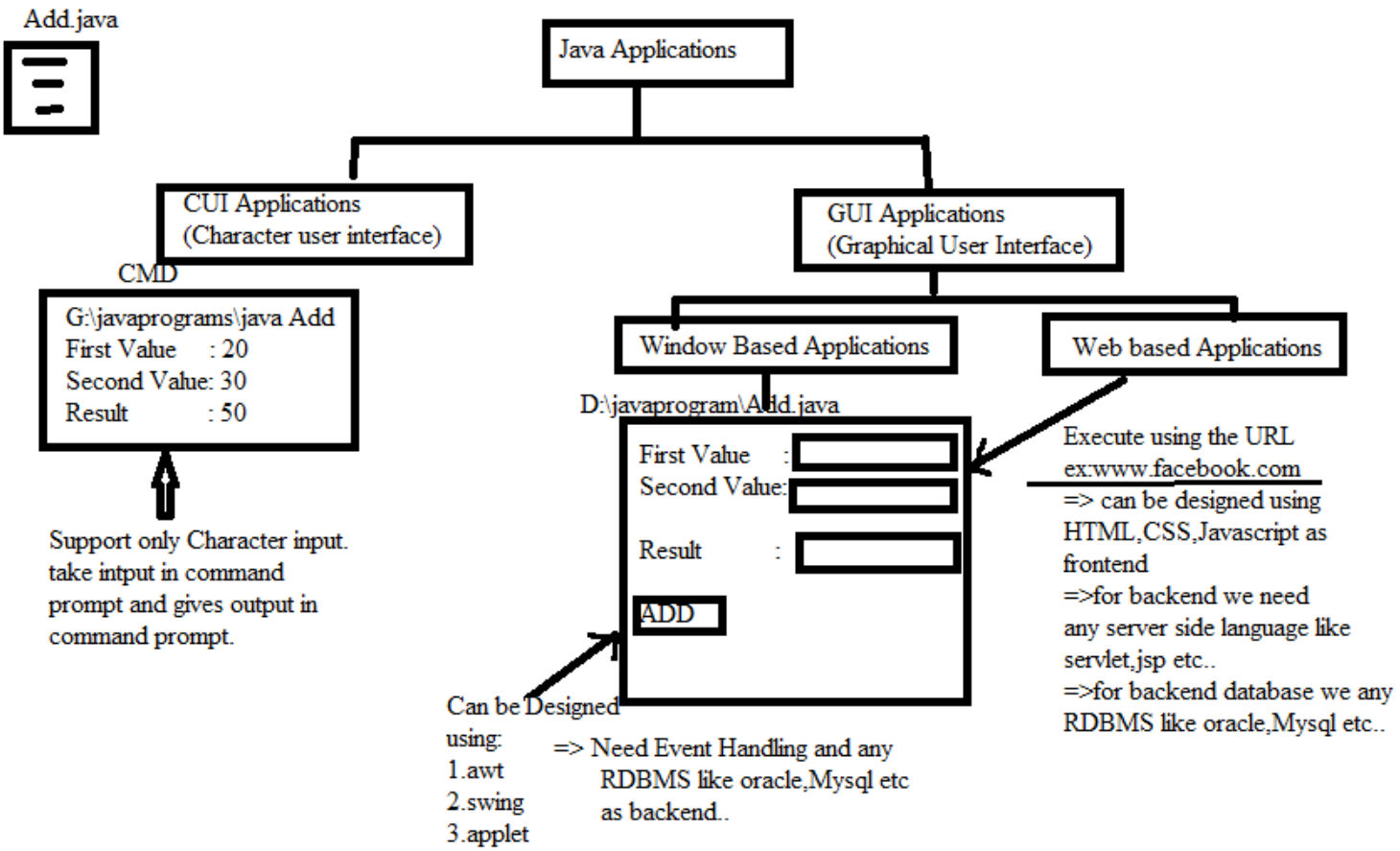
UNIT-5(Event Driven Programming)

Graphics programming:



GUI(Graphical user Interface): Window based GUI or Desktop application can be created using AWT, Swing, Applet using following java API

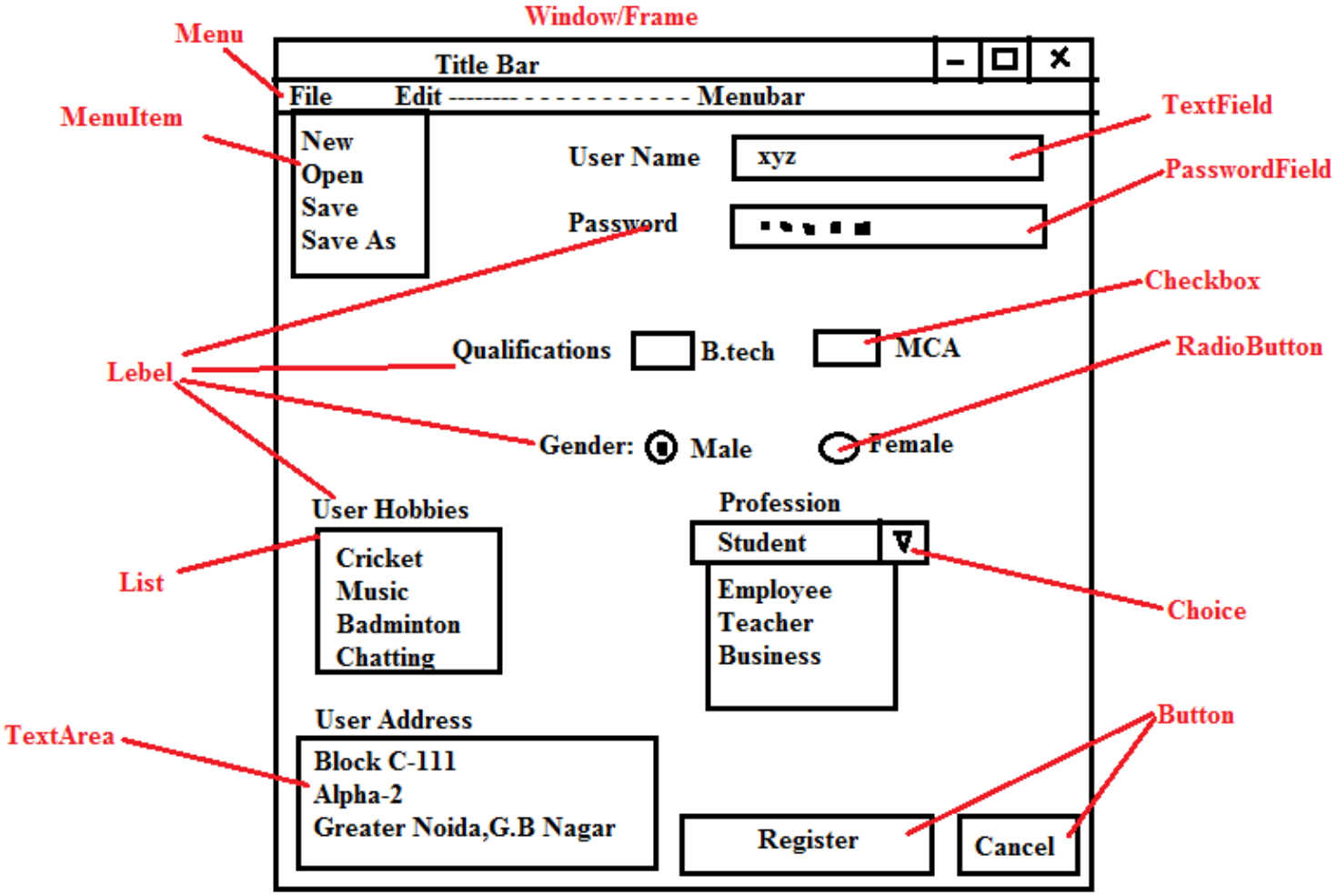
1.java.awt

2.javax.swing

3.java.applet

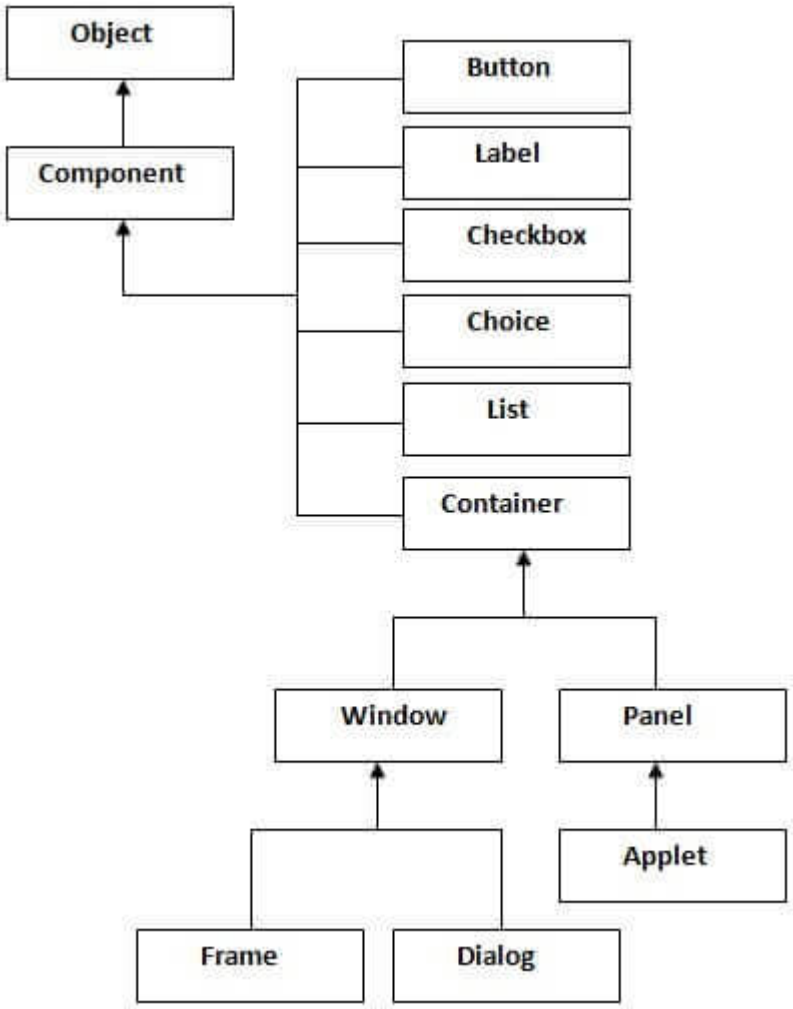
AWT(Abstract Window Toolkit):

=>AWT (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.
=>Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. =>AWT is heavyweight i.e. its components are using the resources of OS.
=>It is a Framework, it provides very good environment to prepare GUI Applications.
=>It is collection of predefined classes and interfaces to prepare Desktop or window based applications.
=>It is an API.
The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.



AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Container

The Container is a component in AWT that can contain another components like Frame, Dialog and Panel.

Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

Frame

The Frame is the container that contain title bar and can have menu bars. It is able to manage some other GUI components like button, textfield ,Label etc.

java.awt.Frame:

Frame class has following constructor.
public Frame()

Ex: Frame f=new frame();

=>It will create a Frame without title

public Frame(String title)
=>It creates a Frame with a specified title

Ex: Frame f=new frame("Login Frame");

Useful Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false. If we provide true, then Frame is visible. If we provide false then Frame is invisible.
public void setBackGround(Color c)	Provide Background color of a frame. Where c may be constants like RED,BLUE, GREEN,…. From java.awt.Color class.
public void setTitle(String title)	

AWT Example

To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

AWT Example by Association

Example of AWT where we are creating instance of Frame class.

Ex1:
```
Import java.awt.Frame;
 class FrameEx{
Public static void main(String args[]){
Frame f=new Frame();
}
}
```
Ex2:
```
import java.awt.*;
 class FrameEx{
public static void main(String args[]){
Frame f=new Frame();
f.setVisible(true);
f.setSize(500,200);
f.setTitle("First Frame");
f.setBackground(Color.green);
}
}
```

AWT Example by Inheritance

Example of AWT where we are inheriting Frame class.

```
import java.awt.*;
class FirstFrame extends Frame{
FirstFrame(){
this.setVisible(true);
this.setSize(1000,600);
this.setTitle("First Frame");
this.setBackground(Color.green);

}
public void paint(Graphics g){
        Font font=new Font("arial",Font.ITALIC+Font.BOLD,35);
        g.setFont(font);
   this.setForeground(Color.BLUE);
        g.drawString("Accurate institute of Management & Technology",150,80);

        }
}
class FrameExample{
public static void main(String args[]){
FirstFrame f=new FirstFrame();
}
}
```

Working with 2D shape:

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.
6. **public abstract void drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2).
7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used to fill a circular or elliptical arc.

10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.

11. **public abstract void setFont(Font font):** is used to set the graphics current font to the specified font.

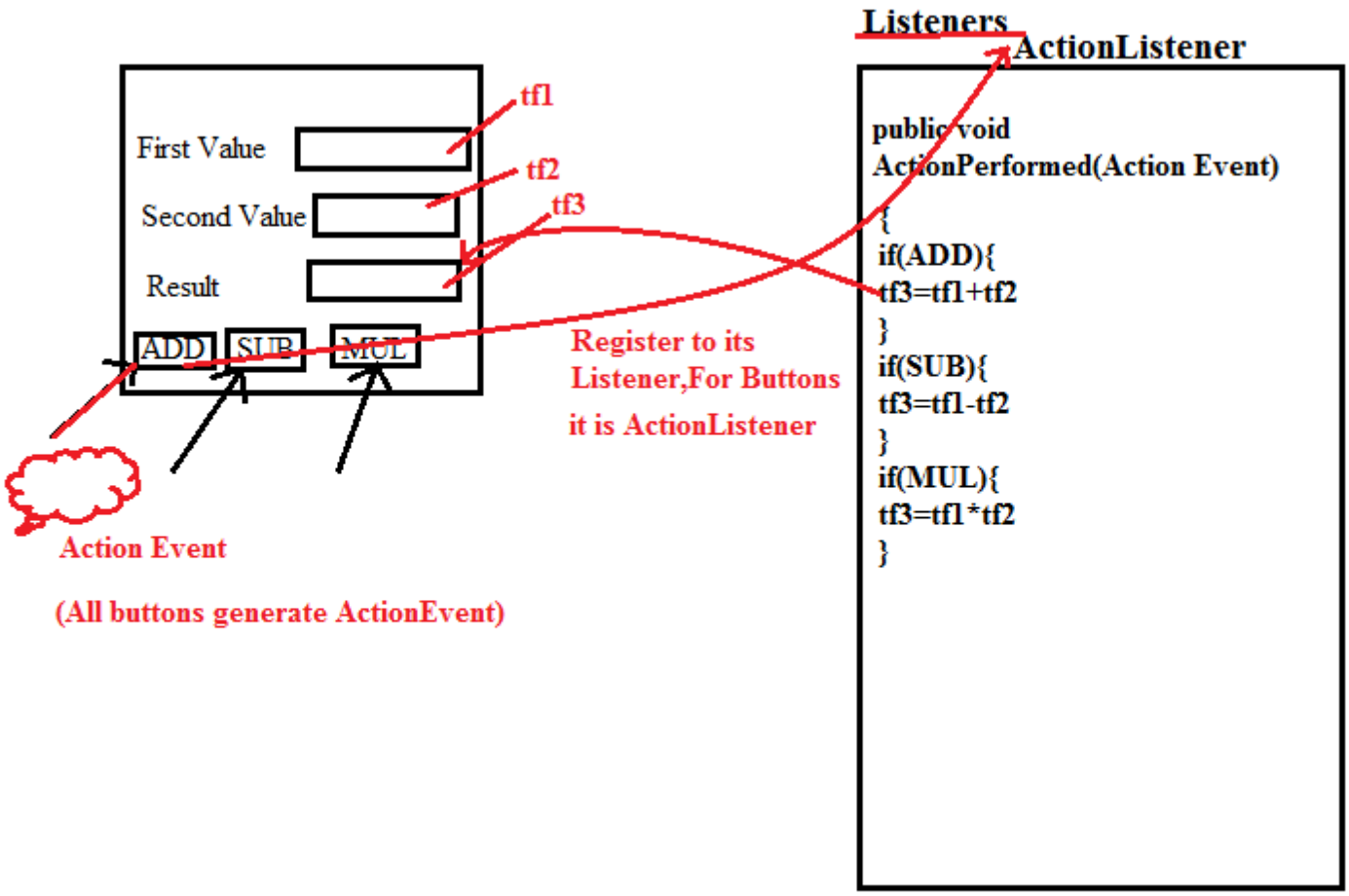
Ex:

```
import java.awt.*;
class FrameFrame extends Frame{
FirstFrame(){
this.setVisible(true);
this.setSize(1000,600);
this.setTitle("First Frame");
this.setBackground(Color.white);
}
public void paint(Graphics g){
      Font font=new Font("arial",Font.ITALIC+Font.BOLD,35);
      g.setFont(font);
      this.setForeground(Color.RED);
      g.drawString("Working with 2D shape",150,80);
       g.fillRect(130, 200,100, 80);
       g.drawOval(30,330,50, 60);
       g.fillOval(130,400,50, 60);
       g.drawArc(30, 400, 40,50,90,60);
       g.fillArc(30, 530, 40,50,180,40);

}
}
class ShapeExample{
public static void main(String args[]){
FirstFrame f=new FirstFrame();
}
}
```

Event Handling or Event Delegation Model:

=>Changing the state of an object is known as an event.
=>For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.



Event Handling or Event Delegation Model

GUI Componet	Event Name	Listener	Listener Methods
Button	ActionEvent	ActionListener	public void actionPerformed(ActionEvent e)
TextField	FocusEvent	FocusListener	public void focusGained(FocusEvent fe) public void focusLost(FocusEvent fe)
TextArea	FocusEvent	FocusListener	public void focusGained(FocusEvent fe) public void focusLost(FocusEvent fe)
Checkbox	ItemEvent	ItemListener	public void itemStateChanged(ItemEvent e)
Radio	ItemEvent	ItemListener	public void itemStateChanged(ItemEvent e)
List	ItemEvent	ItemListener	public void itemStateChanged(ItemEvent e)
Choice	ItemEvent	ItemListener	public void itemStateChanged(ItemEvent e)
Menu	ActionEvent	ActionListener	public void actionListener(ActionEvent ae)
ScrollBar	AdjustmentEvent	AdjustmentListener	public void adjustmentValueChanged(AdjusmentEvent ae)
Window	WindowEvent	WindowListener	public void windowOpened(WindowEvent we) public void windowClosed(WindowEvent we) public void windowClosing(WindowEvent we) public void windowIconified(WindowEvent we) public void windowActivated(WindowEvent we) public void windowDeactivated(WindowEvent we)

Mouse	**MouseEvent**	**MouseListner**	**public void mouseClicked(MouseEvent me)** **public void mousePressed(MouseEvent me)** **public void mouseReleased(MouseEvent me)** **public void mouseEntered(MouseEvent me)** **public void mouseExited(MouseEvent me)**
Keyboard	**KeyEvent**	**KeyListener**	**public void keyPressed(KeyEvent ke)** **public void keyReleased(KeyEvent ke)** **public void keyTyped(KeyEvent ke)**

Steps to perform Event Handling in GUI Applications

1. Create Container class and declare a no-argument constructor, inside the constructor declare a GUI component.
   Ex:
   Class MyFrame extends Frame{
   MyFrame(){
   Button b=new Button("Submit");

   }
   }

2.Select a listener and provide Listener implementation class with the implementation of Listener methods

```
class MyActionListener implements ActionListener{
  public void actionPerformed(ActionEvent ae){
----
}
}
```

Note:In general , in GUI applications we implements Listener interface in the same container class.
Ex:

```
Class MyFrame extends Frame implements ActionListener{
-----
}
```

3.Register Listener to GUI components for example Button:
Public void addxxxListener(xxxListener l)
xxxListener may be ActionListener,ItemListener,.....
Ex: b.addActionListener(new MyActionListener())

WindowEvents Handling:

```
import java.awt.*;
import java.awt.event.*;
class WindowListenerImpl implements WindowListener
{
public void windowOpened(WindowEvent we)
{
System.out.println("window opened");
}
public void windowClosed(WindowEvent we){
System.out.println("window Closed");
}
```

```
public void windowClosing(WindowEvent we){
System.out.println("window Closing");
System.exit(0);
}
public void windowIconified(WindowEvent we){
System.out.println("window Iconified");
}
public void windowDeiconified(WindowEvent we){
System.out.println("window Diconified");
}
public void windowActivated(WindowEvent we){
System.out.println("window Acitvated");
}
public void windowDeactivated(WindowEvent we){
System.out.println("window DeActivated");
}
}
class MyFrame extends Frame
{
MyFrame(){
this.setVisible(true);
this.setSize(500,500);
this.setTitle("window events example");
this.setBackground(Color.green);
this.addWindowListener(new WindowListenerImpl());
}
}
class WindowEventEx{
public static void main(String args[]){
MyFrame f=new MyFrame();
}
}
```

Adapter Classes

Adapter classes *provide the default implementation of listener interfaces*. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it *saves code*.

The adapter classes are found in **java.awt.event**, and **javax.swing.event** packages. The Some Adapter classes with their corresponding listener interfaces are given below.

java.awt.event Adapter classes

Adapter class	Listener interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener

Example:

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

```
class WindowListenerImpl extends WindowAdapter
{
public void windowClosing(WindowEvent we){
System.out.println("window closing");
System.exit(0);
}
}
class MyFrame extends Frame
{
MyFrame(){
this.setVisible(true);
this.setSize(500,500);
this.setTitle("window events example");
this.setBackground(Color.green);
this.addWindowListener(new WindowListenerImpl());
}
}
class WindowAdapterEx{
public static void main(String args[]){
MyFrame f=new MyFrame();
}
}
```

Same above example using Annonymous Class:

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame
{
MyFrame(){
this.setVisible(true);
this.setSize(500,500);
this.setTitle("window events example");
this.setBackground(Color.green);
this.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent we)
          {
     System.exit(0);
    }
});
}
}
class AnnonymousEx{
public static void main(String args[]){
MyFrame f=new MyFrame();
}
}
```

Mouse Event Handling:

```
import java.awt.*;
import java.awt.event.*;
class MouseListenerImpl implements MouseListener{
        public void mouseClicked(MouseEvent me){
                System.out.println("Mouse Clicked");
        }
        public void mousePressed(MouseEvent me){
                System.out.println("Mouse Pressed");
        }
```

```
	public void mouseReleased(MouseEvent me){
		System.out.println("Mouse Released");
	}
	public void mouseEntered(MouseEvent me){
		System.out.println("Mouse Entered");
	}
	public void mouseExited(MouseEvent me){
		System.out.println("Mouse Exited");
	}

}
class MyFrame extends Frame
{
MyFrame(){
this.setVisible(true);
this.setSize(500,500);
this.setTitle("window events example");
this.setBackground(Color.green);
this.addMouseListener(new MouseListenerImpl());
this.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent we){
System.exit(0);
	}
});
}
}
class MouseEventEx{
public static void main(String args[]){
MyFrame f=new MyFrame();
}
}
```

Button:

The button class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

Event Name: ActionEvent

Listener Name:ActionListener

Listener Method: public void ActionPerformed(ActionEvent ae)

Registration method: addActionListener(new ActionListener())

public void setBounds(int xaxis, int yaxis, int width, int height); have been used in the below example that sets the position of the component it may be button, textfield etc.

Example:

```
 import java.awt.*;
 import java.awt.event.*;
class ButtonEx extends Frame{
```

```
    ButtonEx(){
    Button b=new Button("Click Here");
    b.setBounds(50,100,80,30);
     this.add(b);
     this.setSize(400,400);
     this.setLayout(null);
     this.setVisible(true);
     this.setBackground(Color.GREEN);
     this.addWindowListener(new WindowAdapter(){
     public void windowClosing(WindowEvent we){
   System.exit(0);
         }
  });
        }
    }
class  ButtonExample{
        public static void main(String args[]){
                ButtonEx be=new ButtonEx();
        }
}
```

AWT Example:–

Label:-- Label is a control which shows text on window. We use following constructor to of Label class for creating Label.

1. Label()- Creates a label without any text.
2. Label(String s)- Creates a label with the given string.

Example:-

```
import java.awt.*;
class TestLabel extends Frame{
TestLabel(){
Label l=new Label("Welcome to Frame");
l.setBounds(30,100,80,30);// setting Label position
add(l);//adding button into frame
setSize(300,300);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
        }
public static void main(String args[]){
TestLabel t=new TestLabel();
}}
```

TextField:-- TextField control is used to create a TextField on the window. It uses following constructor of TextField class.

1. TextField()-creates a blank textfield.
2. TextField(String s)- creates a textfield with given string.

Example:--

```
import java.awt.*;
class TextFieldEx  extends Frame{
TextFieldEx(){
TextField   t=new TextField("Welcome to Frame");
t.setBounds(30,100,80,30);// setting TextField position
add(t);//adding button into frame
setSize(300,300);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
}
public static void main(String args[]){
TextFieldEx  t=new TextFieldEx();
```

```
}}
```

Button:-

```
import java.awt.*;
class First extends Frame{
First(){
Button b=new Button("click me");
b.setBounds(30,100,80,30);// setting button position
add(b);//adding button into frame
setSize(300,300);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
}
public static void main(String args[]){
First f=new First();
}}
```

Example Event Handling of button using ActionListener

```
   import java.awt.*;
   import java.awt.event.*;
    class ButtonEx extends Frame{
      ButtonEx(){
      Button b=new Button("Click Here");
      TextField tf=new TextField();
      tf.setBounds(50,50, 150,20);
      b.setBounds(50,100,60,30);
      this.setLayout(null);
      this.add(b);
      this.add(tf);
      this.setSize(400,400);
      this.setVisible(true);
      this.setBackground(Color.GREEN);
      this.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent we){
      System.exit(0);
          }
  });
  b.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae){
            tf.setText("Welcome to AWT");
            }
  });
         }
    }
class  ButtonEvent{
          public static void main(String args[]){
          ButtonEx be=new ButtonEx();
          }
}
```

Q.write a program in java take two three text field ,three label and a button on pressing button sum of two text field should Display in third text field.

```
import java.awt.*;
import java.awt.event.*;
class First extends Frame implements ActionListener{
Button b;
```

```
TextField tf1,tf2,tf3;
First f;
First(){
Label l1=new Label("Enter first Number:");
l1.setBounds(10,50,150,30);
add(l1);
tf1=new TextField();
tf1.setBounds(200,50,200,30);
add(tf1);
Label l2=new Label("Enter Second Number:");
l2.setBounds(10,90,150,30);
add(l2);
tf2=new TextField();
tf2.setBounds(200,90,200,30);
add(tf2);
Label l3=new Label("Sum of Numbers:");
l3.setBounds(10,150,150,30);
add(l3);
tf3=new TextField();
tf3.setBounds(200,150,200,30);
add(tf3);
b=new Button("click me");
b.setBounds(200,200,80,30);// setting button position
add(b);//adding button into frame
b.addActionListener(this);
setSize(1200,1200);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
}
public void actionPerformed(ActionEvent e){
        int a= Integer.parseInt(tf1.getText());
        int b= Integer.parseInt(tf2.getText());
        int c=a+b;
        tf3.setText(Integer.toString(c));
}
  public static void main(String args[]){
  First   f=new First();
                  }
            }
```

Swing :

is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

Java AWT	Java Swing
1) AWT components are platform-dependent.	Java swing components are platform-independent.

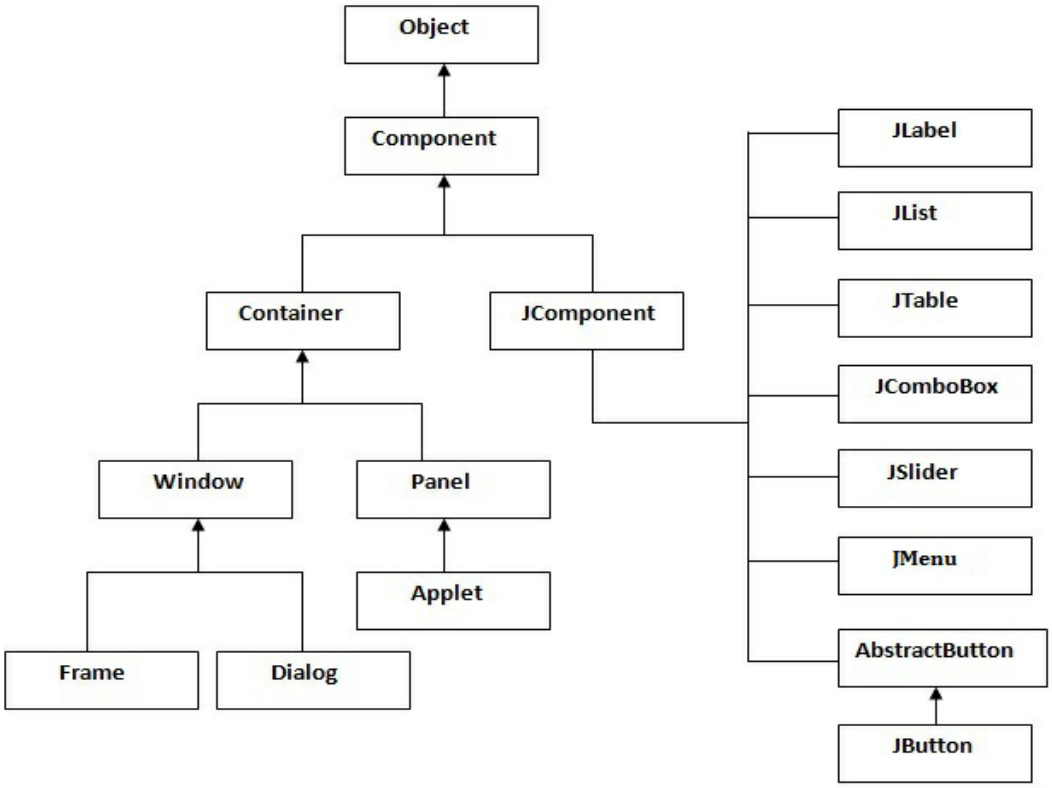
2) AWT components are heavyweight.	Swing components are lightweight.
3) AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel.
4) AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5) 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.

What is JFC:—

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Hierarchy of javax.swing

The hierarchy of java swing API is given below.



Components and Containers

A Swing GUI consists of two key items: *components* and *containers*. However, this distinction is mostly conceptual because all containers are also components. The difference between the two is found in their intended purpose: As the term is commonly used, a *component* is an independent visual control, such as a push button or slider. A container holds a group of components. Thus, a container is a special type of component that is designed to hold other components. Furthermore, in order for a component to be displayed, it must be held within a container. Thus, all Swing GUIs will have at least one container. Because containers are components, a container can also hold other containers. This enables Swing to define what is called a *containment hierarchy*, at the top of which must be a *top-level container*.

Components

In general, Swing components are derived from the **JComponent** class. **JComponent** provides the functionality that is common to all components. For example, **JComponent** supports the pluggable look and feel. **JComponent** inherits the AWT classes **Container** and **Component**. Thus, a Swing component is built on and compatible with an AWT component. All of Swing's components are represented by classes defined within the package **javax.swing**.

Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

JButton

The JButton 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.

Commonly used Constructors:

Constructor	Description
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.

Commonly used Methods of AbstractButton class:

Methods	Description
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 addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JButton Example

```
import javax.swing.*;
public class JButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Java JButton Example with ActionListener

```
import java.awt.event.*;
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    final JTextField tf=new JTextField();
    tf.setBounds(50,50, 150,20);
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    b.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
            tf.setText("Welcome to Java Swing.");
        }
    });
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Example of displaying image on the button:

```
import javax.swing.*;
import java.awt.event.*;
 class JButtonExample extends JFrame implements ActionListener {
 JTextField t;
JButtonExample(){
            t=new JTextField();
            ImageIcon ii=new ImageIcon("C:\\awt2021\\swing\\login.jpg");
JButton b=new JButton(ii);
JLabel l=new JLabel("show result:");
    b.setBounds(150,100,120,80);
    t.setBounds(150,50,150,40);
            l.setBounds(50,50,100,40);
    this.add(b);
            this.add(t);
            this.add(l);
```

```
   this.setSize(500,400);
   this.setLayout(null);
   this.setVisible(true);
   this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  /* b.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
              t.setText("Welcome to Swing");
        }
});*/
b.addActionListener(this);
}
public void actionPerformed(ActionEvent e){
              t.setText("Welcome to Swing Program");
        }
}
class JButtonEx{
public static void main(String[] args) {
   JButtonExample jb=new JButtonExample();

}
}
```

JLabel

The object of JLabel class is a component for placing text in a container. 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.

Commonly used Constructors:

Constructor	Description
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.

Commonly used Methods:

Methods	Description
String getText()	t 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.

JLabel Example

```
import javax.swing.*;
class LabelExample
{
public static void main(String args[])
   {
   JFrame f= new JFrame("Label Example");
   JLabel l1,l2;
```

```
l1=new JLabel("First Label.");
l1.setBounds(50,50, 100,30);
l2=new JLabel("Second Label.");
l2.setBounds(50,100, 100,30);
f.add(l1); f.add(l2);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
}
```

JTextField

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

Commonly used Constructors:

Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

Commonly used Methods:

Methods	Description
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.

JTextField Example

```
import javax.swing.*;
class TextFieldExample
{
public static void main(String args[])
  {
```

```
    JFrame f= new JFrame("TextField Example");
    JTextField t1,t2;
    t1=new JTextField("Welcome to Javatpoint.");
    t1.setBounds(50,100, 200,30);
    t2=new JTextField("AWT Tutorial");
    t2.setBounds(50,150, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```

JTextField Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class TextFieldExample implements ActionListener{
    JTextField tf1,tf2,tf3;
    JButton b1,b2;
    TextFieldExample(){
        JFrame f= new JFrame();
        tf1=new JTextField();
        tf1.setBounds(50,50,150,20);
        tf2=new JTextField();
        tf2.setBounds(50,100,150,20);
        tf3=new JTextField();
        tf3.setBounds(50,150,150,20);
        tf3.setEditable(false);
        b1=new JButton("+");
        b1.setBounds(50,200,50,50);
        b2=new JButton("-");
        b2.setBounds(120,200,50,50);
        b1.addActionListener(this);
        b2.addActionListener(this);
        f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        String s1=tf1.getText();
        String s2=tf2.getText();
        int a=Integer.parseInt(s1);
        int b=Integer.parseInt(s2);
        int c=0;
        if(e.getSource()==b1){
            c=a+b;
        }else if(e.getSource()==b2){
            c=a-b;
        }
        String result=String.valueOf(c);
        tf3.setText(result);
    }
public static void main(String[] args) {
    TextFieldExample tf=new TextFieldExample();
} }
```

JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

Commonly used Constructors:

Constructor	Description
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.

Commonly used Methods:

Methods	Description
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.

JTextArea Example

```
import javax.swing.*;
public class TextAreaExample
{
    TextAreaExample(){
      JFrame f= new JFrame();
      JTextArea area=new JTextArea("Welcome to javatpoint");
      area.setBounds(10,30, 200,200);
      f.add(area);
      f.setSize(300,300);
      f.setLayout(null);
      f.setVisible(true);
    }
public static void main(String args[])
   {
  TextAreaExample ta=new TextAreaExample();
   }}
```

JTextArea Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class TextAreaExample implements ActionListener{
JLabel l1,l2;
JTextArea area;
JButton b;
TextAreaExample() {
    JFrame f= new JFrame();
    l1=new JLabel();
    l1.setBounds(50,25,100,30);
    l2=new JLabel();
    l2.setBounds(160,25,100,30);
    area=new JTextArea();
    area.setBounds(20,75,250,200);
    b=new JButton("Count Words");
    b.setBounds(100,300,120,30);
    b.addActionListener(this);
    f.add(l1);f.add(l2);f.add(area);f.add(b);
    f.setSize(450,450);
    f.setLayout(null);
    f.setVisible(true);
}
public void actionPerformed(ActionEvent e){
    String text=area.getText();
    String words[]=text.split("\\s");
    l1.setText("Words: "+words.length);
    l2.setText("Characters: "+text.length());
}
public static void main(String[] args) {
TextAreaExample ta=new TextAreaExample();
}
}
```

JPasswordField

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

Commonly used Constructors:

Constructor	Description
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.

JPasswordField Example

```
import javax.swing.*;
public class PasswordFieldExample {
```

```
   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);
}
}
```

JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

Commonly used Constructors:

Constructor	Description
JJCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(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.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

Commonly used Methods:

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a string representation of this JCheckBox.

JCheckBox Example

```
import javax.swing.*;
public class CheckBoxExample
{
    CheckBoxExample(){
      JFrame f= new JFrame("CheckBox Example");
      JCheckBox checkBox1 = new JCheckBox("C++");
      checkBox1.setBounds(100,100, 50,50);
      JCheckBox checkBox2 = new JCheckBox("Java", true);
      checkBox2.setBounds(100,150, 50,50);
      f.add(checkBox1);
      f.add(checkBox2);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
```

```
    }
public static void main(String args[])
    {
   CheckBoxExample cb= new CheckBoxExample();
    }}
```

JCheckBox Example with ItemListener

```
import javax.swing.*;
import java.awt.event.*;
public class CheckBoxExample
{
     CheckBoxExample(){
       JFrame f= new JFrame("CheckBox Example");
       final JLabel label = new JLabel();
       label.setHorizontalAlignment(JLabel.CENTER);
       label.setSize(400,100);
       JCheckBox checkbox1 = new JCheckBox("C++");
       checkbox1.setBounds(150,100, 50,50);
       JCheckBox checkbox2 = new JCheckBox("Java");
       checkbox2.setBounds(150,150, 50,50);
       f.add(checkbox1); f.add(checkbox2); f.add(label);
       checkbox1.addItemListener(new ItemListener() {
           public void itemStateChanged(ItemEvent e) {
             label.setText("C++ Checkbox: "
             + (e.getStateChange()==1?"checked":"unchecked"));
           }
         });
       checkbox2.addItemListener(new ItemListener() {
           public void itemStateChanged(ItemEvent e) {
             label.setText("Java Checkbox: "
             + (e.getStateChange()==1?"checked":"unchecked"));
           }
         });
       f.setSize(400,400);
       f.setLayout(null);
       f.setVisible(true);
    }
public static void main(String args[])
{
    CheckBoxExample cb=new CheckBoxExample();
}
}
```

JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

Commonly used Constructors:

Constructor	Description
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.

Commonly used Methods:

Methods	Description
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 addActionListener(ActionListener a)	It is used to add the action listener to this object.

JRadioButton Example

```
import javax.swing.*;
public class RadioButtonExample {
JFrame f;
RadioButtonExample(){
f=new JFrame();
JRadioButton r1=new JRadioButton("A) Male");
JRadioButton r2=new JRadioButton("B) Female");
r1.setBounds(75,50,100,30);
r2.setBounds(75,100,100,30);
ButtonGroup bg=new ButtonGroup();
bg.add(r1);bg.add(r2);
f.add(r1);f.add(r2);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args) {
RadioButton rb=  new RadioButtonExample();
}
}
```

JRadioButton Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
class RadioButtonExample extends JFrame implements ActionListener{
JRadioButton rb1,rb2;
JButton b;
RadioButtonExample(){
rb1=new JRadioButton("Male");
rb1.setBounds(100,50,100,30);
```

```
rb2=new JRadioButton("Female");
rb2.setBounds(100,100,100,30);
ButtonGroup bg=new ButtonGroup();
bg.add(rb1);bg.add(rb2);
b=new JButton("click");
b.setBounds(100,150,80,30);
b.addActionListener(this);
add(rb1);add(rb2);add(b);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){
if(rb1.isSelected()){
JOptionPane.showMessageDialog(this,"You are Male.");
}
if(rb2.isSelected()){
JOptionPane.showMessageDialog(this,"You are Female.");
}
}
public static void main(String args[]){
RadioButtonExample rb= new RadioButtonExample();
}}
```

JComboBox

The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits JComponent class.

Commonly used Constructors:

Constructor	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified array.

Commonly used Methods:

Methods	Description
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.

Java JComboBox Example

```
import javax.swing.*;
public class ComboBoxExample {
JFrame f;
```

```
ComboBoxExample(){
   f=new JFrame("ComboBox Example");
   String country[]={"India","Aus","U.S.A","England","Newzealand"};
   JComboBox cb=new JComboBox(country);
   cb.setBounds(50, 50,90,20);
   f.add(cb);
   f.setLayout(null);
   f.setSize(400,500);
   f.setVisible(true);
}
public static void main(String[] args) {
  ComboBoxExample  cb=  new ComboBoxExample();
}
}
```

JComboBox Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class ComboBoxExample {
JFrame f;
ComboBoxExample(){
   f=new JFrame("ComboBox Example");
   final JLabel label = new JLabel();
   label.setHorizontalAlignment(JLabel.CENTER);
   label.setSize(400,100);
   JButton b=new JButton("Show");
   b.setBounds(200,100,75,20);
   String languages[]={"C","C++","C#","Java","PHP"};
   final JComboBox cb=new JComboBox(languages);
   cb.setBounds(50, 100,90,20);
   f.add(cb); f.add(label); f.add(b);
   f.setLayout(null);
   f.setSize(350,350);
   f.setVisible(true);
   b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
String data = "Programming language Selected: "
  + cb.getItemAt(cb.getSelectedIndex());
label.setText(data);
}
});
}
public static void main(String[] args) {
  ComboBoxExample cb= new ComboBoxExample();
}
}
```

JList

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

Commonly used Constructors:

Constructor	Description
JList()	Creates a JList with an empty, read-only, model.
JList(ary[] listData)	Creates a JList that displays the elements in the specified array.

Commonly used Methods:

Methods	Description
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.

JList Example

```
import javax.swing.*;
public class ListExample
{
    ListExample(){
      JFrame f= new JFrame();
      DefaultListModel<String> l1 = new DefaultListModel<>();
        l1.addElement("Item1");
        l1.addElement("Item2");
        l1.addElement("Item3");
        l1.addElement("Item4");
        JList<String> list = new JList<>(l1);
        list.setBounds(100,100, 75,75);
        f.add(list);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
   {
  ListExample le=new ListExample();
   }}
```

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.

Commonly used Constructors:

Constructor	Description
JScrollBar()	Creates a vertical scrollbar with the initial values.
JScrollBar(int orientation)	Creates a scrollbar with the specified orientation and the initial values.
JScrollBar(int orientation, int value, int extent, int min, int max)	Creates a scrollbar with the specified orientation, value, extent, minimum, and maximum.

JScrollBar Example

```
import javax.swing.*;
class ScrollBarExample
{
ScrollBarExample(){
    JFrame f= new JFrame("Scrollbar Example");
 JScrollBar s=new JScrollBar();
s.setBounds(100,100, 50,100);
f.add(s);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String args[])
{
ScrollBarExample sb=new ScrollBarExample();
}}
```

JMenuBar, JMenu and JMenuItem

The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

JMenuItem and JMenu Example

```
import javax.swing.*;
class MenuExample
{
        JMenu menu, submenu;
        JMenuItem i1, i2, i3, i4, i5;
        MenuExample(){
        JFrame f= new JFrame("Menu and MenuItem Example");
        JMenuBar mb=new JMenuBar();
        menu=new JMenu("Menu");
        submenu=new JMenu("Sub Menu");
        i1=new JMenuItem("Item 1");
        i2=new JMenuItem("Item 2");
        i3=new JMenuItem("Item 3");
        i4=new JMenuItem("Item 4");
        i5=new JMenuItem("Item 5");
        menu.add(i1); menu.add(i2); menu.add(i3);
        submenu.add(i4); submenu.add(i5);
        menu.add(submenu);
        mb.add(menu);
        f.setJMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
}
public static void main(String args[])
{
MenuExample  me=new MenuExample();
```

}}

JDialog

The JDialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Dialog class.

Unlike JFrame, it doesn't have maximize and minimize buttons.

Commonly used Constructors:

Constructor	Description
JDialog()	It is used to create a modeless dialog without a title and without a specified Frame owner.
JDialog(Frame owner)	It is used to create a modeless dialog with specified Frame as its owner and an empty title.
JDialog(Frame owner, String title, boolean modal)	It is used to create a dialog with the specified title, owner Frame and modality.

JDialog Example

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
   private static JDialog d;
   DialogExample() {
      JFrame f= new JFrame();
      d = new JDialog(f , "Dialog Example", true);
      d.setLayout( new FlowLayout() );
      JButton b = new JButton ("OK");
      b.addActionListener ( new ActionListener()
      {
         public void actionPerformed( ActionEvent e )
         {
            DialogExample.d.setVisible(false);
         }
      });
      d.add( new JLabel ("Click button to continue."));
      d.add(b);
      d.setSize(300,300);
      d.setVisible(true);
   }
   public static void main(String args[])
   {
      DialogExample de=new DialogExample();
   }
}
```

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

BorderLayout:

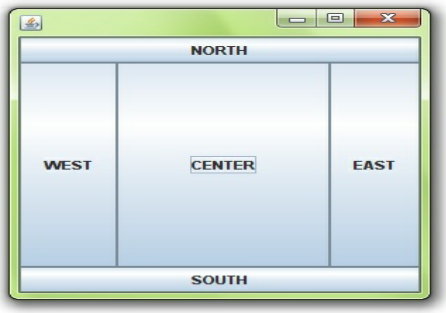
The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. The BorderLayout

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.

Example of BorderLayout class:




```
import java.awt.*;
import javax.swing.*;
public class Border {
JFrame f;
Border(){
   f=new JFrame();
   JButton b1=new JButton("NORTH");;
   JButton b2=new JButton("SOUTH");;
   JButton b3=new JButton("EAST");;
   JButton b4=new JButton("WEST");;
   JButton b5=new JButton("CENTER");;
   f.add(b1,BorderLayout.NORTH);
   f.add(b2,BorderLayout.SOUTH);
   f.add(b3,BorderLayout.EAST);
   f.add(b4,BorderLayout.WEST);
   f.add(b5,BorderLayout.CENTER);
   f.setSize(300,300);
   f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public static void main(String[] args) {
   new Border();
}}
```

GridLayout

The GridLayout 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.
Example of GridLayout class:


```
import java.awt.*;
import javax.swing.*;
public class MyGridLayout{
JFrame f;
MyGridLayout(){
    f=new JFrame();
    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");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
    f.add(b6);f.add(b7);f.add(b8);f.add(b9);
   f.setLayout(new GridLayout(3,3));
    //setting grid layout of 3 rows and 3 columns
   f.setSize(300,300);
    f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public static void main(String[] args) {
    new MyGridLayout();
} }
```

FlowLayout:

The FlowLayout is used to arrange the components in a line, one after another (in a flow). It is the default layout of applet or panel.

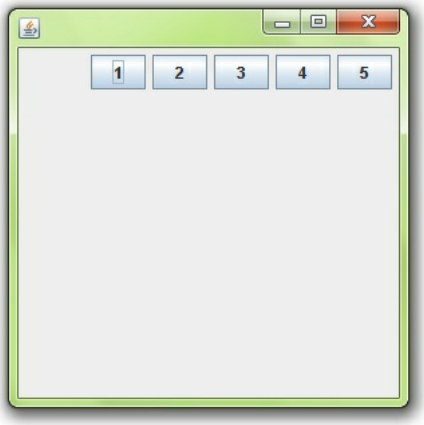
Fields of FlowLayout class:

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

Constructors of FlowLayout class:

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.

Example of FlowLayout class:




```
import java.awt.*;
import javax.swing.*;
  public class MyFlowLayout{
JFrame f;
MyFlowLayout(){
    f=new JFrame();
      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");
      f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
          f.setLayout(new FlowLayout(FlowLayout.RIGHT));
    //setting flow layout of right alignment
      f.setSize(300,300);
    f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public static void main(String[] args) {
    new MyFlowLayout();
}}
```

BoxLayout class:

The BoxLayout is used to arrange the components either vertically or horizontally. For this purpose, BoxLayout provides four constants. They are as follows:

Note: BoxLayout class is found in javax.swing package.

Fields of BoxLayout class:

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 of BoxLayout class:

BoxLayout(Container c, int axis): creates a box layout that arranges the components with the given axis.

Example of BoxLayout class with Y-AXIS:


```
import java.awt.*;
import javax.swing.*;
public class BoxLayoutExample2 extends Frame {
 Button buttons[];
 public BoxLayoutExample2() {
   buttons = new Button [5];
   for (int i = 0;i<5;i++) {
     buttons[i] = new Button ("Button " + (i + 1));
     add (buttons[i]);
    }
setLayout (new BoxLayout(this, BoxLayout.X_AXIS));
setSize(400,400);
setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public static void main(String args[]){
BoxLayoutExample2 b=new BoxLayoutExample2();
} }
```

CardLayout class:

The CardLayout class manages the components in such a manner that only one component is visible at a time. It treats each component as a card that is why it is known as CardLayout.

Constructors of CardLayout class:

CardLayout(): creates a card layout with zero horizontal and vertical gap.
CardLayout(int hgap, int vgap): creates a card layout with the given horizontal and vertical gap.

Commonly used methods of CardLayout class:

public void next(Container parent): is used to flip to the next card of the given container.
public void previous(Container parent): is used to flip to the previous card of the given container.
public void first(Container parent): is used to flip to the first card of the given container.
public void last(Container parent): is used to flip to the last card of the given container.
public void show(Container parent, String name): is used to flip to the specified card with the given name.

Example of CardLayout class:


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

```
 import javax.swing.*;
 public class CardLayoutExample extends JFrame implements ActionListener{
CardLayout card;
JButton b1,b2,b3;
Container c;
   CardLayoutExample(){
        c=getContentPane();
      card=new CardLayout(40,30);
//create CardLayout object with 40 hor space and 30 ver space
      c.setLayout(card);
       b1=new JButton("Apple");
      b2=new JButton("Boy");
      b3=new JButton("Cat");
      b1.addActionListener(this);
      b2.addActionListener(this);
      b3.addActionListener(this);
       c.add("a",b1);c.add("b",b2);c.add("c",b3);
    }
   public void actionPerformed(ActionEvent e) {
   card.next(c);
   }
   public static void main(String[] args) {
      CardLayoutExample cl=new CardLayoutExample();
      cl.setSize(400,400);
      cl.setVisible(true);
      cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
   } }
```