Código del proyecto "Cartelera"

Inicio.java

```
package cartelera;

import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;

import javax.sound.sampled.Line;
import javax.tv.xlet.*;

import org.havi.ui.*;
import org.xml.sax.helpers.LocatorImpl;
import org.dvb.dsmcc.AsynchronousLoadingEventListener;
import org.dvb.dsmcc.DSMCCException;
import org.dvb.dsmcc.DSMCCObject;
import org.dvb.dsmcc.InvalidPathNameException;
import org.dvb.dsmcc.MPEGDeliveryException;
import org.dvb.dsmcc.ServiceDomain;
import org.dvb.dsmcc.ServiceXFRException;
import org.dvb.ui.*;
import org.davic.media.*;
import org.davic.net.InvalidLocatorException;
import org.davic.net.Locator;
import org.davic.net.dvb.DvbLocator;

import excepcion.SonlosMismosException;

import javax.tv.locator.LocatorFactory;
import javax.tv.locator.MalformedLocatorException;
import javax.tv.service.Service;
import javax.tv.service.selection.*;
import java.awt.Scrollbar;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InterruptedIOException;
import java.io.Reader;
import java.net.URL;
import java.util.LinkedList;




// La clase inicial debe implementar la interface Xlet
public class Inicio extends HContainer implements KeyListener, Xlet,
FocusListener {//Runnable

      private static XletContext context;
```

```
	private HScene scene;

	//Cada contenedor representa una pantalla
	private HContainer contPP;//Pantalla Inicial
	private HContainer contCines;
	private HContainer contTeatro;
	private HContainer contCineSalas;
	private HContainer contCineResumen;


	private Image cine;
	private Image teatro;
	private Image botonAtras;
	private Image botonArriba;
	private Image botonAbajo;
	private Image botonAzul;
	private Image botonVerde;
	private Image horario;
	private Image resumen;
	private Image botonRojo;

	private HTextButton textoCine;
	private HTextButton textoHoras;
	private HTextButton textoResumen;
	private HTextButton textoTeatro;
	private HTextButton botonInvisible;
	private HTextButton textoBotonVerde;

	private HGraphicButton botones[];

	//Son los titulos de cada pantalla
	private HStaticText textoCartelera;
	private HStaticText textoAtras;
	private HStaticText textoResumenes;
	private HStaticText textoHorario;


	private LinkedList<Objeto> lasObras;
	private LinkedList<Objeto> lasSalas;
	private LinkedList<Objeto> lasPeli;

	//Cuadro de texto para cargar la info a mostrar, dos por pantalla
	private HStaticText obraTextoUno;
	private HStaticText obraTextoDos;
	private HStaticText peliTextoUno;
	private HStaticText peliTextoDos;

	//paths
	private String pathBase="";
	private String pathCinetxt=pathBase+"info/Peliculas.txt";
	private String pathSalatxt=pathBase+"info/Salas.txt";
	private String pathTeatrotxt=pathBase+"info/Teatros.txt";
	private String pathRojo=pathBase+"images/redspot.png";
	private String pathVerde=pathBase+"images/greenspot.png";
	private String pathAmarillo=pathBase+"images/yellowspot.png";
```

```
	private String pathAzul=pathBase+"images/bluespot.png";
	private String pathFlechaArriba=pathBase+"images/uparrow.png";
	private String pathFlechaAbajo=pathBase+"images/downarrow.png";
	private String pathCine=pathBase+"images/cine.jpg";
	private String pathTeatro=pathBase+"images/teatro.jpeg";
	private String pathResumen=pathBase+"images/resumen.png";
	private String pathHorario=pathBase+"images/horas.jpg";



	private boolean esCarrusel=false;

	//Para el carrusel
	private DSMCCObject peliculasObj;
	private DSMCCObject salasObj;
	private DSMCCObject teatroObj;
	private DSMCCObject redSpot;
	private DSMCCObject greenSpot;
	private DSMCCObject yellowSpot;
	private DSMCCObject blueSpot;
	private DSMCCObject arrowUp;
	private DSMCCObject arrowDown;
	private DSMCCObject cinesImgObj;
	private DSMCCObject teatrosImgObj;
	private DSMCCObject horarioImgObj;
	private DSMCCObject resumenImgObj;

	//Para el locator de acceso al carrusel
	private org.davic.net.dvb.DvbLocator locator;
	private ServiceContextFactory scf;
	private ServiceContext sc;



	public Inicio() {
		//Como se recomienda en MHP el constructor de esta clase no
hace nada, todo se inicializa en el initXlel()
		super();
	}

	// Inicializando la aplicación debe pasarle como parametro un
contexto para que la misma inicie
	//Pide recursos
	public void initXlet(XletContext ctx) throws XletStateChangeException
{

		setName("Inicio");
		this.context = ctx;

		//Se obtiene el screen
		HScreen screen = HScreen.getDefaultHScreen();

		HGraphicsDevice device = screen.getDefaultHGraphicsDevice();

		//Se crea un nuevo template para setear las preferencias
```

```
            HGraphicsConfigTemplate template = new
HGraphicsConfigTemplate();
            template.setPreference(template.IMAGE_SCALING_SUPPORT,
template.PREFERRED);

      template.setPreference(template.ZERO_GRAPHICS_IMPACT,template.REQUIRE
D);

            //Se obtiene la configuracion posible a las preferencias
estipuladas
            HGraphicsConfiguration configuration =
device.getBestConfiguration(template);

            //Se verifica que lo obtenido no sea null
            if (configuration != null){
                  try{
                        device .setGraphicsConfiguration(configuration);
                  }
                  catch(Exception e){
                        System.out.println("Exception!!!!!!!");
                  }
            }

            ////////////

            //Configuro la scena basica que contendra a todos los
componentes y containers, por cada xlet se debe
            //tener un solo Hscene
            HSceneFactory factory = HSceneFactory.getInstance();
            HSceneTemplate hst = new HSceneTemplate();
            hst.setPreference(HSceneTemplate.SCENE_SCREEN_DIMENSION, new
org.havi.ui.HScreenDimension(1, 1), HSceneTemplate.REQUIRED);
            hst.setPreference(HSceneTemplate.SCENE_SCREEN_LOCATION, new
org.havi.ui.HScreenPoint(0, 0), HSceneTemplate.REQUIRED);
            scene = factory.getBestScene(hst);
            scene.setBounds(0,0,900,900);
            scene.setLayout(null);
            scene.setBackgroundMode(HScene.BACKGROUND_FILL);
            scene.setVisible(true);
            scene.add(this);
            this.setSize(scene.getSize());

            //Habilita la deteccion de eventos provocados por le control
remoto
            scene.addKeyListener((KeyListener)this);




      }

      //El receptor comprueba los recursos disponibles y las prioridadesde
de las aplicaciones intereacivas
```

```
      //que se encuentran disponibles y las ejecuta mediante el metodo
startXlet().

      public void startXlet() throws XletStateChangeException {

            validate();
            if(esCarrusel){
                  cargarObjetos();
            }
            scene.setVisible(true);
            scene.requestFocus();//Permite al usuario seleccionar entre los
botones existentes en la pantalla
            cargarPPPage();



      }


      //Si el recerptor cree conveniente pasa pone en pausa la aplicacion
mediante el metodo pauseXlet()
      public void pauseXlet() {
            scene.setVisible(false);
      }


      //Cuando la aplicacion interactiva ha finalizado el receptor debe
destruirla y lo hace invoacando el metodo destroyXlet(true)
      public void destroyXlet(boolean b) {

            if (scene != null) {
                  removeKeyListener(this);
                  scene.remove(this);
                  scene.setVisible(false);
                  HSceneFactory.getInstance().dispose(scene);
                  scene = null;
            }
            context.notifyDestroyed();
      }


//Se pinta la escena, queda fija para todas las pantallas
      public void paint(Graphics g) {
            super.paint(g);

            if(esCarrusel){
                  botonRojo =
Toolkit.getDefaultToolkit().getImage(redSpot.getPath());
            }
            else{
                  botonRojo =
Toolkit.getDefaultToolkit().getImage(pathRojo);
            }
            g.drawImage(botonRojo,20,520,40,40,this);
            g.setColor(Color.BLACK);
```

```
            g.setFont(new Font("Verdana",Font.BOLD,30));
            g.drawString("Salir",70,550);

      }


      // Captura los eventos que son provocados por el control
      public void keyPressed(KeyEvent key) {
            switch(key.getKeyCode()) {
            case 403://boton rojo
                  destroyXlet(true);
                  break;
            case 404://boton verde
                  if (this.textoCine.isSelected()){

                        if (contCines == null){
                              contCines = new HContainer(10, 10, 900, 900);
                              cargarCinesPage();
                        }
                        else {
                              scene.remove(contPP);
                              scene.add(contCines);
                              textoHoras.requestFocus();
                        }
                  } else if (this.textoTeatro.isSelected()){
                        if (contTeatro == null){
                              contTeatro = new HContainer(10, 10, 900, 900);
                              cargarTeatrosPage();

                        }
                        else{
                              scene.remove(contPP);
                              scene.add(contTeatro);

                        }
                  }else if (this.contCines != null && contCines.isShowing()
&& textoHoras.isFocusOwner()){
                        if (contCineSalas==null){
                              contCineSalas = new HContainer(10, 10, 900,
900);
                              cargarCineSalasPage();
                        }
                        else{
                              scene.remove(contCines);
                              scene.add(contCineSalas);
                        }

                  }else if (this.contCines != null && contCines.isShowing()
&& textoResumen.isFocusOwner())
                        if (contCineResumen==null){
                              contCineResumen = new HContainer(10, 10, 900,
900);
                              cargarCineResumenesPage();
                        }
                        else{
```

```
                              scene.remove(contCines);
                              scene.add(contCineResumen);
                        }

                  break;
            case 405 ://boton amarillo

                  if (contCines !=null && contCines.isShowing()){
                        scene.remove(contCines);
                        scene.add(contPP);
                        textoCine.requestFocus();
                  }
                  if (contTeatro != null && contTeatro.isShowing()){
                        scene.remove(contTeatro);
                        scene.add(contPP);
                        textoCine.requestFocus();
                  }
                  if (contCineSalas !=null && contCineSalas.isShowing()){
                        scene.remove(contCineSalas);
                        scene.add(contCines);
                        textoHoras.requestFocus();
                  }
                  if (contCineResumen !=null &&
contCineResumen.isShowing()){
                        scene.remove(contCineResumen);
                        scene.add(contCines);
                        textoHoras.requestFocus();
                  }
                  break;
            case KeyEvent.VK_DOWN:

                  if (contTeatro != null && contTeatro.isShowing()){
                        actualizarPaginaTeatroAbajo();
                  }
                  if (contCineSalas !=null && contCineSalas.isShowing()){
                        actualizarPaginaSalasAbajo();
                  }
                  if (contCineResumen !=null &&
contCineResumen.isShowing()){
                        actualizarPaginaResumenAbajo();
                  }
                  break;

            case KeyEvent.VK_UP:

                  if (contTeatro != null && contTeatro.isShowing()){
                        actualizarPaginaTeatroArriba();
                  }
                  if (contCineSalas !=null && contCineSalas.isShowing()){
                        actualizarPaginaSalasArriba();
                  }
                  if (contCineResumen !=null &&
contCineResumen.isShowing()){
                        actualizarPaginaResumenArriba();
                  }
```

```
                    break;
                case 406:

                    if (contCineResumen !=null &&
contCineResumen.isShowing()){
                            System.out.println("Cargando cines salas...");
                            if (contCineSalas==null){
                                  contCineSalas = new HContainer(10, 10, 900,
900);
                                  scene.remove(contCineResumen);
                                  cargarCineSalasPage();
                            }
                            else{
                                  scene.remove(contCineResumen);

                                  botonInvisible = new
HTextButton("Atras",10,50,50,25);
                                  botonInvisible.addKeyListener(this);

      botonInvisible.addFocusListener((FocusListener) this);
                                  botonInvisible.setFont(new
Font("Verdana",Font.BOLD,5));

      botonInvisible.setBackgroundMode(Color.TRANSLUCENT);
                                  botonInvisible.setForeground(Color.BLACK);
                                  botonInvisible.setFocusable(true);
                                  contCineSalas.add(botonInvisible);
                                  scene.add(contCineSalas);
                                  botonInvisible.requestFocus();
                            }

                    }
                    else{
                                  if (contCineSalas !=null &&
contCineSalas.isShowing()){
                                        System.out.println("Cargando
resumenes...");
                                        if (contCineResumen==null){
                                              contCineResumen = new
HContainer(10, 10, 900, 900);

                                              scene.remove(contCineSalas);
                                              cargarCineResumenesPage();
                                        }
                                        else{
                                              scene.remove(contCineSalas);

                                              botonInvisible = new
HTextButton("Atras",10,50,50,25);

      botonInvisible.addKeyListener(this);

      botonInvisible.addFocusListener((FocusListener) this);
                                              botonInvisible.setFont(new
Font("Verdana",Font.BOLD,5));
```

```
		botonInvisible.setBackgroundMode(Color.TRANSLUCENT);

		botonInvisible.setForeground(Color.BLACK);
                                                botonInvisible.setFocusable(true);

		contCineResumen.add(botonInvisible);
                                                scene.add(contCineResumen);
                                                botonInvisible.requestFocus();
                                        }
                                }
                        }
                        break;
                default:
                        break;
                }


        }

        //Que hacer cuando los componentes obtienen el focus
        public void focusGained(FocusEvent e) {
                if(textoCine.isSelected()){
                        textoCine.setBackground(new Color(221,221,221));
                        textoTeatro.setBackground(Color.WHITE);
                }
                else{
                        textoCine.setBackground(Color.WHITE);
                        textoTeatro.setBackground(new Color(221,221,221));

                }
                if (contCines != null){
                        if(textoHoras.isSelected()){
                                textoHoras.setBackground(new Color(221,221,221));
                                textoResumen.setBackground(Color.WHITE);
                        }
                        if (textoResumen.isSelected()){
                                        textoHoras.setBackground(Color.WHITE);
                                        textoResumen.setBackground(new
Color(221,221,221));

                                }

                }

        }

        //Carga todos los componentes asociados a la pantalla principal
        public void cargarPPPage(){

                //Inicializo el contenedor de la pantalla principal
                contPP = new HContainer(10, 10, 900, 900);

                //Inicializo el Array donde se encuentran todos los botones
                botones = new HGraphicButton[9];
```

```
            //defino el titulo de la pantalla inicial
            textoCartelera = new HStaticText("Cartelera de
Espectáculos",0,20,700,67 );
            textoCartelera.setFont(new Font("Verdana",Font.BOLD,35));
            textoCartelera.setBackground(Color.BLACK);
            textoCartelera.setVisible(true);
            textoCartelera.setName("Cartelera");
            textoCartelera.setForeground(new Color(238,0,0));
            contPP.add(textoCartelera);


            if(esCarrusel){
                  cine =
Toolkit.getDefaultToolkit().getImage(cinesImgObj.getPath());
                  teatro =
Toolkit.getDefaultToolkit().getImage(teatrosImgObj.getPath());
                  botonVerde =
Toolkit.getDefaultToolkit().getImage(greenSpot.getPath());
            }
            else{
                  cine = Toolkit.getDefaultToolkit().getImage(pathCine);
                  teatro =
Toolkit.getDefaultToolkit().getImage(pathTeatro);
                  botonVerde =
Toolkit.getDefaultToolkit().getImage(pathVerde);

            }
            botones[0]= new HGraphicButton(cine, 100, 120, 76, 56);
            botones[0].setVisible(true);
            contPP.add(botones[0]);

            botones[1]= new HGraphicButton(teatro, 100, 190, 76, 56);
            botones[1].setVisible(true);
            contPP.add(botones[1]);

            textoCine = new HTextButton("Cartelera de Cines", 196, 120,
359, 56);
            textoCine.addKeyListener(this);
            textoCine.addFocusListener((FocusListener) this);
            textoCine.setFocusable(true);
            textoCine.setForeground(Color.BLACK);
            textoCine.setBackground(Color.white);
            Font fuente = new Font("Verdana",Font.BOLD,30);
            textoCine.setFont(fuente);
            textoCine.setVisible(true);

            contPP.add(textoCine);

            textoTeatro = new HTextButton("Cartelera de Teatros", 196, 190,
359, 56);
            textoTeatro.addKeyListener(this);
            textoTeatro.addFocusListener((FocusListener) this);
            textoTeatro.setForeground(Color.BLACK);
            textoTeatro.setBackground(Color.white);
```

```
            textoTeatro.setFont(fuente);

            textoTeatro.setVisible(true);

            botones[8]= new HGraphicButton(botonVerde, 550,510,40,40);
            contPP.add(botones[8]);
            textoBotonVerde = new HTextButton("Entrar");
            textoBotonVerde.setVisible(true);
            textoBotonVerde.setBounds(598,490, 100, 80);
            textoBotonVerde.setForeground(Color.BLACK);
            textoBotonVerde.setFont(new Font("Verdana",Font.BOLD,30));
            textoBotonVerde.setBackgroundMode(Color.TRANSLUCENT);
            contPP.add(textoBotonVerde);


            contPP.add(textoTeatro);
            scene.add(contPP);
            scene.setVisible(true);
            textoCine.requestFocus();
            textoCine.setFocusTraversal(textoTeatro,
textoTeatro,null,null);
            textoTeatro.setFocusTraversal(textoCine, textoCine, null,null);


      }

      //Carga todos los componentes asociados a la pantalla que muestra los
resumenes de las peliculas
      private void cargarCineResumenesPage() {

            HStaticText textoTituloCines = new HStaticText("Cine -
Resúmenes de Películas",0,20,700,67 );
            textoTituloCines.setFont(new Font("Verdana",Font.BOLD,35));
            textoTituloCines.setBackground(Color.BLACK);
            textoTituloCines.setForeground(new Color(238,0,0));
            textoTituloCines.setVisible(true);
            textoTituloCines.setFocusable(true);

            scene.remove(contCines);
            contCineResumen.add(textoTituloCines);
            textoTituloCines.requestFocus();

////////////////////////////////////////////////////////////////////////
            botonInvisible = new HTextButton("focus",10,50,50,25);
            botonInvisible.addKeyListener(this);
            botonInvisible.addFocusListener((FocusListener) this);
            botonInvisible.setBackgroundMode(Color.TRANSLUCENT);
            botonInvisible.setFocusable(true);
            contCineResumen.add(botonInvisible);
            Auxiliar aux = new Auxiliar();
////////////////////////////////////////////////////////////////////////
            if(esCarrusel){
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(yellowSpot.getPath());
```

```
                botonArriba =
Toolkit.getDefaultToolkit().getImage(arrowUp.getPath());
                botonAbajo =
Toolkit.getDefaultToolkit().getImage(arrowDown.getPath());
                botonAzul =
Toolkit.getDefaultToolkit().getImage(blueSpot.getPath());
                lasPeli =
(LinkedList)aux.obtenerPeliculas(peliculasObj.getPath());
            }
            else{
                botonAtras =
Toolkit.getDefaultToolkit().getImage(pathAmarillo);
                botonArriba =
Toolkit.getDefaultToolkit().getImage(pathFlechaArriba);
                botonAbajo =
Toolkit.getDefaultToolkit().getImage(pathFlechaAbajo);
                botonAzul =
Toolkit.getDefaultToolkit().getImage(pathAzul);
                lasPeli = (LinkedList)aux.obtenerPeliculas(pathCinetxt);
            }



            botonArriba =
Toolkit.getDefaultToolkit().getImage("images/uparrow.png");
            botonAbajo =
Toolkit.getDefaultToolkit().getImage("images/downarrow.png");
            botonAzul =
Toolkit.getDefaultToolkit().getImage("images/bluespot.png");
            botones[4]= new HGraphicButton(botonAtras, 140,510,40,40);
            botones[5]= new HGraphicButton(botonArriba, 660,150,40,40);
            botones[6]= new HGraphicButton(botonAbajo, 660,400,40,40);
            botones[7]= new HGraphicButton(botonAzul, 320,510,40,40);
            botones[4].setVisible(true);
            botones[5].setVisible(true);
            botones[6].setVisible(true);
            botones[7].setVisible(true);
            contCineResumen.add(botones[4]);
            contCineResumen.add(botones[5]);
            contCineResumen.add(botones[6]);
            contCineResumen.add(botones[7]);
            botones[4].setVisible(true);
            textoAtras = new HStaticText("Atrás");
            textoAtras.setFont(new Font("Verdana",Font.BOLD,30));
            textoAtras.setVisible(true);
            textoAtras.setBounds(190, 490, 80, 80);
            textoAtras.setBackgroundMode(Color.TRANSLUCENT);
            textoHorario = new HStaticText("Horarios");
            textoHorario.setFont(new Font("Verdana",Font.BOLD,30));
            textoHorario.setVisible(true);
            textoHorario.setBounds(350, 490, 160, 80);
            textoHorario.setBackgroundMode(Color.TRANSLUCENT);
            contCineResumen.add(textoAtras);
            contCineResumen.add(textoHorario);
            contCineResumen.add(botones[4]);
```

```
            Pelicula peliUno = (Pelicula) lasPeli.get(0);
            String texto = aux.prepararPresentacion(peliUno);
            peliTextoUno = new HStaticText(texto,10,150,650,115);
            peliTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
            peliTextoUno.setBackground(new Color(221,221,221));

      peliTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      peliTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      peliTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
            peliTextoUno.setVisible(true);
            peliTextoUno.setFocusable(true);
            peliUno.setShowing(true);
            contCineResumen.add(peliTextoUno);

            if (lasPeli.size()> 1){
                  Pelicula peliDos = (Pelicula) lasPeli.get(1);
                  String textoDos = aux.prepararPresentacion(peliDos);
                  peliTextoDos = new HStaticText(textoDos,10,325,650,115);
                  peliTextoDos.setFont(new Font("Verdana",Font.BOLD,18));
                  peliTextoDos.setBackground(new Color(221,221,221));

      peliTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      peliTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      peliTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                  peliTextoDos.setVisible(true);
                  peliTextoDos.setFocusable(true);
                  peliDos.setShowing(true);
                  contCineResumen.add(peliTextoDos);

            }

            contCineResumen.remove(botones[5]);
            scene.add(contCineResumen);
            botonInvisible.requestFocus();
}

      //Carga todos los componentes asociados a la pantalla que muestra las
salas donde se exhiben las peliculas
      private void cargarCineSalasPage() {
            HStaticText textoTituloCines = new HStaticText("Cines-Salas y
Horarios",0,20,700,67 );
            textoTituloCines.setFont(new Font("Verdana",Font.BOLD,35));
```

```
            textoTituloCines.setBackground(Color.BLACK);
            textoTituloCines.setForeground(new Color(238,0,0));
            textoTituloCines.setVisible(true);
            textoTituloCines.setFocusable(true);

            scene.remove(contCines);
            contCineSalas.add(textoTituloCines);
            textoTituloCines.requestFocus();

////////////////////////////////////////////////////////////////////////
            botonInvisible = new HTextButton("focus",10,50,50,25);
            botonInvisible.addKeyListener(this);
            botonInvisible.addFocusListener((FocusListener) this);
            botonInvisible.setBackgroundMode(Color.TRANSLUCENT);
            botonInvisible.setFocusable(true);
            contCineSalas.add(botonInvisible);
////////////////////////////////////////////////////////////////////////
            Auxiliar aux = new Auxiliar();

            if(esCarrusel){
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(yellowSpot.getPath());
                  botonArriba =
Toolkit.getDefaultToolkit().getImage(arrowUp.getPath());
                  botonAbajo =
Toolkit.getDefaultToolkit().getImage(arrowDown.getPath());
                  botonAzul =
Toolkit.getDefaultToolkit().getImage(blueSpot.getPath());
                  lasSalas =
(LinkedList)aux.obtenerSalas(salasObj.getPath());
            }
            else{
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(pathAmarillo);
                  botonArriba =
Toolkit.getDefaultToolkit().getImage(pathFlechaArriba);
                  botonAbajo =
Toolkit.getDefaultToolkit().getImage(pathFlechaAbajo);
                  botonAzul =
Toolkit.getDefaultToolkit().getImage(pathAzul);
                  lasSalas = (LinkedList)aux.obtenerSalas(pathSalatxt);
            }

            botones[4]= new HGraphicButton(botonAtras, 140,510,40,40);
            botones[5]= new HGraphicButton(botonArriba, 660,150,40,40);
            botones[6]= new HGraphicButton(botonAbajo, 660,400,40,40);
            botones[7]= new HGraphicButton(botonAzul, 300,510,40,40);
            botones[4].setVisible(true);
            botones[5].setVisible(true);
            botones[6].setVisible(true);
            contCineSalas.add(botones[4]);
            contCineSalas.add(botones[5]);
            contCineSalas.add(botones[6]);
            contCineSalas.add(botones[7]);
            textoAtras = new HStaticText("Atrás");
```

```
            textoAtras.setFont(new Font("Verdana",Font.BOLD,30));
            textoAtras.setVisible(true);
            textoAtras.setBounds(190, 490, 80, 80);
            textoAtras.setBackgroundMode(Color.TRANSLUCENT);
            contCineSalas.add(textoAtras);
            textoResumenes = new HStaticText("Resúmenes");
            textoResumenes.setFont(new Font("Verdana",Font.BOLD,30));
            textoResumenes.setVisible(true);
            textoResumenes.setBounds(350, 490, 180, 80);
            textoResumenes.setBackgroundMode(Color.TRANSLUCENT);
            contCineSalas.add(textoResumenes);

            Sala salaUno = (Sala) lasSalas.get(0);
            String texto = aux.prepararPresentacion(salaUno);


            obraTextoUno = new HStaticText(texto,10,100,650,65 +
salaUno.getCantidadPeliculas()*20);
            obraTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
            obraTextoUno.setBackground(new Color(221,221,221));

      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
            obraTextoUno.setVisible(true);
            obraTextoUno.setFocusable(true);
            salaUno.setShowing(true);
            contCineSalas.add(obraTextoUno);
            Sala salaDos = (Sala) lasSalas.get(1);

            int cantidadTotal = salaUno.getCantidadPeliculas() +
salaDos.getCantidadPeliculas();
            if (cantidadTotal <= 14){ //Entonces puedo cargar una segunda
sala
                  texto = aux.prepararPresentacion(salaDos);
                  obraTextoDos = new HStaticText(texto,10,175
+salaUno.getCantidadPeliculas()*20,650,65 +
salaDos.getCantidadPeliculas()*20);
                  obraTextoDos.setFont(new Font("Verdana",Font.BOLD,18));
                  obraTextoDos.setBackground(new Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                  obraTextoDos.setVisible(true);
```

```
                  obraTextoDos.setFocusable(true);
                  salaDos.setShowing(true);
                  contCineSalas.add(obraTextoDos);

            }
            contCineSalas.remove(botones[5]);
            contCineSalas.add(botones[4]);



            scene.add(contCineSalas);
            botonInvisible.requestFocus();

      }

      //Carga todos los componentes asociados a la pantalla que muestra las
opciones de la cartelera de Cine
      private void cargarCinesPage() {


            HStaticText textoTituloCines = new HStaticText("Cartelera de
Cines",0,20,700,67 );
            textoTituloCines.setFont(new Font("Verdana",Font.BOLD,35));
            textoTituloCines.setBackground(Color.BLACK);
            textoTituloCines.setForeground(new Color(238,0,0));
            textoTituloCines.setVisible(true);
            textoTituloCines.setFocusable(true);

            scene.remove(contPP);
            contCines.add(textoTituloCines);
            textoTituloCines.requestFocus();

            if(esCarrusel){
                  horario =
Toolkit.getDefaultToolkit().getImage(horarioImgObj.getPath());
                  resumen =
Toolkit.getDefaultToolkit().getImage(resumenImgObj.getPath());
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(yellowSpot.getPath());
                  botonVerde =
Toolkit.getDefaultToolkit().getImage(greenSpot.getPath());
            }
            else{
                  horario =
Toolkit.getDefaultToolkit().getImage(pathHorario);
                  resumen =
Toolkit.getDefaultToolkit().getImage(pathResumen);
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(pathAmarillo);
                  botonVerde =
Toolkit.getDefaultToolkit().getImage(pathVerde);


            }
```

```
            botones[2]= new HGraphicButton(horario, 100, 120, 76, 56);
            botones[2].setVisible(true);

            contCines.add(botones[2]);

            botones[3]= new HGraphicButton(resumen, 100, 190, 76, 56);
            botones[3].setVisible(true);
            contCines.add(botones[3]);

            textoHoras = new HTextButton("Ver Salas y Horarios", 196, 120,
460, 56);
            textoHoras.addKeyListener(this);
            textoHoras.addFocusListener((FocusListener) this);
            textoHoras.setFocusable(true);
            textoHoras.setForeground(Color.BLACK);
            textoHoras.setBackground(Color.white);

            Font fuente = new Font("Verdana",Font.BOLD,30);
            textoHoras.setFont(fuente);
            textoHoras.setVisible(true);
            contCines.add(textoHoras);

            textoResumen = new HTextButton("Ver Resúmenes de Películas",
196, 190, 460, 56);
            textoResumen.addKeyListener(this);
            textoResumen.addFocusListener((FocusListener) this);
            textoResumen.setFocusable(true);
            textoResumen.setForeground(Color.BLACK);
            textoResumen.setBackground(Color.white);
            textoResumen.setFont(fuente);
            textoResumen.setVisible(true);
            contCines.add(textoResumen);


            botones[4]= new HGraphicButton(botonAtras, 310,510,40,40);
            botones[4].setVisible(true);
            contCines.add(botones[4]);

            textoAtras = new HStaticText("Atrás");
            textoAtras.setFont(new Font("Verdana",Font.BOLD,30));
            textoAtras.setVisible(true);
            textoAtras.setBounds(360, 490, 80, 80);
            textoAtras.setBackgroundMode(Color.TRANSLUCENT);
            contCines.add(textoAtras);

            contCines.add(botones[4]);

            botones[8]= new HGraphicButton(botonVerde, 550,510,40,40);
            contCines.add(botones[8]);
            textoBotonVerde = new HTextButton("Entrar");
            textoBotonVerde.setVisible(true);
            textoBotonVerde.setBounds(598,490, 100, 80);
            textoBotonVerde.setForeground(Color.BLACK);
            textoBotonVerde.setFont(new Font("Verdana",Font.BOLD,30));
            textoBotonVerde.setBackgroundMode(Color.TRANSLUCENT);
```

```
            contCines.add(textoBotonVerde);

            scene.add(contCines);
            textoHoras.requestFocus();
            textoHoras.setFocusTraversal(textoResumen,
textoResumen,null,null);
            textoResumen.setFocusTraversal(textoHoras, textoHoras,
null,null);



      }

      //Carga todos los componentes asociados a la pantalla que muesta la
cartelera de Teatro
      private void cargarTeatrosPage() {

            Auxiliar aux = new Auxiliar();
            if(esCarrusel){
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(yellowSpot.getPath());
                  botonArriba =
Toolkit.getDefaultToolkit().getImage(arrowUp.getPath());
                  botonAbajo =
Toolkit.getDefaultToolkit().getImage(arrowDown.getPath());
                  lasObras =
(LinkedList)aux.obtenerObras(teatroObj.getPath());
            }
            else{
                  botonAtras =
Toolkit.getDefaultToolkit().getImage(pathAmarillo);
                  botonArriba =
Toolkit.getDefaultToolkit().getImage(pathFlechaArriba);
                  botonAbajo =
Toolkit.getDefaultToolkit().getImage(pathFlechaAbajo);
                  lasObras = (LinkedList)aux.obtenerObras(pathTeatrotxt);
            }

            HStaticText textoTituloTeatro = new HStaticText("Cartelera de
Teatros",0,20,700,67 );
            textoTituloTeatro.setFont(new Font("Verdana",Font.BOLD,35));
            textoTituloTeatro.setBackground(Color.BLACK);
            textoTituloTeatro.setForeground(new Color(238,0,0));
            textoTituloTeatro.setVisible(true);
            textoTituloTeatro.setFocusable(true);

            scene.remove(contPP);
            contTeatro.add(textoTituloTeatro);
////////////////////////////////////////////////////////////////////

            botonInvisible = new HTextButton("Focus",10,50,50,25);
            botonInvisible.addKeyListener(this);
            botonInvisible.addFocusListener((FocusListener) this);
            botonInvisible.setBackgroundMode(Color.TRANSLUCENT);
            botonInvisible.setFocusable(true);
```

```
            contTeatro.add(botonInvisible);

////////////////////////////////////////////////////////////////////

            botones[4]= new HGraphicButton(botonAtras, 310,510,40,40);
            botones[5]= new HGraphicButton(botonArriba, 660,150,40,40);
            botones[6]= new HGraphicButton(botonAbajo, 660,400,40,40);
            botones[4].setVisible(true);
            botones[5].setVisible(true);
            botones[6].setVisible(true);
            contTeatro.add(botones[4]);
            contTeatro.add(botones[6]);

            textoAtras = new HStaticText("Atrás");
            textoAtras.setFont(new Font("Verdana",Font.BOLD,30));
            textoAtras.setVisible(true);
            textoAtras.setBounds(360, 490, 80, 80);
            textoAtras.setBackgroundMode(Color.TRANSLUCENT);
            contTeatro.add(textoAtras);
            contTeatro.setVisible(true);



            Obra obraUno = (Obra) lasObras.get(0);
            obraUno.setShowing(true);
            String texto = aux.prepararPresentacion(obraUno);
            obraTextoUno = new HStaticText(texto,10,100,650,200);
            obraTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
            obraTextoUno.setBackground(new Color(221,221,221));

      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
            obraTextoUno.setVisible(true);
            obraTextoUno.setFocusable(true);
            contTeatro.add(obraTextoUno);

            Obra obraDos = (Obra) lasObras.get(1);
            obraDos.setShowing(true);
            String textoDos = aux.prepararPresentacion(obraDos);
            obraTextoDos = new HStaticText(textoDos,10,310,650,200);
            obraTextoDos.setFont(new Font("Verdana",Font.BOLD,18));
            obraTextoDos.setBackground(new Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);
```

```
      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
            obraTextoDos.setVisible(true);
            obraTextoDos.setFocusable(true);
            contTeatro.add(obraTextoDos);


            scene.add(contTeatro);
            scene.setVisible(true);
            botonInvisible.requestFocus();


      }

      //Actualiza el contenido de cartelera de teatros en el caso de que
que la flecha hacia arriba sea accionada
      private void actualizarPaginaTeatroArriba() {

            Auxiliar aux = new Auxiliar();
            LinkedList<Objeto> nuevasObras;
            try {
                  nuevasObras = aux.obtenerAnteriores(this.lasObras);
                  contTeatro.remove(obraTextoUno);
                  contTeatro.remove(obraTextoDos);
                  Obra obraUno = (Obra) nuevasObras.get(0);
                  obraUno.setShowing(true);

                  String texto = aux.prepararPresentacion(obraUno);
                  obraTextoUno = new HStaticText(texto,10,100,650,200);
                  obraTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
                  obraTextoUno.setBackground(new Color(221,221,221));

      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                  obraTextoUno.setVisible(true);
                  obraTextoUno.setFocusable(true);
                  contTeatro.add(obraTextoUno);
                  Obra obraDos = (Obra) nuevasObras.get(1);

                  obraDos.setShowing(true);
                  String textoDos = aux.prepararPresentacion(obraDos);
                  obraTextoDos = new HStaticText(textoDos,10,310,650,200);
                  obraTextoDos.setFont(new Font("Verdana",Font.BOLD,18));
                  obraTextoDos.setBackground(new Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);
```

```
      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                  obraTextoDos.setVisible(true);
                  obraTextoDos.setFocusable(true);
                  contTeatro.add(obraTextoDos);
                  contTeatro.add(botones[6]);//boton abajjo
                  if (obraDos.getNumero() == 2){
                        contTeatro.remove(botones[5]);
                  }


                  scene.add(contTeatro);
                  scene.setVisible(true);
                  botonInvisible.requestFocus();
            } catch (SonlosMismosException e) {
                  // TODO Auto-generated catch block

            }

}

      //Actualiza el contenido de cartelera de cines por resumenes en el
caso de que la flecha hacia arriba sea accionada
      private void actualizarPaginaResumenArriba() {

                  Auxiliar aux = new Auxiliar();
                  LinkedList<Objeto> nuevasPelis;
                  try {
                        nuevasPelis = aux.obtenerAnteriores(this.lasPeli);
                        contCineResumen.remove(peliTextoUno);
                        contCineResumen.remove(peliTextoDos);
                        Pelicula peliUno = (Pelicula) nuevasPelis.get(0);

                        String texto = aux.prepararPresentacion(peliUno);
                        peliTextoUno = new
HStaticText(texto,10,150,650,115);
                        peliTextoUno.setFont(new
Font("Verdana",Font.BOLD,18));
                        peliTextoUno.setBackground(new Color(221,221,221));

      peliTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      peliTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      peliTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                        peliTextoUno.setVisible(true);
                        peliTextoUno.setFocusable(true);
                        peliUno.setShowing(true);
```

```
                        contCineResumen.add(peliTextoUno);

                        if(peliUno.getNumero()== 1){
                              contCineResumen.remove(botones[5]);
                        }

                        if (nuevasPelis.size() > 1){
                              Pelicula peliDos = (Pelicula)
nuevasPelis.get(1);
                              texto = aux.prepararPresentacion(peliDos);
                              peliTextoDos = new
HStaticText(texto,10,325,650,115);
                              peliTextoDos.setFont(new
Font("Verdana",Font.BOLD,18));
                              peliTextoDos.setBackground(new
Color(221,221,221));

      peliTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      peliTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      peliTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                              peliTextoDos.setVisible(true);
                              peliTextoDos.setFocusable(true);
                              peliDos.setShowing(true);
                              contCineResumen.add(peliTextoDos);



                        }

                        contCineResumen.add(botones[6]);
                        scene.add(contCineResumen);
                        scene.setVisible(true);
                        botonInvisible.requestFocus();

                  } catch (SonlosMismosException e) {
                        // TODO Auto-generated catch block
                        System.out.println("Son los mismos no hace falta
actualizar");
                  }
            //}


      }

      //Actualiza el contenido de cartelera de cines por sala en el caso de
que la flecha hacia arriba sea accionada
      private void actualizarPaginaSalasArriba() {
            Auxiliar aux = new Auxiliar();
            LinkedList<Objeto> nuevasSalas;
            try {
```

```
                    nuevasSalas = aux.obtenerAnteriores(this.lasSalas);
                    contCineSalas.remove(obraTextoUno);
                    contCineSalas.remove(obraTextoDos);
                    Sala salaUno = (Sala) nuevasSalas.get(0);

                    String texto = aux.prepararPresentacion(salaUno);
                    obraTextoUno = new HStaticText(texto,10,100,650,65 +
salaUno.getCantidadPeliculas()*20);
                    obraTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
                    obraTextoUno.setBackground(new Color(221,221,221));

      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                    obraTextoUno.setVisible(true);
                    obraTextoUno.setFocusable(true);
                    salaUno.setShowing(true);
                    contCineSalas.add(obraTextoUno);
                    if(salaUno.getNumero()== 1){
                          contCineSalas.remove(botones[5]);
                    }

                    Sala salaDos = (Sala) nuevasSalas.get(1);

                    int cantidadTotal = salaUno.getCantidadPeliculas() +
salaDos.getCantidadPeliculas();
                    if (cantidadTotal <= 14){ //Entonces puedo cargar una
segunda sala
                          texto = aux.prepararPresentacion(salaDos);
                          obraTextoDos = new HStaticText(texto,10,175
+salaUno.getCantidadPeliculas()*20,650,65 +
salaDos.getCantidadPeliculas()*20);
                          obraTextoDos.setFont(new
Font("Verdana",Font.BOLD,18));
                          obraTextoDos.setBackground(new Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                          obraTextoDos.setVisible(true);
                          obraTextoDos.setFocusable(true);
                          salaDos.setShowing(true);
                          contCineSalas.add(obraTextoDos);

                    }
```

```
                    contCineSalas.add(botones[6]);
                    scene.add(contCineSalas);
                    scene.setVisible(true);
                    botonInvisible.requestFocus();

              } catch (SonlosMismosException e) {
                    // TODO Auto-generated catch block
              }
        }

        //Actualiza el contenido de cartelera de cines por resumenes en el
caso de que la flecha hacia abajo sea accionada
        private void actualizarPaginaResumenAbajo() {

                    Auxiliar aux = new Auxiliar();
                    LinkedList<Objeto> nuevasPelis;
                    try {
                          nuevasPelis = aux.obtenerSiguientes(this.lasPeli);
                          contCineResumen.remove(peliTextoUno);
                          contCineResumen.remove(peliTextoDos);
                          Pelicula peliUno = (Pelicula) nuevasPelis.get(0);
                          String texto = aux.prepararPresentacion(peliUno);
                          peliTextoUno = new
HStaticText(texto,10,150,650,115);
                          peliTextoUno.setFont(new
Font("Verdana",Font.BOLD,18));
                          peliTextoUno.setBackground(new Color(221,221,221));

        peliTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

        peliTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

        peliTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                          peliTextoUno.setVisible(true);
                          peliTextoUno.setFocusable(true);
                          peliUno.setShowing(true);
                          contCineResumen.add(peliTextoUno);

                          if (nuevasPelis.size() > 1){
                                Pelicula peliDos = (Pelicula)
nuevasPelis.get(1);
                                texto = aux.prepararPresentacion(peliDos);
                                peliTextoDos = new
HStaticText(texto,10,325,650,115);
                                peliTextoDos.setFont(new
Font("Verdana",Font.BOLD,18));
                                peliTextoDos.setBackground(new
Color(221,221,221));

        peliTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);
```

```
      peliTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.LINE_ORIENTATION_HORIZONTAL);

      peliTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VERTICAL_START_ALIGN);
                              peliTextoDos.setVisible(true);
                              peliTextoDos.setFocusable(true);
                              peliDos.setShowing(true);
                              contCineResumen.add(peliTextoDos);
                              if (this.lasPeli.size() == peliDos.getNumero()){
                                    contCineResumen.remove(botones[6]);
                              }


                        }


                        if (this.lasPeli.size() == peliUno.getNumero()){

                              contCineResumen.remove(botones[6]);
                        }
                        contCineResumen.add(botones[5]);
                        scene.add(contCineResumen);
                        scene.setVisible(true);
                        botonInvisible.requestFocus();

                  } catch (SonlosMismosException e) {
                        // TODO Auto-generated catch block

                  }
            //}

      }


      //Actualiza el contenido de cartelera de cines por sala en el caso de que la flecha hacia abajo sea accionada
      private void actualizarPaginaSalasAbajo() {

                  Auxiliar aux = new Auxiliar();
                  LinkedList<Objeto> nuevasSalas;
                  try {
                        nuevasSalas = aux.obtenerSiguientes(this.lasSalas);
                        contCineSalas.remove(obraTextoUno);
                        contCineSalas.remove(obraTextoDos);
                        Sala salaUno = (Sala) nuevasSalas.get(0);

                        String texto = aux.prepararPresentacion(salaUno);
                        obraTextoUno = new HStaticText(texto,10,100,650,65 + salaUno.getCantidadPeliculas()*20);
                        obraTextoUno.setFont(new Font("Verdana",Font.BOLD,18));
                        obraTextoUno.setBackground(new Color(221,221,221));
```

```
      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                        obraTextoUno.setVisible(true);
                        obraTextoUno.setFocusable(true);
                        salaUno.setShowing(true);
                        contCineSalas.add(obraTextoUno);


                        if (nuevasSalas.size() > 1){
                              Sala salaDos = (Sala) nuevasSalas.get(1);
                              int cantidadTotal =
salaUno.getCantidadPeliculas() + salaDos.getCantidadPeliculas();
                              if (cantidadTotal <= 14){ //Entonces puedo
cargar una segunda sala


                              texto = aux.prepararPresentacion(salaDos);
                              obraTextoDos = new HStaticText(texto,10,175
+salaUno.getCantidadPeliculas()*20,650,65 +
salaDos.getCantidadPeliculas()*20);
                              obraTextoDos.setFont(new
Font("Verdana",Font.BOLD,18));
                              obraTextoDos.setBackground(new
Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                              obraTextoDos.setVisible(true);
                              obraTextoDos.setFocusable(true);
                              salaDos.setShowing(true);
                              contCineSalas.add(obraTextoDos);
                              if (this.lasSalas.size() ==
salaDos.getNumero()){
                                    contCineSalas.remove(botones[6]);
                              }
                              }

                        }
                        if (this.lasSalas.size() == salaUno.getNumero()){
                              contCineSalas.remove(botones[6]);
                        }
                        contCineSalas.add(botones[5]);
```

```
                        scene.add(contCineSalas);
                        scene.setVisible(true);
                        botonInvisible.requestFocus();

                  } catch (SonlosMismosException e) {
                        // TODO Auto-generated catch block

                  }
            //}

      }

      //Actualiza el contenido de cartelera de teatros en el caso de que
que la flecha hacia arriba sea accionada
      private void actualizarPaginaTeatroAbajo() {

                  Auxiliar aux = new Auxiliar();
                  LinkedList<Objeto> nuevasObras;
                  try {
                        nuevasObras = aux.obtenerSiguientes(this.lasObras);
                        contTeatro.remove(obraTextoUno);
                        contTeatro.remove(obraTextoDos);
                        Obra obraUno = (Obra) nuevasObras.get(0);
                        obraUno.setShowing(true);

                        String texto = aux.prepararPresentacion(obraUno);
                        obraTextoUno = new
HStaticText(texto,10,100,650,200);
                        obraTextoUno.setFont(new
Font("Verdana",Font.BOLD,18));
                        obraTextoUno.setBackground(new Color(221,221,221));

      obraTextoUno.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoUno.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoUno.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                        obraTextoUno.setVisible(true);
                        obraTextoUno.setFocusable(true);
                        contTeatro.add(obraTextoUno);

                        if (nuevasObras.size()> 1){
                              Obra obraDos = (Obra) nuevasObras.get(1);

                              obraDos.setShowing(true);
                              String textoDos =
aux.prepararPresentacion(obraDos);
                              obraTextoDos = new
HStaticText(textoDos,10,310,650,200);
                              obraTextoDos.setFont(new
Font("Verdana",Font.BOLD,18));
```

```
                              obraTextoDos.setBackground(new
Color(221,221,221));

      obraTextoDos.setComponentOrientation(ComponentOrientation.LEFT_TO_RIG
HT);

      obraTextoDos.setHorizontalAlignment(org.dvb.ui.DVBTextLayoutManager.L
INE_ORIENTATION_HORIZONTAL);

      obraTextoDos.setVerticalAlignment(org.dvb.ui.DVBTextLayoutManager.VER
TICAL_START_ALIGN);
                              obraTextoDos.setVisible(true);
                              obraTextoDos.setFocusable(true);
                              contTeatro.add(obraTextoDos);
                        }else {
                              contTeatro.remove(botones[6]);//Boton abajo
                        }
                        contTeatro.add(botones[5]);//Boton arriba
                        scene.add(contTeatro);
                        scene.setVisible(true);
                        botonInvisible.requestFocus();
                  } catch (SonlosMismosException e) {
                        // TODO Auto-generated catch block
                        contTeatro.remove(botones[6]);//Boton abajo
                  }
            //}
}



      public void keyTyped(KeyEvent ignored) { }

      public void keyReleased(KeyEvent ignored) { }


      public void focusLost(FocusEvent e) {
            // TODO Auto-generated method stub


      }


      //Carga de objetos del carrusel

      public void cargarObjetos(){

            ServiceDomain carousel = new ServiceDomain();


            try {

                  /*Por lo tanto primero se obtiene una referencia al
service context en el que corre la aplicación. Para hacer esto se pide el
service context que contiene al Xlet context. Es posible que se largue una
excepcion*/
```

```
            scf = ServiceContextFactory.getInstance();
            sc = scf.getServiceContext(context);


            //Ahora se obtiene el objeto JavaTV que representa al servicio
            Service s;
            s=sc.getService();

            //Finalmente se obtiene el locator del presente servicio
            locator = (DvbLocator) s.getLocator();

            carousel.attach(locator);


            } catch (ServiceXFRException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (MPEGDeliveryException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (InterruptedIOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (DSMCCException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (ServiceContextException e) {
                  // No hay nada que se pueda hacer al respecto
                  System.out.println("Error al obtener el service
context!");
                  e.printStackTrace();
            }
            } catch (SecurityException e) {
                  e.printStackTrace();
            }




            // tenemos que crear nuestro DSMCCObject con un path absolute,
lo que
            //significa que tenemos que tener un mount point para el
service domain

            peliculasObj = new
DSMCCObject(carousel.getMountPoint(),pathCinetxt);
            teatroObj = new
DSMCCObject(carousel.getMountPoint(),pathTeatrotxt);
            salasObj = new
DSMCCObject(carousel.getMountPoint(),pathSalatxt);
            redSpot = new DSMCCObject(carousel.getMountPoint(),pathRojo);
            greenSpot = new
DSMCCObject(carousel.getMountPoint(),pathVerde);
```

```
            yellowSpot = new
DSMCCObject(carousel.getMountPoint(),pathAmarillo);
            blueSpot = new DSMCCObject(carousel.getMountPoint(),pathAzul);
            arrowUp = new
DSMCCObject(carousel.getMountPoint(),pathFlechaArriba);
            arrowDown = new
DSMCCObject(carousel.getMountPoint(),pathFlechaAbajo);
            cinesImgObj = new
DSMCCObject(carousel.getMountPoint(),pathCine);
            teatrosImgObj = new
DSMCCObject(carousel.getMountPoint(),pathTeatro);
            horarioImgObj = new
DSMCCObject(carousel.getMountPoint(),pathHorario);
            resumenImgObj = new
DSMCCObject(carousel.getMountPoint(),pathResumen);

            AsynchronousLoadingEventListener myListener = null;

            try {
                  peliculasObj.asynchronousLoad(myListener);
                  teatroObj.asynchronousLoad(myListener);
                  salasObj.asynchronousLoad(myListener);
                  redSpot.asynchronousLoad(myListener);
                  greenSpot.asynchronousLoad(myListener);
                  yellowSpot.asynchronousLoad(myListener);
                  blueSpot.asynchronousLoad(myListener);
                  arrowUp.asynchronousLoad(myListener);
                  arrowDown.asynchronousLoad(myListener);
                  cinesImgObj.asynchronousLoad(myListener);
                  teatrosImgObj.asynchronousLoad(myListener);
                  horarioImgObj.asynchronousLoad(myListener);
                  resumenImgObj.asynchronousLoad(myListener);
            } catch (InvalidPathNameException e1) {
                  // TODO Auto-generated catch block
                  e1.printStackTrace();
            }


            System.out.println("Path: "+redSpot.getAbsolutePath());
            try {

                  BufferedReader in = new BufferedReader(new
FileReader(peliculasObj));
                  String ayuda="";
                  String texto ="";
                  String line="";
                  while((line = in.readLine()) != null){
                        texto=texto+line;

                  }
                  ayuda = texto;
                  System.out.println(ayuda);

                   in = new BufferedReader(new FileReader(teatroObj));
                  ayuda="";
```

```
                    texto ="";
                    line="";
                    while((line = in.readLine()) != null){
                          texto=texto+line;

                    }
                    ayuda = texto;
                    System.out.println(ayuda);


                     in = new BufferedReader(new FileReader(salasObj));
                          ayuda="";
                          texto ="";
                          line="";
                          while((line = in.readLine()) != null){
                                texto=texto+line;

                          }
                          ayuda = texto;
                          System.out.println(ayuda);

                          in = new BufferedReader(new FileReader(redSpot));
                          System.out.println("REdSPot: "+in.read());
                          in = new BufferedReader(new FileReader(greenSpot));
                          System.out.println("greenSPot: "+in.read());
                          in = new BufferedReader(new
FileReader(yellowSpot));
                          System.out.println("yellowPot: "+in.read());
                          in = new BufferedReader(new FileReader(blueSpot));
                          System.out.println("blueSPot: "+in.read());
                          in = new BufferedReader(new FileReader(arrowUp));
                          System.out.println("arrowUp: "+in.read());
                          in = new BufferedReader(new FileReader(arrowDown));
                          System.out.println("arrowDown: "+in.read());
                          in = new BufferedReader(new
FileReader(cinesImgObj));
                          System.out.println("cinesImgObj: "+in.read());
                          in = new BufferedReader(new
FileReader(teatrosImgObj));
                          System.out.println("teatrosImgObj: "+in.read());
                          in = new BufferedReader(new
FileReader(horarioImgObj));
                          System.out.println("horarioImgObj: "+in.read());
                          in = new BufferedReader(new
FileReader(resumenImgObj));
                          System.out.println("resumenImgObj: "+in.read());

                    in.close();


              } catch (FileNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
```

```
                  e.printStackTrace();
            }


      }
      public static void main (String[] args){
            Inicio inicio = new Inicio();
            inicio.cargarObjetos();
      }

}
```

Auxiliar.java

```
package cartelera;


import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;


import excepcion.SonlosMismosException;



public class Auxiliar {


      private BufferedReader in;



      public Auxiliar(){

      }

      public List<Sala> obtenerSalas(String ipath){

            LinkedList<Sala> ayuda = new LinkedList<Sala>();

            String path = ipath;
            String res = preparoTexto(path);

            //Ahora que tengo el archivo preparado creo las instancias de
las salas

            String [] todo = res.split(";;;;");
            for (int j=1 ; j< todo.length; j++){
                  Sala sala = new Sala(j);
```

```
                String actual = todo[j];
                String[] porSala = actual.split("::");
                int i=0;
                int cantidad =1;
                while (i < porSala.length){
                      if (i > 3){
                            cantidad++;
                            String pelis = sala.getPeliculas();
                            sala.setPeliculas(pelis+"\r\n"+porSala[i]);

                      }else{
                            switch (i){
                            case  0:
                                  sala.setNombre(porSala[i]);
                                  break;
                            case 1:
                                  sala.setDireccion(porSala[i]);
                                  break;
                            case 2:
                                  sala.setTelefono(porSala[i]);
                                  break;
                            case 3:
                                  sala.setPeliculas(porSala[i]);
                            }
                      }
                      i++;
                }
                sala.setCantidadPeliculas(cantidad);
                ayuda.add(sala);
          }



          return ayuda;

    }

    private LinkedList<Obra> reordenarObras(LinkedList<Obra> lista) {
          LinkedList<Obra> ayuda = new LinkedList<Obra>();
          int cantidad = lista.size();

          for (int i=0; i < cantidad; i++){
                Obra obra = lista.get(i);
                obra.setNumero(i+1);
                ayuda.add(obra);
          }
          return ayuda;
    }

    public String preparoTexto(String path){

          String ayuda="";
          try {
                this.in = new BufferedReader(new FileReader(path));
```

```
                  String texto ="";
                  String line="";
                  while((line = in.readLine()) != null){
                        texto=texto+line;

                  }
                  ayuda = texto;
                  this.in.close();
            } catch (FileNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return ayuda;

      }
      public void prepararResumen(Obra iobra){

            String aux="";
            String resumen = iobra.getResumen();
            iobra.setResumen("Resumen: "+resumen);
            resumen = iobra.getResumen();
            int i=0;

            double largo = resumen.length();

            if ((largo/67) > 3.0){
                  String ayuda = resumen.substring(0, 201);

                  resumen =  new String(ayuda);
                  iobra.setResumen(ayuda+" ...");

            }


                  String [] palabras = iobra.getResumen().split(" ");
                  int contador =0;
                  int cantLineas = 1;
                  while (contador < iobra.getResumen().length() &&
cantLineas < 4){
                        String linea = "";
                        boolean termine = false;
                        while (!termine && linea.length()< 68 &&
i<palabras.length && (linea.length()+palabras[i].length()) < 68){

                              linea=linea.concat(palabras[i]+" ");
                              contador= contador+palabras[i].length()+1;//Le
sumo el espacio
                              if (i == ((palabras.length)-2)){
                                    //no incremento
                                    linea=linea.concat(palabras[i+1]+" ");
```

```
                                        contador=
contador+palabras[i+1].length();//Le sumo el espacio
                                        termine=true;
                                  }
                                  else{
                                        i++;
                                  }


                            }
                            if (termine || cantLineas==3){
                                  aux=aux.concat(linea);
                            }
                            else{
                                  aux=aux.concat(linea)+"\r\n";
                            }
                            cantLineas++;

                      }

                iobra.setResumen(aux);


      }
      public void prepararResumenPelicula(Pelicula ipeli){

            String aux="";
            String resumen = ipeli.getResumen();
            ipeli.setResumen("Resumen: "+resumen);
            resumen = ipeli.getResumen();

            int i=0;

            double largo = resumen.length();


            if ((largo/67) > 5.0){
                  String ayuda = resumen.substring(0, 335);

                  resumen =  new String(ayuda);
                  ipeli.setResumen(ayuda);

            }


            String [] palabras = ipeli.getResumen().split(" ");
            int contador =0;
            int cantLineas = 1;
            while (contador < ipeli.getResumen().length() && cantLineas <
6){
                  String linea = "";
                  boolean termine = false;
                  while (!termine && linea.length()< 68 &&
i<palabras.length && (linea.length()+palabras[i].length()) < 68){
```

```
                        linea=linea.concat(palabras[i]+" ");
                        contador= contador+palabras[i].length()+1;//Le sumo
el espacio
                        if (i == ((palabras.length)-2)){
                              //no incremento
                              linea=linea.concat(palabras[i+1]+" ");
                              contador= contador+palabras[i+1].length();//Le
sumo el espacio
                              termine=true;
                        }
                        else{
                              i++;
                        }


                  }
                  if (termine || cantLineas==5){
                        aux=aux.concat(linea);
                  }
                  else{
                        aux=aux.concat(linea)+"\r\n";
                  }
                  cantLineas++;

            }
            ipeli.setResumen(aux);


      }
      public String  prepararPresentacion(Obra iobra){

            String aux="";

            aux="Obra:        "+iobra.getNombre()+"\r\n"+"Sala:
"+iobra.getSala()+"\r\n"+"Teléfono:
"+iobra.getTelefono()+"\r\n"+"Dirección:
"+iobra.getDireccion()+"\r\n"+"Precio:
"+iobra.getPrecio()+"\r\n"+"Horario:
"+iobra.getHorario()+"\r\n"+"Director:
"+iobra.getDirector()+"\r\n"+"Elenco:
"+iobra.getElenco()+"\r\n"+iobra.getResumen();

            return aux;

      }
      public String prepararPresentacion(Sala isala){
            String aux="";

            aux="Sala: "+isala.getNombre()+"\r\n"+"Dirección:
"+isala.getDireccion()+"\r\n"+"Teléfono:
"+isala.getTelefono()+"\r\n"+"Películas: "+"\r\n"+isala.getPeliculas();

            return aux;
      }
```

```
	public String prepararPresentacion(Pelicula ipeli){
		String aux="";

		aux="Nombre:   "+ipeli.getNombre()+"\r\n"+ipeli.getResumen();

		return aux;
	}
	public LinkedList<Objeto> obtenerSiguientes(LinkedList<Objeto>
lasObras) throws SonlosMismosException {
		// Obtnego el la ultima obra que se encuentre desplejada
		LinkedList<Objeto> res = new LinkedList<Objeto>();
		int i=0;
		Objeto obra  = (Objeto) lasObras.get(i);
		while (!obra.isShowing()){
			i++;
			obra= (Objeto) lasObras.get(i);
		}
		if (lasObras.size() > (i+3)){
			lasObras.get(i).setShowing(false);
			lasObras.get(i+1).setShowing(false);
			res.add(0, lasObras.get(i+2));
			res.add(1, lasObras.get(i+3));
		}
		if (lasObras.size() == (i+3)){
			lasObras.get(i).setShowing(false);
			lasObras.get(i+1).setShowing(false);
			res.add(0, lasObras.get(i+2));

		}
		if (lasObras.size() < (i+3)){

			System.out.println("Son los mismos");
			throw new SonlosMismosException();
		}
		return res;
	}

	public LinkedList<Objeto> obtenerAnteriores(LinkedList<Objeto>
lasObras) throws SonlosMismosException {
		LinkedList<Objeto> res = new LinkedList<Objeto>();
		int largo = lasObras.size();
		int i=largo;
		Objeto obra  = (Objeto) lasObras.get(i-1);
		while (!obra.isShowing()){
			i--;
			obra= (Objeto) lasObras.get(i-1);
		}
		if (i==2){
			throw new SonlosMismosException();
		}

		if (i == largo && !lasObras.get(i-2).isShowing()){


				lasObras.get(i-1).setShowing(false);
```

```
                        res.add(0, lasObras.get(i-3));
                        res.add(1, lasObras.get(i-2));

                }else{
                        lasObras.get(i-1).setShowing(false);
                        lasObras.get(i-2).setShowing(false);
                        res.add(0, lasObras.get(i-4));
                        res.add(1, lasObras.get(i-3));
                }

                return res;
        }

        public List<Obra> obtenerObras(String ipath){
                Auxiliar aux = new Auxiliar();
                LinkedList<Obra> ayuda = new LinkedList<Obra>();

                String path = ipath;

                String res = preparoTexto(path);

                //Ahora que tengo el archivo preparado creo las instancias de
las Obras

                String[] todos = res.split("::");
                int contador=1;
                Obra obra = new Obra(1);
                for (int i =1 ; i < todos.length; i++){

                        switch (contador % 9){
                        case 1:
                                obra.setNombre(todos[i]);
                        break;
                        case 2:
                                obra.setSala(todos[i]);
                        break;
                        case 3:
                                obra.setTelefono(todos[i]);
                                break;
                        case 4:
                                obra.setDireccion(todos[i]);
                                break;
                        case 5:
                                obra.setHorario(todos[i]);
                                break;
                        case 6:
                                obra.setPrecio(todos[i]);
                                break;
                        case 7:
                                obra.setDirector(todos[i]);
                                break;
                        case 8:
                                obra.setElenco(todos[i]);
                                break;
                        case 0:
```

```
                        obra.setResumen(todos[i]);
                        aux.prepararResumen(obra);
                        ayuda.add(obra);
                        obra = new Obra(i);
                  }
                  contador++;
            }


            return reordenarObras(ayuda);

      }

      public List<Pelicula> obtenerPeliculas(String ipath){
            Auxiliar aux = new Auxiliar();
            LinkedList<Pelicula> ayuda = new LinkedList<Pelicula>();

            String path =ipath;
            String res = preparoTexto(path);

            //Ahora que tengo el archivo preparado creo las instancias de
las Obras

            String[] todos = res.split("::");
            int num =1;
            int contador=1;
            Pelicula pelicula = new Pelicula();
            for (int i =1 ; i < todos.length; i++){

                  switch (contador % 2){
                  case 1:
                        pelicula.setNombre(todos[i]);
                  break;
                  case 0:
                        pelicula.setResumen(todos[i]);
                        aux.prepararResumenPelicula(pelicula);
                        pelicula.setNumero(num);
                        num++;
                        ayuda.add(pelicula);
                        pelicula = new Pelicula();
                  }
                  contador++;
            }


            return ayuda;

      }


}
```

Objeto.java

```
package cartelera;

public class Objeto {

	private int numero;
	private String nombre;
	private boolean isShowing;


	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public boolean isShowing() {
		return isShowing;
	}
	public void setShowing(boolean isShowing) {
		this.isShowing = isShowing;
	}

}
```

Obra.java

```
package cartelera;

public class Obra extends Objeto{

	private String sala;
	private String telefono;
	private String direccion;
	private String horario;
	private String precio;
	private String director;
	private String elenco;
	private String resumen;


	public Obra(int inumero){
		super.setNumero(inumero);

	}
```

```
	public String getNombre() {

		return super.getNombre();
	}

	public void setNombre(String nombre) {
		super.setNombre(nombre);
	}

	public String getSala() {
		return sala;
	}

	public void setSala(String sala) {
		this.sala = sala;
	}

	public String getTelefono() {
		return telefono;
	}

	public void setTelefono(String telefono) {
		this.telefono = telefono;
	}

	public String getDireccion() {
		return direccion;
	}

	public void setDireccion(String direccion) {
		this.direccion = direccion;
	}

	public String getHorario() {
		return horario;
	}

	public void setHorario(String horario) {
		this.horario = horario;
	}

	public String getPrecio() {
		return precio;
	}

	public void setPrecio(String precio) {
		this.precio = precio;
	}

	public String getDirector() {
		return director;
	}

	public void setDirector(String director) {
```

```
		this.director = director;
	}

	public String getElenco() {
		return elenco;
	}

	public void setElenco(String elenco) {
		this.elenco = elenco;
	}

	public String getResumen() {
		return resumen;
	}

	public void setResumen(String resumen) {
		this.resumen = resumen;
	}

	public int getNumero() {
		return super.getNumero();
	}

	public void setNumero(int numero) {
		super.setNumero(numero);
	}

	public boolean isShowing() {
		return super.isShowing();
	}

	public void setShowing(boolean isShowing) {
		super.setShowing(isShowing);
	}


}
```

Película.java

```
package cartelera;

public class Pelicula extends Objeto {



	private String resumen;



	public int getNumero() {
		return super.getNumero();
```

```
	}
	public void setNumero(int numero) {
		super.setNumero(numero);
	}
	public String getNombre() {
		return super.getNombre();
	}
	public void setNombre(String nombre) {
		super.setNombre(nombre);
	}
	public String getResumen() {
		return resumen;
	}
	public void setResumen(String resumen) {
		this.resumen = resumen;
	}
	public boolean isShowing() {
		return super.isShowing();
	}
	public void setShowing(boolean isShowing) {
		super.setShowing(isShowing);
	}
}
```

Sala.java

```
package cartelera;

public class Sala extends Objeto{


	private String direccion;
	private String telefono;
	private String peliculas;
	private int cantidadPeliculas;



	public boolean isShowing() {
		return super.isShowing();
	}

	public void setShowing(boolean isShowing) {
		super.setShowing(isShowing);
	}

	public Sala (int num){
		super.setNumero(num);
	}

	public int getNumero() {
		return super.getNumero();
```

```
	}
	public void setNumero(int numero) {
		this.setNumero(numero);
	}
	public String getNombre() {
		return super.getNombre();
	}
	public void setNombre(String nombre) {
		super.setNombre(nombre);
	}
	public String getDireccion() {
		return direccion;
	}
	public void setDireccion(String direccion) {
		this.direccion = direccion;
	}
	public String getTelefono() {
		return telefono;
	}
	public void setTelefono(String telefono) {
		this.telefono = telefono;
	}
	public String getPeliculas() {
		return peliculas;
	}
	public void setPeliculas(String peliculas) {
		this.peliculas = peliculas;
	}
	public int getCantidadPeliculas() {
		return cantidadPeliculas;
	}
	public void setCantidadPeliculas(int cantidadPeliculas) {
		this.cantidadPeliculas = cantidadPeliculas;
	}


}
```

SonlosMismosException.java

```
package excepcion;

public class SonlosMismosException extends Exception{


	private int key;

	public SonlosMismosException() {
		super();
	}
	public int getKey(){
```

```
            return this.key;
          }
          public void setKey(int num){
            this.key = num;
          }
}
```