Seminario de Lenguajes JAVA

Seminario de Lenguajes

Temas

- Componentes avanzadas Swing
  - Listas
  - Tablas

Conexión de Vistas y Modelo

Para el diseño de aplicaciones con interfaces sofisticados se utiliza el patrón de Arquitectura de software: **Modelo Vista Controlador (MVC).** La lógica de un interfaz de usuario cambia con más frecuencia que los datos que tenemos almacenados y la lógica de negocio. Si realizamos un diseño complicado, es decir, una mezcla de componentes de interfaz y de negocio, entonces cuando necesitemos cambiar la interfaz, tendremos que modificar trabajosamente los componentes de negocio:

mayor trabajo y más riesgo de error.

MVC fomenta un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

Modelo: datos y reglas de negocio

Vista: muestra la información del modelo al usuario

Controlador: gestiona las entradas del usuario

MVC

- El patrón de diseño MVC (Model View Controller, Modelo Vista Controlador) provee un mecanismo que posibilita separar los datos (el modelo) de la forma en que estos serán visualizados (la vista).

- Algunos Componentes de Swing que permiten visualizar información implementan este patrón de diseño

- Ejemplo : un componente **JList** puede mostrar objetos que se encuentran en:
  a) un array,
  b) un Vector
  c) una Collection
  d) un archivo o
  e) una tabla de una base de datos.

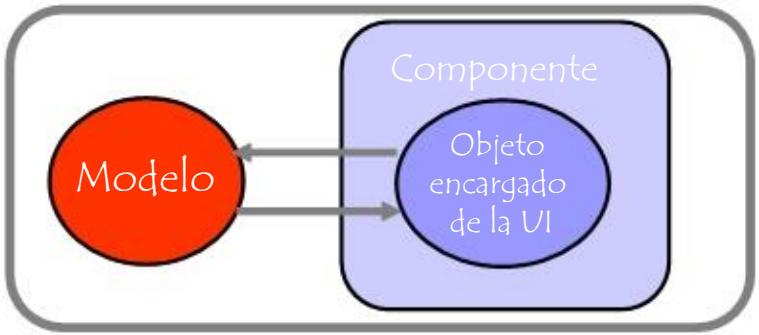
El componente (la vista) está totalmente independizado del modelo (o soporte) de los datos

Componentes Swing en profundidad

Cual es la arquitectura y cómo trabajar con componentes de GUI más sofisticadas, como lo son **JList** y **JTable**.

- La arquitectura de las componentes Swing responde a una versión especializada del patrón MVC. Está basada en 2 objetos: un objeto Modelo y un objeto encargado del display o apariencia y del manejo de los eventos, que representan la Vista + el Controlador en el patrón MVC.

Una Componente de GUI de Swing

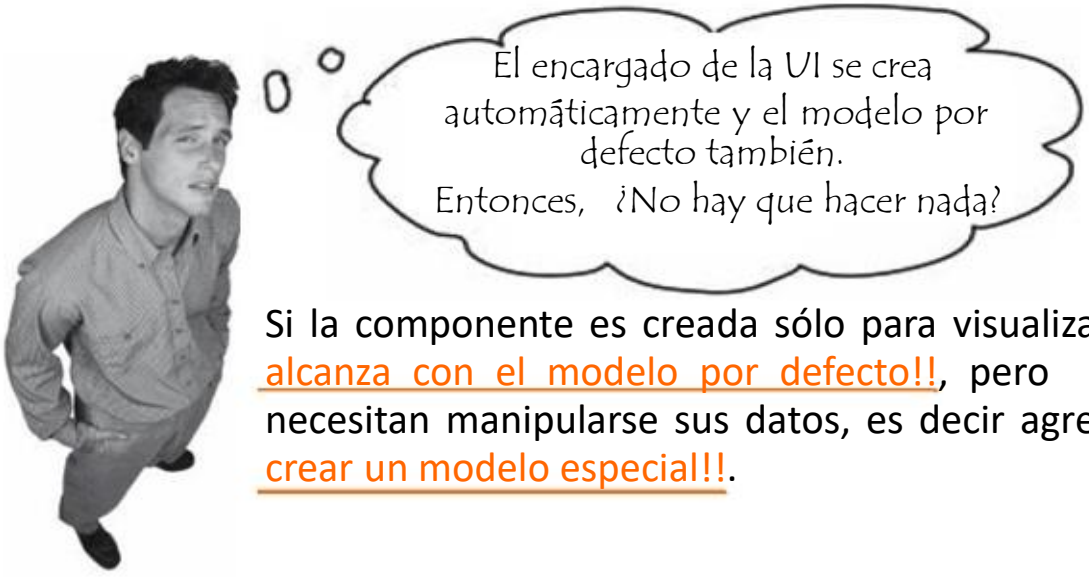


El Modelo es tratado como un objeto separado

La Vista y el Controlador están acoplados en un único objeto encargado de la UI

Componentes Swing en profundidad

- Cuando se crea una componente Swing, se crea automáticamente su objeto encargado de la UI, no hay que preocuparse por eso!!.
- El objeto Modelo es responsable de almacenar el estado del componente. Si una aplicación, no provee un modelo explícito para una componente, Swing le crea un modelo por defecto!!!. Swing provee un modelo por defecto para cada componente visual. Por ejemplo: JTable -> DefaultTableModel, . . .



Si la componente es creada sólo para visualizar y permitir seleccionar datos, en general alcanza con el modelo por defecto!!, pero … si la componente es más compleja, y necesitan manipularse sus datos, es decir agregar/eliminarse datos del modelo, se debe crear un modelo especial!!.

Componentes Swing – JList

¿Cómo creo una lista de selección con un modelo por defecto?

La clase JList, tiene varios constructores:

- JList() - JList
- JList(ListModel arg0) - JList

Si inicializamos la lista con estos constructores, podemos agregar y borrar elementos.

- JList(Object[] arg0) - JList
- JList(Vector<?> arg0) - JList

Si inicializamos la lista con un arreglo o un vector, el constructor crea un modelo por defecto, que es INMUTABLE!!!

Para trabajar con listas, que sólo muestren opciones fijas:

NO es necesario crear Modelos. La componente provee uno por defecto.

Componentes Swing – JList

Creación de una lista de selección con un modelo por defecto.

Lista.java | *ListaPlanetas.java

```
public class ListaPlanetas extends JFrame {
    private JList lista;
    private JButton imprimir = new JButton("imrimir selección");

    public void init() {
        String items[] = {"Mercurio", "Tierra", "Saturno", "Júpiter"};
        lista = new JList(items);
        lista.setVisibleRowCount(4);
        Container cp = this.getContentPane();

        imprimir.addActionListener(new MiActionListener());

        cp.setLayout(new FlowLayout());
        cp.add(new JScrollPane(lista));
        cp.add(imprimir);
    }

    class MiActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e){
            System.out.println(lista.getSelectedValue());
        }
    }

    public static void main(String[] args) {
}
```

Se crea el arreglo con los valores

El constructor crea un modelo por defecto con los datos del arreglo

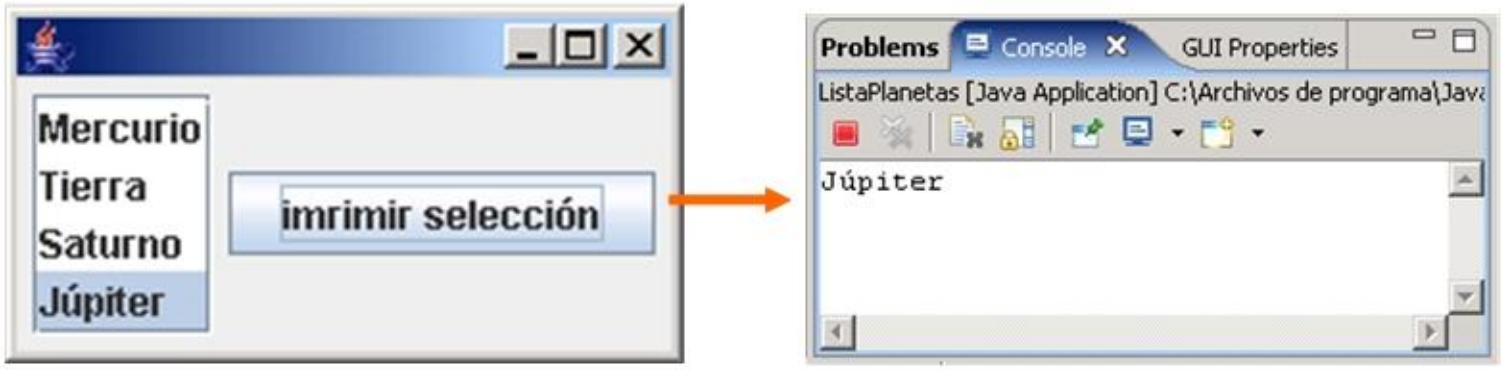
Las listas (JList) no poseen barras de desplazamiento, por lo tanto se las debe colocar adentro de un JScrollPane, quien la hace desplazarse.

Si ejecutamos una main() como este …

```
public static void main(String[] args) {
    ListaPlanetas app = new ListaPlanetas(); app.init();
    app.pack();
    app.setVisible(true);
}
```

Componentes Swing – JList

¿Cómo funciona la aplicación que implementamos?

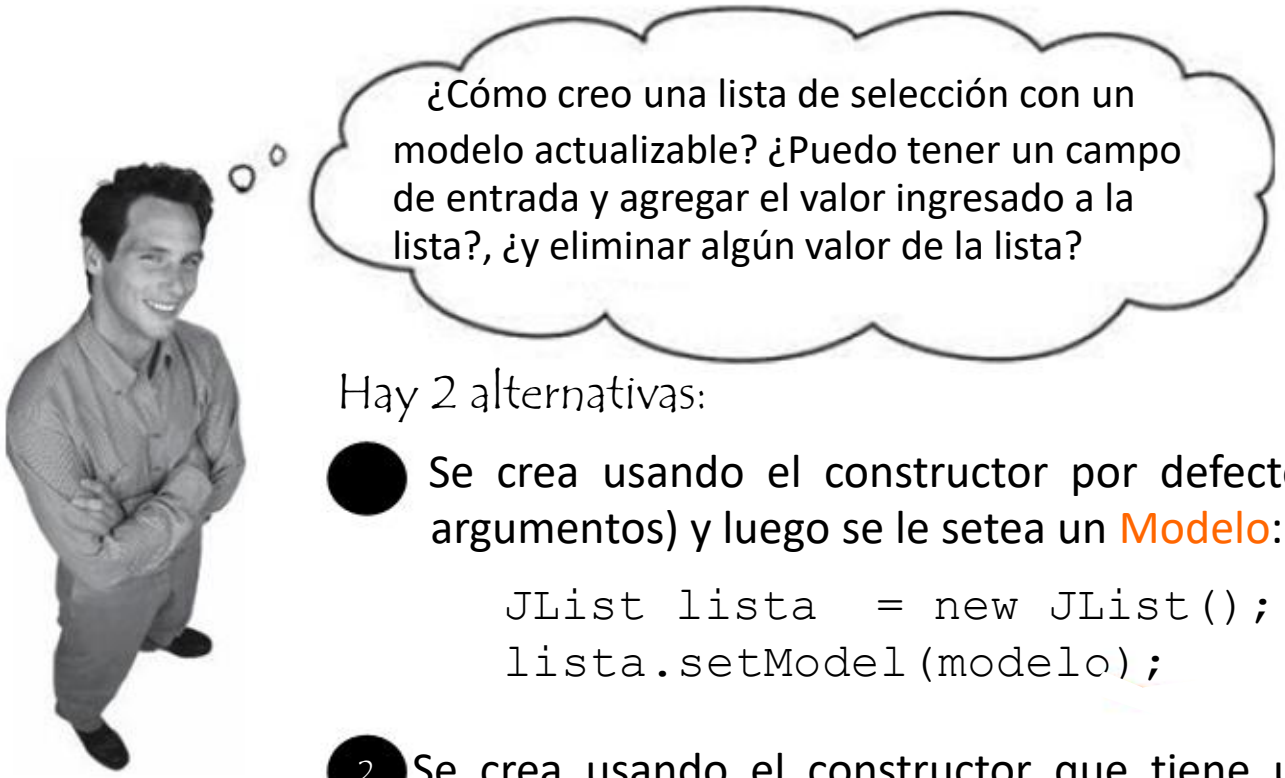


Sólo se pueden visualizar datos fijos, y seleccionar items de la lista.

No se puede

eliminar,
reemplazar,
agregar valores al Modelo por defecto creado automáticamente

Componentes Swing – JList



Hay 2 alternativas:

1. Se crea usando el constructor por defecto (el que no tiene argumentos) y luego se le setea un Modelo:

```
JList lista  = new JList();
lista.setModel(modelo);
```

2. Se crea usando el constructor que tiene un Modelo como argumento:

```
JList lista  = new JList(modelo);
```

Componentes Swing – JList

*Lista.java ✕ ListaPlanetas.java

```
import java.awt.Container;

public class Lista extends JFrame{
    private JList lista= new JList();
    private DefaultListModel modelo = new DefaultListModel();
    private JButton agregar = new JButton("Agregar");
    private JButton quitar = new JButton("Eliminar");
    private JTextField editor = new JTextField("");

    public void init() {
        String items[] = {"Mercurio", "Tierra", "Saturno", "Júpiter"};
        for (int i = 0; i < items.length; ++i)
            modelo.addElement(items[i]);
        lista.setVisibleRowCount(4);
        lista.setModel(modelo);

        Container cp = this.getContentPane();
        agregar.addActionListener(new MiActionListenerAgrega());
        quitar.addActionListener(new MiActionListenerBorra());
        cp.setLayout(new FlowLayout());
        cp.add(new JScrollPane(lista));
        editor.setColumns(15);
        cp.add(editor);
        cp.add(agregar);
        cp.add(quitar);
    }
    class MiActionListenerAgrega implements ActionListener{
    class MiActionListenerBorra implements ActionListener{
    public static void main(String[] args) {
}
```

Se instancia un objeto DefaultListModel, para poder agregar y eliminar elementos de la lista.

Se crea el arreglo

Se itera el arreglo y se guardan los valores en el Modelo

Se liga el Modelo a la componente JList !!

Registramos listeners en los botones

Si ejecutamos una main() como este …

```
public static void main(String[] args) {
    ListaPlanetas app = new ListaPlanetas(); app.init();
    app.pack();
    app.setVisible(true);
}
```

Componentes Swing – JList

¿Cómo funciona ahora la aplicación?

Al insertar en el Modelo de la componentes Swing, se actualiza la Vista. Además como no pueden visualizarse todos los valores, aparecen las barras de desplazamiento.

Elimina el item seleccionado

Al eliminar del Modelo el item seleccionado de la JList, se actualiza la Vista. Además como pueden visualizarse todos los valores, desaparecen las barras de desplazamiento.

Componentes Swing – JTable

¿Cómo se manejan las tablas con Swing?

. Cuando se crea un JTable se pueden especificar los valores de las celdas con alguno de los siguientes constructores:

- JTable(TableModel dm) - JTable

Si inicializamos la lista con estos constructores, podemos agregar y borrar elementos.

- JTable(Object[][] rowData, Object[] columnNames) - JTable
- JTable(Vector rowData, Vector columnNames) - JTable

Si usamos estos constructores crea un modelo por defecto, que es INMUTABLE!!!

Para trabajar con tablas, que sólo muestren filas de datos con formato String, NO es necesario crear ó instanciar Modelos.

Para crear tablas dónde se puedan agregar/borrar filas, alcanza con instanciar un DefaultTableModel (clase que implementa la interface TableModel).

Componentes Swing - JTable

¿Cómo creo una tabla con un modelo actualizable? ¿Cómo hago para agregar y eliminar filas de la tabla?

Hay 2 alternativas:

1. Se crea usando el constructor por defecto y luego se le setea un Modelo:

```
JTable tabla  = new JTable();
tabla.setModel(modelo);
```

modelo es de tipo DefautTableModel o subclase

2. Se crea usando el constructor que tiene un Modelo como argumento:

```
JTable tabla  = new JTable(modelo);
```

Componentes Swing – JTable

Creando un JTable con objeto DefaultTableModel para poder modificar datos de la tabla

ReservaFrameTextual.java

```
package tablas;

import java.awt.BorderLayout;

public class ReservaFrameTextual {
    private JFrame f = new JFrame();
    private JTable tabla = new JTable();
    private JButton borrar = new JButton("Borrar");
    private JButton agregar = new JButton("Agregar");
    private JPanel panelBotones = new JPanel();

    private Object[] titulos = {"Titular","Nacionalidad","Origen","Destino","Fecha","# Adul
    private DefaultTableModel modelo = new DefaultTableModel(titulos, 0);

    public ReservaFrameTextual(){
      Container contentPane = f.getContentPane();
      contentPane.setLayout(new BorderLayout());
      agregar.addActionListener(new AgregarListener());
      borrar.addActionListener(new BorrarListener());
      panelBotones.add(agregar);
      panelBotones.add(borrar);
      tabla.setModel(modelo);
      contentPane.add(panelBotones, BorderLayout.NORTH);
      contentPane.add(new JScrollPane(tabla), BorderLayout.CENTER);
      f.pack();
      f.setVisible(true);
    }

    class BorrarListener implements ActionListener{
    class AgregarListener implements ActionListener{

    public static void main(String[] args) {
      ReservaFrameTextual ven = new ReservaFrameTextual();
    }
```

Se usa el constructor sin argumentos!!

Se instancia un objeto DefaultTableModel , para poder agregar y eliminar elementos de la lista y se lo liga a la tabla.

Se liga el Modelo a la componente JTable !!

Si ejecutamos con el main()

Componentes Swing – JTable

Esta ventana contiene una tabla que fue creada con una instancia de la clase DefaultTableModel. La Tabla puede visualizar cualquier tipo de objeto en sus celdas, pero siempre lo muestra como un objetos de tipo String.

¿Puedo ver a la columna Nacionalidad como una imagen o a la columna Bodega como un objeto checkbox?

Si!!, pero hay que trabajar un poco más con el modelo.

Componentes Swing - JTable

Los datos de la tabla son mostrados como String porque usamos como modelo una instancia de la clase DefaultTableModel. Esta clase tiene una manera por defecto de mostrar los datos y es en formato String. Para poder cambiar esto, debemos crear una clase que extienda a DefaultTableModel y sobreescribir un método para indicar el tipo de dato de cada celda.

Se extiende el DefaultTableModel

Se define un constructor que recibe como parámetro los títulos y datos.

Este método se debe sobreescribir para indcar, de que tipo es cada columna. De lo contrario siempre se muestran como String

ModeloReservas.java

```
package tablas;

import javax.swing.table.DefaultTableModel;


public class ModeloReservas extends DefaultTableModel {

    public ModeloReservas(final Object[][] datos, final String[] titulos){
      super(datos, titulos);
    }
    public Class getColumnClass(final int column) {
        return this.getValueAt(0, column).getClass();
    }
}
```

Componentes Swing – JTable

Creando un JTable con objeto customizado de DefaultTableModel para poder modificar la visibilidad de los datos.

ReservaFrame.java

```
public class ReservaFrame {
  private JFrame f = new JFrame();
  private JTable tabla = new JTable();
  private JButton borrar = new JButton("Borrar");
  private JButton agregar = new JButton("Agregar");
  private JPanel panelBotones = new JPanel();

  private String[] titulos={"Titular","Nacionalidad","Origen","Destino","Fecha","# Adultos","# Niños","Bodega","Cate
  private Object[][] datos={
   {"Martinez, Maria", new ImageIcon("Argentina.gif"), "Buenos Aires", "Colonia", new Date(), 2,2,true,"Turista"),
   {"Berima, Julia", new ImageIcon("Canada.gif"), "Buenos Aires", "Colonia", new Date(), 2,2,true,"Turista"),
   {"Hernandez, Juan", new ImageIcon("Argentina.gif"), "Buenos Aires", "Colonia", new Date(), 2,2,true,"Business"),
  };
  public ReservaFrame(){
    Container contentPane = f.getContentPane();
    contentPane.setLayout(new BorderLayout());
    tabla.setModel(new ModeloReservas(datos, titulos));
    agregar.addActionListener(new AgregarListener());
    borrar.addActionListener(new BorrarListener());
    panelBotones.add(agregar);
    panelBotones.add(borrar);
    contentPane.add(panelBotones, BorderLayout.NORTH);
    contentPane.add(new JScrollPane(tabla), BorderLayout.CENTER);
    f.pack();
    f.setVisible(true);
  }
  public static void main(String[] args) {
     ReservaFrame vent = new ReservaFrame();
  }
  class BorrarListener implements ActionListener{
  class AgregarListener implements ActionListener{
}
```

Se usa el constructor sin argumentos!!

Se crea una instancia del modelo customizado para visualizar cada datos según su tipo y se lo liga a la tabla.

Componentes Swing – JTable

La tabla tiene la siguiente apariencia. Ahora cada datos se visualiza diferente, según la clase a la que pertenece:

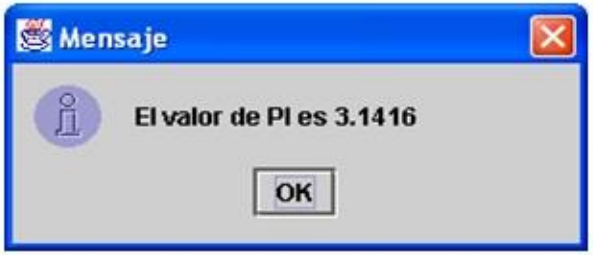
Las tablas pueden ser provistas de más características, se puede configurar tamaño fijo o variable para las columnas, se puede:

- Permitir o prohibir la edición de una o más columnas,
- Se puede fijar una columna y permitir el scroll del resto, etc.

Las analizadas son algunas de las funcionalidades más utilizadas con los objetos JTable.

Componentes Swing – Ventanas de Diálogo

Swing también provee una clase que automatiza muchas de las actividades que un programador haría para crear ventanas de diálogo: JOptionPane. Esta clase tiene muchos métodos de clase que permiten crear diálogos con íconos, mensajes, campos de entrada y botones.




```
JOptionPane.showMessageDialog(
    null,                          //padre
    "El valor de PI es "+pi,       //mensaje
    "Mensaje",                     // título
    JOptionPane.INFORMATION_MESSAGE);  //hay 5 tipos de mensajes
}
```


```
JOptionPane.showConfirmDialog(
   boton,                          //padre
   "¿Salva antes de salir?",       //mensaje
   "Mensaje",                      //título
    JOptionPane.YES_NO_CANCEL_OPTION,    //tipo de opción
    JOptionPane.WARNING_MESSAGE,         // tipo de mensaje
    new ImageIcon("duke.gif"));          //ícono
```

Todos los diálogos creados con JOptionPane, son modales.