



UNIVERSIDAD AUTOMA DEL CARMEN
DES CIENCIAS DE LA INFOMACION
INGENIERIA EN SISTEMAS COMPUTACIONALES



Integrante:

Cesar Julio Beltran

Profesor (a):

José Ángel Pérez Rejón

Materia:

Programación Visual

Carrera:

Ing. En Sistemas Computacionales

Manual técnico:

“Sistema de venta de boletos”

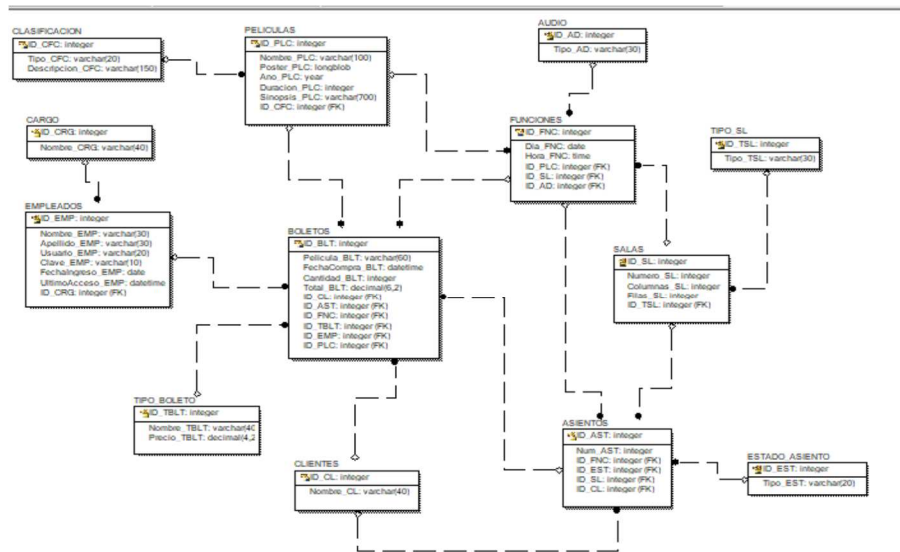
Fecha:

02/12/19

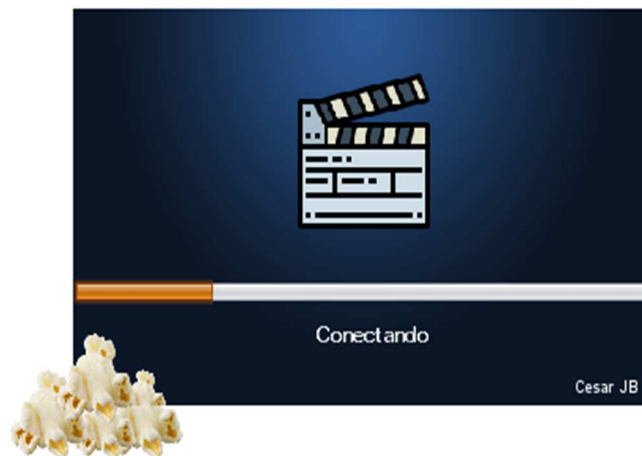
Objetivo general del sistema

Satisfacer las necesidades de un cine para poder llevar el control de ventas de los boletos vendidos. Poder agregar, modificar, eliminar, consultar y generar reportes de las películas, funciones de cine, salas, empleados y todo lo necesario para llevar el control del cinema en cuestión.

Modelo de base de datos:



Carga de componentes del programa:



Al inicio del sistema se muestra una venta donde se cargan los componentes del sistema, con una función timer se proporciona el tiempo en milisegundos de en cuanto tiempo se abrirá el sistema.

```

public VentanaDeCarga()
{
    initComponents();

    //
    this.setLocationRelativeTo(null);
    
```

```
//
AWTUtilities.setWindowOpaque(this, false);

al = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //
        x += 1;
        //
        barraDeProgreso.setValue(x);

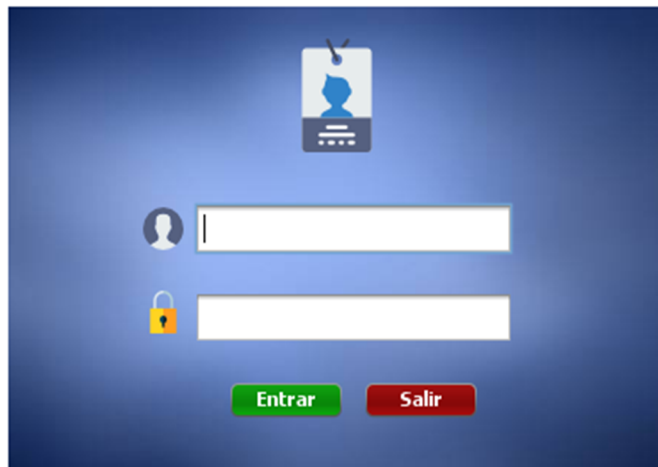
        //
        if(barraDeProgreso.getValue() == 1)
        {
            mostrarCarga.setText("Iniciando componentes");
            ConexionBD a = new ConexionBD();
        }
        if(barraDeProgreso.getValue() == 20)
        {
            mostrarCarga.setText("Conectando");
        }
        if(barraDeProgreso.getValue() == 45)
        {
            mostrarCarga.setText("Cargando recursos");
        }
        if(barraDeProgreso.getValue() == 65)
        {
            mostrarCarga.setText("Iniciando proyecto");
        }
        if(barraDeProgreso.getValue() == 80)
```

```

{
    mostrarCarga.setText("Abriendo proyecto");
}
if(barraDeProgreso.getValue() == 100)
{
    dispose();
    VentanaInicioDeSesion vis = new VentanaInicioDeSesion();
    vis.setVisible(true);
    tm.stop();
}
}
};

```

Inicio de sesión:



En la ventana de inicio de sesión, se hace una búsqueda en la base de datos con el nombre de usuario y la contraseña y dependiendo de que tipo de cargo tenga el empleado, se abrirá el panel de administrador o la parte para poder vender los boletos.

```

private void botonEntrarMouseClicked(java.awt.event.MouseEvent evt) {
    GetSetEmpleados gse = new GetSetEmpleados();

    Date fecha = new Date();
    DateFormat formato = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
}

```

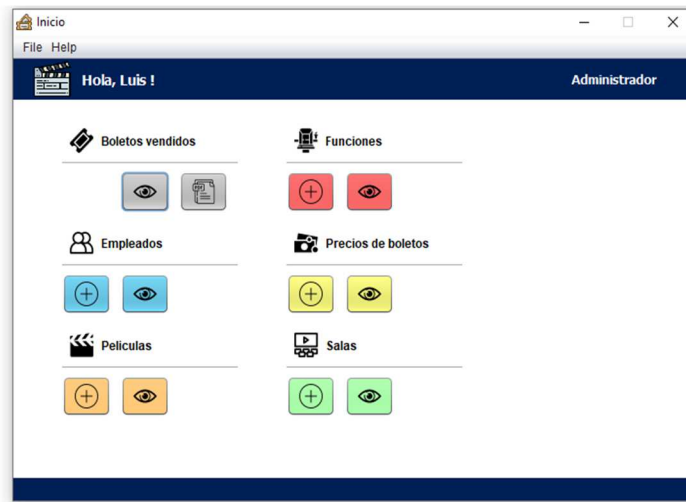
```
String clave = new String(claveLogin.getPassword());

if(!usuarioLogin.getText().equals("") && !clave.equals(""))
{
    gse.setUsuario(usuarioLogin.getText());
    gse.setClave(clave);
    gse.setUltimoAcceso(formato.format(fecha));

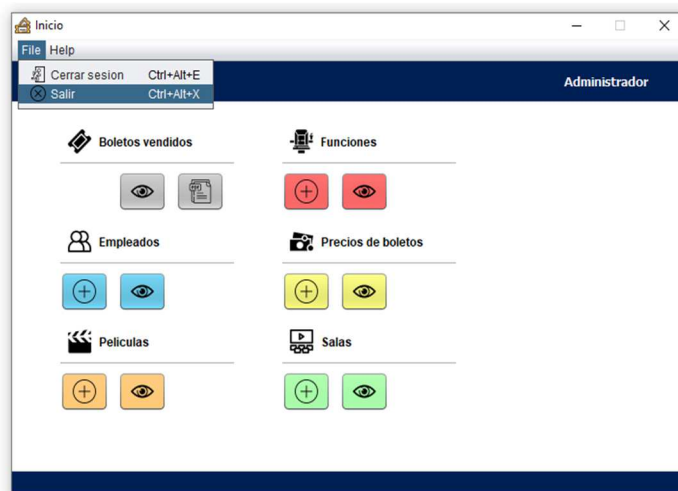
    if(cnbd.getComprobarLogin(gse))
    {
        VentanaPrincipalAdministrador vpa = new VentanaPrincipalAdministrador(gse);
        VentanaPrincipalCajero vpb = new VentanaPrincipalCajero(gse);

        if(gse.getCargo() == 1)
        {
            this.dispose();
            vpa.setVisible(true);
        }
        else if(gse.getCargo() == 2)
        {
            this.dispose();
            vpb.setVisible(true);
        }
    }
    else
        JOptionPane.showMessageDialog(null,"Los datos son incorrectos","Inciar
sesion", 0);
}
else
    JOptionPane.showMessageDialog(null,"Los campos estan vacios","Inciar sesion", 2);
}
```

Inicio del panel de Administrador:



File -> Cerrar Sesión -> Salir:



Al mostrar el panel del administrador da la posibilidad de cerrar sesión para poder regresar a la ventana de inicio de sesión y cambiar de usuario o salir de todo el programa.

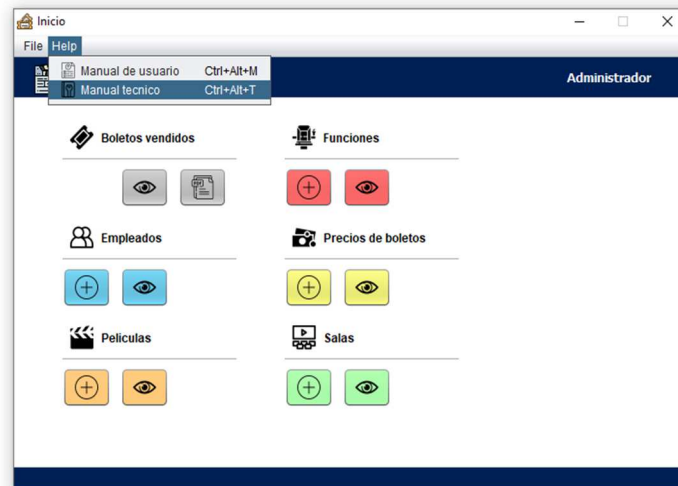
```
private void menuCerrarSesionActionPerformed(java.awt.event.ActionEvent evt) {  
    VentanaInicioDeSesion vis = new VentanaInicioDeSesion();  
  
    this.dispose();  
    vis.setVisible(true);  
}  
  
private void menuSalirActionPerformed(java.awt.event.ActionEvent evt) {
```

```

System.exit(0);
}

```

Help -> Manual de usuario -> Manual técnico:



Ver boletos vendidos:



En la ventana creo una tabla y realizo una consulta de todos los boletos vendidos, además de poder visualizar por cada película en un combo Box.

```

public void setLlenarTablaVentas(String consulta)
{
    try
    {
        ResultSet rs = cnbd.getLlenarComboBox(consulta);
    }
}

```

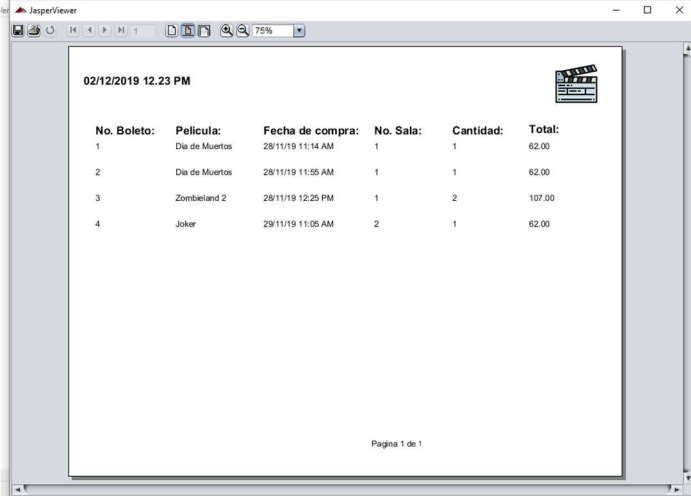


```

while(rs.next())
{
    dtm.addRow(new Object[] {rs.getInt("ID_BLT"), rs.getString("Pelicula_BLT"),
rs.getString("FechaCompra_BLT"), rs.getInt("NumeroSala_BLT"), rs.getInt("Cantidad_BLT"),
rs.getString("Dia_BLT") + " " + rs.getString("Hora_BLT"), rs.getFloat("Total_BLT")});
}
}
catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: "+e,"!! ERROR !!", 2);
}
}

```

Reporte de boletos vendidos:



No. Boleto:	Pelicula:	Fecha de compra:	No. Sala:	Cantidad:	Total:
1	Dia de Muertos	28/11/19 11:14 AM	1	1	62.00
2	Dia de Muertos	28/11/19 11:55 AM	1	1	62.00
3	Zombieland 2	28/11/19 12:25 PM	1	2	107.00
4	Joker	28/11/19 11:05 AM	2	1	62.00

Pagina 1 de 1

Abro un documento de tipo JasperReport con toda la información de las ventas realizadas.

```

private void botonReporteBoletosMouseClicked(java.awt.event.MouseEvent evt) {
    try
    {
        ConexionBD cnbd = new ConexionBD();

        JasperReport reporte = null;
    }
}

```

```

String path = "src\\Reportes\\reporteVentas.jasper";

reporte = (JasperReport) JRLoader.loadObjectFromFile(path);

JasperPrint jPrint = JasperFillManager.fillReport(path, null, cnbd.getConectar());

JasperViewer view = new JasperViewer(jPrint, false);

view.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

view.setVisible(true);
}
catch (JRException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Reporte generado", 2);
}
}

```

Agregar empleados:

The screenshot shows a Java Swing window titled "Registrar empleados". The window has a standard Mac OS-style title bar with "Registrar empleados" and standard window controls. Below the title bar is a menu bar with "File" and "View". The main content area has a dark blue header with the text "Registro de empleados". The form contains the following fields and controls:

- ID:** A text input field containing the number "5".
- Cargo:** A dropdown menu with the text "Selecciona" and a downward arrow.
- Nombre:** A text input field.
- Apellido(s):** A text input field.
- Fecha de ingreso:** A text input field next to a small calendar icon.
- Usuario:** A text input field.
- Contraseña:** A text input field.
- Repetir contraseña:** A text input field.
- Guardar:** A button at the bottom center of the form.

Para poder registrar un usuario, verifico que los campos estén correctos y si no muestro un mensaje en pantalla de cada error. También compruebo que el nombre de usuario no este ocupado por otro empleado.

```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {
```

```
    GetSetEmpleados gse = new GetSetEmpleados();
```

```
    String cl1 = claveEmpleado.getText();
```

```
    String cl2 = claveConfirmaEmpleado.getText();
```

```
    boolean bnd = false;
```

```
    if(nombreEmpleado.getText().equals(""))
```

```
        errorNombre.setText("**Nombre incorrecto");
```

```
    else
```

```
        errorNombre.setText(null);
```

```
    if(apellidoEmpleado.getText().equals(""))
```

```
        errorApellido.setText("**Apellido incorrecto");
```

```
    else
```

```
        errorApellido.setText(null);
```

```
    try
```

```
    {
```

```
        if(ingresoEmpleado.getDate().toString().length() > 1)
```

```
            errorFecha.setText(null);
```

```
        bnd = true;
```

```
    }
```

```
    catch(Exception e)
```

```
    {
```

```
        errorFecha.setText("**Fecha incorrecta");
```

```
    }
```

```
    if(comboCargo.getSelectedItem() == "Selecciona")
```

```

        errorCargo.setText("**Seleccione una opcion valida");
    else
        errorCargo.setText(null);

    if(usuarioEmpleado.getText().equals(""))
        errorUsuario.setText("**Usuario incorrecto");
    else
        errorUsuario.setText(null);

    if(claveEmpleado.getText().equals(""))
        errorCl1.setText("**Contraseña incorrecta");
    else
        errorCl1.setText(null);

    if(claveConfirmaEmpleado.getText().equals(""))
        errorCl2.setText("**Contraseña incorrecta");
    else
        errorCl2.setText(null);

    if(!cl1.equals(cl2))
    {
        errorCl11.setText("**Contraseña incorrecta");
        errorCl22.setText("**Contraseña incorrecta");
    }
    else
    {
        errorCl11.setText(null);
        errorCl22.setText(null);
    }

    if(!cl1.equals("") && !cl2.equals("") && !usuarioEmpleado.equals("") & bnd &&
    comboCargo.getSelectedItem() != "Selecciona"

```

```

        && !nombreEmpleado.getText().equals("") && !apellidoEmpleado.getText().equals(""))
    {
        if(cl1.equals(cl2))
        {
            if(cnbd.getComprobarUsuario(usuarioEmpleado.getText()) == 0)
            {

                Date dt = ingresoEmpleado.getDate();

                long tmp = dt.getTime();

                java.sql.Date fecha = new java.sql.Date(tmp);

                gse.setId(Integer.parseInt(idEmpleado.getText()));
                gse.setNombre(nombreEmpleado.getText());
                gse.setApellido(apellidoEmpleado.getText());
                gse.setUsuario(usuarioEmpleado.getText());
                gse.setClave(claveEmpleado.getText());
                gse.setFechaIngreso(fecha.toString());
                gse.setCargo(comboCargo.getSelectedIndex());

                if(cnbd.getRegistrarUsuarios(gse))
                {
                    JOptionPane.showMessageDialog(null,"El empleado ha sido guardado","Empleado
agregado", 1);
                    setLimpiarFormulario();
                    idEmpleado.setText(String.valueOf(cnbd.getGenerarId("SELECT MAX(ID_EMP)
FROM empleados")));
                }
                else
                {
                    JOptionPane.showMessageDialog(null,"No se han podido guardar los
datos","Registrar empleado", 2);
                }
            }
        }
    }

```

```

        else

            JOptionPane.showMessageDialog(null,"El usuario ya esta registrado","Usuario
existente", 0);

        }

        else

            JOptionPane.showMessageDialog(null,"Las contraseñas ingresadas son
diferentes","Contraseña incorrecta", 0);

        }

        else

            JOptionPane.showMessageDialog(null,"Llene todos los campos para continuar","Datos
incorrectos", 2);

        }

```

Ver Empleados:



Hago una búsqueda de los nombres de los empleados y muestro una lista de ellos en el combo box, además tengo la posibilidad de buscarlos por su id de usuario.

```

public void setLLenarComboEmpleados()
{
    ArrayList<String> lista = new ArrayList <String>();
    ResultSet rs = cnbd.getLLenarComboBox("SELECT * FROM empleados");

    try

```

```

{
    while(rs.next())
    {
        lista.add(rs.getInt("ID_EMP") + " - " + rs.getString("Nombre_EMP") + " " +
rs.getString("Apellido_EMP"));
    }
}
catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Llenar combo box", 2);
}

for(int i = 0; i < lista.size(); i++)
    comboEmpleados.addItem(lista.get(i));
}

```



Una vez seleccionado el empleado hago una consulta con su id, para poder mostrar toda su información.

```

public void setMostrarEmpleados(char id)
{

```

```
ResultSet rs = cnbd.getConsultas("SELECT e.*, c.Nombre_CRG FROM empleados e INNER  
JOIN cargo c ON e.ID_CRG = c.ID_CRG WHERE e.ID_EMP = " + id);
```

```
try  
{  
    if(rs.next())  
    {  
        idEmpleado.setText(String.valueOf(rs.getInt("ID_EMP")));  
        nombreEmpleado.setText(rs.getString("Nombre_EMP"));  
        apellidoEmpleado.setText(" " + rs.getString("Apellido_EMP"));  
        cargoEmpleado.setText(rs.getString("Nombre_CRG"));  
        ingresoEmpleado.setText(rs.getString("FechaIngreso_EMP"));  
        usuarioEmpleado.setText(rs.getString("Usuario_EMP"));  
        accesoEmpleado.setText(rs.getString("UltimoAcceso_EMP"));  
        claveEmpleado.setText(rs.getString("Clave_EMP"));  
    }  
}  
catch (SQLException e)  
{  
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Informacion", 2);  
}  
}
```

The screenshot shows a Java Swing window titled "Ver empleados" with a menu bar containing "File" and "View". The window displays employee information for "Cesar Julio Beltran". The interface includes a search bar at the top with a magnifying glass icon. Below the search bar, the employee's name is split into "Cesar" and "Julio Beltran" in separate text boxes. To the right of these boxes are icons for editing (a pencil) and deleting (a trash can). The "Cargo" is set to "Administrador" in a dropdown menu. The "Fecha de ingreso" is "2019-11-01". The "Nombre de usuario" is "admin" and the "Contraseña" is "1234". The "Ultimo acceso" is "2019-11-29 11:23:44.0". At the bottom, there are two buttons: "Cancelar" (red) and "Guardar" (green). A small icon of a person with a clipboard is visible on the left side of the form.

Al seleccionar el botón editar, tengo la posibilidad de editar los campos y si doy clic en el botón de guardar se cambiarán los datos en la base de datos, si doy a cancelar los campos regresarán a la normalidad.

```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

    if(!editUsuario.getText().equals("") && !editClave.getText().equals("") &&
    !editNombre.getText().equals("") && !editApellido.getText().equals(""))

    {

        String consulta = "SELECT COUNT(ID_EMP) FROM empleados WHERE Usuario_EMP = "
+ editUsuario.getText() + " AND ID_EMP != " + idEmpleado.getText();

        if(cnbd.getBuscar(consulta) == 0)

        {

            GetSetEmpleados gsee = new GetSetEmpleados();

            int crg = editCargo.getSelectedIndex() + 1;

            gsee.setId(Integer.parseInt(idEmpleado.getText().trim()));
            gsee.setNombre(editNombre.getText().trim());
            gsee.setApellido(editApellido.getText().trim());
            gsee.setUsuario(editUsuario.getText().trim());
            gsee.setClave(editClave.getText().trim());
            gsee.setCargo(crg);

            if(cnbd.getModificarEmpleados(gsee))

            {

                this.setOcultarComponentes();

                comboEmpleados.setSelectedItem("Selecciona");

                this.setRemoverItems();
```

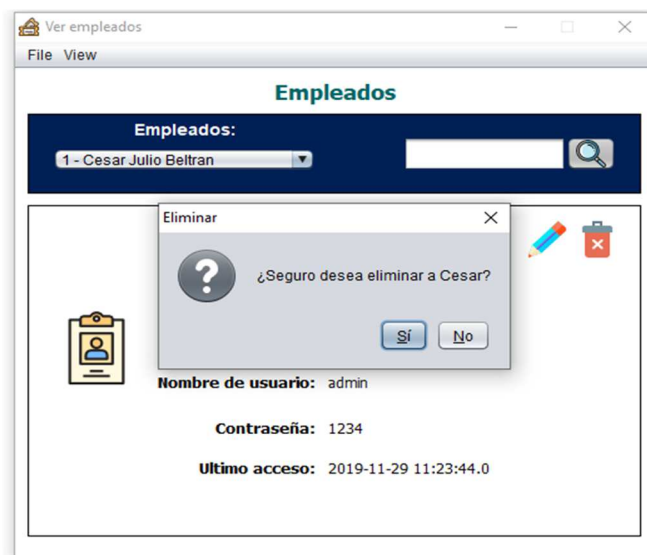
```

        if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)
            this.setRemoverItems();

        if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)
            this.setRemoverItems();

        if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)
            this.setRemoverItems();
        this.setLlenarComboEmpleados();
        JOptionPane.showMessageDialog(null,"El empleado ha sido modificado","Empleado
modificado", 1);
    }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido modificar al
empleado","Modificar empleados", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Este usuario ya esta registrado","Usuario ya
registrado", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Modificar
empleado", 2);
    }

```



Si doy clic al botón de eliminar, pregunto si quiere eliminar y si da clic que si el empleado será eliminado.

```
private void eliminarEmpleadoMouseClicked(java.awt.event.MouseEvent evt) {  
    int op = 0;  
  
    op = JOptionPane.showConfirmDialog(null, "¿ Seguro desea eliminar a " +  
nombreEmpleado.getText() + "?", "Eliminar", JOptionPane.YES_NO_OPTION);  
  
    if(op == JOptionPane.YES_OPTION)  
    {  
        if(cnbd.getEliminar("DELETE FROM empleados WHERE ID_EMP = " +  
idEmpleado.getText().trim()))  
        {  
            this.setOcultarComponentes();  
  
            comboEmpleados.setSelectedItem("Selecciona");  
  
            this.setRemoverItems();  
  
            if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)  
                this.setRemoverItems();  
  
            if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)  
                this.setRemoverItems();  
  
            if(comboEmpleados.getItemCount() == 1 || comboEmpleados.getItemCount() > 1)  
                this.setRemoverItems();  
  
            this.setLLenarComboEmpleados();  
  
            JOptionPane.showMessageDialog(null, nombreEmpleado.getText() + " ha sido  
eliminado","Empleado eliminado", 1);  
        }  
    }  
}
```

```

else

    JOptionPane.showMessageDialog(null,"No se ha podido eliminar al empleado","Eliminar
empleado", 2);

}

}

```

Agregar películas:

Compruebo que el nombre de la película no este ya agregada y que los demás campos estén correctos, una vez verificados agrego la película.

```

private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

```

```

    boolean band = false;

```

```

    if(textTitulo.getText().equals(""))

```

```

        errorTitulo.setText("**Titulo incorrecto");

```

```

    else

```

```

        errorTitulo.setText(null);

```

```

    if (comboClasificacion.getSelectedItem() == "Selecciona")

```

```

        errorClasificacion.setText("**Seleccione una opcion valida");

```

```

    else

```

```

        errorClasificacion.setText(null);

```

```

if(textSinopsis.getText().equals(""))
    errorSinopsis.setText("*Sinopsis incorrecto");
else
    errorSinopsis.setText(null);

try
{
    if(!lgsp.getPoster().equals(""))
        errorImagen.setText(null);
}
catch(Exception e)
{
    errorImagen.setText("*Seleccione una imagen");
}

if(textDuracion.getValue() > 1)
{
    errorDuracion.setText(null);
    band = true;
}
else
    errorDuracion.setText("*Duracion incorrecta");

if(!textTitulo.getText().equals("") && !textSinopsis.equals("") &&
!comboClasificacion.equals("Selecciona") &&
    textDuracion.getValue() > 1 & band)
{
    String consulta = "SELECT COUNT(ID_PLC) FROM peliculas WHERE Nombre_PLC = " +
textTitulo.getText() + """;

    if(cnbd.getBuscar(consulta) == 0)
    {

```

```

        gsp.setId(Integer.parseInt(textId.getText()));
        gsp.setTitulo(textTitulo.getText());
        gsp.setAño(textAño.getYear());
        gsp.setDuracion(textDuracion.getValue());
        gsp.setClasificacion(comboClasificacion.getSelectedIndex());
        gsp.setSinopsis(textSinopsis.getText());

        if(cnbd.getRegistrarPeliculas(gsp))
        {
            JOptionPane.showMessageDialog(null,"La pelicula ha sido guardado","Registrar
empleado", 1);

            setLimpiarFormulario();

            textId.setText(String.valueOf(cnbd.getGenerarId("SELECT MAX(ID_PL) FROM
peliculas"))));
        }
        else
            JOptionPane.showMessageDialog(null,"No se han podido guardar los
datos","Registrar peliculas", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"La pelicula ya esta registrada","Pelicula ya
registrada", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Llene todos los campos para continuar","Datos
incorrectos", 2);
}

private void comboClasificacionItemStateChanged(java.awt.event.ItemEvent evt) {
    if(comboClasificacion.getSelectedItem() == "Selecciona")
        errorClasificacion.setText("**Seleccione una opcion valida");
    else
        errorClasificacion.setText(null);
}

```

```

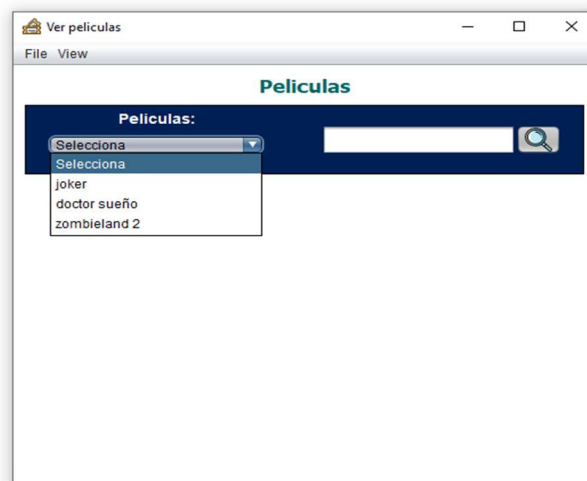
}

private void textTituloKeyTyped(java.awt.event.KeyEvent evt) {
    if(textTitulo.getText().equals(""))
        errorTitulo.setText("*Titulo incorrecto");
    else
        errorTitulo.setText(null);
}

private void textSinopsisKeyTyped(java.awt.event.KeyEvent evt) {
    if(textSinopsis.getText().equals(""))
        errorSinopsis.setText("*Sinopsis incorrecto");
    else
        errorSinopsis.setText(null);
}

```

Ver películas -> Selecciona nombre de la película:



Hago una consulta de el nombre de las películas y muestro la lista de nombres de las películas.

```

public void setMostrarPelículas(int id)
{
    panelPoster.removeAll();

    this.setPoster("SELECT Poster_PLC FROM peliculas WHERE ID_PLC = " + id);
}

```

```
ResultSet rs = cnbd.getConsultas("SELECT p.*, c.Tipo_CFC FROM peliculas p INNER JOIN  
clasificacion c ON p.ID_CFC = c.ID_CFC where ID_PLC = " + id);
```

```
try  
{  
  
    if(rs.next())  
    {  
        idPelicula.setText(String.valueOf(rs.getInt("ID_PLC")));  
        tituloPelicula.setText(rs.getString("Nombre_PLC"));  
        clasificacionPelicula.setText(rs.getString("Tipo_CFC"));  
        anoPelicula.setText(String.valueOf(rs.getInt("Ano_PLC")));  
        duracionPelicula.setText(String.valueOf(rs.getInt("Duracion_PLC")));  
        sinopsisPelicula.setText(rs.getString("Sinopsis_PLC"));  
    }  
}  
catch (SQLException e)  
{  
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Informacion", 2);  
}
```




```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

    if(!editTitulo.getText().equals("") && !sinopsisPelicula.getText().equals("") &&
    editAno.getValue() != 0 && editDuracion.getValue() != 0)

    {

        String consulta = "SELECT COUNT(ID_PLC) FROM peliculas WHERE Nombre_PLC = " +
        tituloPelicula.getText() + " AND ID_PLC != " + idPelicula.getText();

        if(cnbd.getBuscar(consulta) == 0)

        {

            GetSetPeliculas gsp = new GetSetPeliculas();

            int cl = editClasificacion.getSelectedIndex() + 1;

            gsp.setId(Integer.parseInt(idPelicula.getText()));
            gsp.setTitulo(editTitulo.getText().trim());
            gsp.setAno(editAno.getValue());
            gsp.setDuracion(editDuracion.getValue());
            gsp.setSinopsis(sinopsisPelicula.getText());
            gsp.setClasificacion(cl);

            if(cnbd.getModificarPeliculas(gsp))

            {

                this.setOcultarComponentes();

                comboPeliculas.setSelectedItem("Selecciona");

                this.setRemoverItems();

                if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)

                    this.setRemoverItems();

                if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)
```

```

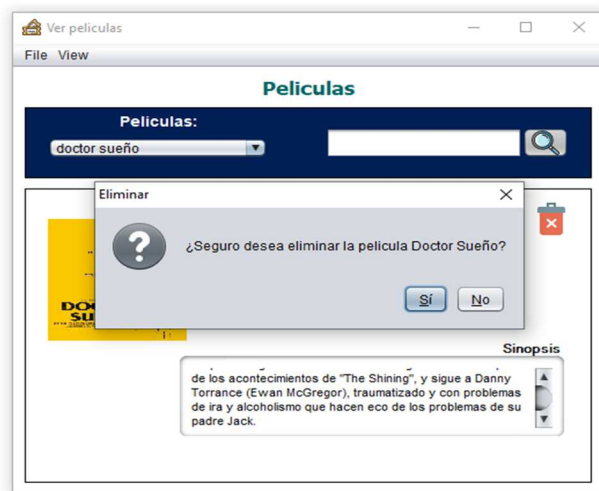
        this.setRemoverItems();

        if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)
            this.setRemoverItems();

        this.setLLenarComboPeliculas();

        JOptionPane.showMessageDialog(null,"La pelicula ha sido modificada","Pelicula
modificada", 1);
    }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido modificar la
pelicula","Modificar peliculas", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Esta pelicula ya esta registrada","Pelicula ya
rehistrada", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Modificar
pelicula", 2);
    }

```



```

private void eliminarPeliculaMouseClicked(java.awt.event.MouseEvent evt) {

    int op = 0;

    boolean band = false;

    ArrayList<Integer> lista = new ArrayList <Integer>();

    ResultSet rs = cnbd.getLlenarComboBox("SELECT f.ID_FNC FROM funciones f INNER JOIN
    peliculas p ON f.ID_PLC = p.ID_PLC WHERE p.ID_PLC = " + idPelicula.getText().trim());

    try
    {

        while(rs.next())
        {
            lista.add(rs.getInt("ID_FNC"));
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Informacion", 2);
    }

    for(int i = 0; i < lista.size(); i++)
    {
        System.out.println(lista.get(i));

        cnbd.getEliminar("DELETE FROM asientos where ID_FNC = " + lista.get(i));
    }

    op = JOptionPane.showConfirmDialog(null, "¿Seguro desea eliminar la pelicula " +
    tituloPelicula.getText() + "?", "Eliminar", JOptionPane.YES_NO_OPTION);

    if(op == JOptionPane.YES_OPTION)

```

```

{
    /*if(band)
    {*/

        if(cnbd.getEliminar("DELETE FROM funciones WHERE ID_PLC = " +
idPelicula.getText().trim()))
        {
            if(cnbd.getEliminar("DELETE FROM peliculas WHERE ID_PLC = " +
idPelicula.getText().trim()))
            {
                this.setOcultarComponentes();

                comboPeliculas.setSelectedItem("Selecciona");

                this.setRemoverItems();

                if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)
                    this.setRemoverItems();

                if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)
                    this.setRemoverItems();

                if(comboPeliculas.getItemCount() == 1 || comboPeliculas.getItemCount() > 1)
                    this.setRemoverItems();

                this.setLLenarComboPeliculas();

                JOptionPane.showMessageDialog(null,tituloPelicula.getText() + " ha sido
eliminada","Pelicula eliminada", 1);
            }
        }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido eliminar la funcion","Eliminar
funcion", 2);
    /*}

```

```

else
    JOptionPane.showMessageDialog(null,"No se ha podido eliminar los asientos","Eliminar
funcion", 2); */
}
}

```

Agregar funciones:

Verifico que los campos estén correctos y que no haya una función con la misma película, la misma sala, el mismo día y la misma hora.

```

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {

```

```

    GetSetFunciones gsf = new GetSetFunciones();

```

```

    if(comboAudio.getSelectedItemId() == "Selecciona")
        errorAudio.setText("**Seleccione una opcion valida");

```

```

    else
        errorAudio.setText(null);

```

```

    if(comboSala.getSelectedItemId() == "Selecciona")
        errorSala.setText("**Seleccione una opcion valida");

```

```

    else

```

```
errorSala.setText(null);

if(comboPeliculas.getSelectedItem() == "Selecciona")
    errorPelicula.setText("**Seleccione una opcion valida");
else
    errorPelicula.setText(null);

if(comboPeliculas.getSelectedItem() != "Selecciona" && comboAudio.getSelectedItem() !=
"Selecciona" &&
    comboSala.getSelectedItem() != "Selecciona")
{
    int plc = cnbd.consultarPeliculas(comboPeliculas.getSelectedItem().toString());

    Date dt = fechaFuncion.getDate();

    long tmp = dt.getTime();

    java.sql.Date fecha = new java.sql.Date(tmp);

    Date hr = (Date) horaFuncion.getValue();

    long tmpo = hr.getTime();

    java.sql.Time hora = new java.sql.Time(tmpo);

    String consulta = "SELECT ID_SL FROM salas WHERE Numero_SL = " +
    comboSala.getSelectedItem();

    int sl = 0;

    ResultSet rs = cnbd.consultas(consulta);

    try
```

```
{
    if(rs.next())
        sl = rs.getInt("ID_SL");
}
catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Sala de cine", 2);
}
```

```
String consultaComp = "SELECT COUNT(ID_FNC) FROM funciones WHERE ID_PLC = " +
plc + " AND ID_SL = " + sl + " AND DIA_FNC = " + fecha.toString() + " AND Hora_FNC = " +
hora.toString() + "";
```

```
if(cnbd.getBuscar(consultaComp) == 0)
{
```

```
    gsf.setId(Integer.parseInt(idFuncion.getText()));
    gsf.setDia(fecha.toString());
    gsf.setHora(hora.toString());
    gsf.setPelicula(plc);
    gsf.setSala(sl);
    gsf.setAudio(comboAudio.getSelectedIndex());
```

```
String consultaCp = "SELECT Capacidad_SL FROM salas WHERE ID_SL = " + sl;
int capacidad = 0;
```

```
ResultSet rs2 = cnbd.getConsultas(consultaCp);
```

```
try
{
    if(rs2.next())
        capacidad = rs2.getInt("Capacidad_SL");
}
```

```

catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Consultar asientos", 2);
}

if(cnbd.getRegistrarFuncion(gsf) & cnbd.getRegistrarAsientos(gsf, capacidad))
{
    JOptionPane.showMessageDialog(null,"La funcion ha sido guardada","Funcion
agregada", 1);

    setLimpiarFormulario();

    idFuncion.setText(String.valueOf(cnbd.getGenerarId("SELECT MAX(ID_FNC) FROM
funciones"))));
}
else

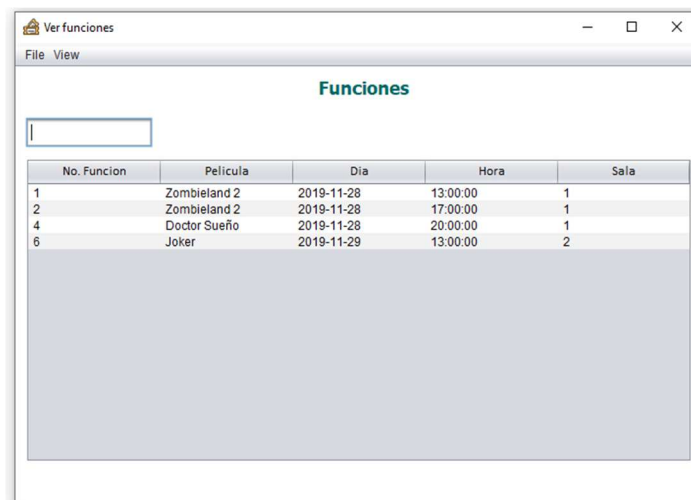
    JOptionPane.showMessageDialog(null,"No se han podido guardar los
datos","Registrar funcion", 2);
}
else

    JOptionPane.showMessageDialog(null,"La funcion ya ha sido registrada","Funcion ya
registrada", 2);
}
else

    JOptionPane.showMessageDialog(null,"Los campos seleccionados son incorrectos","Datos
incorrectos", 2);
}

```

Ver funciones:



No. Funcion	Película	Dia	Hora	Sala
1	Zombieland 2	2019-11-28	13:00:00	1
2	Zombieland 2	2019-11-28	17:00:00	1
4	Doctor Sueño	2019-11-28	20:00:00	1
6	Joker	2019-11-29	13:00:00	2


```

public void setLlenarTablaFunciones(String consulta)
{
    try
    {
        ResultSet rs = cnbd.getLlenarComboBox(consulta);

        while(rs.next())
        {
            dtm.addRow(new Object[] {rs.getInt("ID_FNC"), rs.getString("Nombre_PLC"),
rs.getString("Dia_FNC"), rs.getString("Hora_FNC"), rs.getInt("Numero_SL")});
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: "+e,"!! ERROR !!", 2);
    }
}

```



Una vez que selecciono un campo me da aparece la información que seleccione y me da la posibilidad de modificar, cancelar para que dejen de aparecer arriba la información del campo seleccionado y eliminar la función.

```

private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {
    int plc = cnbd.getConsultarPeliculas(comboPeliculas.getSelectedItem().toString());
}

```

```

Date dt = fechaFuncion.getDate();

long tmp = dt.getTime();

java.sql.Date fecha = new java.sql.Date(tmp);

Date hr = (Date) horaFuncion.getValue();

long tmpo = hr.getTime();

java.sql.Time hora = new java.sql.Time(tmpo);

String consulta = "SELECT ID_SL FROM salas WHERE Numero_SL = " +
comboSala.getSelectedItemId();

int sl = 0;

ResultSet rs = cnbd.getConsultas(consulta);

try
{
    if(rs.next())
        sl = rs.getInt("ID_SL");
}
catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Sala de cine", 2);
}

String consultaComp = "SELECT COUNT(ID_FNC) FROM funciones WHERE ID_PLC = " +
plc + " AND ID_SL = " + sl + " AND DIA_FNC = " + fecha.toString() + " AND Hora_FNC = " +
hora.toString() + " AND ID_FNC != " + textold.getText().trim();

```

```

if(cnbd.getBuscar(consultaComp) == 0)
{
    GetSetFunciones gsf = new GetSetFunciones();

    gsf.setId(Integer.parseInt(textold.getText().trim()));
    gsf.setDia(fechar.toString());
    gsf.setHora(hora.toString());
    gsf.setPelicula(plc);
    gsf.setSala(sl);

    if(cnbd.getModificarFunciones(gsf))
    {
        JOptionPane.showMessageDialog(null,"La funcion ha sido modificada","Funcion
modificada", 1);

        /*int a = dtm.getRowCount()-1;

        for(int i = 0; i < a; i++)
            dtm.removeRow(i);*/

        //this.setLLenarTablaFunciones("SELECT f.*, s.Numero_SL, p.Nombre_PLC FROM
funciones f INNER JOIN peliculas p ON f.ID_PLC = p.ID_PLC INNER JOIN salas s on f.ID_SL =
s.ID_SL");

        this.setOcultarComponentes();
    }
    else
        JOptionPane.showMessageDialog(null,"No se han podido modificar los datos","Modificar
funcion", 2);
}
else
    JOptionPane.showMessageDialog(null,"La funcion ya ha sido registrada","Funcion ya
registrada", 2);

```

```

    }

    private void botonEliminarMouseClicked(java.awt.event.MouseEvent evt) {

        int op = 0;

        op = JOptionPane.showConfirmDialog(null, "¿Seguro desea eliminar esta funcion?",
"Eliminar", JOptionPane.YES_NO_OPTION);

        if(op == JOptionPane.YES_OPTION)
        {
            if(cnbd.getEliminar("DELETE FROM asientos WHERE ID_FNC = " +
textold.getText().trim()))
            {
                if(cnbd.getEliminar("DELETE FROM funciones WHERE ID_FNC = " +
textold.getText().trim()))
                {
                    JOptionPane.showMessageDialog(null,"La funcion ha sido eliminada","Funcion
eliminada", 1);

                    this.setOcultarComponentes();
                }
            }
            else
                JOptionPane.showMessageDialog(null,"No se ha podido eliminar la funcion","Eliminar
funcion", 2);
        }
        else
            JOptionPane.showMessageDialog(null,"No se ha podido eliminar los asientos","Eliminar
funcion", 2);
        }
    }
}

```

Agregar precios de boletos:



Registrar precios

File View

Registro de precios

Niños: \$

Adultos: \$

3era Edad: \$

Guardar

Verifico que no estén ingresados los precios y si lo están, no permito volverlos a agregar.

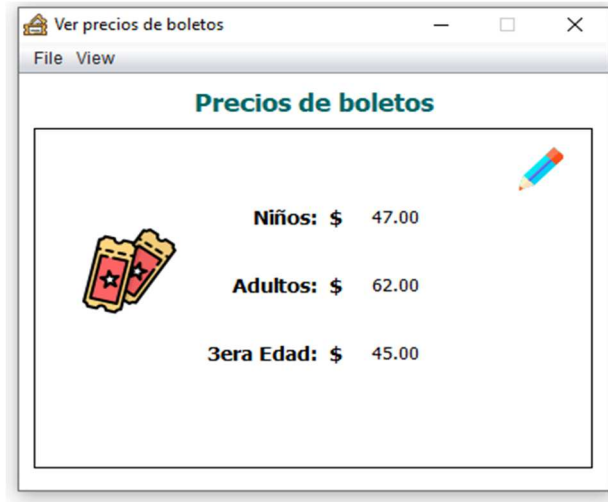
```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {  
    GetSetPrecioBoleto gspb = new GetSetPrecioBoleto();  
  
    if (precioNino.getText().length() == 0)  
        errorNino.setText("**El precio es incorrecto");  
    else  
        errorNino.setText(null);  
  
    if (precioAdulto.getText().length() == 0)  
        errorAdulto.setText("**El precio es incorrecto");  
    else  
        errorAdulto.setText(null);  
  
    if (precioAnciano.getText().length() == 0)  
        errorAnciano.setText("**El precio es incorrecto");  
    else  
        errorAnciano.setText(null);  
}
```

```
        if(!precioNino.getText().equals("") && !precioAdulto.getText().equals("") &&
!precioAnciano.getText().equals(""))
        {
            String consulta = "SELECT COUNT(ID_TBTLT) FROM tipo_boleto WHERE ID_TBTLT = 1";

            if(cnbd.getBuscar(consulta) == 0)
            {
                gspb.setPrecioNino(Float.parseFloat(precioNino.getText()));
                gspb.setPrecioAdulto(Float.parseFloat(precioAdulto.getText()));
                gspb.setPrecioAnciano(Float.parseFloat(precioAnciano.getText()));

                if(cnbd.getRegistrarPrecioBoleto(gspb))
                {
                    JOptionPane.showMessageDialog(null,"Los precios han sido guardado","Precios
agregados", 1);
                    setLimpiarFormulario();
                }
                else
                    JOptionPane.showMessageDialog(null,"No se han podido guardar los
datos","Registrar precios", 2);
            }
            else
                JOptionPane.showMessageDialog(null,"Los precios ya han sido registrados","Precios ya
registrados", 2);
        }
        else
            JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Registrar
precios", 2);
    }
}
```

Ver precios de boletos:



```
public void setPreciosBoletos()
{
    ArrayList<String> lista = new ArrayList <String>();

    ResultSet rs = cnbd.getLlenarComboBox("SELECT * FROM TIPO_BOLETO");

    try
    {
        while(rs.next())
        {
            lista.add(rs.getString("Precio_TBLT"));
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Precios", 2);
    }

    precioNino.setText(lista.get(0));
    precioAdulto.setText(lista.get(1));
    precioAnciano.setText(lista.get(2));
}
```



```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

    if(!editNino.getText().equals("") && !editAdulto.getText().equals("") &&
    !editAnciano.getText().equals(""))
    {
        GetSetPrecioBoleto gspb = new GetSetPrecioBoleto();

        gspb.setPrecioNino(Float.parseFloat(editNino.getText()));
        gspb.setPrecioAdulto(Float.parseFloat(editAdulto.getText()));
        gspb.setPrecioAnciano(Float.parseFloat(editAnciano.getText()));

        if(cnbd.getModificarPrecios(gspb))
        {
            this.setOcultarComponentes();

            this.setPreciosBoletos();

            JOptionPane.showMessageDialog(null,"Los precios han sido modificados","Precios
modificados", 1);
        }
        else
            JOptionPane.showMessageDialog(null,"No se han podido modificar los datos","Modificar
precios", 2);
    }
}
```



```

    }
    else
        JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Modificar
precios", 2);
    }

```

Agregar salas:

Verifico que no exista un numero de sala igual al que se quiere agregar.

```

private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

```

```

    GetSetSala gss = new GetSetSala();

```

```

    if(comboTipoSala.getSelectedItem() == "Selecciona")

```

```

        errorTipo.setText("**Seleccione una opcion valida");

```

```

    else

```

```

        errorTipo.setText(null);

```

```

    if(comboAsientos.getSelectedItem() == "Selecciona")

```

```

        errorAsientos.setText("**Seleccione una opcion valida");

```

```

    else

```

```

        errorAsientos.setText(null);

```

```

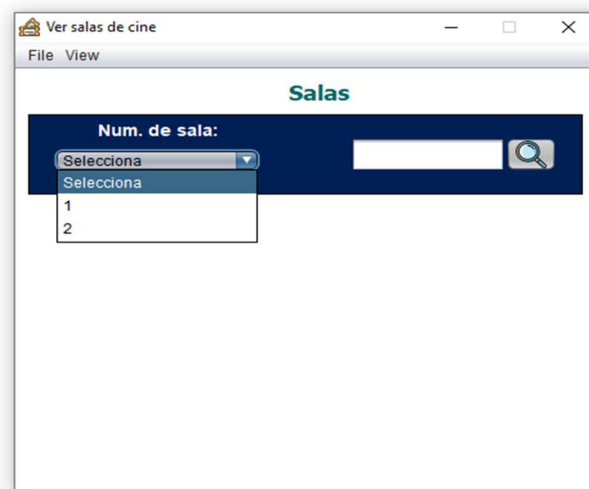
        if(comboTipoSala.getSelectedItemId() != "Selecciona" && comboAsientos.getSelectedItemId() !=
"Selecciona")
        {
            String consulta = "SELECT COUNT(ID_SL) FROM salas WHERE Numero_SL = " +
numeroSala.getValue();

            if(cnbd.getBuscar(consulta) == 0)
            {
                gss.setId(Integer.parseInt(idSala.getText()));
                gss.setNumero(numeroSala.getValue());
                gss.setCapacidad(Integer.parseInt(comboAsientos.getSelectedItemId().toString()));
                gss.setTipo(comboTipoSala.getSelectedIndex());

                if(cnbd.getRegistrarSala(gss))
                {
                    JOptionPane.showMessageDialog(null,"La sala ha sido guardada","Sala agregada", 1);
                    setLimpiarFormulario();
                    idSala.setText(String.valueOf(cnbd.getGenerarId("SELECT MAX(ID_SL) FROM
salas"))));
                }
                else
                {
                    JOptionPane.showMessageDialog(null,"No se han podido guardar los
datos","Registrar sala", 2);
                }
                else
                {
                    JOptionPane.showMessageDialog(null,"El numero de sala ya esta registrada","Sala ya
registrada", 2);
                }
                else
                {
                    JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Registrar
sala", 2);
                }
            }
        }
    }

```

Ver salas -> Seleccionar sala:



```
public void setLlenarComboSalas()
{
    ArrayList<String> lista = new ArrayList <String>();
    ResultSet rs = cnbd.getLlenarComboBox("SELECT * FROM salas");

    try
    {
        while(rs.next())
        {
            lista.add(rs.getString("Numero_SL").toLowerCase());
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Llenar combo box", 2);
    }

    for(int i = 0; i < lista.size(); i++)
        comboSalas.addItem(lista.get(i));
}
```



```

public void setInfoSala(int numSI)
{
    ResultSet rs = cnbd.getConsultas("SELECT s.*, t.Tipo_TSL FROM salas s INNER JOIN
    tipo_sala t ON s.ID_TSL = t.ID_TSL WHERE Numero_SL = " + numSI);

    try
    {
        if(rs.next())
        {
            idSala.setText(String.valueOf(rs.getInt("ID_SL")));
            numSala.setText(String.valueOf(rs.getInt("Numero_SL")));
            capacidadSala.setText(String.valueOf(rs.getInt("Capacidad_SL")));
            tipoSala.setText(rs.getString("Tipo_TSL"));
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Informacion", 2);
    }
}

```



```
private void botonGuardarMouseClicked(java.awt.event.MouseEvent evt) {

    if(!editNumSala.getText().equals(""))
    {
        String consulta = "SELECT COUNT(ID_SL) FROM salas WHERE Numero_SL = " +
        editNumSala.getText() + " AND ID_SL != " + idSala.getText();

        if(cnbd.getBuscar(consulta) == 0)
        {
            GetSetSala gss = new GetSetSala();

            int tSl = editTipoSala.getSelectedIndex() + 1;

            gss.setId(Integer.parseInt(idSala.getText()));
            gss.setNumero(Integer.parseInt(editNumSala.getText()));
            gss.setTipo(tSl);

            if(cnbd.getModificarSalas(gss))
            {
                this.setOcultarComponentes();
            }
        }
    }
}
```

```
        comboSalas.setSelectedItem("Selecciona");

        this.setRemoverItems();

        if(comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

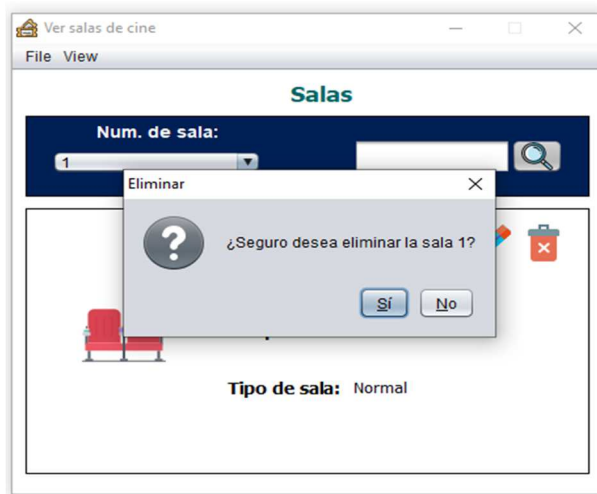
        if(comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

        if(comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

        this.setLLenarComboSalas();

        JOptionPane.showMessageDialog(null,"La sala ha sido modificada","Sala modificada",
1);
    }
    else
        JOptionPane.showMessageDialog(null,"No se han podido modificar los
datos","Modificar salas", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Esta sala ya esta registrada","Sala ya
registrada", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"Los datos ingresados son incorrectos","Modificar
sala", 2);

    }
```



```
private void eliminarSalasMouseClicked(java.awt.event.MouseEvent evt) {
    int op = 0;

    op = JOptionPane.showConfirmDialog(null, "¿Seguro desea eliminar la sala " +
numSala.getText() + "?", "Eliminar", JOptionPane.YES_NO_OPTION);

    if(op == JOptionPane.YES_OPTION)
    {
        if(cnbd.getEliminar("DELETE FROM asientos WHERE ID_SL = " + idSala.getText().trim()))
        {
            if(cnbd.getEliminar("DELETE FROM funciones WHERE ID_SL = " +
idSala.getText().trim()))
            {
                if(cnbd.getEliminar("DELETE FROM salas WHERE ID_SL = " +
idSala.getText().trim()))
                {
                    this.setOcultarComponentes();

                    comboSalas.setSelectedItem("Selecciona");

                    this.setRemoverItems();
                }
            }
        }
    }
}
```

```

        if (comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

        if (comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

        if (comboSalas.getItemCount() == 1 || comboSalas.getItemCount() > 1)
            this.setRemoverItems();

        this.setLLenarComboSalas();

        JOptionPane.showMessageDialog(null,"La sala " + numSala.getText() + " ha sido
        eliminada","Sala eliminada", 1);
    }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido eliminar la sala","Eliminar
        sala", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido eliminar la funcion","Eliminar
        sala", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"No se han podido eliminar los asientos","Eliminar
        sala", 2);
    }
}

```


Inicio de venta de boletos:



File -> Salir -> Cerrar sesión:

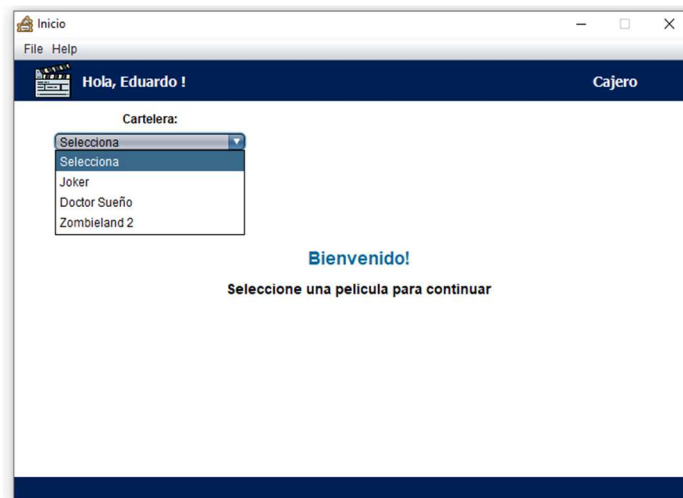


```
private void menuCerrarSesionActionPerformed(java.awt.event.ActionEvent evt) {  
    VentanaInicioDeSesion vis = new VentanaInicioDeSesion();  
  
    this.dispose();  
    vis.setVisible(true);  
}
```

Help -> Manual de usuario:



Cartelera -> Selecciona película:



```
public void setLlenarComboPeliculas()
{
    ArrayList<String> lista = new ArrayList <String>();
    ResultSet rs = cnbd.getLlenarComboBox("SELECT * FROM peliculas");

    comboPeliculas.addItem("Selecciona");

    try
```

```

{
    while(rs.next())
    {
        lista.add(rs.getString("Nombre_PLC"));
    }
}

catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Llenar combo box", 2);
}

for(int i = 0; i < lista.size(); i++)
    comboPeliculas.addItem(lista.get(i));
}

```

Cartelera -> Información de la película:



```

public void setPoster(String consulta)
{
    ResultSet rs = cnbd.getConsultas(consulta);

    BufferedImage bi = null;

```

```

byte [] image = null;

try
{
    while(rs.next())
    {
        image = rs.getBytes("Poster_PLC");

        InputStream bs = rs.getBinaryStream(1);

        try
        {
            bi = ImageIO.read(bs);

            CargarImagen imagen = new CargarImagen(panelPoster.getHeight(),
panelPoster.getWidth(), bi);

            panelPoster.add(imagen);
            panelPoster.repaint();
        }
        catch (IOException e)
        {
            JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Poster", 2);
        }
    }
}
catch (SQLException e)
{
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Poster", 2);
}
}

```

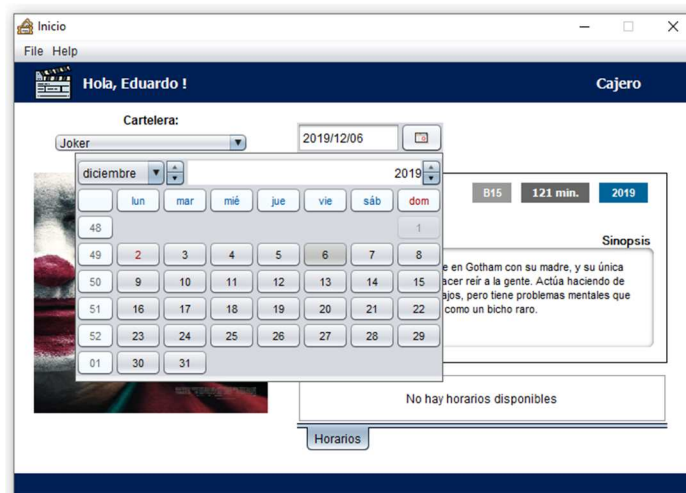
```

public void setInformacionPeliculas(int id)
{
    ResultSet rs = cnbd.getConsultas("SELECT p.*, c.Tipo_CFC FROM peliculas p INNER JOIN
clasificacion c ON p.ID_CFC = c.ID_CFC where ID_PLC = " + id);

    try
    {
        if(rs.next())
        {
            tituloPelicula.setText(rs.getString("Nombre_PLC"));
            clasificacionPelicula.setText(rs.getString("Tipo_CFC"));
            anoPelicula.setText(String.valueOf(rs.getInt("Ano_PLC")));
            duracionPelicula.setText(String.valueOf(rs.getInt("Duracion_PLC")) + " min.");
            sinopsisPelicula.setText(rs.getString("Sinopsis_PLC"));
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Informacion", 2);
    }
}

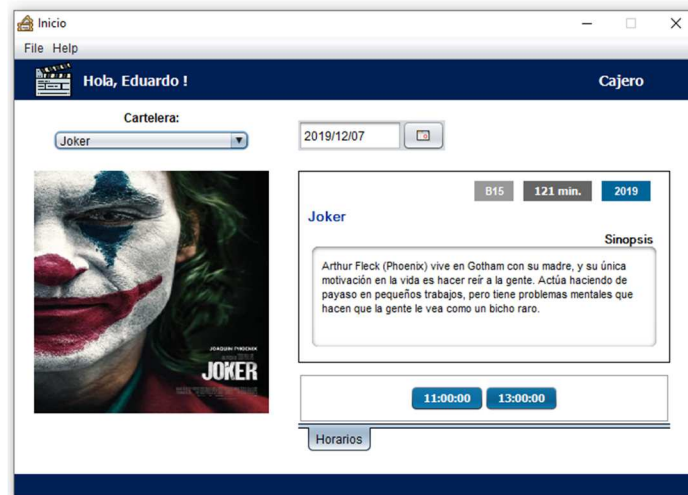
```

Cartelera -> Selecciona fecha:



```
private void fechaFuncionPropertyChange(java.beans.PropertyChangeEvent evt) {  
    fechaFuncion.getDateEditor().addPropertyChangeListener(new PropertyChangeListener()  
    {  
        int band = 3;  
        public void propertyChange(PropertyChangeEvent e)  
        {  
            if(band == 3)  
            {  
                if(fechaFuncion.getDate() != null)  
                {  
                    Date dt = fechaFuncion.getDate();  
                    long tmp = dt.getTime();  
                    java.sql.Date fecha = new java.sql.Date(tmp);  
  
                    fch = fecha.toString();  
  
                    setHorarios();  
                }  
                band = 1;  
            }  
            else if(band == 1)  
                band = 2;  
            else if(band == 2)  
                band = 0;  
        }  
    }  
    );  
}
```

Cartelera -> Fecha -> Selecciona horario:



```
public void setHorarios()
{
    int idPI = cnbd.getConsultarPeliculas(comboPeliculas.getSelectedItem().toString());
    boolean bnd = false;

    panelHorarios.removeAll();
    panelHorarios.repaint();

    ArrayList lista = new ArrayList <String>();

    try
    {
        String consultaHrFn = "SELECT Hora_FNC FROM funciones WHERE Dia_FNC = " + fch +
        "" AND ID_PLC = " + idPI;

        ResultSet rs2 = cnbd.getLlenarComboBox(consultaHrFn);

        while(rs2.next())
        {
            lista.add(rs2.getString("Hora_FNC"));
            bnd = true;
        }
    }
}
```

```
    }  
}  
catch (SQLException e)  
{  
    bnd = false;  
}  
  
if(bnd)  
{  
    Font fuente = new Font("Tahoma", Font.BOLD, 11);  
  
    for(int i = 0; i < lista.size(); i++)  
    {  
        JButton btn = new JButton((String) lista.get(i));  
  
        btn.setCursor(new Cursor(HAND_CURSOR));  
        btn.setFont(fuente);  
        btn.setBackground(new Color(0,73,112));  
        btn.setForeground(Color.WHITE);  
  
        panelHorarios.add(btn);  
  
        btn.addActionListener(new ActionListener()  
        {  
  
            @Override  
            public void actionPerformed(ActionEvent ev)  
            {  
  
                int idFn = 0;
```



```
String consultaFn = "SELECT ID_FNC FROM funciones WHERE Dia_FNC = " + fch  
+ "" + "AND Hora_FNC = " + btn.getText().toString() + "" + "AND ID_PLC = " + idPI;
```

```
ResultSet rs3 = cnbd.getConsultas(consultaFn);
```

```
try  
{  
    if(rs3.next())  
        idFn = rs3.getInt("ID_FNC");  
}  
catch (SQLException e)  
{  
    JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Consultar funcion",  
2);  
}
```

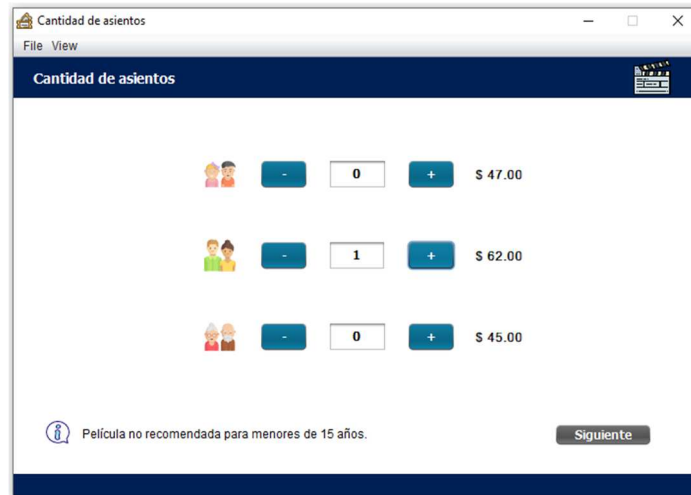
```
VentanaCantidad vc = new VentanaCantidad(gse, idPI, idFn);
```

```
dispose();  
vc.setVisible(true);  
}  
});  
}
```

```
panelHorarios.updateUI();  
}  
else  
{  
    JLabel lb = new JLabel("No hay horarios disponibles");  
  
    panelHorarios.add(lb);  
    panelHorarios.updateUI();  
}
```

```
}
```

Selecciona la cantidad de asientos:



```
public void setPrecioBoletos()
{
    ResultSet rs = cnbd.getLlenarComboBox("SELECT Precio_TBLT FROM TIPO_BOLETO");

    try
    {
        while(rs.next())
        {
            lista.add(rs.getString("Precio_TBLT"));
        }
    }
    catch (SQLException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Precios", 2);
    }

    precioNino.setText("$ " + lista.get(0));
    precioAdulto.setText("$ " + lista.get(1));
}
```

```

precioAnciano.setText("$ " + lista.get(2));
}

```

Selecciona los asientos a comprar:



```

public void setBotones()
{
    int cnt = 1;

    Font fuente = new Font("Arial", Font.BOLD, 12);

    botonSel = new JToggleButton[filas][columnas];

    for(int i = 0; i < filas; i++)
    {
        for(int j = 0; j < columnas; j++)
        {
            botonSel[i][j] = new JToggleButton();

            botonSel[i][j].setText("" + cnt);

            botonSel[i][j].setFont(fuente);

```

```

        botonSel[i][j].setBackground(new Color(31, 222, 101));

        botonSel[i][j].setCursor(new Cursor(HAND_CURSOR));

        AccionBotones acb = new AccionBotones();

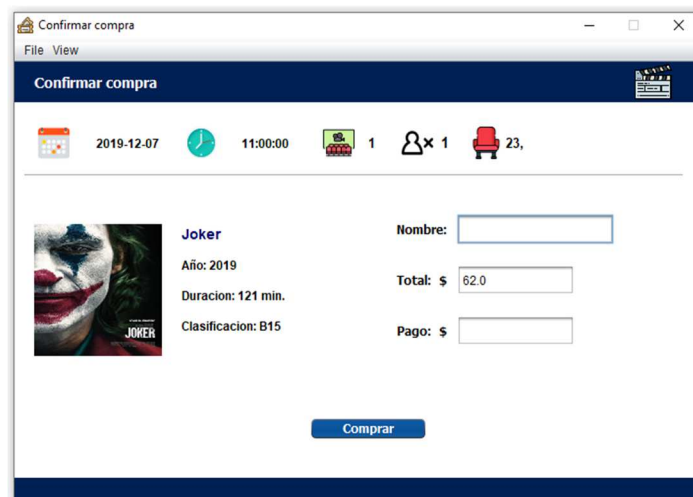
        botonSel[i][j].addActionListener(acb);

        panelAsientos.add(botonSel[i][j]);

        cnt ++;
    }
}
}

```

Resumen de compra:



```

private void botonComprarMouseClicked(java.awt.event.MouseEvent evt) {
    if(!textNombre.getText().equals("") && !textPago.getText().equals(""))
    {
        float pago = Float.parseFloat(textPago.getText());
        float total = Float.parseFloat(precioTotal.getText());
    }
}

```

```
int idBl = cnbd.getGenerarId("SELECT MAX(ID_BLT) FROM boletos");
int idCl = cnbd.getGenerarId("SELECT MAX(ID_CL) FROM clientes");

java.util.Date fecha = new java.util.Date();
DateFormat formato = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

if(pago >= total)
{
    GetSetBoletos gsb = new GetSetBoletos();

    gsb.setIdBl(idBl);
    gsb.setIdCl(idCl);
    gsb.setNombrePelicula(tituloPelicula.getText());
    gsb.setNombreCl(textNombre.getText().trim());
    gsb.setVendedor(gse.getNombre() + " " + gse.getApellido());
    gsb.setFechaCmp(formato.format(fecha));
    gsb.setNumSala(Integer.parseInt(numeroSala.getText().trim()));
    gsb.setCantBl(Integer.parseInt(cantidadPersona.getText().trim()));
    gsb.setAsientos(asientos.getText().trim());
    gsb.setDia(diaFuncion.getText().trim());
    gsb.setHora(horaFuncion.getText().trim());
    gsb.setTotalBl(Float.parseFloat(precioTotal.getText().trim()));

    if(cnbd.getRegistrarClientes(gsb))
    {
        if(cnbd.getRegistrarBoletos(gsb))
        {
            GetSetAsientos gsa = new GetSetAsientos();

            boolean band = false;

            for(int i = 0; i < lista.size(); i++)
```

```

        {
            gsa.setIdCl(idCl);
            gsa.setIdFuncion(idFn);
            gsa.setNumAsiento(Integer.parseInt(lista.get(i).toString()));

            if(cnbd.getModificarAsientos(gsa))
                band = true;
            else
                band = false;
        }

        MetodosCine mc = new MetodosCine();

        float cambio = mc.getObtenerCambio(pago, total);

        if(band)
        {
            VentanaBoletoCompra vbc = new VentanaBoletoCompra(gse, cambio,
gsb.getIdBl());

            JOptionPane.showMessageDialog(null,"Se ha realizado la compra","Boleto
comprado", 1);

            this.dispose();

            vbc.setVisible(true);
        }
        else
            JOptionPane.showMessageDialog(null,"No se ha podido guardar el asiento","Error
en la compra", 2);
    }
    else
        JOptionPane.showMessageDialog(null,"No se ha podido realizar la compra","Error
en la compra", 2);

```

```

    }

    else

        JOptionPane.showMessageDialog(null,"No se ha podido guardar el cliente","Error en la compra", 2);

    }

    else

        JOptionPane.showMessageDialog(null,"La cantidad pagada es incorrecta","Pago incorrecto", 0);

    }

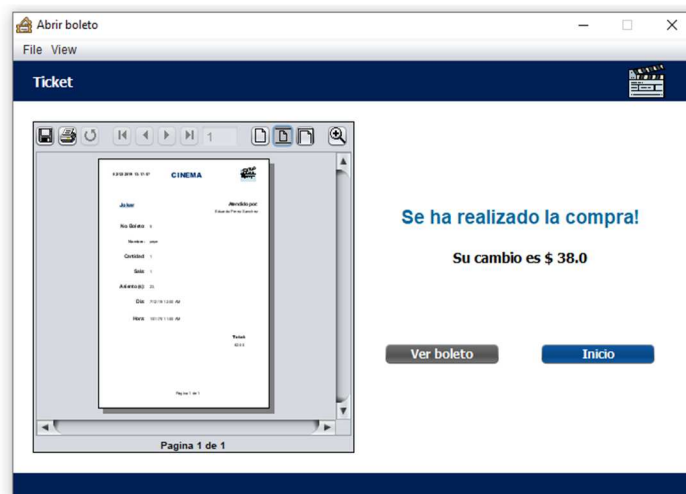
    else

        JOptionPane.showMessageDialog(null,"Los campos ingresados son incorrectos","Campos incorrectos", 2);

    }

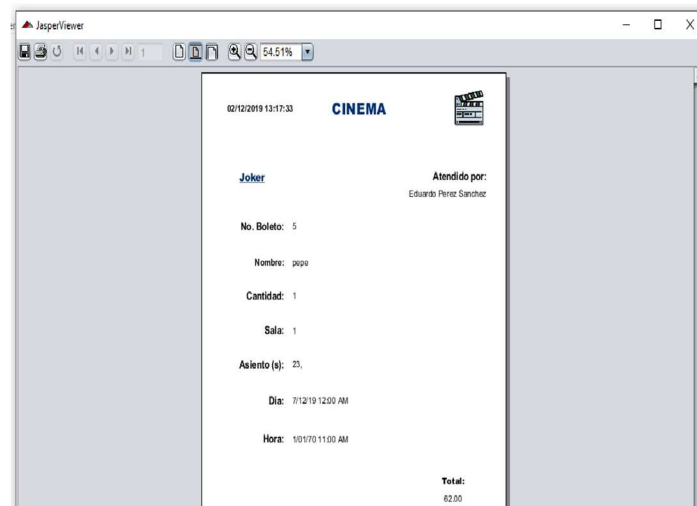
```

Cambio e impresión del boleto:



```
cambioCompra.setText("Su cambio es $ " + cambio);
```

Ver boleto:



```
private void botonGenerarBoletoMouseClicked(java.awt.event.MouseEvent evt) {  
    try  
    {  
        ConexionBD cnbd = new ConexionBD();  
  
        JasperReport reporte = null;  
  
        String path = "src\\Reportes\\CompraBoleto.jasper";  
  
        Map parametro = new HashMap();  
  
        parametro.put("idBoleto", idBI);  
  
        reporte = (JasperReport) JRLoader.loadObjectFromFile(path);  
  
        JasperPrint jPrint = JasperFillManager.fillReport(path, parametro, cnbd.getConectar());  
  
        JasperViewer view = new JasperViewer(jPrint, false);  
  
        view.setDefaultCloseOperation(DISPOSE_ON_CLOSE);  
    }  
}
```



```
        view.setVisible(true);
    }
    catch (JRException e)
    {
        JOptionPane.showMessageDialog(null,"Existe un error: " + e,"Reporte generado", 2);
    }
}
```