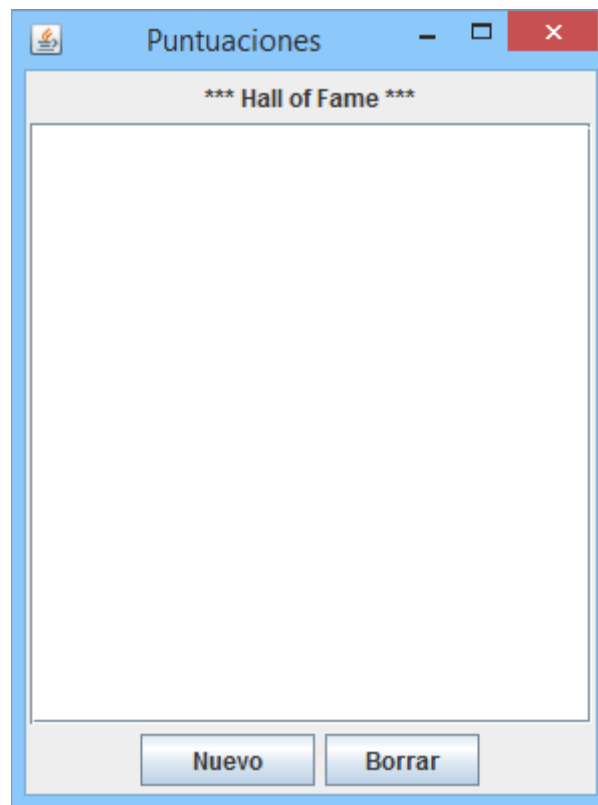


Proyecto Hall of Fame

Crea una aplicación que nos permita almacenar nombres y puntuaciones de jugadores en un fichero de disco. La interfaz de la ventana sería:

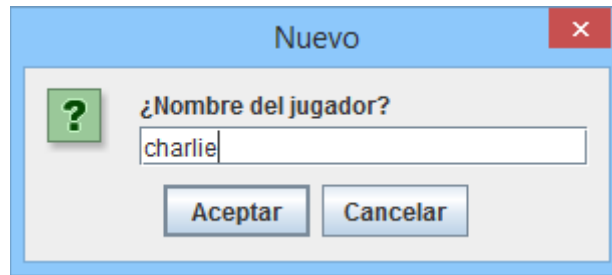


El área en blanco que ocupa casi toda la ventana es un JList.

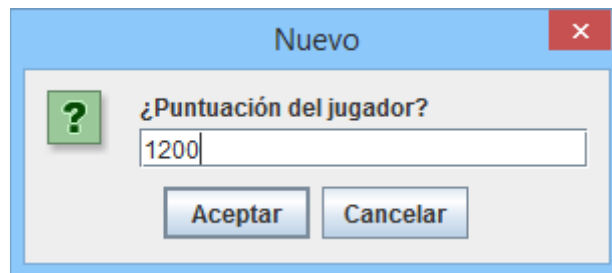
El nombre del archivo de disco sería "ficheros/jugadores.txt". Los datos se guardarían en formato [CSV](#) (separados por comas, sin espacio entre datos y comas). Por ejemplo:

```
charlie,1200  
nemesis,3025
```

Al pulsar el botón "Nuevo" se mostrarán dos paneles de opciones (JOptionPane) para introducir el nombre y la puntuación del jugador.

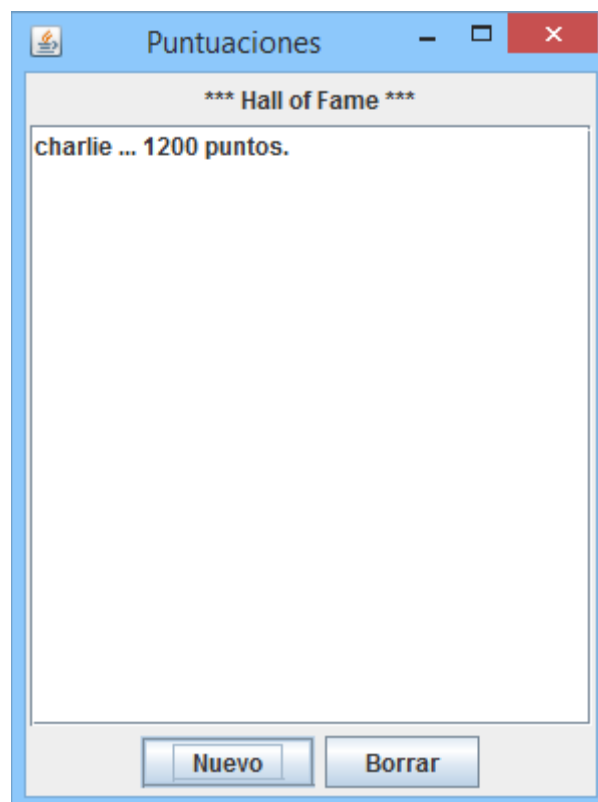


A dialog box titled "Nuevo" with a red close button. It contains a green question mark icon, the text "¿Nombre del jugador?", a text input field containing "charlie", and two buttons: "Aceptar" and "Cancelar".



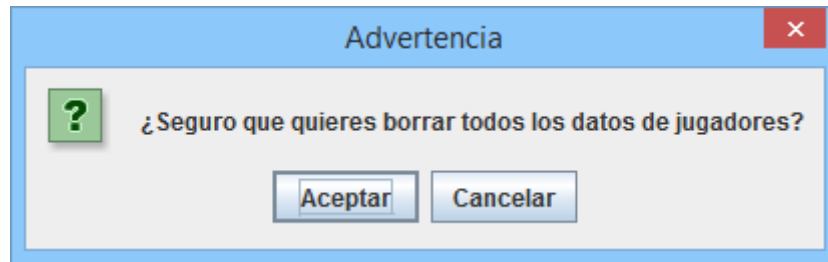
A dialog box titled "Nuevo" with a red close button. It contains a green question mark icon, the text "¿Puntuación del jugador?", a text input field containing "1200", and two buttons: "Aceptar" and "Cancelar".

A continuación se creará un nuevo registro en el archivo de puntuaciones y se actualizará la lista en la ventana principal (el nombre y la puntuación se muestran separados por puntos suspensivos):

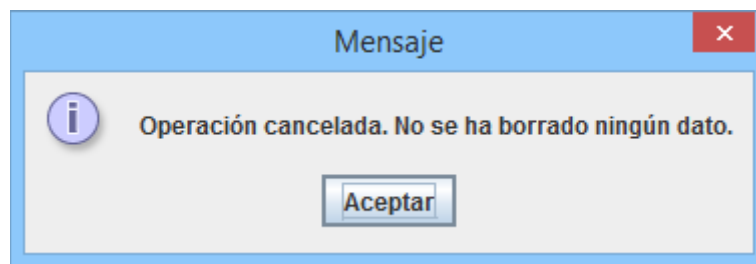


A main window titled "Puntuaciones" with standard window controls. It features a header "*** Hall of Fame ***" and a list box containing the text "charlie ... 1200 puntos.". At the bottom, there are two buttons: "Nuevo" and "Borrar".

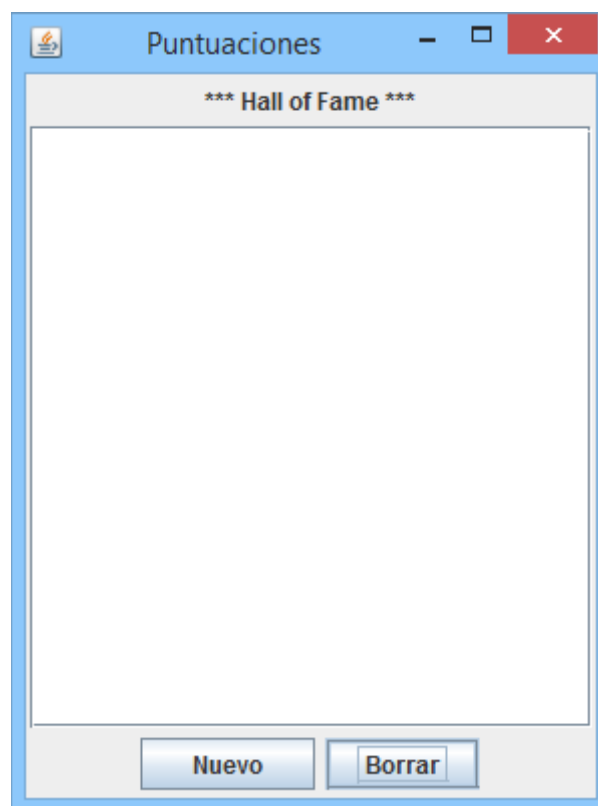
Si se pulsa el botón "Borrar" se pedirá confirmación al usuario:



En caso de que se pulse "Cancelar", se mostrará un mensaje tranquilizador y no se cambiará nada:

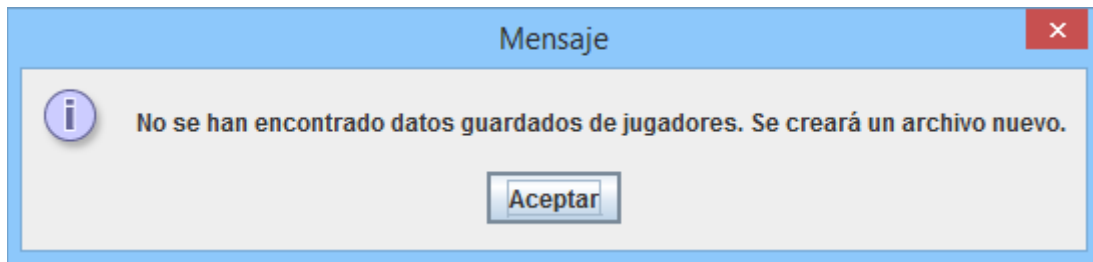


En cambio, si se ha pulsado "Aceptar", se borrará el contenido del archivo de puntuaciones y se actualizará la lista de la ventana principal, dejándola vacía:

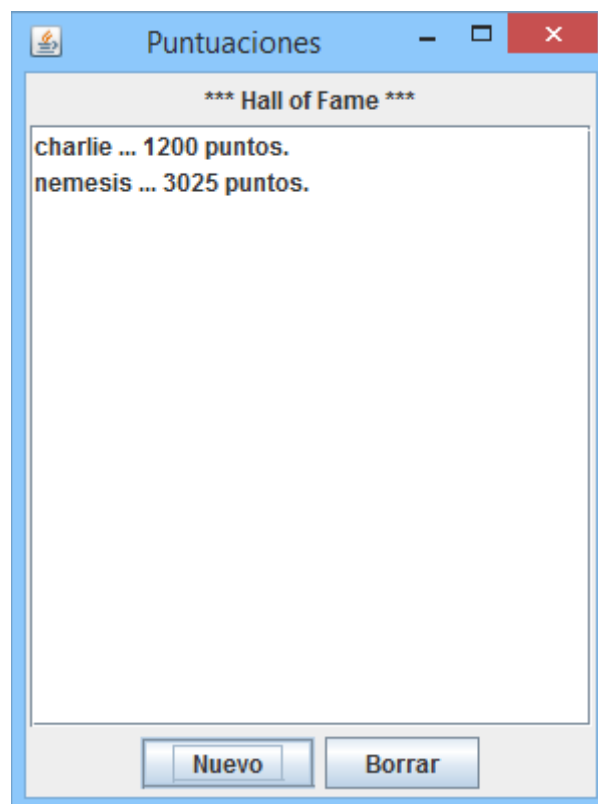


Además:

- El JList debe mostrar barras de desplazamiento si el contenido no cabe.
- Cuando se abre el programa por primera vez se informa de que no existe el archivo de puntuaciones y se creará uno nuevo:



- Al cerrar el programa todos los datos deberán estar guardados en el disco.
- Al abrir de nuevo el programa, los datos se leerán y se mostrarán en la ventana principal:



- Si se produce un error en alguno de los try-catch se mostrará el texto "Error de E/S: " y la descripción del error en un JOptionPane. De todas formas, en principio no debería haber errores.

- Para manejar los eventos, utiliza clases anónimas. Por ejemplo:
`btnNuevo.addActionListener(new ActionListener() {...}).`

Orientaciones

- Para añadir elementos o borrarlos de un JList deberás usar un objeto *DefaultListModel*. Tienes un ejemplo [aquí](#).
- Puedes usar un FlowLayout, y ajustar el tamaño del JList con el método:

```
panelJugadores.setPreferredSize(new Dimension(280, 300));
```

- Una forma sencilla de leer datos de un archivo CSV es leer línea a línea y utilizar el método *split()* de la clase String.
- Puedes borrar el contenido de un archivo de texto simplemente abriéndolo y cerrándolo.
- Crea una clase Jugador con los atributos nombre y puntuacion y los constructores y métodos de acceso que necesites.
- Divide el trabajo a realizar creando los siguientes métodos auxiliares:

```
/**
 * Añade un jugador nuevo y su puntuación. Este método es llamado desde el
 * listener del botón Nuevo.
 */
private void nuevoJugador() {}

/**
 * Pide al usuario los datos de un nuevo jugador y devuelve un objeto Jugador.
 *
 * @return Un objeto Jugador con los datos leídos, o null si los datos no son
 *         válidos.
 */
private Jugador leerDatosJugador() {}

/**
 * Guarda la información de un jugador en el archivo de disco, en formato csv.
 *
 * @param jugador Objeto Jugador con el nombre y la puntuación a añadir.
 */
public void guardarJugador(Jugador jugador) {}

/**
 * Lee los datos sobre nombres y puntuaciones de jugadores. Los añade a la lista
 * que se muestra en la ventana.
```

```
*/  
public void cargarJugadores() {}  
  
/**  
 * Borra el contenido del archivo de jugadores, dejándolo en blanco. Y la lista  
 * de jugadores en la ventana. Este método es llamado desde el listener del botón  
 * Borrar.  
 */  
public void borrarArchivo() {}
```