Chapter 6: Applet & Graphics Programming

An **applet** is a Java program that runs in a web browser. Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

- An applet is a Java class that extends the java.applet.Applet class.
- A main()method is not invoked on an applet & an applet class will not define main().
- Applets are designed to be embedded within an HTML page.
- A JVM is required to view an applet.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.

Advantage of Applet

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many platforms, like Linux, Windows etc

Drawback of Applet: Plug-in is required at client browser to execute applet.

Differences between an applet and a standalone application

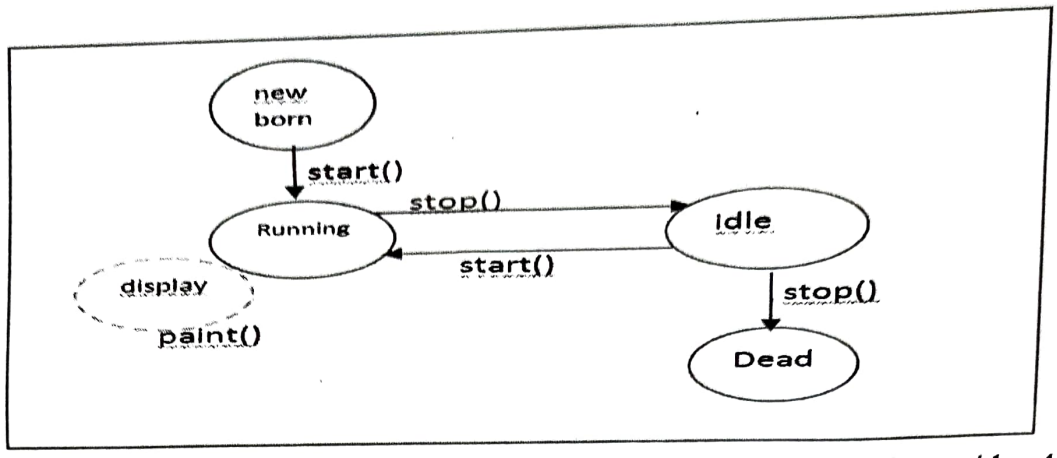
Application program	Applet program
Application program runs independently without using a web browser	applet program runs with the help of a browser
compiled and interpreted	only compiled
main method is required for execution.	Does not require a main method() for its execution
application programs larger in size	Small programs that are designed to be included in a HTML web document
Requires prior installation in the system	Prior installation is not needed
Not interactive	interactive
Applications are designed to work with client as well as server	Applications are designed for client side programming purpose
No security concerns are there.	Requires security for the system
Applications have full access to local file system and network.	Applets have no access for files on the local computer.

Life Cycle of an Applet:

An applet undergoes various states between its object creation and object removal known as applet life cycle. When an applet is executed in the web browser, it goes through 5 stages of its life cycle:. They are represented in 5 different states

1. New born state
2. Running state
3. Idle state/stopped state
4. Display state
5. Destroyed/dead state

Applet life cycle defined as how the object created, started, stopped and destroyed during the entire execution of the application. Applet life cycle has 5 methods init(), start(), stop(), paint() and destroy().These methods are invoked by the browser to execute. Applet works on the client side and has less processing time.



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

1. **Init:** is used to initialized the Applet. It is invoked only once.
   Syntax: **public void init():**
2. **start** : is invoked after the init() method. It is used to start the Applet.
   Syntax: **public void start()**
3. **Stop:** is used to stop the Applet. It is invoked when Applet is stopped.
   Syntax: **public void stop()**
4. **Destroy:** is used to destroy the Applet. It is invoked only once.
   Syntax: **public void destroy()**
5. **paint**: The method paint() is automatically called whenever there is a need to display an applet window. Whenever the applet needs to be refreshed, the paint method is called each time to redraw the applet.
   Syntax: public void pait(Graphics g)

Initialization state: Applet enters the born state when it is 1st loaded. This is achieved by init() method of applet class. Initialization occurs only once in the life cycle of the applet . The initializations such as variable initialization and objects like image & sound are loaded in init() method. init() method acts as a constructor.

syntax:
```
public void init()
{
  ----------(action)
}
```

Running state: Applet enters running state when the system calls start() method .It also occurs if the applet is already in idle state. start() method may be called more than once.

Syntax:
```
public void start()
{
-------(action)
}
```

Idle state/stopped state: An applet becomes idle when it is stopped from running. This occurs automatically when we leave the page containing the currently running applet or by explicitly calling stop() method.

Syntax:
```
public void stop()
{
----------(action)
}
```

Dead state: An applet is said to be dead when it is removed from the memory. This occurs automatically when we quit the browser where destroy () method is called. Similar to initialization, destroying occurs only once in the life cycle of an applet.

syntax:
```
public void destroy()
{
----------(display statements)
}
```

Display state: Applet moves to display state to perform o/p operation on the screen. This is done when the applet enters the running state. paint() method is called to display contents. paint() method is defined in applet class which is inherited from the super class Component

Syntax: public void paint(Graphics g)
{
----------(display statements)
}

Building Applet Code: Java applets are like Java applications; their creation follows the same 3-step process of write, compile and run. The difference is, instead of running on the desktop, they run as part of a web page. The creation and running a simple Java applet is achieved by following these basic steps:

1. Write the Java code in a text file
2. Save the file
3. Compile the code
4. Fix any errors
5. Reference the applet in a HTML page
6. Run the applet by viewing the web page

To build the applet code 2 classes of java library are essential namely Applet and Graphics. The Applet class is contained in java.applet package provides life and behaviour to the applet through its methods such as int(), start() and paint(). The Applet class therefore maintains the life cycle of an applet. To display the result of the applet code, the paint() method of the Applet class is called up. The output may be test, graphics or sound. The syntax of paint() method which requires a Graphic object as an argument, is defined as follows

public void paint(Graphics g)

The applet code imports the **java.awt package** that contains the Graphic class. All output operations of an applet are performed using the methods defined in the graphics class.

The general format of applet code is as following:

```
import java.awt.*;
import java.applet.*;
.....
.....
public class applet classname extends Applet
{
.....
..... //statements
public void  paint(Graphics g)
{
.....
.....//Applet operations code
}
```

e.g.:

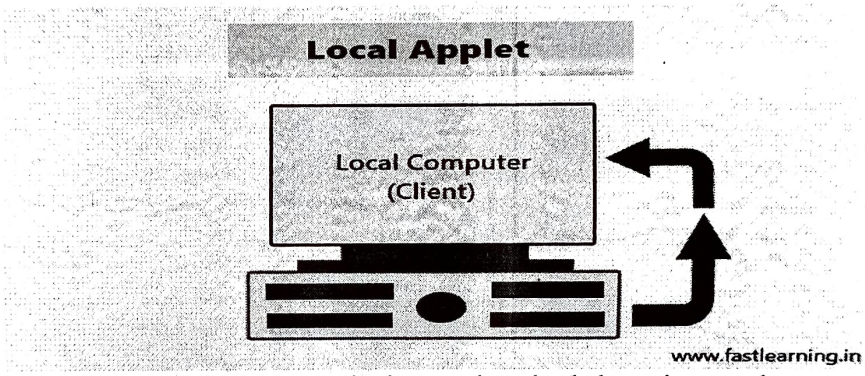
```
//HelloApplet.java
import java.applet.Applet;
import java.awt.*;
 public class HelloApplet extends Applet
 {
  public void paint(Graphics g)
  {
   g.drawString ("Welcome to Applet Tutorial !",100, 100);
  }
 }
```

Embedding Applet in HTML (Web Page)

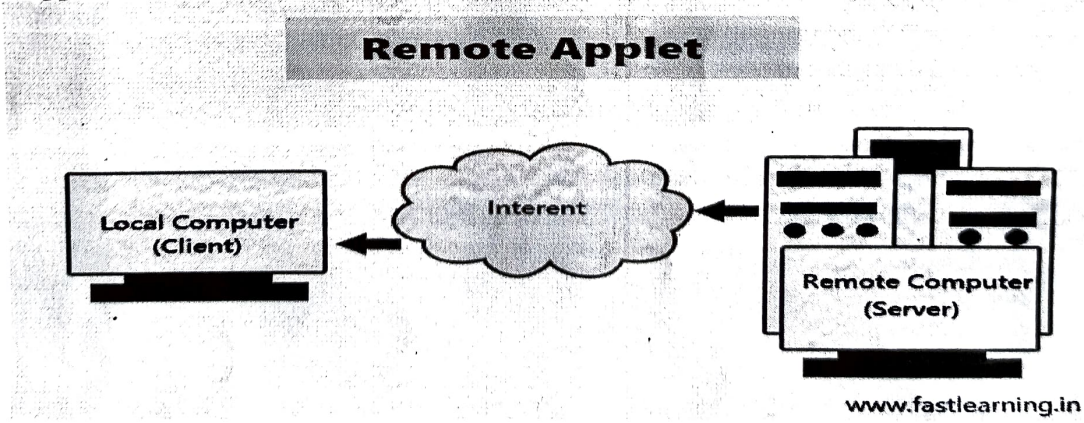
```
<HTML>
<HEAD>
<TITLE>
Hello World Applet
</TITLE>
</HEAD>
<body>
<h1>Hi, This is My First Java Applet on the Web!</h1>
<APPLET CODE="HelloApplet.class" width=500 height=400>
</APPLET>
</body>
</HTML>
```

Types of Applets: Web pages can contain 2 types of applets which are named after the location at which they are stored.

1. **Local Applets**: A local applet is one that is stored on our own computer system.
2. **Remote Applets**: A remote applet is located on a remote computer system.



We can create our own applet by own design and embed them into web pages. Local applets are developed in a single system and it is stored in a local system. The web page will search the local system directories, find the local applet and execute it. Execution of local applets doesn't require any internet connection.



Remote Applets: A remote applet is developed by someone else and stored on a remote computer connected to the Internet. A remote applet is that which is developed by someone else and stored on a remote computer connected to the internet.

The only difference between Local Applet and Remote Applet is the value of the codebase attribute. In the first case, codebase specifies a local folder, and in the second case, it specifies the URL at which the applet is located.

Graphics Programming: Graphics is one of the most important features of Java. Java applets can be written to draw lines, arcs, figures, images and text in different fonts and styles. Different colors can also be incorporated in display.

The Graphics Class: The graphics class defines a number of drawing functions; each shape can be drawn (edgeonly) or filled. To draw a shape, we need to use the appropriate method with the required arguments.

Drawing Methods of the Graphics Class

Sr.No	Method	Description
1.	clearRect()	Erase a rectangular area of the canvas.
2.	copyArea()	Copies a rectangular area of the canvas to another area
3.	drawArc()	Draws a hollow arc
4.	drawLine()	Draws a straight line
5	drawOval()	Draws a hollow oval
6	drawPolygon()	Draws a hollow polygon
7	drawRect()	Draws a hollow rectangle
8.	drawRoundRect()	Draws a hollow rectangle with rounded corners
9.	drawString()	Display a text string
10.	FillArc()	Draws a filled arc
11.	fillOval()	Draws a filled Oval
12.	fillPolygon()	Draws a filled Polygon
13.	fillRect()	Draws a filled rectangle
14.	fillRoundRect()	Draws a filled rectangle with rounded corners
15.	getColor()	Retrieves the current drawing color
16.	getFont()	Retrieves the currently used font
17.	getFontMetrics()	Retrieves the information about the current font
18.	setColor()	Sets the drawing color
19.	setFont()	Sets the font

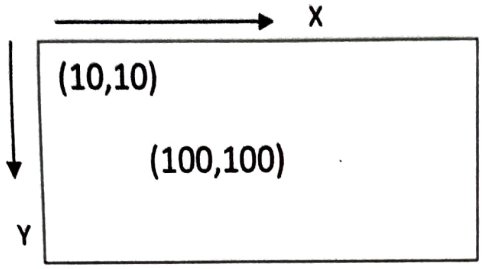
Graphics class with example.

Method	Description
drawLine(int x1,int y1,int x2,int y2) e.g. : drawLine(35,45,75,95);	Used to draw a straight line from point(x1,y1) to (x2,y2)
dawRect(int x,int y,int width,int length) ex:drawRect(35,45,25,35);	Used to draw a rectangle with the upper left corner at (x,y) and with the specified width and length
drawRoundRect(int x,int y,int width,int length,int arcWidth,int archeight) e.g. drawRoundRect(35,4,5,25,35,10,10);	Used to draw a rounded edged rectangle. The amount of rounding is controlled by arcWidth and archeight.
drawOval(int x,int y,int width,int length) e.g. : drawOval(25,45,25,35);	Used to draw an oval inside an imaginary rectangle whose upper left corner is at(x,y).to draw a circle, keep the width and length the same.
drawArc(int x,int y,int width,int length,int startAngle,int arcAngle) e.g. :drawArc(35,45,75,95,0,90);	Used to draw an arc inside an imaginary rectangle whose upper left corner is at (x,y).The arc is drawn from the startAngle to startAngle + arcAngle and is measured in degrees.
drawPolygon(int x[],int y[],int n) e.g. : drawPolygon(x,y,6); int []x={20,35,50,65,80,95}; int []y={60,105,105,110,95,95};	Used to draw a polygon by n line segments.the command will close the polygon.(x-coordinates go in one array with accompanying y-coordinates in the other.)
drawString(String str,int x,int y) e.g. :drawString("Java is col",40,70);	Draws a string starting at the point (x,y).Be sure you leave enough room from the top of the screen for the size of the font.

The graphics class with in the abstract windowing tool kit (AWT) makes it possible for a java programmer to draw simple geometric shapes, print text and position images within the borders of a component, such as an applet, frame, panel or canvas. The graphics class used draw lines, shapes, characters and images to the screen inside an applet.

The graphics co-ordinate system

To draw an object on the screen, you call one of the drawing methods available in the graphics class. All the drawing methods have arguments representing end points, sorting locations of the objects as values in the applet's co-ordinate system.



Drawing Lines: To draw a line, you need to use the drawLine method of the Graphics class. This method takes four arguments (x1, y1)co-ordinates of the starting point and (x2, y2) co-ordinates of the ending point. If (x1, y1) and (x2, y2) are the same point, it will draw a point.

voiddrawLine (int x1, int y1, int x2, int y1);

```
public void paint(Graphics g)
{
g.drawLine(10, 10, 40, 10);
}
```

Since the y-coordinates are the same, this will be a horizontal line.

Drawing Rectangles:

The drawRect() and fillRect() methods display an outlined and filled rectangle, respectively.

Syntax

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

The upper-left corner of the rectangle is at(top,left). The dimensions of the rectangle are specified by width and height. The methods drawRoundRect() or fillRoundRect() are used to draw a rounded rectangle. A rounded rectangle has rounded corners.

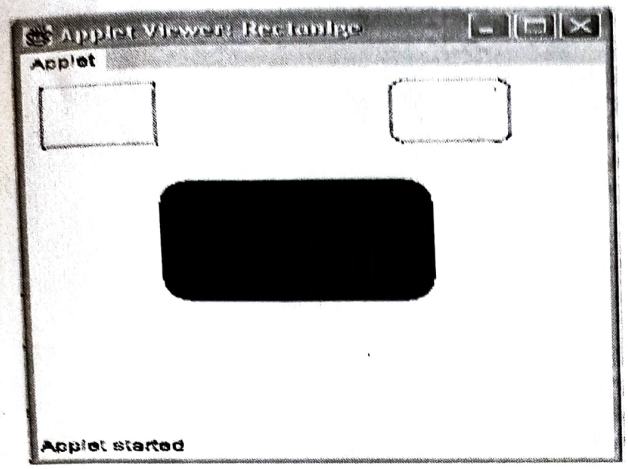
Syntax

void drawRoundRect(int top, int left, int width, int height int Xdiam, int YDiam)
void fillRoundRect(int top, int left, int width, int height int Xdiam, int YDiam)

The upper-left corner of the rounded rectangle is at (top,left). The dimensions of the rectangle are specified by width and height.

The diameter of the rounding along the X axis are specified by x Diam. The diameter of the rounding are along the Y axis is specified by Y Diam.

```
import java.awt.*;
import java.applet.*;
public class Rectanlge extends Applet
{
    public void paint(Graphics g)
    {
        g.drawRect(10,10,60,50);
        g.fillRect(100,100,100,0);
        g.drawRoundRect(190,10,60,50,15,15);
        g.fillRoundRect(70,90,140,100,30,40);
    }
}
```



Circles and Ellipses : The Graphics class does not contain any method for circles or ellipses. To draw an ellipse, use drawOval(). To fill an ellipse, use fillOval().

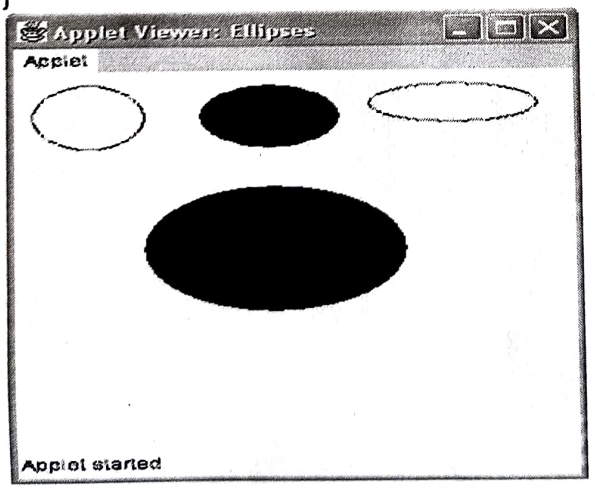
Syntax

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

The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by (top,left) and whose width and height are specified by width and height. To draw a circle, specify a square as the bounding rectangle i.e get height = width.

The following program draws several ellipses:

```
import java.awt.*;
import java.applet.*;
public class Ellipses extends Applet
{
    public void paint(Graphics g)
    {
        g.drawOval(10,10,60,50);
        g.fillOval(100,10,75,50);
        g.drawOval(190,10,90,30);
        g.fillOval(70,90,140,100);
    }
}
```



Drawing Arcs:

An arc is a part of oval. Arcs can be drawn with draw Arc() and fillArc() methods.
An arc is a part of an Oval, the easiest way to think of an arc is as a complete Oval.

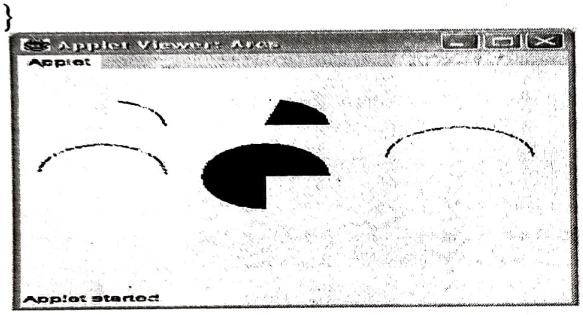
Syntax

void drawArc(int top, int left, int width, int height, int startAngle, int sweetAngle)
void fillArc(int top, int left, int width, int height, int startAngle, int sweetAngle)

The arc is bounded by the rectangle whose upper-left corner is specified by (top,left) and whose width and height are specified by width and height. The arc is drawn from startAngle through the angular distance specified by sweepAngle. Angles are specified in degree. '0' is on the horzontal, at the 30' clock's position. The arc is drawn conterclockwise if sweepAngle is positive, and clockwise if sweetAngle is negative.

The following applet draws several arcs:

```
import java.awt.*;
import java.applet.*;
public class Arcs extends Applet
{
    public void paint(Graphics g)
    {
        g.drawArc(10,40,70,70,0,75);
        g.fillArc(100,40,70,70,0,75);
        g.drawArc(10,100,70,80,0,175);
        g.fillArc(100,100,70,90,0,270);
        g.drawArc(200,80,80,80,0,180);
    }
}
```



Drawing polygons : Polygons are shapes with an unlimited number of sides. To draw a polygon, we need a set of 'x' and 'y' co-ordinates.

The drawPolygon() and fillPolygon() methods takes 3 arguments:

- An array of integers representing x-co-ordinates.
- An array of integers representing y-co-ordinates.
- An integer for the total number of points.

Polygons are shapes with many sides. It may be considered a set of lines connected together. The end of the first line is the beginning of the second line, the end of the second line is the beginning of the third line, and so on.

Syntax

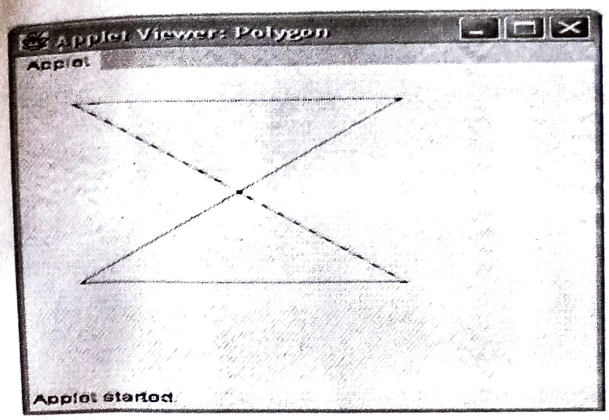
void drawPolygon(int ,[] , int y[], int numPointer)
void fillPolygon(int ,[] , int y[], int numPointer)

The polygon's endpoints are specified by the coordinate pairs contained within the x and y arrays. The number of points defined by x and y is specified by numPoints.

It is obvious that x and y arrays should be of the same size and we must repeat the first point at the end of the array for closing the polygon.

The following appplet draws an hourglass shape

```
import java.awt.*;
import java.applet.*;
public class Polygon extends Applet
{
    public void paint(Graphics g)
    {
        int xpoints[]={30,200,30,200,30};
        int ypoints[]={30,30,200,200,30};
        int num=5;
        g.drawPolygon(xpoints,ypoints,num);
}
```

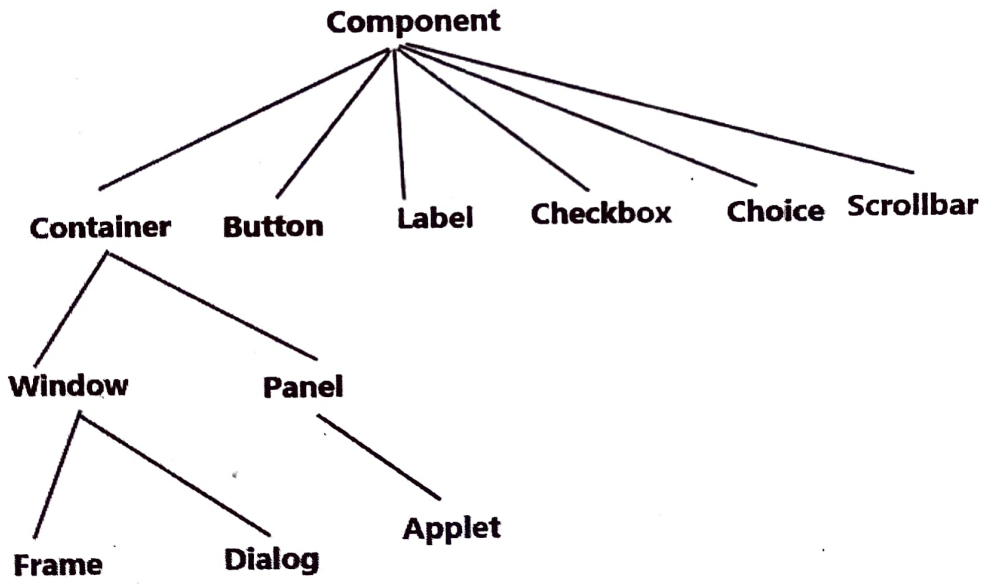


lab programs

Write a java applet program to print some text.
Write a java applet program to understand the properties of the font.
Write a java applet program to demonstrate the drawing function

Introduction to AWT:

AWT stands for **Abstract Window Toolkit**. It is a platform dependent API for creating Graphical User Interface (GUI) for java programs.
AWT class hierarchy: AWT is the class of the package java.awt.



The AWT class contained in java.awt package supplies the following user Interface component [UI] :

- Buttons
- Checkboxes
- Single line text fields
- Larger text display and editing areas
- Labels
- Lists
- Popup lists of choices
- Sliders and scrollbars
- Drawing areas
- Menus
- Containers

Methods for using UI Components in Applet:

- add() -> Adds the specified components
- remove() -> Removes the specified components
- setLayout() -> Sets the layout manager.

Windows fundamentals: The AWT defines windows according to a class hierarchy. The two most common windows are :

- Derived from Panel – This is used by applets.
- Derived from Frame – This creates standard window.

a. Component: A component is an object having a graphical representation that can be displayed on the screen and can interact with the user.

e.g. Buttons, Checkbox etc.,

b. Container: A generic Abstract Window Toolkit[AWT] container object is a component that can contain other AWT components.

Components added to a container are treated as a list. The components are added to the end of the list.

c. Panel: Panel is the simplest container class. A Panel provides space in which an application can attach any other component, including other panels.

d. Window: A Window object is a top-level window with no border and no menu bar. The default layout for a window is BorderLayout. A Window must have a frame, dialog or another window defined as its owner when it's constructed.

e. Frame: A Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. The default layout for a frame is BorderLayout.

f. Canvas: A Canvas component represents a blank rectangular area of the screen onto which the application can draw or from which the application can trap input events from the user. The paint method must be overridden in order to perform custom graphics on the screen.

****