OBJECT ORIENTED PROGRAMMING USING JAVA
Paper Code: CSCS231

MODULE – III

GUI components –Overview of Swing components –Displaying Text and Images in a Window - Text Fields , Introduction to Event Handling- GUI Event Types and Listener Interfaces - layout manager, Swings Vs AWT

Graphical User Interface (GUI)

Applet with AWT (Abstract Window Toolkit) and Swing are the Graphical User Interface (GUI) in java. Applets are small programs transferred through Internet, automatically installed and run as part of web-browser. Applets implements functionality of a client. Applet is a dynamic and interactive program that runs inside a Web page displayed by a Java-capable browser. We don't have the concept of Constructors in Applets. Applets can be invoked either through browser or through Appletviewer utility provided by JDK.

Life Cycle of Applet

► init() method - called when an applet is first loaded. This method is called only once in the entire cycle of an applet. This method usually intialize the variables to be used in the applet.

► start( ) method - called each time an applet is started.

► paint() method - called when the applet is minimized or refreshed. This method is used for drawing different strings, figures, and images on the applet window.

► stop( ) method - called when the browser moves off the applet's page.

► destroy( ) method - called when the browser is finished with the applet.

Advantage of Applet

There are many advantages of applet. They are as follows:

It works at client side so less response time.

Secured

It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

Plugin is required at client browser to execute applet.

Applet Example

```
import java.applet.Applet;
import java.awt.*;
public class GraphicsDemo extends Applet{
public void paint(Graphics g){
        g.setColor(Color.red);
        g.drawString("Welcome",50, 50);
        g.drawLine(20,30,20,300);
        g.drawRect(70,100,30,30);
        g.fillRect(170,100,30,30);
        g.drawOval(70,200,30,30);
        g.setColor(Color.pink);
        g.fillOval(170,200,30,30);
        g.drawArc(90,150,30,30,30,270);
        g.fillArc(270,150,30,30,0,180);
}
}
```

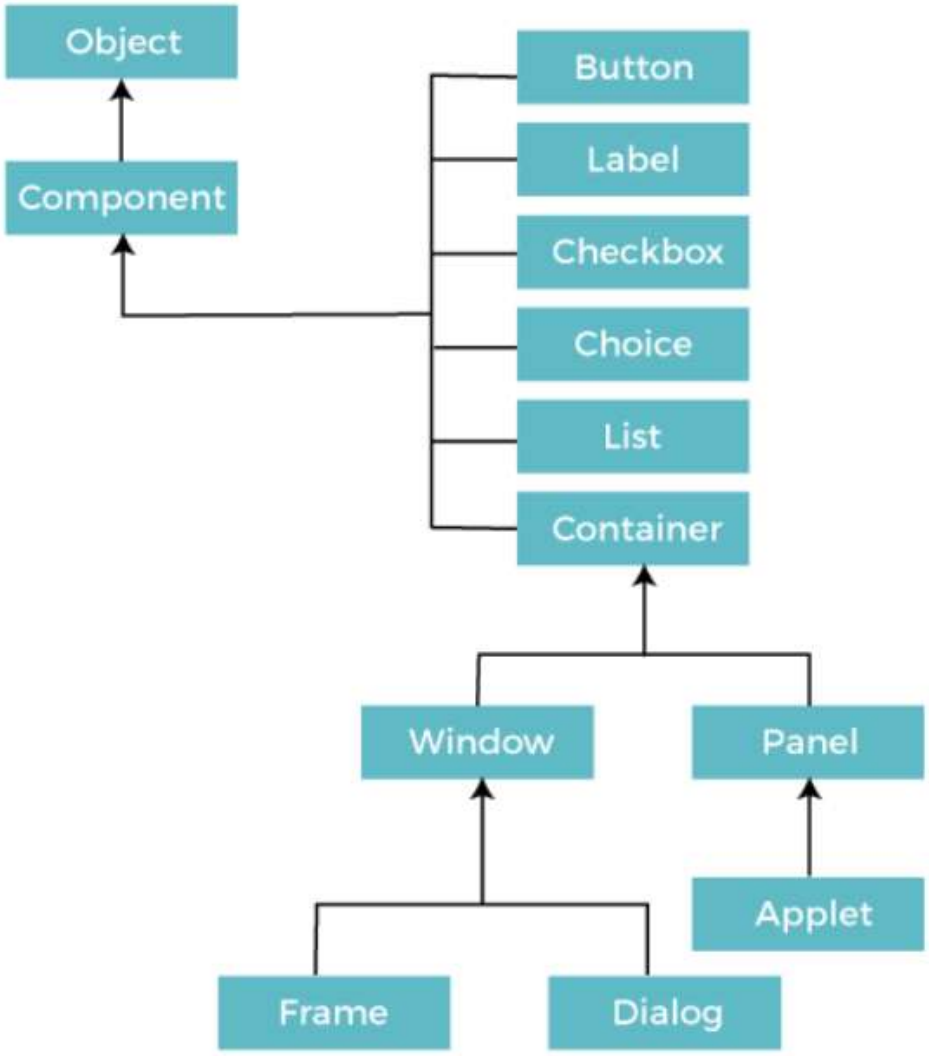
Run Applet using HTML file

```
<html>
<body>
<applet code="GraphicsDemo.class" width="300" height="300">
</applet>
</body>
</html>
```

AWT Component

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).



The Swing

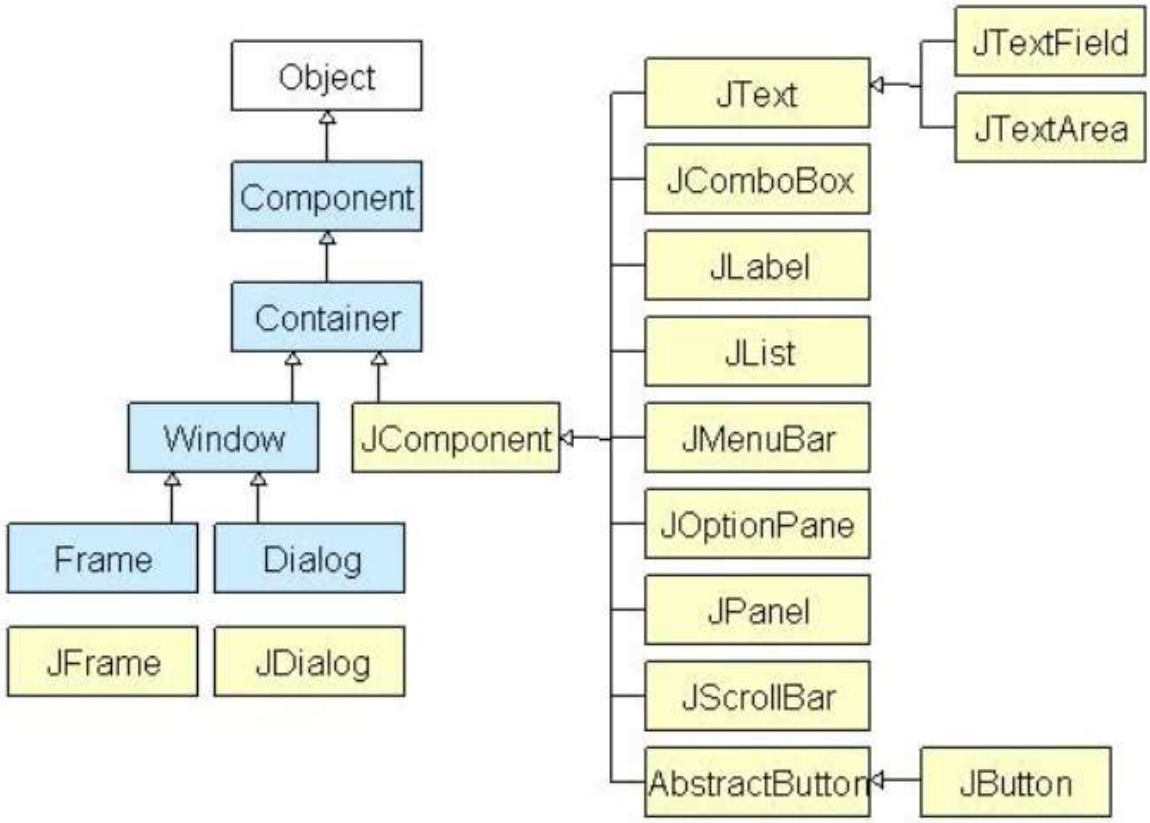
Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT, API and entirely written in java.

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.

Swing in Java is a Graphical User Interface (GUI) toolkit that includes the GUI components. Swing provides a rich set of objects to make good GUI components for Java applications. Swing is a part of Java Foundation Classes(JFC), which is an API for Java GUI programing that provide GUI.

Java Swing class Hierarchy Diagram



Components

All the elements like the button, text fields, scroll bars, etc. are called components. In Java AWT, there are classes for each component as shown in above diagram. In order to place every component in a particular position on a screen, we need to add them to a container.

Container

Container classes are classes that can have other components on it. So for creating a Java Swing GUI, we need at least one container object. There are 3 types of Java Swing containers.

1. **Panel**: It is a pure container and is not a window in itself. The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.
2. **Frame**: It is a fully functioning window with its title and icons. The Frame is the container that contain title bar and border and can have menu bars. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.
3. **Dialog**: It can be thought of like a pop-up window that pops out when a message has to be displayed. It is not a fully functioning window like the Frame.

Control	Class	Description
Label	JLabel	Putting plain text in a container
Text box	JTextField	Single line text box
Text area	JTextArea	Text box with Multiple Line
Button	JButton	used to perform some operations when the user clicks on it
Check box	JCheckBox	Lets the user select or deselect an option.
Radio button	JRadioButton	Lets the user select an option from a group of options.
List	JList	Lets the user select one or more items from a list of items.
Combo box	JComboBox	Lets the user select a single item from a drop-down list of items. A combo box can also let the user enter text into the text portion of the combo box.

JText Field

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.

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);
   }
   }
```

Images in Windows

For displaying image, we can use the method drawImage() of Graphics class.

drawImage(Image img, int x, int y, ImageObserver observer): is used draw the specified image.

```
import java.awt.*;
import javax.swing.JFrame;
public class MyCanvas extends Canvas{
        public void paint(Graphics g) {
                Toolkit t=Toolkit.getDefaultToolkit();
                Image i=t.getImage("p3.gif");
                g.drawImage(i, 50,60,this);
        }
                public static void main(String[] args) {
                MyCanvas m=new MyCanvas();
                JFrame f=new JFrame();
                f.add(m);
                f.setSize(400,400);
                //f.setLayout(null);
                f.setVisible(true);
        }

}
```

Event and Listener

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 Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

Button

```
public void addActionListener(ActionListener a){}
```

MenuItem

```
public void addActionListener(ActionListener a){}
```

TextField

```
public void addActionListener(ActionListener a){}
public void addTextListener(TextListener a){}
```

TextArea

```
public void addTextListener(TextListener a){}
```

Checkbox

```
public void addItemListener(ItemListener a){}
```

Choice

public void addItemListener(ItemListener a){}

List

public void addActionListener(ActionListener a){}

public void addItemListener(ItemListener a){}

Example

```
import java.awt.*;
import java.awt.event.*;
class MyText extends Frame implements ActionListener{
TextField tf;
MyTextt(){
//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
Button b=new Button("click me");
b.setBounds(100,120,80,30);

//register listener
b.addActionListener(this);//passing current instance

//add components and set size, layout and visibility
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
public static void main(String args[]){
new MyText();
}
}
```

Java Layout Manager

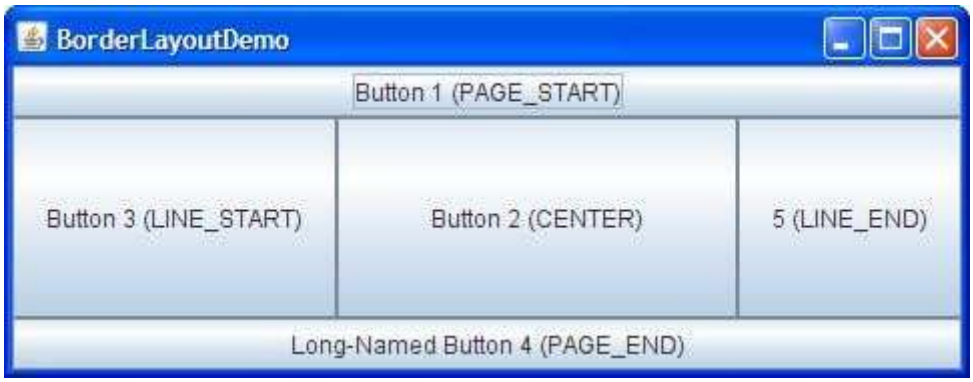
The layout is used to enhance the look and feel of the application. To arrange the components in a container, the various layout classes can be used such as Flow layout and Border Layout. These layouts use relative positioning to place the components on the container, which means the components automatically adjust their position according to the frame size. For example, setLayout is a method used to set the layout of the containers. Java provides the various layouts.

- Border Layout
- Flow Layout
- Grid and Grid Bag Layout
- Grid Bag Layout
- Card Layout

Java BorderLayout

A BorderLayout places components in up to five areas: top, bottom, left, right, and center. It is the default layout manager for every java JFrame.

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 a frame or window. The BorderLayout provides five constants for each region:



Flow Layout

FlowLayout is the default layout manager for every JPanel. It simply lays out components in a single row one after the other.

Java FlowLayout

Grid and GridBagLayout

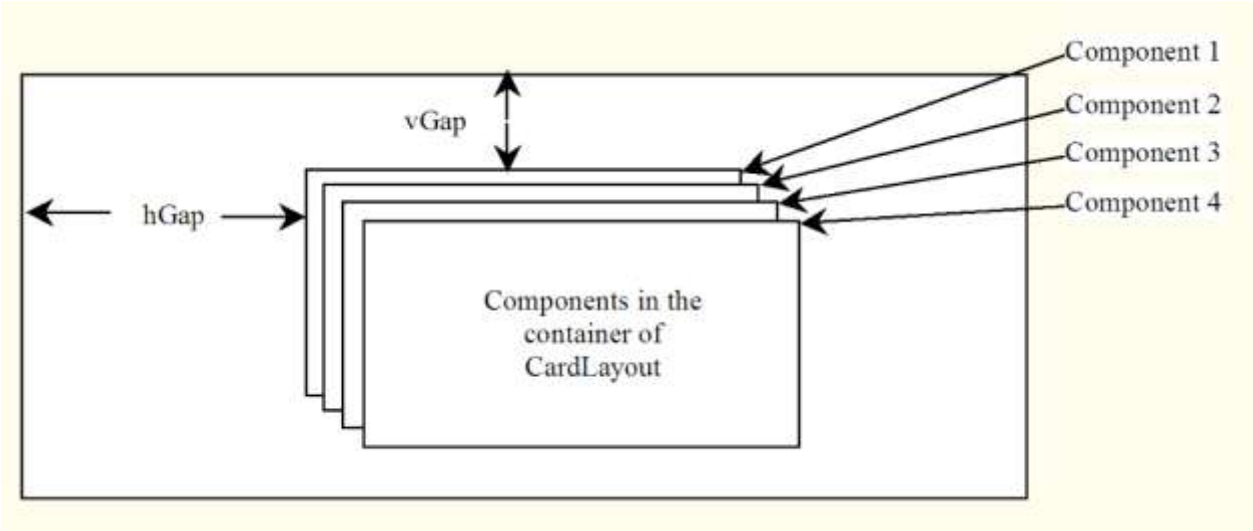
The Java GridLayout class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

It is the more sophisticated of all layouts. It aligns components by placing them within a grid of cells, allowing components to span more than one cell.

CardLayout

CardLayout places components in the container as cards. Only one card is visible at a time, and the container acts as a stack of cards. The ordering of cards is determined by the container's own internal ordering of its component objects. CardLayout defines a set of methods that allow an application to flip through the cards sequentially or to show a

specified card directly.



Sl. No.	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**.
6)	Java AWT is slower than swing in terms of performance.	Java Swing is faster than the AWT
7)	Java AWT needs a higher amount of memory for the execution.	Java Swing needs less memory space as compared to Java AWT.