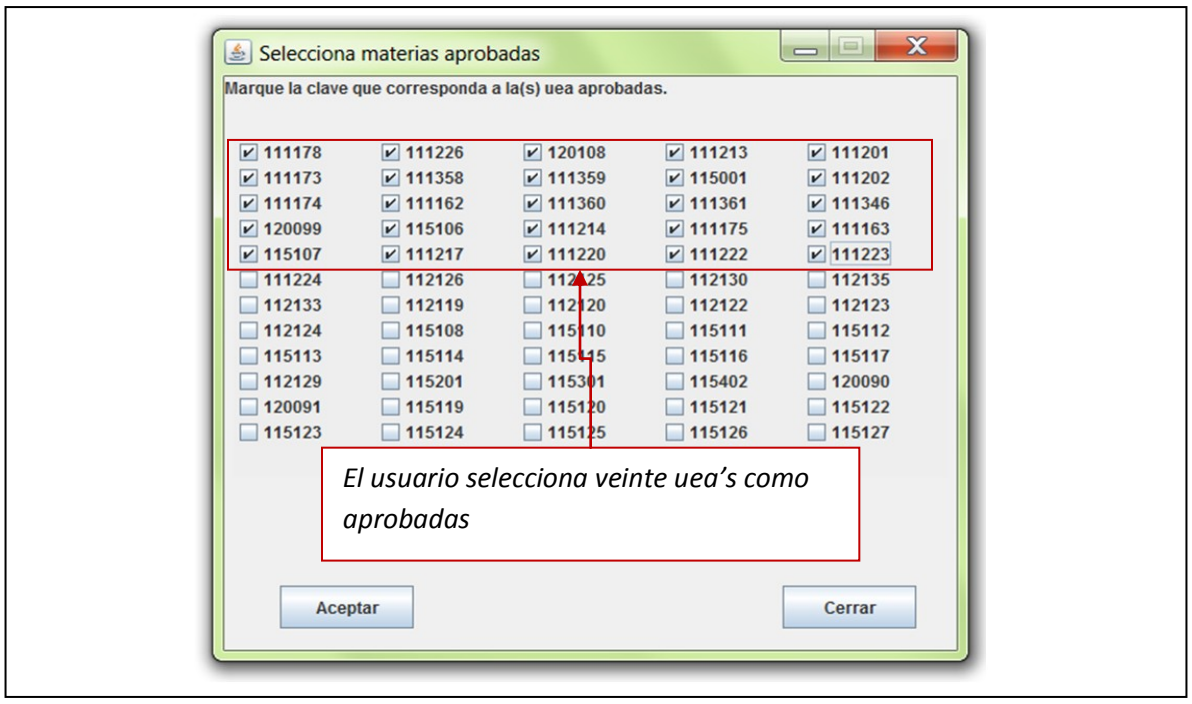

Ilustración 5. Selección de uea's en Windows

3) Dar clic en el botón **Aceptar**. Ilustración 6.

   a. Si desea corregir algunas claves de clic en el botón **Regresar**

   b. Una vez corregidas las claves puede dar clic en **Aceptar** y se muestran los nuevos resultados.

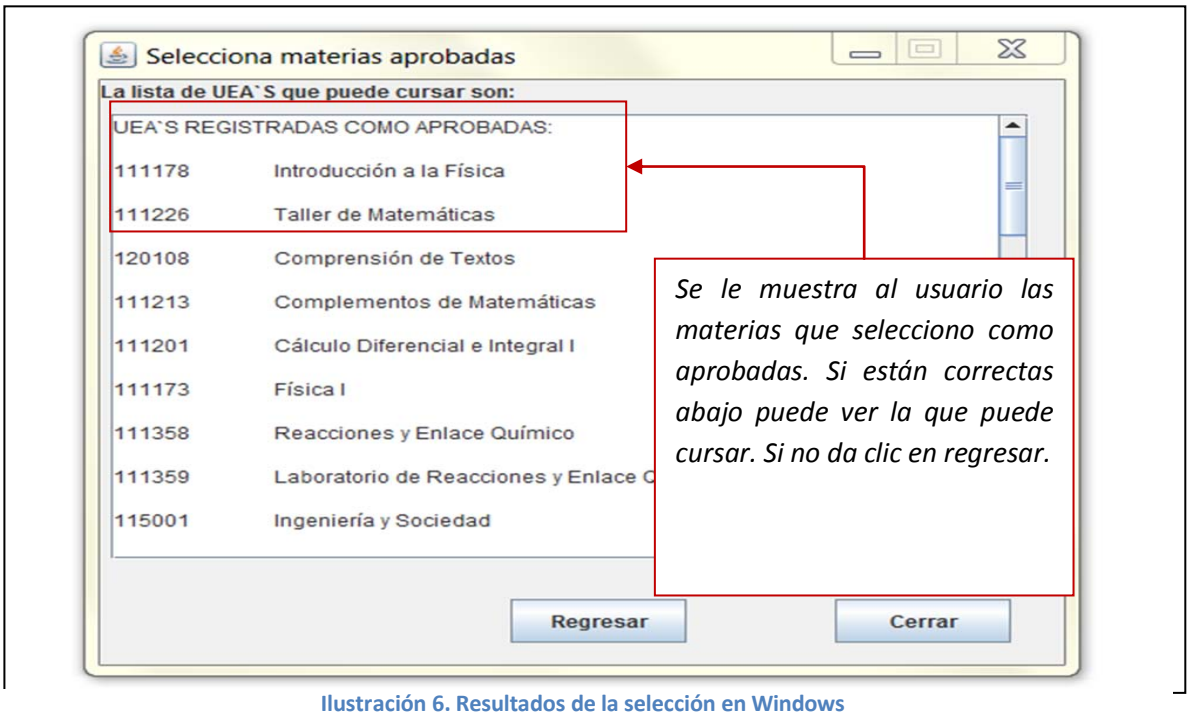

Ilustración 6. Resultados de la selección en Windows

4) Se muestran los resultados de la aplicación. Para terminar solo de clic en el botón **Cerrar.** Ilustración 7.

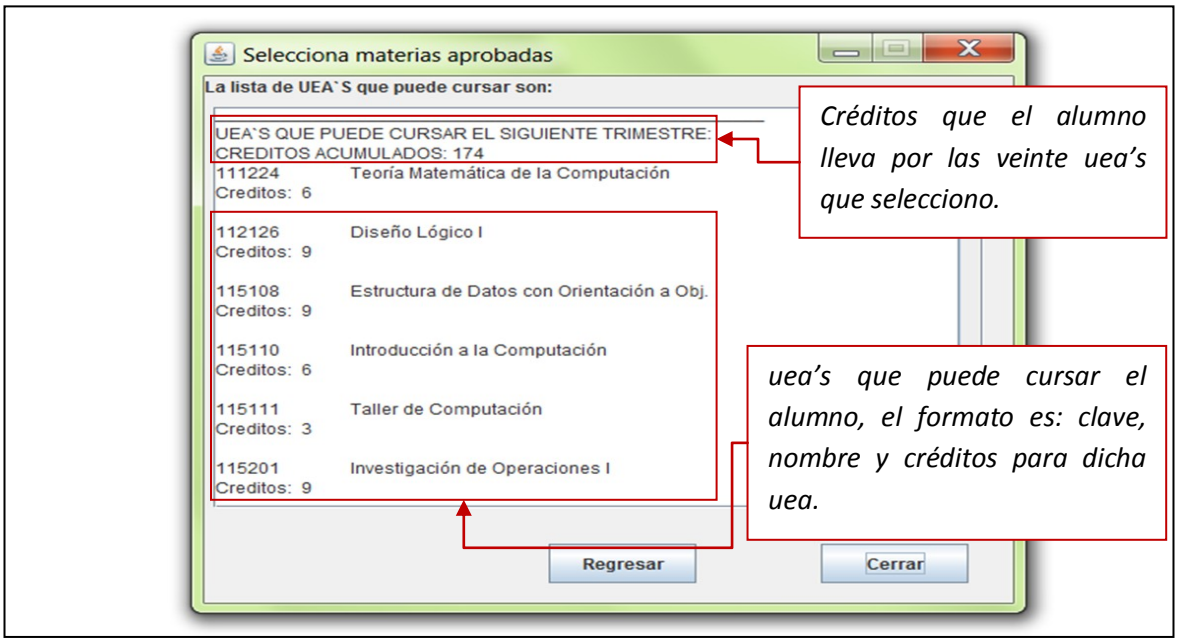

Ilustración 7. Resultados del programa

Ejecución de selecciona.jar desde Linux

Para realizar la ejecución de la aplicación ***selecciona.jar*** desde el sistema operativo Linux, se debe abrir la herramienta **Terminal** (consola de comandos). Localice el archivo ***selecciona.jar*** y vaya a ese directorio desde la terminal.

1) Teclear el siguiente comando para su ejecución: ***java –jar selecciona.jar***.

Esto ejecutara la aplicación. Ilustración 8.

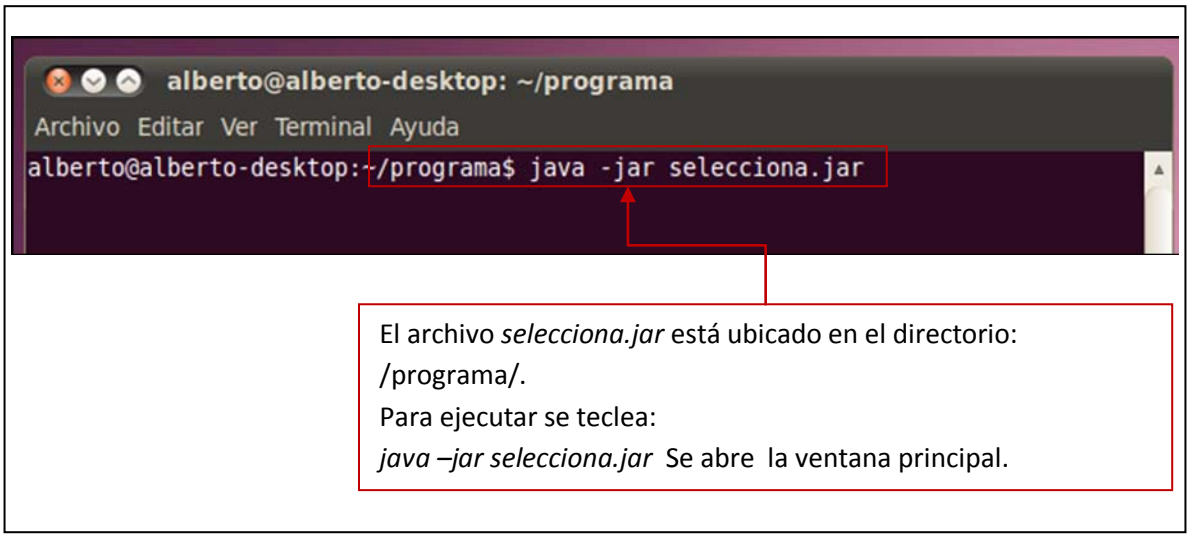

Ilustración 8 Ejecutando en Linux

2) Iniciada la aplicación seleccione las claves de las uea's que tiene aprobadas.

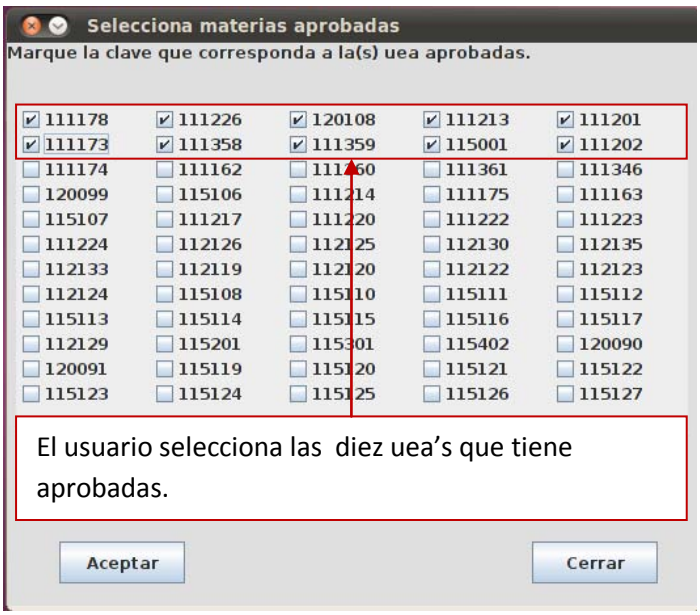

Ilustración 9. Selección de materias en Linux

3) Dar clic en el botón **Aceptar**. Ilustración 10.

a. Si desea corregir algunas claves de clic en el botón **Regresar**

b. Una vez corregidas las claves puede dar clic en **Aceptar** y se muestran los resultados.

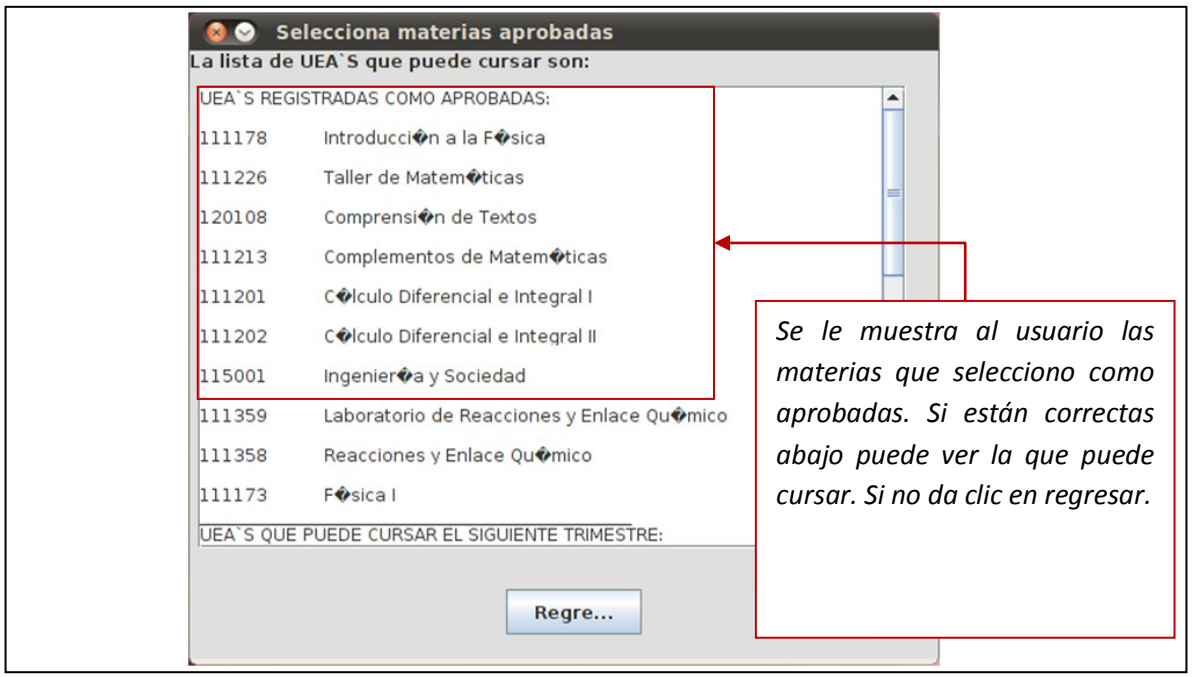

Ilustración 10. Resultados de la selección en Linux

4) Se muestran los resultados de la aplicación. Para terminar solo de clic en el botón **Cerrar.**

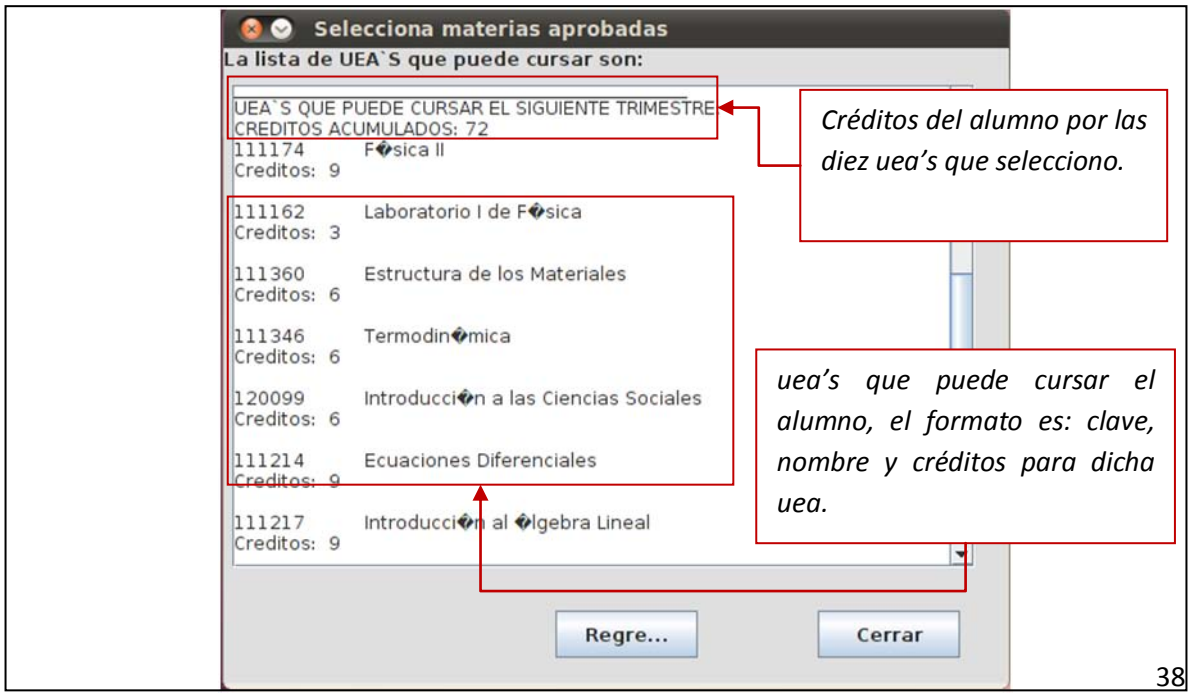

Ilustración 11. Resultados

Ejecución de organiza.jar

La segunda aplicación, ***organiza.*jar** presenta una ventana (Ilustración 12) donde solo se muestran dos botones. El primer botón tiene el nombre "**Cargar Archivo**", al dar clic en este se abre una segunda ventana (Ilustración 13) donde puede seleccionar el archivo con los horarios disponibles. Y el segundo botón "**Salir**" es para terminar la aplicación.

La aplicación solo acepta archivos con extensión ".**uea**". Estos deben ser creados desde un editor de texto plano. La forma de crearlos se muestra en el apartado "**Crear archivo.uea de horarios**"

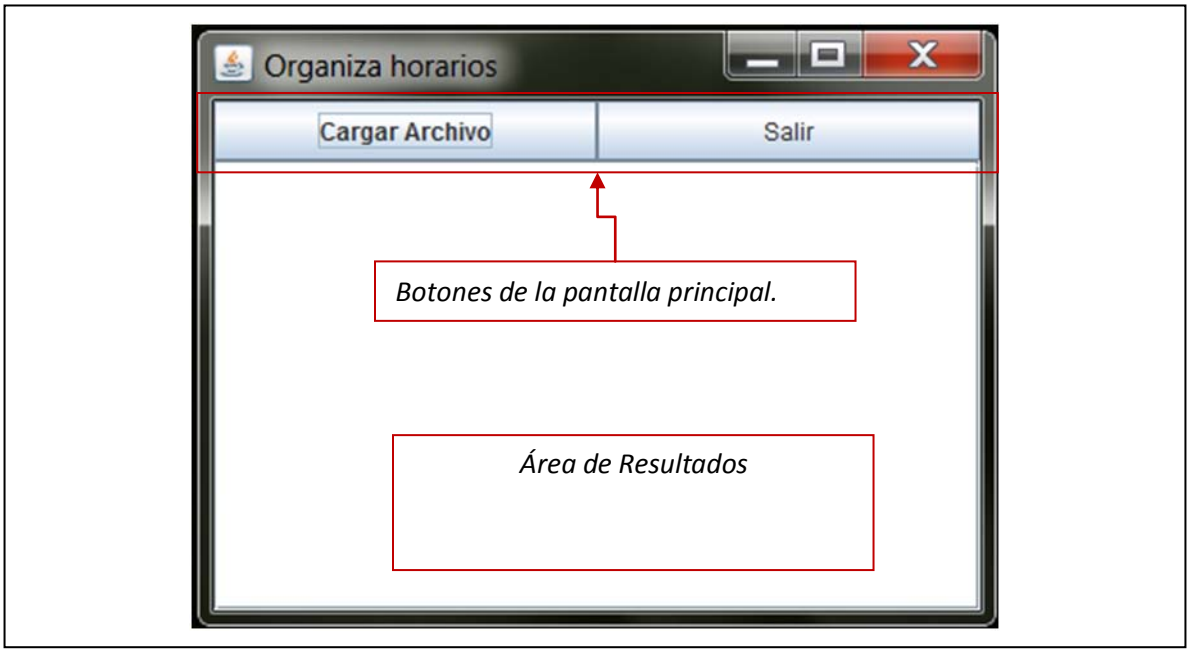

Ilustración 12 Ventana principal de *Organiza horarios*

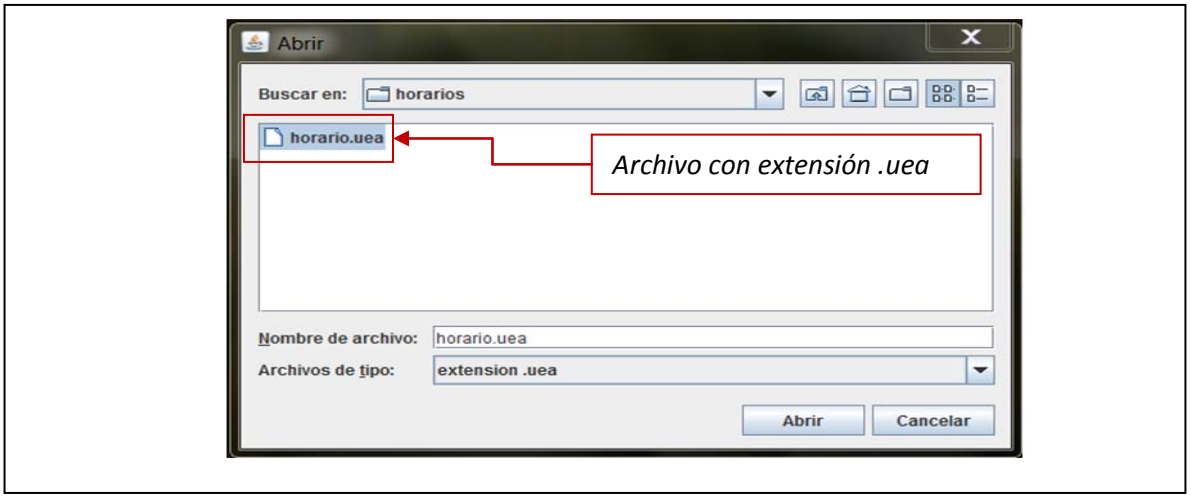

Ilustración 13 Ventana navegación de archivos

Crear archivo.uea de horarios

La aplicación contiene un archivo de demostración con el nombre ***horario.uea***, este puede servirle de plantilla para crear su propio archivo.

Para crear el archivo, realice una copia de ***horario.uea*** y ábralo con un editor de texto plano, después siga estos pasos:

1) Al abrir el archivo se observan varios campos de izquierda a derecha estos son.

    a. Clave: Compuesta de seis dígitos, todas las materias deben tener su clave.

    b. Nombre uea: Se debe anteponer "**n:**" y enseguida el nombre de la uea.

    c. Créditos: Valor numérico, debe anteponer la etiqueta "**cred:**" y el valor.

    d. Días: Los días en que se imparte la materia deben colocarse estrictamente abreviados. Ejemplo Lunes = Lu, si el martes no se da la asignatura debe colocar "**-*-**" como separador e indicando con el "*****" que no está disponible ese día y así se repite para todos los días.

    e. Hora de inicio: Cada uea tiene una hora para comenzar, esta se marca con "**inicio:**" y enseguida coloca la hora en el formato de 24hrs.

    f. Hora de termino: el final de la clase para cierta uea, se utiliza la etiqueta "**fin:**" y enseguida se coloca la hora. También en el formato de 24hrs.

2) Debe agregar los horarios disponibles de las materias que cursara. Por ejemplo, supóngase que ingresa los siguientes horarios:

Clave	**nombre uea**	**créditos**	**días**	**comienza**	**termina**
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:8:00	fin:9:30
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:10:00	fin:11:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:11:00	fin:12:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:10:00	fin:11:30
111162	n:Laboratorio I de Física	cred:3	dias:*-*-*-Ju-*	inicio:11:00	fin:14:00
111360	n:Estructura de los Materiales	cred:6	dias:*-Ma-*-Ju-*	inicio:9:00	fin:10:30
115111	n:Taller de Computación	cred:3	dias:Lu-*-Mi-*-Vi	inicio:10:00	fin:11:30
115111	n:Taller de Computación	cred:3	dias:Lu-*-Mi-*-Vi	inicio:15:00	fin:16:30

Se mantienen el formato descrito en el paso (1).

3) Lo siguiente es guardar el archivo y asegurarse que tiene extensión ***.uea***. Ver la ilustración 15.

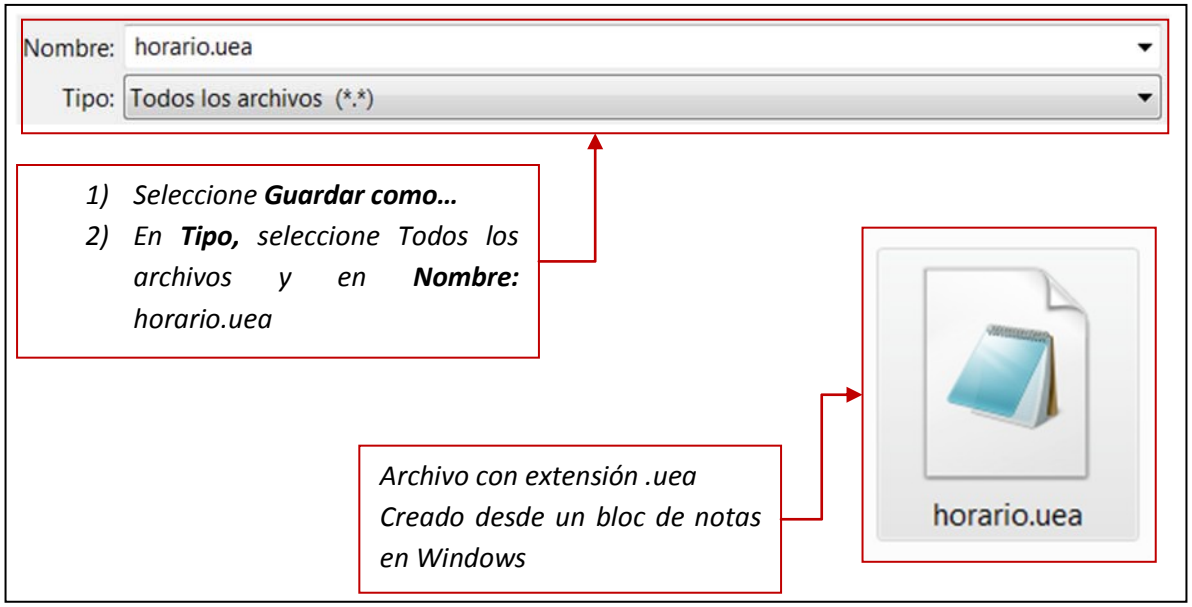

Ilustración 14. Guardando archivo *horario.uea* en Windows

Para crear el archivo en Linux, se debe seguir una secuencia similar. Puede utilizar cualquier editor de texto y salvar el archivo con la extensión (*uea*) definida anteriormente.

Ejecución de organiza.jar desde Windows

1) Para la ejecución desde Windows se realiza como cualquier programa normal, dar doble clic sobre el archivo ***organiza.jar*** iniciará la aplicación.

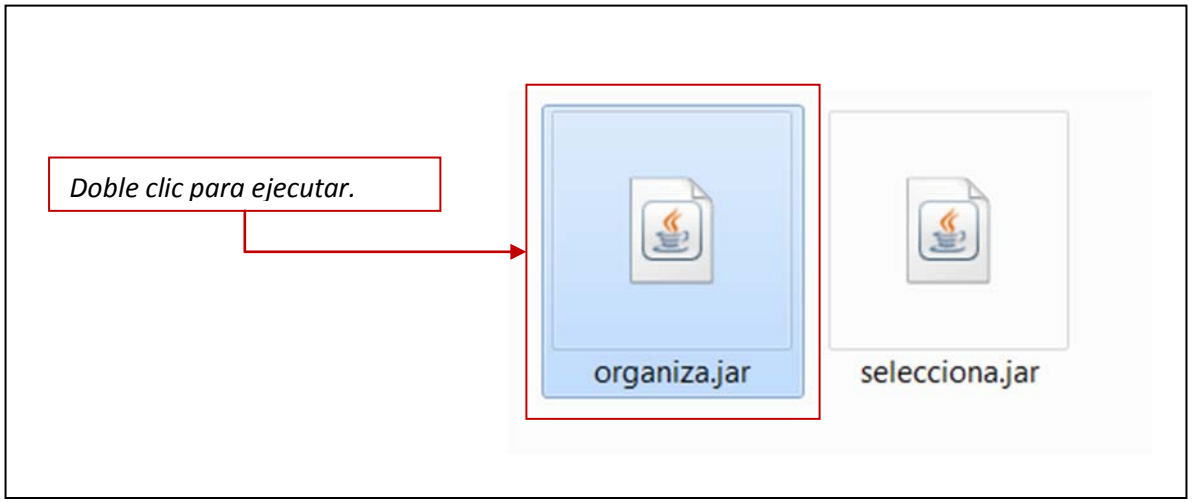

Ilustración 15 Ejecutar *organiza.jar* en Windows

2) En la ventana principal de clic en ***Cargar Archivo…*** inmediatamente se abre la segunda ventana para seleccionar un archivo. Ilustración 17.

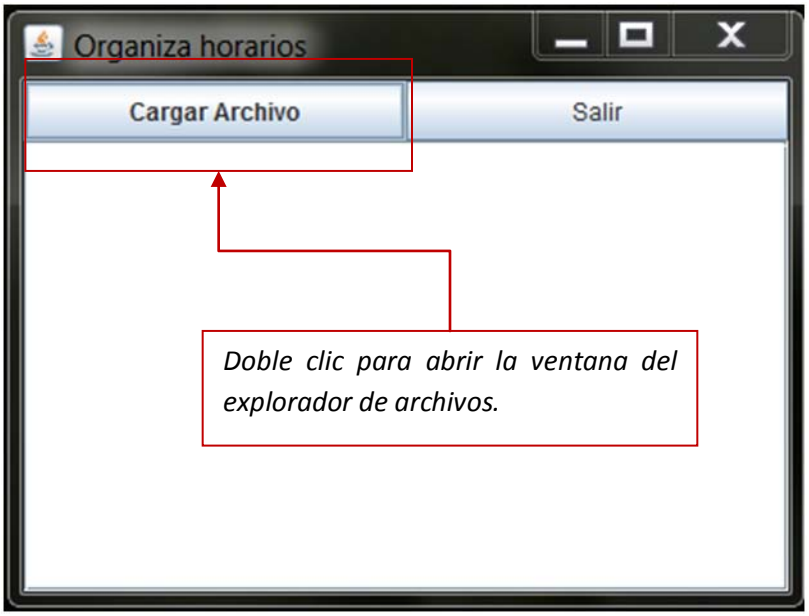

Ilustración 16. Cargar Archivo

3) En el explorador de archivos debe navegar hasta la ruta donde se encuentra el archivo con los horarios disponibles. Nota: El explorador solo reconoce archivos con extensión **.uea**, si no creo correctamente su archivo no podrá seleccionarlo.

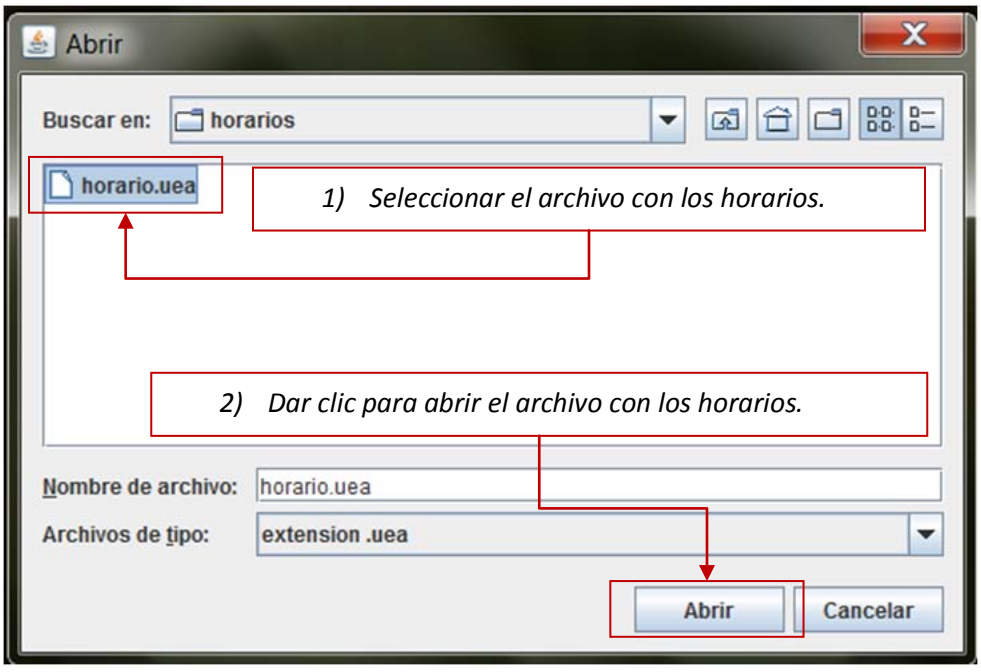

Ilustración 17. Seleccionar archivo

4) Si el archivo se cargo correctamente, se muestra la ventana con los resultados obtenidos.

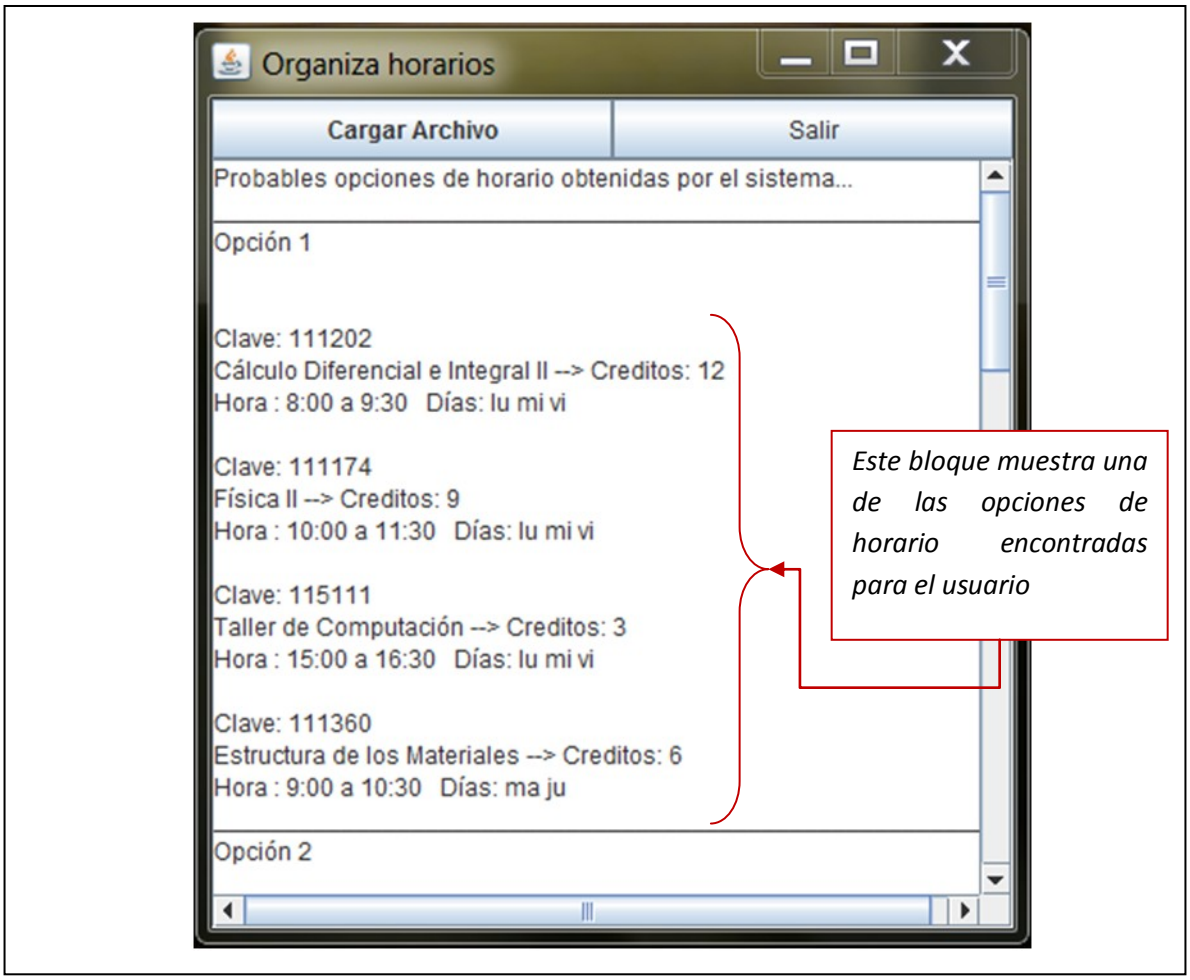

Ilustración 18. Resultados obtenidos

En la ilustración 19 se muestran los resultados de la ejecución de la aplicación. Se observa la primera opción donde el usuario puede cursar cuatro uea's. El horario sería de la siguiente forma:

Lunes	Martes	Miércoles	Jueves	Viernes
Calculo I 8:00-9:300		Calculo I 8:00-9:300		Calculo I 8:00-9:300
Física II 10:00-11:30		Física II 10:00-11:30		Física II 10:00-11:30
Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30
	Estruct. Mat. 9:00-10:30		Estruct. Mat. 9:00-10:30	

Debido a que los horarios publicados ya están programados, es muy difícil lograr un horario donde no haya huecos entre clases. Por esta razón el sistema muestra varias opciones de horario y se deja a elección del usuario la que mejor le convenga.
Veamos otra de las opciones mostradas por la aplicación, para el mismo archivo la opción tres muestra una diferencia poco significativa, pero quizá al usuario le agrade más.

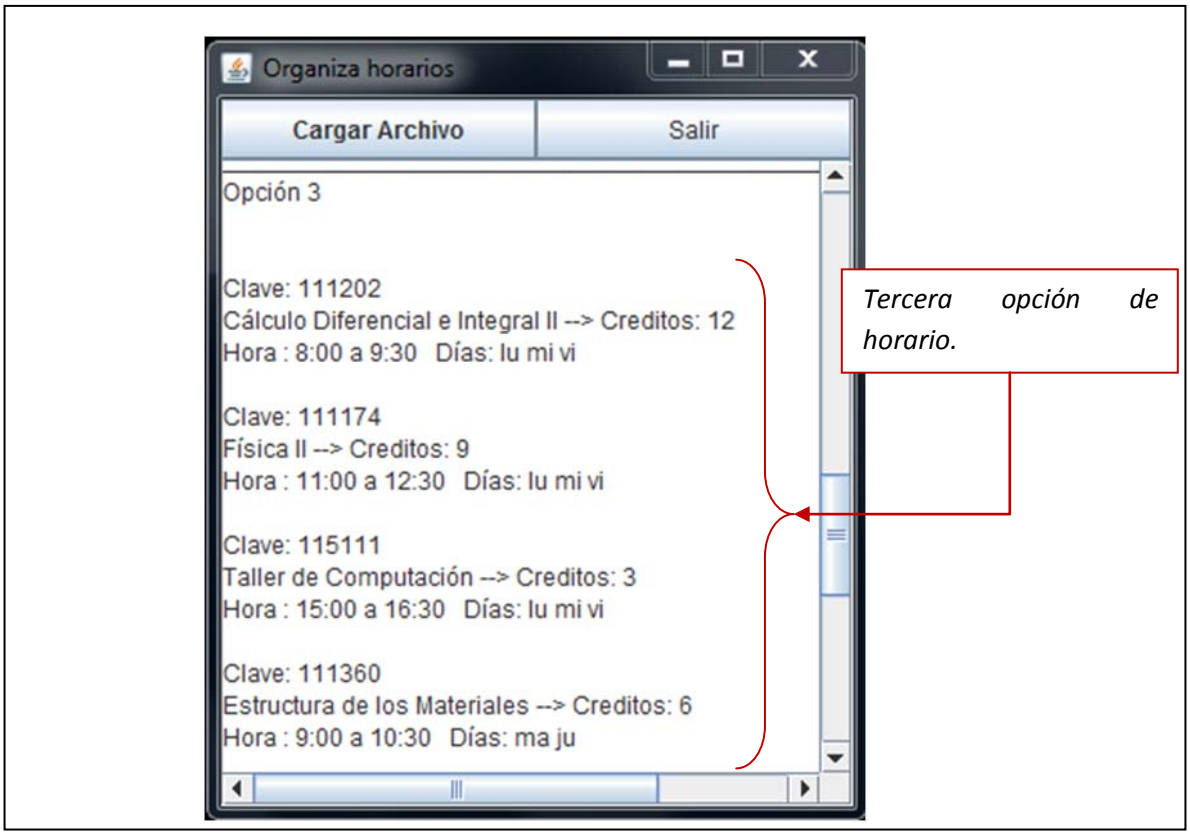

Ilustración 19. Tercera opción de horario

En esta opción (Ilustración 20) se muestran la misma cantidad de materias, solo que existe una variación en cuanto al horario en que se imparte Física II. Queda organizado de la siguiente forma:

Lunes	Martes	Miércoles	Jueves	Viernes
Calculo I 8:00-9:300		Calculo I 8:00-9:300		Calculo I 8:00-9:300
Física II 11:00-12:30		Física II 11:00-12:30		Física II 11:00-12:30
Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30
	Estruct. Mat. 9:00-10:30		Estruct. Mat. 9:00-10:30	

5) Para terminar la aplicación solo de clic en el botón **Salir** o bien solo cierre dando clic en el botón superior derecho de la ventana.

Ilustración 20. Cerrar la aplicación.

El programa no puede decir cuál es el mejor horario para un usuario. Solo presenta las opciones que puede tener.

Ejecución de organiza.jar desde Linux

En Linux debe abrir la herramienta **Terminal** (consola de comandos). Localice la ubicación del archivo **organiza.jar** y vaya a ese directorio desde la terminal.

1) Teclear el siguiente comando para su ejecución: ***java –jar organiza.jar***.

Esto ejecutara la aplicación. Ilustración 21.

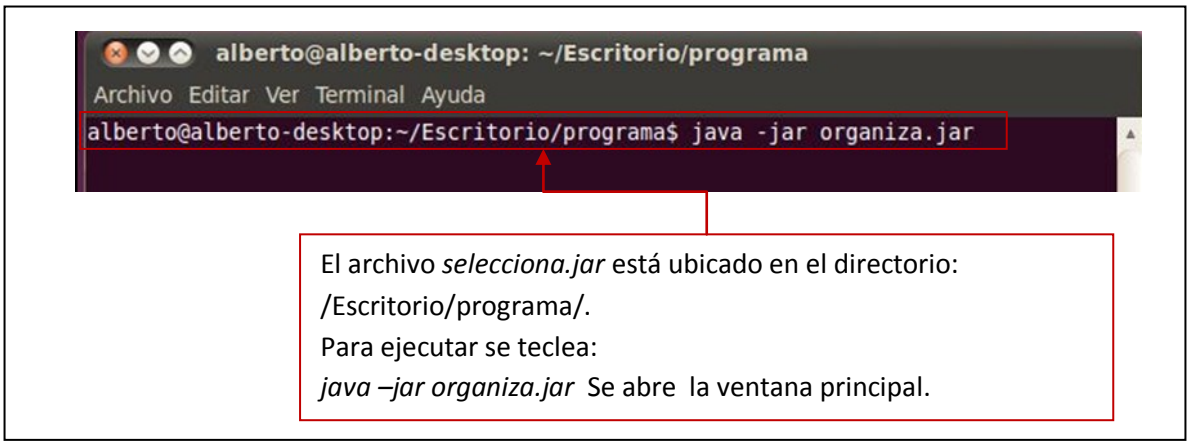

Ilustración 21. Ejecutando *organiza.jar* en Linux

2) Si se ejecuto correctamente el comando del paso anterior entonces debe de aparecer la ventana principal de la aplicación ***organiza.jar***.

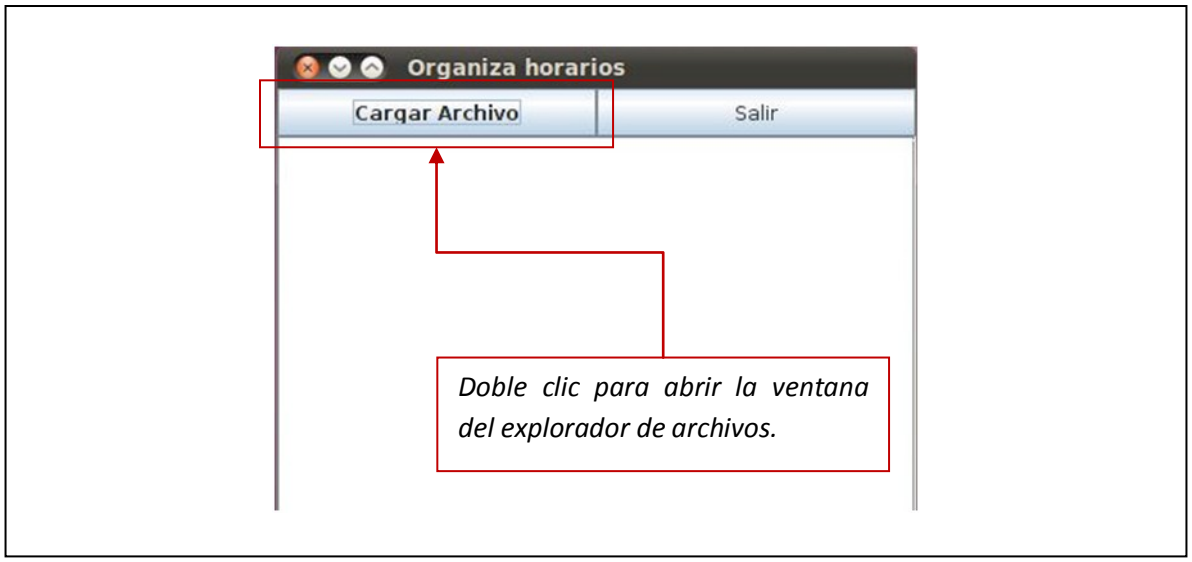

Ilustración 22. Cargar archivo desde Linux.

3) En el explorador de archivos debe navegar hasta la ruta donde se encuentra el archivo con los horarios disponibles. Nota: El explorador solo reconoce archivos con extensión **.uea**, si no creo correctamente su archivo no podrá seleccionarlo.

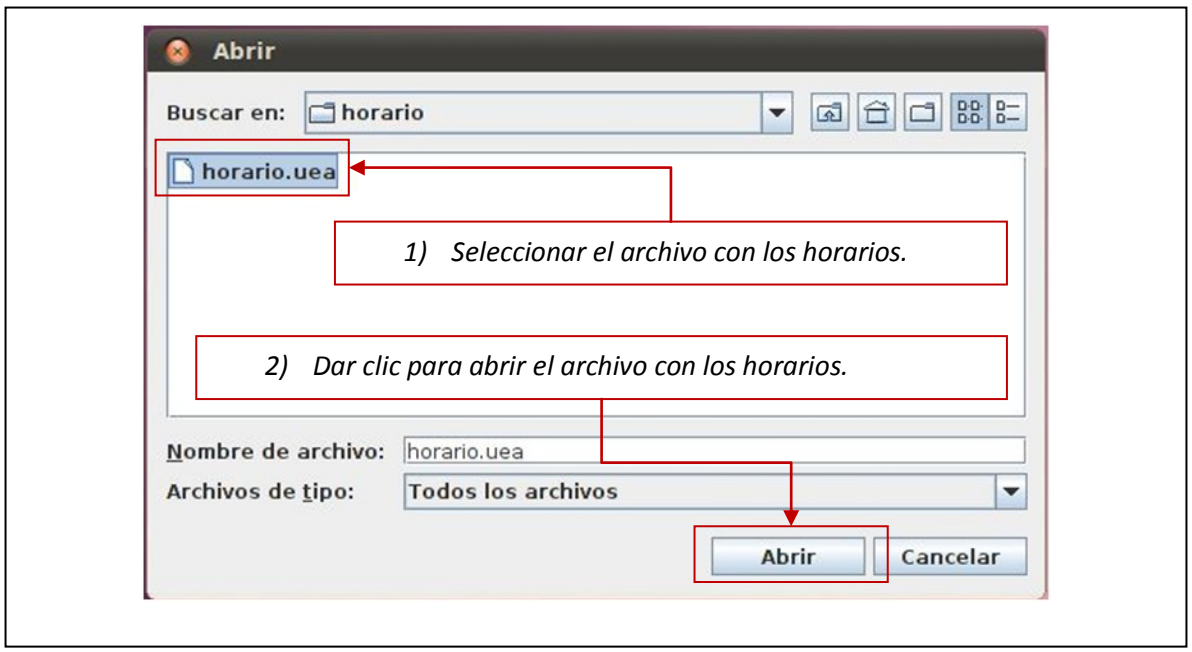

Ilustración 23. Seleccionar archivo en Linux

4) En cuanto se carga un archivo satisfactoriamente. Se muestra la ventana con los resultados obtenidos.

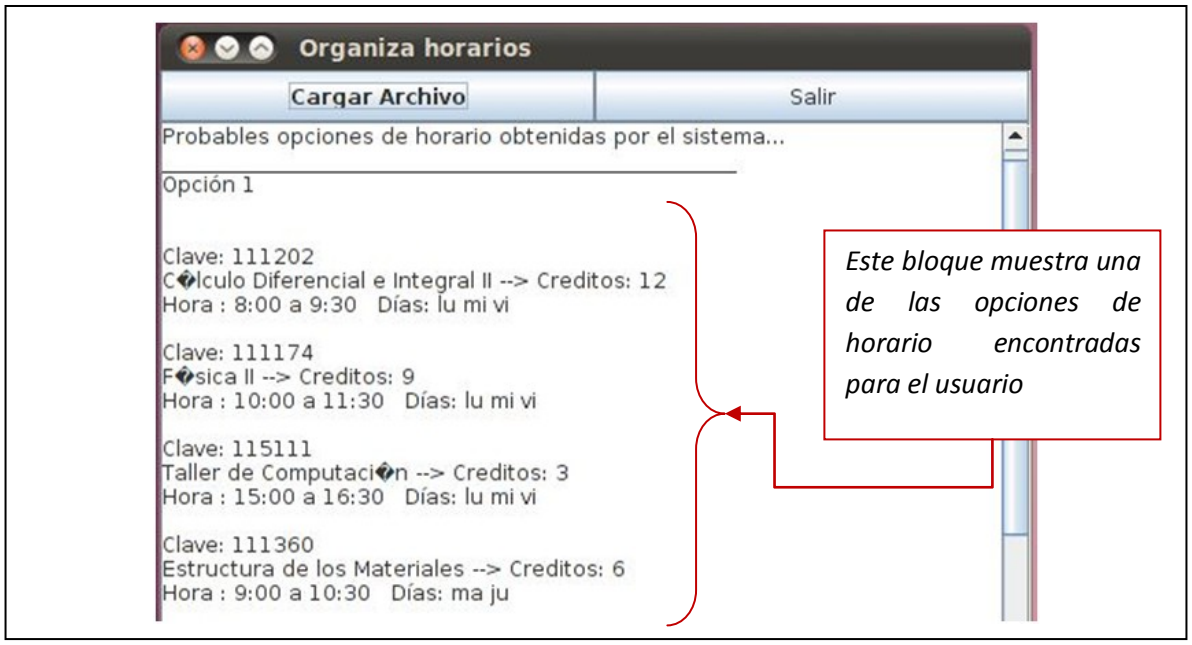

Ilustración 24. Resultados obtenidos en Linux

Los resultados mostrados en estas opciones, son los mismos que se obtienen de la ejecución en Windows. El problema que se presenta en Linux es que los caracteres que están acentuados se muestran como un símbolo no reconocido en Linux.

5) Para terminar la aplicación solo de clic en el botón **Salir** o bien cierre dando clic en el botón superior izquierdo de la ventana.

Ilustración 25. Cerrar la aplicación en Linux

Errores comunes en selecciona.jar y organiza.jar

Algunos de los mensajes de error que puede mostrar la aplicación ***selecciona.jar*** pueden ser los siguientes:

1) **No selecciono claves**: si no selecciona ninguna casilla de verificación el sistema le mostrara el mensaje ******Asegúrese de que selecciono clave(s) de uea***** (Ilustración).

Causa: Esto se debe a que dio clic en el botón aceptar sin haber seleccionado ninguna casilla.

Solución: simplemente seleccione las uea's que tiene aprobadas y de clic nuevamente en el botón **Aceptar.**

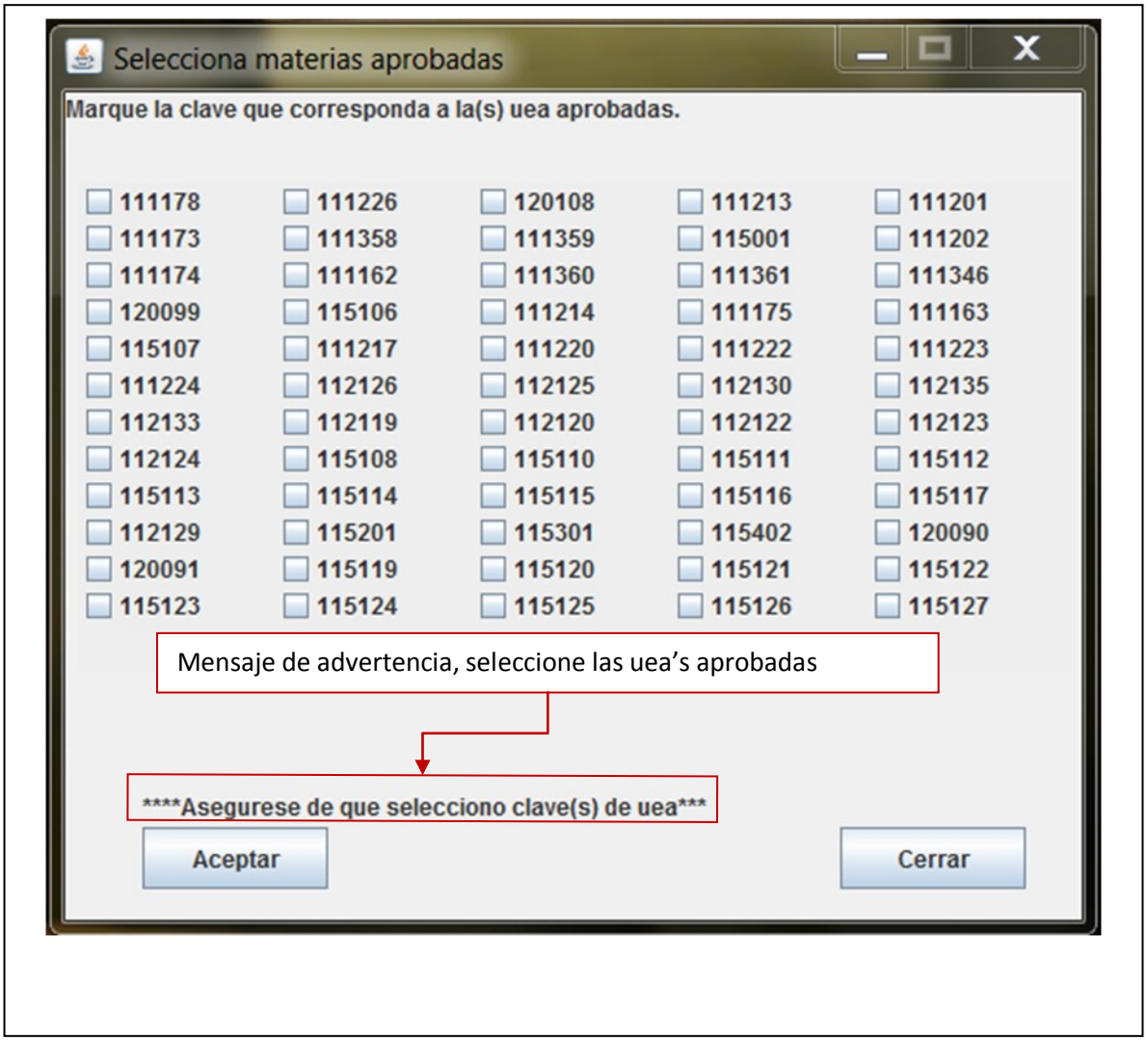

Ilustración 26. Mensaje de advertencia en *selecciona.jar*

2) **No selecciono ningún archivo**: Se muestra la advertencia "**No selecciono ningún archivo".**

Causa: Dio clic en el botón ***Cargar Archivo…*** y no selecciono ningún archivo, al dar clic en el botón **Cancelar** del navegador de archivos. Aparece la advertencia.

Solución: Vuelva a dar clic en **Cargar Archivo…** y elija un archivo.uea.

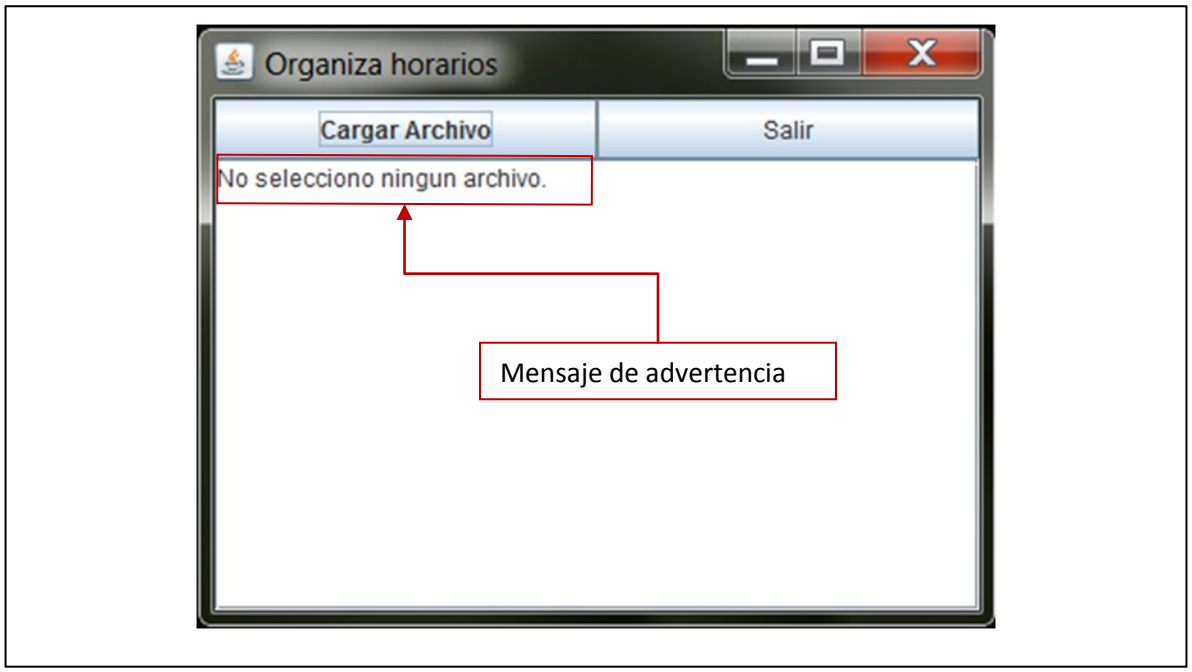

Ilustración 27. Mensaje de advertencia en *organiza.jar*

3) **Uea repetida en el archivo.uea:** En *organiza.jar* se muestra el mensaje "**revise las materias y los horarios…**"

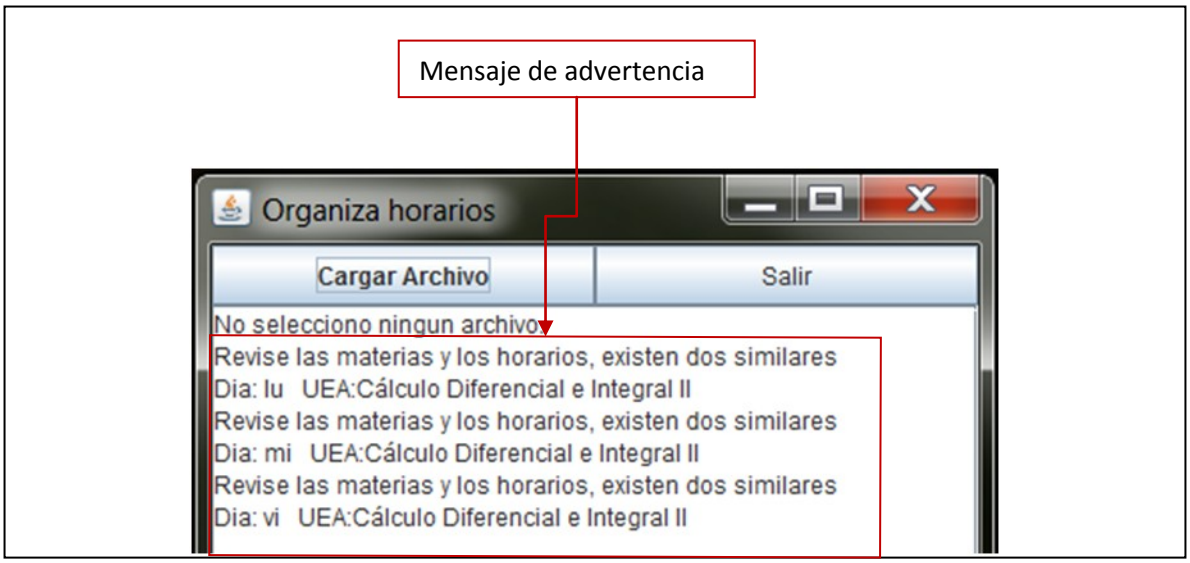

Ilustración 28. Mensaje de uea's repetidas en *organiza.jar*

Causa: El archivo.uea que contiene los horarios tiene una o algunas uea's repetidas. Esto quiere decir que se imparten a la misma hora y el mismo día.

Solución: Debe quitar las uea's que están repetidas, no tiene sentido que coloque dos uea's en el mismo día y hora. Si puede elegir manualmente entre una u otra.

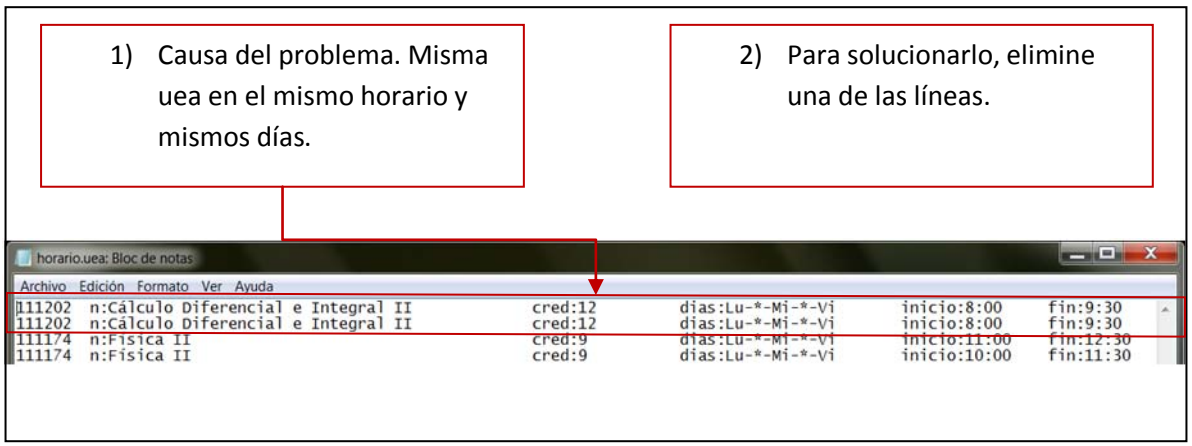

Ilustración 29. Solución de uea repetida

En general, pudieran surgir otros problemas que no estén contemplados en este manual. Las aplicaciones son una versión no concluida en la interfaz. Pueden existir casos especiales de prueba que no hayan sido contemplados al realizar el organizador de horarios.

Requisitos mínimos del sistema

Procesador: 800 MHz

Memoria RAM: 512 MB

Disco Duro: 1 MB*

Software: Maquina virtual de Java

Sistema operativo: Windows o Linux**

* El espacio para almacenar las aplicaciones en el disco duro es mínimo.

**No se ha experimentado en otra distribución Linux distinta a Ubuntu.

Glosario

Clave: Cada asignatura tiene un número compuesto por seis dígitos que la identifica.

Comando: Línea de órdenes que se desea ejecute la computadora.

Consola: Interfaz de línea de comandos, permite al usuario dar instrucciones a la computadora.

Créditos: Es el puntaje que recibe el alumno al aprobar una uea. Estos son acumulables.

Editor texto plano: Es un programa que permite editar texto sin ningún formato.

Jar: Son las iníciales de **J**ava **Ar**chive, permite ejecutar aplicaciones programadas en lenguaje Java.

Java: Lenguaje de programación orientada a objetos.

Linux: Sistema operativo libre.

Maquina virtual: Software que permite ejecutar programas emulando una computadora.

Paquete: Es un conjunto de programas, los programas que se instalan en Linux son llamados así.

Plantilla: Es un formato general que permite llevar duplicados similares.

Terminal: Es una consola, programa instalado en Linux.

UEA: Las asignaturas en la UAM son llamadas por las iníciales de Unidad de Enseñanza Aprendizaje.

Bibliografía

Agustín Froufe Quintas, "Java 2: manual de usuario y tutorial"
Alafaomega 3ed. (2003)

Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Propuesta de proyecto terminal

Software para la organización del horario del alumno por medio de la coloración de grafos

Alumno: *Yáñez Castillo José Alberto*

Matrícula: 206300036

Firma:

Trimestre: Primavera 2010

Fecha de entrega de la propuesta: 9 de julio del 2010

Asesora: *M. en C. Rafaela Blanca Silva López*

Número económico: 17114

Firma:

Índice

Objetivo General	3
Objetivos Particulares	3
Antecedentes	3
Justificación	5
Descripción técnica	5
Especificación técnicas	7
Entregables	9
Calendario de trabajo	9
Recursos	11
Bibliografía	11

I. Objetivo general

Realizar un software que genere el horario escolar de un alumno por medio de un algoritmo secuencial de coloración de grafos.

II. Objetivos particulares

1. Diseñar el software para la organización de horario del alumno por medio de la coloración de grafos.
    a. Diseñar y modelar el sistema con base en el modelo UML[1].
2. Implementar el algoritmo secuencial de la coloración de grafos.
3. Implementar el sistema para la organización de horarios del alumno por medio de la coloración de grafos en lenguaje de programación JAVA.
4. Realizar pruebas al sistema, asegurando que el producto final funcione adecuadamente.
5. Documentar el sistema realizado.
6. Construir un manual de usuario para el proyecto realizado.

III. Antecedentes

Un grafo es un conjunto de puntos llamados vértices o nodos que están unidos por líneas que llevan por nombre aristas o arcos. Un grafo nos permite representar relaciones entre elementos de conjuntos.
Un ejemplo en el cual se utiliza un grafo para representar un problema es el siguiente: Existe una empresa en la que se deben asignar tareas a distintos empleados. Con base en él podemos establecer un conjunto T (tareas) y un conjunto E (empleados). Un punto importante dentro del problema es que una tarea o todas pueden ser realizadas por un mismo empleado. La posible solución es formar un grafo en donde los empleados calificados y las tareas son nodos y una arista une al empleado *e* con la tarea *t.* Las tareas serán hechas por empleados calificados con objeto de que cada uno realice a lo más una tarea, de esta forma se obtiene un grafo bipartito[2] [1]. Ver figura 1.

[1] Unified Modeling Language (Lenguaje Unificado de Modelado): Lenguaje gráfico para especificar, construir y documentar un sistema.
[2] Un grafo bipartito es aquel en el cual sus vértices se pueden separar en dos conjuntos independientes sin existir adyacencia entre dos vértices del mismo conjunto.

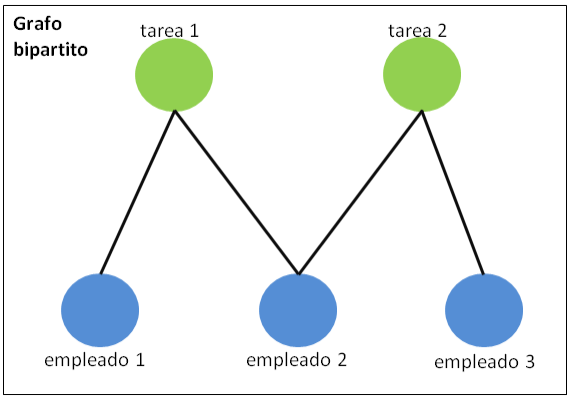

Figura 1 Grafo bipartito

Existen otros problemas donde se utilizan grafos y, específicamente, su coloración, que consiste en colorear los nodos de modo que no existan dos nodos adyacentes del mismo color. Los colores pueden verse como elementos que pertenecen a un conjunto. Uno de estos problemas consiste en la organización de horarios, donde se trata de asignar un salón de clases a una materia en un horario disponible, de manera que no ocurran empalmes de asignaturas en un mismo salón [2].

Una referencia externa similar al problema de asignación de horarios se encuentra en internet. En ella se presenta el *"*Problema de Asignación de Horarios*"*[3]. Este problema consiste en armar un horario único para el profesor y el alumno, tomando en cuenta las materias que cada alumno piensa cursar durante el periodo y que se deben impartir sin tiempos perdidos entre clases. La solución que los autores dieron al problema fue crear un grafo donde los nodos corresponden al conjunto de las materias y se relacionan mediante una arista sólo para aquellas en las que un alumno esté inscrito. Una vez creado el grafo, se le aplica un algoritmo para su coloración cuya solución se traduce en grupos de materias con el mismo color que pueden ser impartidas en una misma hora.

Por otra parte, en el artículo "Problema de asignación de horarios con coloreo de grafos"[4], el problema es analizado específicamente para el Departamento de Sistemas y Computación del Instituto Tecnológico de Cd. Madero, aunque no está implementado. Los autores proponen resolver el problema en dos pasos: primero, asignar materias a profesores por medio de la búsqueda tabú[3] y, segundo, construir una programación de horarios. Este segundo paso lo proponen resolver por medio de la coloración de grafos.

La semejanza entre los proyectos citados y el propuesto reside en la utilización de coloración de grafos para resolver un problema de asignación. Sin embargo, el planteamiento de este problema es diferente pues está pensado para los alumnos de la UAM, con lo cual se tienen las siguientes condiciones: a) las materias no deben de sobrepasar el límite de créditos permitidos en el trimestre, b) las materias se imparten en horas y días especificos, lo cual juega un papel importante en la programación del horario y c) la seriación de las mismas. El proyecto toma en cuenta las materias que el alumno ha

[3] Es una técnica heurística que puede utilizarse en combinación con algún otro método de búsqueda para resolver problema de optimización combinatoria.

cursado y aprobado de modo que, con base en éstas y las que se impartirán en un cierto trimestre junto con la seriación, el programa genere una propuesta de horario.

En cuanto a proyectos terminales reportados dentro de la Universidad Autónoma Metropolitana unidad Azcapotzalco. No se encontraron proyectos que tuvieran una propuesta similar a la presentada en este proyecto.

IV. Justificación

El sistema pretende facilitar la selección de materias y su organización en un horario. El programa generará automáticamente el horario más adecuado dependiendo del avance curricular del alumno. La propuesta de horario generada será la más conveniente en cuanto al aprovechamiento del tiempo dentro de la universidad con el fin de que el alumno termine sus materias dentro del tiempo establecido.

A futuro, este proyecto se puede ampliar para que, además de organizar el horario del alumno, tenga en cuenta el horario de los profesores y la demanda de materias en un determinado trimestre. De este modo, considerando todos estos conjuntos, podrían crearse horarios accesibles de trabajo y estudio que aprovechen mejor los recursos y beneficien a profesores y alumnos.

Las características de este proyecto indican que su realización debe estar a cargo de un estudiante de Ingeniería en computación que es quien cuenta con las bases académicas necesarias, como son: conocimientos de ingeniería de software, manejo de lenguajes de programación y diseño de algoritmos, entre otros.

V. Descripción técnica

El proyecto se desarrollará por módulos de trabajo. Los componentes principales considerados en primera instancia como necesarios para que el sistema funcione son los siguientes:

1. Entrada y formato de datos.
2. Selección de materias a cursar.
3. Creación del grafo de materias.
4. Coloración del grafo de materias.
5. Presentación del horario.

La interacción entre estos componentes puede verse en el siguiente diagrama. Ver figura 2.

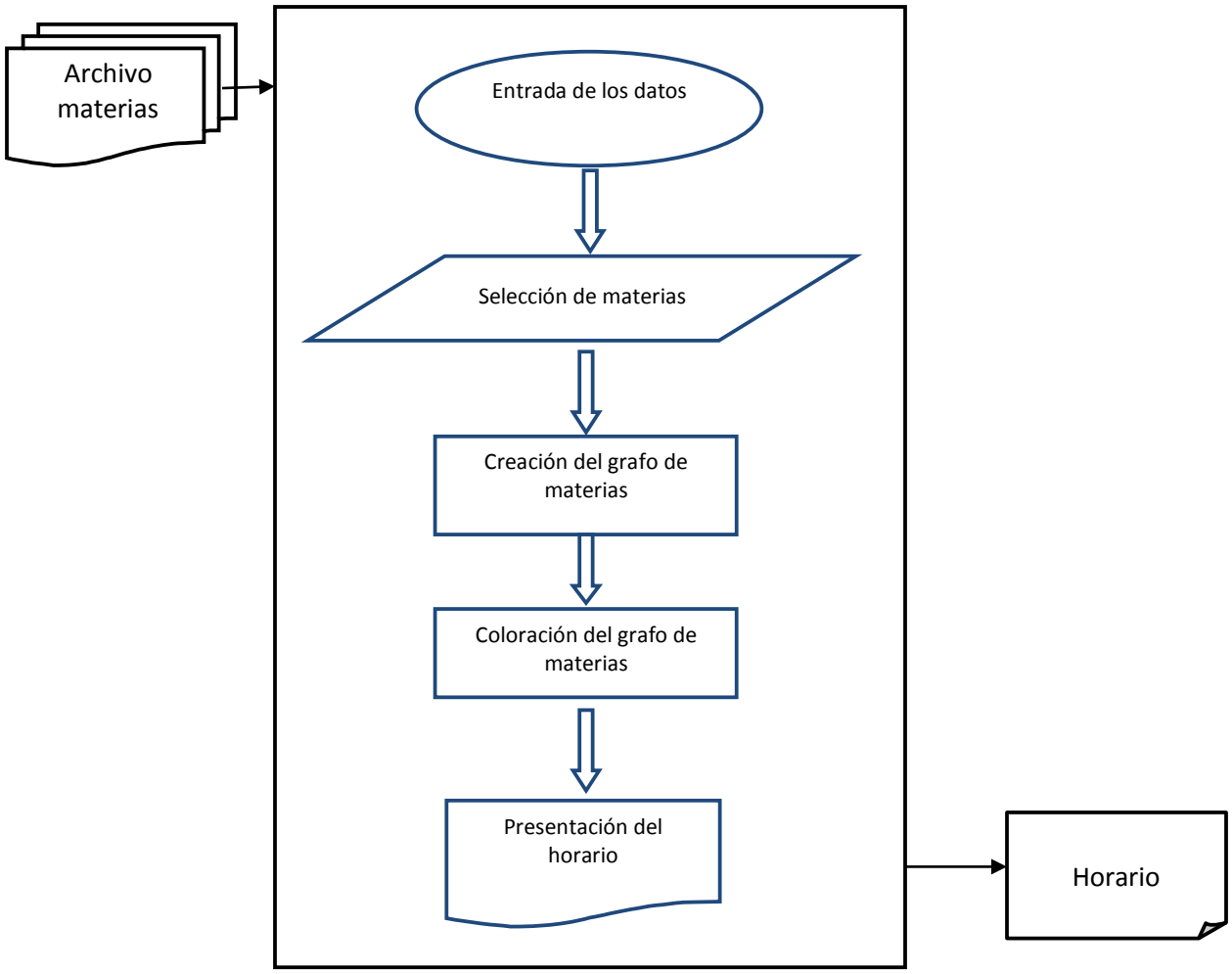

Figura 2. Diagrama de componentes

Cada una de las funciones de los componentes representados en el diagrama se describe a continuación:

1. Entrada y formato de datos: El bloque se encargará de procesar el archivo (contiene materias que el alumno cursó y aprobó) que el usuario le indique, leerá los datos y los almacenará en una estructura para su uso posterior.

2. Selección de materias a cursar: Los datos provenientes del bloque anterior se utilizan en éste y se procesan con un algoritmo que se encargará de ver cuáles materias puede cursar el alumno el siguiente trimestre. La salida serán las materias, las cuales se almacenarán en otra estructura de datos y quizá se haga un respaldo en archivo.

3. Creación del grafo de materias: La salida del bloque de selección de materias se utiliza para crear un grafo G.

4. Coloración del grafo de materias: Este bloque es considerado el más importante pues debe aplicar el algoritmo de coloración de grafos a la salida del bloque anterior

y dar la mejor solución. Se selecciona un algoritmo secuencial[4] para realizar la coloración del grafo. A continuación se presenta el algoritmo secuencial básico [5]:

- Entrada: Un grafo G de materias (nodos).
- Salida: Coloración de los nodos.
- Primer paso: Asignar el primer color al nodo v_1.
- Segundo paso: Si se han coloreado $v_1, v_2, \ldots, v_k$ con j colores, se asigna a v_{k+1} el color t, donde $t <= j+1$ es el mínimo color permitido para v_{k+1}, según los colores ya asignados a sus vecinos.

5. Presentación de horario: Una vez que todos los bloques realizaron sus tareas, éste toma los resultados y los presenta en forma que el usuario los pueda interpretar fácilmente.

VI. Especificaciones técnicas

El proyecto será desarrollado en el lenguaje de programación JAVA[5] utilizando el software Eclipse[6] sobre el sistema operativo Microsoft Windows. Ver Figura 3. Se utiliza JAVA como lenguaje de programación porque se tienen más conocimientos de éste y que el software final se pueda ejecutar en Linux o Windows, esto se logra a que solo se necesita tener instalada la máquina de virtual de java.

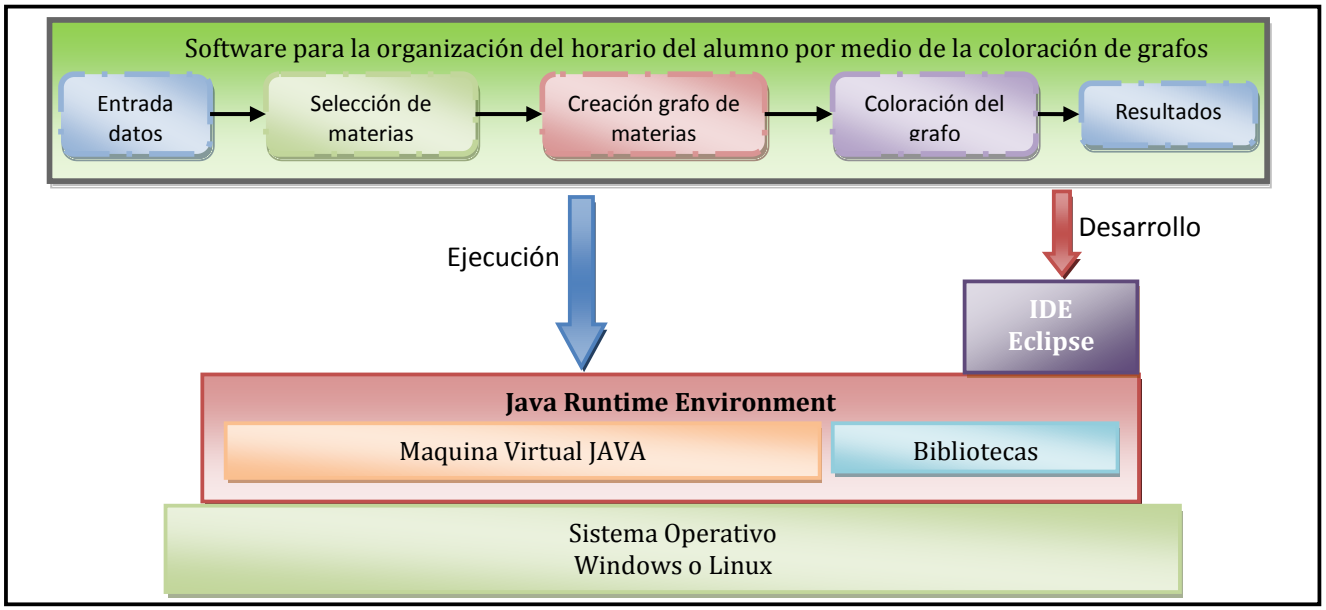

Figura 3. Diagrama a bloques de las tecnologías y componentes que conforman el proyecto.

[4] Un algoritmo secuencial lo que hace es, si se desea colorear el vértice *v*, teniendo colores ordenados numéricamente, se asigna a *v* el color más pequeño que no está entre los asignados a los vecinos de *v* ya coloreados.
[5] Lenguaje de programación orientado a objetos propiedad de Sun Mycrosystems.
[6] Entorno de desarrollo integrado de código abierto para crear aplicaciones.

Por la cantidad de trabajo que requiere este proyecto se establece un cierto orden en lo que se va a realizar y se delimita el proyecto, dividiéndolo en cuatro módulos. Los primeros dos módulos corresponden al análisis y diseño, el tercero es un análisis de la funcionalidad del algoritmo de coloración de grafos y el cuarto es esencial, pues es en éste donde se elaborarán todos los componentes mencionados en la descripción técnica. Lo que se realizará en cada módulo es lo siguiente:

1) Módulo de requerimientos.- En este módulo se realiza la obtención de requerimientos y restricciones que el sistema debe manejar, y también se obtienen los requerimientos para el usuario. Una vez obtenidos los requerimientos, se procede a su análisis y a determinar los resultados que el software debe entregar. Esta documentación se elaborará para su entrega posterior.

2) Módulo de diseño y modelado.- Una vez completados los requerimientos en este módulo, se definen las clases necesarias para el funcionamiento del sistema. Aquí, también se elaboran los diagramas de clases, de estado y de secuencia y se modelarán los casos de uso. Toda esta documentación se elaborará en su totalidad para su entrega.

3) Módulo de funcionalidad.- Con base en los requisitos del sistema se implementa una primera versión del algoritmo de coloración y se analiza la funcionalidad para el sistema propuesto.

4) Módulo de implementación.- Completados los tres módulos anteriores, se realiza la programación y la integración de los componentes del sistema que han sido mencionados en la descripción técnica. Cada componente debe cumplir con lo siguiente:

    1. Entrada y formato de datos: Este componente debe, como mínimo, leer desde un archivo de texto plano las materias que el alumno ha cursado y almacenarlas en una estructura de datos si el formato del archivo de texto es el permitido. El componente pudiera ser mejorado para que lea desde otros tipos de archivo o de una interfaz interactiva, pero esto no se hará en este proyecto.

    2. Selección de materias a cursar: Por medio de un algoritmo se deben seleccionar las materias que el alumno puede cursar el siguiente trimestre. Aún no se ha decidido cuál será este algoritmo, pero se supone que será uno de menor dificultad que el de coloración de grafos. En su caso se diseñará y programará con los conocimientos adquiridos.

    3. Creación del grafo de materias: Tomando el conjunto de las materias a cursar que se consiguió en el componente “Selección de materias a cursar”, se crea un grafo donde los nodos corresponden a las materias y son enlazados por medio de una arista si cumplen ciertas propiedades

como en los ejemplos descritos en Antecedentes. Este componente también se programará.

4. Coloración del grafo de materias: Este componente es un algoritmo más que se programará y que, teniendo como entrada un grafo de N nodos, al aplicar la coloración de grafos tendrá como salida la solución al problema propuesto.

5. Presentación del horario: El componente se encargará de mostrar los resultados generados por los componentes anteriores en un formato que el usuario pueda comprender fácilmente.

VII. Entregables

- Documentación obtenida en el análisis.
- Documentación hecha en el modelado.
- Código fuente de los algoritmos correspondientes a los componentes "Selección de materias a cursar" y "Coloración del grafo de materias".
- Aplicación funcionando.
- Manual de usuario.
- Reporte final.

VIII. Calendario de trabajo

La fecha de realización para el PT[7]01 será en el trimestre 2010 otoño. Las actividades programadas para este son consideradas como las adecuadas para los nueve créditos de la uea. El PT02 se programa para el trimestre 2011 invierno. El PT02 por tener una cantidad de 18 créditos se le han asignado actividades de mucho mayor peso a comparación del PT01. El plan de trabajo para ambos trimestres se muestra a continuación:

[7] Abreviación de la uea proyecto terminal=PT

Plan de trabajo Proyecto Terminal 1	
Actividad	**Semana**
Establecer los requerimientos del sistema	1ª 2ª
Analizar los requerimientos establecido para el sistema.	3ª
Diseñar el sistema en base al modelo UML.	4ª 5ª
Modelar el sistema en base al modelo UML.	6ª 7ª
Programar el algoritmo de la coloración de grafos en el sistema de generación de horarios de un alumno. Primera versión	8ª 9ª
Revisar la funcionalidad de la primera versión del algoritmo con respecto a lo requerido por el sistema. Si es necesario rediseñar e implementar una segunda versión del algoritmo.	10ª
Elaborar el reporte	11ª

Plan de trabajo Proyecto Terminal 2	
Actividad	**Semana**
Programación del componente “Entrada y formato de datos”.	1ª
Programación del componente “Selección de materias a cursar”.	2ª 3ª
Programación del componente “Creación de grafo de materias”.	4ª 5ª
Integrar los tres componentes anteriores junto con el de “Coloración de grafos”. Revisar su funcionalidad.	6ª 7ª
Programar el componente “Presentación del horario”	8ª
Integración de todos los componentes construidos.	8ª
Realización de pruebas al software	9ª
Corrección de posibles errores en el software	9ª
Elaborar documentación.	10ª
Realizar el manual de usuario	10ª
Elaboración del reporte final.	11ª

IX. Recursos

Los recursos disponibles para el proyecto son:

- Software Eclipse[8]
- Software JAVA Development Kit[9]
- Software StarUML[10]

Para la realización del proyecto no se requiere adquirir ningún software o hardaware especial.

X. Bibliografía

[1] Disset, V Luis, "Clase 22 Teoría de grafos" en *Matemática Discreta*, en http://www.mat.puc.cl/~ldissett/cursos/iic2252-032/clase22.pdf
Revisada el14/mayo/2010

[2] Parada Daza, Víctor, Gestión de proyectos informáticos del Departamento de Ingeniería en Informática de la Universidad de Santiago de Chile, en http://www.algovidea.cl/algovidea/index.php?option=com_content&view=article&id=50&Itemid=60
Revisada el 14/mayo/2010

[3] Arraz Ordoñez, Alvaro y Evaristo Cuesta Guzmán, "Problema de asignación de horarios" en *Fundamentos de la teoría de grafos*, en
http://www.sol.com/algoritmos/index.asp
Revisada el 9/mayo/2010

[4] Cruz Reyes, Laura et. al , "El problema de la programación de horarios con coloreo de grafos", en
http://prometeo.cic.ipn.mx/2002/papers/m06.pdf
Revisada el 18/mayo/2010

[5] Muñoz Rodríguez, Javier "Coloración de grafos" Trabajo de fin de carrera en
http://www.dma.fi.upm.es/gregorio/grafos/ColorListasJuego/descargas/Proyecto%20fin%20de%20carrera.pdf
Revisada el 28/junio/2010

[8] Software IDE gratuito, disponible en la página web: http://www.eclipse.org/downloads/
[9] Software gratuito, disponible en la página web: http://java.sun.com/javase/downloads/
[10] Software gratuito, disponible en la página web: http://staruml.sourceforge.net/en/download.php

Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Reporte del Proyecto terminal 1

Software para la organización del horario del alumno por medio de la coloración de grafos

Alumno: *Yáñez Castillo José Alberto*

Matrícula: 206300036

Asesora: *M. en C. Rafaela Blanca Silva López*

Número económico: 17114

Trimestre: Otoño del 2010

Índice

Objetivo……..2

Descripción……..2

Desarrollo…….2

1. Establecer los requerimientos del sistema……..2

2. Modelado de los casos de uso…….4

3. Definición y análisis de clases…….5

4. Diseño del sistema……5

5. Programación del componente selección de materias……5

Conclusiones……7

Bibliografía……..7

Objetivo.-

Describir el desarrollo de las actividades que se llevaron a cabo para cubrir el plan de trabajo para el proyecto terminal uno.

Descripción.-

En la propuesta del proyecto terminal "Software para la organización del horario del alumno por medio de la coloración de grafos" se describieron cuatro módulos de trabajo. Estos son:

1) Modulo de requerimientos
2) Modulo de diseño y modelado
3) Modulo de funcionalidad
4) Modulo de implementación

Así mismo se propuso el plan de trabajo que consta de once semanas de trabajo. El calendario muestra las actividades a cubrir cada semana.

Los tres primeros módulos de trabajo son parte de la primera etapa del proyecto terminal.

Desarrollo.-

1. Establecer los requerimientos del sistema.

Antes de comenzar a establecer los requerimientos del sistema se realiza una *Descripción más amplia sobre el problema de organización del horario*[1] de un alumno en la UAM-A[2]. Con la descripción anterior y analizándola se obtienen los requerimientos del usuario, esto es, la necesidad del usuario que se desea cubrir con el sistema a elaborar. Los requerimientos funcionales son también un resultado importante del análisis del problema.

Se mencionan los resultados obtenidos a partir del primer análisis del problema:

1.1 Definición del requerimiento del usuario.

El SOHA-CG[3] debe elaborar automáticamente una propuesta del horario de un alumno en base al plan de estudio y las materias aprobadas por el alumno.

[1] Esta descripción puede ser consultada en el documento de "Especificación de requerimientos"
[2] UAM-A: Universidad Autónoma Metropolitana, unidad Azcapotzalco
[3] SOHA-CG son las iníciales de Software para la Organización de Horario del Alumno por medio de la Coloración de Grafos

1.2 Requerimientos funcionales.-

- Es deseado que el software pueda ejecutarse desde un sistema operativo Windows o Linux.
- La seriación de las materias para la carrera de ingeniería en computación debe ser conocida por el SOHA-CG.
- Al ser ejecutado el software este requiere saber que materias han sido aprobadas por el alumno.
- Las materias aprobadas por el alumno se ingresan al SOHA-CG y se almacenan dentro del software.
- La forma en que el usuario ingresan las materias (datos de entrada para el software) puede ser: introduciendo las claves de cada materia una por una o bien ingresar todo el conjunto de materias desde un archivo de texto plano, este deberá tener un formato específico.
- Los datos de entrada (materias aprobadas) deben pasar por una validación, esta debería consistir en revisar que las materias pertenecen al plan de estudios.
- Si las materias no son validas se deben de corregir.
- Una vez que se validaron las materias aprobadas por el alumno, el software seleccionará las materias que el alumno puede cursar.
- El software utilizará el algoritmo secuencial para la coloración de grafos. Con el algoritmo como herramienta para organizar las materias previamente seleccionadas se pretende organizar un horario.
- El formato que deberá tener el horario (datos de salida que el software entrega) debe ser interpretado fácilmente por el usuario.

1.3 No funcionales.-

Interfaz de usuario:

Para la primera etapa de este proyecto no se contempla interfaz de usuario. El software solo se ejecuta en una ventana sencilla. El usuario solo podrá ingresar los datos solicitados por el programa y recibirá como salida el horario propuesto.

- No se implementará interfaz para este software.
- Se le mostrarán mensajes de aviso al usuario indicando el procedimiento que se está realizando.
- Si existe algún error se le indica al usuario por medio de un aviso.

Hardware:

- Se pretende que el hardware necesario para ejecutar la aplicación sea mínimo.

Software:

- Para ejecutar la aplicación el usuario debe tener instalada la maquina virtual de JAVA.

Eficiencia:

El nivel de desempeño para el software se supone debe ser de un nivel medio, no se sabe si alcanzará un nivel alto pues el algoritmo a utilizar estará a prueba en este software.

- Es posible que el sistema no encuentre el horario más adecuado, debido a la poca oferta de materias ofrecidas durante el trimestre en curso.
- El tiempo para que el software obtenga una solución adecuada es desconocido. Conforme se avance en el proyecto y se realicen las pruebas correspondientes se podrá ampliar más este punto.

Con el análisis de requerimientos funcionales se obtienen los primeros casos de uso[4] para el sistema. A continuación se enlistan los casos de uso:

- Ingresar materias
- Corregir materias
- Generar propuesta de horario
- Mostrar horario
- Desplegar en pantalla
- Guardar archivo

2. Modelado de los casos de uso

Los seis casos de uso obtenidos son documentados, esto es, el flujo de operaciones que un usuario del sistema común realizaría para manejar el sistema. Con lo anterior se consigue una secuencia de pasos lógica que cubre los requerimientos funcionales.

A cada caso de uso documentado se le realiza su correspondiente *Diagrama de actividad*[5]. Los diagramas de actividad son una representación grafica de los pasos a seguir dentro de los casos de uso, con ellos podemos darnos cuenta si el sistema tendrá una secuencia correcta.

[4] El diagrama correspondiente a los casos de uso junto con la documentación en los casos de uso puede ser consultada en el documento de "Especificación de requerimientos"

[5] Los diagramas de actividad correspondientes a los casos de uso puede ser consultada en el documento de "Especificación de requerimientos"

3. Definición y análisis de clases

Para la obtención de las posibles clases candidatas se analizo el problema. La forma de seleccionarlas fue tomar los sustantivos que se encontraron en la descripción del problema. Para separar las clases definitivas de las candidatas, se auxilia de la documentación de los casos de uso y de los diagramas de actividad. De entre una lista no muy amplia de clases candidatas se eligen solo cuatro, que son:

- Alumno
- Horario
- Plan de estudios
- UEA

Estas clases son las básicas para comenzar el diseño del sistema. Cada una de las clases cuenta con sus atributos definidos. Aunque, no se han definido métodos para estas clases ya que no se tiene claro la forma de hacer el "match" entre el algoritmo de selección de materias y el algoritmo de coloración de grafos.

4. Diseño del sistema

Los diagramas de secuencia[6] son la herramienta a utilizar para el diseño del sistema. Estos diagramas de secuencia se basan en los casos de uso mencionados con anterioridad. Los diagramas de secuencia implementados son:

- Diagrama de secuencia, caso de uso 1: Ingresar materias
- Diagrama de secuencia, caso de uso 2: Corregir materias ingresadas
- Diagrama de secuencia, caso de uso 3: Generar propuesta de horario
- Diagrama de secuencia, caso de uso 5 y 6: Desplegar en pantalla y Guardar en archivo

El diagrama de secuencia para el caso de uso 6 no fue realizado ya que se prefiere primero investigar e implementar una primera versión del algoritmo de coloración de grafos.

5. Programación del componente selección de materias

Conforme se fue avanzando en el diseño y análisis del sistema. Se noto que antes de realizar una primera versión de un algoritmo de coloración de grafos para la organización

[6] La representación de los diagramas de secuencia se pueden encontrar en el documento de "Especificación de requerimientos"

de horarios, era necesario programar el componente para seleccionar las materias que el alumno puede cursar. Ya que si no se tiene una lista de materias a cursar, pues no hay datos de entrada para el algoritmo de coloración de grafos.

5.1 Formato del archivo de entrada para la selección de materias

El algoritmo de selección de materias requiere saber cuáles son las materias correspondientes al plan de estudio de la carrera de ingeniería en computación, para ello se carga un archivo de texto plano con el siguiente formato:

```
//Para intorudcir comentarios utilizar doble diagonal => //
//TroncoGeneral Total de crditos en este nivel  144
111178      s:*   n:Introducción a la Física    corregistro:*     cred:4
111226      s:*   n:Taller de Matemáticas       corregistro:*     cred:7
120108      s:*   n:Comprensión de Textos       corregistro:*     cred:4
```

El primer campo corresponde a la clave de la uea, deben ser seis dígitos. Cada separación esta dado por un tabulador. A continuación la letra ‘s:’ seguida de dos puntos esto indica cual es la seriación de la materia, si no tiene ninguna seriación es obligatorio colocar un asterisco ‘*’, de lo contrario se colocan la clave(s) de las materias que están seriadas. Después del siguiente tabulador se coloca ‘n:’ y el nombre de la uea a la cual pertenece la clave. Por último se tiene el campo de ‘corregistro’, que se utiliza si la materia se puede cursar con el previo corregistro de otra uea, el campo debe llenarse por la clave de la uea (seis dígitos) y ‘cred’ que corresponden a los créditos que equivale cada materia.

La elaboración de este archivo es parte de este proyecto, así que el usuario no debe de preocuparse por elaborarlo.

5.2 Implementación del componente selección de materias versión 1

Como se menciono con anterioridad no es un componente que resulte difícil de elaborar, pero si requiere de tiempo.

La implementación está a cargo de una lista ligada donde los nodos corresponden a cada uea. Cada nodo tiene como atributos:

```
private String _clave_uea;
private String _nombre;
private int _creditos;
private char _aprobada;
private String[] _seriacion;
private String _corregistro;
```

El alumno necesita ingresar las materias que tiene aprobadas, esto se hará por medio de una interfaz que le solicita ingresar sus materias aprobadas (solo se requiere la clave). Una vez que el sistema sabe cuáles son dichas materias utiliza los siguientes tres métodos:

```
//CARGAMOS EL ARCHIVO PARA LAS UEA DEL TRONCO GENERAL
```

```
            Tronco_general crearPlanTG = new Tronco_general();
            crearPlanTG.crearTronco("claves_tronco_general.txt");

//PREGUNTARLE AL LAUMNO LAS UEA QUE TIENE APROBADAS
            crearPlanTG.preguntarAprobadas();

//BUSCAR UEA QUE EL ALUMNO PUEDE CURSAR, EN BASE A LAS APROBADAS
            crearPlanTG.ueaCursar();

//DESPLEGAR LAS UEA QUE SE PUEDEN CURSAR EN BASE A LAS MATERIAS APROBADAS
            crearPlanTG.verUeaDisponible();
```

Con la lista y los tres métodos anteriores se genera una lista de las materias que el alumno puede cursar. Esta salida puede ser utilizada para el siguiente componente que es el de la organización de horarios por coloración de grafos que se diseñará en implementará en el proyecto terminal dos.

Conclusiones

En esta primera parte es importante resaltar que el sistema para la generación de horarios solo se centrara en la carrera de ingeniería en computación dentro de la UAM-A. Esto debido a que se deben de tomar en cuenta el plan de estudios y la seriación de las materias. Si se tomarán en cuenta las demás carreras, el sistema requeriría de unos módulos extra para su funcionamiento, así que por el momento se prefiere enfocarse a un solo plan de estudios.

El análisis de clases ha quedado inconcluso en cuanto a los métodos que se requerirán para el total funcionamiento del sistema. En el diseño queda un diagrama de secuencia pendiente. Esto no representa problema alguno ya que con el avance obtenido hasta este punto se ha logrado implementar el algoritmo de selección de materias, este algoritmo se encuentra funcionando correctamente para los requerimientos establecidos del sistema.

El UML que se eligió en un principio para trabajar nos permite agregar el material restante al análisis y diseño del sistema. Así mismo lograr enlazar las ideas que se tienen para la implementación del sistema.

Bibliografía

Booch, Grady, "El lenguaje unificado de modelado"
Addison-Wesley (1999)

Agustín Froufe Quintas, "Java 2: manual de usuario y tutorial"
Alafaomega 3ed. (2003)

Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Reporte del Proyecto terminal 2

Software para la organización del horario del alumno por medio de la coloración de grafos

Alumno: *Yáñez Castillo José Alberto*

Matrícula: 206300036

Asesora: *M. en C. Rafaela Blanca Silva López*

Número económico: 17114

Trimestre: Invierno del 2011

Objetivo.-

Informar sobre las actividades realizadas durante el proyecto terminal dos. Mencionar los cambios dentro de esta etapa y concluir sobre el proyecto final.

Descripción.-

El proyecto comprende cuatro módulos, el último fue realizado durante el proyecto terminal dos.

Para el ultimo, "Modulo de implementación" se hizo un cambio en cuanto al orden de programar los componentes, esto es, se programo primero el componente "Selección de materias" en lugar del algoritmo de coloración de grafos, ya que la entrada para el componente organización de horarios es saber cuáles son las materias disponibles que puede cursar el alumno. Todo lo demás fue construido de acuerdo con lo programado en el calendario.

Existe un cambio importante en el sistema, el algoritmo de coloración de grafos propuesto para resolver el problema fue sustituido. ¿Por qué cambiar el algoritmo?, el punto de este proyecto es resolver el problema de organizar el horario para un alumno y darle opciones. Por lo tanto, el problema de organización de horarios tiene restricciones fuertes que no pueden ser cubiertas por el algoritmo de coloración, una de ellas era encontrar relaciones que permitan crear un grafo bipartito y sin este, no se puede crear un grafo de todas las materias para su coloración.

Para resolver el problema se busco otra alternativa, crear relaciones entre materias que se ofrecen el mismo día y que el horario en el cual se imparte sea el más próximo a la materia anterior. Una vez realizado lo anterior, se crea un grafo para cada día de la semana, se debe encontrar el camino más largo entre todas las opciones de horarios que se tienen y almacenarla. Los caminos más largos para cada día son buenos candidatos. Por último se deben crear combinaciones con las opciones que se tienen y presentárselas al usuario.

Desarrollo.-

1. ***Selección de materias***

   Este componente fue programado en su totalidad durante el proyecto terminal uno.

   Durante el proyecto terminal dos, se le creó una interfaz gráfica muy sencilla pero que facilita la entrada de datos para el usuario. Ahora no se necesita crear un archivo de texto con las materias que aprobó el alumno. Solo se deben seleccionar las claves de

las uea's que tienen aprobadas y el sistema le muestra las materias que puede cursar el siguiente trimestre.

La pantalla principal para seleccionar las claves de las materias es la siguiente:

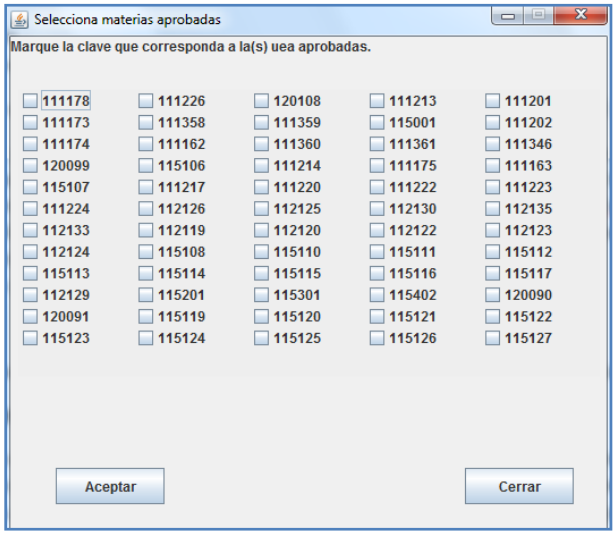

Ilustración 1 Pantalla principal de Selección de materias

En el manual de usuario se especificará más a fondo de cómo el usuario debe utilizarlo.

La implementación el componente *Selección de materias* se encuentra descrita en el reporte del proyecto terminal uno.

1.1 Implementación de la interfaz grafica para el componente selección de materias

Para crear la interfaz se hace uso de Java Swing. Se crea la clase *Seleeciona.java* que es una extensión de la clase *JFrame* esto nos facilita la creación de la interfaz. Aquí no se comenta a fondo los métodos y componentes de esa clase, puesto que se encuentra dentro el código documentado.

2. Organización de horarios

2.1 Entrada de datos

La salida que se obtiene del componente *Selección de materias* es la que se utilizará para crear las posibles opciones de horario. A esta salida se le debe dar un formato específico para que el programa pueda obtener una solución.

Supongamos que se obtiene la siguiente salida del componente *Selección de materias:*

111173 Física I
Creditos: 9

111202 Cálculo Diferencial e Integral II
Creditos: 12

Para la salida anterior se le da el siguiente formato:

111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:8:00	fin:9:30
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:11:00	fin:12:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:11:00	fin:12:30

Se puede observar que la uea con clave: 111202 tiene dos opciones de horario.
Cada línea está conformada por seis campos, de izquierda a derecha estos son: clave de la uea, nombre de la uea, créditos, días en que se imparte, inicio y fin del horario de la materia.

- **Clave**: Este campo debe colocarse los seis dígitos correspondientes a la clave de la uea.
- **Nombre**: Antes de colocar el nombre de la uea se debe anteponer los dos caracteres "*n:*" (una *n* y dos puntos). Esto para que el programa pueda identificar de que campo se trata.
- **Créditos**: La etiqueta que se antepone es "*cred:*" enseguida puede colocar el valor de créditos que tiene esta uea.
- **Días**: Se colocan las primeras dos letras para cada día en que se imparte la uea. Ejemplo si la uea se imparte lunes, miércoles y viernes. Entonces la etiqueta *días:* va seguida de *Lu-*-Mi-*-Vi*, donde "-*-" es indispensable para separar los días y con el asterisco indicar que ese día no se imparte la materia.
- **Inicio**: La hora en que comienza a impartirse la uea. Colocar la etiqueta "*inicio:*" y después la hora.
- **Fin**: Se debe colocar la hora en la que termina la clase para cierta uea. Siempre se debe colocar la etiqueta "*fin:*" antes de colocar la hora. El formato de las horas, tanto para la hora de inicio como de fin es de 24 hrs.

El archivo de los horarios disponibles de las uea's a cursar se debe guardar con la extensión "*.uea*", la única razón por la que se manejo de esta forma es para que el usuario detecte el archivo de prueba que contiene el sistema. Hasta este punto el usuario lo único que debe hacer es crear el archivo de datos de entrada para poder entregárselo al sistema y este genere las opciones de horario.

Para poder separar los campos del archivo de entrada se crea el método ***private*** *Uea_horario separarDatos(String linea)*, dentro de la clase *Manejo_Archivo_Horarios* que se encuentra en el paquete *útil*. Ver ilustración 2.

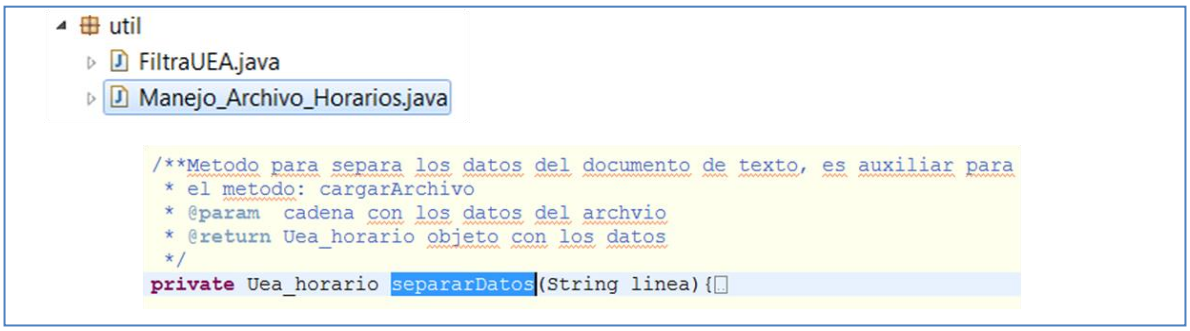

Ilustración 2 Ubicación de la clase *Manejo_Archivo_Horarios.java*

El método anterior trabaja en base al manejo de cadenas. Cada línea es separada en los atributos correspondientes para el objeto de tipo *Uea_horario.java*. Todos los objetos creados son almacenados dentro de una estructura de datos, específicamente una lista ligada. La elección de una lista ligada se debe a que podemos guardar y buscar los elementos de forma relativamente rápida y porque no se maneja una gran cantidad de ellos.

2.2 Ordenar por día y hora

Hasta este punto solo se tiene una lista de objetos de tipo *Uea_horario* almacenados dentro de una estructura de datos. El primer paso a realizar es ordenarlos por día y enseguida por hora, de esta forma se lleva un control sobre datos ordenados y se crean listas independientes para cada día de la semana. Obteniendo cinco listas como resultado.

La clase implementada para realizar el procesamiento descrito anteriormente lleva por nombre: *Ordena_dia_hora.java* está dentro del paquete *lógica.local*. Ver ilustración 3.

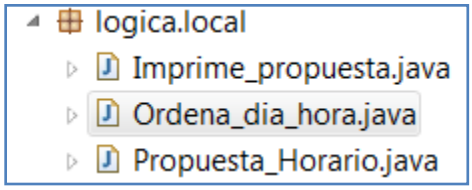

Ilustración 3 Ubicación de la clase Oredena_dia_hora.java

El ordenamiento por día se realiza de la siguiente forma, cada objeto tienen el atributo días y de esta forma solo se separan solo preguntando a que día pertenece la uea.

Para la ordenación por horas se utilizo el algoritmo de ordenamiento por burbuja, se eligió este porque su implementación es sencilla y es eficiente en este problema.

2.3 Crear la propuesta de horario

La clase más importante dentro del desarrollo es *Propuesta_Horario.java* esta es encargada de procesar las cinco listas que contiene ordenadas las materias por horario. La función que debe realizar esta clase es relacionar las materias que se pueden cursar el mismo día y casi enseguida una de otra en el horario, esta relación se realiza por parejas. En caso de que solo fuera una materia la que existe ese día, no hay ninguna a relación a crear e inmediatamente es candidata para ser cursada. En cuanto se terminan de crear las relaciones, el algoritmo debe encontrar la cadena más larga de materias, cuando hablamos de la cadena más larga nos referimos a que cada pareja de materias es enlazadas con el horario más aproximo para ese día. La cadena de materias considerada como la más óptima es la más larga, porque contiene el mayor número de asignaturas que se pueden cursar. En caso de que hubiese más de una cadena optima del mismo tamaño, es considera como otra solución al problema y se guarda como candidato.

La ubicación de la clase *Propuesta_Horario.java* se encuentra dentro del paquete *lógica.local*. Ver ilustración 4. El código se encuentra documentado, ahí puede ver la descripción de cada función.

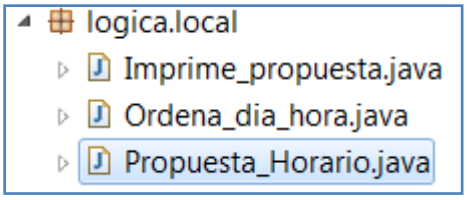

Ilustración 4. Ubicación de la clase *Propuesta_horario.java*

2.4 Mostrar los resultados

Una vez finalizada la búsqueda de las cadenas para los cinco días de la semana, el siguiente paso es crear combinaciones de los horarios para cada día. Se realiza a través de un método llamado *combina()* dentro de la clase *Imprime_propuesta.java.*en cuanto son obtenidas las posibles combinaciones los resultados obtenidos deben ser mostrados al usuario de una forma que los pueda comprender. *Imprime_propuesta.java* se encarga de dar el formato adecuado a los horarios obtenidos.

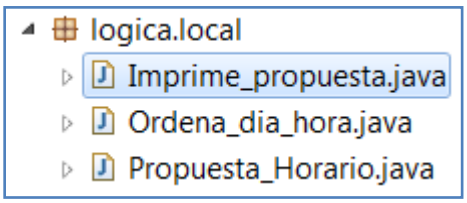

Ilustración 5. Ubicación de la clase Imprime_horario.java

3. Interfaz gráfica

El desarrollo de este proyecto no tenía contemplado implementar una interfaz gráfica que le permitiría al usuario interactuar con la aplicación. Aún con un tiempo ajustado se intento crear una interfaz básica que permita ejecutar de forma sencilla la aplicación. Utilizando la biblioteca gráfica de Java, mejor conocida como Java Swing se realizo la interfaz para la aplicación. Requirió de tiempo y trabajo puesto que no se tienen conocimientos amplios con Swing.

4. Documentación del código fuente

Java cuenta con la herramienta Javadoc, a través de esta se genero la documentación de nuestro código fuente. El formato que se obtiene en Javadoc es HTML. Esta documentación no se incluye en este reporte, pero puede ser consultada por separado en la carpeta de Javadoc.

5. Elaboración del manual de usuario

Para que el usuario pueda interactuar adecuadamente con las aplicaciones es necesario contar con un manual de usuario. La elaboración del manual se trato de hacer lo más sencilla posible, no utilizando términos tan complejos para facilitar su consulta. Se ilustro con capturas de pantalla para hacerlo más amigable.

Conclusiones

En el seminario del proyecto terminal se propuso resolver el problema de organizar el horario de un alumno por medio de la coloración de grafos, a continuación se describe cuales fueron algunos de los aspectos más importantes durante este trabajo y el resultado.

El primer objetivo consistía en diseñar y modelar el sistema, en este punto se debe hacer la corrección que no es un sistema, es una aplicación que permitirá obtener opciones de horario para un alumno de la carrera de ingeniería en computación. En base al análisis se observo que primero habría que programar el componente que le permitiera al usuario saber cuáles son las materias que puede cursar el siguiente trimestre. La aplicación de selección de materias no requiere de un algoritmo difícil de realizar pero requirió de tiempo para diseñarlo. La elaboración de estas tareas fue parte del primer proyecto terminal y solo hubo el pequeño cambio de reprogramar algunas actividades.

Con el análisis hecho en el primer proyecto sale a relucir que las restricciones del problema son fuertes y resolver el problema por medio de la coloración se volvía difícil. Se me complicaba encontrar alguna forma de crear las relaciones bipartitas, que son la base para crear el grafo al cual debía aplicarse el algoritmo de coloración. Dentro de algunas investigaciones que realice, muchos han planteado una solución pero las restricciones cambian. Una de ellas es que las materias no tienen un horario preestablecido, así que solo

se resuelve el que no existan empalmes y todos los alumnos puedan cursar sus materias. En nuestro problema es diferente, los horarios ya están establecidos, "es lo que hay". Algunos realizan una combinación de algoritmos genéticos y coloración para resolver problemas de horarios.

Si se permitiera que el programa estableciera los horarios desde un principio, se facilitaría la solución. Para no congelar el proyecto, se opto por buscar otra solución utilizando algoritmo de fuerza bruta. Es probable que no sea el mejor método pero se obtienen una solución al problema.

Al final se opto por dividir la aplicación en dos, una de ellas se encarga de ofrecerle al alumno las materias que puede cursar el siguiente trimestre y la segunda en base a los horarios establecidos ofrece opciones de horarios para el usuario.

Se cumplió el objetivo de realizar un organizador de horario pero no contaba con una interfaz sencilla que le permitiera al usuario interactuar con la aplicación. El último par de semanas se asigno un poco de tiempo para crear una pequeña interfaz. Que facilitaría la ejecución de la aplicación en un sistema con Windows o Linux.

La elaboración del manual y el reporte es un tanto breve, pero se trataron de cubrir los puntos más importantes dentro del desarrollo del proyecto.

El proyecto da para ir más allá que el simple hecho de mostrar opciones de horario a un alumno, si se continúa trabajando en este problema se puede pensar en un sistema que obtenga directamente los datos académicos del alumno desde la base de datos y con ellos saber cuáles son las materias que puede inscribir el alumno. Ofreciendo horarios donde la mayoría de los alumnos no tengan problemas para armar el propio.

Bibliografía

Agustín Froufe Quintas, "Java 2: manual de usuario y tutorial"
Alafaomega 3ed. (2003)

Universidad Autónoma Metropolitana

Unidad Azcapotzalco

Proyecto Terminal de Ingeniería en Computación

Manual de usuario

Aplicación para la organización del horario

Autor: Yáñez Castillo José Alberto

Trimestre: invierno 2011

Índice

Introducción 4
Instalación de software para ejecutar la aplicación 4
Windows 4
Linux 5
Ejecución de la aplicación 6
Ejecución de selecciona.jar 6
Ejecución de selecciona.jar desde Windows 7
Ejecución de selecciona.jar desde Linux 9
Ejecución de organiza.jar 11
Crear archivo.uea de horarios 12
Ejecución de organiza.jar desde Windows 13
Ejecución de organiza.jar desde Linux 17
Errores comunes en selecciona.jar y organiza.jar 20
Requisitos mínimos del sistema 22
Glosario 23

Tabla de ilustraciones

Ilustración 1. Instalar máquina virtual Java 4
Ilustración 2. Paquetes Java para Linux 5
Ilustración 3. Ventana inicial de *selecciona.jar* 6
Ilustración 4. Ubicación del archivo *selecciona.jar* 7
Ilustración 5. Selección de uea's en Windows 7
Ilustración 6. Resultados de la selección en Windows 8
Ilustración 7. Resultados del programa 8
Ilustración 8 Ejecutando en Linux 9
Ilustración 9. Selección de materias en Linux 9
Ilustración 10. Resultados de la selección en Linux 10
Ilustración 11. Resultados 10
Ilustración 12 Ventana principal de *Organiza horarios* 11
Ilustración 13 Ventana navegación de archivos 11
Ilustración 14. Guardando archivo *horario.uea* en Windows 13
Ilustración 15 Ejecutar *organiza.jar* en Windows 13
Ilustración 16. Cargar Archivo 14
Ilustración 17. Seleccionar archivo 14
Ilustración 18. Resultados obtenidos 15
Ilustración 19. Tercera opción de horario 16
Ilustración 20. Cerrar la aplicación. 17
Ilustración 21. Ejecutando *organiza.jar* en Linux 17
Ilustración 22. Cargar archivo desde Linux. 18
Ilustración 23. Seleccionar archivo en Linux 18
Ilustración 24. Resultados obtenidos en Linux 19
Ilustración 25. Cerrar la aplicación en Linux 19
Ilustración 26. Mensaje de advertencia en *selecciona.jar* 20
Ilustración 27. Mensaje de advertencia en *organiza.jar* 21
Ilustración 28. Mensaje de uea's repetidas en *organiza.jar* 21
Ilustración 29. Solución de uea repetida 22

Introducción

Este documento brinda ayuda en el uso de la aplicación. El software necesario para ejecutar la aplicación y muestra un ejemplo de cómo funciona el sistema.

Instalación de software para ejecutar la aplicación

Windows

Cuando la aplicación es ejecutada en el sistema operativo Windows debe asegurarse de tener instalado el software de Java, también llamado *máquina virtual Java*.
¿Cómo saber si su sistema tiene instalado Java? Si no está seguro de esto, diríjase a la dirección: http://www.java.com/es/download/ ahí se encontrará con la siguiente ventana (ilustración 1).

Ilustración 1. Instalar máquina virtual Java

Al descargar el software Java siga las instrucciones que se le presentan para su instalación, no debería tener ninguna complicación para su instalación. Si tiene problemas consulte la ayuda de Java.

Linux

En el sistema operativo Linux también se requiere tener instalada la maquina virtual de java. Para instalarla, diríjase a la dirección: http://www.java.com/es/download/ , de clic en el botón de descarga y elija el paquete que cumpla con los requisitos para su equipo. Ver ilustración 2.

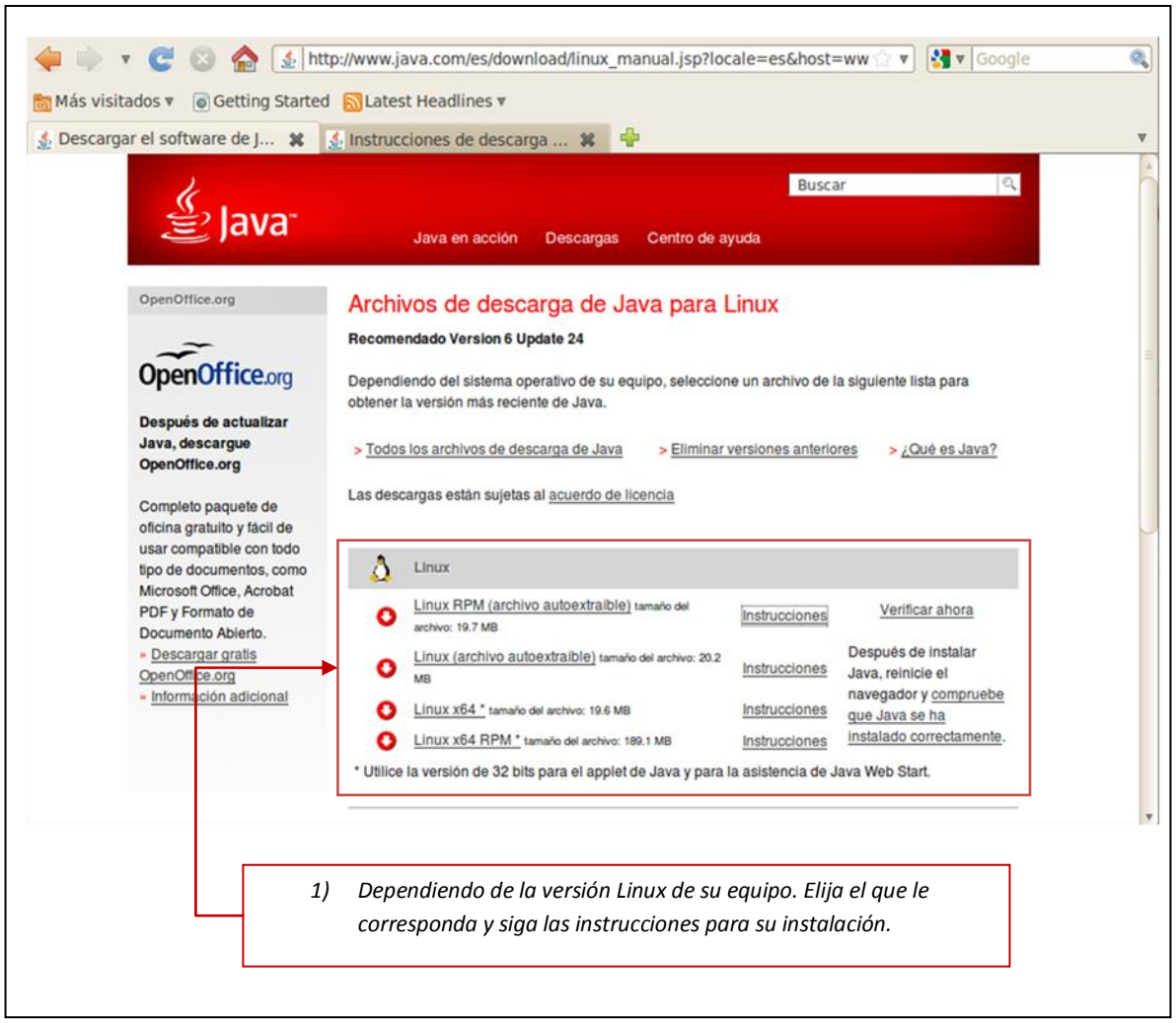

Ilustración 2. Paquetes Java para Linux

Los usuarios de este sistema suelen ser mucho más experimentados, entonces no tendrán problemas para instalarla.

Ejecución de la aplicación

La aplicación para organizar el horario esta conformada por dos pequeños aplicaciones una de ellos es ***selecciona.jar*** y el segundo es ***organiza.jar***.

Ejecución de selecciona.jar

La primera aplicación ***selecciona.*jar** presenta una interfaz donde se le muestra al usuario un conjunto de casillas de verificación, cada una está asociada a una clave de uea. El usuario debe seleccionar las que ya ha cursado para que la aplicación muestre las materias que puede cursar. Una vez que el usuario selecciono las claves que tiene aprobadas, puede dar clic en el botón **Aceptar** para obtener el resultado o bien **Cerrar** para terminar el programa.

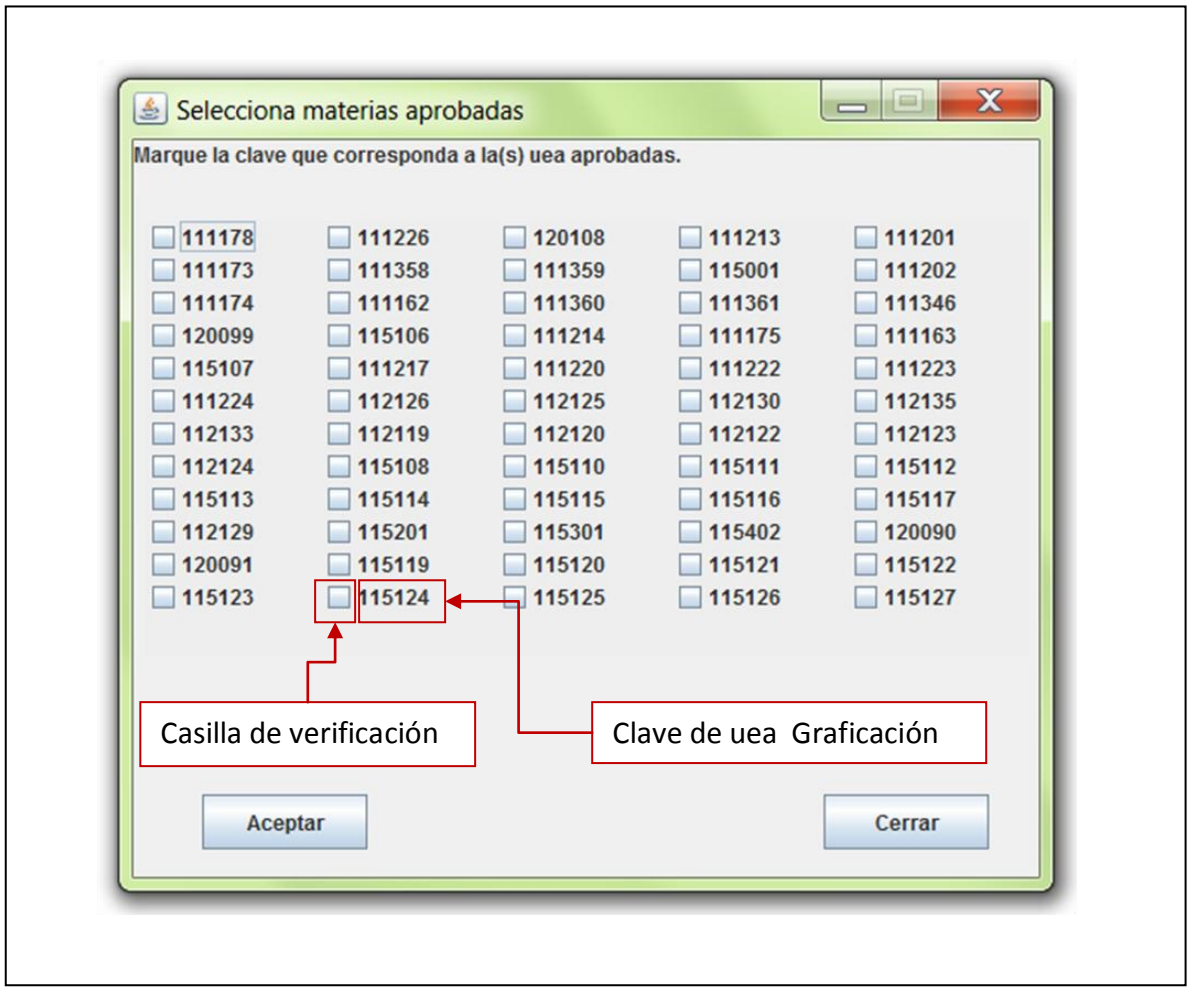

Ilustración 3. Ventana inicial de *selecciona.jar*

Ejecución de selecciona.jar desde Windows

En Windows la ejecución se realiza de forma normal sí instalo correctamente el software para ejecutar la aplicación.

1) Localice el archivo **selecciona.jar**, este debe estar donde usted lo guardo. Para ejecutarlo de doble clic sobre **selecciona.jar**. Ilustración 4.

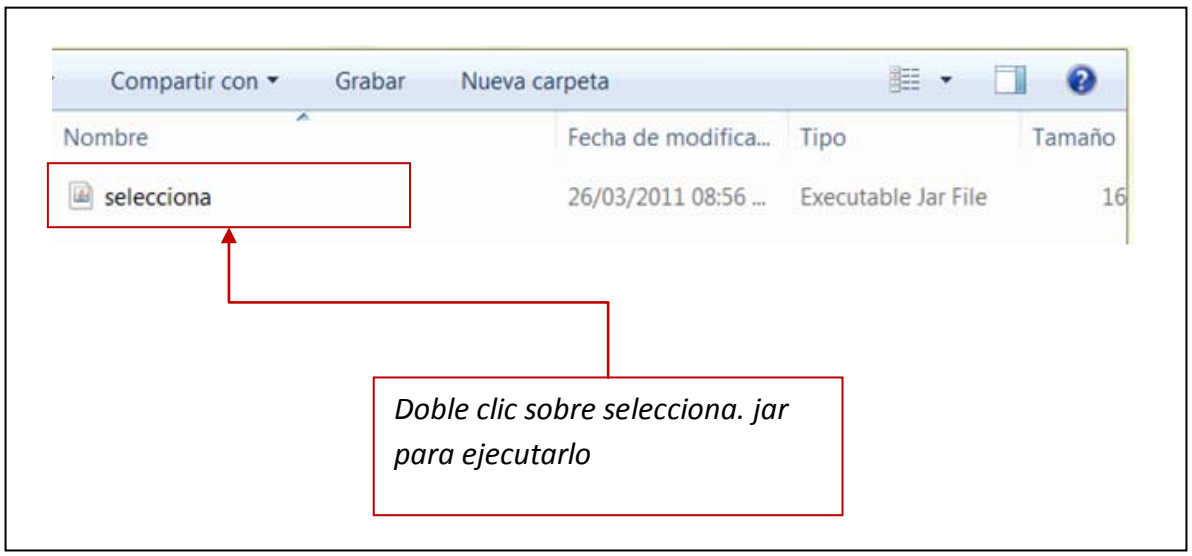

Ilustración 4. Ubicación del archivo *selecciona.jar*

2) Seleccione las claves de las uea's que tiene aprobadas. Ilustración 5.

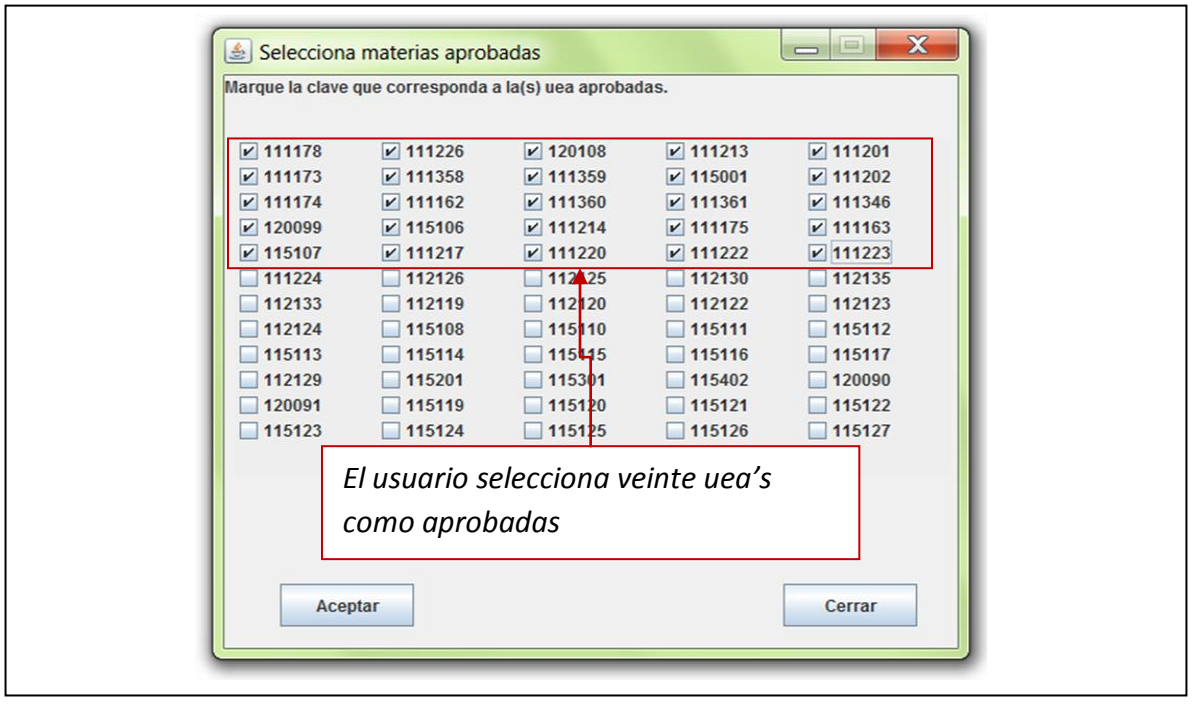

Ilustración 5. Selección de uea's en Windows

3) Dar clic en el botón **Aceptar**. Ilustración 6.
   a. Si desea corregir algunas claves de clic en el botón **Regresar**
   b. Una vez corregidas las claves puede dar clic en **Aceptar** y se muestran los nuevos resultados.

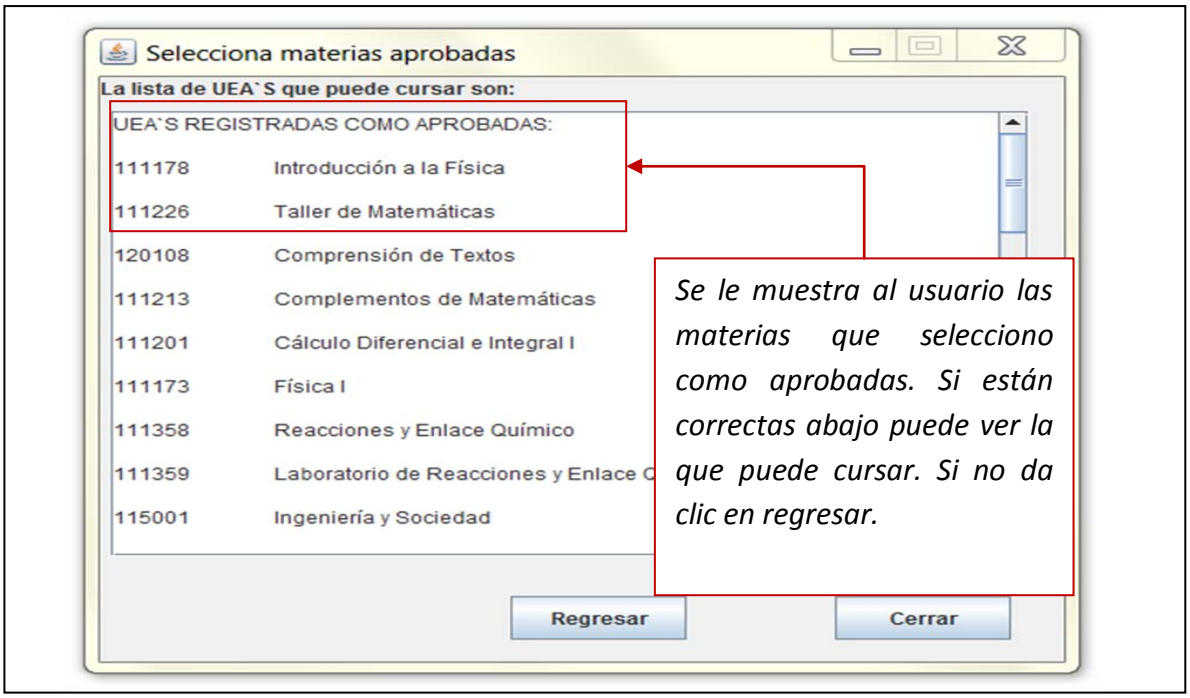

Ilustración 6. Resultados de la selección en Windows

4) Se muestran los resultados de la aplicación. Para terminar solo de clic en el botón **Cerrar.** Ilustración 7.

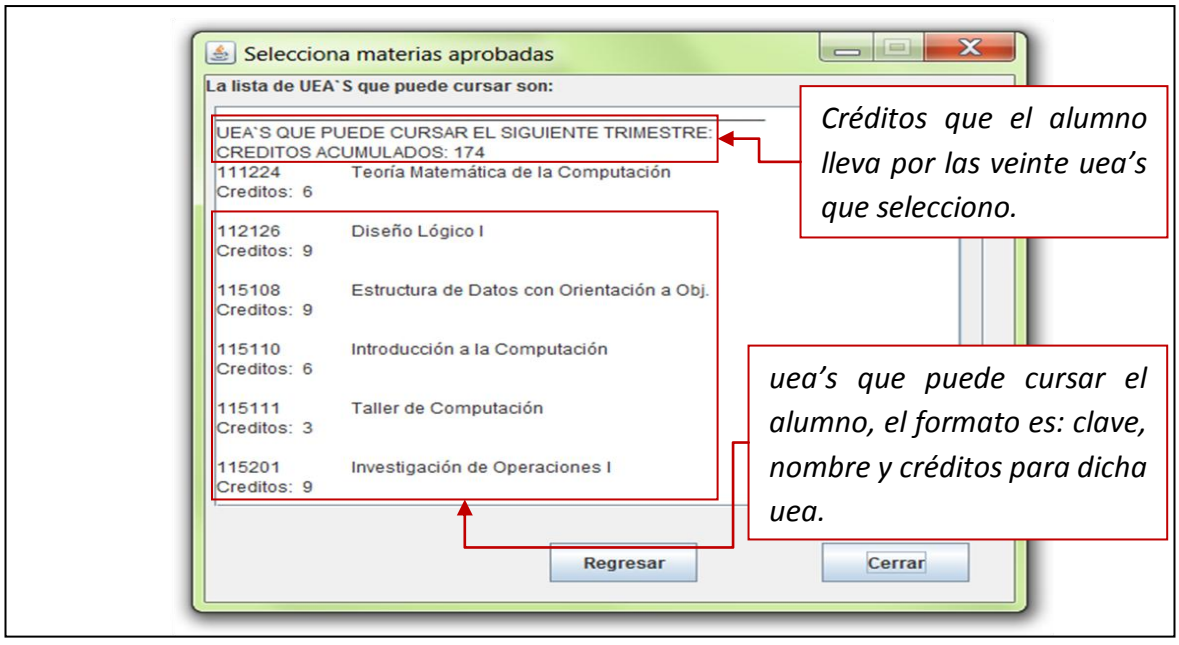

Ilustración 7. Resultados del programa

Ejecución de selecciona.jar desde Linux

Para realizar la ejecución de la aplicación ***selecciona.jar*** desde el sistema operativo Linux, se debe abrir la herramienta **Terminal** (consola de comandos). Localice el archivo ***selecciona.jar*** y vaya a ese directorio desde la terminal.

1) Teclear el siguiente comando para su ejecución: ***java –jar selecciona.jar***.
Esto ejecutara la aplicación. Ilustración 8.

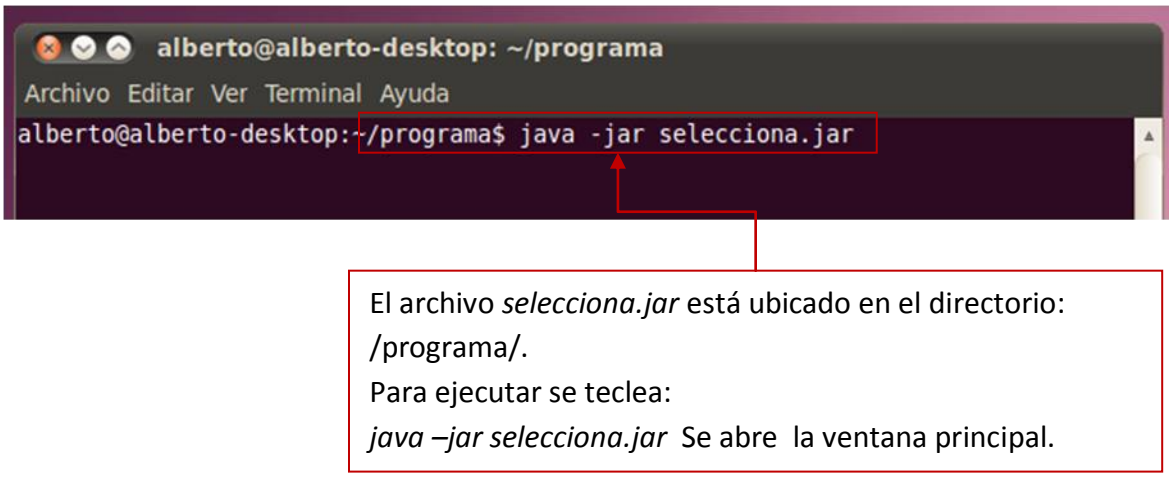

Ilustración 8 Ejecutando en Linux

2) Iniciada la aplicación seleccione las claves de las uea's que tiene aprobadas.

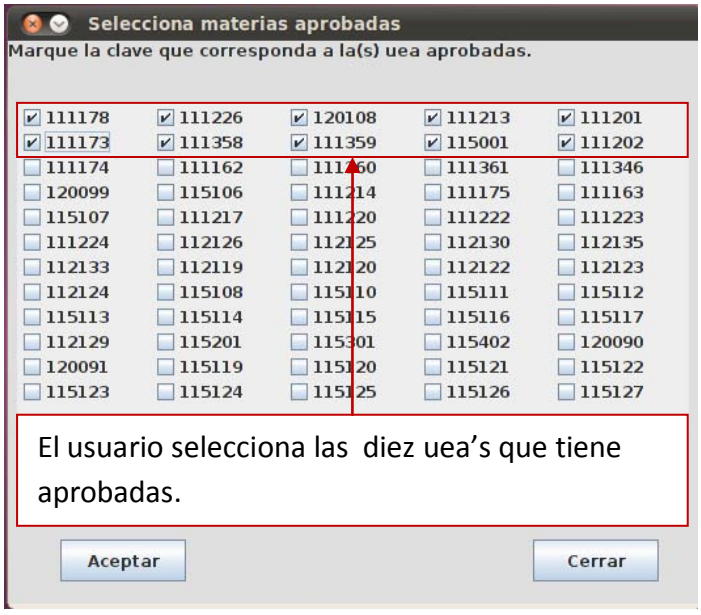

Ilustración 9. Selección de materias en Linux

3) Dar clic en el botón **Aceptar**. Ilustración 10.

a. Si desea corregir algunas claves de clic en el botón **Regresar**
b. Una vez corregidas las claves puede dar clic en **Aceptar** y se muestran los resultados.

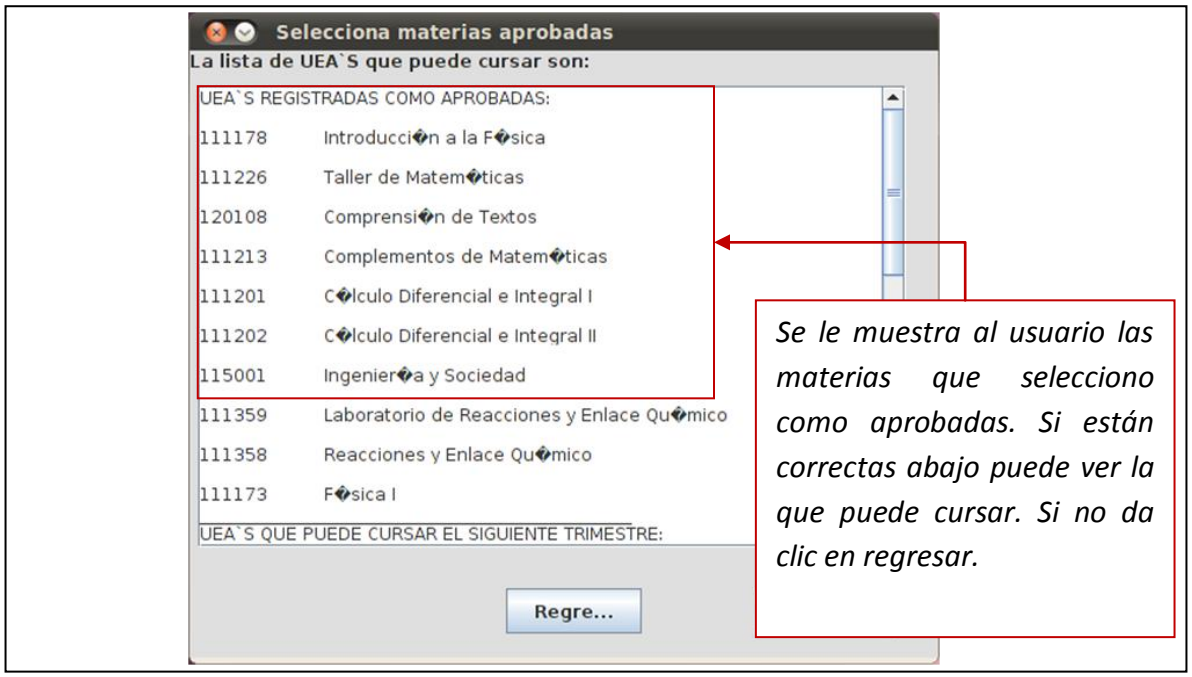

Ilustración 10. Resultados de la selección en Linux

4) Se muestran los resultados de la aplicación. Para terminar solo de clic en el botón **Cerrar.**

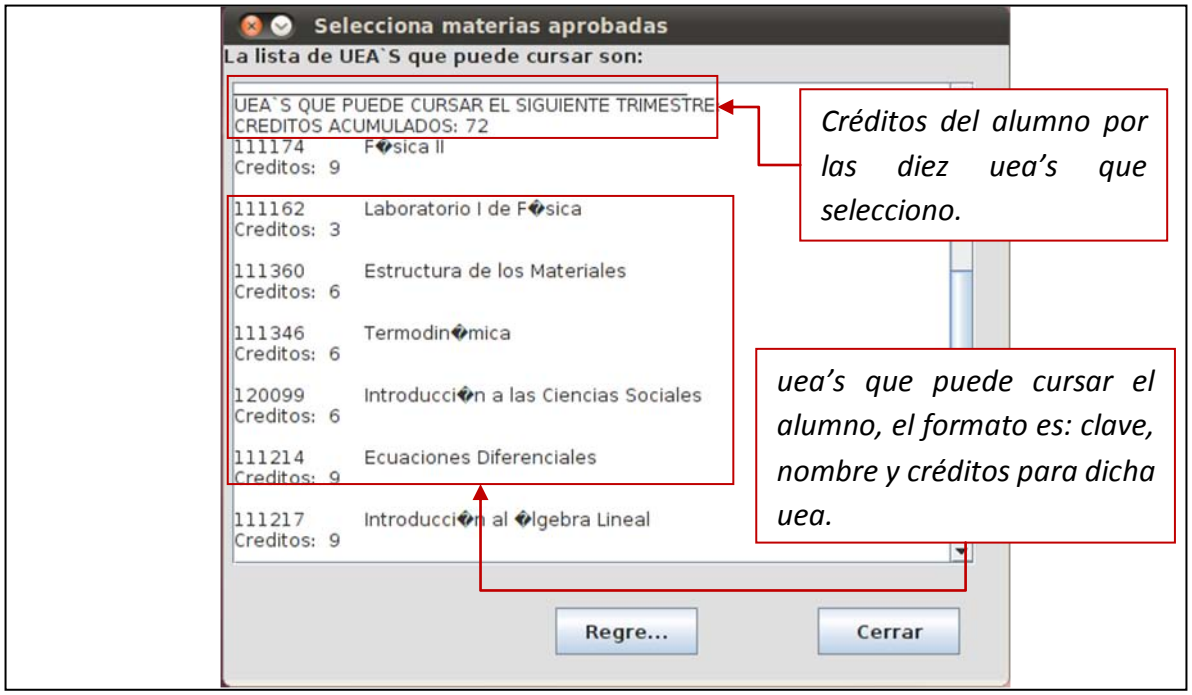

Ilustración 11. Resultados

Ejecución de organiza.jar

La segunda aplicación, ***organiza*.jar** presenta una ventana (Ilustración 12) donde solo se muestran dos botones. El primer botón tiene el nombre "**Cargar Archivo**", al dar clic en este se abre una segunda ventana (Ilustración 13) donde puede seleccionar el archivo con los horarios disponibles. Y el segundo botón "**Salir**" es para terminar la aplicación.

La aplicación solo acepta archivos con extensión ".**uea**". Estos deben ser creados desde un editor de texto plano. La forma de crearlos se muestra en el apartado "**Crear archivo.uea de horarios**"

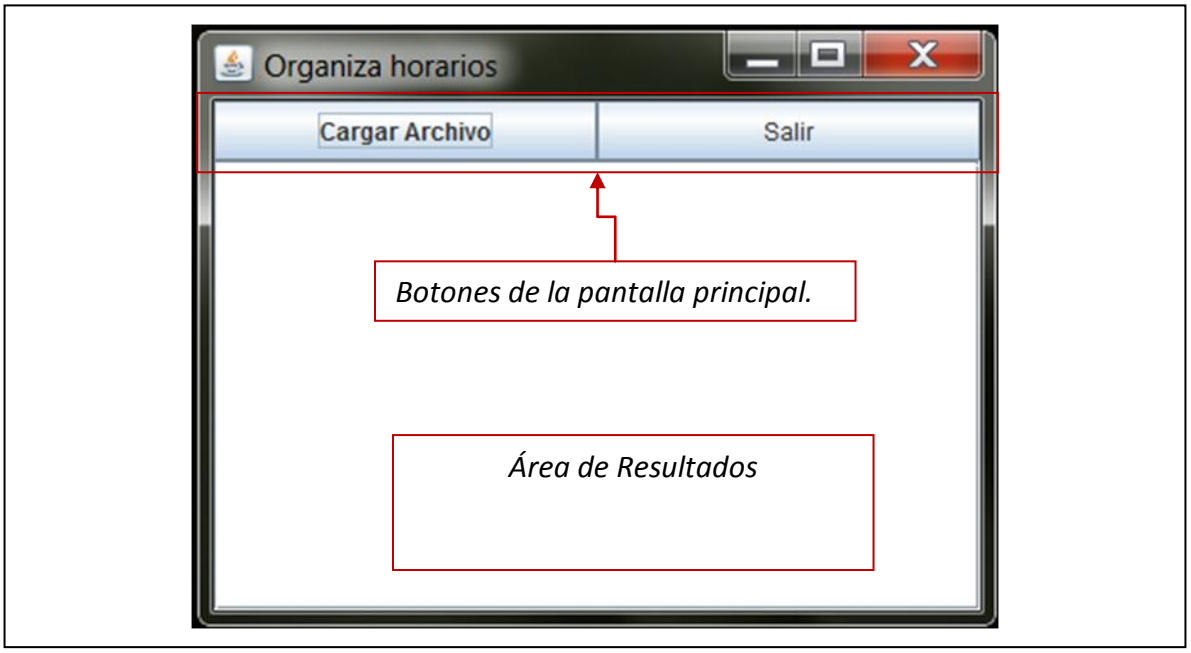

Ilustración 12 Ventana principal de *Organiza horarios*

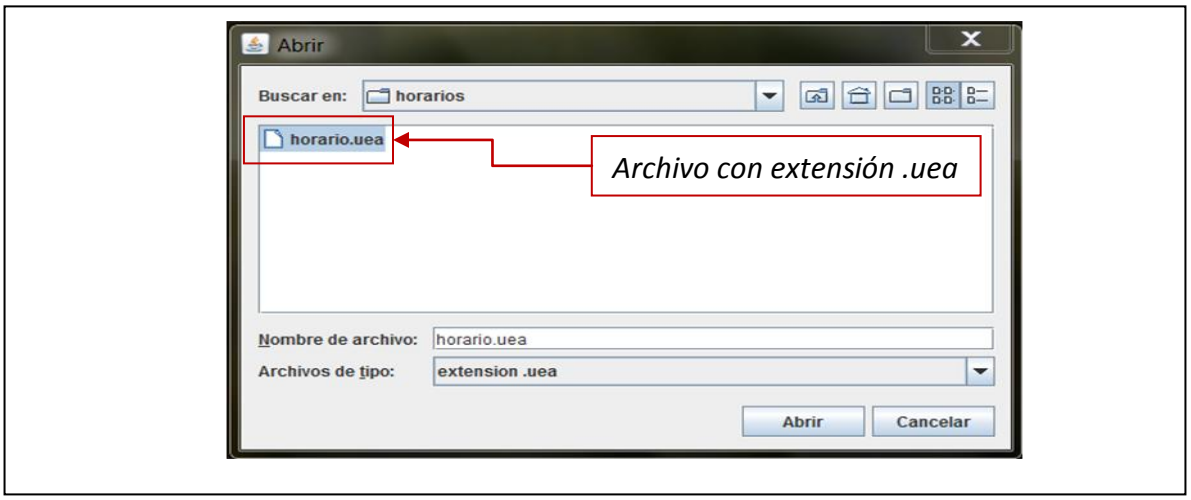

Ilustración 13 Ventana navegación de archivos

Crear archivo.uea de horarios

La aplicación contiene un archivo de demostración con el nombre ***horario.uea***, este puede servirle de plantilla para crear su propio archivo.

Para crear el archivo, realice una copia de ***horario.uea*** y ábralo con un editor de texto plano, después siga estos pasos:

1) Al abrir el archivo se observan varios campos de izquierda a derecha estos son.

    a. Clave: Compuesta de seis dígitos, todas las materias deben tener su clave.
    b. Nombre uea: Se debe anteponer "**n:**" y enseguida el nombre de la uea.
    c. Créditos: Valor numérico, debe anteponer la etiqueta "**cred:**" y el valor.
    d. Días: Los días en que se imparte la materia deben colocarse estrictamente abreviados. Ejemplo Lunes = Lu, si el martes no se da la asignatura debe colocar "**-*-**" como separador e indicando con el "*****" que no está disponible ese día y así se repite para todos los días.
    e. Hora de inicio: Cada uea tiene una hora para comenzar, esta se marca con "**inicio:**" y enseguida coloca la hora en el formato de 24hrs.
    f. Hora de termino: el final de la clase para cierta uea, se utiliza la etiqueta "**fin:**" y enseguida se coloca la hora. También en el formato de 24hrs.

2) Debe agregar los horarios disponibles de las materias que cursara. Por ejemplo, supóngase que ingresa los siguientes horarios:

Clave	**nombre uea**	**créditos**	**días**	**comienza**	**-**	**termina**
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:8:00		fin:9:30
111202	n:Cálculo Diferencial e Integral II	cred:12	dias:Lu-*-Mi-*-Vi	inicio:10:00		fin:11:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:11:00		fin:12:30
111174	n:Física II	cred:9	dias:Lu-*-Mi-*-Vi	inicio:10:00		fin:11:30
111162	n:Laboratorio I de Física	cred:3	dias:*-*-*-Ju-*	inicio:11:00		fin:14:00
111360	n:Estructura de los Materiales	cred:6	dias:*-Ma-*-Ju-*	inicio:9:00		fin:10:30
115111	n:Taller de Computación	cred:3	dias:Lu-*-Mi-*-Vi	inicio:10:00		fin:11:30
115111	n:Taller de Computación	cred:3	dias:Lu-*-Mi-*-Vi	inicio:15:00		fin:16:30

Se mantienen el formato descrito en el paso (1).

3) Lo siguiente es guardar el archivo y asegurarse que tiene extensión ***.uea***. Ver la ilustración 15.

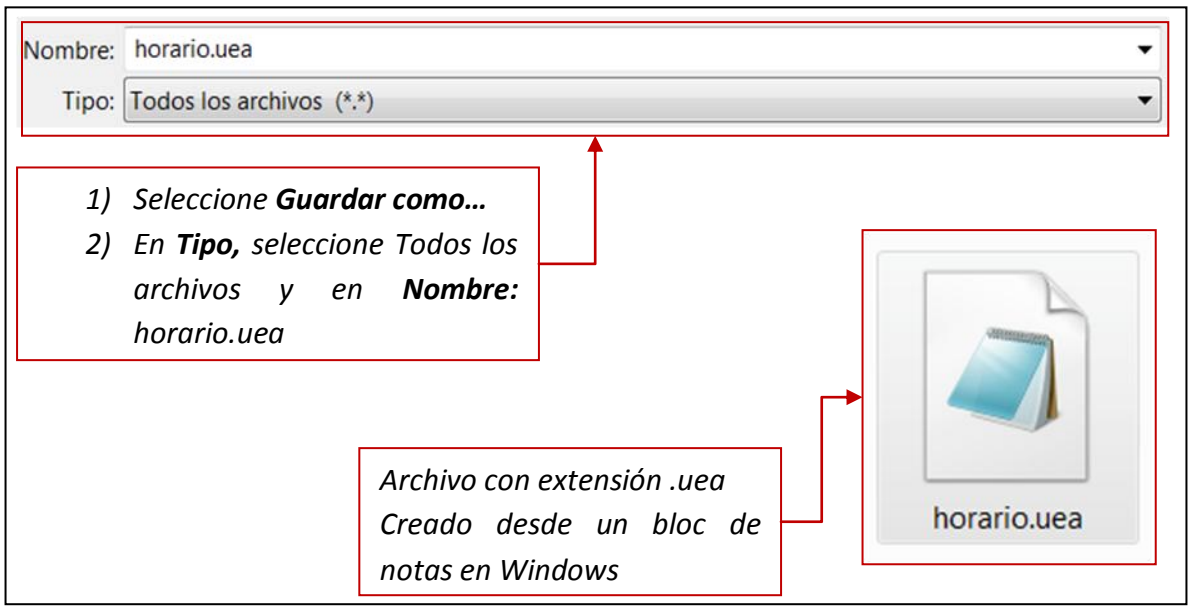

Ilustración 14. Guardando archivo *horario.uea* en Windows

Para crear el archivo en Linux, se debe seguir una secuencia similar. Puede utilizar cualquier editor de texto y salvar el archivo con la extensión (*uea*) definida anteriormente.

Ejecución de organiza.jar desde Windows

1) Para la ejecución desde Windows se realiza como cualquier programa normal, dar doble clic sobre el archivo ***organiza.jar*** iniciará la aplicación**.**

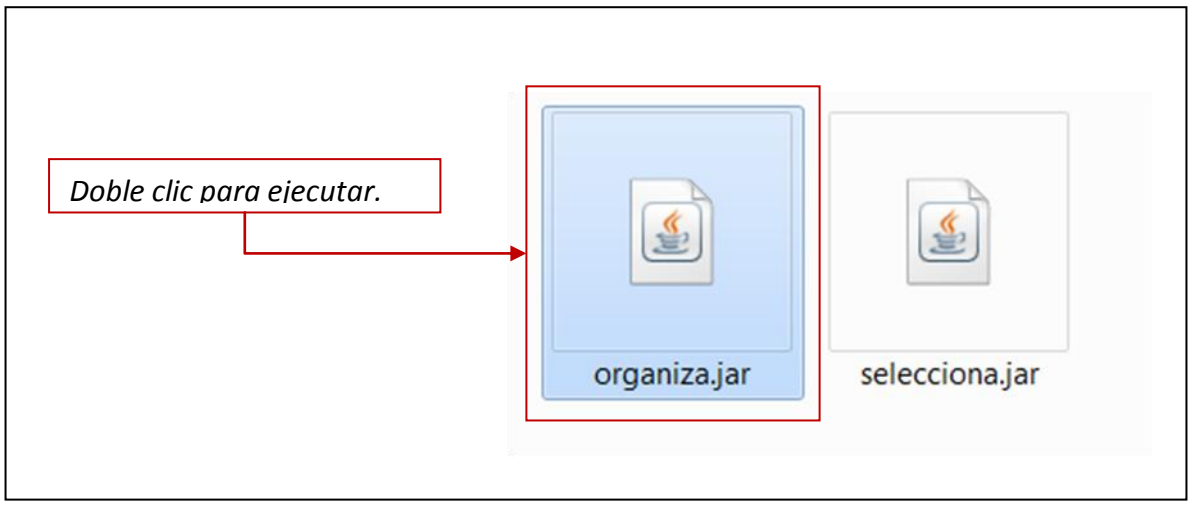

Ilustración 15 Ejecutar *organiza.jar* en Windows

2) En la ventana principal de clic en ***Cargar Archivo…*** inmediatamente se abre la segunda ventana para seleccionar un archivo. Ilustración 17.

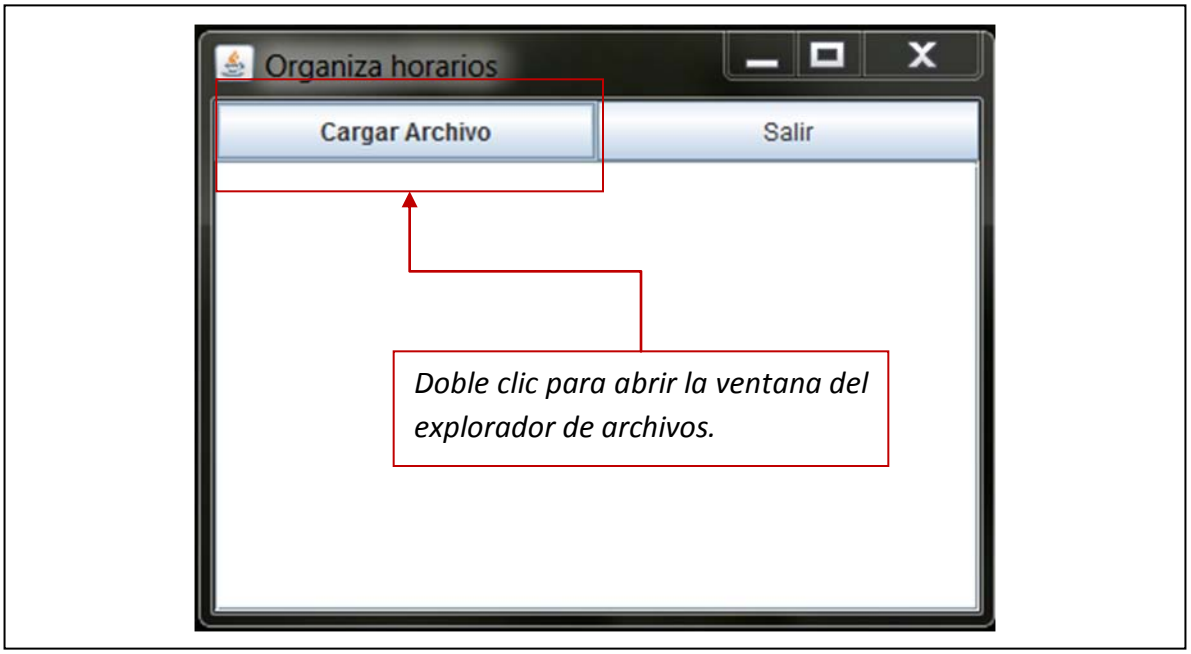

Ilustración 16. Cargar Archivo

3) En el explorador de archivos debe navegar hasta la ruta donde se encuentra el archivo con los horarios disponibles. Nota: El explorador solo reconoce archivos con extensión **.uea**, si no creo correctamente su archivo no podrá seleccionarlo.

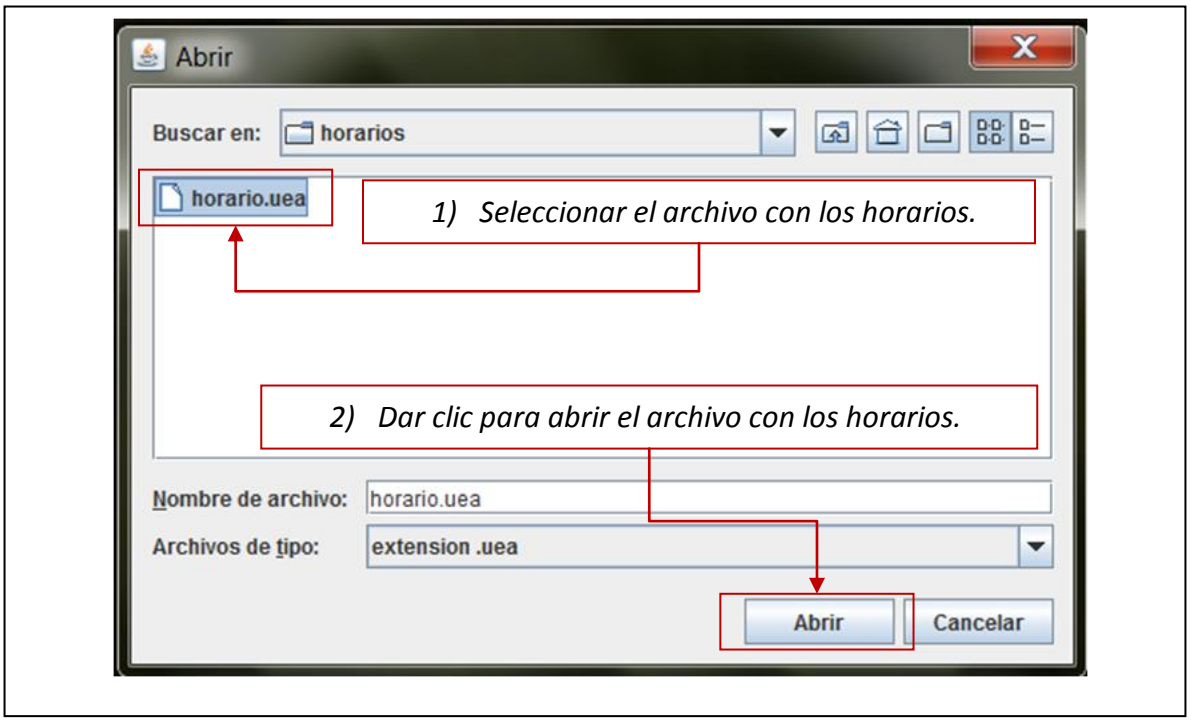

Ilustración 17. Seleccionar archivo

4) Si el archivo se cargo correctamente, se muestra la ventana con los resultados obtenidos.

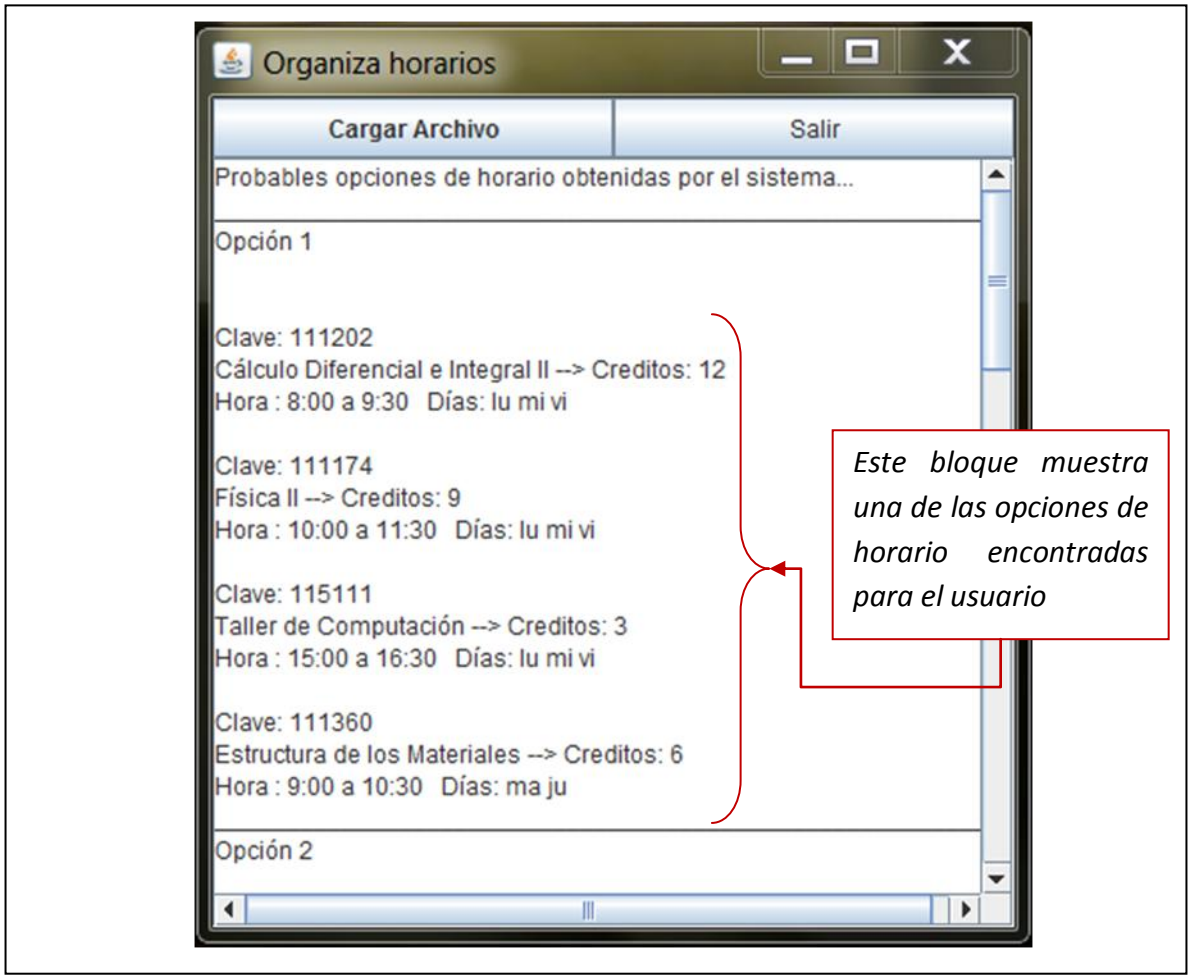

Ilustración 18. Resultados obtenidos

En la ilustración 19 se muestran los resultados de la ejecución de la aplicación. Se observa la primera opción donde el usuario puede cursar cuatro uea's. El horario sería de la siguiente forma:

Lunes	Martes	Miércoles	Jueves	Viernes
Calculo I 8:00-9:300		Calculo I 8:00-9:300		Calculo I 8:00-9:300
Física II 10:00-11:30		Física II 10:00-11:30		Física II 10:00-11:30
Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30
	Estruct. Mat. 9:00-10:30		Estruct. Mat. 9:00-10:30	

Debido a que los horarios publicados ya están programados, es muy difícil lograr un horario donde no haya huecos entre clases. Por esta razón el sistema muestra varias opciones de horario y se deja a elección del usuario la que mejor le convenga.

Veamos otra de las opciones mostradas por la aplicación, para el mismo archivo la opción tres muestra una diferencia poco significativa, pero quizá al usuario le agrade más.

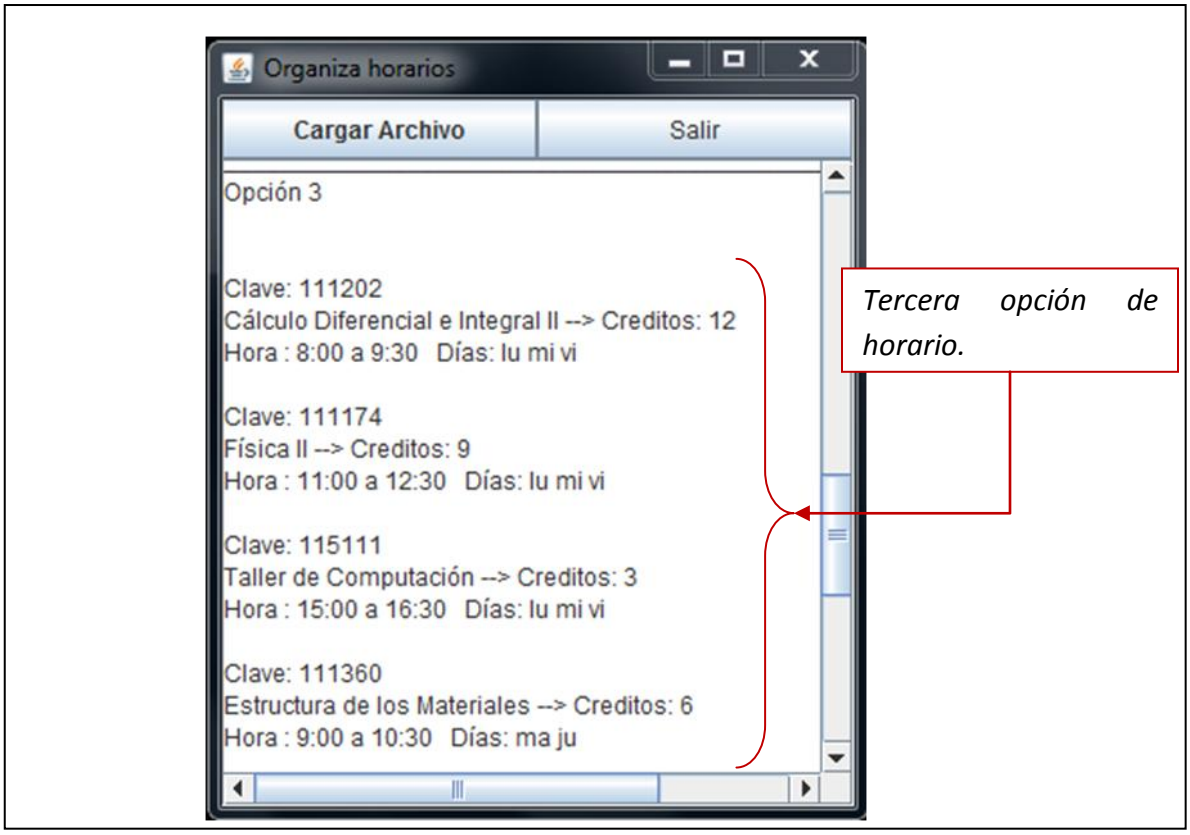

Ilustración 19. Tercera opción de horario

En esta opción (Ilustración 20) se muestran la misma cantidad de materias, solo que existe una variación en cuanto al horario en que se imparte Física II. Queda organizado de la siguiente forma:

Lunes	Martes	Miércoles	Jueves	Viernes
Calculo I 8:00-9:300		Calculo I 8:00-9:300		Calculo I 8:00-9:300
Física II 11:00-12:30		Física II 11:00-12:30		Física II 11:00-12:30
Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30		Taller de Comp. 15:00 – 16:30
	Estruct. Mat. 9:00-10:30		Estruct. Mat. 9:00-10:30	

5) Para terminar la aplicación solo de clic en el botón **Salir** o bien solo cierre dando clic en el botón superior derecho de la ventana.

Ilustración 20. Cerrar la aplicación.

El programa no puede decir cuál es el mejor horario para un usuario. Solo presenta las opciones que puede tener.

Ejecución de organiza.jar desde Linux

En Linux debe abrir la herramienta **Terminal** (consola de comandos). Localice la ubicación del archivo **organiza.jar** y vaya a ese directorio desde la terminal.

1) Teclear el siguiente comando para su ejecución: ***java –jar organiza.jar***.
Esto ejecutara la aplicación. Ilustración 21.

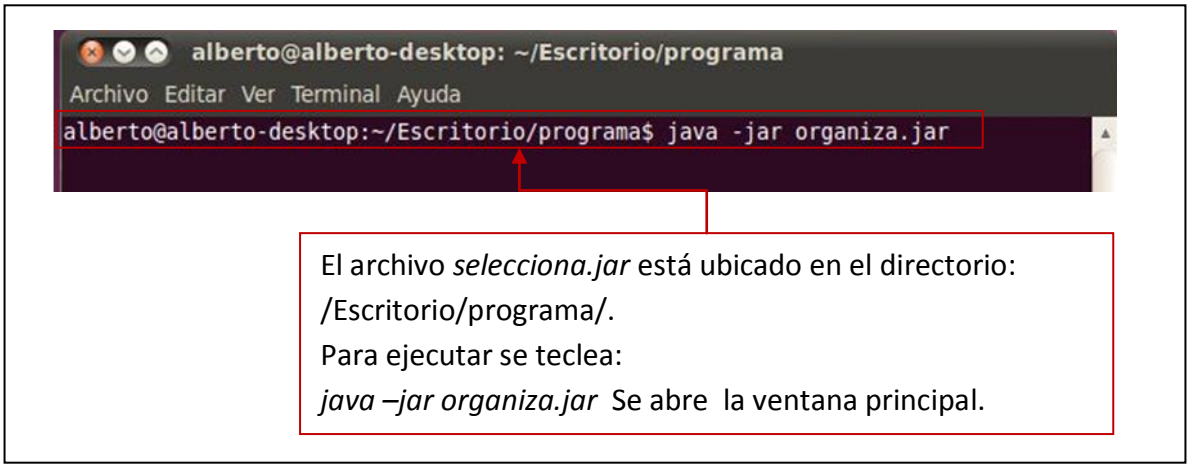

Ilustración 21. Ejecutando *organiza.jar* en Linux

2) Si se ejecuto correctamente el comando del paso anterior entonces debe de aparecer la ventana principal de la aplicación ***organiza.jar***.

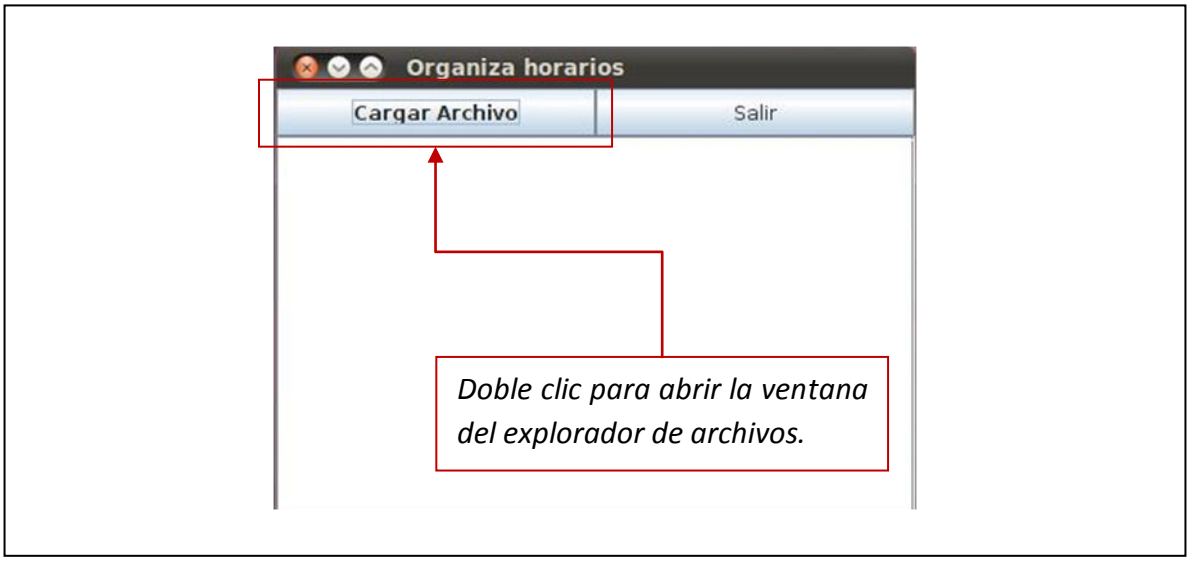

Ilustración 22. Cargar archivo desde Linux.

3) En el explorador de archivos debe navegar hasta la ruta donde se encuentra el archivo con los horarios disponibles. Nota: El explorador solo reconoce archivos con extensión **.uea**, si no creo correctamente su archivo no podrá seleccionarlo.

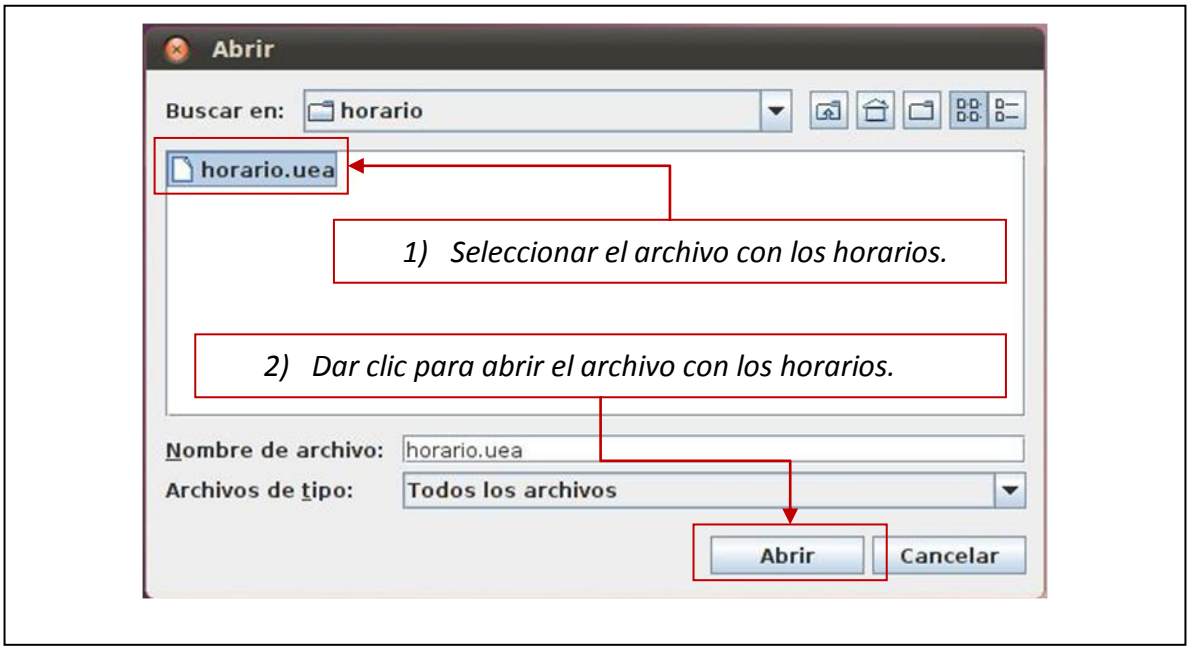

Ilustración 23. Seleccionar archivo en Linux

4) En cuanto se carga un archivo satisfactoriamente. Se muestra la ventana con los resultados obtenidos.

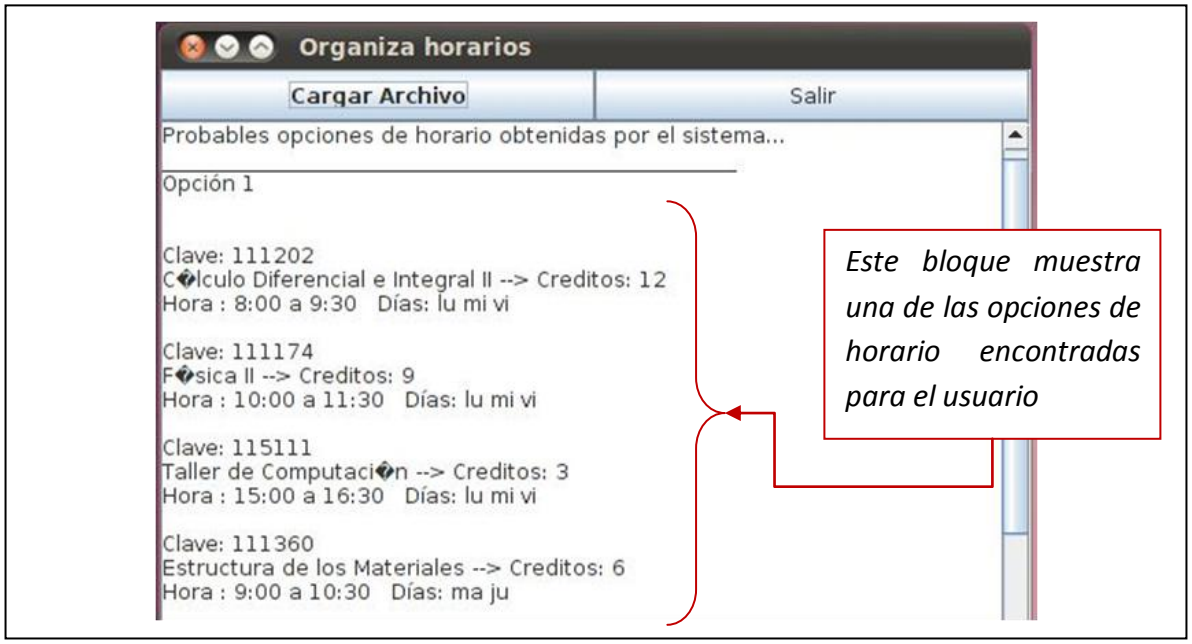

Ilustración 24. Resultados obtenidos en Linux

Los resultados mostrados en estas opciones, son los mismos que se obtienen de la ejecución en Windows. El problema que se presenta en Linux es que los caracteres que están acentuados se muestran como un símbolo no reconocido en Linux.

5) Para terminar la aplicación solo de clic en el botón **Salir** o bien cierre dando clic en el botón superior izquierdo de la ventana.

Ilustración 25. Cerrar la aplicación en Linux

Errores comunes en selecciona.jar y organiza.jar

Algunos de los mensajes de error que puede mostrar la aplicación ***selecciona.jar*** pueden ser los siguientes:

1) **No selecciono claves**: si no selecciona ninguna casilla de verificación el sistema le mostrara el mensaje ******Asegúrese de que selecciono clave(s) de uea***** (Ilustración).
   Causa: Esto se debe a que dio clic en el botón aceptar sin haber seleccionado ninguna casilla.
   Solución: simplemente seleccione las uea's que tiene aprobadas y de clic nuevamente en el botón **Aceptar.**

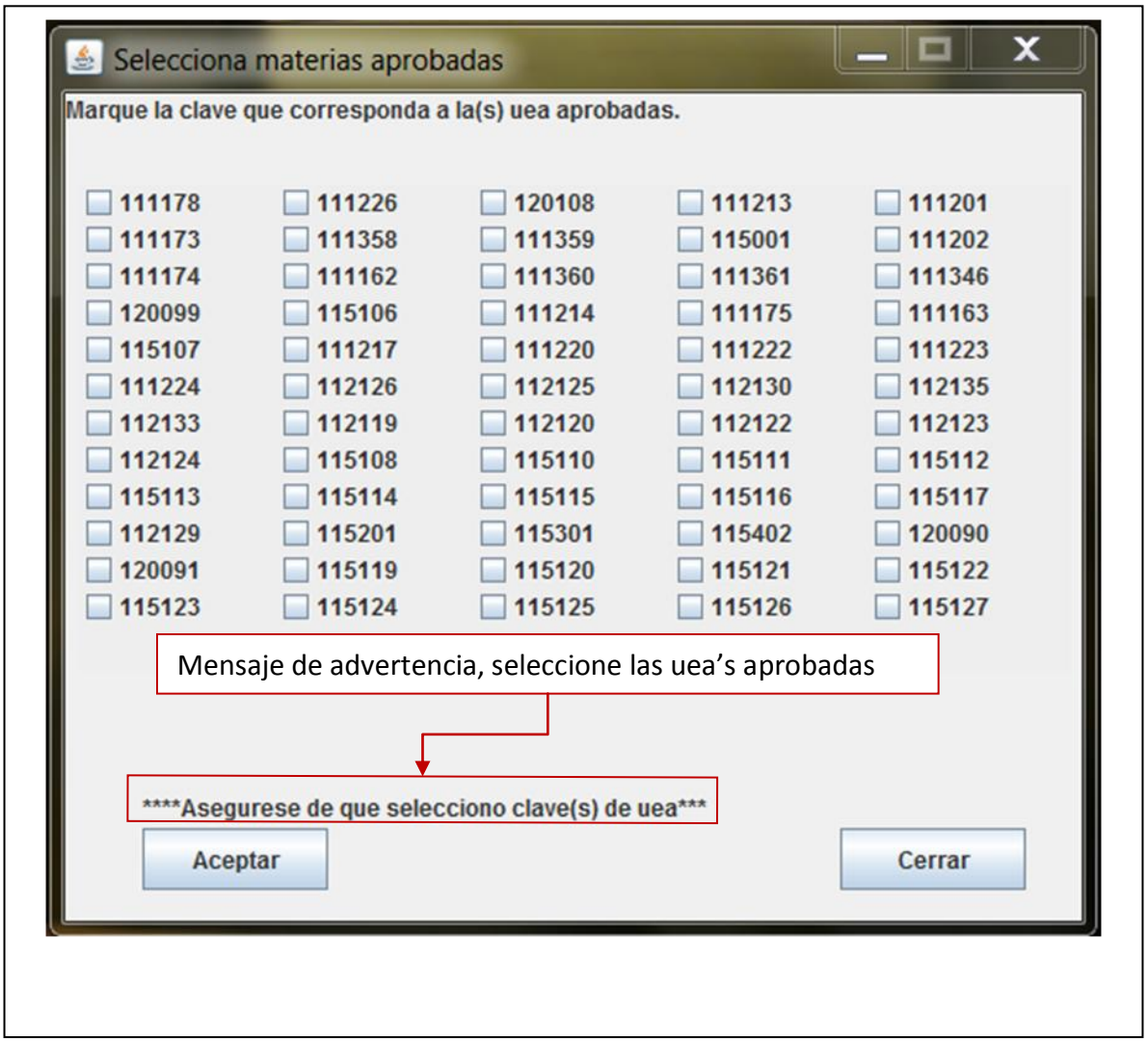

Ilustración 26. Mensaje de advertencia en *selecciona.jar*

2) **No selecciono ningún archivo**: Se muestra la advertencia "**No selecciono ningún archivo".**
Causa: Dio clic en el botón ***Cargar Archivo…*** y no selecciono ningún archivo, al dar clic en el botón **Cancelar** del navegador de archivos. Aparece la advertencia.
Solución: Vuelva a dar clic en **Cargar Archivo…** y elija un archivo.uea.

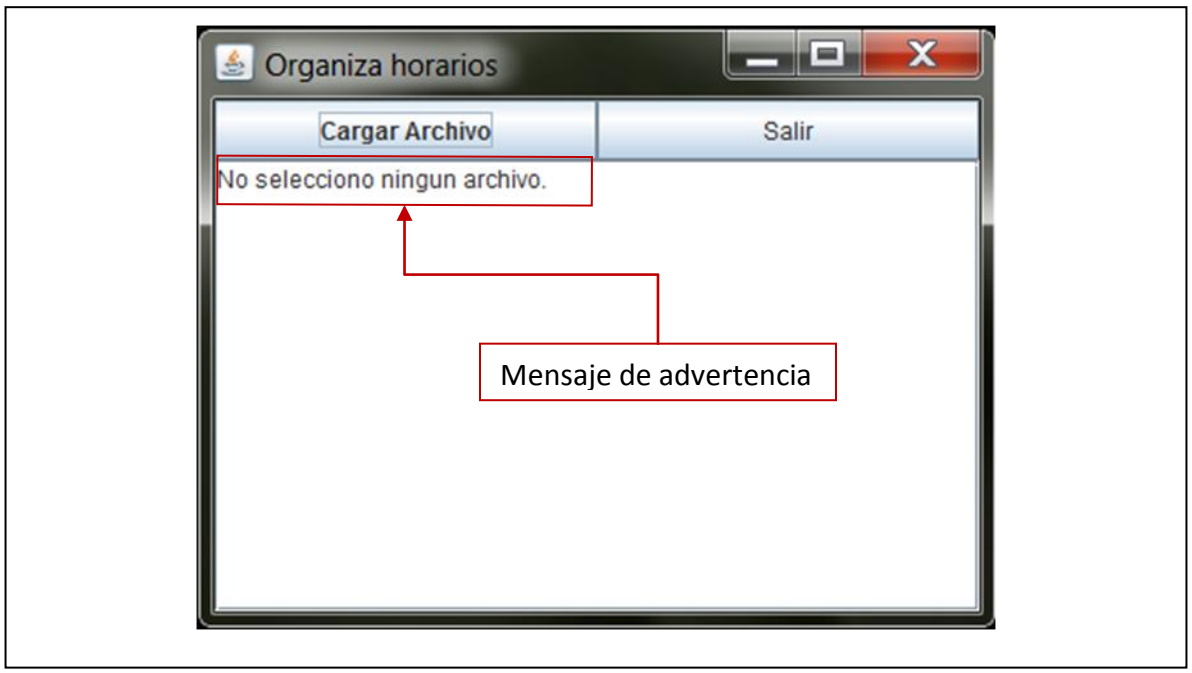

Ilustración 27. Mensaje de advertencia en *organiza.jar*

3) **Uea repetida en el archivo.uea:** En *organiza.jar* se muestra el mensaje "**revise las materias y los horarios…**"

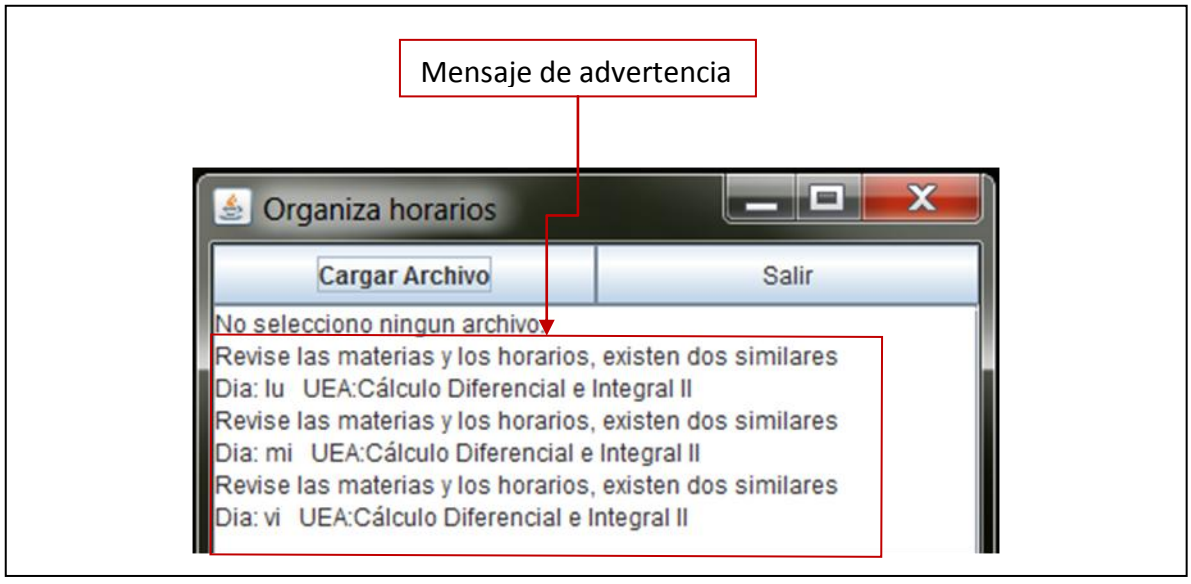

Ilustración 28. Mensaje de uea's repetidas en *organiza.jar*

Causa: El archivo.uea que contiene los horarios tiene una o algunas uea's repetidas. Esto quiere decir que se imparten a la misma hora y el mismo día.
Solución: Debe quitar las uea's que están repetidas, no tiene sentido que coloque dos uea's en el mismo día y hora. Si puede elegir manualmente entre una u otra.

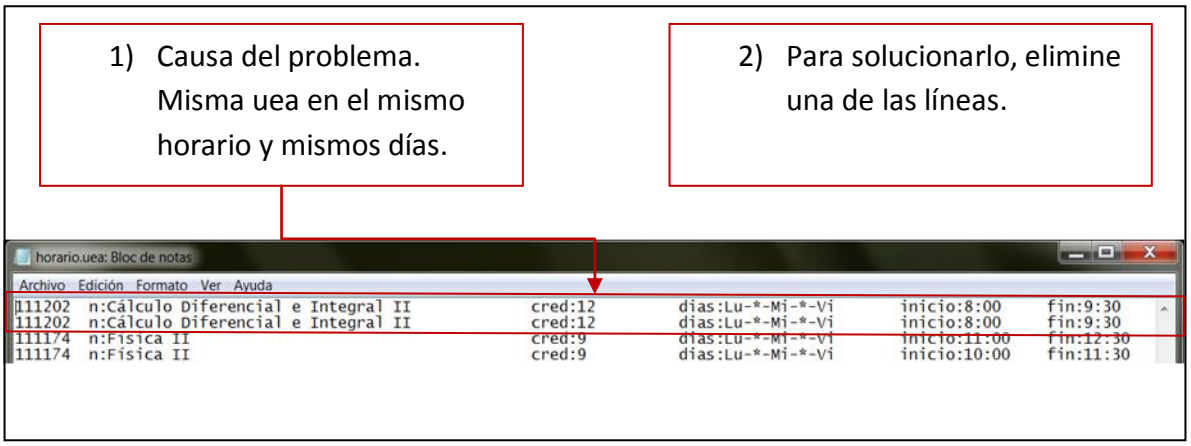

Ilustración 29. Solución de uea repetida

En general, pudieran surgir otros problemas que no estén contemplados en este manual. Las aplicaciones son una versión no concluida en la interfaz. Pueden existir casos especiales de prueba que no hayan sido contemplados al realizar el organizador de horarios.

Requisitos mínimos del sistema

Procesador: 800 MHz

Memoria RAM: 512 MB

Disco Duro: 1 MB*

Software: Maquina virtual de Java

Sistema operativo: Windows o Linux**

* El espacio para almacenar las aplicaciones en el disco duro es mínimo.

**No se ha experimentado en otra distribución Linux distinta a Ubuntu.

Glosario

Clave: Cada asignatura tiene un número compuesto por seis dígitos que la identifica.

Comando: Línea de órdenes que se desea ejecute la computadora.

Consola: Interfaz de línea de comandos, permite al usuario dar instrucciones a la computadora.

Créditos: Es el puntaje que recibe el alumno al aprobar una uea. Estos son acumulables.

Editor texto plano: Es un programa que permite editar texto sin ningún formato.

Jar: Son las iníciales de **J**ava **Ar**chive, permite ejecutar aplicaciones programadas en lenguaje Java.

Java: Lenguaje de programación orientada a objetos.

Linux: Sistema operativo libre.

Maquina virtual: Software que permite ejecutar programas emulando una computadora.

Paquete: Es un conjunto de programas, los programas que se instalan en Linux son llamados así.

Plantilla: Es un formato general que permite llevar duplicados similares.

Terminal: Es una consola, programa instalado en Linux.

UEA: Las asignaturas en la UAM son llamadas por las iníciales de Unidad de Enseñanza Aprendizaje.