UNIT I - JAVA FUNDAMENTALS

- Java Data types
- Class – Object
- I / O Streams
- File Handling concepts
- Threads
- Applets
- Swing Framework
- Reflection

Presented by,
B.Vijayalakshmi
Computer Centre
MIT Campus
Anna University

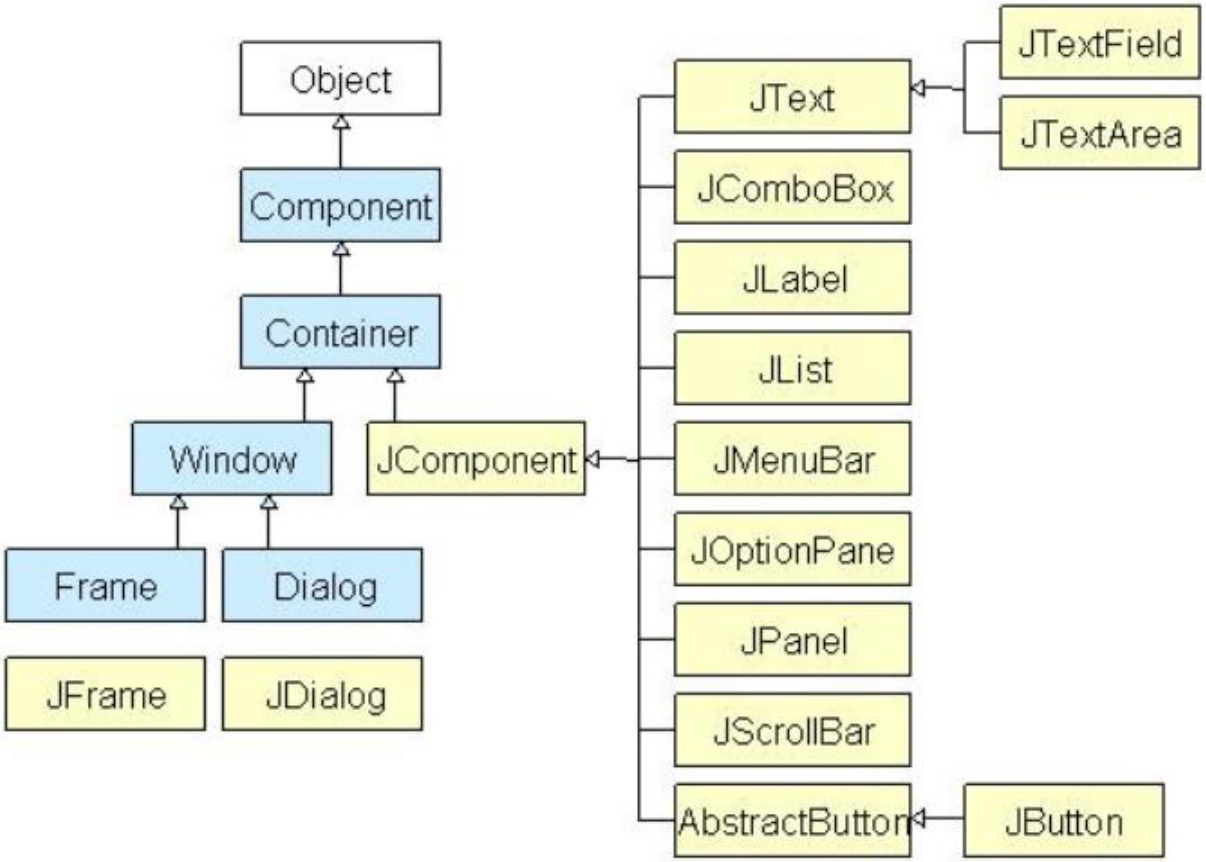
Swing

- Swing is a **part of Java Foundation classes (JFC)**
- It is a preferred API for window based applications because of its **platform independent and light-weight nature.**
- Swing is **built upon AWT API** and **entirely written in java**
- It is platform independent unlike AWT & provides a look and feel unrelated to the underlying OS.
- It has more powerful and flexible and lightweight components than AWT.
- In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Difference between AWT and Swing

Java AWT	Java Swing
AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
AWT components are **heavyweight**.	Swing components are **lightweight**.
AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

AWT and Swing Hierarchy



- All the components in swing like JButton, JComboBox, JList, JLabel are inherited from the JComponent class which can be added to the container classes.
- Containers are the windows like frame and dialog boxes.
- Basic swing components are the building blocks of any GUI application.
- Methods like setLayout override the default layout in each container.
- Containers like JFrame and JDialog can only add a component to itself.

Swing Component

- It is an **independent visual control** and Java Swing Framework contains a large set of these components which provide rich functionalities and allow high level of customization.
- They all are **derived from JComponent class**.
- All these components are **lightweight components.**
- This class provides some common functionality like pluggable look and feel, support for accessibility, drag and drop, layout, etc.

Swing Containers

- **It holds a group of components**.
- It provides a space where a component can be managed and displayed.
- Containers are of two types:
  - **Top level Containers**
    - It **inherits Component and Container of AWT**.
    - It **cannot be contained within other containers**.
    - Heavyweight.
    - Example: JFrame, JDialog, JApplet
  - **Lightweight Containers**
    - It **inherits JComponent class**.
    - It is a general purpose container.
    - It can be used to organize related components together.
    - Example: JPanel

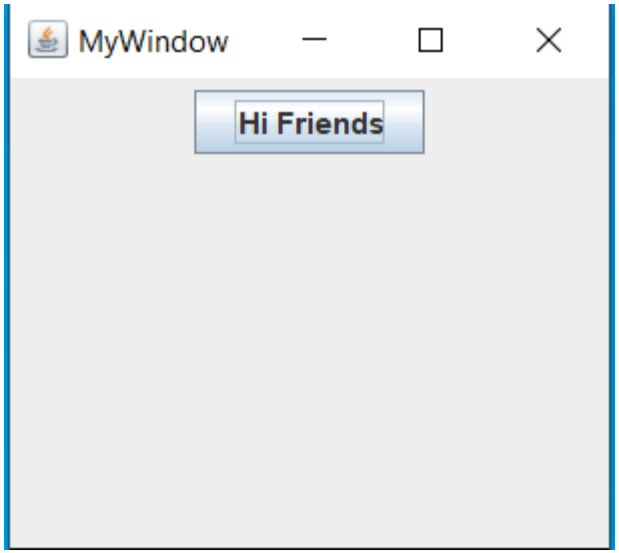
Swing Program by creating a JFrame

- Swing classes are defined in javax.swing package and its sub-packages.

- There are two ways to create a JFrame Window.
  - By instantiating JFrame class.
  - By extending JFrame class.

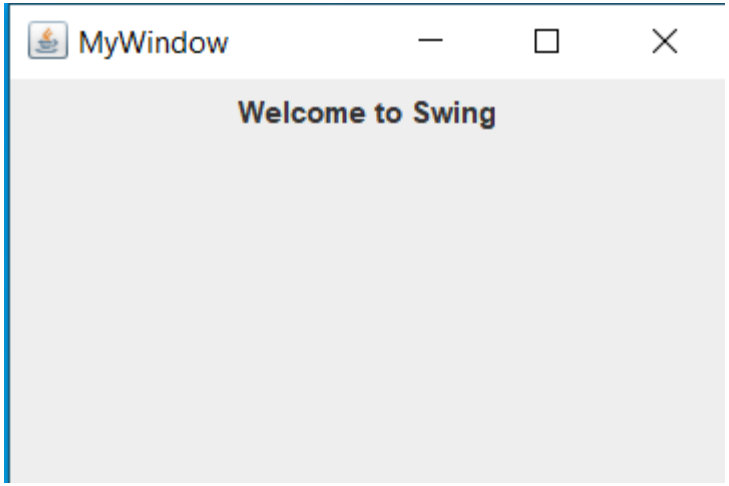
Creating a Jframe By instantiating JFrame class Example

```
import javax.swing.*;  //importing swing package
import java.awt.*;     //importing awt package
public class JFrameEx1
{
  JFrame jf;
  public JFrameEx1()
  {
        jf = new JFrame("MyWindow");          //Creating a Jframe
        JButton btn = new Jbutton(“Hi Friends”);//Creating a Button
        jf.add(btn);                          //adding button to frame
        jf.setLayout(new FlowLayout());       //setting layout using FlowLayout object
        jf.setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE); //setting close  operation.
        jf.setSize(200, 300);  //setting size
        jf.setVisible(true); //setting frame visibility
  }
  public static void main(String[] args)
  {
                new JFrameEx1();
  }
}
```



Creating a Jframe By extending JFrame class Example

```
import javax.swing.*; //importing swing package
import java.awt.*; //importing awt package
public class JFrameEx2 extends JFrame
{
   public JFrameEx2()
  {
     setTitle("MyWindow"); //setting title of frame as  MyWindow
     JLabel lb = new JLabel("Welcome to Swing");//Creating a label
     add(lb);                    //adding label to frame.
     setLayout(new FlowLayout());    //setting layout using FlowLayout object.
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //setting close operation.
     setSize(300, 200);            //setting size
     setVisible(true);            //setting frame visibility
   }
    public static void main(String[] args)
    {
                          new JFrameEx2();
    }
}
```



Swing Components

JButton

- This class is used to create a labeled button that has platform independent implementation.
- The application result in some action when the button is pushed.
- It inherits AbstractButton class.
- **Constructors:**
  - JButton()It creates a button with no text and icon.
  - JButton(String s)It creates a button with the specified text.
  - JButton(Icon i)It creates a button with the specified icon object.
- **Methods:**
  - void setText(String s) →It is used to set specified text on button
  - String getText()→It is used to return the text of the button
  - void setEnabled(boolean b)→It is used to enable or disable the button
  - void setIcon(Icon b)→It is used to set the specified Icon on the button
  - Icon getIcon()→It is used to get the Icon of the button
  - void setMnemonic(int a)→It is used to set the mnemonic on the button
  - void addActionListener(ActionListener a)→It is used to add the action listener to this object.

JTextField

- It is used for taking input of single line of text.
- It is most widely used text component.
- It inherits JTextComponent class.
- **Constructors:**
  - JTextField(int cols)
  - JTextField(String str, int cols)
  - JTextField(String str)
  - *cols* represent the number of columns in text field.
- **Methods:**
  - void addActionListener(ActionListener l)→It is used to add the specified action listener to receive action events from this textfield.
  - Action getAction()→It returns the currently set Action for this ActionEvent source, or null if no Action is set
  - void setFont(Font f)→It is used to set the current font
  - void removeActionListener(ActionListener l)→It is used to remove the specified action listener so that it no longer receives action events from this textfield.

JCheckBox

- This class is used to create checkbox in swing framework.
- It is used to turn an option on (true) or off (false).
- It inherits JToggleButton class.
- **Constructors:**
  - JCheckBox(String s)→Creates an initially unselected check box with text
  - JCheckBox(String text, boolean selected)→Creates a check box with text and specifies whether or not it is initially selected.
- **Methods:**
  - AccessibleContext getAccessibleContext()→It is used to get the AccessibleContext associated with this JCheckBox.
  - protected String paramString()→It returns a string representation of this JCheckBox.

JRadioButton

- This class is used to create a radio button.
- It is used to choose one option from multiple options.
- It should be added in ButtonGroup to select one radio button only.
- **Constructors:**
  - JRadioButton()→ Creates an unselected radio button with no text.
  - JRadioButton(String s)→Creates an unselected radio button with specified text.
  - JRadioButton(String s, boolean selected)→Creates a radio button with the specified text and selected status.
- **Methods:**
  - void setText(String s)→It is used to set specified text on button.
  - String getText()→It is used to return the text of the button.
  - void setEnabled(boolean b)→It is used to enable or disable the button.
  - void setIcon(Icon b)→It is used to set the specified Icon on the button.
  - Icon getIcon()→It is used to get the Icon of the button.
  - void setMnemonic(int a)→It is used to set the mnemonic on the button.
  - void addActionListener(ActionListener a)→It is used to add the action listener to this object.

JComboBox

- Combo box is a combination of text fields and drop-down list.
- Choice selected by user is shown on the top of a menu.
- It inherits JComponent class.
- **Constructors:**
  - JComboBox()→Creates a JComboBox with a default data model
  - JComboBox(Object[] items)→Creates a JComboBox that contains the elements in the specified array.
  - JComboBox(Vector<?> items)→Creates a JComboBox that contains the elements in the specified Vector.
- **Methods:**
  - void addItem(Object anObject)→It is used to add an item to the item list.
  - void removeItem(Object anObject)→It is used to delete an item to the item list.
  - void removeAllItems()→It is used to remove all the items from the list.
  - void setEditable(boolean b)→It is used to determine whether the JComboBox is editable
  - void addActionListener(ActionListener a)→It is used to add the ActionListener
  - void addItemListener(ItemListener i)→It is used to add the ItemListener.

JLabel

- It is used to display a single line of read only text.
- The text can be changed by an application but a user cannot edit it directly.
- It inherits JComponent class.
- **Constructors:**
  - JLabel()→Creates a JLabel instance with no image and with an empty string for the title.
  - JLabel(String s)→Creates a JLabel instance with the specified text.
  - JLabel(Icon i)→Creates a JLabel instance with the specified image.
  - JLabel(String s, Icon i, int horizontalAlignment)→Creates a JLabel instance with the specified text, image, and horizontal alignment.
- **Methods:**
  - String getText()→ It returns the text string that a label displays.
  - void setText(String text)→It defines the single line of text this component will display.
  - void setHorizontalAlignment(int alignment)→It sets the alignment of the label's contents along the X axis.
  - Icon getIcon()→It returns the graphic image that the label displays.
  - int getHorizontalAlignment()→It returns the alignment of the label's contents along the X axis.

JPasswordField

- It is specifically used for password and it can be edited.
- It inherits JTextField class.
- **Constructors:**
- JPasswordField()→Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.
- JPasswordField(int columns)→Constructs a new empty JPasswordField with the specified number of columns.
- JPasswordField(String text)→Constructs a new JPasswordField initialized with the specified text.
- JPasswordField(String text, int columns)→Construct a new JPasswordField initialized with the specified text and columns.

JTextArea

- It is used for displaying multiple-line text.
- It allows the editing of multiple line text.
- It inherits JTextComponent class
- **Constructors:**
  - JTextArea()→Creates a text area that displays no text initially.
  - JTextArea(String s)→Creates a text area that displays specified text initially.
  - JTextArea(int row, int column)→Creates a text area with the specified number of rows and columns that displays no text initially.
  - JTextArea(String s, int row, int column)→Creates a text area with the specified number of rows and columns that displays specified text.
- **Methods:**
  - void setRows(int rows)→It is used to set specified number of rows.
  - void setColumns(int cols)→It is used to set specified number of columns.
  - void setFont(Font f)→It is used to set the specified font.
  - void insert(String s, int position)→It is used to insert the specified text on the specified position.
  - void append(String s)→It is used to append the given text to the end of the document.

JTable

- It used to draw a table to display data.
- It is composed of rows and columns.
- **Constructors:**
- JTable()→Creates a table with empty cells.
- JTable(Object[][] rows, Object[] columns)→Creates a table with the specified data.

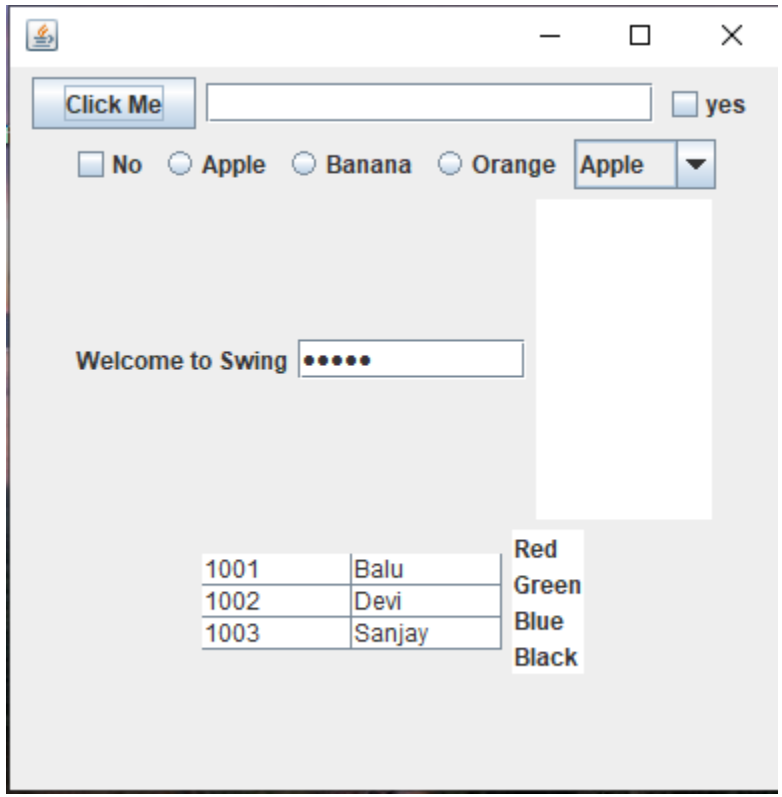
JList

- It is used to represent a list of items together.
- One or more than one item can be selected from the list.
- It inherits JComponent class.
- **Constructors:**
  - JList()→Creates a JList with an empty, read-only, model.
  - JList(ary[] listData)→Creates a JList that displays the elements in the specified array.
  - JList(ListModel<ary> dataModel)→Creates a JList that displays elements from the specified, non-null, model.
- **Methods:**
  - Void addListSelectionListener(ListSelectionListener listener)→It is used to add a listener to the list, to be notified each time a change to the selection occurs.
  - int getSelectedIndex()→It is used to return the smallest selected cell index.
  - ListModel getModel()→It is used to return the data model that holds a list of items displayed by the JList component.
  - void setListData(Object[] listData)→It is used to create a read-only ListModel from an array of objects.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class JFrameEx3 extends JFrame
{
 JFrameEx3()
 {
        JButton bt1 = new JButton("Click Me");   //creating JButton
        add(bt1);
        JTextField jtf = new JTextField(20);  //creating JTextField.
        add(jtf);     //adding JTextField to frame.
        JCheckBox jcb1 = new JCheckBox("yes");   //creating JCheckBox.
        add(jcb1);         //adding JCheckBox to frame.
        JCheckBox jcb2 = new JCheckBox("No");
        add(jcb2);
        JRadioButton jrb1 = new JRadioButton("Apple"); //creating JRadioButton.
        JRadioButton jrb2 = new JRadioButton("Banana");
        JRadioButton jrb3 = new JRadioButton("Orange");
        ButtonGroup bg=new ButtonGroup();
        bg.add(jrb1); bg.add(jrb2);bg.add(jrb3);  //adding JRadioButton to ButtonGroup.
        add(jrb1);add(jrb2);add(jrb3); //adding JRadioButton to frame.
```

```
String frname[]={"Apple","Banana","Orange","Guava","Grapes"};
//JComboBox jc = new JComboBox(frname);//initialzing combo box with list of name.
JComboBox<String> jc=new JComboBox<>(frname);
add(jc);    //adding JComboBox to frame.
JLabel lbl=new JLabel("Welcome to Swing"); //creating Jlabel
add(lbl); //adding JLabel
JPasswordField psw = new JPasswordField("12345",10);//creating JPasswordField()
add(psw);//adding JPasswordField()
JTextArea ta=new JTextArea(10,8); //creating JTextArea
add(ta); //adding JTextArea
String studata[][]={ {"1001","Balu"}, {"1002","Devi"}, {"1003","Sanjay"}};
            String column[]={"SID","SNAME"};
            JTable tbl=new JTable(studata,column);
add(tbl);
DefaultListModel<String> listM = new DefaultListModel<>();
 listM.addElement("Red");
 listM.addElement("Green");
 listM.addElement("Blue");
 listM.addElement("Black");
 JList<String> list = new JList<>(listM);
```

```
            add(list);
            setSize(400, 400);   //setting size of Jframe
            setLayout(new FlowLayout());  //setting layout using FlowLayout object
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     //setting close operation.
            setVisible(true);
}
public static void main(String[] args)
  {
    new JFrameEx3();
  }
}
```



Java LayoutManagers

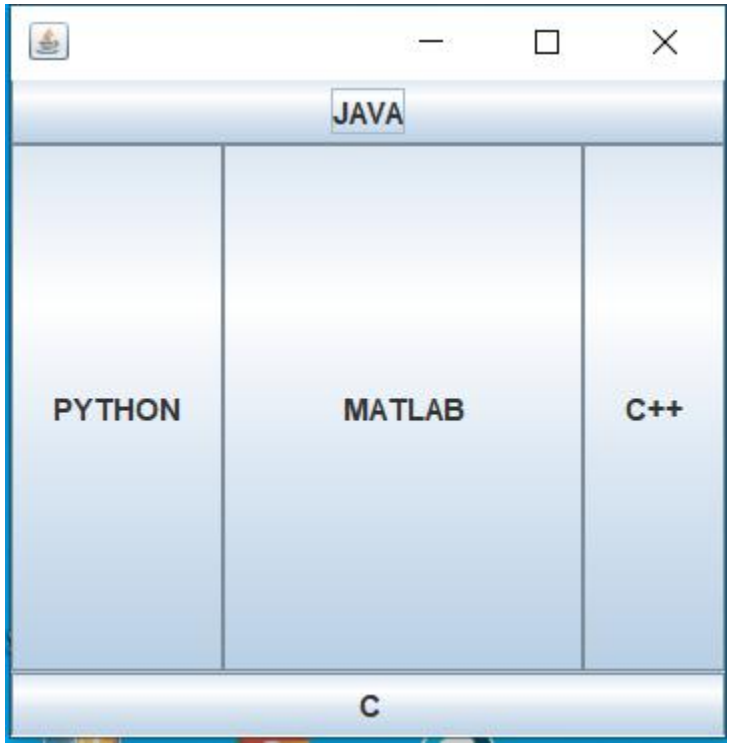
- The LayoutManagers are **used to arrange components in a particular manner.**
- **LayoutManager is an interface** that is implemented by all the classes of layout managers.
- There are following classes that represents the layout managers:
  - java.awt.BorderLayout
  - java.awt.FlowLayout
  - java.awt.GridLayout
  - java.awt.CardLayout
  - java.awt.GridBagLayout
  - javax.swing.BoxLayout
  - javax.swing.GroupLayout
  - javax.swing.ScrollPaneLayout
  - javax.swing.SpringLayout etc.

Java BorderLayout

- This Layout is used to arrange the components in five regions: north, south, east, west and center.
- Each region (area) may contain one component only.
- It is the **default layout of frame or window**.
- The BorderLayout provides five constants for each region:
  - **public static final int NORTH**
  - **public static final int SOUTH**
  - **public static final int EAST**
  - **public static final int WEST**
  - **public static final int CENTER**
- Constructors of BorderLayout class:
  - **BorderLayout():** creates a border layout but with no gaps between the components.
  - **JBorderLayout(int hgap, int vgap):** creates a border layout with the given horizontal and vertical gaps between the components.

BorderLayout Example

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutEx extends JFrame
{
        BorderLayoutEx()
        {
                JButton b1=new JButton("JAVA");
                JButton b2=new JButton("C");
                 JButton b3=new JButton("C++");;
                 JButton b4=new JButton("PYTHON");
                 JButton b5=new JButton("MATLAB");

                add(b1,BorderLayout.NORTH);
                add(b2,BorderLayout.SOUTH);
                add(b3,BorderLayout.EAST);
                add(b4,BorderLayout.WEST);
                add(b5,BorderLayout.CENTER);
                setSize(300,300);
                setVisible(true);
        }
        public static void main(String[] args)
        {
           new BorderLayoutEx();
        }
}
```

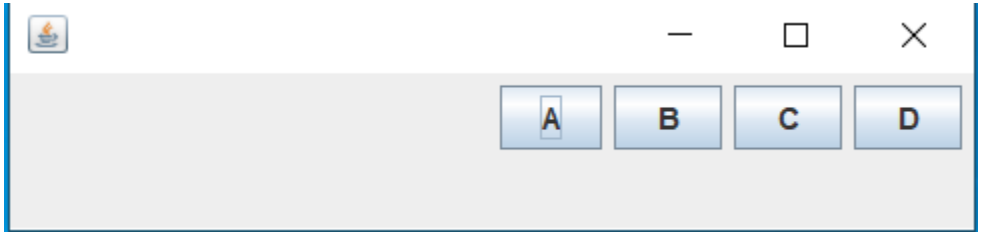


Java FlowLayout

- This Layout is used to arrange the components in a line, one after another (in a flow).
- It is the default layout of applet or panel.
- **Constructors:**
  - **FlowLayout():** creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
  - **FlowLayout(int align):** creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
  - **FlowLayout(int align, int hgap, int vgap):** creates a flow layout with the given alignment and the given horizontal and vertical gap.
- **Fields of FlowLayout class**
  - **public static final int LEFT**
  - **public static final int RIGHT**
  - **public static final int CENTER**
  - **public static final int LEADING**
  - **public static final int TRAILING**

FlowLayout Example

```
import java.awt.*;
import javax.swing.*;
public class FlowLayoutEx extends JFrame
{
   FlowLayoutEx()
   {
      JButton b1=new JButton("A");
            JButton b2=new JButton("B");
            JButton b3=new JButton("C");
            JButton b4=new JButton("D");
            add(b1);
            add(b2);
            add(b3);
            add(b4);
           //setting flow layout of right alignment
            setLayout(new FlowLayout(FlowLayout.RIGHT));
            setSize(400,100);
            setVisible(true);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args)
 {
   new FlowLayoutEx();
 }
}
```

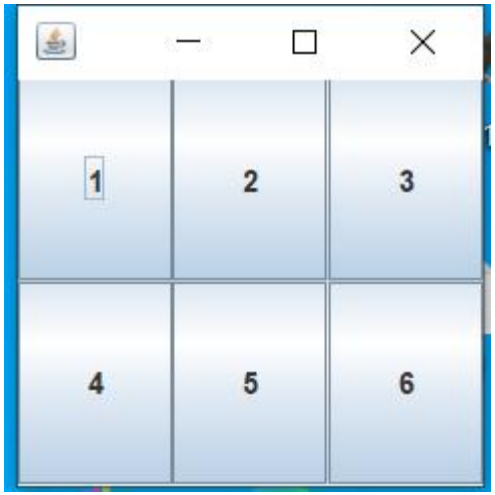


Java GridLayout

- This layout is used to arrange the components in rectangular grid.
- One component is displayed in each rectangle.
- Constructors of GridLayout class
  - **GridLayout():** creates a grid layout with one column per component in a row.
  - **GridLayout(int rows, int columns):** creates a grid layout with the given rows and columns but no gaps between the components.
  - **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns alongwith given horizontal and vertical gaps.

GridLayout Example

```
import java.awt.*;
import javax.swing.*;
public class GridLayoutEx extends JFrame
{
  GridLayoutEx()
  {
   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");
   add(b1);   add(b2); add(b3);
   add(b4);   add(b5);  add(b6);
   setLayout(new GridLayout(2,3)); //setting grid layout of 2 rows and 3 columns
   setSize(200,200);
   setVisible(true);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args)
  {
   new GridLayoutEx();
  }
}
```

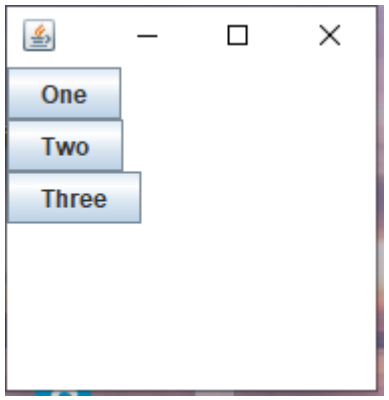


Java BoxLayout

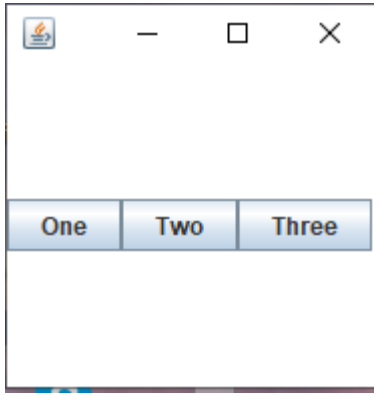
- It is found in javax.swing package.
- This layout is used to arrange the components either vertically or horizontally.
- For this purpose, BoxLayout provides four constants.
  - **public static final int X_AXIS**
  - **public static final int Y_AXIS**
  - **public static final int LINE_AXIS**
  - **public static final int PAGE_AXIS**
- **Constructor:**
  - **BoxLayout(Container c, int axis):** creates a box layout that arranges the components with the given axis.

BoxLayout Example

```
import java.awt.*;
import javax.swing.*;
public class BoxLayoutEx  extends Frame
{
 JButton b1,b2,b3;
 public BoxLayoutEx ()
 {
   b1=new JButton("One");
   b2=new JButton("Two");
   b3=new JButton("Three");
   add (b1);
   add (b2);
   add (b3);
   setLayout (new BoxLayout (this,BoxLayout.Y_AXIS));
   setSize(200,200);
   setVisible(true);
 }
 public static void main(String args[])
 {
    new BoxLayoutEx();
 }
}
```



Y_AXIS



X_AXIS

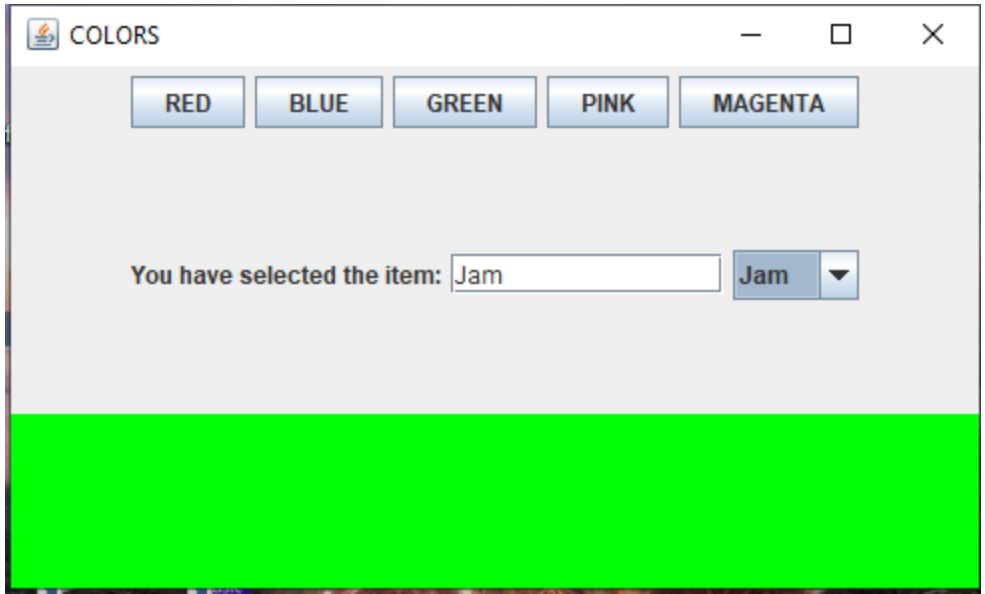
Swing Example

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class ColorSet extends JFrame implements ActionListener,ItemListener
{
  JPanel panel1,panel2;
  JButton b1,b2,b3,b4,b5;
  JTextField tf1;
  JLabel lb1;
  JComboBox<String> cb;

 ColorSet()
 {
   setTitle("COLORS");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   panel1 = new JPanel();
   panel1.setSize(100, 50);

   b1 = new JButton("RED");
   b1.addActionListener(this);
   b2 = new JButton("BLUE");
   b2.addActionListener(this);
```

```
b3 = new JButton("GREEN");
 b3.addActionListener(this);
 b4 = new JButton("PINK");
 b4.addActionListener(this);
 b5 = new JButton("MAGENTA");
 b5.addActionListener(this);
  //Adding buttons to the Panel
 panel1.add(b1);
 panel1.add(b2);
 panel1.add(b3);
 panel1.add(b4);
 panel1.add(b5);

String product[]={"Bread","Butter","Jam","Milk","Sugar"};
cb=new JComboBox<>(product);
lb1=new JLabel("You have selected the item:");
tf1=new JTextField(12);
cb.addItemListener(this);
 panel2 = new JPanel();
panel2.setSize(100, 50);
panel2.add(lb1);
panel2.add(tf1);
panel2.add(cb);
```

```
  getContentPane().add(panel1);
  getContentPane().add(panel2,"South");
  setSize(500,300);
  setVisible(true);
 // setLayout(new FlowLayout());
 setLayout(new GridLayout(3,1));
}
 public void actionPerformed(ActionEvent e)
 {
      Object obj = e.getSource();
       if(obj ==(b1))
      {
        getContentPane().setBackground(java.awt.Color.RED);
       }
       if(obj == b2)
      {
          getContentPane().setBackground(Color.blue);
       }
       if(obj == b3)
      {
        getContentPane().setBackground(Color.green);
       }
```

```
 if(obj == b4)
       {
          getContentPane().setBackground(Color.pink);
       }
       if(obj == b5)
      {
        getContentPane().setBackground(java.awt.Color.magenta);
       }


   }
 public void itemStateChanged(ItemEvent e)
  {
     String txt = cb.getItemAt(cb.getSelectedIndex());
    tf1.setText(txt);
    }
}
class SwingExample
 {
   public static void main(String[] args)
  {
      ColorSet o = new ColorSet();
   }
}
```



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.

message dialog box

input dialog box

JScrollBar

- The object of JScrollbar class is used to add horizontal and vertical scrollbar.
- It is an implementation of a scrollbar.
- It inherits JComponent class.

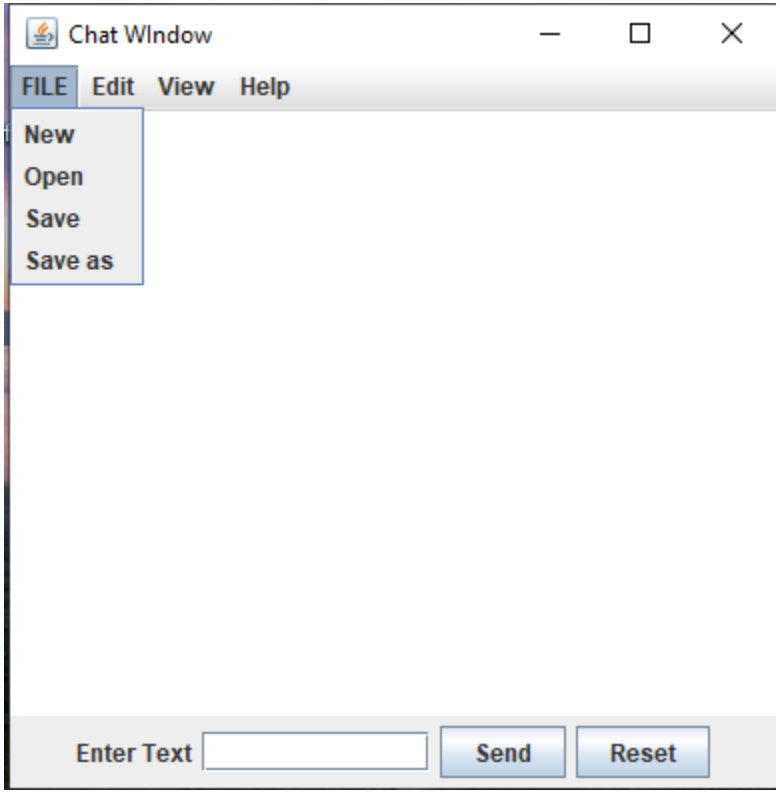
JMenuBar, JMenu and JMenuItem

- In Java, Swing toolkit contains a JMenuBar, JMenu and JMenuItem class.
- It is under package javax.swing.JMenuBar, javax.swing.JMenu and javax.swing.JMenuItem class.
- The JMenuBar class is used for displaying menubar on the frame.
- The JMenu Object is used for pulling down the components of the menu bar.
- The JMenuItem Object is used for adding the labelled menu item.

JSeparator

- In Java, Swing toolkit contains a JSeparator Class.
- It is under package javax.swing.JSeparator class.
- It is used for creating a separator line between two components.

MenuBarExample

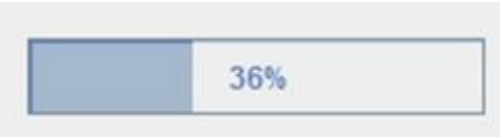


```
import javax.swing.*;
import java.awt.*;
class MenuExample extends JFrame
 {
   MenuExample()
   {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(400, 400);
      setTitle("Chat WIndow");
      //Creating the MenuBar and adding components
      JMenuBar mb = new JMenuBar();
      JMenu m1 = new JMenu("FILE");
      JMenu m2 = new JMenu("Edit");
      JMenu m3 = new JMenu("View");
      JMenu m4 = new JMenu("Help");
      mb.add(m1);
      mb.add(m2);
      mb.add(m3);
      mb.add(m4);
      JMenuItem m11 = new JMenuItem("New");
      JMenuItem m12 = new JMenuItem("Open");
      JMenuItem m13 = new JMenuItem("Save");
      JMenuItem m14 = new JMenuItem("Save as");
      m1.add(m11);
```

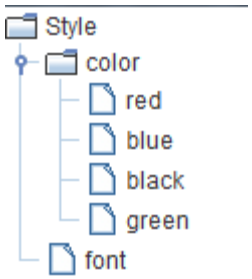
```
  m1.add(m12);
   m1.add(m13);
   m1.add(m14);
  //Creating the panel at bottom and adding components
   JPanel panel = new JPanel(); // the panel is not visible in output
   JLabel label = new JLabel("Enter Text");
   JTextField tf = new JTextField(10); // accepts upto 10 characters
   JButton send = new JButton("Send");
   JButton reset = new JButton("Reset");
   panel.add(label); // Components Added using Flow Layout
   panel.add(tf);
   panel.add(send);
   panel.add(reset);

   // Text Area at the Center
   JTextArea ta = new JTextArea();

   //Adding Components to the frame.
   getContentPane().add(BorderLayout.SOUTH, panel);
   getContentPane().add(BorderLayout.NORTH, mb);
   getContentPane().add(BorderLayout.CENTER, ta);
   setVisible(true);
}
```

```
 public static void main(String args[])
   {

      new MenuExample();
   }
}
```

JProgressBar



- In Java, Swing toolkit contains a JProgressBar Class.
- It is under package javax.swing.JProgressBarclass.
- It is used for creating a progress bar of a task.



JTree

- In Java, Swing toolkit contains a JTree Class.
- It is under package javax.swing.JTreeclass.
- It is used for creating a tree-structured of data. It is a very complex component.

A Visual Guide to Swing Components (Windows Look and Feel)

https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html