Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

UNIT NO 5

EVENT DRIVEN PROGRAMMING

5.6 Introduction to Swing

Layout Management

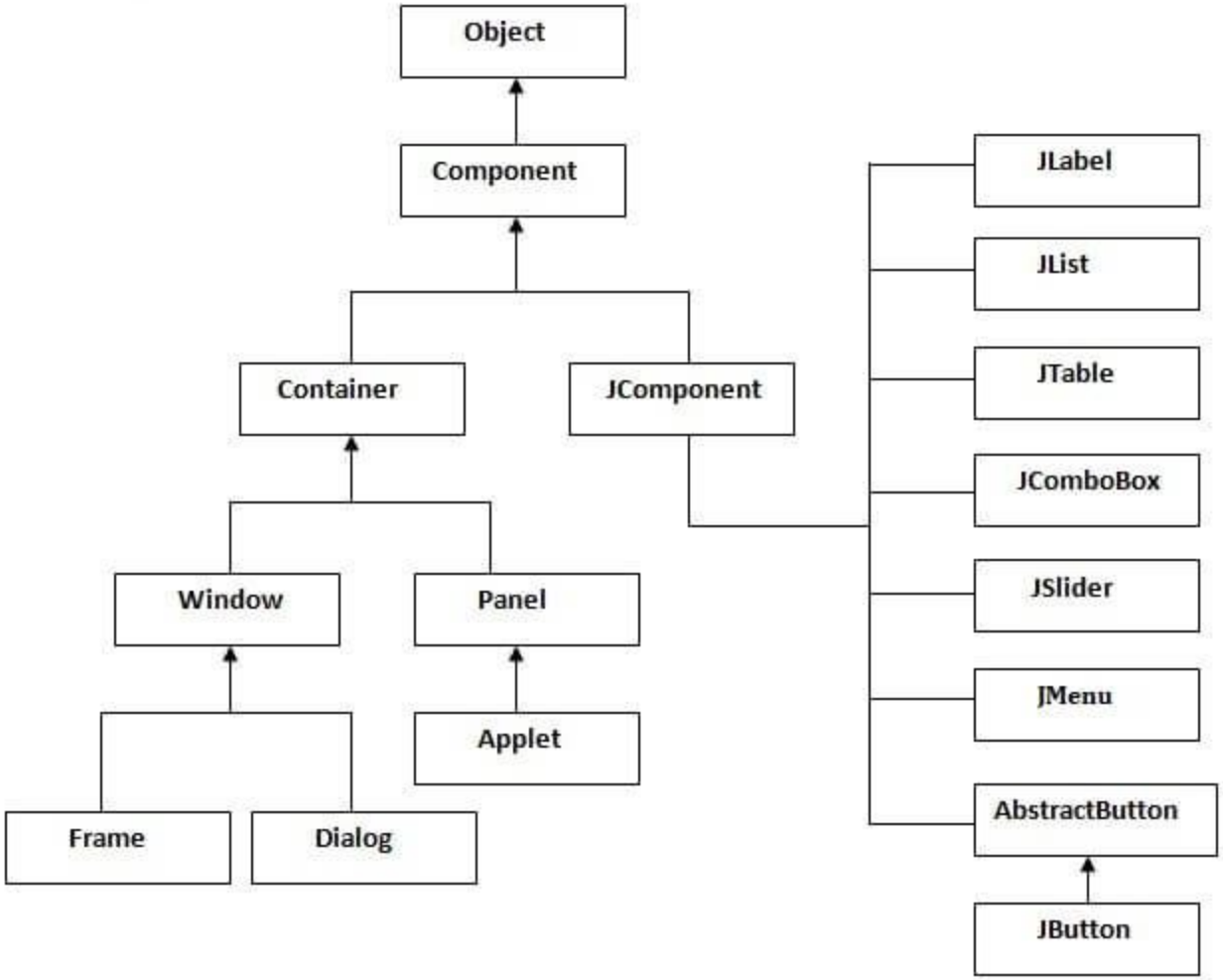
COMPUTER SCIENCE & ENGINEERING

Swing - Introduction

- Java Swing is a part of Java Foundation Classes (JFC) which was designed for enabling large-scale enterprise development of Java applications.
- The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.
- Java Swing is a set of APIs that provides graphical user interface (GUI) for Java programs. Java Swing is also known as Java GUI widget toolkit.
- Java Swing or Swing was developed based on earlier APIs called Abstract Windows Toolkit (AWT).
- Swing provides richer and more sophisticated GUI components than AWT. The GUI components are ranging from a simple label to complex tree and table

Difference between AWT and Swing

Swing Hierarchy



Swing Example

```
import javax.swing.*;

public class FirstSwingExample   {
        public static void main(String[] args)   {
                JFrame f=new JFrame();//creating instance of JFrame

                JButton b=new JButton("click");//creating instance of JButton
                b.setBounds(130,100,100, 40);//x axis, y axis, width, height

                f.add(b);//adding button in JFrame

                f.setSize(400,500);//400 width and 500 height
                f.setLayout(null);//using no layout managers
                f.setVisible(true);//making the frame visible
        }
}
```

Layout Management

- Layout refers to the arrangement of components within the container. In other way, placing the components at a particular position within the container. The task of layouting the controls is done automatically by the Layout Manager.

Layout Manager

- The layout manager automatically positions all the components within the container
- If layout manager is not specified then the components are positioned by the default layout manager.
- Java provides various layout manager to position the controls. The properties like size,shape and arrangement varies from one layout manager to other layout manager.
- The layout managers adapt to the dimensions of appletviewer or the application window.
- The layout manager is associated with every Container object. A layout manager is an instance of any class that implements the LayoutManager interface.
- The layout manager is set by the setLayout( ) method.
- The setLayout( ) method has the following general form:

  void setLayout(LayoutManager layoutObj)

  layoutObj is a reference to the desired layout manager.
- Each layout manager keeps track of a list of components that are stored by their names.
- The layout manager is notified each time you add a component to a container
- Java has several predefined LayoutManager classes

Layout Manager (Contd..)

- Java has several predefined Layout Manager classes
  - Flow Layout
  - Border Layout
  - Grid Layout
  - Card Layout
  - GridBag Layout

Layout Manager - Flow Layout

- **FlowLayout** implements a simple layout style, The direction of the layout is governed by the container's component orientation property, which, by default, is left to right, top to bottom.
- Therefore, by default, components are laid out line-by-line beginning at the upper-left corner. In all cases, when a line is filled, layout advances to the next line.
- A small space is left between each component, above and below, as well as left and right.
- The constructors for **FlowLayout** are

FlowLayout( )	- default Layout
FlowLayout(int *how*)	- specify how each line is aligned
FlowLayout(int *how*, int *horz*, int *vert*)	- specifies the horizontal and vertical space

Valid values for how are as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

FlowLayout.LEADING

FlowLayout.TRAILING

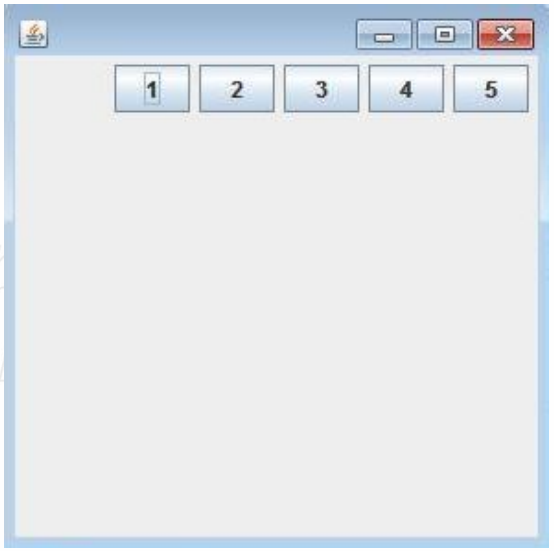
Layout Manager - Flow Layout - Example

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



Layout Manager - Border Layout

- The BorderLayout class implements a common layout style for top-level windows.
- It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center.
- The constructors of **BorderLayout**:

  BorderLayout( ) - creates a default border layout

  BorderLayout(int *horz*, int *vert*) - specifies the horizontal and vertical space left between components

- **BorderLayout** defines the following constants that specify the regions:

  BorderLayout.CENTER
  BorderLayout.SOUTH
  BorderLayout.EAST
  BorderLayout.WEST
  BorderLayout.NORTH

Layout Manager - Border Layout - Example

```
import java.awt.*;
import javax.swing.*;
public class MyBorderLayout {
   JFrame f;

   MyBorderLayout() {
          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);

         //setting flow layout of right alignment
         f.setSize(300,300); f.setVisible(true);
}
public static void main(String[] args) {
          new MyBorderLayout();
}
}
```

Layout Manager - Grid Layout

- GridLayout lays out components in a two-dimensional grid
- The constructors are:

GridLayout( ) - creates a single-column grid layout

GridLayout(int numRows, int numColumns) - creates a grid layout with the specifed number of rows and columns

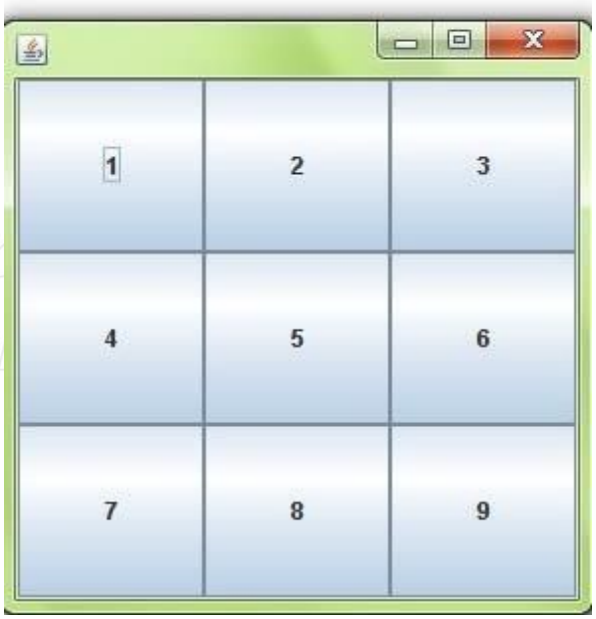
GridLayout(int numRows, int numColumns, int horz, int vert) - specify the horizontal and vertical space left between components in horz and vert

- Specifying *numRows* as zero allows for unlimitedlength columns. Specifying *numColumns* as zero allows for unlimited-length rows.

Layout Manager - Grid Layout - Example

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



Layout Manager - GridBag Layout

- GridBagLayout is one of the most flexible — and complex — layout managers the Java platform provides.
- A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns.
- The general procedure for using a grid bag is to first create a new GridBagLayout object and to make it the current layout manager. Then, set the constraints that apply to each component that will be added to the grid bag. Finally, add the components to the layout manager.
- GridBagLayout defines only one constructor:

GridBagLayout( )

Layout Manager - Card Layout

- The **CardLayout** class is unique among the other layout managers in that it stores several different layouts.
- Each layout can be thought of as being on a separate index card in a deck that can be shuffled so that any card is on top at a given time.
- **CardLayout** provides these two constructors:

CardLayout( ) - creates a default card layout

CardLayout(int *horz*, int *vert*) - specify the horizontal and vertical space left between components in *horz* and *vert*

- The cards are typically held in an object of type **Panel**. This panel must have **CardLayout** selected as its layout manager.
- The cards that form the deck are also typically objects of type **Panel**. then add these panels to the panel for which CardLayout is the layout manager.
- Finally, add this panel to the window.