FAST

National University of Computer and Emerging Sciences Peshawar

Lecture # 19

Software Construction and Development (Java Programming)

Instructor: Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE

الذی علم بالقلم۔ علم الانسان ما لم یعلم۔

Layout Managers in Java (GUI)

Contents

Layout Managers

- Layout Managers are used to form the appearance of your GUI.

- They are concerned with the arrangement of components of GUI. A general question is “why we cannot place components at our desired location (may be using the x,y coordinate position?”.

- The answer is that you can create your GUI without using Layout Managers and you can also do VB style positioning of components at some x,y co-ordinate in Java, but that is generally not advisable if you desire to run the same program on different platforms.

Layout Managers…

- The appearance of the GUI also depends on the underlying platform and to keep that same the responsibility of arranging layout is given to the LayoutManagers so they can provide the same look and feel across different platforms.

Commonly used layout managers are

1. Flow Layout
2. Grid Layout
3. Border Layout
4. Box Layout
5. Card Layout
6. GridBag Layout and so on

Let us discuss the top three in detail one by one with code examples. These top three will meet most of your basic needs

1) Flow Layout

- Position components on line by line basis. Each time a line is filled, a new line is started.

- The size of the line depends upon the size of your frame. If you stretch your frame while your program is running, your GUI will be disturbed.

Example Code Flow Layout

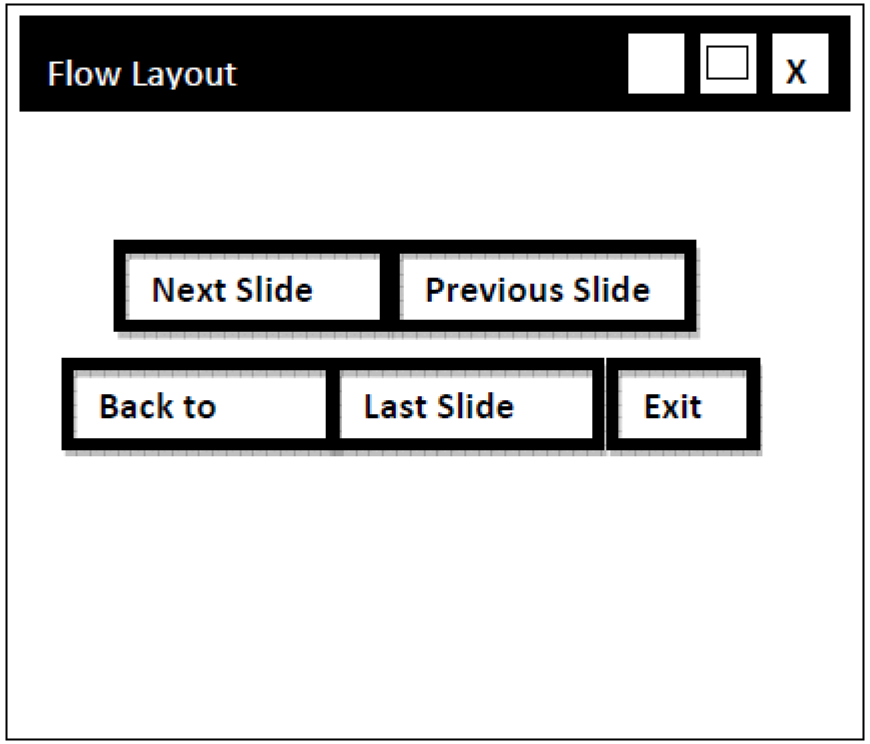
```
// File FlowLayoutTest.

java import java.awt.*;

import javax.swing.*;

public class FlowLayoutTest {

JFrame myFrame ;

JButton b1, b2, b3, b4, b5;

public void initGUI ( ) {            //method used for setting layout of GUI

myFrame = new JFrame("Flow Layout");

Container c = myFrame.getContentPane();
```

Example Code of Flow Layout…

```
c.setLayout( new FlowLayout( ) );
b1 = new JButton(“Next Slide”);
b2 = new JButton(“Previous Slide”);
b3 = new JButton(“Back to Start”);
b4 = new JButton(“Last Slide”);
b5 = new JButton(“Exit”);
c.add(b1);
c.add(b2);
c.add(b3);
c.add(b4);
c.add(b5);
```

Example Code of Flow Layout…

```
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
myFrame.setSize(300,150);
myFrame.setVisible(true);
}       //end initGUI method
public FlowLayoutTest () {      // default constructor
initGUI ();
}
public static void main (String args[ ]) {
FlowLayoutTest flTest = new FlowLayoutTest();
}
} // end of class
```



Grid Layout

- Splits the panel/window into a grid(cells) with given number of rows and columns.

- Forces the size of each component to occupy the whole cell. Size of each component is same .

- Components are added row wise. When all the columns of the first row are get filled the components are then added to the next row.
- Only one component can be added into each cell.

Example Code of Grid Layout

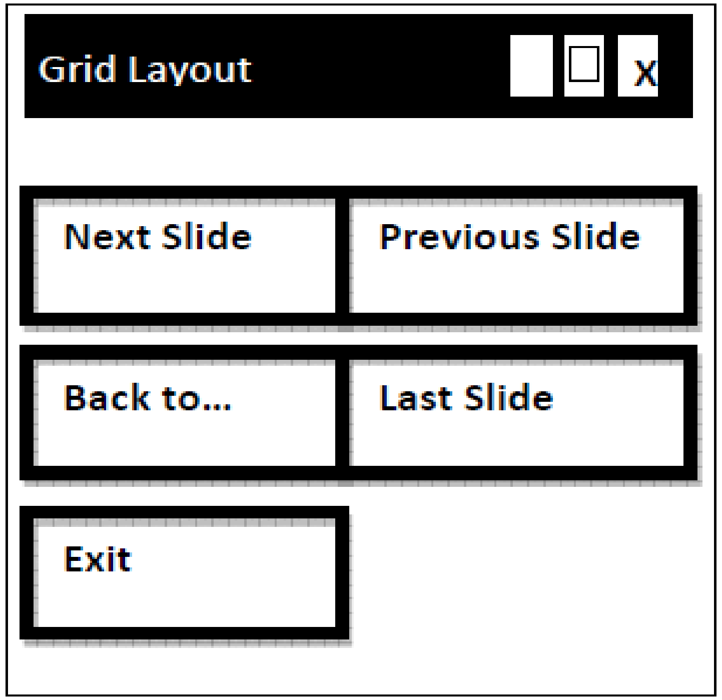
```
// File GridLayoutTest.java
import java.awt.*;
import javax.swing.*;
public class GridLayoutTest {
JFrame myFrame ;
JButton b1, b2, b3, b4, b5;
public void initGUI ( ) {       //method used for setting layout of GUI
myFrame = new JFrame("Grid Layout");
Container c = myFrame.getContentPane();
```

Example Code of Grid Layout…

```
c.setLayout( new GridLayout( 3 , 2 ) );       // 3 rows , 2 cols
b1 = new JButton(“Next Slide”);
b2 = new JButton(“Previous Slide”);
b3 = new JButton(“Back to Start”);
b4 = new JButton(“Last Slide”);
b5 = new JButton(“Exit”);
c.add(b1);
c.add(b2);
c.add(b3);
c.add(b4);
c.add(b5);
```

Example Code of Grid Layout…

```
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
myFrame.setSize(300,150);
myFrame.setVisible(true);
}    //end initGUI method
public GridLayoutTest () {       // default constructor
initGUI ();
}
public static void main (String args[ ]) {
GridLayoutTest glTest = new GridLayoutTest();
}
} // end of class
```



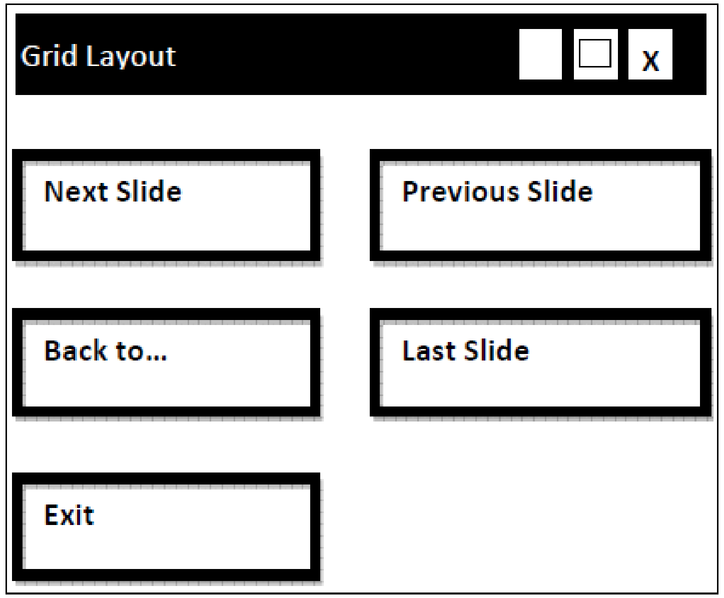
Modification

The grid layout also allows the spacing between cells. To achieve spacing between cells, modify the above program.

Pass additional parameters to the constructor of GridLayout, spaces between rows & spaces between columns as shown below

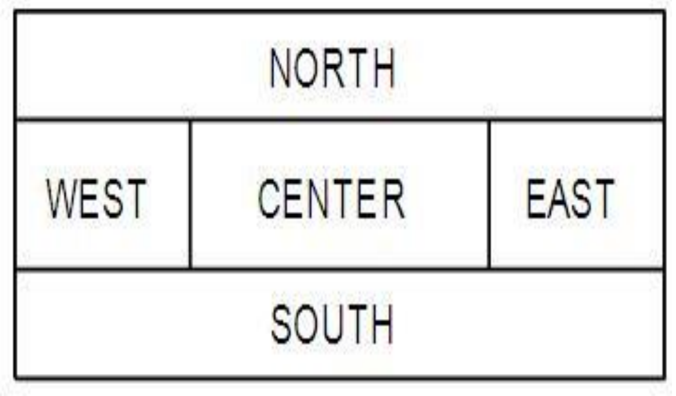
c.setLayout(new GridLayout(3 , 2,10 , 20));

The output is look similar to one given below.



Border Layout

- Divides the area into five regions. North, South, East, West and Center.
- Components are added to the specified region
- If any region not filled, the filled regions will occupy the space but the center region will still appear as background if it contains no component.
- Only one component can be added into each region.



Example code of Border Layout

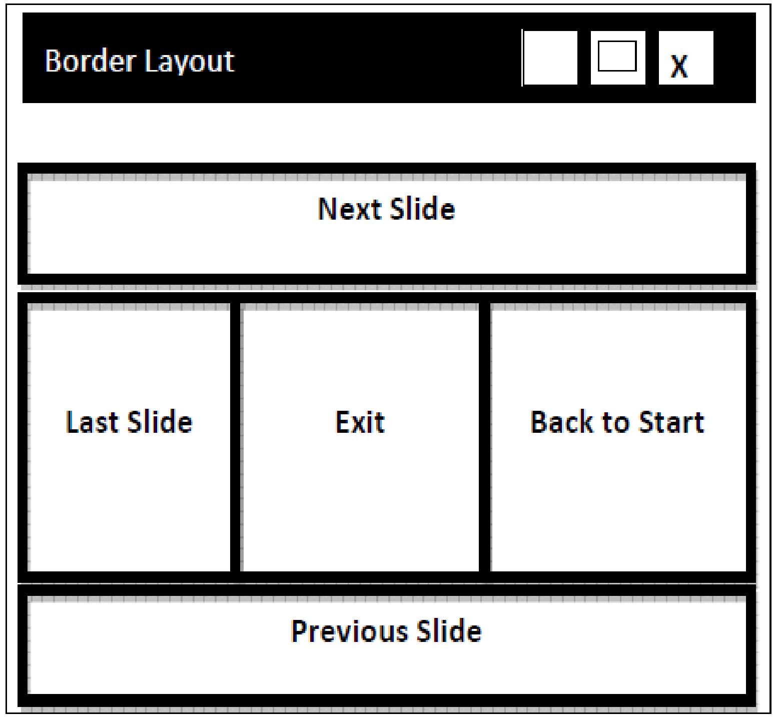
```
// File BorderLayoutTest.
java import java.awt.*;
import javax.swing.*;
public class BorderLayoutTest {
JFrame myFrame ;
JButton b1, b2, b3, b4, b5;
//method used for setting layout of GUI
public void initGUI ( ) {
myFrame = new JFrame(“Border Layout”);
Container c = myFrame.getContentPane();
c.setLayout( new BorderLayout( ) );
```

Example code of Border Layout…

```
b1 = new JButton(“Next Slide”);
b2 = new JButton(“Previous Slide”);
b3 = new JButton(“Back to Start”);
b4 = new JButton(“Last Slide”);
b5 = new JButton(“Exit”);
c.add( b1 , BorderLayout.NORTH );
c.add( b2 , BorderLayout.SOUTH );
c.add( b3 , BorderLayout.EAST );
c.add( b4 , BorderLayout.WEST );
c.add( b5 , BorderLayout.CENTER);
```

Example code of Border Layout…

```
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
myFrame.setSize(300,150);
myFrame.setVisible(true);
}        //end initGUI method
public BorderLayoutTest () {         // default constructor
initGUI ();
}
public static void main (String args[ ]) {
BorderLayoutTest glTest = new BorderLayoutTest();
}      } // end of class
```



Border Layout

Points to Remember

- Revisit the code of adding components, we specify the region in which we want to add component or otherwise they will not be visible.

- Consider the following segment of code: BorderLayout.NORTH, as you guessed correctly NORTH is a constant (final) defined in BorderLayout class public access modifier. Similarly the other ones are defined. Now you understand why so much emphasis has been made on following the naming conventions.

Java JPanel

The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class.

It doesn't have title bar.

JPanel class declaration

public class JPanel **extends** JComponent **implements** Accessible

Java Jpanel…

Commonly used Constructors:

Constructor	Description
JPanel()	It is used to create a new JPanel with a double buffer and a flow layout.
JPanel(boolean isDoubleBuffered)	It is used to create a new JPanel with FlowLayout and the specified buffering strategy.
JPanel(LayoutManager layout)	It is used to create a new JPanel with the specified layout manager.

Java Jpanel…

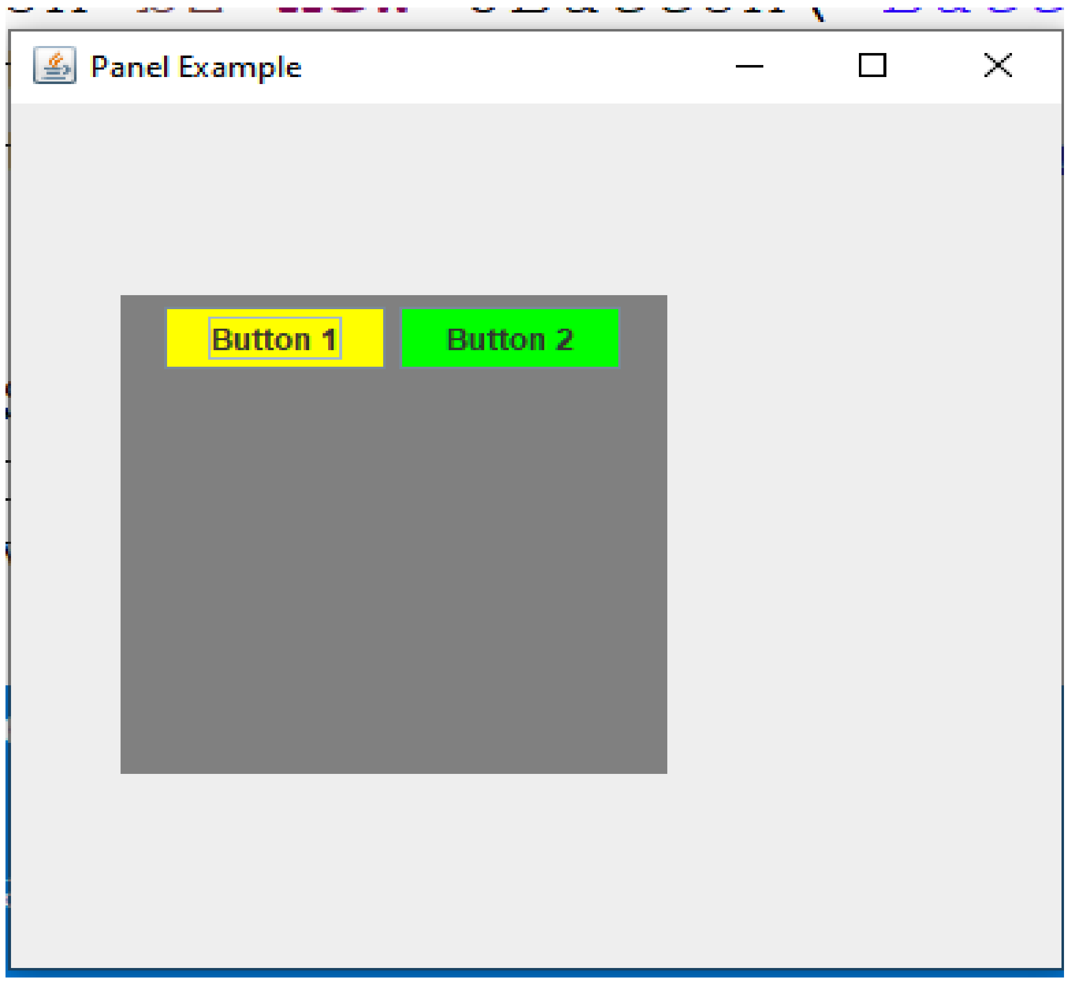
```
package java_gui_work;
import java.awt.*;
import javax.swing.*;
public class PanelExample {
    PanelExample()
      {
      JFrame f= new JFrame("Panel Example");
      JPanel panel=new JPanel();
      panel.setBounds(40,80,200,200);
      panel.setBackground(Color.gray);
```

Java Jpanel…

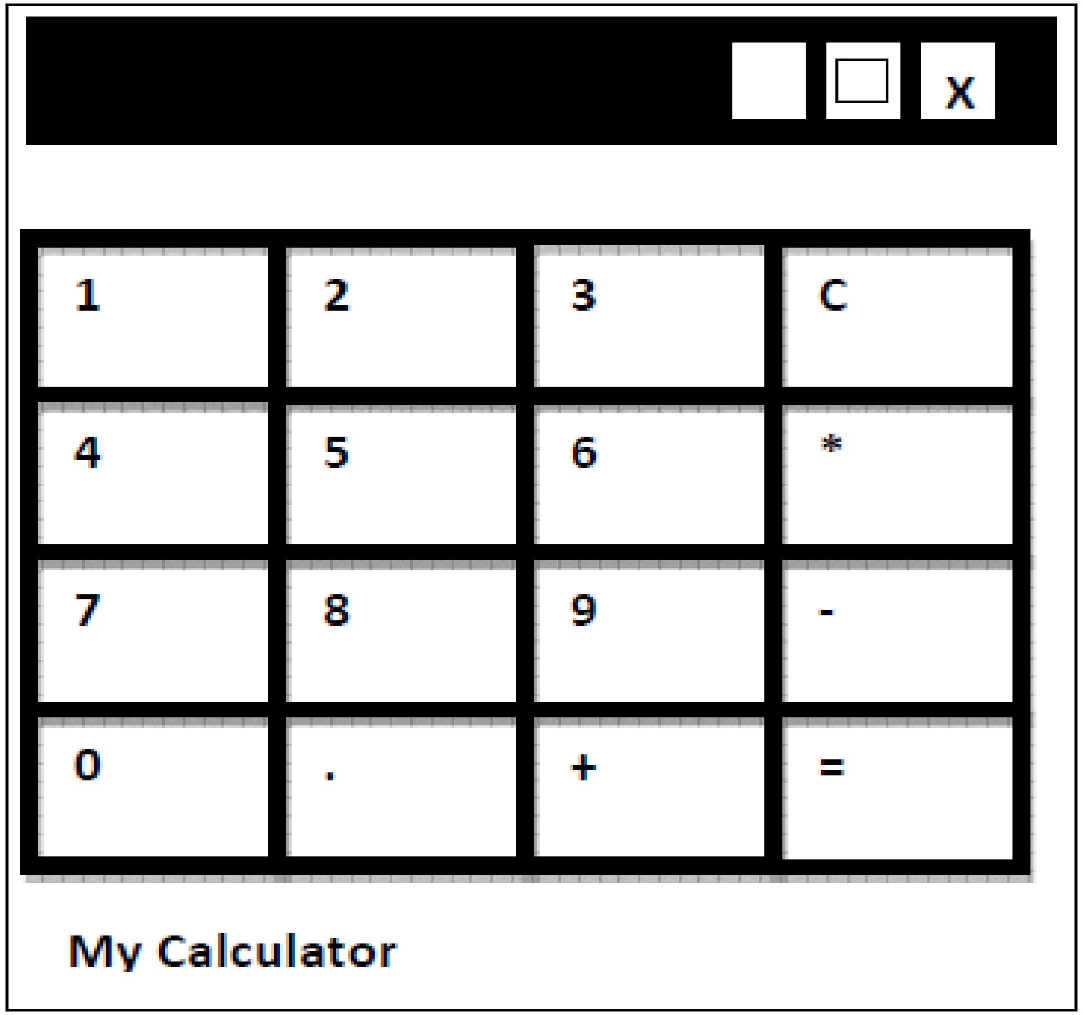
```
    JButton b1=new JButton("Button 1");
    b1.setBounds(50,100,80,30);
    b1.setBackground(Color.yellow);
    JButton b2=new JButton("Button 2");
    b2.setBounds(100,100,80,30);
    b2.setBackground(Color.green);
    panel.add(b1);
    panel.add(b2);
    f.add(panel);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
```

Java Jpanel…

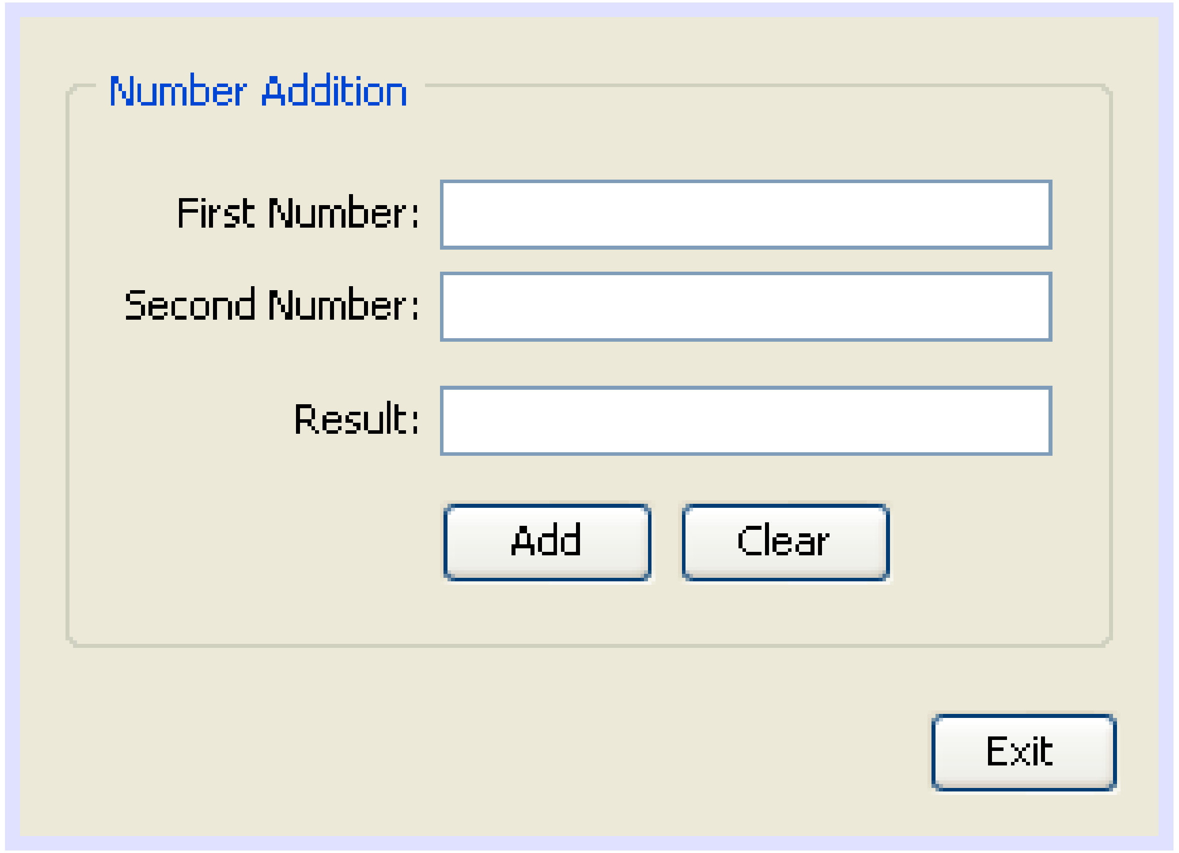
```
public static void main(String args[])
        {
        new PanelExample();
        }
}   // PanelExample Class Body Closed
```



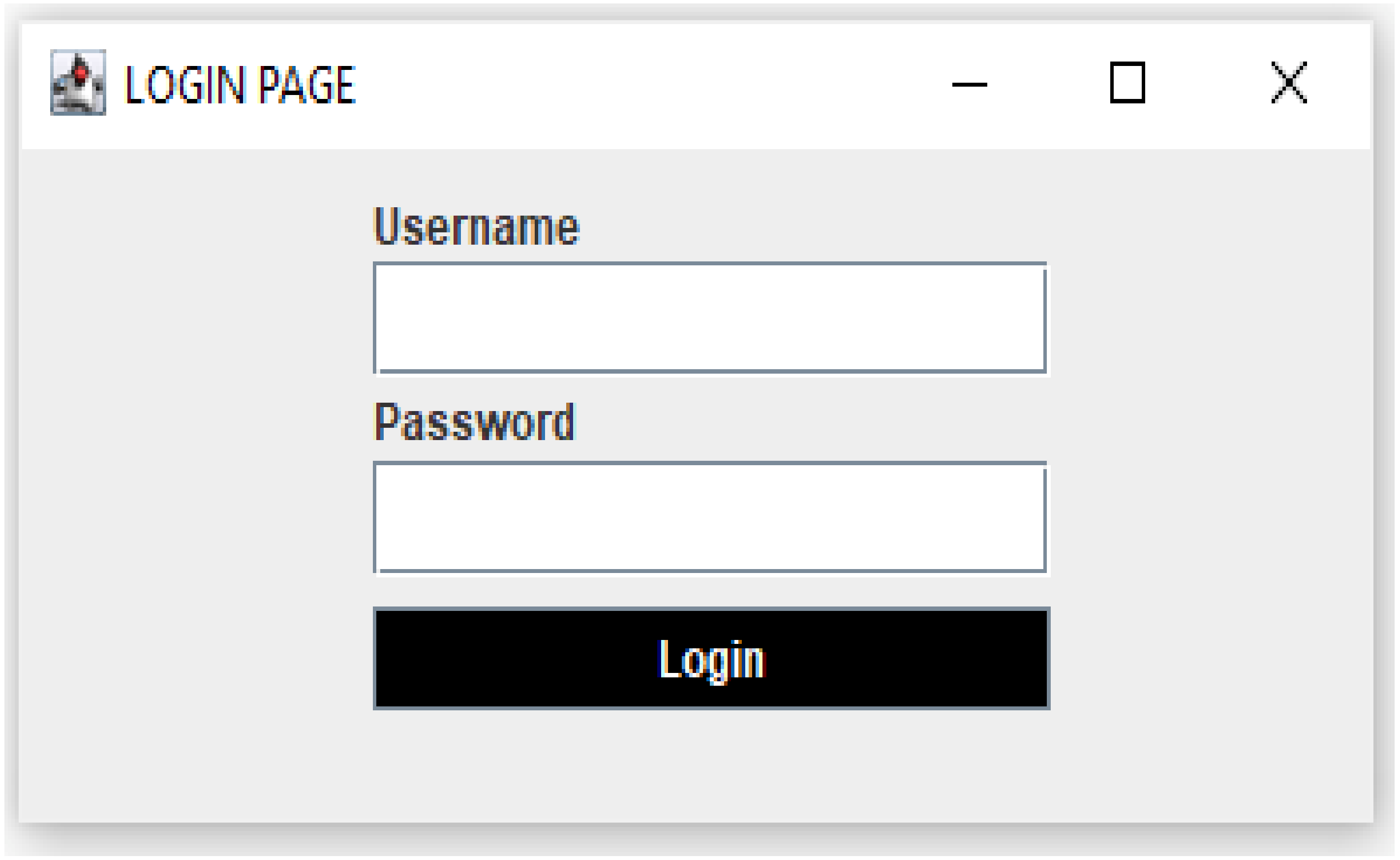
Task # 01



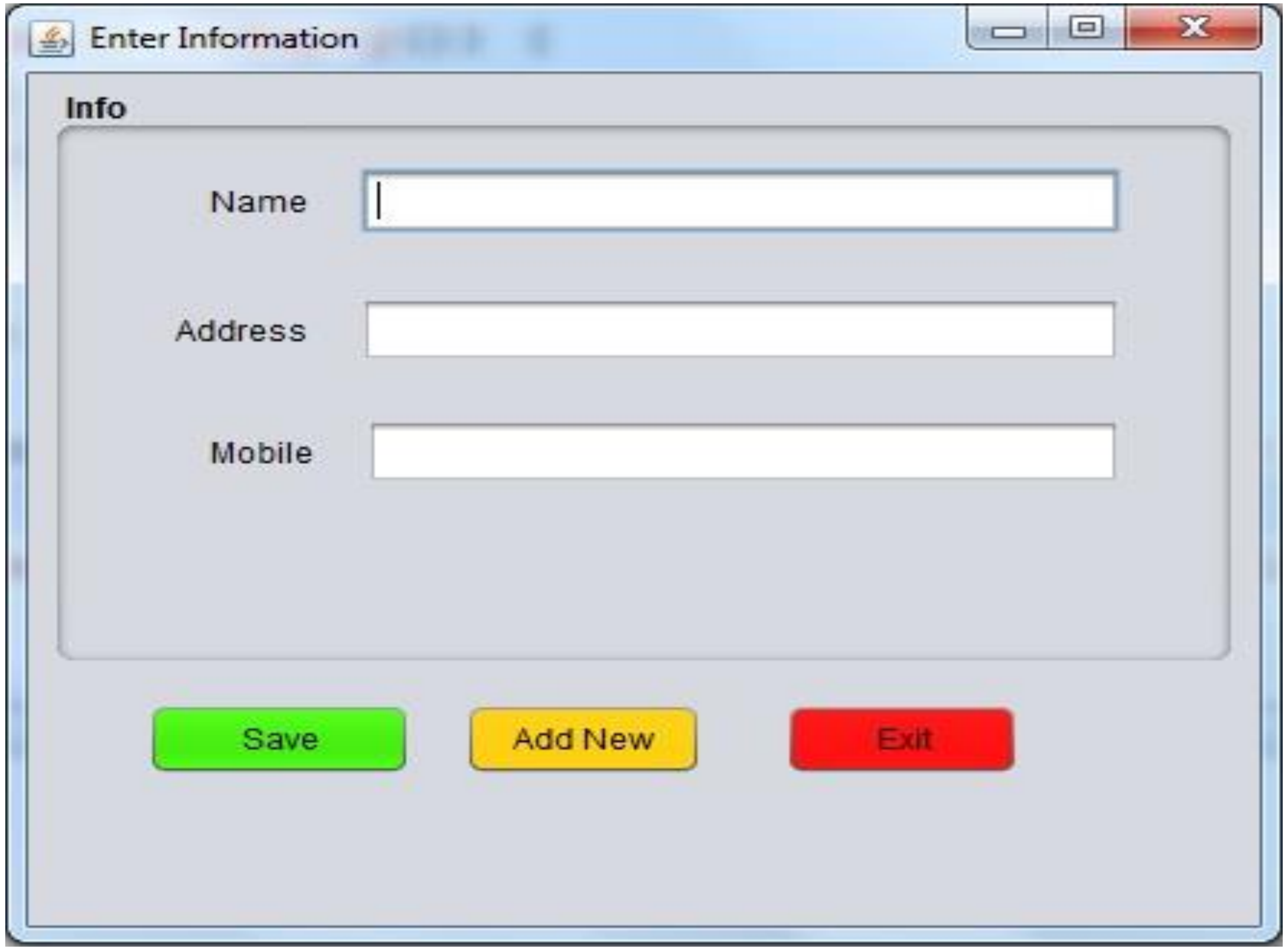
Task # 02


Number Addition
First Number:
Second Number:
Result:
Add
Clear
Exit

Task # 03



Task # 04


Enter Information
Info
Name
Address
Mobile
Save
Add New
Exit

References

https://www.javatpoint.com/java-jpanel