Swing :

It is a part of Java foundation class (JFC) which is lightweight and platform independent. It is used for creating window based applications. It is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is build on top of the AWT API and entirely written in java. It becomes easier to build applications since we already have GUI components like button, checkbox etc.

Graphical User Interface (GUI) acts as an intermediate and facilitates the interaction between the user and the program using buttons, pictures, and other graphical entities. The Swing toolkit in Java has a rich set of graphical components for building GUI interfaces and making the program more interactive.

All the swing classes are present in **javax.swing** package. JFC is a set of libraries that supports programmers in creating applications using java. Using swing one can develop a standalone desktop application.

Container Class

A component is a self-contained visual entity that can be customized and inserted into applications. A container is a special type of component that can hold a group of components. Any class which has other components in it is called as a container class. For building GUI applications at least one container class is necessary.

Containers are of two types:

Top-level Containers: They inherit components and containers of AWT. Top-level containers cannot be contained within other containers. They can be directly viewed on a desktop. **Example: - JFrame, JDialog.**

JFrame - Frame is a fully functioning window with icons and texts.

JDialog - Dialog is a pop-up window**.**

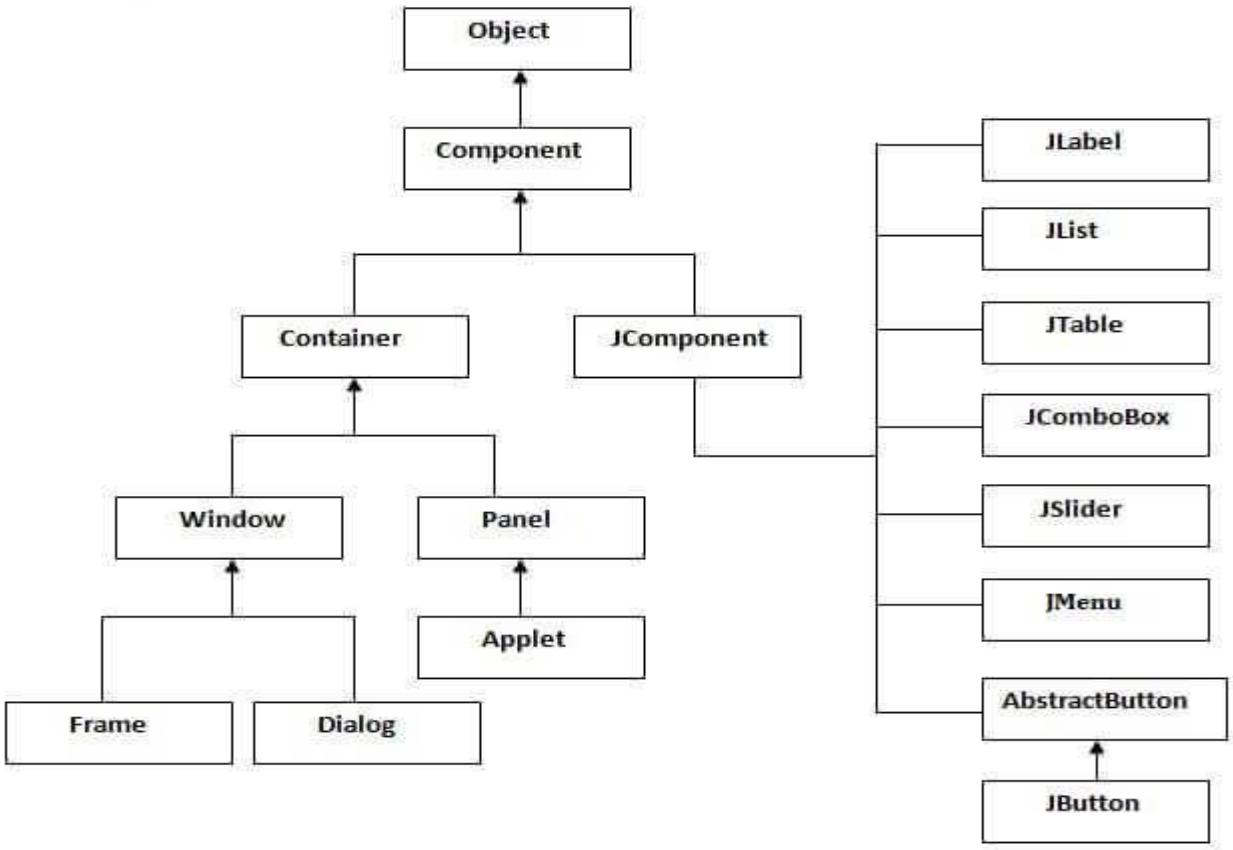
Light-weight Containers : Light-weight containers are general-purpose containers. They inherit JComponent class and they can be displayed within the top-level containers.

Example : - JPanel, JScrollPane, JToolBar.

JPanel - The panel is a container used to organize components in a window.

Difference Between AWT and Swing

AWT	SWING
Platform Dependent	Platform Independent
Does not follow MVC	Follows MVC
Lesser Components	More powerful components
Does not support pluggable look and feel	Supports pluggable look and feel
Heavyweight	Lightweight



JFrame

JFrame is a container class inside javax.swing. It is treated as the main window, all the other components are added to the frame such as labels, buttons, panels, etc.

JFrame() : creates a new JFrame that is initially invisible.

JButton Class

It is used to create a labelled button. Using the ActionListener it will result in some action when the button is pushed. It inherits the AbstractButton class and is platform independent.

JTextField Class

It inherits the JTextComponent class and it is used to allow editing of single line text.

JScrollBar Class

It is used to add scroll bar, both horizontal and vertical.

JPanel Class

It inherits the JComponent class and provides space for an application which can attach any other component.

Example: will show difference with panel or without panel

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

public class panel_ex {
   panel_ex() {
      JFrame a1 = new JFrame("without panel");
      JButton b1 = new JButton("click me");
      b1.setBounds(60, 50, 80, 40);
      a1.add(b1);
      a1.setSize(400, 400);
      a1.setLayout(null);
      a1.setVisible(true);
```

```
      JFrame a = new JFrame("panel");
      JPanel p = new JPanel();
      p.setBounds(40, 70, 200, 200); // panel doesn't have border.. so we can't see panel border
      JButton b = new JButton("click me");
      b.setBounds(60, 50, 80, 40);
      p.add(b);
      a.add(p);
      a.setSize(400, 400);
      a.setLayout(null);
      a.setVisible(true);
```

```
    }

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

JMenu Class

It inherits the JMenuItem class, and is a pull down menu component which is displayed from the menu bar.

JList Class

It inherits JComponent class, the object of JList class represents a list of text items.

JLabel Class

It is used for placing text in a container. It also inherits JComponent class.

JComboBox Class

It inherits the JComponent class and is used to show pop up menu of choices.

Program 1: Sample example on swing components:

```
import javax.swing.*;

public class swing_example {
    swing_example() {
    // set frame with title
    JFrame a = new JFrame("Java Swing Example");
    // set button
    JButton b = new JButton("click me");
    b.setBounds(30, 30, 100, 30);
    // set textfield
    JTextField tf = new JTextField("text field");
    tf.setBounds(150, 30, 100, 30);
    // set scroll pane
    JScrollBar scr = new JScrollBar();
    scr.setBounds(420, 30, 40, 420);
    // set menu
```

```
JMenu menu;
JMenuItem a1, a2;
menu = new JMenu("options");
JMenuBar m1 = new JMenuBar();
a1 = new JMenuItem("example");
a2 = new JMenuItem("example1");
menu.add(a1);
menu.add(a2);
m1.add(menu);
// set list
DefaultListModel<String> l = new DefaultListModel<>();
l.addElement("first item");
l.addElement("second item");
JList<String> jl = new JList<>(l);
jl.setBounds(30, 80, 75, 75);
// set label
JLabel b1;
b1 = new JLabel("this is sample code");
b1.setBounds(125, 80, 130, 30);
// set combo box
String courses[] = { "core java", "advance java", "java servlet" };
JComboBox cb = new JComboBox(courses);
cb.setBounds(270, 80, 120, 30);
// add comppnants to fram and set frame
a.add(b);
a.add(tf);
a.add(scr);
a.setJMenuBar(m1);
a.add(jl);
a.add(b1);
a.add(cb);
a.setSize(500, 500);
a.setLayout(null);
a.setVisible(true);
```

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

Image on button

```
import javax.swing.*;

public class ButtonExample {
    ButtonExample() {
        JFrame f = new JFrame("Button Example");
        JButton b = new JButton(new ImageIcon("image/register.png"));
        b.setBounds(100, 100, 200, 80);
        f.add(b);
        f.setSize(600, 600);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new ButtonExample();
    }
}
```

Another code for set image on button

```
import javax.swing.*;

import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
```

```
public class ButtonExample {
   ButtonExample() throws IOException {
      JFrame f = new JFrame("Button Example");
      // to set image size.... use follow code and set it in button
      BufferedImage bufferedImage = ImageIO.read(new File("image/register.png"));
      Image image = bufferedImage.getScaledInstance(75, 40, Image.SCALE_DEFAULT);
      ImageIcon icon = new ImageIcon(image);
      JButton b = new JButton(icon);
      b.setBounds(100, 100, 200, 80);
      f.add(b);
      f.setSize(600, 600);
      f.setLayout(new FlowLayout());
      f.setVisible(true);
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
   public static void main(String[] args) throws IOException {
      new ButtonExample();
   }
}
```

JRadioButton

This component allows the user to select only one item from a group item. By using the JRadioButton component you can choose one option from multiple options.

Syntax: JRadioButton obj= new JRadioButton();

JRadioButton Constructors

1. **JRadioButton():** It is used to create an unselected radio button with no text.
2. **JRadioButton(Label):** It is used to create an unselected radio button with specified text.
3. **JRadioButton(Label, boolean):** It is used to create a radio button with the specified text and selected status.

JCheckBox

This component allows the user to select multiple items from a group of items. It is used to create a CheckBox. It is used to turn an option ON or OFF.

JCheckBox Constructors

1. **JCheckBox():** It is used to create an initially unselected checkbox button with no text, no icon.
2. **JCheckBox(Label):** It is used to create an initially unselected checkbox with text.
3. **JCheckBox(Label, boolean):** It is used to create a checkbox with text and specifies whether or not it is initially selected.
4. **JCheckBox(Action a):** It is used to create a checkbox where properties are taken from the Action supplied.

JTextField

The JTextField component allows the user to type some text in a single line. It basically inherits the JTextComponent class.

Syntax: JTextField obj = new JTextField();

JTextField Constructors

1. **JTextField():** It is used to create a new Text Field.
2. **JTextField(String text):** It is used to create a new Text Field initialized with the specified text.
3. **JTextField(String text, int columns):** It is used to create a new Text field initialized with the specified text and columns.
4. **JTextField(int columns):** It is used to create a new empty TextField with the specified number of columns.

JTextArea

The JTextArea component allows the user to type the text in multiple lines. It also allows the editing of multiple-line text. It basically inherits the JTextComponent class.

Syntax: JTextArea obj= new JTextArea();

JTextArea Constructors

1. **JTextArea():** It is used to create a text area that displays no text initially.
2. **JTextarea(String s):** It is used to create a text area that displays specified text initially.
3. **JTextArea(int row, int column):** It is used to create a text area with the specified number of rows and columns that display no text initially.
4. **JTextarea(String s, int row, int column):** It is used to create a text area with the specified number of rows and columns that display specified text.

Java JTable

The JTable class is used to display data in tabular form. It is composed of rows and columns.

Commonly used Constructors:

Constructor	Description
JTable()	Creates a table with empty cells.
JTable(Object[][] rows, Object[] columns)	Creates a table with the specified data.

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

public class radiobutton extends JFrame implements ActionListener {
   JRadioButton eng, doc;
   ButtonGroup bg;
   JTextField jtf;
   JCheckBox bcd, ccb, acb;
   JTextArea jta;
   radiobutton() {
      eng = new JRadioButton("Engineer");
      doc = new JRadioButton("Doctor");
      bg = new ButtonGroup();
      bg.add(eng);
```

```
    bg.add(doc);
    jtf = new JTextField(20);
    bcd = new JCheckBox("Bike");
    ccb = new JCheckBox("Car");
    acb = new JCheckBox("Aeroplane");
    jta = new JTextArea(3, 20);
    Container c = this.getContentPane();
    // setLayout(new FlowLayout( default set in center
    // c.setLayout(new FlowLayout(FlowLayout.LEFT)); set in left align, if we write
    // right set in right align
    c.setLayout(new FlowLayout(FlowLayout.LEFT, 20, 25));
    // Registering the listeners with the components
    eng.addActionListener(this);
    doc.addActionListener(this);
    bcd.addActionListener(this);
    ccb.addActionListener(this);
    acb.addActionListener(this);
    c.add(eng);
    c.add(doc);
    c.add(jtf);
    c.add(bcd);
    c.add(ccb);
    c.add(acb);
    c.add(jta);
    this.setVisible(true);
    this.setSize(500, 500);
    this.setTitle("Selection Example");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == eng) {
        jtf.setText("You are an Engineer");
    }
    if (ae.getSource() == doc) {
```

```
            jtf.setText("You are an Doctor");
        }
        String str = " ";
        if (bcd.isSelected()) {
            str += "Bike\n";
        }
        if (ccb.isSelected()) {
            str += "Car\n";
        }
        if (acb.isSelected()) {
            str += "Aeroplane\n";
        }
        jta.setText(str);
    }
    public static void main(String[] args) {
        new radiobutton();
    }
}
```

Example of java table

```
import javax.swing.*;
public class java_table {
    JFrame f;
    java_table() {
```

```
        f = new JFrame();
        String data[][] = { { "101", "Amit", "670000" },
                { "102", "Jai", "780000" },
                { "103", "Sachin", "50000" },
                { "104", "Sumit", "60000" },
                { "105", "Nilay", "7800" },
                { "106", "Neeta", "10000" } };
        String column[] = { "ID", "NAME", "SALARY" };
        JTable jt = new JTable(data, column);
        jt.setBounds(30, 40, 200, 300);
        JScrollPane sp = new JScrollPane(jt);
        f.add(sp);
        f.setSize(400, 400);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new java_table();
    }
}
```

Java JOptionPane:

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

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

public class OptionPaneExample extends WindowAdapter {
    JFrame f;
    OptionPaneExample() {
        f = new JFrame();
        f.addWindowListener(this);
        f.setSize(300, 300);
```

```
    f.setLayout(null);
    f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    f.setVisible(true);
  }
  public void windowClosing(WindowEvent e) {
    int a = JOptionPane.showConfirmDialog(f, "Are you sure?");
    if (a == JOptionPane.YES_OPTION) {
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
  }
  public static void main(String[] args) {
    new OptionPaneExample();
  }
}
```

Java JSpinner

A single line input field that lets the user select a number or an object value from an ordered sequence. Spinners typically provide a pair of tiny arrow buttons for stepping through the elements of the sequence. The keyboard up/down arrow keys also cycle through the elements. The user may also be allowed to type a (legal) value directly into the spinner. Although combo boxes provide similar functionality, spinners are sometimes preferred because they don't require a drop down list that can obscure important data.

Commonly used Contructors:

Constructor	Description
JSpinner()	It is used to construct a spinner with an Integer SpinnerNumberModel with initial value 0 and no minimum or maximum limits.

JSpinner(SpinnerModel model)	It is used to construct a spinner for a given model.

Commonly used Methods:

Method	Description
void addChangeListener(ChangeListener listener)	It is used to add a listener to the list that is notified each time a change to the model occurs.
Object getValue()	It is used to return the current value of the model.

```
import javax.swing.*;
import javax.swing.event.*;

public class SpinnerExample {
   public static void main(String[] args) {
      JFrame f = new JFrame("Spinner Example");
      final JLabel label = new JLabel();
      label.setHorizontalAlignment(JLabel.CENTER);
      label.setSize(250, 100);
      SpinnerModel value = new SpinnerNumberModel(5, // initial value
            0, // minimum value
            10, // maximum value
            1); // step
      JSpinner spinner = new JSpinner(value);
      spinner.setBounds(100, 100, 50, 30);
      f.add(spinner);
      f.add(label);
      f.setSize(300, 300);
      f.setLayout(null);
      f.setVisible(true);
      spinner.addChangeListener(new ChangeListener() {
```

```
        public void stateChanged(ChangeEvent e) {
            label.setText("Value : " + ((JSpinner) e.getSource()).getValue());
        }
      });
   }
}
```

Java JScrollPane

A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.

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

public class scorllpane {
   public static void main(String[] args) {
      final JFrame frame = new JFrame("JScrollbar Demo");
      final JLabel label = new JLabel();
      JScrollBar hbar = new JScrollBar(JScrollBar.HORIZONTAL, 30, 20, 0, 500);
      JScrollBar vbar = new JScrollBar(JScrollBar.VERTICAL, 30, 40, 0, 500);
      class MyAdjustmentListener implements AdjustmentListener {
         public void adjustmentValueChanged(AdjustmentEvent e) {
            label.setText("Slider's position is " + e.getValue());
            frame.repaint();
         }
      }
      hbar.addAdjustmentListener(new MyAdjustmentListener());
      vbar.addAdjustmentListener(new MyAdjustmentListener());
      frame.setLayout(new BorderLayout());
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(300, 200);
```

```
        frame.getContentPane().add(label);
        frame.getContentPane().add(hbar, BorderLayout.SOUTH);
        frame.getContentPane().add(vbar, BorderLayout.EAST);
        frame.getContentPane().add(label, BorderLayout.CENTER);
        frame.setVisible(true);
    }
}
```

JPasswordField Swing Control in Java

It is a text component specialized for password entry. It allows the editing of a single line of text. It basically inherits the JTextField class.

Syntax: JPasswordField obj= new JPasswordField();

JPasswordFiled Constructors

1. **JPasswordField():** It is used to construct a new JPasswordField, with a default document, null starting text string, and column width.
2. **JPasswordField(int columns):** It is used to construct a new empty JPasswordField with the specified number of columns.
3. **JPasswordField(String text):** It is used to construct a new JPasswordField initialized with the specified text.
4. **JPasswordField(String text, int columns):** It is used to construct a new JPasswordField initialized with the specified text and columns.

```
import javax.swing.*;

public class JPassword_ex {
    public static void main(String[] args) {
        JFrame f = new JFrame("Password Field Example");
        JPasswordField value = new JPasswordField();
        JLabel l1 = new JLabel("Password:");
        l1.setBounds(20, 100, 80, 30);
        value.setBounds(100, 100, 100, 30);
```

```
        f.add(value);
        f.add(l1);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

Java JToggleButton

JToggleButton is used to create toggle button, it is two-states button to switch on or off.

```
import java.awt.FlowLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.JFrame;
import javax.swing.JToggleButton;

public class toogle_button extends JFrame implements ItemListener {
    public static void main(String[] args) {
        new toogle_button();
    }
    private JToggleButton button;
    toogle_button() {
        setTitle("JToggleButton with ItemListener Example");
        setLayout(new FlowLayout());
        setJToggleButton();
        setAction();
        setSize(200, 200);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    private void setJToggleButton() {
        button = new JToggleButton("ON");
        add(button);
```

```
    }
    private void setAction() {
        button.addItemListener(this);
    }
    public void itemStateChanged(ItemEvent eve) {
        if (button.isSelected())
            button.setText("OFF");
        else
            button.setText("ON");
    }
}
```

Program for Generate Random number:

```
import java.awt.Font;
import java.awt.event.*;
import java.util.Random;
import javax.swing.*;
public class randomno {
    public static void main(String[] args) {
        // Declare a Frame object
        JFrame frame = new JFrame();
        // Disable default frame layout
        frame.setLayout(null);
        // Create a button object
        JButton btn = new JButton("Generate Number");
        // Set the font type of the button
        btn.setFont(new Font("Verdana", Font.BOLD, 20));
        // Set the button position
        btn.setBounds(45, 50, 250, 40);
        // Add button to the frame
        frame.add(btn);
        // Add the action listener for the button
        btn.addActionListener(new ActionListener() {
```

```
        public void actionPerformed(ActionEvent e) {
            // Create a random object
            Random num = new Random();
            // Generate a random number
                        //
            int n = num.nextInt(1000000) + 1;
            // Convert integer to string
            String rnum = String.valueOf(n);
            // Display the random number in the message box
            JOptionPane.showMessageDialog(frame, "Generated Number: " + rnum);
        }
    });
    // Set the title
    frame.setTitle("Java Swing Example-14");
    // Set the window size
    frame.setSize(350, 200);
    // Disable the resize option
    frame.setResizable(false);
    // Set window position
    frame.setLocationRelativeTo(null);
    // Make the window visible
    frame.setVisible(true);
  }
}
```

Layout Manager:

The LayoutManagers are used to arrange components in a particular manner. The Java LayoutManagers facilitates us to control the positioning and size of the components in GUI forms. LayoutManager is an interface that is implemented by all the classes of layout managers.

There are the following classes that represent the layout managers:

1. java.awt.BorderLayout
2. java.awt.FlowLayout

3. java.awt.GridLayout
4. Java.awt.GridbagLayout
5. java.awt.CardLayout
6. javax.swing.BoxLayout

Border Layout

The default layout manager for every JFrame is BorderLayout. It places components in upto five places which is top, bottom, left, right and center.

Constructors of BorderLayout class:

- **BorderLayout():** creates a border layout but with no gaps between the components.
- **BorderLayout(int hgap, int vgap):** creates a border layout with the given horizontal and vertical gaps between the components.

```
// Java program to illustrate the BorderLayout
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.Button;
import java.awt.Color;
// class extends JFrame
public class borderlauout_ex extends JFrame {
   // Constructor of BorderDemo class.
   public borderlauout_ex() {
      // set the layout
      setLayout(new BorderLayout());
      // set the background
      setBackground(Color.red);
     // creates Button (btn1)
      Button btn1 = new Button("north");
      // creates Button (btn2)
      Button btn2 = new Button("south");
```

```
        // creates Button (btn3)
        Button btn3 = new Button("east");
        // creates Button (btn4)
        Button btn4 = new Button("west");
        // creates Button (btn5)
        Button btn5 = new Button("center");
        // Adding JButton "btn1" on JFrame.with position
        add(btn1, "North");
        // Adding JButton "btn2" on JFrame.
        add(btn2, "South");
        // Adding JButton "btn3" on JFrame.
        add(btn3, "East");
        // Adding JButton "btn4" on JFrame.
        add(btn4, "West");
        // Adding JButton "btn5" on JFrame.
        add(btn5, "Center");
        // function to set the title
        setTitle("Learning a Border Layout");
        // Function to set size of JFrame.
        setSize(350, 300);
        // Function to set visible status of JFrame
        setVisible(true);
    }
    // Main Method
    public static void main(String args[]) {
        // calling the constructor
        new borderlauout_ex();
    }
}
```

Java GridLayout

The Java GridLayout class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

Constructors of GridLayout class

1. **GridLayout():** creates a grid layout with one column per component in a row.
2. **GridLayout(int rows, int columns):** creates a grid layout with the given rows and columns but no gaps between the components.
3. **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

The GridLayout() constructor creates only one row.

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

public class gridlayout_ex {
   JFrame frameObj, f1, f2;
```

```
   // constructor
   gridlayout_ex() {
      frameObj = new JFrame("simpple grid");
      // creating 9 buttons
      JButton btn1 = new JButton("1");
      JButton btn2 = new JButton("2");
      JButton btn3 = new JButton("3");
      JButton btn4 = new JButton("4");
      JButton btn5 = new JButton("5");
      JButton btn6 = new JButton("6");
```

```
      // adding buttons to the frame since, we are using the parameterless
      // constructor, therfore;
      // the number of columns is equal to the number of buttons we are adding to the
      // frame. The row count remains one.
      frameObj.add(btn1);
      frameObj.add(btn2);
      frameObj.add(btn3);
      frameObj.add(btn4);
```

```
frameObj.add(btn5);
frameObj.add(btn6);
```

```
// setting the grid layout using the parameterless constructor
frameObj.setLayout(new GridLayout());
```

```
frameObj.setSize(300, 300);
frameObj.setVisible(true);
```

```
// another grid with matrix form
f1 = new JFrame("buttons set in row and col form ");
JButton b1 = new JButton("1");
JButton b2 = new JButton("2");
JButton b3 = new JButton("3");
JButton b4 = new JButton("4");
JButton b5 = new JButton("5");
JButton b6 = new JButton("6");
JButton b7 = new JButton("7");
JButton b8 = new JButton("8");
JButton b9 = new JButton("9");
// adding buttons to the frame
f1.add(b1);
f1.add(b2);
f1.add(b3);
f1.add(b4);
f1.add(b5);
f1.add(b6);
f1.add(b7);
f1.add(b8);
f1.add(b9);
// setting grid layout of 3 rows and 3 columns
f1.setLayout(new GridLayout(4, 2));
f1.setSize(300, 300);
f1.setVisible(true);
```

```
        f2 = new JFrame("with gape between buttons ");
        // creating 9 buttons
        JButton bt1 = new JButton("1");
        JButton bt2 = new JButton("2");
        JButton bt3 = new JButton("3");
        JButton bt4 = new JButton("4");
        JButton bt5 = new JButton("5");
        JButton bt6 = new JButton("6");
        JButton bt7 = new JButton("7");
        JButton bt8 = new JButton("8");
        JButton bt9 = new JButton("9");
        // adding buttons to the frame since, we are using the parameterless
        // constructor, therefore; the number of columns is equal to the number of buttons we
        // are adding to the frame. The row count remains one.
        f2.add(bt1);
        f2.add(bt2);
        f2.add(bt3);
        f2.add(bt4);
        f2.add(bt5);
        f2.add(bt6);
        f2.add(bt7);
        f2.add(bt8);
        f2.add(bt9);
        // setting the grid layout a 3 * 3 grid is created with the horizontal gap 20
        // and vertical gap 25
        f2.setLayout(new GridLayout(3, 3, 20, 25));
        f2.setSize(300, 300);
        f2.setVisible(true);
    }
    // main method
    public static void main(String argvs[]) {
        new gridlayout_ex();
    }
}
```

Card Layout:

The class **CardLayout** arranges each component in the container as a card. Only one card is visible at a time, and the container acts as a stack of cards.

Class Constructors

1	**CardLayout()** Creates a new card layout with the gaps of size zero.
2	**CardLayout(int hgap, int vgap)** Creates a new card layout with the specified horizontal and vertical gaps.

Card example:

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.*;
public class cardLayout_java extends JFrame {
   // Initializing the value of currCard to 1 .
   private int currCard = 1;
   // Declaring of objects of the CardLayout class.
   private CardLayout cObj1;
   // constructor of the class
   public cardLayout_java() throws IOException {
      // Method to set the Title of the JFrame
      setTitle("Card Layout Methods");
      // Method to set the visibility of the JFrame
      setSize(700, 700);
      // Creating an Object of the "Jpanel" class
      JPanel jp = new JPanel();
```

```
// Initializing of the object "cObjl" of the CardLayout class.
cObjl = new CardLayout();
// setting the layout
jp.setLayout(cObjl);
// Initializing the object "jPanel1" of the JPanel class.
JPanel jP1 = new JPanel();
JPanel jP2 = new JPanel();
JPanel jP3 = new JPanel();
JPanel jP4 = new JPanel();
// taking image in card
BufferedImage buImg1 = ImageIO.read(new File("image/scene.jpg"));
Image image1 = buImg1.getScaledInstance(500, 400, Image.SCALE_DEFAULT);
ImageIcon img1 = new ImageIcon(image1);
// Initializing the object "jl1" of the JLabel class.
JLabel jL1 = new JLabel(img1);
BufferedImage buImg2 = ImageIO.read(new File("image/flower.jpg"));
Image image2 = buImg2.getScaledInstance(500, 400, Image.SCALE_DEFAULT);
ImageIcon img2 = new ImageIcon(image2);
JLabel jL2 = new JLabel(img2);
BufferedImage buImg3 = ImageIO.read(new File("image/green.jpg"));
Image image3 = buImg3.getScaledInstance(500, 400, Image.SCALE_DEFAULT);
ImageIcon img3 = new ImageIcon(image3);
JLabel jL3 = new JLabel(img3);
BufferedImage buImg4 = ImageIO.read(new File("image/tiger.jpg"));
Image image4 = buImg4.getScaledInstance(500, 400, Image.SCALE_DEFAULT);
ImageIcon img4 = new ImageIcon(image4);
JLabel jL4 = new JLabel(img4);
// Adding JLabel "jLabel1" to the JPanel "jPanel1".
jP1.add(jL1);
jP2.add(jL2);
jP3.add(jL3);
jP4.add(jL4);
// Add the "jPanel1" on cPanel
jp.add(jP1, "1");
```

```
jp.add(jP2, "2");
jp.add(jP3, "3");
jp.add(jP4, "4");
// Creating an Object of the "JPanel" class
JPanel btnPanel = new JPanel();
// Initializing the object of the JButton class.
JButton firstButton = new JButton("First");
JButton nextButton = new JButton("->");
JButton previousButton = new JButton("<-");
JButton lastButton = new JButton("Last");
// Adding the JButton on the JPanel.
btnPanel.add(firstButton);
btnPanel.add(nextButton);
btnPanel.add(previousButton);
btnPanel.add(lastButton);
// adding Button in the ActionListener
firstButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        cObj1.first(jp);
        currCard = 1;
    }
});
lastButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        cObj1.last(jp);
        currCard = 4;
    }
});
nextButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        if (currCard < 4) {
            currCard = currCard + 1;
            cObj1.show(jp, "" + (currCard));
        } else {
```

```
                currCard = 1;
                cObjl.show(jp, "" + (currCard));
            }
        }
    });
    // add previousButton in ActionListener
    previousButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
            if (currCard > 1) {
                // decrease the value of currCard by 1
                currCard = currCard - 1;
                // show the value of currCard
                cObjl.show(jp, "" + (currCard));
            } else {
                currCard = 4;
                cObjl.show(jp, "" + (currCard));
            }
        }
    });
    // using to get the content pane
    getContentPane().add(jp, BorderLayout.NORTH);
    // using to get the content pane
    getContentPane().add(btnPanel, BorderLayout.SOUTH);
  }
  // main method
  public static void main(String argvs[]) throws IOException {
    // Creating an object of the CardLayoutExample3 class.
    cardLayout_java cll = new cardLayout_java();
    // method to set the default operation of the JFrame.
    cll.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // aethod to set the visibility of the JFrame.
    cll.setVisible(true);
  }
}
```

Flow Layout:

The class FlowLayout components in a left-to-right flow.

Class Constructor:

FlowLayout()

Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.

FlowLayout(int align)

Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.

FlowLayout(int align, int hgap, int vgap)

Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps.

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.FlowLayout;
```

```
public class flow {
    public static void main(String[] args) {
        // JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame fj = new JFrame("Demonstration of Flow Layout");
        fj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Define new buttons
        JButton jbtn1 = new JButton("Button A");
        JButton jbtn2 = new JButton("Button B");
        JButton jbtn3 = new JButton("Button C");
        JPanel pnl = new JPanel();
        pnl.setLayout(new FlowLayout());
        pnl.add(jbtn1);
        pnl.add(jbtn2);
        pnl.add(jbtn3);
        fj.add(pnl);
        fj.pack();
        fj.setVisible(true);
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
```

```
import javax.swing.*;

public class flow_layouts {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;
```

```
    public flow_layouts() {
        prepareGUI();
    }
```

```
    public static void main(String[] args) {
        flow_layouts flo = new flow_layouts();
        flo.showFlowLayoutDemo();
    }
```

```
    private void prepareGUI() {
        mainFrame = new JFrame("Java SWING Examples");
        mainFrame.setSize(400, 400);
        mainFrame.setLayout(new GridLayout(3, 1));
```

```
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
```

```
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
```

```
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
```

```
    private void showFlowLayoutDemo() {
        headerLabel.setText("Layout in action: FlowLayout");
```

```
        JPanel panel = new JPanel();
        panel.setBackground(Color.darkGray);
        panel.setSize(200, 200);
        FlowLayout layout = new FlowLayout();
        layout.setHgap(10);
        layout.setVgap(10);
```

```
        panel.setLayout(layout);
        panel.add(new JButton("OK"));
        panel.add(new JButton("Cancel"));
        controlPanel.add(panel);
```

```
        mainFrame.setVisible(true);
    }
}
```

How to change TitleBar icon in Java AWT and Swing

The setIconImage() method of Frame class is used to change the icon of Frame or Window. It changes the icon which is displayed at the left side of Frame or Window.

The Toolkit class is used to get instance of Image class in AWT and Swing. Toolkit class is the abstract super class of every implementation in the Abstract Window Toolkit(AWT). Subclasses of Toolkit are used to bind various components. It inherits Object class.

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

class change_icon {
   change_icon() {
      JFrame f = new JFrame();
      Image icon = Toolkit.getDefaultToolkit().getImage("image/login.png");
      f.setIconImage(icon);
      f.setLayout(null);
      f.setSize(200, 200);
      f.setVisible(true);
   }
   public static void main(String args[]) {
      new change_icon();
   }
}
```

GridBagLayout :

It arranges the components in a horizontal and vertical manner.

GridBagLayout()

Creates a grid bag layout manager.

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

public class GridBagLayoutDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;
   private JLabel msglabel;

   public GridBagLayoutDemo() {
      prepareGUI();
   }

   public static void main(String[] args) {
      GridBagLayoutDemo GBLD = new GridBagLayoutDemo();
      GBLD.showGridBagLayoutDemo();
   }

   private void prepareGUI() {
      mainFrame = new JFrame("Java SWING Examples");
      mainFrame.setSize(400, 400);
      mainFrame.setLayout(new GridLayout(3, 1));

      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("", JLabel.CENTER);
      statusLabel.setSize(350, 100);

      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
         }
```

```
        });
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());

        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showGridBagLayoutDemo() {
        headerLabel.setText("Layout in action: GridBagLayout");

        JPanel panel = new JPanel();
        panel.setBackground(Color.darkGray);
        panel.setSize(300, 300);
        GridBagLayout layout = new GridBagLayout();

        panel.setLayout(layout);
        GridBagConstraints gbc = new GridBagConstraints();

        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
        gbc.gridy = 0;
        panel.add(new JButton("Button 1"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 0;
        panel.add(new JButton("Button 2"), gbc);

        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipady = 20;
        gbc.gridx = 0;
        gbc.gridy = 1;
```

```
        panel.add(new JButton("Button 3"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 1;
        panel.add(new JButton("Button 4"), gbc);

        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridwidth = 2;
        panel.add(new JButton("Button 5"), gbc);

        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }
}
```

GroupLayout

GroupLayout groups its components and places them in a Container hierarchically. The grouping is done by instances of the Group class.

Group is an abstract class, and two concrete classes which implement this Group class are SequentialGroup and ParallelGroup.

SequentialGroup positions its child sequentially one after another whereas ParallelGroup aligns its child on top of each other. The GroupLayout class provides methods such as createParallelGroup() and createSequentialGroup() to create groups.

GroupLayout treats each axis independently. That is, there is a group representing the horizontal axis, and a group representing the vertical axis. Each component must exist in both a horizontal and vertical group, otherwise an IllegalStateException is thrown during layout or when the minimum, preferred, or maximum size is requested.

```
import java.awt.Container;
```

```
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import static javax.swing.GroupLayout.Alignment.*;

public class GroupExample {
   public static void main(String[] args) {
      JFrame frame = new JFrame("GroupLayoutExample");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container myPanel = frame.getContentPane();

      GroupLayout groupLayout = new GroupLayout(myPanel);
      groupLayout.setAutoCreateGaps(true);
      groupLayout.setAutoCreateContainerGaps(true);
      myPanel.setLayout(groupLayout);

      JButton b1 = new JButton("Button One");
      JButton b2 = new JButton("Button Two");
      JButton b3 = new JButton("Button Three");

      groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup()
               .addGroup(groupLayout.createParallelGroup(LEADING).addComponent(b1).addCompo
nent(b3))
            .addGroup(groupLayout.createParallelGroup(TRAILING).addComponent(b2)));

      groupLayout.setVerticalGroup(groupLayout.createSequentialGroup()
            .addGroup(groupLayout.createParallelGroup(BASELINE).addComponent(b1).addCompo
nent(b2))
            .addGroup(groupLayout.createParallelGroup(BASELINE).addComponent(b3)));

      frame.pack();
      frame.setVisible(true);
   }
}
```

Java SpringLayout

A SpringLayout arranges the children of its associated container according to a set of constraints. Constraints are nothing but horizontal and vertical distance between two-component edges. Every constraint is represented by a SpringLayout.Constraint object. Each child of a SpringLayout container, as well as the container itself, has exactly one set of constraints associated with them.

Each edge position is dependent on the position of the other edge. If a constraint is added to create a new edge, than the previous binding is discarded. SpringLayout doesn't automatically set the location of the components it manages.

Constructor

SpringLayout(): The default constructor of the class is used to instantiate the SpringLayout class.

```
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

public class MySpringDemo {
   private static void createAndShowGUI() {
      JFrame frame = new JFrame("MySpringDemp");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
      Container contentPane = frame.getContentPane();
      SpringLayout layout = new SpringLayout();
      contentPane.setLayout(layout);
```

```
      JLabel label = new JLabel("Label: ");
      JTextField textField = new JTextField("My Text Field", 15);
      contentPane.add(label);
      contentPane.add(textField);
```

```
        layout.putConstraint(SpringLayout.WEST, label, 6, SpringLayout.WEST, contentPane);
        layout.putConstraint(SpringLayout.NORTH, label, 6, SpringLayout.NORTH, contentPane);
        layout.putConstraint(SpringLayout.WEST, textField, 6, SpringLayout.EAST, label);
        layout.putConstraint(SpringLayout.NORTH, textField, 6, SpringLayout.NORTH, contentPane);
        layout.putConstraint(SpringLayout.EAST, contentPane, 6, SpringLayout.EAST, textField);
        layout.putConstraint(SpringLayout.SOUTH, contentPane, 6, SpringLayout.SOUTH, textField);
```

```
        frame.pack();
        frame.setVisible(true);
    }
```

```
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

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

public class SwingSpringLayout {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
```

```
    public SwingSpringLayout() {
        prepareGUI();
```

```
}

public static void main(String[] args) {
   SwingSpringLayout swingLayoutDemo = new SwingSpringLayout();
   swingLayoutDemo.showSpringLayoutDemo();
}

private void prepareGUI() {
   mainFrame = new JFrame("Java SWING Examples");
   mainFrame.setSize(400, 400);
   mainFrame.setLayout(new GridLayout(3, 1));

   headerLabel = new JLabel("", JLabel.CENTER);
   statusLabel = new JLabel("", JLabel.CENTER);
   statusLabel.setSize(350, 100);

   mainFrame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent windowEvent) {
         System.exit(0);
      }
   });
   controlPanel = new JPanel();
   controlPanel.setLayout(new FlowLayout());

   mainFrame.add(headerLabel);
   mainFrame.add(controlPanel);
   mainFrame.add(statusLabel);
   mainFrame.setVisible(true);
}

private void showSpringLayoutDemo() {
   headerLabel.setText("Layout in action: SpringLayout");
   SpringLayout layout = new SpringLayout();
```

```
        JPanel panel = new JPanel();
        panel.setLayout(layout);
        JLabel label = new JLabel("Enter Name: ");
        JTextField textField = new JTextField("", 15);
        panel.add(label);
        panel.add(textField);
```

```
        layout.putConstraint(SpringLayout.WEST, label, 5, SpringLayout.WEST, controlPanel);
        layout.putConstraint(SpringLayout.NORTH, label, 5, SpringLayout.NORTH, controlPanel);
        layout.putConstraint(SpringLayout.WEST, textField, 5, SpringLayout.EAST, label);
        layout.putConstraint(SpringLayout.NORTH, textField, 5, SpringLayout.NORTH,
                controlPanel);
```

```
        layout.putConstraint(SpringLayout.EAST, panel, 5, SpringLayout.EAST, textField);
        layout.putConstraint(SpringLayout.SOUTH, panel, 5, SpringLayout.SOUTH, textField);
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }
}
```

Examples:

Login demo code

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
```

```
import javax.swing.JTextField;
public class LoginDemo extends JFrame implements ActionListener {
   JPanel panel;
   JLabel user_label, password_label, message;
   JTextField userName_text;
   JPasswordField password_text;
   JButton submit, cancel;
   LoginDemo() {
      // User Label
       user_label = new JLabel();
       user_label.setText("User Name :");
       userName_text = new JTextField();
       // Password
       password_label = new JLabel();
       password_label.setText("Password :");
       password_text = new JPasswordField();
       // Submit
      submit = new JButton("SUBMIT");
       panel = new JPanel(new GridLayout(3, 1));
       panel.add(user_label);
       panel.add(userName_text);
       panel.add(password_label);
       panel.add(password_text);
```

```
       message = new JLabel();
       panel.add(message);
       panel.add(submit);
```

```
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
       // Adding the listeners to components..
       submit.addActionListener(this);
       add(panel, BorderLayout.CENTER);
       setTitle("Please Login Here !");
```

```
        setSize(300, 100);
        setVisible(true);

    }
    public static void main(String[] args) {
        new LoginDemo();
    }
    @Override
    public void actionPerformed(ActionEvent ae) {
        String userName = userName_text.getText();
        String password = password_text.getText();
        if (userName.trim().equals("admin") && password.trim().equals("admin")) {
            message.setText(" Hello " + userName
                    + "");
        } else {
            message.setText(" Invalid user.. ");
        }
    }
}
```

Icon change

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

class p12_change_icon {
    p12_change_icon() {
        JFrame f = new JFrame();
        Image icon = Toolkit.getDefaultToolkit().getImage("image/login.png");
        f.setIconImage(icon);
        f.setLayout(null);
        f.setSize(200, 200);
        f.setVisible(true);
    }
```

```
    public static void main(String args[]) {
        new p12_change_icon();
    }
}
```

Layouts in one code

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import javax.swing.GroupLayout;

import java.awt.*;
public class all_layouts_demo {
    public static void main(String[] args) {
        // Create and set up a frame window
        // JFrame.setDefaultLookAndFeelDecorated(true);
        // Flow Layouts
        JFrame f = new JFrame("Layout");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Define new buttons
        JButton jb1 = new JButton("Button 1");
        JButton jb2 = new JButton("Button 2");
        JButton jb3 = new JButton("Button 3");
        // Define the panel to hold the buttons
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout());
        p.add(jb1);
        p.add(jb2);
        p.add(jb3);
        // Set the window to be visible as the default to be false
        f.add(p);
```

```
f.setSize(300, 300);
f.pack();
f.setVisible(true);
// Border Lyouts
JFrame f1 = new JFrame("frame with direction");
// Define new buttons with different regions
JButton jbn = new JButton("NORTH");
JButton jbs = new JButton("SOUTH");
JButton jbw = new JButton("WEST");
JButton jbe = new JButton("EAST");
JButton jbc = new JButton("CENTER");
// Define the panel to hold the buttons
JPanel p1 = new JPanel();
p1.setLayout(new BorderLayout());
p1.add(jbn, BorderLayout.NORTH);
p1.add(jbs, BorderLayout.SOUTH);
p1.add(jbw, BorderLayout.WEST);
p1.add(jbe, BorderLayout.EAST);
p1.add(jbc, BorderLayout.CENTER);
f1.add(p1);
f1.pack();
f1.setSize(300, 300);
f1.setVisible(true);
// Grid layouts
JFrame f2 = new JFrame("frame with grid");
JPanel p2 = new JPanel();
p2.setLayout(new GridLayout(3, 2));
JButton jb1g = new JButton("Button 1");
JButton jb2g = new JButton("Button 2");
JButton jb3g = new JButton("Button 3");
JButton jb4g = new JButton("Button 4");
JButton jb5g = new JButton("Button 5");
p2.add(jb1g);
p2.add(jb2g);
```

```
p2.add(jb3g);
p2.add(jb4g);
p2.add(jb5g);
f2.add(p2);
f2.pack();
f2.setVisible(true);
f2.setSize(300, 300);
// Define the panel to hold the components
// Put constraints on different buttons with gridbag layouts
JFrame fg = new JFrame("gridbag constrain");
JPanel pg = new JPanel();
GridBagLayout layout = new GridBagLayout();
pg.setLayout(layout);
GridBagConstraints gbc = new GridBagConstraints();
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.gridx = 0;
gbc.gridy = 0;
pg.add(new JButton("Button1"), gbc);
gbc.gridx = 1;
gbc.gridy = 0;
pg.add(new JButton("Button 2"), gbc);
gbc.gridx = 0;
gbc.gridy = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.gridwidth = 2;
pg.add(new JButton("Button 3"), gbc);
fg.add(pg);
fg.pack();
fg.setVisible(true);
fg.setSize(300, 300);
// spring layouts
JFrame sf = new JFrame("spring Layout");
sf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Define the panel to hold the components
```

```
JPanel sp = new JPanel();
SpringLayout slayout = new SpringLayout();
JLabel label = new JLabel("Label: ");
JTextField text = new JTextField("Text field", 15);
sp.setSize(300, 300);
sp.setLayout(slayout);
sp.add(label);
sp.add(text);
// Put constraint on components
slayout.putConstraint(SpringLayout.WEST, label, 5, SpringLayout.WEST, sp);
slayout.putConstraint(SpringLayout.NORTH, label, 5, SpringLayout.NORTH, sp);
slayout.putConstraint(SpringLayout.WEST, text, 5, SpringLayout.EAST, label);
slayout.putConstraint(SpringLayout.NORTH, text, 5, SpringLayout.NORTH, sp);
// Set the window to be visible as the default to be false
sf.add(sp);
sf.pack();
sf.setVisible(true);
sf.setSize(300, 300);
JFrame fgl = new JFrame("Group Layout");
fgl.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Define new buttons with different width on help of the ---
JButton jbgl1 = new JButton("1");
JButton jbgl2 = new JButton("2");
JButton jbgl3 = new JButton("3");
JButton jbgl4 = new JButton("4");
// Define the panel to hold the buttons
JPanel pgl = new JPanel();
pgl.setSize(300, 300);
GroupLayout gl = new GroupLayout(pgl);
gl.setAutoCreateGaps(true);
gl.setAutoCreateContainerGaps(true);
pgl.setLayout(gl);
// Set for horizontal and vertical group
gl.setHorizontalGroup(gl.createSequentialGroup().addComponent(jbgl1).addComponent(jbgl2)
```

```
            .addGroup(gl.createSequentialGroup().addGroup(
                    gl.createParallelGroup(GroupLayout.Alignment.LEADING).addComponent(jbgl3)
                            .addComponent(jbgl4))));
        gl.setVerticalGroup(
                gl.createSequentialGroup().addComponent(jbgl1).addComponent(jbgl2).addComponent(j
bgl3)
                        .addComponent(jbgl4));
        // Set the window to be visible as the default to be false
        fgl.add(pgl);
        fgl.pack();
        fgl.setVisible(true);
        fgl.setSize(300, 300);
    }
}
```