Molinaroli Carlos Atilio, Blanco Valentín Nicolas

Entregable Teórico N2

1) Los datos leídos de los archivos movies.csv y rating.csv fueron almacenados en un comienzo en 2 listas, una lista de objetos películas que contiene los atributos de cada película del archivo movies.csv (nombre e id de la película) y otra lista de objetos usuarios que contiene los atributos de cada usuario sobre cada película extraídos del archivo rating.csv (id del usuario, rating e id de la película). Luego de extraer dichos datos esta información fue almacenada en un arreglo ordenado para facilitar la muestra ordenada de datos cuando se selecciona la cantidad del mismo ya que si se hubiese elegido otra estructura ordenada como por ejemplo una lista hubiese sido mucho mas ineficiente ordenar y desplegar dichos datos ya que se accede de forma secuencial por lo que se debería haber recorrido desde el inicio cada vez que se quería agregar una película de forma ordenada y ademas para desplegar los datos se debía acceder al puntero del siguiente elemento (lo que consume mas ram que un acceso de un arreglo donde se le indica el elemento y se accede de forma directa).

2) La lectura de los datos fue resuelta a través de ir leyendo por lineas y dentro de linea cada letra hasta llegar a una coma, en el caso de leer números (como el id de una película) estos eran transformados en integer (restándole 48 al carácter ya que en el código ascii el 0 representa el 48),almacenados en una variable y multiplicados por 10 en caso de no llegar a la coma, en cambio si era un carácter, este era leído y concatenado a un String. Después de esta lectura dichas variables eran almacenadas en las listas mencionadas en 1). Para leer los datos usamos la clase BufferedReader que como mencionamos antes nos facilitaba la lectura de linea por linea. Para el caso que los archivos no se encuentren en el filesystem o de no contar con permisos de lectura esto se resolvía imprimiendo el StackTrace mostrando la secuencia de métodos invocados que nos llevó al punto que disparó la excepción en consola.

```
try {
FileInputStream fstream1 = new
FileInputStream(Paths.get("src","movies.csv").toString());
FileInputStream fstream2 = new
FileInputStream(Paths.get("src","ratings.csv").toString());
BufferedReader reader1 = new BufferedReader(new
InputStreamReader(fstream1));
BufferedReader reader2 = new BufferedReader(new
InputStreamReader(fstream2));
… // la solucion de la lectura linea por linea y carácter por carácter fue omitida en la parte del codigo ya que eran demasiadas lineas por lo que optamos por mostrar la partew de la apertura de datos y solucion del error
}
catch (FileNotFoundException ef) { ef.printStackTrace();} catch
(IOException e) { e.printStackTrace();
```

3) Para que no se “congele” la interfaz de usuario se opto por usar threads (hilos) lo que permite al hilo de lectura de datos ceder cpu a otro hilo (como la interfaz de usuario que era el main thread) en caso de que se solicite luego de que el hilo actual indique que ya realizo suficiente trabajo y puede ceder cpu a otro hilo(a través del método yield). Para la lectura de datos como ya se menciono esta fue tratada como un hilo en donde implementamos la interfaz runnable que nos obligaba a llamar a un método run aunque no poseyera aun características de un hilo, para ello creamos un objeto de la clase Thread de nombre hilo (`private Thread hilo=null;`) que nos permitía invocar el método start y de esta forma tratar a la clase como un hilo (`hilo.start();`). Ademas en nuestro caso optamos por

también añadir una barra de carga y tratarla como un hilo por mas que no fuese necesario.
Código:

```
private Thread hilo=null;
private BarraProgreso ba;
private TablaCant t;
private SelectorCant s;
public ProcesarDatos(TablaCant t, SelectorCant s,BarraProgreso ba) {
            this.ba=ba;
            this.t=t;
            this.s=s;
            start();
      }
      public void start() {
            if (hilo==null) {
                  hilo=new Thread(this);
                  hilo.start();
            }
      }
      public void run() {
      … //dentro del run esta todo el procesamiento de los datos y cada
vez que se leía una linea se llamaba al método yield
      Thread.yield();
      …
      }
```

4) a) Las componentes de GUI utilizadas para la visualización de la tabla de películas fueron la clase JTable para la tabla en si, JComboBox para el selector de cantidades mostradas por la tabla y JScrollPane que nos permite desplazarse en la tabla.
Una vez procesados los datos los eventos de interfaz de usuario que se pueden observar son: la cantidad de películas que quiero visualizar en la tabla, como se explico antes en caso de haber mas películas de las que se pueden visualizar es posible desplazarse y en caso de seleccionar una película mostrar un rating en el histograma.
Para el uso de la tabla los campos llenados en primer lugar fueron solo los títulos (Nombre de película, Usuarios y Votos) y después (una vez cargado los datos en la matriz) se carga linea a linea en la tabla la cantidad de películas seleccionadas.
b) Para manejar el evento del selector agregamos una “ActionListener” y utilizamos el método actionPerformed en donde en caso de seleccionar una de las cantidades se asignaba dicha cantidad en una variable (con el comando `box.getSelectedItem()`), se borraba la tabla (a través de un for en donde utilizamos el metodo `mt.removeRow(tabla.getRowCount()-1)` borrando cada fila) y se hace un switch de la variable con la cantidad en donde a través de un for se asignan las películas en un arreglo de String (película, votos, rating) y con el comando `mt.addRow(s);` (s siendo el String) se agrega la fila a la tabla. Para manejar el evento de la tabla con el histograma se agrego un “ListSelectionListener” y se utilizo el método valueChanged en donde en caso de clickear un elemento de la tabla se pintaba el histograma.
Otra solución posible al manejo del primer evento podría ser a través del uso de jTextField en donde se rellenaría el campo con un numero o la palabra “TODOS” y clickeando un botón para confirmar dicha selección se desplegarían en la tabla.
c) Si es posible en nuestro código esto fue implementado como ya se explico antes a traves del metodo valueChanged. Código de esta solución:

```
public void histoTabla(PelisUsuariosVotos[] arreglo) {
    tabla.getSelectionModel().addListSelectionListener(new ListSelectionListener(){
        public void valueChanged(ListSelectionEvent event) {
            if (tabla.getSelectedRow()!=-1) {
                h.setAlturas(arreglo[tabla.getSelectedRow()].getArregloRating());
            }
        }
    });
}
```

d) En nuestro código la posibilidad de filtrar por nombre usuarios y votos no se realizo aunque es posible. Esta se podría realizar instanciando un objeto de la clase TableRowSorter y asignándole a la tabla ese objeto(con el comando setRowSorter).

5) El histograma fue implementado a través del uso de un objeto de la clase Graphics2d (subclase de Graphics, llamado a este objeto "grafico") al que le asignamos color negro en un primer momento para realizar un plano cartesiano (se utilizo `grafico.drawLine` para realizar las lineas horizontales y verticales del plano) donde en x ubicamos las puntuaciones de la película (yendo este de 0 a 5 y utilizando `grafico.drawString` para graficarlo) y en y, el numero de votos. Después de realizar el plano cartesiano se cambio el color del grafico a azul para graficar los rectángulos dentro de dicho plano (para realizar estos rectángulos se uso el comando `grafico.fillRect`). Como ya se menciono una de las componentes de GUI utilizadas fue la clase Graphics2d aunque ademas de esta también se utilizo un objeto de la clase Jlabel con la etiqueta de histograma para aclarar que ese grafico hacia referencia al histograma. Ambas componentes fueron contenidas sobre un Layout Manager de tipo FlowLayout ya que hicimos que la clase histograma extendiera de Jpanel y este layout es el predeterminado para dicha clase.

Cada vez que se selecciona una película el histograma debía limpiarse y esto fue hecho a través del comando repaint (ubicado dentro del método setAlturas que recibe los nuevos datos de la película seleccionada) que vuelve a llamar al método paintComponent ejecutando lo antes mencionado.

6)Aspectos que se pueden distinguir de nuestro programa son por ejemplo el agregado de un evento en caso de querer cerrar la ventana ya que sino se debía recurrir al administrador de tareas, otro caso que se puede observar son la barra tratada como un hilo a parte en el procesamiento de datos, la posibilidad de dejar la tabla vaciá y la lectura de carácter por carácter en el procesamiento de datos en lugar de usar el método split.