Práctica 1: Interfaz Grafica

Objetivo:

- Iniciar con los conceptos básicos para el manejo de Interfaz grafica en Java.
- Conocer y trabajar con algunos de los más importantes Layouts

Tiempo destinado: 2hrs.

Fundamentos:

Las interfaces gráficas contienen varios elementos:

1. **Componentes/Controles (Controls):** son los elementos gráficos que tenemos a disposición para crear las GUIs (Interfases Graficas del Usuario), los cuales son objetos de clases que heredan de la clase Component, por ejemplo: Button, List, TextField, TextArea, Label, etc.
2. **Contenedores (Containers):** son componentes que contienen a otros componentes, por ejemplo: Panel, Window y Frame.
3. **Distribución de los componentes (Layouts):** es la forma en que se distribuyen los componentes dentro de un container.

Existen dos tipos de layout: absolutos y relativos.

En los **layouts absolutos** se emplean principalmente cuando se usan herramientas de diseño visual. En los **layouts relativos** definen reglas para distribuir y los componentes se acomodan automáticamente dentro del container en función de las reglas que impone el layout.

Existen dos tipo de APIs para desarrollar GUIs en Java **AWT(Abstract Window Toolkit)** y **Swing**, debido a que la API AWT es muy limitada se usa con mayor frecuencia la API Swing.

Todo el manejo de eventos y de layout es exactamente el mismo para Swing y para AWT. Los Layouts se pueden combinar (uno puede contener a otros).

Existen cuatro tipos de distribuciones relativas: FlowLayout, BorderLayout, GridLayout y GridBagLayout.

FlowLayout.

Distribuye los componentes uno al lado del otro en la parte superior del contenedor (container). Por defecto provee una alineación centrada, pero también puede alinearlos a la izquierda o derecha. Las instrucciones para configurar y emplear el FlowLayout son:

- setLayout (new Flowlayout());// Configurar el FlowLayout Componentes centrados por omisión
- setLayout (new FlowLayout (Flowlayout.LEFT)); //Componentes alineados a la izquierda
- setLayout (new FlowLayout (FlowLayout.RIGTH)); //Componentes alineados ala derecha

Para agregar componentes al FlowLayout se emplea la instrucción:

- add(nombre_componente); //adicionar un componente dentro del layout

BorderLayout

Divide el espacio del container en 5 regiones: NORTH, SOUTH, EAST, WEST y CENTER. Admite un único componente por región.

Para configurar el layout BorderLayout:

- setLayout (new BorderLayout()); //definir el tipo de layout

Para agregar/colocar componentes al layout:

- add (nombre-componente, BorderLayout.NORTH) //colocar un componente al norte
- add (nombre-componente, BorderLayout.SOUTH) //colocar un componente al sur
- add (nombre-componente, BorderLayout.WEST) //colocar un componente al oeste
- add (nombre-componente, BorderLayout.EAST) //colocar un componente al este
- add (nombre-componente, BorderLayout.CENTER) //colocar un componente al centro

GridLayout

Divide el espacio del container en una rejilla de n filas por m columnas donde todas las celdas son de igual tamaño. Admite exactamente n (filas) por m (columnas) componentes, uno por celda. Para configurar el Layout GridLayout:

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

Para agregar componentes al GridLayout:

- add(nombre_componente); //los componentes se agregan primero de izquierda a derecha (primer fila) y luego de arriba hacia abajo (segunda fila, tercer fila ….).

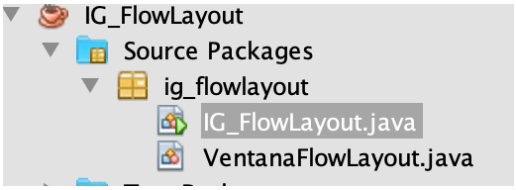
GridLayout no permite agregar los componentes en forma aleatoria, por lo contrario, tienen que agregarse en orden de izquierda a derecha y de arriba abajo iniciando por la primer fila.

Instrucciones:

1. Realice correctamente cada paso en la sección de desarrollo, documente la práctica con capturas de pantalla, mostrando cada paso realizado y los resultados obtenidos.
2. Suba su reporte al espacio asignado en el aula virtual.

Desarrollo:

1. Genere un nuevo proyecto de Java en NetBeans, llamado IG_FlowLayout con la siguiente estructura de clases:



   a. Dentro de la clase VentanaFlowLayout capture el siguiente código:

```
package ig_flowlayout;

import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.Frame;

/**
 *
 * @author jjpg
 */
public class VentanaFlowLayout extends Frame{
    private Button btn1, btn2, btn3, btn4;

    public VentanaFlowLayout() {
        super("Ventana usando FlowLayout");
        setLayout(new FlowLayout());

        btn1 = new Button("Boton 1");
        add(btn1);

        btn2 = new Button("Boton 2");
        add(btn2);

        btn3 = new Button("Boton 3");
        add(btn3);

        btn4 = new Button("Boton 4");
        add(btn4);

        setSize(300, 300);

        setVisible(true);
    }

}
```

b. En la clase IG_FlowLayout capture el siguiente código:

```
package ig_flowlayout;

/**
 *
 * @author jjpg
 */
public class IG_FlowLayout {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        VentanaFlowLayout ventana = new VentanaFlowLayout();
    }

}
```

2. Ejecute su aplicación, deberá de observar una ventana que contiene los botones agregados al Frame, modifique el tamaño de la ventana y observe el acomodo de los botones.

Nota: Nuestra aplicación no tiene eventos, por lo que no se puede cerrar la ventana, tendra que detener la ejecución.

3. Responda:
   a. ¿Cuál es el contenedor de los botones?
   b. ¿Cómo se organizan los botones según se modifica el tamaño de la ventana? ¿Qué orden sigue?

4. De igual forma, realice un nuevo proyecto y pruebe el border layout con el siguiente código:

```
6   package ig_flowlayout;
7
8   import java.awt.BorderLayout;
9   import java.awt.Button;
10  import java.awt.FlowLayout;
11  import java.awt.Frame;
12
13  /**
14   *
15   * @author jjpg
16   */
17  public class VentanaBorderLayout extends Frame {
        private Button btnNorte, btnSur, btnEste, btnOeste, btnCentro;
19
20      public VentanaBorderLayout() {
21          super("Ventana usando BorderLayout");
            setLayout(new FlowLayout());
23
24          btnNorte = new Button("Boton Norte");
25          btnSur = new Button("Boton Sur");
26          btnEste = new Button("Boton Este");
27          btnOeste = new Button("Boton Oeste");
28          btnCentro = new Button("Boton Centro");
29
            add(btnNorte, BorderLayout.NORTH);
            add(btnSur, BorderLayout.SOUTH);
            add(btnEste, BorderLayout.EAST);
            add(btnOeste, BorderLayout.WEST);
            add(btnCentro, BorderLayout.CENTER);
35
            setSize(300, 300);
            setVisible(true);
38      }
39  }
```

5. Y por último realice un nuevo proyecto para GridLayout

```
import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.GridLayout;

/**
 *
 * @author jjpg
 */
public class VentanaGridLayout extends Frame {

    private Button btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9;

    public VentanaGridLayout() {
        super("Ventana usando BorderLayout");
        setLayout(new GridLayout(3,3));

        btn1 = new Button("Boton 1");
        btn2 = new Button("Boton 2");
        btn3 = new Button("Boton 3");
        btn4 = new Button("Boton 4");
        btn5 = new Button("Boton 5");
        btn6 = new Button("Boton 6");
        btn7 = new Button("Boton 7");
        btn8 = new Button("Boton 8");
        btn9 = new Button("Boton 9");

        add(btn1);
        add(btn2);
        add(btn3);
        add(btn4);
        add(btn5);
        add(btn6);
        add(btn7);
        add(btn8);
        add(btn9);

        setSize(300, 300);
        setVisible(true);
    }
```

6. Analice cada layout y realice sus observaciones y conclusiones.