Técnicas de programación avanzada

Curso 2023-2024

Tema 4.1 Interfaces gráficas

Dra. Nieves Cubo Mateo (@Nicuma3)
Alejandro Alonso Puig (@mundobot)

Tema 1: Objetos y memoria.

1.1 Características básicas del lenguaje. Primer programa. Compilación y Ejecución. IDE.

1.2 Sentencias de control. Secuencia, selección e iteración.

1.3 Abstracción. Clases, objetos, métodos y atributos.

1.4 Sobrecarga de métodos y encapsulamiento.

Tema 2. Otros conceptos fundamentales de la Programación Orientada a Objetos

2.1 Herencia. Interfaces y clases abstractas. Agregación.

2.2 Polimorfismo.

2.3 Gestión de Excepciones.

2.4 Genericidad y plantillas.

2.5 Utilidades. Entrada y Salida.

2.6 Anotaciones.

Tema 3. Patrones de Diseño.

3.1 Concepto de Patrones de Diseño.

3.2 Patrones de creación.

3.3 Patrones estructurales.

3.4 Patrones de comportamiento.

Tema 4. Programación de Interfaces.

4.1 Interfaces Gráficas de Usuario.

4.2 Gestión de eventos.

Tema 5. Temas Avanzados.

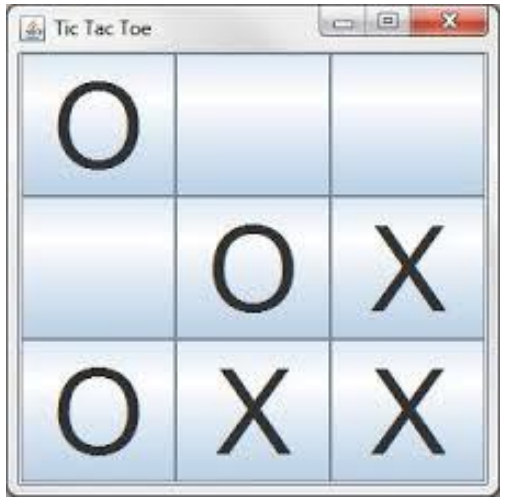
5.1 Concurrencia.

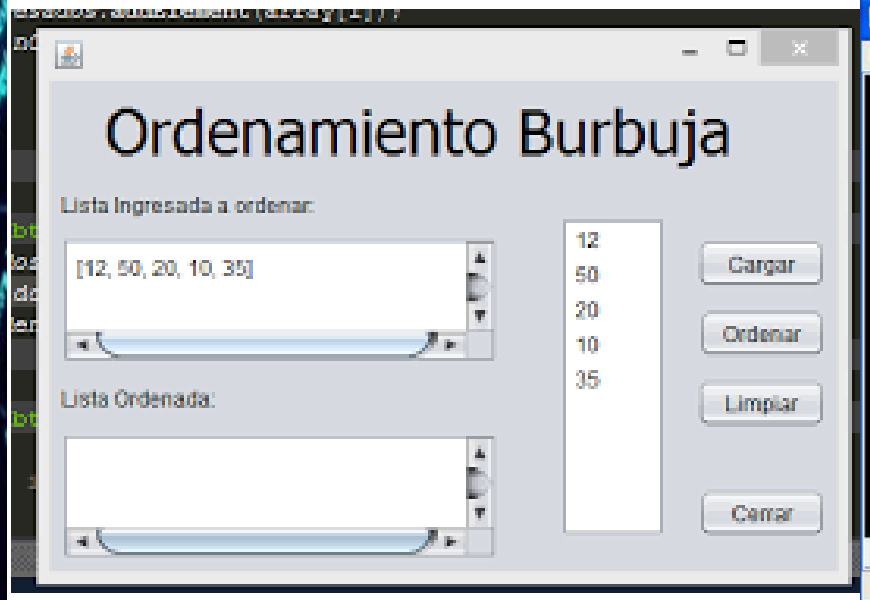
5.2 Inversión de Control. Definición y ejemplos. Inyección de dependencias.

5.3 Expresiones avanzadas del lenguaje.

4.1 Interfaces gráficas de Usuario

Ejemplos





4.1 Interfaces gráficas de Usuario

Algunas de las principales opciones para trabajar con gráficos en Java:

- **Swing**: Swing es una biblioteca gráfica incluida en la plataforma Java Standard Edition (Java SE) que permite crear interfaces de usuario (UI) basadas en ventanas. Swing proporciona componentes gráficos como botones, etiquetas, cuadros de texto, tablas y más. Puedes personalizar la apariencia de tu aplicación Java utilizando Swing y crear interfaces de usuario atractivas.

- **JavaFX**: JavaFX es una plataforma moderna de desarrollo de aplicaciones gráficas para Java que se ha convertido en la sucesora de Swing. Ofrece una forma más rica y versátil de crear aplicaciones de escritorio con gráficos 2D y 3D. JavaFX incluye una amplia variedad de componentes, soporte para animaciones y una mejor integración con tecnologías multimedia.

- **AWT** (Abstract Window Toolkit): AWT es una biblioteca gráfica más antigua que Swing y es parte integral de Java. Proporciona una forma de crear interfaces de usuario y trabajar con gráficos. AWT se utiliza a menudo en combinación con Swing para un mayor control sobre los componentes de la interfaz de usuario.

4.1 Interfaces gráficas de Usuario

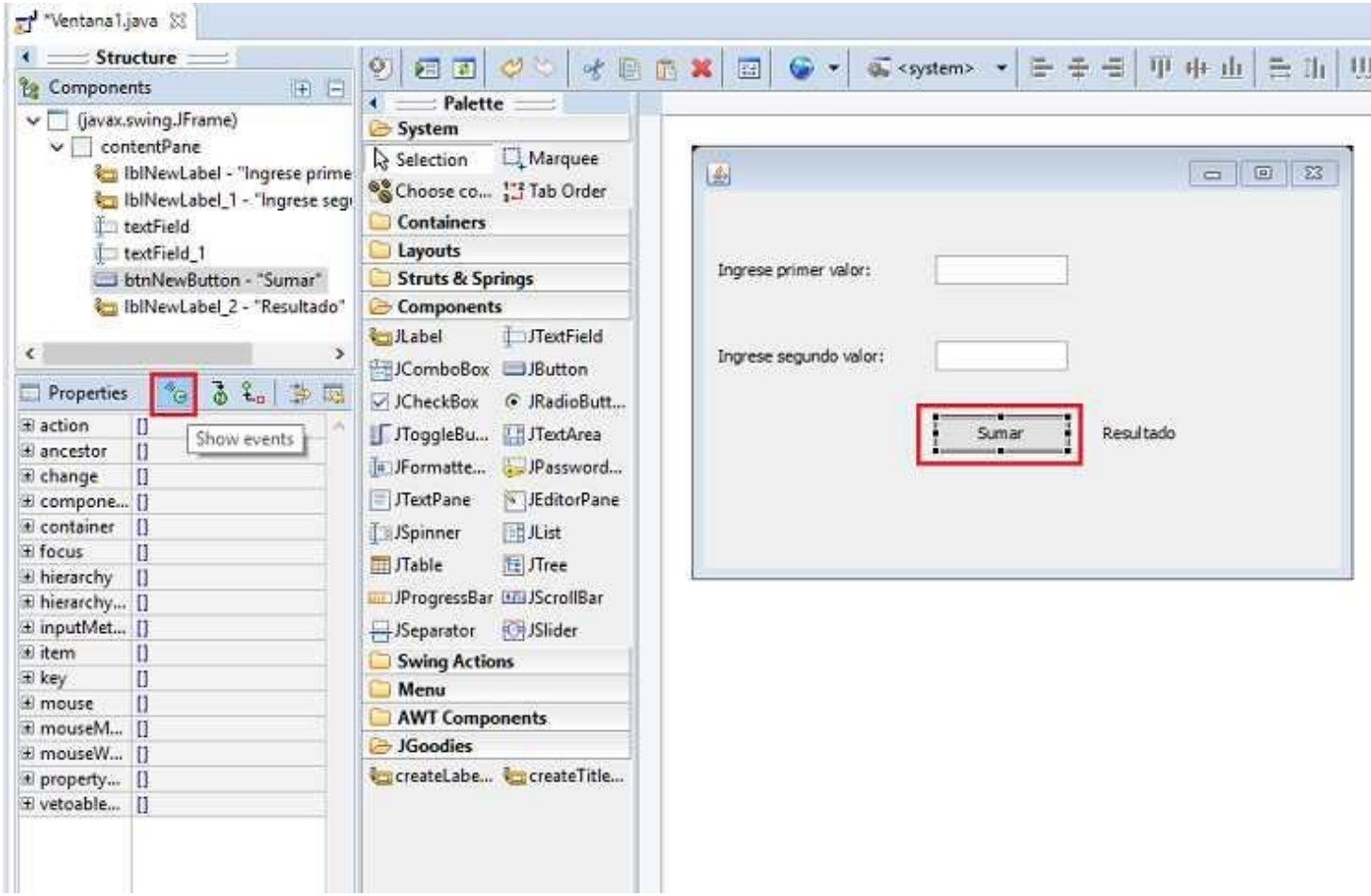
Java Swing

Java Swing es una librería gráfica ligera que incluye un amplio conjunto de widgets.

https://docs.oracle.com/javase/tutorial/uiswing/components/index.html

Podemos crear las GUI introduciendo el código a mano, o a través de IDEs gráficos (compatibles o incluso integrados en el IDE), como:

- WindowBuilder
- NetBeans IDE (Apache)



4.1 Interfaces gráficas de Usuario

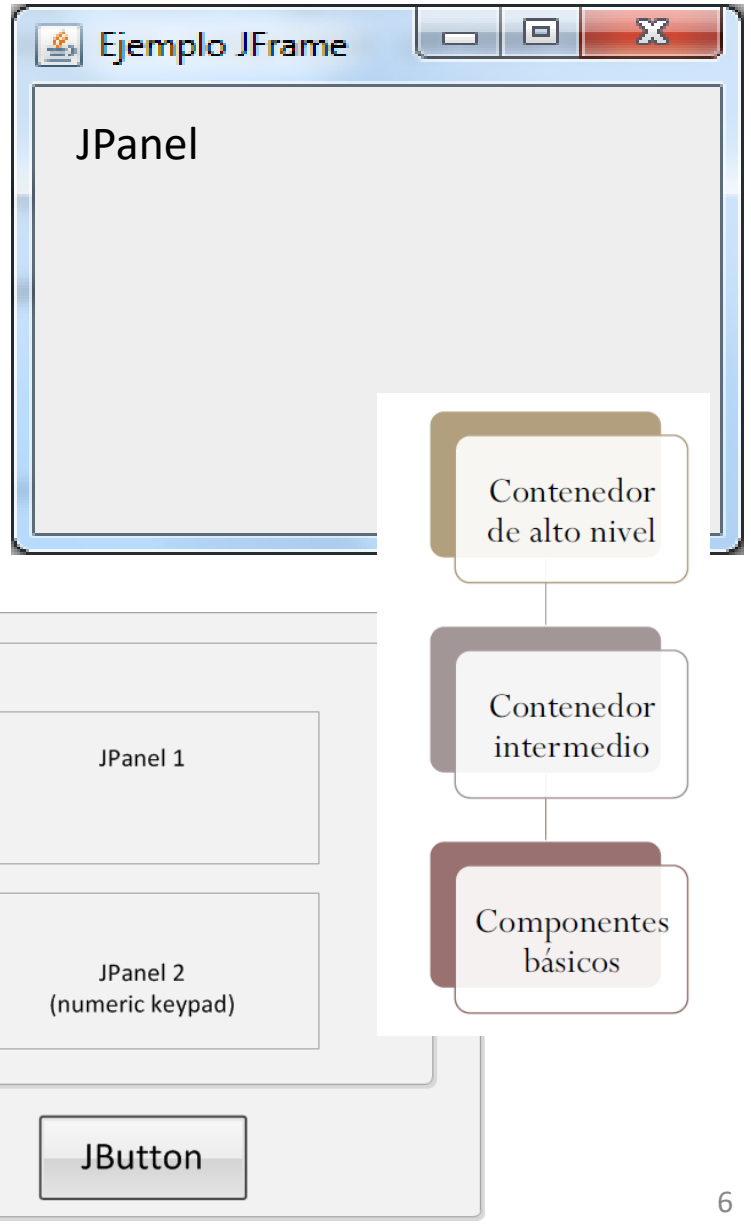
Componentes básicos (Swing)

UNIVERSIDAD NEBRIJA

Contenedores:

Los contenedores son componentes que se utilizan para organizar y agrupar otros componentes (como botones, etiquetas, campos de texto, etc.). Algunos de los más comunes:

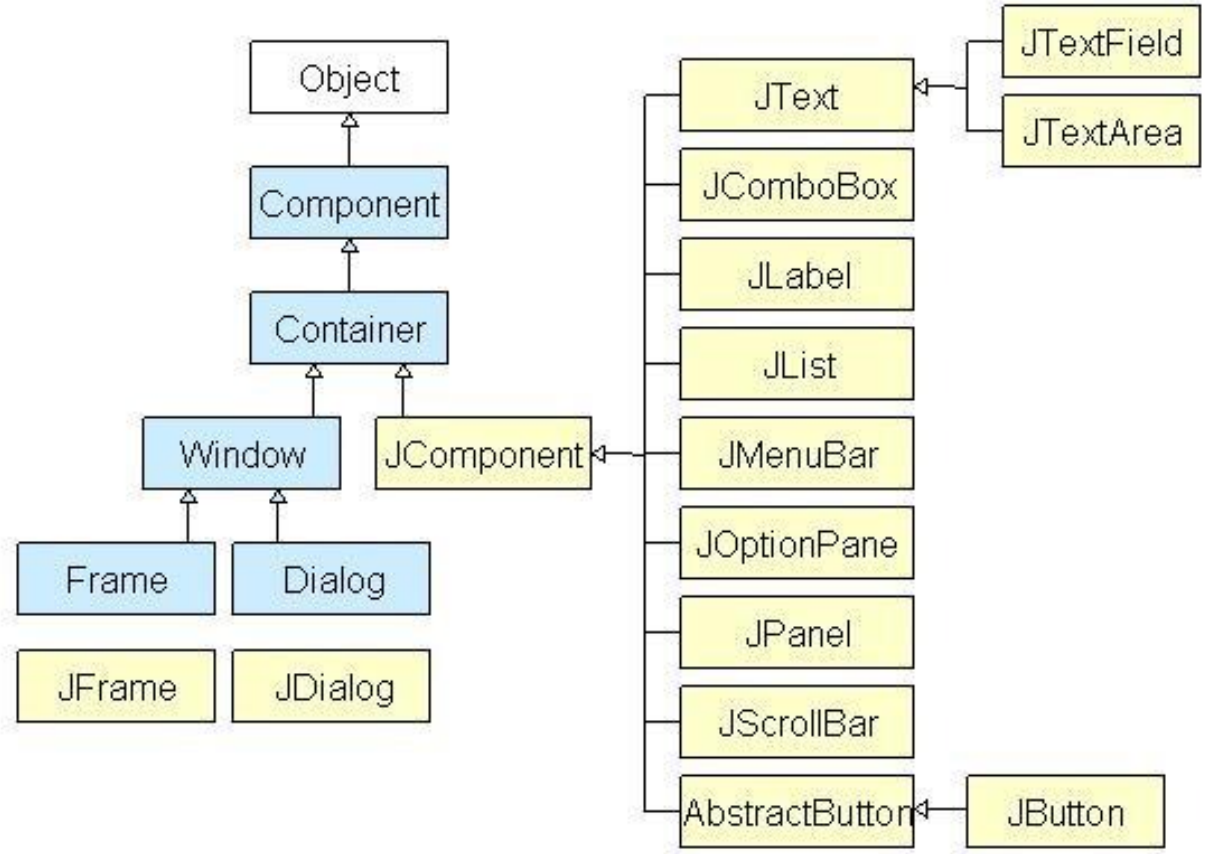
- **JFrame** es la ventana principal del programa, debe haber sólo uno. Tiene icono, y sale en la barra de tareas.
- **JPanel**: es un contenedor genérico que se utiliza para agrupar y organizar otros componentes en una ventana. Puedes anidar paneles dentro de otros paneles para crear una estructura de diseño más compleja. Los paneles son muy versátiles y se utilizan comúnmente para crear áreas de contenido personalizadas.
- **JDialog** es un contenedor especializado que se utiliza para crear ventanas de diálogo o ventanas emergentes en una aplicación. Si se configura como "modal" no deja tocar el resto de las ventanas hasta cerrar/terminar esta.
- **JScrollPane**, **JTabbedPane**, **JSplitPane**, **JScrollPane**, **JLayeredPane**, **JDesktopPane** y **JInternalFrame**



Componentes:

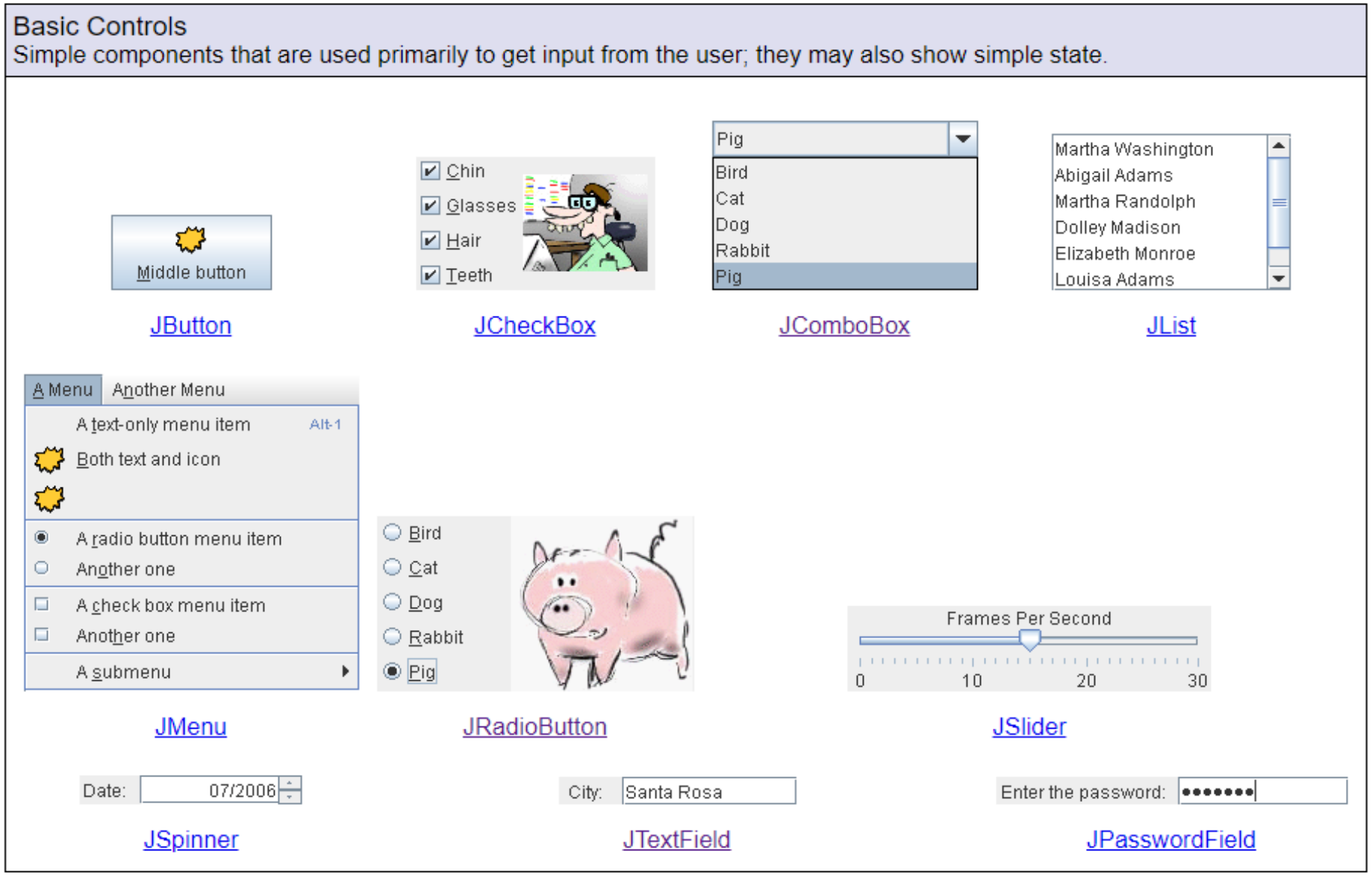
Son objetos gráficos que interactúan con los usuarios y proporcionan funcionalidad específica.

AbstractButton, BasicInternalFrameTitlePane, Box, Box.Filler, JColorChooser, JComboBox, JFileChooser, JInternalFrame, JInternalFrame.JDesktopIcon, JLabel, JLayer, JLayeredPane, JList, JMenuBar, JOptionPane, JPanel, JPopupMenu, JProgressBar, JRootPane, JScrollBar, JScrollPane, JSeparator, JSlider, JSpinner, JSplitPane, JTabbedPane, JTable, JTableHeader, JTextComponent, JToolBar, JToolTip, JTree, JViewport



4.1 Interfaces gráficas de Usuario

JComponents



https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

4.1 Interfaces gráficas de Usuario

JComponents

1. **AbstractButton**
    1. **JButton**: El botón típico de cualquier aplicación gráfica
    2. **JMenuItem**
        1. **JMenu**: Elemento de un menú
        2. **JCheckBoxMenuItem**: Elemento de un menú que puede ser seleccionado
        3. **JRadioButtonMenuItem**: Elemento de un menú que forma parte de un conjunto del que sólo puede haber seleccionado uno
    3. **JToggleButton**: Botón de dos estados
        1. **JCheckBox**: Elemento que puede estar seleccionado o no
        2. **JRadioButton**: Se usa junto con **ButtonGroup**, y sólo puede haber uno seleccionado
2. **JColorChooser**: Panel de selección de color
3. **JComboBox**: Lista desplegable de la que se puede elegir un elemento
4. **JDesktopPane**: Contenedor de frames internos
5. **JFileChooser**: Panel de selección de fichero

4.1 Interfaces gráficas de Usuario

JComponents

6. **JInternalFrame**: Frame que puede colocarse dentro de otro contenedor
7. **JLabel**: Etiqueta donde se pueden poner texto e imágenes
8. **JLayeredPane**: Panel donde los objetos pueden estar a distinta profundidad
9. **JList**: Lista de elementos de la que podemos elegir uno o más elementos
10. **JMenuBar**: Barra superior del programa que contiene **JMenu's**
11. **JOptionPane**: Permite mostrar un diálogo (junto con **JDialog**)
12. **JPanel**: Contenedor genérico sobre el que se añaden otros componentes
13. **JPopUpMenu**: Menú emergente que aparece al hacer click con el botón derecho del ratón
14. **JProgressBar**: Barra de progreso típica que se usa cuando una operación lleva cierto tiempo
15. **JScrollBar**: Barra de desplazamiento
16. **JScrollPane**: Panel contenedor con dos barras de desplazamiento
17. **JSeparator**: Línea separadora (por ejemplo, dentro de un menú)

4.1 Interfaces gráficas de Usuario

JComponents

18. **JSlider**: Barra para seleccionar valores gráficamente
19. **JSpinner**: Permite seleccionar valores de una lista pulsando arriba y abajo
20. **JSplitPane**: Panel dividido en dos partes
21. **JTabbedPane**: Contenedor múltiple en el que seleccionamos un conjunto de componentes a través de pestañas
22. **JTable**: Componente para mostrar información de forma tabular
23. **JTextComponent**
    1. **JEditorPane**: Facilita la creación de un editor
    2. **JTextArea**: Permite la inserción de texto en múltiples líneas
    3. **JTextField**: Igual que el anterior, pero sólo en una línea
       1. **JFormattedField**: Permite introducir texto con formato
       2. **JPasswordField**: El texto se oculta con el símbolo que prefiramos
24. **JToolBar**: Contenedor de iconos que suele aparecer en la parte superior
25. **JToolTip**: Texto emergente que aparece al situar el ratón sobre un control
26. **JTree**: Permite mostrar información jerarquizada en forma de árbol

Ejemplo de programa: ASOCIACIÓN
como **Objeto** dentro de un ***método***

UNIVERSIDAD NEBRIJA

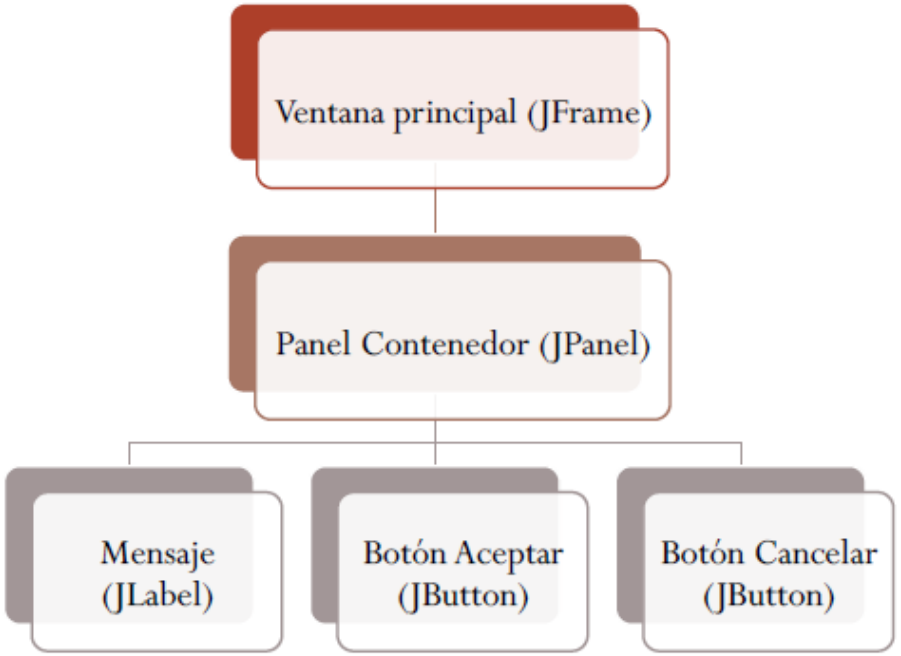
```
import javax.swing.*;

public class Gui2 {

    private static void createAndShowGUI() {

        // Crea y configura ventana
        JFrame frame = new JFrame("Ejemplo de JFrame con JPanel");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setBounds(100, 100, 500, 200); // (pos_pantalla(x,y), size(x,y))
        // frame.setSize(500, 300); //Similar set size and default position. pixels

        // Crea y configura JPanel
        JPanel contentPane = new JPanel();
        frame.add(contentPane);
        contentPane.setLayout(null);

        // Crea y configura botón
        JButton initButton = new JButton("Presionar");
        initButton.setBounds(200, 50, 100, 50); // (pos_pantalla(x,y), size(x,y))
        contentPane.add(initButton);

        frame.setVisible(true); // Muestra ventana
    }
}
```

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

Importante para poder usar coordenadas (null)

4.1 Interfaces gráficas de Usuario

Creación de la interfaz



Pasos básicos en la construcción de una interfaz

1. Crear una nueva clase para nuestra ventana (o directamente instanciar JFrame)
2. Configurar los parámetros de la ventana
3. Crear los componentes de nuestra interfaz
4. Crear uno o varios contenedores intermedios
5. Asociar los componentes al contenedor
6. Asociar el contenedor a la ventana
7. Hacer visible la ventana

4.1 Interfaces gráficas de Usuario

Creación de la interfaz

Ejemplo de programa: HERENCIA
como ***Clase hija*** de ***JFrame***

UNIVERSIDAD NEBRIJA

```
import javax.swing.*;

// 1) Creamos la clase ventana
class ExFrameClass extends JFrame {

  public ExFrameClass() {

    // 2) Configuramos los parámetros de la ventana
    setTitle("Ejemplo de JFrame con JPanel");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 500, 200); //(pos_pantalla(x,y), size(x,y))

    // 3) Crear los componentes
    JButton initButton = new JButton("Presionar");
    initButton.setBounds(200, 50, 100, 50); //(pos_pantalla(x,y), size(x,y))

    // 4) Crear un contenedor
    JPanel contentPane = new JPanel();

    // 5) Asociar los componentes al contenedor
    contentPane.add(initButton);

    // 6) Asociar el contenedor a la ventana
    setContentPane(contentPane);
    contentPane.setLayout(null);

    // 7) Hacer visible la ventana
    setVisible(true);   //Hace visible todo
  }
}
```

```
public class Gui3 {

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

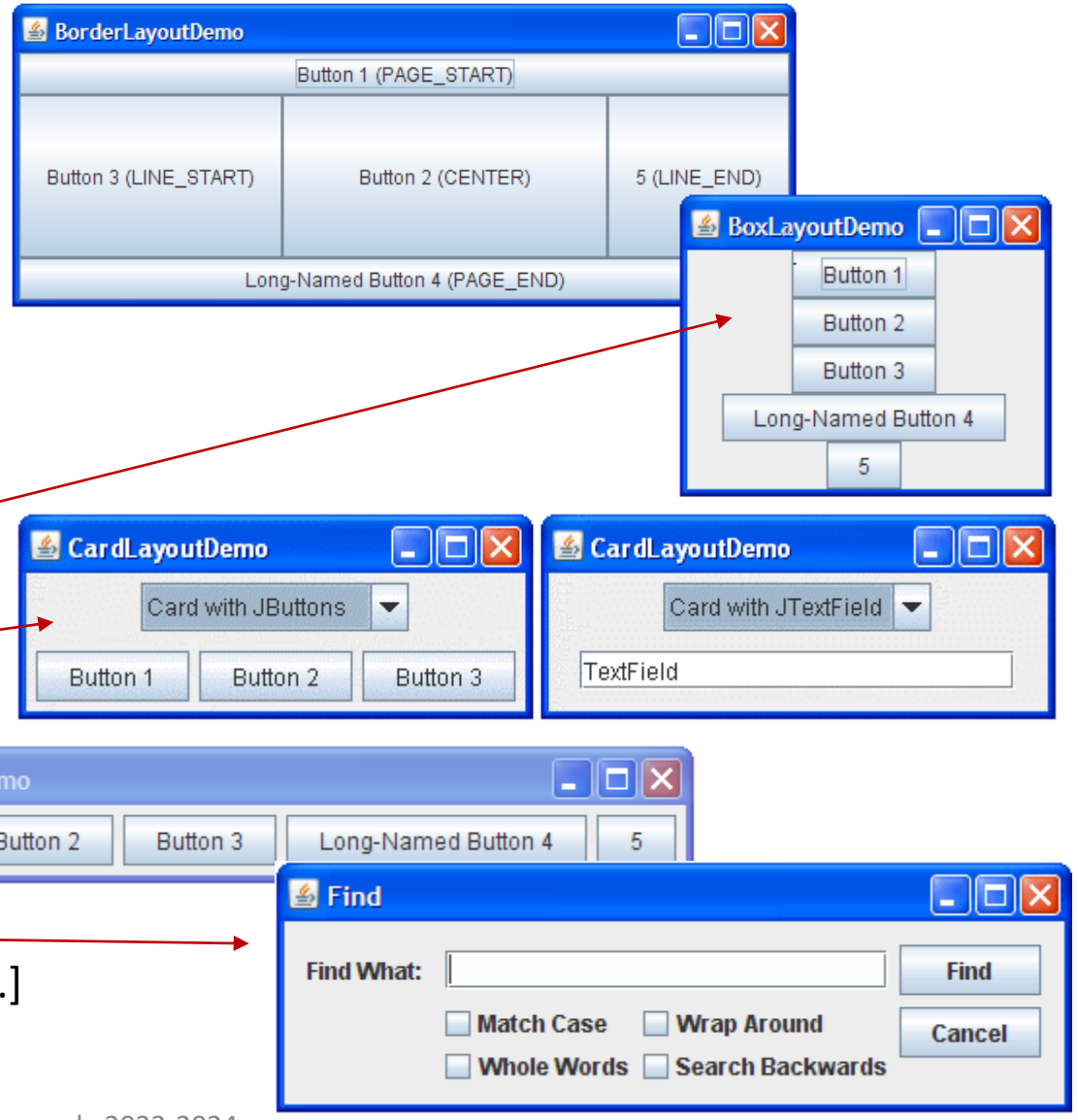
4.1 Interfaces gráficas de Usuario

Jcomponents – Cómo colocarlos automáticamente (Java Layout Manager)

Si no desea colocar cada elemento manualmente mediante coordenadas, también se pueden emplear gestores de capas para realizar colocaciones automáticas. Existen diferentes gestores de Layouts de Swing/AWT:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

[...]



https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

4.1 Interfaces gráficas de Usuario

Java Layouts

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridLayout
- GridBagLayout
- GroupLayout
- SpringLayout

Se pueden usar “a mano”

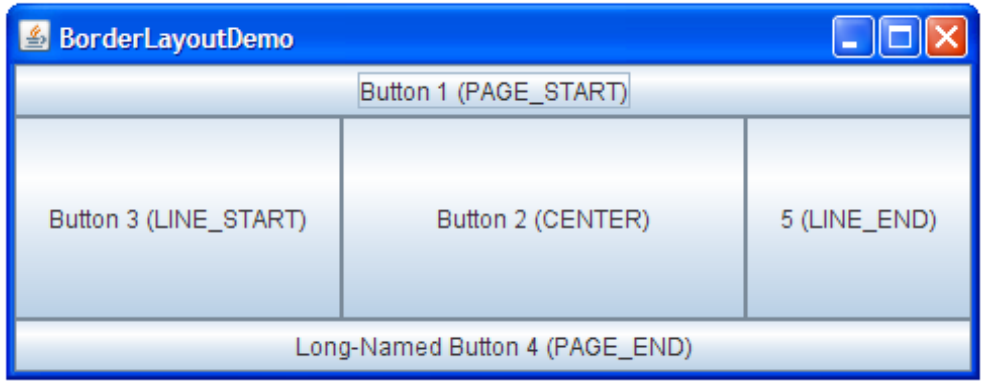
A medio camino

Mejor usar con un IDE

https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

BorderLayout

- Los paneles de contenido de los JFrame, por defecto, están inicializados con este tipo de Layout
- Permite colocar componentes (simples o contenedores) en cinco posiciones: Arriba, Abajo, Izquierda, Derecha y Centro
- Las posiciones se indican al añadir el componente al contenedor (mediante el método add que ya hemos visto)
  - Arriba: PAGE_START o NORTH
  - Abajo: PAGE_END o SOUTH
  - Izquierda: LINE_START o WEST
  - Derecha: LINE_END o EAST
  - Centro: CENTER



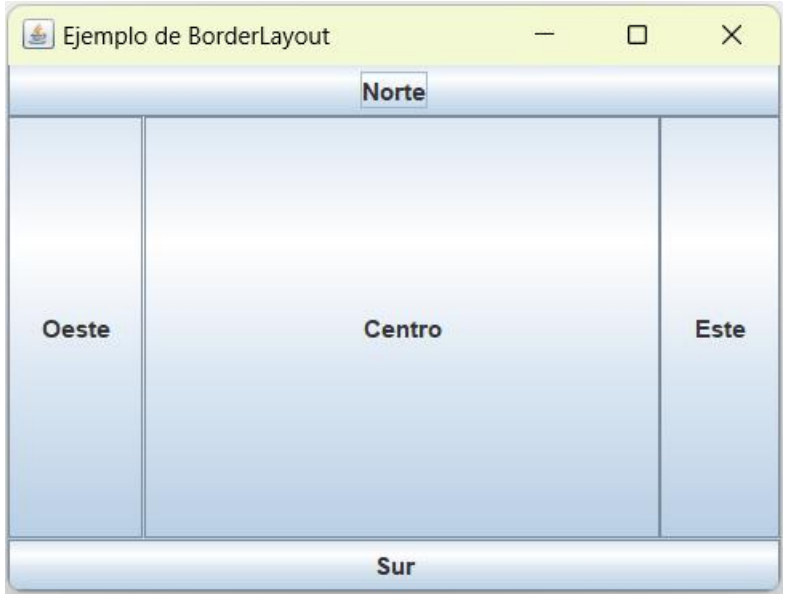
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

4.1 Interfaces gráficas de Usuario

Java Layouts. Ejemplo BorderLayout

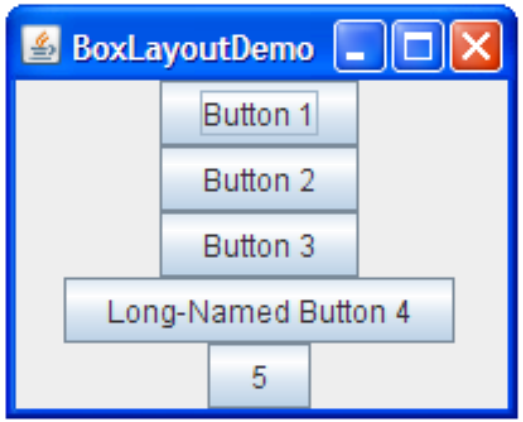
```
import javax.swing.*;
import java.awt.*;

public class Interfaz_con_BorderLayout extends JFrame {

    public Interfaz_con_BorderLayout() {

    //Configuramos los parámetros de la ventana
        setTitle("Ejemplo de BorderLayout");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
    // Creamos componentes para cada región
        JButton botonNorte = new JButton("Norte");
        JButton botonSur = new JButton("Sur");
        JButton botonEste = new JButton("Este");
        JButton botonOeste = new JButton("Oeste");
        JButton botonCentro = new JButton("Centro");
    // Establecemos el layout de JFrame en BorderLayout
        setLayout(new BorderLayout());
    // Agregamos los componentes a las regiones del BorderLayout
        add(botonNorte, BorderLayout.NORTH);
        add(botonSur, BorderLayout.SOUTH);
        add(botonEste, BorderLayout.EAST);
        add(botonOeste, BorderLayout.WEST);
        add(botonCentro, BorderLayout.CENTER);
    //Hacemos visible la ventana
        setVisible(true);
    }
}
```

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



BoxLayout

- Este layout nos permite dos tipos de disposición de los componentes:
  - Unos encima de otros (sólo un componente por fila)
  - Unos al lado de los otros (todos en la misma fila)
- En el constructor del Layout se le asocia con el contenedor
  - También se indica sobre qué eje se va a hacer la ordenación
    - Horizontal: BoxLayout.X_AXIS
    - Vertical: BoxLayout.Y_AXIS
- Los componentes se crean y añaden normalmente
  - Podemos decidir cómo se alinearán entre ellos mediante un valor real que medirá el desplazamiento con respecto a los bordes. Hay varias constantes predefinidas: TOP_ALIGNMENT, BOTTOM_ALINGMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT y CENTER_ALIGNMENT

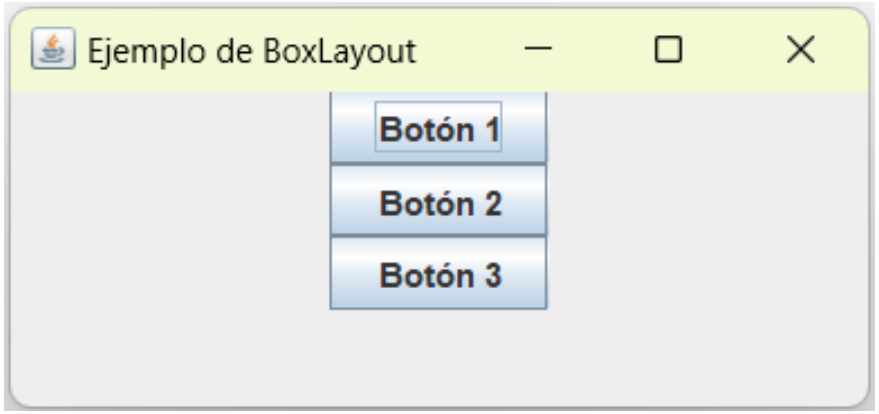


4.1 Interfaces gráficas de Usuario

Java Layouts. Ejemplo BoxLayout

```
import javax.swing.*;
import java.awt.*;

public class Interfaz_con_BoxLayout extends JFrame {
    public Interfaz_con_BoxLayout() {

    // Configuramos los parámetros de la ventana
        setTitle("Ejemplo de BoxLayout");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(320, 150);
    // Creamos componentes
        JButton boton1 = new JButton("Botón 1");
        JButton boton2 = new JButton("Botón 2");
        JButton boton3 = new JButton("Botón 3");
        boton1.setAlignmentX(Component.CENTER_ALIGNMENT);
        boton2.setAlignmentX(Component.CENTER_ALIGNMENT);
        boton3.setAlignmentX(Component.CENTER_ALIGNMENT);
    // Creamos un panel con BoxLayout vertical
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    // Agregamos los componentes
        panel.add(boton1);
        panel.add(boton2);
        panel.add(boton3);
    // Agregamos el panel al JFrame
        getContentPane().add(panel);
        setLocationRelativeTo(null); // Centra la ventana en la pantalla
        setVisible(true);
    }
}
```

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

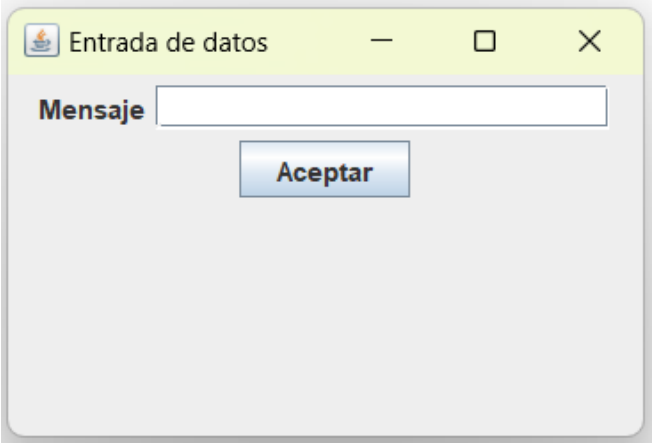


Ejercicios

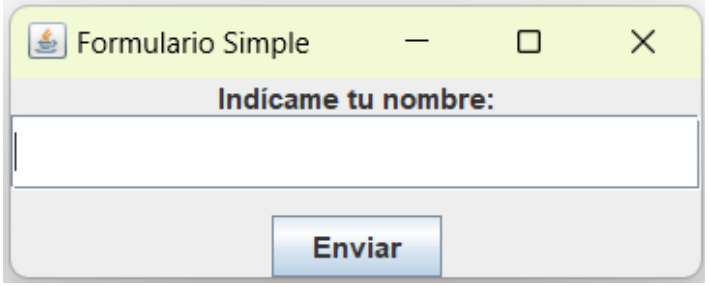
4.1 Interfaces gráficas de Usuario

Ejercicios

1. Escriba un programa con Java Swing de una ventana gráfica que tenga un título, un texto (etiqueta), un campo de texto rellenable y un botón "aceptar" sin acciones asociadas, Utilizando herencia.



2. Escribe un programa Java Swing que utilice BoxLayout para mostrar arriba el texto "Indícame tu nombre", a continuación, un campo de texto rellenable, a continuación, un espacio y a continuación el botón "Enviar"



Bibliografía

- Deitel, H. M. & Deitel, P. J.(2008). *Java: como programar*. Pearson education. Séptima edición.
- The Java tutorials. https://docs.oracle.com/javase/tutorial/
- Páginas de bibliotecas, tutoriales, etc.:
  - https://docs.oracle.com/javase/tutorial/uiswing/learn/index.html
  - https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.wb.doc.user%2Fhtml%2Findex.html
  - https://www.w3schools.com/java/default.asp
  - https://docstore.mik.ua/orelly/java-ent/jnut/index.htm

Técnicas de Programación Avanzada

```
exit(); //Gracias!
```