



**UNIVERSIDAD PRIVADA FRANZ TAMAYO**  
**CARRERA INGENIERIA DE SISTEMAS**

# **HITO4 – TAREA FINAL**

## **MANEJO DE SWING Y SPRING BOOT**

Alumno: Adriana Contreras Arancibia  
Materia: Programación III  
Carrera: Ing. De Sistemas  
Paralelo: Progra(1)  
Docente: Lic. William R. Barra Paredes  
Fecha: 17/06/2020  
Github:

Cochabamba-Bolivia

## TAREA FINAL

### MANEJO DE SWING Y SPRING BOOT

#### 1. TEORIA

##### 1.1 DEFINA QUE ES SWING, A QUE SE REFIERE CUANDO SE HABLA DE AWT

Es un paquete que hace parte de la Java Foundation Classes o más conocida como JFC, la cual provee herramientas o facilidades para la construcción de GUI's o interfaces Graficas de Usuario (graphical user interface).

Podemos decir que Swing es la evolución del AWT (Abstract Window Toolkit), la cual al igual que Swing es un conjunto de librerías enfocadas a la construcción de interfaces, solo que con esta se presentaron algunos problemas en cuanto a portabilidad principalmente cuando se desarrollaban aplicaciones para diferentes sistemas operativos, AWT nace Swing y con las mejoras no solo en aspectos visuales sino también en portabilidad y comportamiento.

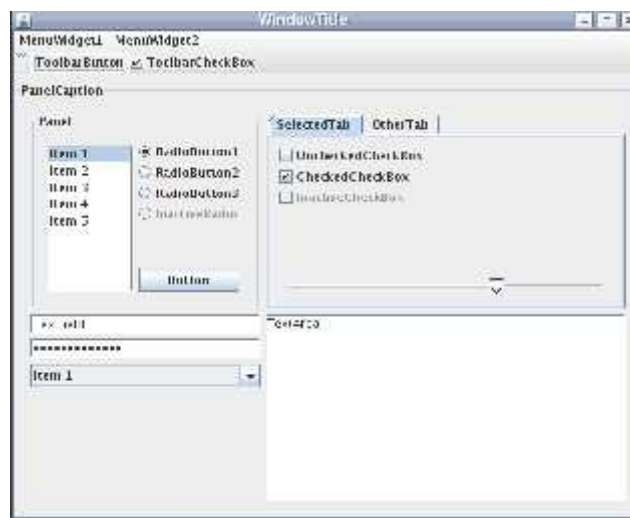


Figura I  
Interfaz gráfica de swing de su biblioteca grafica

AWT permite hacer interfaces gráficas mediante artefactos de interacción con el usuario, como botones, menús, texto, botones para selección, barras de deslizamiento, ventanas de diálogo, selectores de archivos, etc. Y por supuesto despliegue gráfico general. La siguiente figura muestra algunos de estos artefactos de interacción:

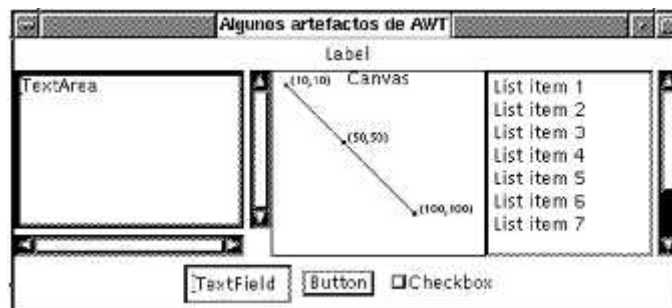


Figura II  
Artefactos de AWT

## 1.2 QUÉ SON LOS COMPONENTE SWING, MENCIONAR ALGUNOS DE ELLOS.

Los componentes de Swing son como botones, tablas, marcos, etc. Los componentes Swing se identifican porque pertenecen al paquete javax. swing. Swing existe desde la JDK 1.1 (como un agregado).

Algunos componentes de swing son:

### Contenedores

- JFrame – Es la Ventana de aplicación, el contenedor principal



- JDialog – Una ventana de tipo Ventana de diálogo, tambien puede ser un contenedor principal.



- JPanel – Permite la creación de paneles independientes donde se almacenan otros componentes.



- JScrollPane – permite la vinculación de barras de desplazamiento en un contenedor.



- JSplitPane – permite la creación de un contenedor dividido en 2 secciones.



- JTabbedPane – Permite la creación de pestañas, cada pestaña representa un contenedor independiente.



- JDesktopPane – Permite crear ventanas dentro de una ventana principal



- JToolBar – Permite introducir una Barra de herramientas.



Síntesis grafica de los contenedores de Java swing



Figura III  
Ejemplo de contenedores

## Componentes Atómicos

Los componentes atómicos son los elementos que no pueden almacenar otros objetos o componentes graficos, por ejemplo, un JPanel no es Atómico, ya que en el podemos almacenar JButtons, JTextField

- JLabel – Permite Vincular Etiquetas, tanto de texto como de imágenes

```
JLabel miLabel;  
miLabel= new JLabel();  
miLabel.setText("Esto es un Label");
```

- JButton – Permite vincular Botones simples.

```
JButton miBoton;  
miBoton= new JButton();  
miBoton.setText("Boton");
```

- JCheckBox – Son Casilla de verificación, ideal para selección múltiples.

```
JCheckBox miCheckbox;  
miCheckbox = new JCheckBox();  
miCheckbox.setText("Check1");
```

- JRadioButton – Permite presentar opciones de selección similares a las checkbox, solo que el enfoque de estas es de única selección.

```
JRadioButton miRadioButton;  
miRadioButton = new JRadioButton();  
miRadioButton.setText("Radio1");
```

- JToggleButton – Botón que al oprimirlo se quedará presionado hasta que se ejecute otro evento.

```
JToggleButton miToggleButton;  
miToggleButton = new JToggleButton();  
miToggleButton.setText("Activar");
```

- JComboBox – Permite mostrar una lista de elementos como un combo de selección.

```
JComboBox miCombo;  
miCombo = new JComboBox();  
miCombo.addItem("Opciones");  
miCombo.addItem("Opcion1");  
miCombo.addItem("Opcion2");  
miCombo.addItem("Opcion3");  
miCombo.addItem("Opcion4");
```

- JScrollBar – Permite mostrar una barra de desplazamiento, regularmente usada en Areas de texto o paneles donde el contenido es mayor que el tamaño del componente.

- JSeparator – Permite separar opciones, es una barra simple.

```
JSeparator separadorHorizontal;  
separadorHorizontal = new JSeparator();  
separadorHorizontal.setBounds(430, 92,  
100,5);
```

- JSlider - Permite vincular un Deslizador en nuestra ventana.

```
JSlider miDeslizado;  
miDeslizador = new JSlider(JSlider.HORIZONTAL,  
0, 100, 30);  
miDeslizador.setBounds(430, 140, 100, 30);  
miDeslizador.setValue(0);
```

- JSpinner – permite vincular una caja de texto con botones integrados para seleccionar algún valor.

```
JSpinner miSpinner;  
miSpinner = new JSpinner();
```

- JProgressBar – Establece una barra de progreso.

```
JProgressBar miBarra;  
miBarra = new JProgressBar();  
miBarra.setBounds(450, 180, 110, 20);
```

Sintetis grafica de los componentes atómicos de Java swing



Figura IV  
Ejemplo de los componentes atómicos

## Componentes de Texto.

Permiten procesar cadenas de texto, sea como entrada o salida de información.

- **TextField** – Permite introducir un campo de texto simple.

```
cajaDeTexto = new JTextField();
cajaDeTexto.setText("CoDejaVu");
cajaDeTexto.setBounds(90, 60, 90, 23);
```

- **JFormattedTextField** – Permite introducir un campo de texto con formato, (si definimos que solo recibe números no permitirá letras...)
- **JPasswordField** – Campo de texto que oculta los caracteres ingresados.

```
campoContraseña = new JPasswordField();
campoContraseña.setBounds(530, 60, 80, 23);
```

- **TextArea** – Permite vincular un área de texto donde el usuario ingresara información o simplemente para presentar cadenas de texto.

```
areaDeTexto = new JTextArea();
areaDeTexto.setText(CadenaConElTexto);
areaDeTexto.setBounds(90, 90, 520, 103);
```

- **EditorPane** – Permite vincular un área de texto con propiedades de formato.

```
areaEditorPane = new JEditorPane();
areaEditorPane.setBounds(90, 200, 520, 103);
/**Definimos el tipo de texto que utiliza*/
areaEditorPane.setContentType("text/html");
areaEditorPane.setText(CadenaConTextoHTML);
```

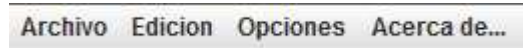
- **TextPane** – Similar al anterior, permitiendo otras opciones de formato, colores, iconos entre otros.

```
StyleContext estilo = new StyleContext();
Style estiloRojo = estilo.addStyle(null,
null);
StyleConstants.setForeground( estiloRojo,
Color.red );
DefaultStyledDocument estiloPorDefecto=new
DefaultStyledDocument(estilo);
areaTextPane = new
JTextPane(estiloPorDefecto);
areaTextPane.setCharacterAttributes(estiloRojo, false);
areaTextPane.setBounds(90, 310, 520, 103);
```

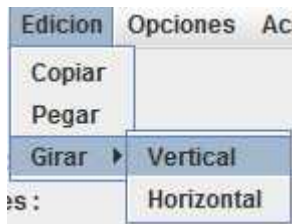
## Componentes de Menús.

Estos componentes permiten vincular opciones de menú en nuestras ventanas, tipo menú principal, como por ejemplo el conocido Inicio, Archivo, Edición etc.

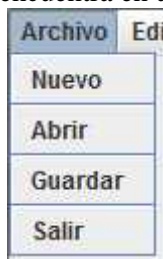
- JMenuBar – Permite vincular una barra de menús.



- JMenu– Permite vincular botones o enlaces que al ser pulsados despliegan un menú principal.



- JMenuItem – Botón u opción que se encuentra en un menú.



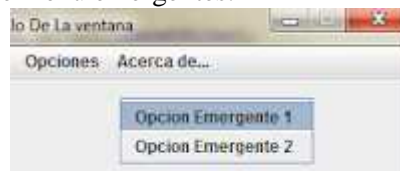
- JCheckBoxMenuItem– Elemento del menú como opciones de checkbox.



- JRadioButtonMenuItem– Elemento del menú como botón de selección.



- JPopupMenu– Opciones de menú emergentes.





### 1.3 IMPORTANCIA DE JFRAME.

JFrame es una clase utilizada en Swing (biblioteca gráfica) es muy importante ya que nos permite generar ventanas sobre las cuales añadir distintos objetos para interactuar con el usuario, también es importante ya que JFrame posee algunas nociones típicas de una ventana como minimizar, cerrar, maximizar y poder moverla.

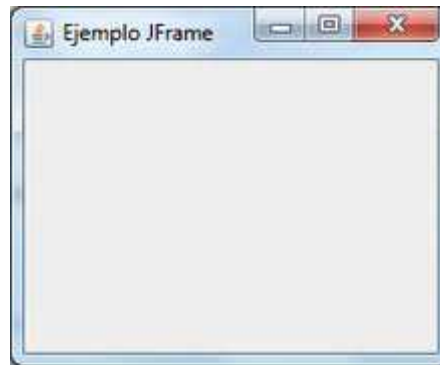


Figura V  
Ejemplo grafico de JFrame

### 1.4 QUÉ ES UN LISTENER EN SWING.

Los Listeners (oyentes o escuchadores en español) se encargan de controlar los eventos, esperan a que el evento se produzca y realiza una serie de acciones. Según el evento, necesitaremos un Listener que lo controle.

Cada Listener tiene una serie de métodos que debemos implementar obligatoriamente, aunque solo queramos usar uno solo de ellos.

#### Listeners que existen

Nombre Listener	Descripción	Métodos	Eventos
ActionListener	Se produce al hacer click en un componente, también si se pulsa Enter teniendo el foco en el componente.	public void actionPerformed(ActionEvent)	<ul style="list-style-type: none"><li>▪ <b>JButton</b>: click o pulsar Enter con el foco activado en él</li><li>▪ <b>JList</b>: doble click en un elemento de la lista.</li><li>▪ <b>JMenuItem</b>: selecciona una opción del menú.</li><li>▪ <b>TextField</b>: al pulsar Enter con el foco activado.</li></ul>

KeyListener	Se produce al pulsar una tecla. según el método cambiara la forma de pulsar la tecla.	<pre>public void keyTyped(KeyEvent e)  public void keyPressed(KeyEvent e)  public void keyReleased(KeyEvent e)</pre>	<p>Cuando pulsamos una tecla, según el Listener:</p> <ul style="list-style-type: none"> <li>▪ <b>keyTyped:</b> al pulsar y soltar la tecla</li> <li>▪ <b>keyPressed:</b> al pulsar la tecla.</li> <li><b>keyReleased:</b> al soltar la tecla</li> </ul>
FocusListener	Se produce cuando un componente gana o pierde el foco, es decir, que esta seleccionado.	<pre>public void focusGained(FocusEvent e)  public void focusLost(FocusEvent e)</pre>	Recibir o perder el foco.
MouseListener	Se produce cuando realizamos una acción con el ratón.	<pre>public void mouseClicked(MouseEvent e)  public void mouseEntered(MouseEvent e)  public void mouseExited(MouseEvent e)  public void mousePressed(MouseEvent e)  public void mouseReleased(MouseEvent e)</pre>	<p>Según el Listener:</p> <ul style="list-style-type: none"> <li>▪ <b>mouseClicked:</b> pinchar y soltar.</li> <li>▪ <b>mouseEntered:</b> entrar en un componente con el puntero.</li> <li>▪ <b>mouseExited:</b> salir de un componente con el puntero</li> <li>▪ <b>mousePressed:</b> presionar el botón.</li> <li>▪ <b>mouseReleased:</b> soltar el botón.</li> </ul>
MouseMotionListener	Se produce con el movimiento del mouse.	<pre>public void mouseDragged(MouseEvent e)  public void mouseMoved(MouseEvent e)</pre>	<p>Según el Listener:</p> <ul style="list-style-type: none"> <li>▪ <b>mouseDragged:</b> click y arrastrar un componente</li> <li>▪ <b>mouseMoved:</b> al mover el puntero sobre un elemento</li> </ul>

## 1.5 DESCRIBA CÓMO FUNCIONA PASO A PASO EL COMPONENTE GRIDLAYOUT.

Un GridLayout es un administrador de diseño que coloca componentes dentro de una cuadrícula con el mismo tamaño de celda. Puede establecer el número de filas, columnas, la brecha horizontal y la brecha vertical utilizando los siguientes métodos:

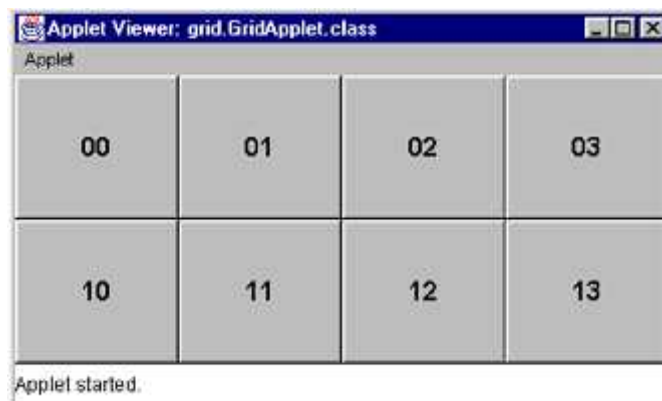
```
) setRows(int rows)
) setColumns(int columns)
) setHgap(int hgap)
) setVgap(int vgap)
```

O puedes configurarlos con los siguientes constructores:

```
) GridLayout(int rows, int columns)
) GridLayout(int rows, int columns, int hgap, int vgap)
```



Figura VI  
Aplicación De un GridLayout a una calculadora



Ejemplo de GridLayout

## 1.6 DESCRIBA CÓMO FUNCIONA PASO A PASO EL COMPONENTE FLOWLAYOUT.

El FlowLayout, es aquel layout q ubica a todos los componentes en forma horizontal, en el orden q le digamos. Primero tenemos que crear el contenedor (JFrame, JPanel, etc), y luego atraves del metodo “setLayout()” asignarle el layout correspondiente.

La propiedad layout tiene un editor asociado que es una caja de selección, elegimos el elemento FlowLayout hace que los botones se alinean en el centro y en la parte superior

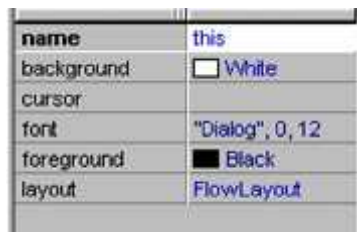
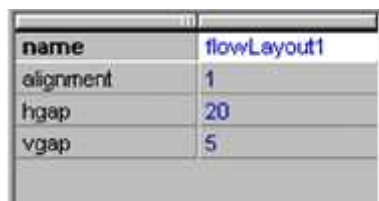


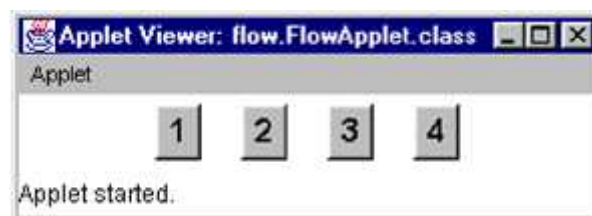
Figura VII

Referencia de Flowlayout

En el panel de estructura y situamos el cursor en flowLayout1, aparece su hoja de propiedades



FlowLayout es un gestor que pone los controles en una línea, como puede verse en la figura



## 1.7 DESCRIBA CÓMO FUNCIONA PASO A PASO EL COMPONENTE BORDERLAYOUT.

Un contenedor que organiza y define el tamaño de los componentes atendiendo a cinco regiones: NORT, SOUTH, EAST, WEST y CENTER en la que cada región no puede contener más de un componente

Los pasos para establecer el gestor BorderLayout son distintos a los empleados para el gestor FlowLayout.

1. Crear los botones (objetos de la clase Button) y el gestor de diseño (objeto de la clase BorderLayout)

```
Button btnOeste = new Button();  
BorderLayout borderLayout1 = new BorderLayout();
```

2. Establecer sus propiedades en init

```
btnOeste.setFont(new Font("Dialog", 1, 16));  
btn1.setLabel("Oeste");
```

3. Añadir los controles al applet (o a un panel) mediante add, indicando en el segundo argumento la posición que ocupará cada control en el panel mediante miembros estáticos de la clase BorderLayout.

```
this.add(btnOeste, BorderLayout.WEST);
```

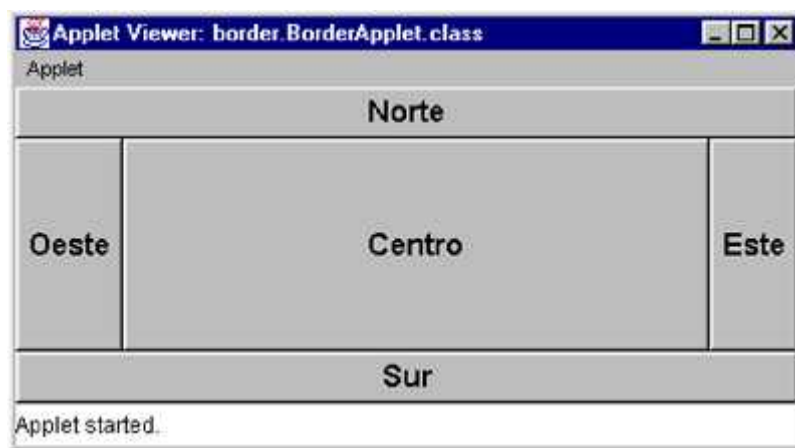


Figura VIII  
Ejemplo de BorderLayout



Figura IX  
Segundo ejemplo de BorderLayout

## 2. PRACTICA

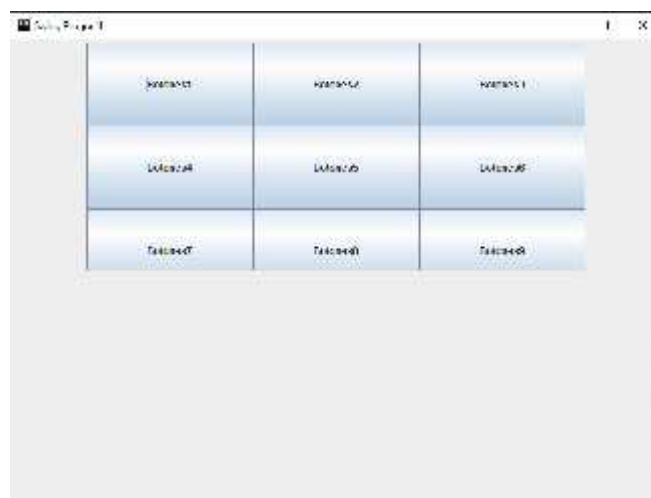
### 2.1 GRIDLAYOUT

Mostrar un ejemplo en donde se use un GridLayout.

- Como se usa:  
Se para que todos los componentes se adaptan al tamaño de la ventana
- Cómo funciona:  
Funciona implementado la siguiente línea de código

 **setLayout(new GridLayout(3, 3));**

Captura de la aplicación corriendo



Aquí Podemos ver que los botones se van ajustando a la ventana

## Código

```
@Component
public class GridLayoutEjemplo extends JFrame {
    JButton[] botonera= new JButton[9];


    public GridLayoutEjemplo(){
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(400,300);
        IniciarComponentes();
    }

    private void IniciarComponentes() {
        setLayout(new GridLayout(3,3));
        for(int i=0;i< botonera.length;i++) {
            botonera[i] = new JButton("Botones" + (i + 1));
            add(botonera[i]);
        }
    }
}
```

## 2.1 FLOWLAYOUT

Mostrar un ejemplo en donde se use un FLOWLAYOUT.

- Como se usa:  
Se usa para poner todos los componentes en fila, se encarga de que todos los componentes quepan en la ventana
- Como Funciona:  
Funciona implementado la siguiente línea de código

 **FlowLayout ejemploLayout = new FlowLayout(FlowLayout.CENTER);**  
**setLayout(ejemploLayout);**

## Captura de la aplicación corriendo



## Código

```
@Component
public class FlowLayoutEjemplo extends JPanel {
    JButton boton1, boton2, boton3, boton4;

    public FlowLayoutEjemplo () {
        System.setProperty("butBackColor", "#C1ECF1");
        System.setProperty("textColor", "#0B0BF6");

        this.setPreferredSize(new Dimension(600, 300));/--190
        this.setBackground(Color.blue);

        IniciarComponentes();
    }

    private void IniciarComponentes() {
        FlowLayout ejemploLayout = new FlowLayout(FlowLayout.CENTER);
        setLayout(ejemploLayout);

        boton1 = new JButton();
        boton1.setText("primer boton");

        boton2 = new JButton();
        boton2.setText("segundo boton");

        boton3 = new JButton();
        boton3.setText("tercer boton");

        boton4 = new JButton();
        boton4.setText("cuarto boton");

        add(boton1);
        add(boton2);
        add(boton3);
        add(boton4);

    }
}
```



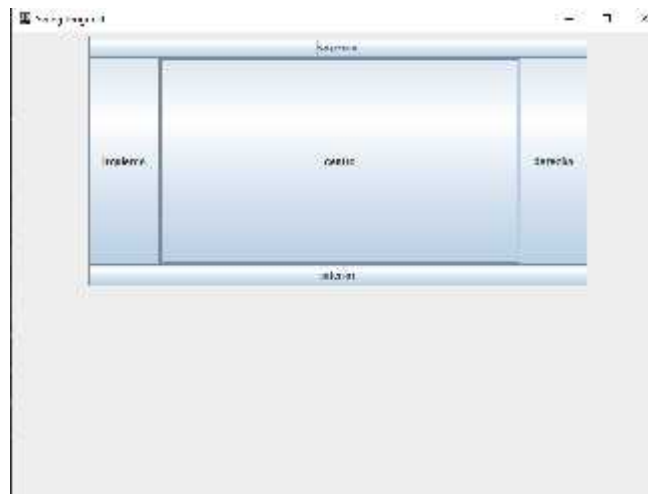
## 2.3 BORDERLAYOUT

Mostrar un ejemplo en donde se use un BORDERLAYOUT.

- Como se usa:  
Permite definir posiciones pensando en aprovechar todos los espacios de la interfaz, divide la ventana en 5 partes este, oeste, norte, sur y centro
- Como Funciona:  
Funciona implementado la siguiente línea de código



Captura de la aplicación corriendo



Código

```
@Component
public class BorderLayoutEjemplo extends JPanel {

    JButton boton1, boton2, boton3, boton4, boton5;

    public BorderLayoutEjemplo () {
        this.setPreferredSize(new Dimension(600, 300));
        this.setBackground(Color.blue);
        this.setLayout(new GridLayout(5, 0));
        IniciarComponentes();
    }

    private void IniciarComponentes() {
        setLayout(new BorderLayout());

        boton1 = new JButton();
        boton1.setText("centro");

        boton2 = new JButton();
        boton2.setText("superior");
```

```
        boton3 = new JButton();  
        boton3.setText("derecha");  
  
        boton4 = new JButton();  
        boton4.setText("izquierda");  
  
        boton5 = new JButton();  
        boton5.setText("inferior");  
  
        add(boton1, BorderLayout.CENTER);  
        add(boton2, BorderLayout.NORTH);  
        add(boton3, BorderLayout.EAST);  
        add(boton4, BorderLayout.WEST);  
        add(boton5, BorderLayout.SOUTH);  
  
    }  
}
```