

SESION N12: INTERFACES GRÁFICAS

Agenda Semana 11

- Definición de Interfaz Gráfica
- Manejo de Eventos
- Principales componentes
- Desarrollo de aplicaciones simples con componentes gráficos
- Preguntas

Definición

- El lenguaje de programación Java proporciona distintos paquetes con bibliotecas de clases que implementan los elementos gráficos y posibilitan la creación de interfaces gráficas de una forma sencilla desde la versión 1.0 con AWT (Abstract Windowing Toolkit).
- En el curso usaremos los componentes Swing y su modelo de delegación de eventos junto a sus diversos elementos gráficos como por ejemplo botones, cuadros de texto, etiquetas, tablas, entre otras

Pasos Básicos

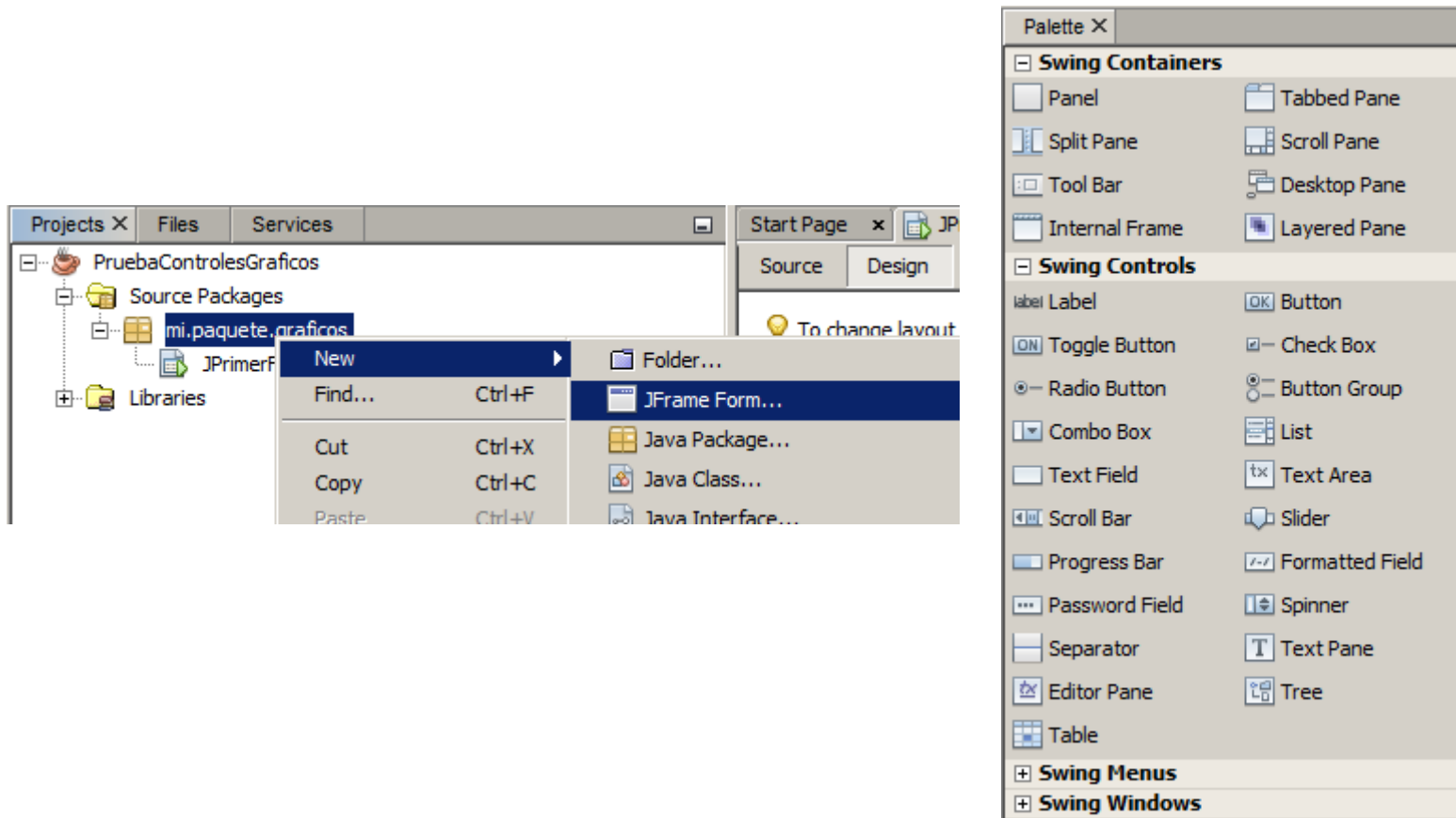
1. El diseño y composición de la apariencia de la interfaz gráfica de la aplicación.- Normalmente implica la elección de la ventana principal(contenedor) en la que se va a incluir el resto de los elementos gráficos (componentes).
2. Escribir el código que crea los componentes y su apariencia en la interfaz, modificando atributos a través de las características propias de presentación, color, tamaño, etc.
3. Escribir código que proporciona el comportamiento de dicha interfaz(evento), por ejemplo, cuando el usuario hace click sobre un botón.

Manejo de Eventos

- Los eventos se generan producto de la interacción del usuario con alguno de los elementos que conforman la interfaz gráfica.
- La gestión de estos eventos, que implica la ejecución de un método o métodos en respuesta a la acción del usuario, permite conseguir el comportamiento dinámico que da la interactividad a la aplicación
- Los eventos tienen una jerarquía de clases cuya raíz es la clase `java.util.EventObject`

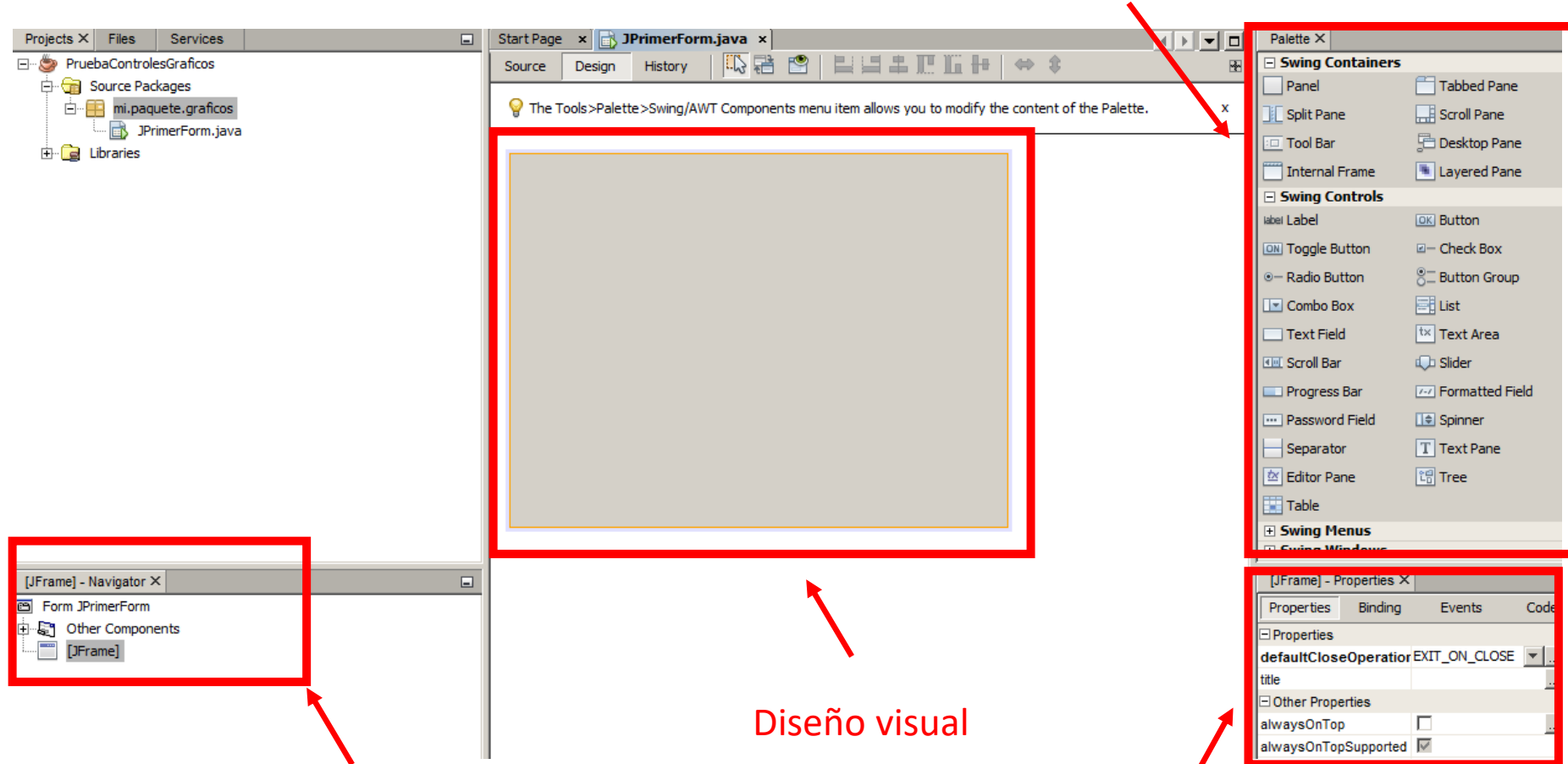
Clase JFrame

- Se emplea para crear la ventana principal de una aplicación, es una subclase de `java.awt.Frame`
- Es una ventana que incluye los controles habituales de cambio de tamaño y cierre.



Clase JFrame

Paleta de controles Swing



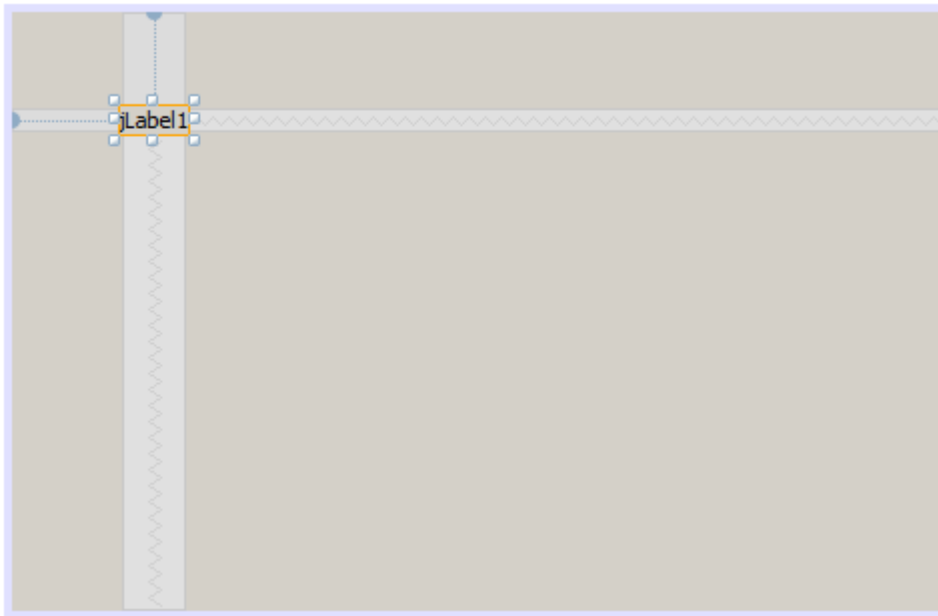
Diseño visual

Nombre de Controles

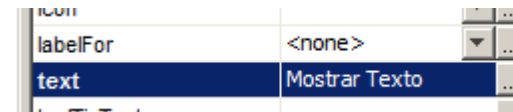
Propiedades de Contrioles

Clase JLabel

- Se emplea para implementar una etiqueta que puede contener una cadena de texto, un icono o ambos. Solo muestra su contenido no es seleccionable ni editable por el usuario

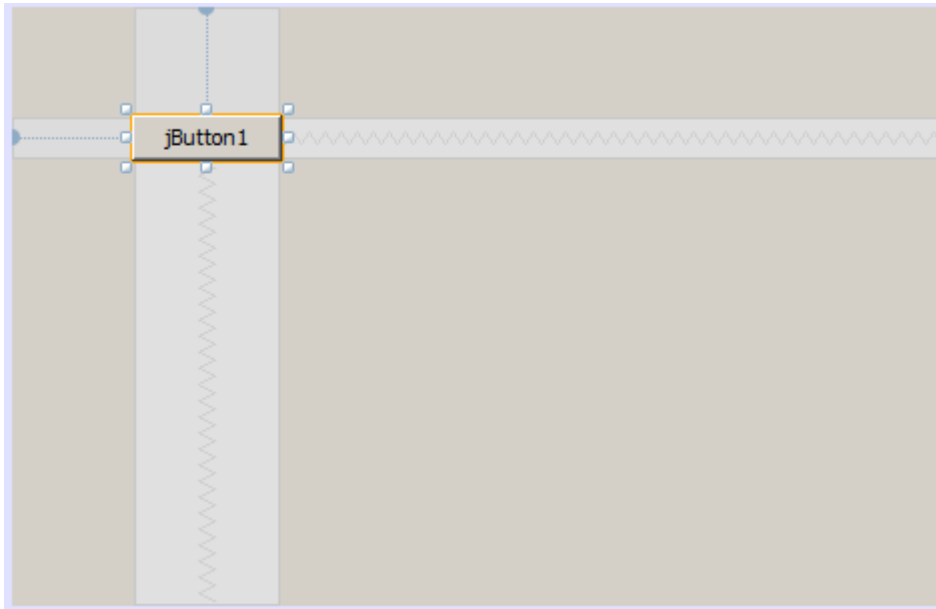


- Propiedad que cambia el texto que se muestra

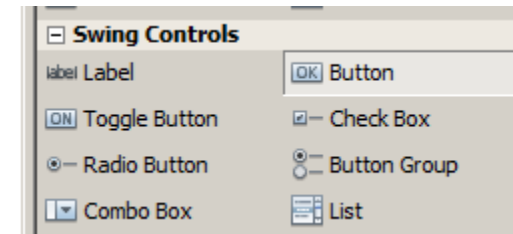


Clase JButton

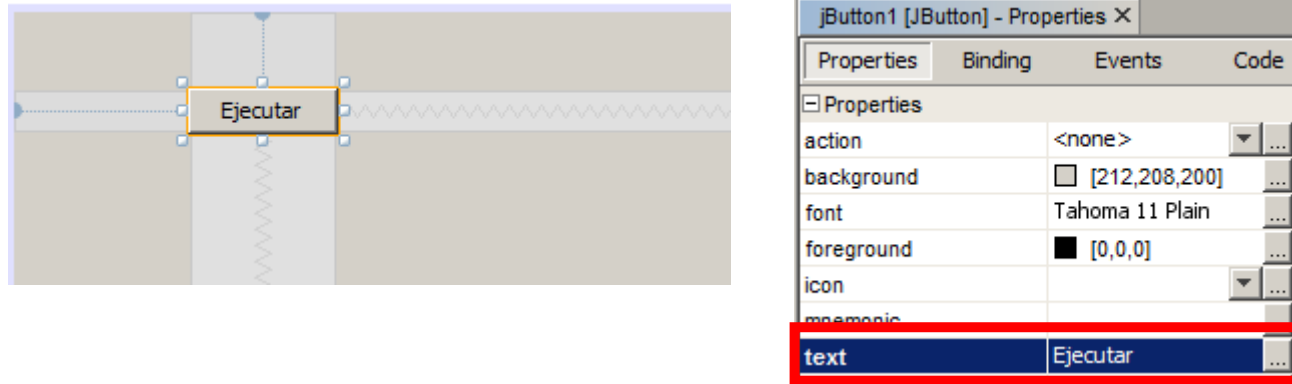
- Se emplea para implementar la forma mas habitual de botón grafico de interacción que sirve para ejecutar una acción haciendo click sobre él. También se puede activar mediante una combinación de teclado.



jButton2



Clase JButton



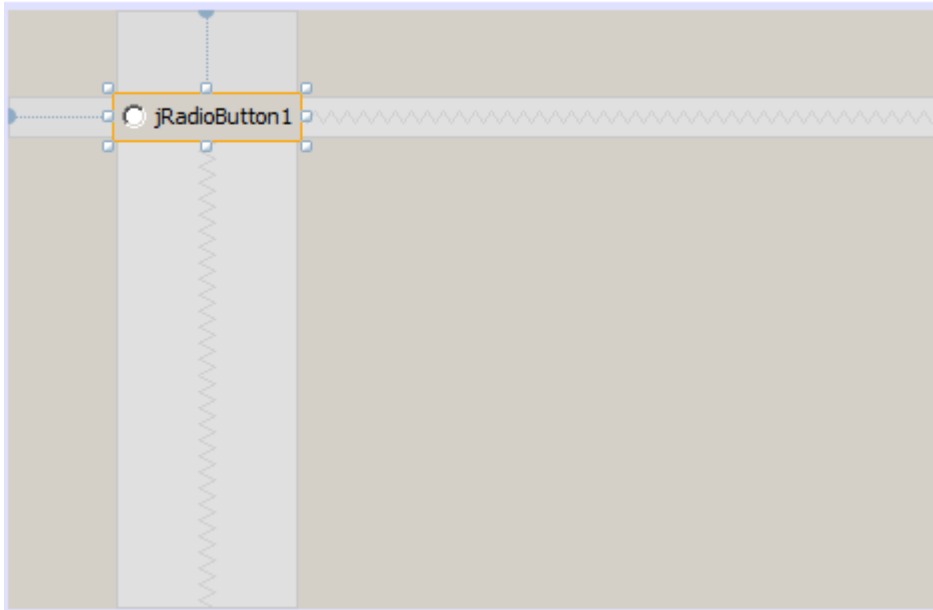
- Al hacer doble click sobre el control, se muestra lo siguiente

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

- Ahí agregamos el código que se ejecutará cada vez que hacemos click al botón

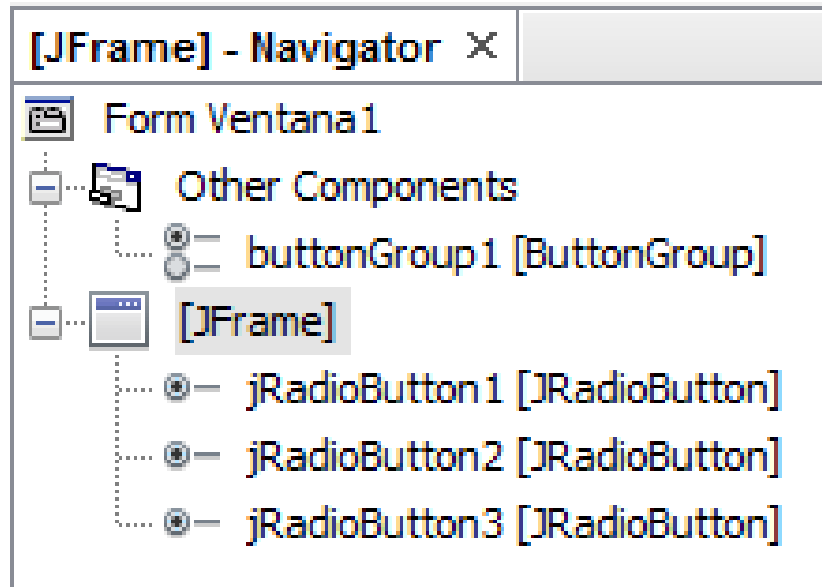
Clase JRadioButton

- Se emplea para implementar una selección de opciones y se caracteriza porque solo uno de ellos puede estar seleccionado, es decir son mutuamente excluyentes.



Clase JRadioButton

- Para que solo se seleccione uno de los JRadioBurrton, se tiene que añadir un objeto del tipo **ButtonGroup** al formulario. ¡Atención! Este objeto es invisible, y no se verá en el formulario, sin embargo, se puede ver en el Inspector, en la parte de “Otros Componentes”



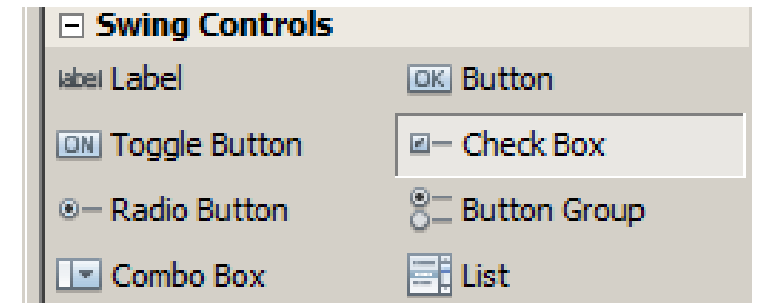
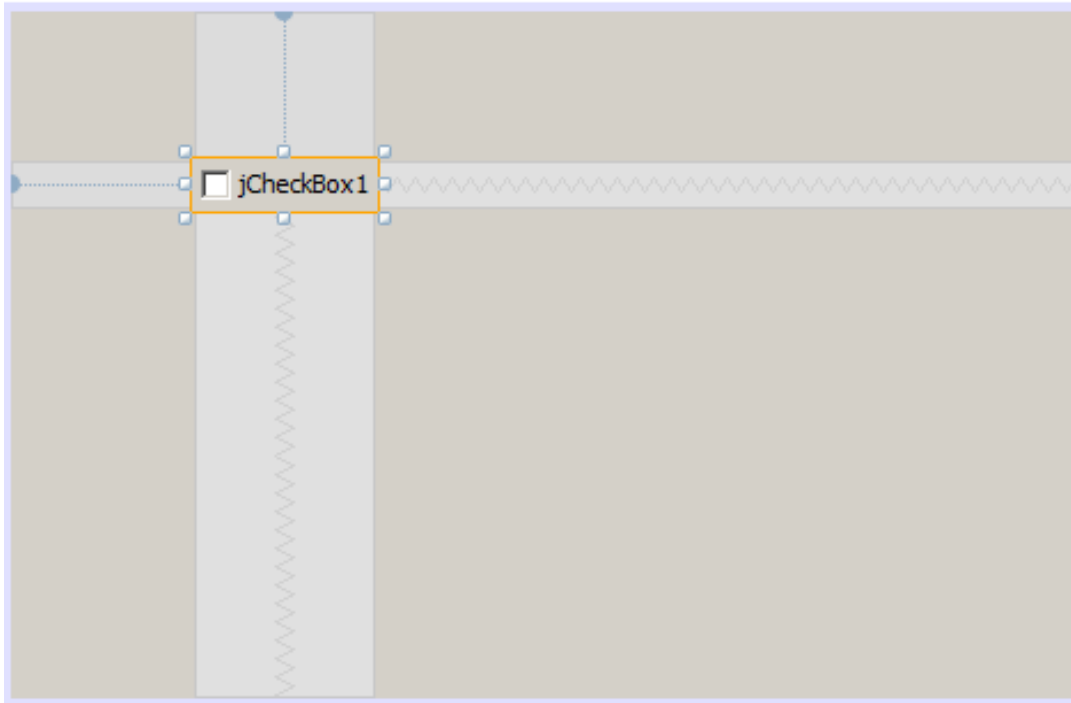
Clase JRadioButton

- Para lograr que un JRadioButton pertenezca al grupo creado se tiene que cambiar la característica “buttonGroup” y así con cada uno de los JRadioButton que se requiera

[-] Properties		
action	<none>	...
background	<input type="checkbox"/> [240,240,240]	...
model	<default>	...
buttonGroup	<none>	...
font	<none>	...
foreground	buttonGroup1	...

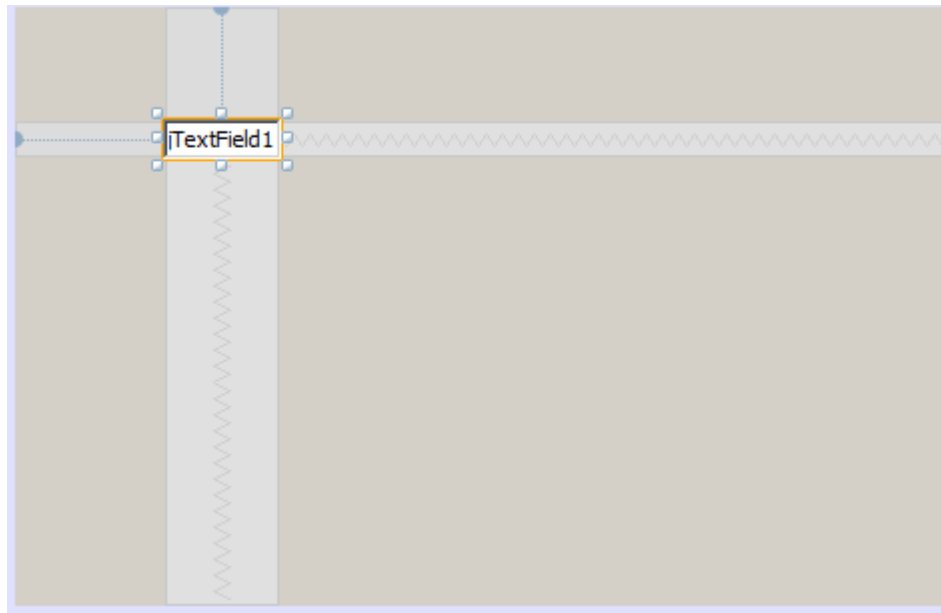
Clase JCheckBox

- Se emplea para implementar una casilla de verificación y tiene dos estados posibles seleccionada o no seleccionada. Generalmente es usada para que el usuario decida una opción o no.

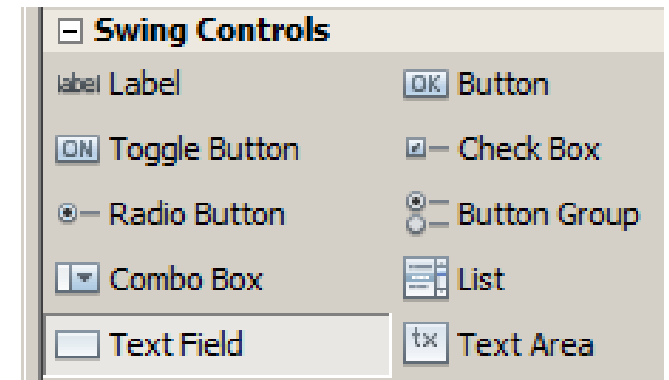


Clase JTextField

- Se emplea para mostrar y editar una única línea de texto, así el usuario podrá ingresar datos de entrada breves en forma de texto como por ejemplo nombre o un dato numérico, este ultimo con el cambio de dato para poder ser trabajado.

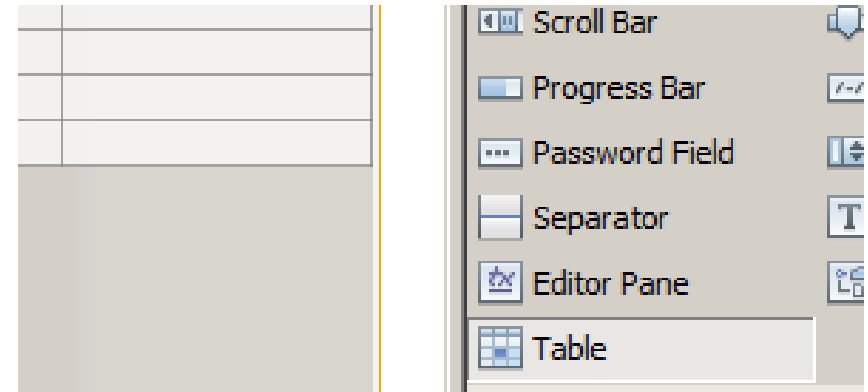
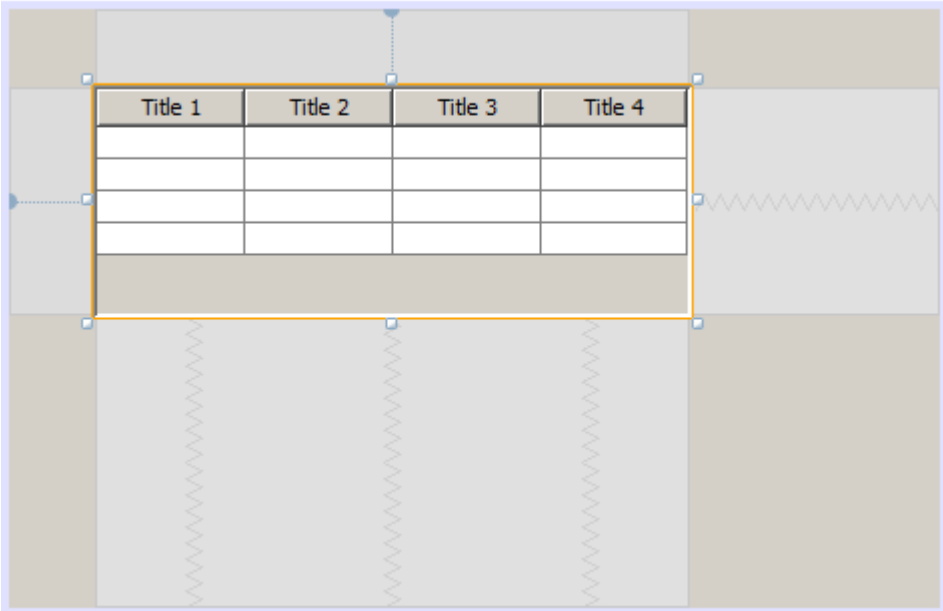


jTextField2



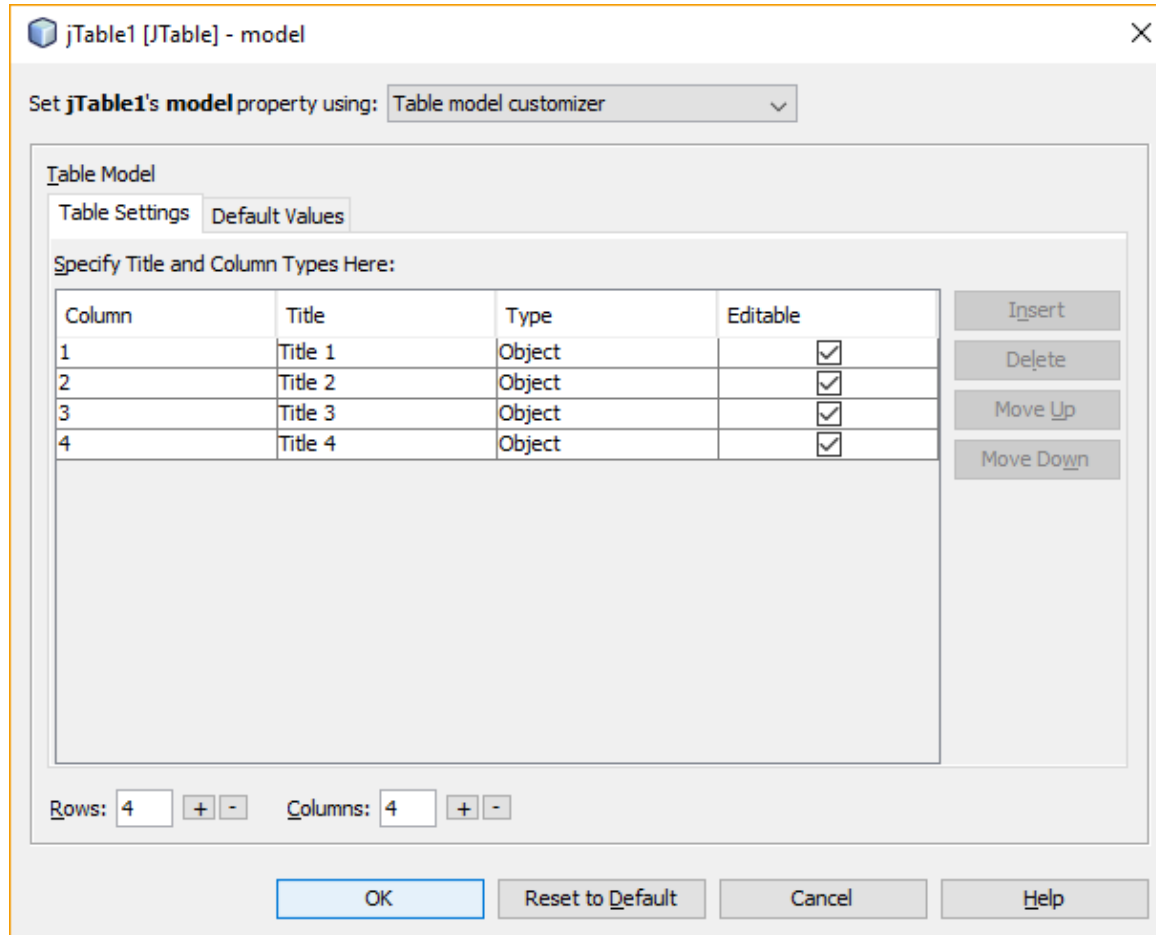
Clase JTable


- Se emplea para implementar un tabla de datos, el cual puede ser llenado en modo ejecución o puede ser carado mediante una conexión a base de datos



Clase JTable

- Para “customizar” el número de filas y columnas, ingresar a la propiedad “model”



foreground	 [0,0,0]	...
model	[TableModel]	...
toolTipText		...

Clase JTable

- Para “customizar” el ingreso de columnas es necesario utilizar una instancia de la clase DefaultTableModel
- El ingreso de datos se realiza utilizando el método addRow del objeto anterior

```
DefaultTableModel model;

/**
 * Creates new form JPrimeraForm
 */
public JPrimeraForm() {
    initComponents();
    model = new DefaultTableModel();
    model.addColumn("Nombre");
    model.addColumn("Sexo");
    model.addColumn("Habilidad 1");
    this.jTable1.setModel(model);
}
```

```
String[] datos = new String[3];
datos[0]=jTextField1.getText();
jTextField1.setText(null);
datos[1]="Femenino";
if(jRadioButton1.isSelected())
{
    datos[1]="Masculino";
}
jRadioButton1.setSelected(true);
if(jCheckBox1.isSelected())
{
    datos[2]="Habil";
}
jCheckBox1.setSelected(false);
model.addRow(datos);
```

Gracias, ¿Preguntas?

