Lab 12

Graphical User Interface - Layout Managers

Objective:

The purpose of lab is to make students understand basic concepts of Layouts of GUI in Java. The students will learn about the three basic types of layouts and understand the difference between them.

Activity Outcomes:

Students will be able to create frames with different layouts. Students will be able to create simple to medium level complex GUI

Instructor Note:

As pre-lab activity, read Chapter 15 from the text book “Introduction to Java Programming”, Y. Daniel Liang, Pearson, 2019.

1) Useful Concepts

In many other window systems, the user-interface components are arranged by using hardcoded pixel measurements. For example, put a button at location (10, 10) in the window using hard-coded pixel measurements, the user interface might look fine on one system but be unusable on another. Java's layout managers provide a level of abstraction that automatically maps your user interface on all window systems. The Java GUI components are placed in containers, where they are arranged by the container's layout manager. In the preceding program, you did not specify where to place the OK button in the frame, but Java knows where to place it, because the layout manager works behind the scenes to place components in the correct locations. A layout manager is created using a layout manager class.
Layout managers are set in containers using the SetLayout(aLayoutManager) method. For example, you can use the following statements to create an instance of XLayout and set it in a container:
LayoutManagerlayoutManager = new XLayout(); container.setLayout(layoutManager);

FlowLayout

FlowLayout is the simplest layout manager. The components are arranged in the container from left to right in the order in which they were added. When one row is filled, a new row is started. You can specify the way the components are aligned by using one of three constants: FlowLayout.RIGHT, FlowLayout.CENTER, or FlowLayout.LEFT. You can also specify the gap between components in pixels. The class diagram for FlowLayout is shown in Figure below

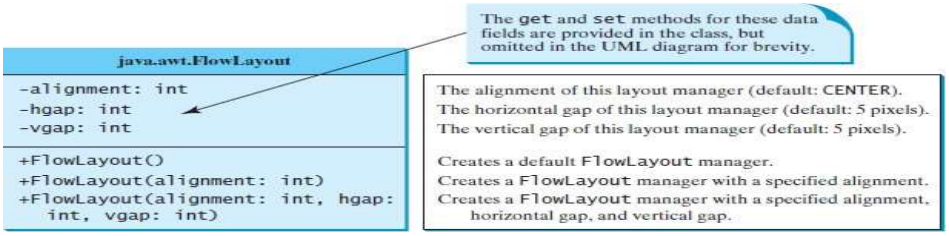


Figure 1 Flow Layout

Grid Layout

The GridLayout manager arranges components in a grid (matrix) formation. The components are placed in the grid from left to right, starting with the first row, then the second, and so on, in the order in which they are added. The class diagram for GridLayout is shown in Figure below.

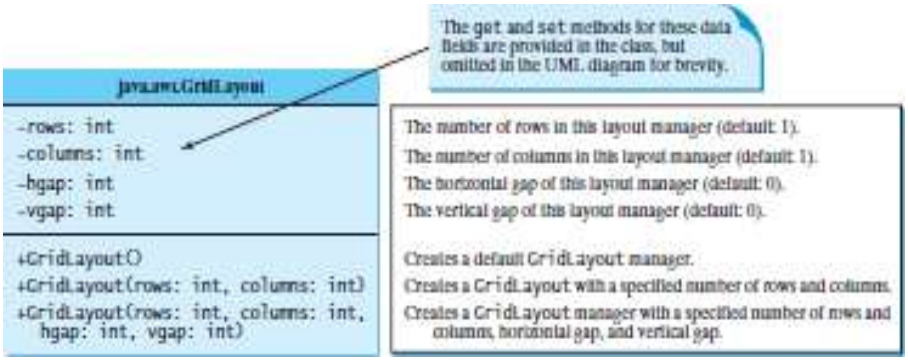


Figure 2 Grid Layout

BorderLayout

The BorderLayout manager divides a container into five areas: East, South, West, North, and Center. Components are added to a BorderLayout by using add(Component, index), where index is a constant as mentioned below:

- BorderLayout.EAST,
- BorderLayout.SOUTH,
- BorderLayout.WEST,
- BorderLayout.NORTH,
- BorderLayout.CENTER.

The class diagram for BorderLayout is shown in Figure below:

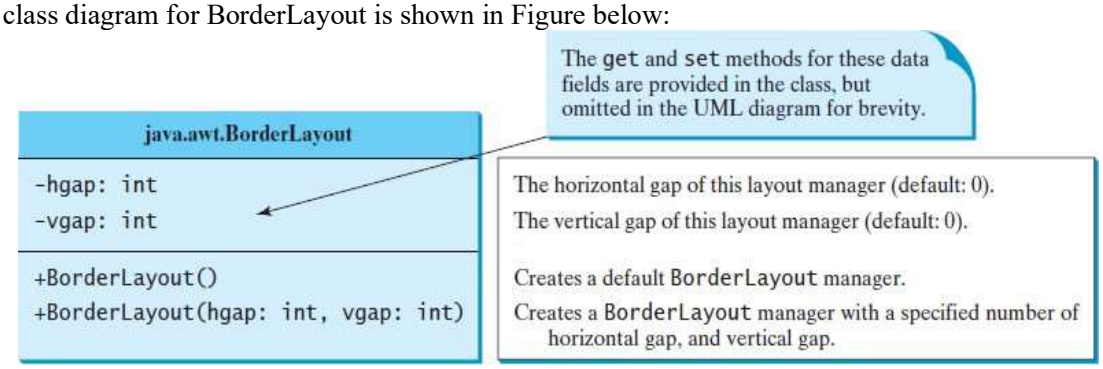


Figure 3 Border Layout

Panels as SubContainers

Suppose that you want to place ten buttons and a text field in a frame. The buttons are placed in grid formation, but the text field is placed on a separate row. It is difficult to achieve the desired look by placing all the components in a single container. With Java GUI programming, you can divide a window into panels. Panels act as subcontainers to group user-interface components.
You add the buttons in one panel, then add the panel into the frame. The Swing version of panel is JPanel. You can use new JPanel() to create a panel with a default FlowLayout manager or new JPanel(LayoutManager) to create a panel with the specified layout manager.
Use the add(Component) method to add a component to the panel. For example, the following code creates a panel and adds a button to it:
JPanel p = new JPanel(); p.add(new JButton("OK"));
Panels can be placed inside a frame or inside another panel. The following statement places panel p into frame f:

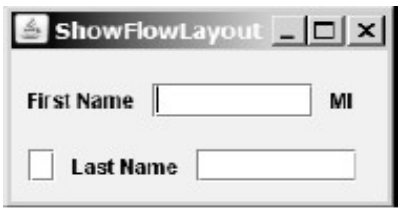
f.add(p);

2) Solved Lab Activities (Allocated Time 1 Hr.)

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*15 mins*	*High*	*CLO-4*
Activity 2	*15 mins*	*High*	*CLO-4*
Activity 3	*15 mins*	*High*	*CLO-4*
Activity 4	*15 mins*	*High*	*CLO-4*

Activity 1:

Create the following frame using Flow Layout.



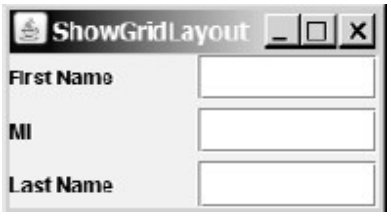
Solution:

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;
```

```
public class ShowFlowLayout extends JFrame {

    public ShowFlowLayout() {
        //Set FlowLayout, aligned left with
horizontal gap 10
        //and vertical gap 20 between components
        setLayout(new
FlowLayout(FlowLayout.LEFT, 10, 20));
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
        add(new JLabel("Last Name"));
        add(new JTextField(8));

    }

    /**
     * Main method
     */
    public static void main(String[] args) {
        ShowFlowLayout frame = new
ShowFlowLayout();
        frame.setTitle("Show FlowLayout");
        frame.setSize(200, 200);
        frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
        frame.setVisible(true);
    }
}
```

Activity 2*:*

Create the following frame using Grid Layout



Solution:

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.GridLayout;

public class ShowGridLayout extends JFrame {

    public ShowGridLayout() {
        //Set GridLayout, 3 rows, 2 columns, and
gaps 5 between
        //components horizontally and vertically
        setLayout(new GridLayout(3,2,5,5));
        //Add labels and text fields to the
frame
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
        add(new JLabel("Last Name"));
        add(new JTextField(8));

    }

    /**
     * Main method
     */
    public static void main(String[] args) {
        ShowGridLayout frame = new
ShowGridLayout();
        frame.setTitle("Show GridLayout");
        frame.setSize(200, 125);
```

```
        frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
        frame.setVisible(true);
    }
}
```

Activity 3:

Run the below code and ensure that the output matches the UI below the code.

Solution:

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.BorderLayout;

public class ShowBorderLayout extends JFrame {

    public ShowBorderLayout() {
```

```
        //Set BorderLayout with horizontal gaps
5 and vertical gap 10
        setLayout(new BorderLayout(5, 10));
        //Add buttons to the frame
        add(new JButton("EAST"),
BorderLayout.EAST);
        add(new JButton("SOUTH"),
BorderLayout.SOUTH);
        add(new JButton("WEST"),
BorderLayout.WEST);
        add(new JButton("NORTH"),
BorderLayout.NORTH);
        add(new JButton("CENTER"),
BorderLayout.CENTER);

    }

    /**
     * Main method
     */
    public static void main(String[] args) {
        ShowBorderLayout frame = new
ShowBorderLayout();
        frame.setTitle("Show BorderLayout");
        frame.setSize(300, 200);
        frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
        frame.setVisible(true);
    }
}
```

Activity 4:

Run the below code and ensure that the output is similar to below:

Solution:

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

public class TestPanels extends JFrame {

    public TestPanels() {
        // Create panel p1 for the buttons and
set GridLayout
        JPanel p1 = new JPanel();
        p1.setLayout(new GridLayout(4, 3));

        //Add buttons to the panel
        for (int i = 1; i <= 9; i++) {
            p1.add(new JButton("" + i));
        }
        p1.add(new JButton("" + 0));
        p1.add(new JButton("Start"));
        p1.add(new JButton("Stop"));

        //Create panel p2 to hold a text field
and p1
        JPanel p2 = new JPanel(new
BorderLayout());
        p2.add(new JTextField("Time to be
displayed here"),
                BorderLayout.NORTH);
        p2.add(p1, BorderLayout.CENTER);

        // add contents into the frame
        add(p2, BorderLayout.EAST);
```

```
        add(new JButton("Food to be placed
here"),
                BorderLayout.CENTER);
    }

    /**
     * Main method
     */
    public static void main(String[] args) {
        TestPanels frame = new TestPanels();
        frame.setTitle("The Front View of a
Microwave Oven");
        frame.setSize(400, 250);
        frame.setLocationRelativeTo(null);
//Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
        frame.setVisible(true);

    }
}
```

3) Graded Lab Tasks (Allocated Time 1 Hr.)

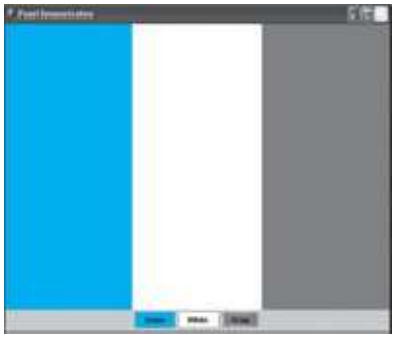
Note: The instructor can design graded lab activities according to the level of difficulty and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

Create the following GUI. You do not have to provide any functionality.

Lab Task 2

Create the following GUI. You do not have to provide any functionality.



Lab Task 3

Create the following GUI. You do not have to provide any functionality.

Lab Task 4

Create the following GUI. You do not have to provide any functionality.

Lab Task 5

Create the following GUI. You do not have to provide any functionality.

Note: Use JScrollPane class for the creating the scroll pane.

Lab 13

Graphical User Interface – Event Driven Programming

Objective:

In this lab, student will learn and practice the basic concepts of events-based programming in GUI based interfaces in Java. They will learn event generation and event handling.

Activity Outcomes:

After completing this lesson, you should be able to do the following:

- Understand why events are needed in GUI
- Understand the mechanics of event generation and event handling
- Practice simple event-based programming
- Create a simple but useful GUI based program

Instructor Note:

As pre-lab activity, read Chapter 15 from the text book “Introduction to Java Programming”, Y. Daniel Liang, Pearson, 2019.

1) Useful Concepts

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Event describes the change of state of any object.

Example : Pressing a button, Entering a character in Textbox Event handling has three main components,

- **Events :** An event is a change of state of an object.
- **Events Source :** Event source is an object that generates an event.
- **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.

A source generates an Event and sends it to one or more listeners registered with the source. Once event is received by the listener, they processe the event and then return. Events are supported by a number of Java packages, like **java.util**, **java.awt** and **java.awt.event**.

Important Event Classes and Interface

Event Classe	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	**ActionListener**
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	**MouseListener**
KeyEvent	generated when input is received from keyboard	**KeyListener**

ItemEvent	generated when check-box or list item is clicked	**ItemListener**
TextEvent	generated when value of textarea or textfield is changed	**TextListener**
MouseWheelEvent	generated when mouse wheel is moved	**MouseWheelListener**
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	**WindowListener**
ComponentEvent	generated when component is hidden, moved, resized or set visible	**ComponentEventListener**
ContainerEvent	generated when component is added or removed from container	**ContainerListener**
AdjustmentEvent	generated when scroll bar is manipulated	**AdjustmentListener**
FocusEvent	**generated when component gains or loses keyboard focus**	**FocusListener**

2) Solved Lab Activities (Allocated Time 45 Min.)

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*15 mins*	*High*	*CLO-4*
Activity 2	*15 mins*	*High*	*CLO-4*
Activity 3	*15 mins*	*High*	*CLO-4*

Activity 1:

Run the below code. It should create a label and a button. The label should have text "Hello" but when the button the pressed the text changes to "Bye"

Solution:

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Test extends JFrame {

    private JLabel label;

    private JButton b;

    public Test() {

        this.setLayout(new
FlowLayout(FlowLayout.LEFT, 10, 20));
        label = new JLabel("Hello");
        this.add(label);
        b = new JButton("Toggle");
        b.addActionListener(new myHandler());
        add(b);
    }

    class myHandler implements ActionListener {

        public void actionPerformed(ActionEvent
e) {
            label.setText("Bye");
        }

    }

    public static void main(String[] args) {

// TODO Auto-generated method stub
        Test f=new Test();
        f.setTitle("Hi and Bye");
        f.setSize(400, 150);
```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
;
        f.setLocationRelativeTo(null);
        f.setVisible(true);

    }

}
```

Activity 2*:*

Run and understand the below code. Basically this code sets the text in label on button press event. Any text entered in the textfield is copied into the label. Ensure that it is so and understand how it works.

Solution:

```
import java.awt.FlowLayout;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Test extends JFrame {

    private JTextField tf1;
    private JLabel label;
    private JButton b;

    public Test() {

        this.setLayout(new
FlowLayout(FlowLayout.LEFT, 10, 20));
        tf1 = new JTextField(8);
        this.add(tf1);
        label = new JLabel("New Text");
        this.add(label);
        b = new JButton("Change");
        b.addActionListener(new myHandler());
        add(b);
    }
```

```
    class myHandler implements ActionListener {

        public void actionPerformed(ActionEvent
e) {
            String s = tf1.getText();
            label.setText(s);

            tf1.setText("");

        }

    }

    public static void main(String[] args) {

// TODO Auto-generated method stub
        Test f = new Test();
        f.setTitle("Copy Text");
        f.setSize(400, 150);

f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
;
        f.setLocationRelativeTo(null);;
        f.setVisible(true);

    }

}
```

Activity 3*:*

Run and understand the below code. We now first see which button triggered the event through the getSource event and then either disapper one button or copy text in the TextField into the label.

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Test extends JFrame {
```

```
    private JTextField tf1;
    private JLabel label;
    private JButton b;
    private JButton b1;

    public Test() {

        this.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
        tf1 = new JTextField(8);
        this.add(tf1);
        label = new JLabel("New Text");
        this.add(label);
        b = new JButton("Change");
        b.addActionListener(new myHandler());
        add(b);

        b1 = new JButton("Disappear");
        b1.addActionListener(new myHandler());
        add(b1);
    }

    class myHandler implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            if (e.getSource() == b) {

                String s = tf1.getText();
                label.setText(s);
                tf1.setText("");
            }

            if (e.getSource() == b1) {
                b.setVisible(false);
            }

        }

    }

    public static void main(String[] args) {

// TODO Auto-generated method stub
```

```
        Test f = new Test();
        f.setTitle("Copy Text");
        f.setSize(400, 150);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLocationRelativeTo(null);;
        f.setVisible(true);

    }

}
```

3) Graded Lab Tasks (Allocated Time 1 Hr 15 min)

Lab Task 1

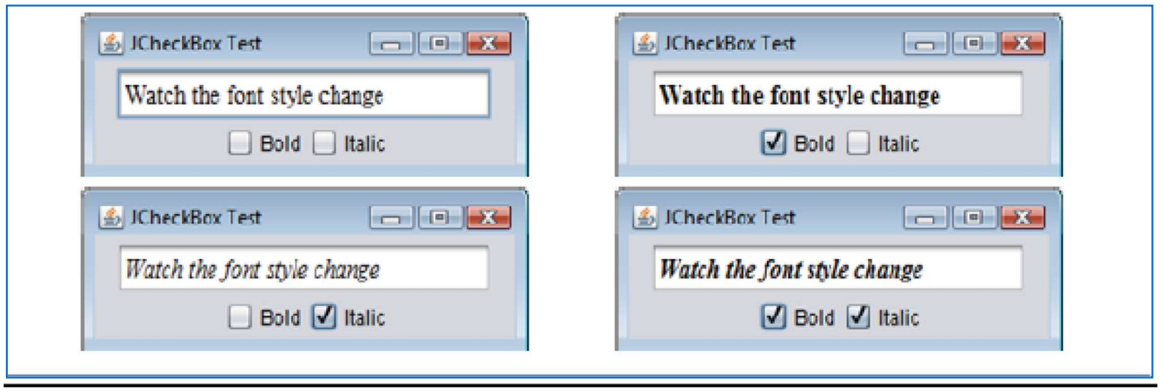
Create a frame with one label, one textbox and a button. Display the information entered in textbox on button click.

Lab Task 2

Create frames as follows:

Lab Task 3

Create Frames as follows:



Lab Task 4

Make a functional nonscientific calculator. Recall the last task of the previous lab.