1. Los datos los guardamos en un HashMap donde su key seria el ID de película y sus values (o valores del hash) serian un objeto película con los siguientes atributos:

```
public class Pelicula implements
Comparable<Pelicula> {
	private double cantVotos = 0;
	private int cantUsuarios = 0;
	private String Nombre;
	private int[] votos = {0,0,0,0,0,0};

...}
```

La razón para utilizar un HashMap es para que al momento de Leer el segundo Archivo(ratings.csv), sea más fácil encontrar a la película que esta referida el voto, ya que la Key del Hash es la ID de la película.

```
public class LeerArchivo extends Thread {
	...
	private HashMap<Integer, Pelicula> hashDePelis;

	private static List<Pelicula> list;

...}
```

2. En nuestro caso incluimos los archivos (movies.csv y ratings.csv) en el jar ejecutable para evitar problemas de con la dirección de la ubicación de los archivos, aun así, si llegara a haber algún problema con la lectura de los mismos el error se informaría por la consola, debido a que la lectura de los mismos esta encerrada en un TryCatch. , las clases utilizadas para la lectura de los mismos es el BufferedReader y el InputStreamReader

```
public class LeerArchivo extends Thread {
	...
	public void run() {
		try {
			BufferedReader buffer = new BufferedReader(new
InputStreamReader(getClass().getResourceAsStream("movies.csv")));

			...

			buffer = new BufferedReader(new
		InputStreamReader(getClass().getResourceAsStream("ratings.csv
		")));

			...

			catch (Exception e) {

			...

			}

			...

			list = new ArrayList<Pelicula>(hashDePelis.values());
			Collections.sort(list);

}
```

3. En nuestro caso hicimos Hilo(o thread) que se ejecuta en paralelo al programa al momento de leer los archivos, de esa forma, la GUI sigue activa mientras que en un Hilo secundario se leen los archivos.

```
public class PanelSuperior extends JPanel {
private JButton boton = new JButton("Procesar datos");
private LeerArchivo arch = new LeerArchivo(); //esta es la clase lectora
 ...
            boton.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        try {
                              Thread hilo= new Thread(arch);
                              hilo.start();
                              boton.setEnabled(false);
                        } catch (Exception e1) {
                              e1.printStackTrace();
                        }
                  }
            });
```

La razón de no atrapar la excepción de el error de la lectura de archivo (IOExeption) y atrapar una excepción mas general en el try/catch de la clase leer archivo, es debido a que en una parte de el código utilizamos un método de la clase Thread que “duerme” el proceso y la misma genera una InterruptedExeption.

4.

a. Utilizamos un boton, labels, una barra de progreso, un combo box, JTable y ScrollPane, y fuimos encapsulando dichos elementos en paneles para lograr un orden y el funcionamiento adecuado de la GUI.

Para el llenado de la tabla utilizamos un listener en el ComboBox mencionado anteriormente, el cual actúa cuando se selecciona algún valor de la lista que contiene, una vez seleccionado la cantidad de datos a visualizar en la tabla, se cargan los datos a la misma. Si aun no se cargaron los datos se muestra un dialogo de alerta indicando lo anteriormente mencionado y devuelve el ComboBox a su valor inicial.

Para poder tener la tabla ordenada por promedio de votos, creamos una lista de tipo película ordenada de esa forma utilizando el método sort que utiliza una clase con interfaz comparable.

```
public class CantidadDatosAMostrar extends JPanel{
private JComboBox<String> cant;
 ...
cant.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                           try {
                                  int cantPelis = LeerArchivo.getListaPelis().size();
                                  DefaultTableModel modelo = new DefaultTableModel(titulo, 0);
                                  int a, contador = 0;
                                  if (cant.getSelectedItem().equals("TODOS")) {
                                         a = cantPelis;
                                  } else {
                                         a = Integer.parseInt((String) cant.getSelectedItem());
                                  }
                                  for (Pelicula p : LeerArchivo.getListaPelis()) {
                                         if (contador == a)
                                                break;
                                         modelo.addRow(new Object[] { p.getNombre(), p.getCantUsuarios(),
p.getPromedio() });
                                         contador++;

                                  }
                                  tabla.getTabla().setModel(modelo);
                           } catch (Exception ex) {
                                  JOptionPane.showConfirmDialog(null, "No se cargaron los archivos",
"ERROR",
                                                JOptionPane.PLAIN_MESSAGE, JOptionPane.ERROR_MESSAGE);
                                  cant.setSelectedIndex(0);
                           }

                    }

...
```

b. En nuestro caso utilizamos distintas formas, En el ComboBox visto anteriormente, en el caso del botón creamos un Listener que espere el momento que el usuario clickee el boton, y en el caso de la tabla creamos un listener de mouse para cuando se clickee alguna celda de la tabla.

Otra forma que podríamos haber utilizado para manejar alguno de los eventos hubiera sido con un Listener de teclado esperando que toque alguna tecla en especial para poder ejecutarse.

```
public class Tabla extends JPanel {
private JTable tabla = new JTable() {
public boolean isCellEditable(int row, int column) {
                    return false;
            }
      };...
public Tabla(Histograma p) {
      ...
      tabla.addMouseListener(new MouseAdapter() {
                              public void mouseClicked(MouseEvent e)
{...}
...
```

c. Como mostre en la imagen anterior al clickear alguna celda de la tabla el listener actualiza los valores del histograma con un método creado en la clase del mismo.
d. En nuestro caso no implementamos esa característica, pero una forma de hacer esa solución seria que una vez que se clickea sobre el titulo de alguna de las columnas se cambie el orden de la lista de películas en la clase Lectora y luego volver a cargarla en la tabla.

5. En el caso del Histograma utilizamos 2 Paneles (Uno como papel y otro como encapsulador de tamaño mas grande que el papel.
   En el papel utilizamos del paquete awt la clase graphics2d para hacer el grafico en dos dimensiones, con la cual dibujamos todas las partes del Histograma

```
class Papel extends JPanel {

public Papel() {
                    setLayout(new BorderLayout());
                    setPreferredSize(new Dimension(690, 320));
                    setBackground(Color.WHITE);

...}

public void paintComponent(Graphics g) {
                    super.paintComponent(g);
                    Graphics2D graf = (Graphics2D) g;

...

            graf.setColor(Color.CYAN);
            graf.fillRect(253, 279-alto0, 30, alto0);//Dejamos una altura base
            graf.fillRect(303, 279-alto1, 30, alto1);// la cual se va actualizando
            graf.fillRect(353, 279-alto2, 30, alto2);//con el cambio de los valores en
            graf.fillRect(403, 279-alto3, 30, alto3);//celeste
            graf.fillRect(453, 279-alto4, 30, alto4);
            graf.fillRect(503, 279-alto5, 30, alto5);
          //Se le resta la altura porque se grafica de arriba para abajo.

...}

public void setAlturas(int[] ranks) {

      Metodo que cambia los valores alto0,alto1,etc.

...}
```

6. Agregamos el icono de la compañía que pedia el trabajo, los cuadros de dialogo en caso de error o al finalizar la carga de archivos, el manejo del Hilo en la carga de archivos, el encapsulado de los componentes en disitintos Paneles , el centrado de la aplicación en el centro de la pantalla, que el Frame sea una clase singleton

```
public class Frame extends JFrame {
	private final URL ubiImagen = getClass().getResource("icono.jpg");
	private PanelSuperior pSup = new PanelSuperior();
	private JPanel panelCentro = new JPanel(new BorderLayout());
	private CantidadDatosAMostrar cant =new CantidadDatosAMostrar();
	private PanelDatosGenerales datosGen = new PanelDatosGenerales();
	private static Frame instance = new Frame();
	private Frame() {
		super("Ranking de peliculas");
		ImageIcon img = new ImageIcon(ubiImagen);
		panelCentro.add(datosGen, BorderLayout.NORTH);
		panelCentro.add(cant, BorderLayout.CENTER);
		panelCentro.add(cant.getPintura(), BorderLayout.SOUTH);
		setIconImage(img.getImage());
		setLayout(new BorderLayout());
		add(pSup, BorderLayout.NORTH);
		add(panelCentro, BorderLayout.CENTER);
		Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
		setLocation((int) pantalla.getWidth() / 4, (int) pantalla.getHeight() / 15);
		setResizable(false);
		...
	}
```