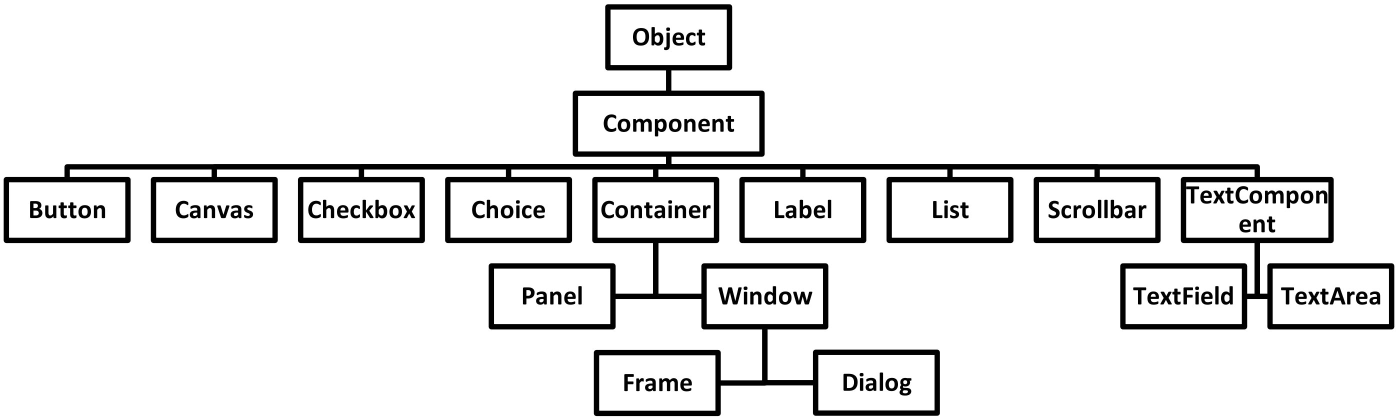
Unit 5

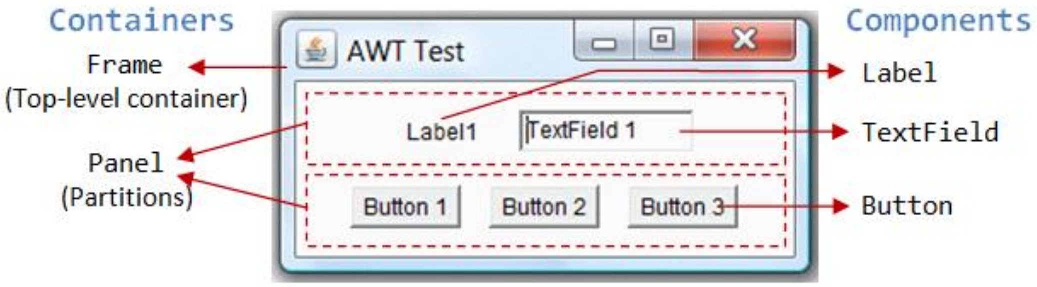
1. Graphics Programming *(Part 1/3)*

The java.awt.Graphics Class: Graphics Context and Custom Painting. A graphics context provides the capabilities of drawing on the screen. The graphics context maintains states such as the color and font used in drawing, as well as interacting with the underlying operating system to perform the drawing.

Definition: The AWT (Abstract Window Toolkit) contains large no.of classes which help to include various graphical components in java programming. The graphical components include text box, buttons, labels, radio buttons, list items and etc., **java.awt package** should be imported.

AWT hierarchy





9 classes – label, etc.,
Component properties:

- Size – width, height
- Location – point of class X and Y
- Bounds – x, y, width, height
- Foreground and Background
- Font
- Color class – RGB
- Cursor – the shape of mouse pointer

Container
The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as 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.

Window class – Frame (Decorated windows)
The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

Canvas
It encapsulates a blank window upon which one can draw. It overrides paint() method.

2. Frame

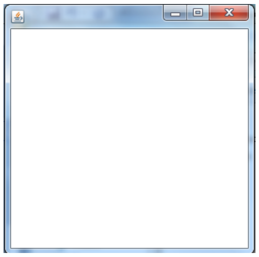
- Frame is a standard graphical window.
- It can be displayed using the **Frame class.**
- It has standard minimize, maximize, and close buttons.
  Syntax, i. Frame() | ii. Frame(String title)

  Methods, void **setResizable**(boolean resizable) | void **setTitle**(String Title) | **setSize**(int width, int height) | **getTitle**() | **setVisible**(boolean visible)
- Frame can be created in two ways,
  - By extending frame class
  - By creating instance of frame class

Example for extending frame class

```
import java.awt.*;
class FrameDemo extends Frame
{
        public static void main(String[] args)
        {
                FrameDemo fr = new FrameDemo();
                fr.setSize(300,300);
                fr.setVisible(true);
        }
}
```

Output:



Example for instance of frame class

```
import java.awt.*;
class FrameDemo
{
        public static void main(String[] args)
        {
                Frame fr = new Frame();
                fr.setSize(300,300);
                fr.setVisible(true);
        }
}
```

3. Program using AWT Components

```
import java.awt.*;
class UsingAwtComponents
{
public static void main(String[] args)
{
Frame fr = new Frame("Displaying Components");
fr.setSize(500,600); //Method 1------------------------------
fr.setVisible(true); //Method 2-------------------------------

//label
Label L1 = new Label("OK");
fr.add(L1);

//Buttons - Puch Buttons
Button B1 = new Button("Submit");
fr.setBounds(30,100,80,30); fr.add(B1); //Method 3 & 4----------------------

//canvas - special area created on frame - used for drawing oval, rect, etc
Canvas C1 = new Canvas();
C1.setSize(100,100);
C1.setBackground(Color.blue); //Method 5----------------------------
fr.setLayout(new FlowLayout());//Method 6---------------------------
fr.add(C1);

//Scrollbars - slider widgets - Horizontal & vertical
Scrollbar HSelector = new Scrollbar(Scrollbar.HORIZONTAL);
Scrollbar VSelector = new Scrollbar(Scrollbar.VERTICAL);
fr.setSize(100,100);
fr.add(HSelector);
fr.add(VSelector);

//TextField
fr.setLayout(new FlowLayout());
Label L2 = new Label("Enter ur name");
TextField input1 = new TextField(20);
fr.add(L2);
fr.add(input1);

//TextArea
fr.setLayout(new FlowLayout());
Label L3 = new Label("Enter ur name");
TextArea input2 = new TextArea(10,20);
fr.add(L3);
fr.add(input2);

//checkbox
Checkbox cb1 = new Checkbox("Candy");
Checkbox cb2 = new Checkbox("Juice");
Checkbox cb3 = new Checkbox("Coffee");
fr.add(cb1);
fr.add(cb2);
fr.add(cb3);

//CheckboxGroup - Radio button
CheckboxGroup cbg =new CheckboxGroup();
Checkbox cb4 = new Checkbox("Candy",cbg,true);
Checkbox cb5 = new Checkbox("Juice",cbg,false);
Checkbox cb6 = new Checkbox("Coffee",cbg,false);
fr.add(cb4);
fr.add(cb5);
fr.add(cb6);

//choice
Choice c1 = new Choice();
c1.add("Mango");
c1.add("Apple");
c1.add("Banana");
fr.add(c1);

//List panel
List Ls1 = new List();
Ls1.add("Rose");
Ls1.add("Lily");
Ls1.add("Jasmine");
fr.add(Ls1);
}}
```

Compile and run

4. Life cycle of applet

Java Applet

- Applets are the small java programs that can be used in **internetworking** environment.
- Applet is a special type of program that is embedded in the webpage to generate the **dynamic display** content. It runs inside the browser and works at client side.
- Applet does not require **main** function.
- Applications, arithmetic operations, graphics, playing sounds, animation, etc.

Advantage of Applet

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

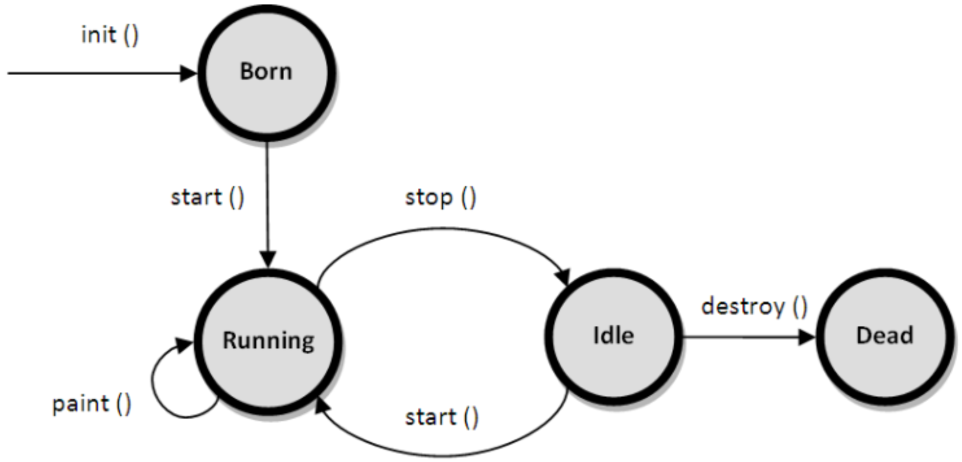
◦It works at client side so less response time.

◦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.



Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.

java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

1.public void **init**(): is used to initialized the Applet. It is invoked only once.

2.public void **start**(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

3.public void **stop**(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

4.public void **destroy**(): is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1.public void **paint**(Graphics g): is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

Executing applet

Two ways

1. Using web browser
2. Using Appletviewer

Using web browser

1. Compile: javac filename.java
2. Type the code in notepad and save as demo.html

```
<applet code "filename.class" width=300 height=100>
</applet>
```

3. Then open the file demo.html in a browser to view output.

Using Appletviewer

```
import java.awt.*;
import java.applet.*;
/*<applet code "lineDem.class" width=300 height=100>
</applet>
*/
public class lineDem extends Component
{
        public void paint(Graphics g)
        {
        g.drawLine(10,10,50,50);
        g.drawLine(10,50,50,10);
        }
}
```

Compile: javac lineDem.java
Run: Appletviewer lineDem.java

5. Working with 2D shapes(Lines, Rectangle, Ellipses, Circles, Arcs)

Drawing Lines:

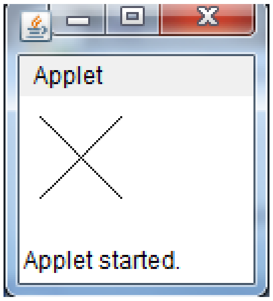
Using drawLine() method lines can be drawn.

Syntax: void **drawLine**(int startX, int startY, int endX, int endY)

```
Import java.awt.*;
Import java.applet.*;
Public class lineDemo extends Applet
{
        Public void paint(Graphics g)
        {
        g.drawLine(10,10,50,50);
        g.drawLine(10,50,50,10);
        }
}
```

Myapplet.html

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



Applet started.

Compile: javac lineDemo.java

Run: Appletviewer Myapplet.html

Similarly, **to draw rectangle,**

Syntax: void **drawRect**(int top, int left, int width, int height)

void **fillRect**(int top, int left, int width, int height)

```
Public void paint(Graphics g)
        {
        g.drawRect(50,60,50,40);
        g.drawRect(80,50,50,60);
        }
```

To draw ellipses and circles,

Syntax: void **drawOval**(int top, int left, int width, int height)

void **fillOval**(int top, int left, int width, int height)

```
Public void paint(Graphics g)
        {
        g.drawOval(20,30,40,40);
        g.drawOval(120,30,60,40);
        }
```

To draw Arcs,

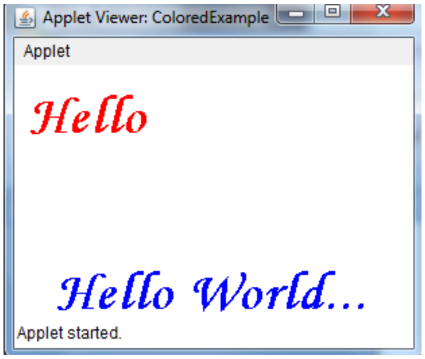
Syntax: void **drawArc**(int top, int left, int width, int height, int startAngel, int sweepAngle)

void **fillArc**(int top, int left, int width, int height, int startAngel, int sweepAngle)

```
Public void paint(Graphics g)
        {
        g.drawArc(30,40,60,50,0,75);
        g.drawArc(30,100,60,50,75,180);
        }
```

6. Working with color, font and images

```
import java.applet.*;
import java.awt.*;
//import java.awt.Color;
 /*
<applet code = "ColoredExample" width = 300 height = 300>
</applet>
*/
 public class ColoredExample extends Applet{
String msg="";
public void init()
{
Font f;
f= new Font("Monotype Corsiva", Font.BOLD,40);
msg="Hello";
setFont(f);
}
public void paint(Graphics g){
g.setColor(Color.blue); //set color to blue
g.drawString("Hello World...",30,180); //print hello world

g.setColor(Color.red);
g.drawString(msg, 10,50);
}}
```



```
Hello

Hello World...
```

Applet started.

Java provides support for two common image formats: **GIF** and **JPEG**. The basic class for representing an image is java.awt.Image. Packages that are relevant to image handling are **java.applet, java.awt and java.awt.image.**

```
 import java.awt.*;

public class img extends Frame
{
public img()
{          setTitle(“Image on Frame”);
           setSize(250, 250);
           setVisible(true);
}
public void paint(Graphics g)
{          Toolkit tk = Toolkit.getDefaultToolkit();
           Image img = tk.getImage(“abc.jpeg”);
           g.drawImage(img,100,100,this);
}
public static void main(String args[]){
new img();//anonymous obj to access constructor
}}
```

```
import java.awt.*;
import java.applet.*;
/*<applet code “lineDem.class” width=300
height=100>
</applet>
*/
public class lineDem extends Applet
{
Image img;
public void init()
{
img=getImage(getDocumentBase(),”abc.jpg”);
 }
public void paint(Graphics g)
{
g.drawImage(img,50,50,this);
}
}
```

Task:

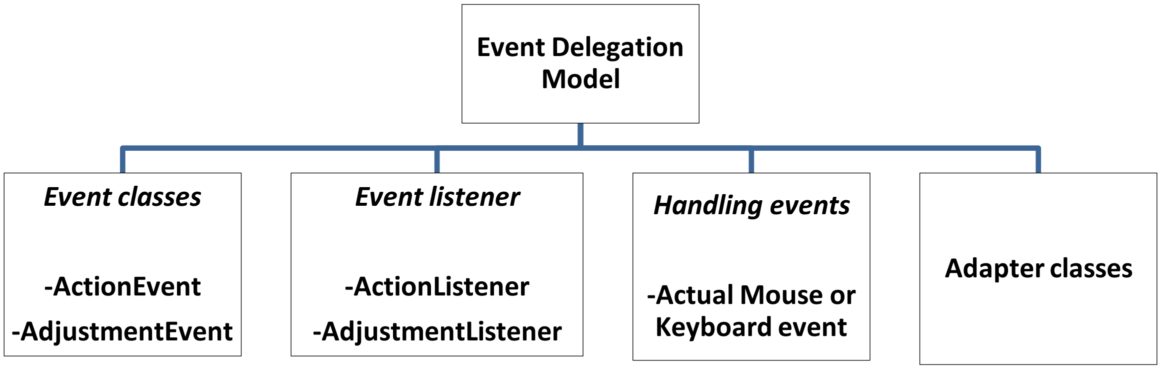
Write an applet program to draw flower with color packages.

7. Event Handling ***(Part 2/3)***

- Event means any activity that interrupts the current ongoing activity.
- An event in Java is an object that is created when something changes within a graphical user interface.
- Ex: a mouse click or key press from a keyboard.
- **Event-handling code** is responsible for all the activity which happen between user and application
- AWT conveys these actions (events) to the programs.

7.1 Event Delegation Model

It is used for understanding the event and for processing it.



7.2Event handlers

7.2.1 Event Classes and Event Listeners

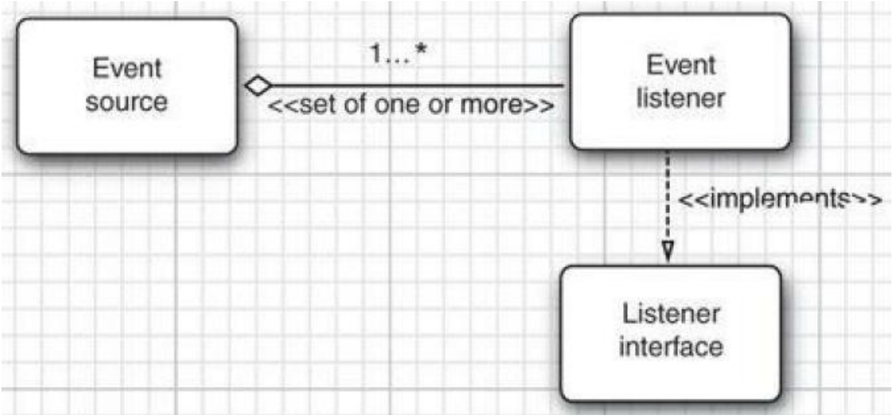
- Event classes are the classes responsible for handling events in the event handling mechanism.

- EventObject class belongs to java.util other events are java.util.event package.

S.No	Event Classes	*Object*	Event Listeners	*Interface*
1.	ActionEvent	*Object generated when button pressed* *Method: String getActionCommand()*	ActionListener	*Void actionPerformed(actionEvent ae)*
2.	AdjustmentEvent	*Object generated when scrollbars used* *Method: Adjustable getAdjustable()*	AdjustmentListener	*Void adjustmentValueChanged(actionEvent ae)*
3.	TextEvent	*Occurs when text field changed.*	TextListener	*Void textChnged(TextEvent tx)*

4.	ContainerEvent	*Occurs when component add or removed* *Method: Container getContainer()*	ContainerListener	*Void componentdded(ContinerEvent ct)* *Void componentRemoved(ContinerEvent ct)*
5.	ComponentEvent	*Occurs when resized, moved, hidden or visible* *Method: Component getComponent()*	ComponentListener	*Void componentShown(ContinerEvent ct)* *Hidden* *Moved* *Resized*
6.	ItemEvent	*Occurs when item selected.*	ItemListener	*Void itemStateChanged(ItemEvent it)*
7.	FocusEvent	*Occurs during keybord focus.* *Method:Component getOppositeComponent()*	FocusListener	*Void focusGained(FocusEvent fo)* *Lost*
8.	KeyEvent	*Occurs when key pressed or released.* *Method: Char getKeyChar()*	KeyListener	*Void keyPressed(keyEvent k)* *Released* *Typed*
9.	WindowEvent	*Generated when window ctivted, maximized, minimized*	WindowListener	*Void windowOpened(WindowEvent w)* *Closed* *Closing* *Activated* *Deactivated* *Iconified* *DeIconified*
10.	MouseEvent	*Generated when mouse clicked, moved, dragged, released.*	MouseListener	*Void mouseClicked(MouseEvent m)* *Pressed* *Released* *Entered* *Exited*

Relationship between Event Sources and Event Listener



An event source is an object that can register listener objects. The listener object is an instance of a class that can implement a special interface called listener interface.

Step1: The event source sends the event objects to all the registered listeners when an event occurs.

Step2: The listener objects will use the information in the event object and then determine reaction.

8. Adapter classes

It is basically a class in java that implements an interface with set of dummy methods. Adapter classes are,

- WindowAdapter
- ComponentAdapter
- ContainerAdapter
- FocusAdapter
- KeyAdapter
- MouseAdapter
- MouseMotionAdapter

Disdvantage: when a class implements such interface all 7 methods should be implemented. WindowAdapter class implements WindowListener interface and make all seven empty implementations.

But when WindowAdapter class is subclassed, we can use any method without restrictions.

Example:

```
public interface **WindowListener**
{
void windowOpened(WindowEvent e);
void windowClosing(WindowEvent e);
void windowClosed(WindowEvent e);
void windowIconified(WindowEvent e);
void windowDeiconified(WindowEvent e);
void windowActivated(WindowEvent e);
void windowDeactivated(WindowEvent e);
}
```

```
class Terminator implements WindowListener
{
public void windowClosing(WindowEvent e)
{
if (user agrees)
System.exit(0);
}
public void windowOpened(WindowEvent e) {}
public void windowClosed(WindowEvent e) {}
public void windowIconified(WindowEvent e) {}
public void windowDeiconified(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}
}
```

Task:

9. Actions

It is common to have multiple ways to activate the same command. The user can choose a certain function through a menu, a keystroke, or a button on a toolbar.

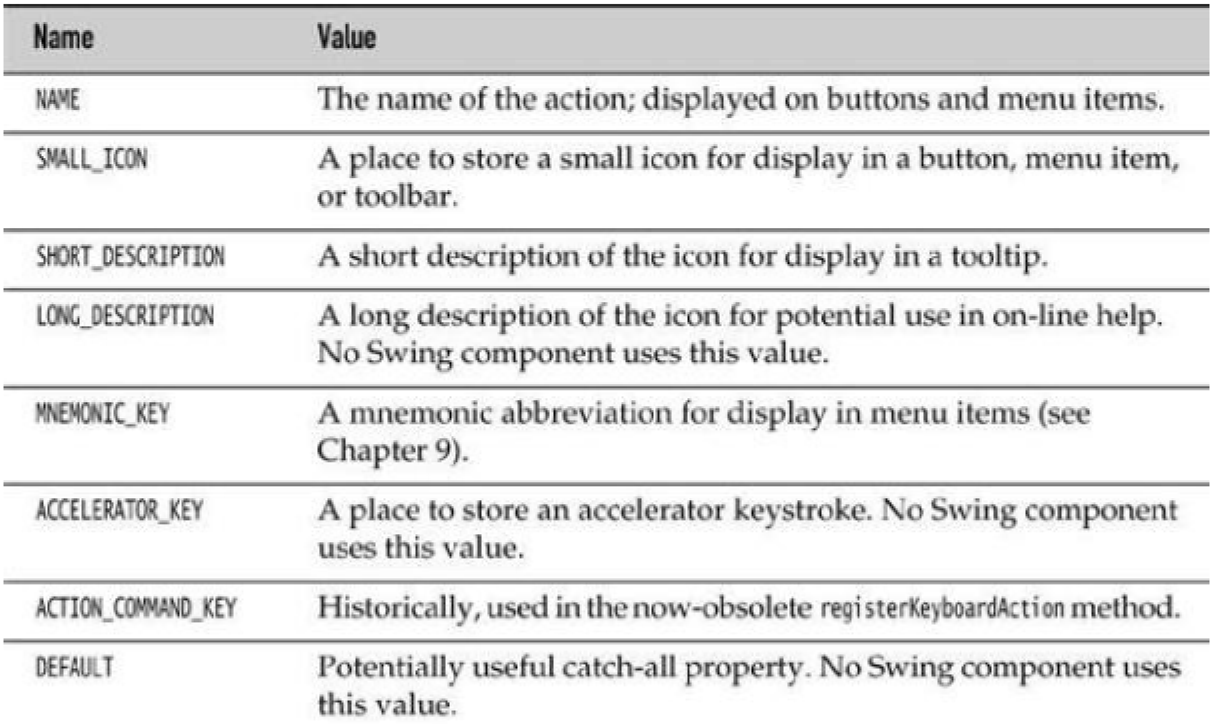
The Swing package provides a very useful mechanism to encapsulate commands and to attach them to multiple event sources: the Action interface.
An *action* is an object that encapsulates

- A description of the command (as a text string and an optional icon); and
- Parameters that are necessary to carry out the command (such as the requested color in our example).

The Action interface has the following few methods:

```
void actionPerformed(ActionEvent event)
void setEnabled(boolean b)
boolean isEnabled()
```

Table 8.1. Predefined Action Table Names



Name	Value
NAME	The name of the action; displayed on buttons and menu items.
SMALL_ICON	A place to store a small icon for display in a button, menu item, or toolbar.
SHORT_DESCRIPTION	A short description of the icon for display in a tooltip.
LONG_DESCRIPTION	A long description of the icon for potential use in on-line help. No Swing component uses this value.
MNEMONIC_KEY	A mnemonic abbreviation for display in menu items (see Chapter 9).
ACCELERATOR_KEY	A place to store an accelerator keystroke. No Swing component uses this value.
ACTION_COMMAND_KEY	Historically, used in the now-obsolete registerKeyboardAction method.
DEFAULT	Potentially useful catch-all property. No Swing component uses this value.

10. Mouse Events

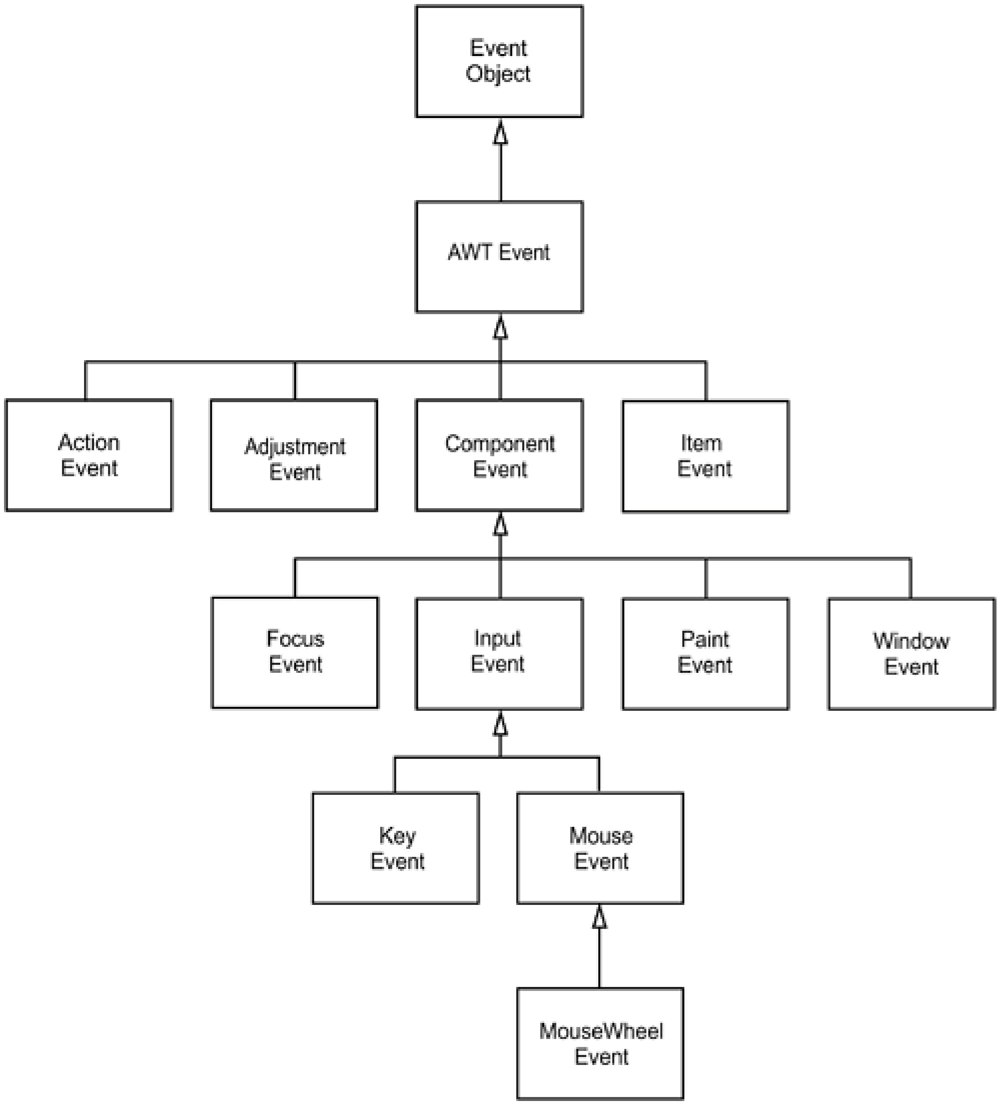
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

/*
<applet code ="MouseDemo" width=300 height=200>
</applet>
*/

public class MouseDemo extends Applet implements MouseListener, MouseMotionListener
{
String msg="";
int xposition=0, yposition=0;

public void init()
{
addMouseListener(this);
addMouseMotionListener(this);
}

public void mouseClicked(MouseEvent m)
{
xposition = m.getX();
yposition= m.getY();
repaint();
}
```

```
public void mousePressed(MouseEvent m)
{
xposition = m.getX();
yposition= m.getY();
msg = "Pressing mouse button";
repaint();
}

public void mouseReleased(MouseEvent m)
{
xposition = m.getX();
yposition= m.getY();
msg = "Releasing mouse button";
repaint();
}

public void mouseEntered(MouseEvent m)
{
xposition = 0;
yposition= 190;
msg = "Mouse  entered";
repaint();
}

public void mouseExited(MouseEvent m)
{
xposition = 0;
yposition= 190;
msg = "Mouse  exited";
repaint();
}

public void mouseDraggeded(MouseEvent m)
{
xposition = m.getX();
yposition= m.getY();
msg = "Dragging Mouse at "+xposition+","+yposition;
repaint();
}


public void mouseMoved(MouseEvent m)
{
xposition = m.getX();
yposition= m.getY();
msg = "Moving Mouse at "+xposition+","+yposition;
repaint();
}

public void paint(Graphics g)
```

```
{
g.drawString(msg, xposition, yposition);
}
}
```

11.AWT Event Hierarchy

The AWTEvent is a class derived from EventObject class. EventObject class is defined in java.util package.



12. Programming with Swing

Swing Fundamentals

JFC –Java Foundation Classes are graphical set of GUI components for java applications that streamline software and cloud application development.

JFC consists of,

AWT
Accessibility API
Java 2D API
Drag and Drop

AWT features	Swing features
Rich set of UI components	All the features of AWT
Robust event handling	100% java certified versions of existing AWT
Shape, color and font classes	Tree view, list box, tabbed panes
Layout managers	Pure java design
Data transfer classes – cut n paste	Pluggable look and feel

13.Layout Manager Classes

Layout Manager

The layout manager **automatically positions all the components** within the container.

If we do not use layout manager then also the components are positioned by the default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

- It is very tedious to handle a large number of controls within the container.
- Oftenly the width and height information of a component is not given when we need to arrange them.

Java provide us with various layout manager to position the controls. The properties like size, shape and arrangement varies from one layout manager to other layout manager. When the size of the applet or the application window changes the size, shape and arrangement of the components also changes in response i.e. the layout managers adapt to the dimensions of appletviewer or the application window.

The layout manager is associated with every Container object. Each layout manager is an object of the class that implements the LayoutManager interface.

AWT Layout Manager Classes:

Following is the list of commonly used controls while designed GUI using AWT.

BorderLayout -4

The borderlayout arranges the components to fit in the five regions: east, west, north, south and center.

CardLayout

The CardLayout object treats each component in the container as a card. Only one card is visible at a time.

FlowLayout(int alignment) – 4

The FlowLayout is the default layout.It layouts the components in a directional flow.

GridLayout(int n, int m)

The GridLayout manages the components in form of a rectangular grid.

GridBagLayout

This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically, horizontally or along their baseline without requiring the components of same size.

Example for border layout:

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

Swing container Hierarchy

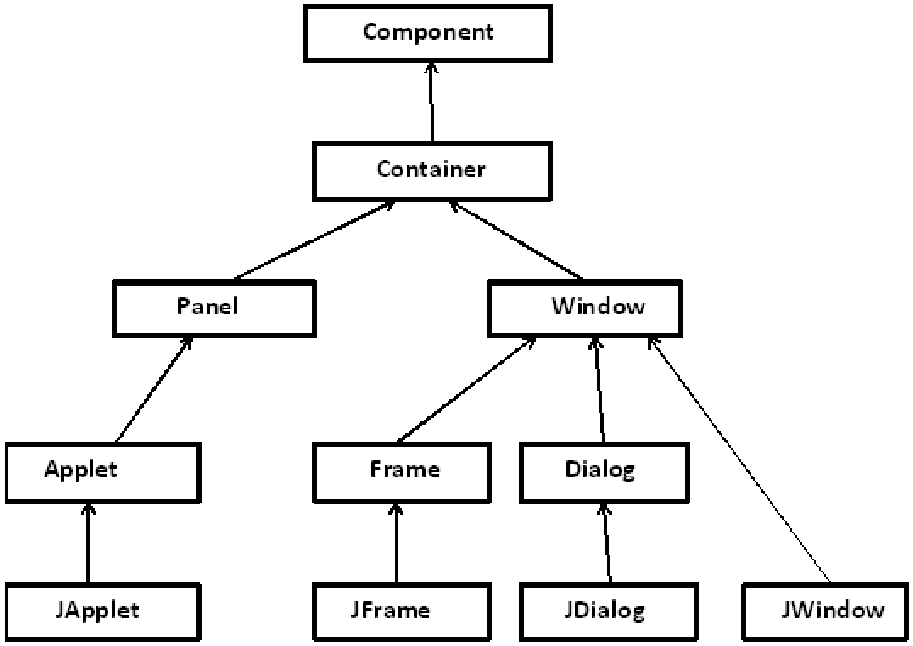


Figure 1.4 Top-Level Container Hierarchy

Swing is a set of classes that provides more powerful and flexible components that are possible with AWT.

Components include lists, buttons, panels, windows.

To use components, we need to place them in a container. A container is a component that holds and manages other components.

Container display components using layout manager.

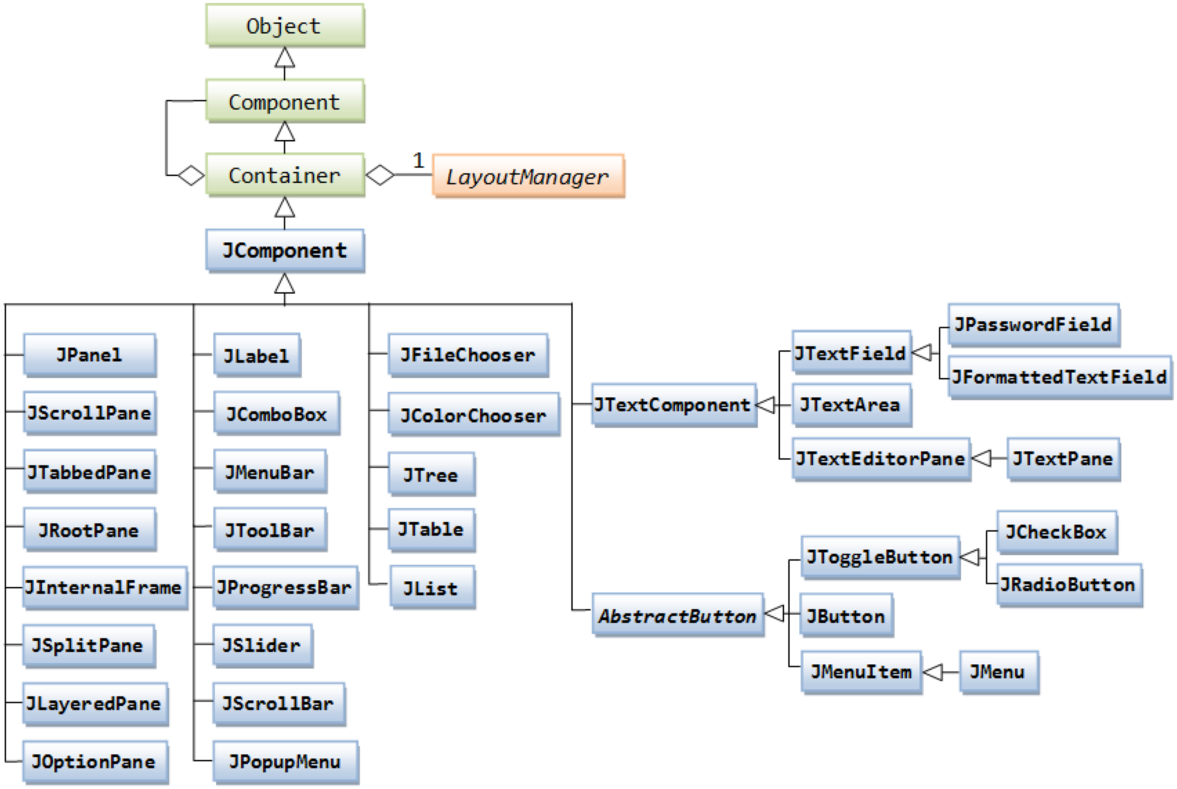
Swing component inherits JComponent class. JComponent inherits Container class in AWT.

Simple swing program

```
Public class abc extends JFrame
{
Public static void main(string args[])
{
New Hello();
}
Hello()
{
JLabel jl = new JLabel(“Hi”);
Add(jl);
This.setSize(100,100);
setVisible(true);
}}
```

14. Draw and explain java swing class hierarchy

Classes of javax.swing



15. Write a program using JButton and JCheckBox

JButton

```
import javax.swing.*;
public class ButtonExample {
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);
}
}
```

JCheckBox

```
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[])
   {
   new CheckBoxExample();
   }}
```

2 marks

1. What is AWT?
2. What is a Frame?
3. List methods used in Frame?
4. What is an applet?
5. **List advantages and drawbacks of applet.**
6. Write synatx to draw Line, Rect.
7. Write syntax to set color, font and image.
8. What is an Event?
9. **What is Event delegation Model.**
10. List few Event Listeners.
11. **What is an Adpater class?**
12. What is an action?
13. List few predefined actions.
14. List few Mouse Event methods.
15. What is an Event Object?
16. Wht is swing?
17. **D/B AWT and Swing.**
18. What is Layout manager. List types.
19. **What is JFC?**
20. List components in swing content hierarchy.

13 marks

1. Explain graphics programming with example
2. **How to work with 2D shapes in java. Example.**
3. Explain life cycle of applet. How applets are created and executed.
4. How to use color, font and image in applet.
5. Explain event handler types.
6. Explain adapter classes.
7. **Explain AWT Event Hierarchy with neat diagram.**
8. **Explain swing class component hierarchy**
9. **Explain Layout manager types. Explain Grid Layout manager.**
10. Write a program to create a phonebook look-up using swing.

Task:

1. **Write a program to create a frame with the following menus, such that the corresponding geometric object is created when a menu is clicked. i. Circle | ii. Rectangle | iii. Line**

2. **Write a program using java swing to insert a picture and scroll bars.**

3. **Write a program to create product enquiry form using frames.**