



Universidad Nacional Autónoma de México



Facultad de Estudios Superiores Aragón

Carrera: ICO

Nombre: Luis Alejandro Romero Martínez

Grupo: 1159

Fecha: 5/06/23

Materia: programación Orientada a Objetos

Semestre: II

Profesor: Jesús Hernández Cabrera.

Documentación del proyecto final

Ventana Princesas

Como primer punto empezaremos revisando la vista o la parte de la interfaz del programa el cual hereda de JFrame

```
public class VentanaPrincesas extends JFrame {
    4 usages
    private GridLayout layoutGeneral;

    //Objetos de panel 1
    4 usages
    private JLabel lblId;
    4 usages
    private JLabel lblNombre;
    4 usages
    private JLabel lblCuento;
    4 usages
    private JLabel lblColorVestido;
    4 usages
    private JLabel lblImagen;
    4 usages
    private JLabel lblAntagonista;
    6 usages
    private JTextField txtId;
    5 usages
    private JTextField txtNombre;
    5 usages
    private JTextField txtCuento;
    5 usages
    private JTextField txtColorVestido;
    5 usages
    private JTextField txtImagenUrl;
    5 usages
    private JTextField txtAntagonista;
    4 usages
    private JButton btnAgregar;
    18 usages
    private JPanel panel1;
```

En la primera imagen se aprecia las variables empleadas en como lo dice la primera línea comentada dentro del panel 1. El cual es el que requiere de mayor cantidad de objetos pues como se observa en la siguiente imagen los siguientes paneles van con muchísimos menos objetos

```
//Objetos Panel 2

4 usages
private JTable tblPrincesas;

8 usages
private JPanel panel2;
4 usages
private JButton btnCargar;
4 usages
private JScrollPane scroll;
//Objetos Panel 3

8 usages
private JPanel panel3;
4 usages
private JLabel lblfila;
4 usages
private JLabel lblImg;
//Objetos Panel 4
8 usages
private JPanel panel4;
4 usages
private JButton btnBorrar;
4 usages
private JButton btnActualizar;
```

Después de la declaración de nuestros objetos que usaremos les declaramos los getters y setters desde la línea 44 a la 247 los cuales estas marcados por comentarios

Y para comenzar a acomodar todo creamos el constructor el cual nos pedirá el titulo de la ventana y al igual que en la declaración de los objetos dividimos por partes el programa para tener un mayor orden.

```
//General
layoutGeneral = new GridLayout( rows: 2, cols: 2);
this.setLayout(layoutGeneral);
```

En la anterior imagen se puede ver como primero se declara el Grid layout donde crearemos la cuadrícula para poner cada panel y se la agregamos a la ventana

Como siguiente paso procedemos a crear el panel 1 darle formato y crear todos los objetos que serán usados dentro de este para al final ser agregados al panel.

```
256     this.setLayout(layoutGeneral);
257
258     //panel 1
259     panel1 = new JPanel();
260     panel1.setBackground(new Color( r: 118, g: 178, b: 204, a: 255));
261
262     lblId = new JLabel( text: "Id");
263     txtId = new JTextField( columns: 4);
264     txtId.setText("0");
265     txtId.setEnabled(false);
266     panel1.add(lblId);
267     panel1.add(txtId);
268
269     lblNombre = new JLabel( text: "Nombre");
270     txtNombre = new JTextField( columns: 29);
271     panel1.add(lblNombre);
272     panel1.add(txtNombre);
273
274     lblCuento = new JLabel( text: "Cuento: ");
275     txtCuento = new JTextField( columns: 35);
276     panel1.add(lblCuento);
277     panel1.add(txtCuento);
278
279     lblColorVestido = new JLabel( text: "Color del vestido: ");
280     txtColorVestido = new JTextField( columns: 30);
281     panel1.add(lblColorVestido);
282     panel1.add(txtColorVestido);
283
284     lblImagen = new JLabel( text: "Imagen: ");
285     txtImagenUrl = new JTextField( columns: 35);
286     panel1.add(lblImagen);
287     panel1.add(txtImagenUrl);
288
289     lblAntagonista = new JLabel( text: "Antagonista: ");
290     txtAntagonista = new JTextField( columns: 24);
291     panel1.add(lblAntagonista);
292     panel1.add(txtAntagonista);
293
294     btnAgregar = new JButton( text: "Agregar");
295     panel1.add(btnAgregar);
296
```

Se repite el paso para los paneles 2,3,4, se crean los paneles se les da formato, se crean los objetos y son agregados a su panel correspondiente los cuales también se encuentran separados por comentarios

```
297
298 //panel 2
299 panel2 = new JPanel();
300 panel2.setBackground(new Color( r: 234, g: 108, b: 108));
301 panel2.setLayout(new FlowLayout());
302 btnCargar = new JButton( text: "Cargar");
303 panel2.add(btnCargar);
304 tblPrincesas = new JTable();
305 scroll = new JScrollPane(tblPrincesas);
306 panel2.add(scroll);
307
308
309 //panel 3
310 panel3 = new JPanel();
311 panel3.setBackground(new Color( r: 142, g: 204, b: 105, a: 255));
312 panel3.setLayout(new FlowLayout());
313 lblfila = new JLabel( text: "Foto:");
314 panel3.add(lblfila);
315 lblImg = new JLabel( text: "...");
316 panel3.add(lblImg);
317
318 //panel 4
319 panel4 = new JPanel();
320 panel4.setBackground(new Color( r: 137, g: 211, b: 168, a: 255));
321 panel4.setLayout(new FlowLayout());
322 btnActualizar = new JButton( text: "Actualizar");
323 btnBorrar = new JButton( text: "Borrar");
324 panel4.add(btnActualizar);
325 panel4.add(btnBorrar);
326
327
```

Para finalizar la parte de la vista solo se le da formato a la ventana y se agregan todos los paneles a la ventana.

```
//Parte final

this.getContentPane().add(panel1, index: 0);
this.getContentPane().add(panel2, index: 1);
this.getContentPane().add(panel3, index: 2);
this.getContentPane().add(panel4, index: 3);
this.setDefaultCloseOperation(EXIT_ON_CLOSE);
this.setSize(width: 1000, height: 900);
this.setVisible(true);
}
```

A lo largo del programa se requirio la creación de un metodo que vaciara los textfields el cual se declara sobre esta clase para ser más practico y simple de realizar nombrado **limpiar**.

```
339     public void limpiar(){
340         txtNombre.setText("");
341         txtCuento.setText("");
342         txtColorVestido.setText("");
343         txtImagenUrl.setText("");
344         txtAntagonista.setText("");
345     }
346
347 }
```

Princesas Disney

En esta clase lo único que tenemos es lo que será la base de nuestra tabla por así decirlo pues son los atributos de las princesas de Disney por así decirlo generales

```

7      public class PrincesasDisney {
          4 usages
8          private int id;
          5 usages
9          private String nombre;
          5 usages
10         private String cuento;
          5 usages
11         private String colorVestido;
          6 usages
12         private String imageUrl;
          5 usages
13         private String antagonista;
14
          2 usages
15         public PrincesasDisney() {
16     }

```

Lo único que se realizó en esta clase son sus getters y setters así como su constructor por defecto y el sobre cargado, además de usar uno donde no se requiera el id para que este no se vea afectado en ciertos procesos.

```

15     public PrincesasDisney() {
16     }
17
18     no usages
19     public PrincesasDisney(String nombre, String cuento, String colorVestido, String imageUrl, String antagonista) {
20         this.nombre = nombre;
21         this.cuento = cuento;
22         this.colorVestido = colorVestido;
23         this.imageUrl = imageUrl;
24         this.antagonista = antagonista;
25     }
26
27     2 usages
28     public PrincesasDisney(int id, String nombre, String cuento, String colorVestido, String imageUrl, String antagonista) {
29         this.id = id;
30         this.nombre = nombre;
31         this.cuento = cuento;
32         this.colorVestido = colorVestido;
33         this.imageUrl = imageUrl;
34         this.antagonista = antagonista;
35     }

```

Por ultimo dentro de esta clase tenemos un método el cual nos ayuda en el proceso del cual a través de la url de la tabla se regresara una imagen. Este se llamo **getImagen**.

```
1 usage
94 public ImageIcon getImagen() throws MalformedURLException {
95
96     URL urlImage = new URL(this.imageUrl);
97     return new ImageIcon(urlImage);
98
99 }
```

Conexión Singleton

Para esta clase lo único que hice fue copiar el código para poder establecer la conexión con nuestra bases de datos además de agregar la implementación para ayudar a la conexión a ser realizada puesto que sin esta no corre. Aquí no se realizó ningún cambio.

Interfaz DAO

En el interfaz DAO no ningún cambio al ser implementado en la siguiente clase ese se quedo tal cual con los métodos abstractos.

```
1 usage 1 implementation
6 public interface InterfazDAO {
7     1 usage 1 implementation
    public abstract boolean insertar(Object obj) throws SQLException;
8
9
10    1 usage 1 implementation
    boolean update(Object obj, int Id) throws SQLException;
11
12    no usages 1 implementation
    public abstract boolean delete(int id) throws SQLException;
13    2 usages 1 implementation
    public abstract ArrayList obtenerTodo() throws SQLException;
14    no usages 1 implementation
    public abstract Object buscarPorId(String id) throws SQLException;
15 }
16
```

Princesas Disney DAO

Para esta clase implemente todos los métodos visto pero los principalmente usados fueron el de **Insertar**: que lo que pide es que es un objeto el cual se insertara como un nuevo registro a la base de datos.

```

15      @Override
16      public boolean insertar(Object obj) throws SQLException {
17          String sqlInsert = "INSERT INTO PrincesasDisney(Nombre,Cuento,ColorVestido,Imagen,Antagonista) VALUES(?,?,?,?,?)";
18          int rowCount = 0;
19
20
21          PreparedStatement pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlInsert);
22          pstmt.setString( parameterIndex: 1, ((PrincesasDisney) obj).getNombre());
23          pstmt.setString( parameterIndex: 2, ((PrincesasDisney) obj).getCuento());
24          pstmt.setString( parameterIndex: 3, ((PrincesasDisney) obj).getColorVestido());
25          pstmt.setString( parameterIndex: 4, ((PrincesasDisney) obj).getImagenUrl());
26          pstmt.setString( parameterIndex: 5, ((PrincesasDisney) obj).getAntagonista());
27          rowCount = pstmt.executeUpdate();
28          return rowCount > 0;
29      }
30

```

El actualizar: El cual lo que hace es sobre escribir la información dentro de un registro solo requiere de un objeto y del id del registro que se remplazara.

```

31      @Override
32      public boolean updtte(Object obj,int Id) throws SQLException {
33          String sqlUpdate = "UPDATE PrincesasDisney SET Nombre = ?, Cuento = ?, ColorVestido = ?, Imagen = ?, Antagonista = ? WHERE Id = "+Id+"";
34          int rowCount = 0;
35
36
37          PreparedStatement pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
38
39
40          pstmt.setString( parameterIndex: 1, ((PrincesasDisney) obj).getNombre());
41          pstmt.setString( parameterIndex: 2, ((PrincesasDisney) obj).getCuento());
42          pstmt.setString( parameterIndex: 3, ((PrincesasDisney) obj).getColorVestido());
43          pstmt.setString( parameterIndex: 4, ((PrincesasDisney) obj).getImagenUrl());
44          pstmt.setString( parameterIndex: 5, ((PrincesasDisney) obj).getAntagonista());
45
46
47          rowCount = pstmt.executeUpdate();
48          return rowCount > 0;
49      }
50

```

Y el de obtener todo: El cual lo que hace es mandar a llamar a todos los registros de la base de datos e insertarlos dentro de un array list para después ser mostrados en la tabla

```

51      @Override
52      public ArrayList obtenerTodo() throws SQLException {
53          String sql = "SELECT * FROM PrincesasDisney";
54          ArrayList<PrincesasDisney> resultado = new ArrayList<>();
55
56
57          Statement stm = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().createStatement();
58          ResultSet rst = stm.executeQuery(sql);
59          while (rst.next()){
60              resultado.add(new PrincesasDisney(rst.getInt( columnIndex: 1),rst.getString( columnIndex: 2),rst.getString( columnIndex: 3),rst.getString( columnIndex: 4),
61              rst.getString( columnIndex: 5),rst.getString( columnIndex: 6)));
62          }
63      }
64

```

Pero igual tuve que agregar un método llamado **"cambiarPorCeldas"** este pide un String que se insertara en la tabla, además como el nombre del método lo dice se requiere la id de la fila y la columna para así poder identificar la celda exacta de cual se cambiara el dato, cabe aclarar que este método no permite que se modifiquen la columna de la "Id".

Lo primero que hace este método es declarar la variable que de sqlUpdate donde se dicta la sentencia que dará la orden a la base de datos, así como la variable del prepared statement la cual reconocerá nuestra base de datos y se encargara de mandar la primera variable a la base de datos y también se declara la variable que detecta si hubo un cambio en las columnas el rowCount.


```

95     public boolean cambiarPorCeldas(String datoActualizado, int Id,int colIndex) throws SQLException {
96         String sqlUpdate;
97         PreparedStatement pstmt;
98         int rowCount;

```

Después de esto ahora si empezamos creando un switch el cual dependiendo de la columna seleccionada este realizara pequeños cambios según el numero de columna que se le asigne.

```

100     switch (colIndex){
101
102         case 1:
103             sqlUpdate = "UPDATE PrincesasDisney SET Nombre = ? WHERE Id = "+Id+"";
104             rowCount = 0;
105             pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
106             pstmt.setString( parameterIndex: 1, (datoActualizado));
107             rowCount = pstmt.executeUpdate();
108
109             return rowCount > 0;
110
111
112         case 2:
113             sqlUpdate = "UPDATE PrincesasDisney SET Cuento = ? WHERE Id = "+Id+"";
114             rowCount = 0;
115             pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
116             pstmt.setString( parameterIndex: 1, (datoActualizado));
117             rowCount = pstmt.executeUpdate();
118
119             return rowCount > 0;
120
121         case 3:
122             sqlUpdate = "UPDATE PrincesasDisney SET ColorVestido = ? WHERE Id = "+Id+"";
123             rowCount = 0;
124             pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
125             pstmt.setString( parameterIndex: 1, (datoActualizado));
126             rowCount = pstmt.executeUpdate();
127
128             return rowCount > 0;
129
130         case 4:
131             sqlUpdate = "UPDATE PrincesasDisney SET Imagen = ? WHERE Id = "+Id+"";
132             rowCount = 0;
133             pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
134             pstmt.setString( parameterIndex: 1, (datoActualizado));
135             rowCount = pstmt.executeUpdate();
136
137             return rowCount > 0;
138
139         case 5:
140             sqlUpdate = "UPDATE PrincesasDisney SET Antagonista = ? WHERE Id = "+Id+"";
141             rowCount = 0;
142             pstmt = ConexionSingleton.getInstance( baseDatos: "PrincesasDisney.db").getConnection().prepareStatement(sqlUpdate);
143             pstmt.setString( parameterIndex: 1, (datoActualizado));
144             rowCount = pstmt.executeUpdate();
145
146             return rowCount > 0;
147
148     }
149
150     return false;

```

En cada método es casi lo mismo que el método de actualizar, la única diferencia es que dependiendo la columna se hace el cambio en el SET del sqlUpdate pues es la columna a modificar y más adelante lo único que se hace es pedir la id para identificar que de que fila se modificara esa columna dándonos la manera de así poder mover las celdas que queramos y lo único que hace es regresar si hubo un cambio en las filas y en caso de no hacerlo manda un false.

Modelo Tabla Princesa

Esta tabla se diseñó como lo hemos estado viendo a través del semestre, primero se implementó el table model para poder sobrescribir los métodos que nos permitirán darle forma a nuestra tabla. Se declara la vista y Princesas DAO para poder usar el contenido de estas de una manera más fácil y nuestra constante que serán nuestro número de columnas que usaremos.

También declaramos un constructor vacío que contenga el array de datos y nuestras princesas dao donde están los métodos y pues se crea la tabla con el diseño visto en clase

```
29      @Override
30      public int getRowCount() { return datos.size(); }
33
34      @Override
35      public int getColumnCount() { return COLUMNAS; }
38
39      @Override
40      public String getColumnName(int columnIndex) {
41
42          switch (columnIndex) {...}
43      }
62
63
64      @Override
65      public Class<?> getColumnClass(int columnIndex) {...}
88
89      @Override
90      public boolean isCellEditable(int rowIndex, int columnIndex) {
91          return true;
92      }
93
94      @Override
95      public Object getValueAt(int rowIndex, int columnIndex) {...}
114
115      @Override
116      public void setValueAt(Object aValue, int rowIndex, int columnIndex) {...}
142
143      @Override
144      public void addTableModelListener(TableModelListener l) {}
147
148      @Override
149      public void removeTableModelListener(TableModelListener l) {}
152
```

De la línea 29 a la 152 no hay gran cambio solo se cambian los nombres de las columnas y el tipo de dato que reciben estas.

Pero dentro de esta clase se crearon múltiples métodos que serán usados dentro del controlador el primer método que tenemos es el de "cargarDatos" el cual manda a llamar el método "obtenerTodo" (De la clase PrincesasDisneyDAO) pero antes lo meto dentro de un array list que contiene objetos de tipo Princesa Disney.

```

153     public void cargarDatos() {
154
155         try {
156             ArrayList<PrincesasDisney> tirar = pdao.obtenerTodo();
157
158             datos = pdao.obtenerTodo();
159         } catch (SQLException e) {
160             System.out.println(e.getMessage());
161         }
162     }
163

```

El siguiente método que tenemos es el de **agregarPrincesas** a la tabla y a la base de datos puesto que primero nos pide un objeto de tipo Princesa Disney. Para luego mandar a llamar desde la clase PrincesasDisneyDAO el método insertar al cual se le asigna objeto que se pidió anteriormente y este nos arroja un booleano para saber si se completo o no además de que este método se nos pide tratar la excepción por lo que lo metemos en un try / catch para evitar errores.

```

164     public boolean agregarPrincesa(PrincesasDisney princesa) {
165         boolean resultado = false;
166         try {
167             if (pdao.insertar(princesa)) {
168                 datos.add(princesa);
169                 resultado = true;
170             } else {
171
172                 resultado = false;
173             }
174         } catch (SQLException e) {
175             System.out.println(e.getMessage());
176         }
177         return resultado;
178     }

```

El siguiente método es el de **getPrincesasAtIndex** el cual nos pide un índice de alguna columna y lo que hace este es regresar el dato que se encuentra en ese índice.

```

3 usages
180     public PrincesasDisney getPrincesaAtIndex(int i) {
181
182         return datos.get(i);
183     }
184

```

Ya casi al final como penúltimo método tenemos el de `borrarFila` el cual nos pide un id y lo único que hace es mandar a llamar el método `delete` y para que la maquina sepa que se borrara se requiere el id. También regresa un booleano para ver si funciona.

```
1 usage
185 public boolean borrarFila( int index) throws SQLException {
186
187     if (pdo.delete(index)) {
188
189         return true;
190
191     } else {
192         return false;
193     }
194 }
195
196
```

Como ultimo metodo tenemos el de `actualizarCelda` el cual nos pide 3 cosas, la primera es la cadena del dato que se actualizara, luego el id y el numero de la columna. Este metodo manda a llamar el metodo `cambiarPorCeldas` de la clase `PrincesasDisneyDAO` y se le otorgan estos 3 datos para que nuestro metodo. Asi como los demas metodos este regresa un booleano para saber si funciona.

```
1 usage
197 public boolean actualizarCelda(String datoActualizado, int id, int colIndex) throws SQLException {
198     if (pdo.cambiarPorCeldas(datoActualizado,id,colIndex)){
199         return true;
200     }else {
201         return false;
202     }
203 }
204
```

Controlador Princesas

Por último tenemos la clase de controlador en la cual para iniciar se implementa el `MouseListener` puesto que solo usaremos ese método y los demás no, procedemos a declarar la vista y el modelo para poder completar nuestro modelo vista controlador y que se conecten. Después se crea un constructor que nos pide la vista y se le declaran todos los botones y objetos que implementaran este `MouseListener` así como el modelo de donde sacaremos los métodos.

```

3 usages
16 public class ControladorPrincesas extends MouseAdapter {
17
18     38 usages
19     private VentanaPrincesas vista;
20     16 usages
21     private ModeloTablaPrincesa modelo;
22
23     1 usage
24     public ControladorPrincesas(VentanaPrincesas vista) {
25         this.vista = vista;
26         this.vista.getTblPrincesas().addMouseListener( this );
27         this.vista.getBtnCargar().addMouseListener( this );
28         this.vista.getBtnAgregar().addMouseListener( this );
29         modelo = new ModeloTablaPrincesa();
30         this.vista.getTblPrincesas().setModel(modelo);
31         this.vista.getBtnActualizar().addMouseListener( this );
32         this.vista.getBtnBorrar().addMouseListener( this );
33     }

```

Entonces ya luego mandamos a llamar el único método que no interesa el mouseClicked y para saber de donde viene la señal se hace con if comparando el objeto que nos interesa con la señal dentro de la variable e.

Primero le dimos su acción al botón de cargar lo que hace es mandar a llamar el método de **cargarDatos** de la clase de ModeloTablaPrincesas para luego mandar ese modelo a la tabla y por último se actualiza esta tabla para mostrar el contenido de la base de datos en la tabla de nuestro programa.

```

33
34 @Override
35 public void mouseClicked(MouseEvent e) {
36     if(e.getSource() == this.vista.getBtnCargar()){
37
38
39         modelo.cargarDatos();
40         this.vista.getTblPrincesas().setModel(modelo);
41         this.vista.getTblPrincesas().updateUI();
42
43
44
45
46     }

```

El siguiente botón al que le dimos su acción al botón de agregar el cual lo que hace es recuperar la información que se encuentra en los text fields para ser asignados dentro de un objeto de tipo princesas para después mandar a llamar el método de **agregarPrincesa** y si este se cumple mande un cuadro de dialogo donde se diga que se completó con éxito la acción y además se actualice la tabla y muestre la implantación de nuestro registro. Además de que cuando acabe se limpien nuestros campos para poder ser llenados de nuevo.

```
47 if(e.getSource() == this.vista.getBtnAgregar()){
48
49     PrincesasDisney princesa = new PrincesasDisney();
50     princesa.setId(0);
51     princesa.setNombre(this.vista.getTxtNombre().getText());
52     princesa.setCuento(this.vista.getTxtCuento().getText());
53     princesa.setColorVestido(this.vista.getTxtColorVestido().getText());
54     princesa.setImagenUrl(this.vista.getTxtImagenUrl().getText());
55     princesa.setAntagonista(this.vista.getTxtAntagonista().getText());
56
57     if(modelo.agregarPrincesa(princesa)){
58         JOptionPane.showMessageDialog(vista, message: "Se agrego correctamente", title: "Confirmacion", JOptionPane.INFORMATION_MESSAGE);
59
60         modelo.cargarDatos();
61         this.vista.getTblPrincesas().setModel(modelo);
62         this.vista.getTblPrincesas().updateUI();
63
64     }else {
65
66     }
67     JOptionPane.showMessageDialog(vista, message: "No se pudo agregar correctamente, revise su conexion", title: "Error", JOptionPane.ERROR_MESSAGE);
68
69     this.vista.limpiar();
70 }
```

El siguiente evento se comparó con la tabla para asignar el evento de seleccionar una fila y nos regresara una imagen esto se hizo primero sabiendo que fila se seleccionó de la tabla convierto ese dato en un tipo entero llamado index para luego con ese indice mandar a llamar el método de **getPrincesaAtIndex** lo cual nos regresaba el dato del arraylist en esa posición para así meterlo en una variable temporal.

Esa variable temporal serviría para de ahí sacar la url de esa princesa en especifico y dársela al método de **getImagen** de la clase PrincesasDisney que lo que hace es transformar esa cadena en un tipo de dato de URL para así poder retornar la imagen e insertarla dentro de un label asignado al panel 3. Todo esto dentro de un catch que se nos solicita por ser una excepción atendida.

```
74
75 try {
76     int index = this.vista.getTblPrincesas().getSelectedRow();
77     PrincesasDisney tmp = modelo.getPrincesaAtIndex(index);
78     this.vista.getLblImg().setIcon(tmp.getImagen());
79 } catch (MalformedURLException ex) {
80     throw new RuntimeException(ex);
81 }
82 this.vista.getLblImg().setText("");
83 }
```

El siguiente botón fue el de borrar el cual lo primero que hace es sacar el id de la fila seleccionada de manera interna mientras le pregunta al usuario si esta seguro de querer borrar el registro seleccionado mostrándole la id de dicha fila. Una vez que se confirma se manda un booleano el cual si es true lo que hace este es mandar a llamar el método

borrarFila y se le asigna el id que habíamos sacado para así eliminar dicho registro de la base de datos y así actualizar la tabla para mostrarse en el programa

```
84
85 if(e.getSource() == this.vista.getBtnBorrar()){
86     int index = this.vista.getTblPrincesas().getSelectedRow();
87     PrincesasDisney tmp = modelo.getPrincesaAtIndex(index);
88     int resultado = JOptionPane.showConfirmDialog(vista, message: "Esta seguro que desea borrar el registro "+(tmp.getId())+"?", title: "Confirmacion", JOptionPane.YES_NO_OPTION);
89     if(resultado == JOptionPane.YES_NO_OPTION) {
90         try {
91             if (modelo.borrarFila(tmp.getId())) {
92                 JOptionPane.showMessageDialog(vista, message: "Se borro correctamente", title: "Aviso", JOptionPane.INFORMATION_MESSAGE);
93                 modelo.cargarDatos();
94                 this.vista.getTblPrincesas().setModel(modelo);
95                 this.vista.getTblPrincesas().updateUI();
96             } else {
97                 JOptionPane.showMessageDialog(vista, message: "No se pudo borrar correctamente, revise su conexion", title: "Error", JOptionPane.ERROR_MESSAGE);
98             }
99         } catch (SQLException ex) {
100             throw new RuntimeException(ex);
101         }
102     }
103 }
104
105
106
107
108 }
```

El ultimo boton asignado es el de actualizar el cual te pregunta que es lo que vas a meter en la celda por medio del cuadro de dialogo y de manera simultanea este programa lo que hace es sacar la id de la fila seleccionada y el numero de columnan para dar con la celda para todo esto otorgarselo al metodo de **actualizarCelda** que nos pide estos tres datos para ya solo esperar el booleano para poder ver si se logro o no y asi solo cargarlo al modelo y del modelo a la tabla para ser actualizada y mostrado en el programa.

```
120
121 if (e.getSource() == this.vista.getBtnActualizar()) {
122     int index = this.vista.getTblPrincesas().getSelectedRow();
123     PrincesasDisney tmp = modelo.getPrincesaAtIndex(index);
124     int id = tmp.getId();
125     String datoActualizado = JOptionPane.showInputDialog(vista, message: "Inserte el dato a actualizar de "+
126     this.vista.getTblPrincesas().getColumnName(vista.getTblPrincesas().getSelectedColumn())+" del registro "
127     +id, title: "Actualizar",JOptionPane.INFORMATION_MESSAGE);
128
129     int colIndex = this.vista.getTblPrincesas().getSelectedColumn();
130
131     System.out.println(colIndex+" "+datoActualizado+" "+id);
132
133     try {
134         if(modelo.actualizarCelda(datoActualizado,id,colIndex)){
135             System.out.println("se logro");
136             modelo.cargarDatos();
137             this.vista.getTblPrincesas().setModel(modelo);
138             this.vista.getTblPrincesas().updateUI();
139         }else{
140             System.out.println("no se logro");
141         }
142     } catch (SQLException ex) {
143         throw new RuntimeException(ex);
144     }
145 }
```

Main

Por último está el main donde lo único que se hace es crear la ventana del programa y después se crea el controlador y se le asigna la vista para poder crear una vez mas el modelo vista controlador y que así opere de mejor manera el programa y sea más ordenado.

```

11 ▶ public class Main {
    no usages
12 ▶ public static void main(String[] args) {
13     VentanaPrincesas vista = new VentanaPrincesas( title: "Princesas Disney");
14
15     ControladorPrincesas controlador = new ControladorPrincesas(vista);
16
17
18
19
20 }
21 }

```

Lo que permite que nuestro programa se vea así:

Princesas Disney

Id:

Nombre:

Cuento:

Color del vestido:

Imagen:

Antagonista:

Agregar

Cargar

Id	Nombre	Fabula	ColorVestido	Imagen	Antagonista

Foto: ...

Actualizar

Borrar

Hasta que se le da al boton de cargar para mostrar el contenido de la base de datos:

Princesas Disney

Id

Nombre

Cuento:

Color del vestido:

Imagen:

Antagonista:

Agregar


Cargar

Id	Nombre	Fabula	ColorVestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image/...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	Moana	Naranja	https://encr...	Tefiti

Actualizar

Borrar

Foto:



Cuando se agrega un registro se le manda un mensaje al usuario de confirmación y por ultimo se actualiza la base de datos.

Princesas Disney

Id **Nombre**

Cuento:

Color del vestido:

Imagen:

Antagonista: **Agregar**

Foto: ...

Cargar					
Id	Nombre	Fabula	ColorVestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	Moana	Naranja	https://encr...	Tefiti

Actualizar **Borrar**

Confirmacion

Se agrego correctamente

OK

Princesas Disney

Id

0

Nombre

Cuento:

Color del vestido:

Imagen:

Antagonista:

Agregar

Cargar

Id	Nombre	Fabula	ColorVestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	Moana	Naranja	https://encr...	Tefiti
5	Mulan	Mulan	Azul	https://encr...	Shan Yu

Foto: ...

Actualizar

Borrar

Y así se ve cuando se selecciona una fila de la tabla para mostrar la foto de la princesa:

Princesas Disney

Id: 0

Nombre:

Cuento:

Color del vestido:

Imagen:

Antagonista:

Agregar

Cargar

Id	Nombre	Fabula	Color/Vestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	Moana	Naranja	https://encr...	Tefiti
5	Mulan	Mulan	Azul	https://encr...	Shan Yu

Foto:

Actualizar

Borrar

Así se ve cuando se ocupa el botón de actualizar, se pide el dato a actualizar y se muestra de que celda se cambiaran los datos:

Princesas Disney

Id 0

Nombre

Cuento:

Color del vestido:

Imagen:

Antagonista:

Agregar

Cargar

Id	Nombre	Fabula	Color/Vestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	MOANA	Naranja	https://encr...	Tefiti
5	Mulan	Mulan	Azul	https://encr...	Shan Yu

Actualizar

Borrar

Foto:

Por ultimo así se ve el botón de borrar para en este caso borrar el registro 6 y al darle confirmar automáticamente se actualiza la tabla y se muestra el cambio:

Confirmacion

?

Esta seguro que desea borrar el registro 6?

Yes

No

Aviso

i

Se borro correctamente

OK

Cargar

Id	Nombre	Fabula	ColorVestido	Imagen	Antagonista
1	Aurora	Bella durmi...	Rosa	https://encr...	Malefica
2	Blancaniev...	Blanca Nie...	Azul con a...	data:image...	Reina Grim...
3	Ariel	La sirenita	morado	https://encr...	Ursula
4	Moana	MOANA	Naranja	https://encr...	Tefiti
5	Mulan	Mulan	Azul	https://encr...	Shan Yu