APPLET

There are two types of java programs:

1. **Java Applications (Stand-alone programs).**
2. **Java Applets.**

   An **applet** is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. Applets are java programs that are primarily used in internet computing. They can be transported over the internet from one computer to another and run using the **Applet Viewer** or any web browser that supports java.

We can embedded applets into web pages in two ways:

1. We can write our own applets and embed them into web pages.
2. We can download an applet from a remote computer system and then embed it into web pages.

An applet developed locally and stored in a local computer is know as ***local applet.*** A ***remote applet*** is that which is developed by someone else and stored on a remote computer connected to the internet.

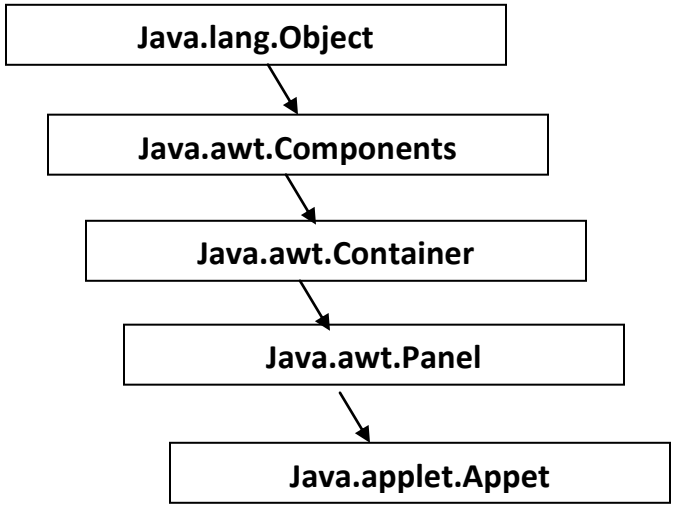
Difference between Applications and Applets:

- Applet do not use the main() method for initializing the execution of the code. Applets, when loaded automatically call certain method of the applet class and start and execute the applet code.
- Unlike stand-alone application, applets cannot run independently. They are run from inside a web page using special features known as HTML Tags.
- Applet cannot read from or write to a file in a local computer.
- Applets cannot communicate with other servers on network.
- Applets are restricted from using libraries from other languages such as C or C++.

Note:

An applet is a Java class that extends the **java.applet.Applet** class.

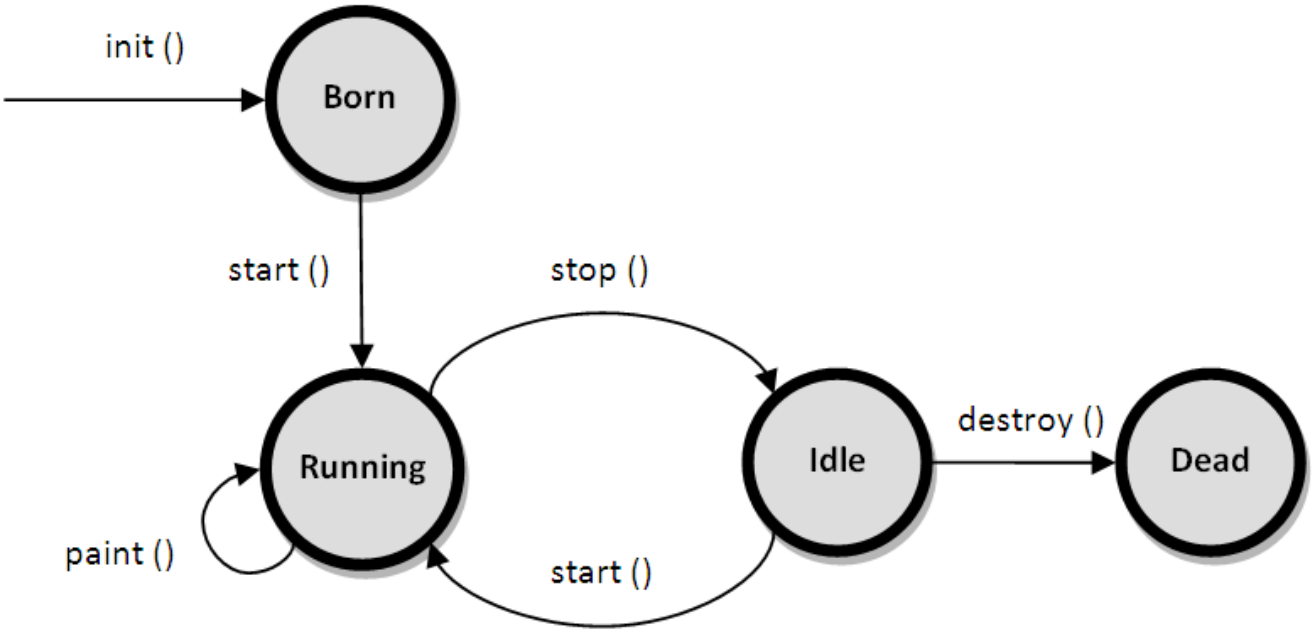
Remember that Applet class itself is a subclass of the Panel class, which is again a subclass of the container class and so on which is shown in following figure:



Life cycle of applet:

Every java applet inherits a set of default methods or behaviors from the Applet class. So when an applet is loaded, it undergoes a series of changes in its state. Applet states include:

- Born or initialization state.
- Running state.
- Idle state.
- Dead or Destroyed state.



1. Initialization State:

- Applet enters the initialization state when it is first loaded. This can be achieved by calling the applet methods called init().
- This method initializes the Applet and is invoked only once in the Applet life cycle. It helps to initialize variables and instantiate the objects and load the GUI of the applet. This is invoked when the page containing the applet is loaded. When the applet is born, we can do following task:
    - Create objects needed by applet.
    - Set the initial value.
    - Load image or fonts
    - Set up colors

  To provide any other of the above behaviors which is mentioned above, we must overrides the init () method as follow:

  Public void init ()

```
{
        //Action
}
```

2. Running State:

- Applet enters the running state when the system calls the start () method of the applet class. This occurs automatically after the applet is initialized. Starting can be also occurred if the applet is in already "Stopped state (idle)".
- **Example:**

  We may leave the web page containing the applet temporarily to another page and return back to the page. This again starts the applet running.
- Note that, unlike init() method, the start() method may be called more than one. We may override the start () method to create a thread to control applet.

```
Public void start ()
{
        //Action
}
```

3. Idle or Stopped State:

- The web browser will call the Applets stop() method, if the user moved to another web page while the applet was executing. So that the applet can take a breather while the user goes off and explores the web some more. The stop() method is called at least once in Applets Lifecycle.
- If we use a thread to run the applet, then we must use stop () method to terminate the thread. We must achieve this by:

```
Public void stop()
{
```

```
        //Action
    }
```

4. Dead State:

- Finally, if the user decides to quit the web browser, the web browser will free up system resources by killing the applet before it closes. To do so, it will call the applets destroy() method. One can override destroy() to perform one-time tasks upon program completion. for example, cleaning up threads which were started in the init() method.
- If the applet has created many any resources, like thread, we may override the destroy() method to clean up these resources.

```
Public void destroy()
{
        //Action
}
```

- **Note:** If the user returns to the applet, the web browser will simply call the applet's start() method again and the user will be back into the program.

5. Display State:

- Applet moves to the display state whenever it has to perform the output operations on the screen. This happens immediately after the applet enters into the running state. The paint() method is called to accomplish this task.
- Almost every applet will have a paint() method. Like other methods in the life cycle, the default version of Paint() method does absolutely nothing.

```
Public void paint (Graphics g)
{
            //Display statements
}
```

- Note that the display method is not a part of applet life cycle. In fact the paint() method defined in the Applet class.

Applet Lifecycle Example:

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.*;
/*<applet code="AppletLifeCycle.class" width="350" height="150">
 </applet>*/
public class AppletLifeCycle extends Applet
{
    public void init()
    {
     setBackground(Color.CYAN);
     System.out.println("init() called");
    }
    public void start()
    {
      System.out.println("Start() called");
    }
    public void paint(Graphics g)
    {
      //System.out.println("Paint(() called");
      g.drawString("Heta",50,50);
    }
    public void stop()
    {
      System.out.println("Stop() Called");
    }
    public void destroy()
    {
      System.out.println("Destroy)() Called");
    }
}
```

First applet Program:

Example:

```
import java.awt.*;
import java.applet.*;
public class SimpleApplet extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString("My First Applet",40,40);
  }
}
```

Note:

- Save the file as **SimpleApplet.java**
- Create **SimpleApplet.class**

- **In this example:**

  - In the first line we imports the Abstract Window Toolkit (AWT) for the **Graphics class.**
  - In the second line we import the Applet package, which contains the class "Applet". As every applet that we create is the subclass of Applet.
  - The next line declares the class SimpleApplet. This class must be declared in public, because it will be accessed by code that is outside the program.
  - Inside simpleApplet, paint() method is declared. This method is defined by the AWT and must be overridden by the Applet. Method paint() is called each time that the applet must redisplay its output.

    This paint () method has parameter of type "Graphics". This parameter contains the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

- Inside paint() method is a call to drawstring(), which is a member of the Graphics class. This method output a String beginning at specified X, Y locations.

Run the applet:

- There are two ways in which one can run an applet, as follows:
  - Executing the applet within a java-compatible web browser.
  - Using an applet viewer, such as the standard SDK tool, "appletviewer". An applet viewer executes your applet in a w3indow. This is generally the fastest and easiest way to test your applet.

Graphics class:

Java's Graphics class includes methods for drawing many different types, from simple lines to polygons to text in a variety of fonts.

Drawing methods of Graphics class:

Methods	Description
drawArc()	Draws a hollow arc.
drawLine()	Draws a straight line.
drawOval()	Draws hollow oval.
drawPolygon()	Draws a hollow Polygon
drawRect()	Draws a hollow rectangle.
drawstring()	Displays a text string.
drawRoundRect()	Draws a hollow rectangle with rounded corners.
fillArc()	Draws a filled arc.
fillOval()	Draws a filled oval
fillPloygon()	Draws filled polygon.
fillRect()	Draws filled rectangle.
fillRoundRect()	Draws filled rectangle with rounded corner.
getcolor()	Retrieve the current drawing color.
getFont()	Retrieves the currently used font.

setColor()	Sets the drawing color.
setFont()	Sets the font.

Designing a web page:

A web page is basically mase up of text and HTML tags that can be interpreted by web browswer or applet viewer.

<html>

<body>

<Applet…>

</Applet>

</body>

</html>

APPLET TAG:

We have seen <Applet> and </Applet> tag in body section. The tag <Applet…> supplies the name of the applet to be loaded and tells the browser how much space the applet requires.

<Applet code="SimpleApplet.class" width=400 height=200>

</Applet>

The html tag tells that browser to load the compiled java applet SimpleApplet.class, which is in the same directory as the HTML file.

<Applet> tag specifies three things:

1. Name of the applet.

2. Width of the applet(in pixels)
3. Height of the applet(in pixels).

Commonly used Graphics methods Syntax:

1. **Public 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 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 void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **public 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 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 boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **public 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 void setColor(Color c):** is used to set the graphics current color to the specified color.
11. **public void setFont(Font font):** is used to set the graphics current font to the specified font.

Example:

```
import java.applet.Applet;
import java.awt.*;
/*
 <applet code="GraphicsDemo.class" height=300 width=300>
 </applet>
 */
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(70,100,70,30);

  g.setColor(Color.blue);
  g.fillRect(170,100,30,30);

  g.setColor(Color.black);
  g.drawOval(70,200,50,30);

  g.setColor(Color.magenta);
  g.fillOval(170,200,60,30);

  g.drawArc(90,150,30,30,30,270);

  g.fillArc(270,150,100,100,0,90);
  g.fillArc(350,150,100,100,10,90);
  g.fillArc(450,150,100,100,20,90);
  g.fillArc(550,150,100,100,30,90);
  g.fillArc(650,150,100,100,40,90);
  g.fillArc(750,150,100,100,50,90);
```

```
  g.fillArc(850,150,100,100,90,90);
}
}
```

How to pass Parameters from HTML page to Applet:

Java applet has the feature of retrieving the parameter values passed from the html page. So, you can pass the parameters from your html page to the applet embedded in your page.

The **param tag**(<param name="" value=""></param>) is used to pass the parameters to an applet.

Passing parameters to an applet code using<param..> tag is something like passing parameters to the main() method using command line argument. For this we need to do two things:

1. Include <param..> Tags in the HTML document.
2. Provide code in the applet to parse these parameters.

Paramters are passed on an applet when it is loaded. We can define init() method in the applet to get hold of parameters define in the <param> Tags. This is done using the **getParamter()** method which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

Example:

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

/*<applet code="ParamExample.class" height=300 width=200>
 <param name="ParamName" value="hello">
```

```
 <param name="ParamRoll" value="1">
 </applet>*/

public class ParamExample extends Applet
{
  String sname;
  String Roll_number;
  public void init()
  {
    sname=getParameter("ParamName");
    Roll_number=getParameter("ParamRoll");
  }
  public void paint(Graphics g)
  {
    g.drawString(sname,30,30);
    g.drawString(Roll_number,70,70);
  }
}
```

How to take input from user in Textbox in applet

Example:

```
import java.awt.*;
import java.applet.*;
/*
<applet code="User_input_Textfield.class" width=400 height=400>
</applet>
*/

public class User_input_Textfield extends Applet
{
  TextField t1,t2;
  public void init()
  {
```

```
    t1=new TextField(5);
    t2=new TextField(5);
    add(t1);
    add(t2);
  }
  public void paint(Graphics g)
  {
    int s;

    s=Integer.parseInt(t1.getText());
    int sqrt=s*s;
    t2.setText(String.valueOf(sqrt));

  }
}
```

Displaying Image in Applet

Applet is mostly used in games and animation. For this purpose image is required to be displayed. The java.awt.Graphics class provide a method **drawImage()** to display the image.
Syntax:

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

How to get the object of Image:

The java.applet.Applet class provides **getImage()** method that returns the object of Image.
Syntax:
public Image getImage(URL u, String image){}

public URL getDocumentBase(): is used to return the URL of the document in which applet is embedded.

public URL getCodeBase(): is used to return the base URL.

Example:

```
import java.awt.*;

import java.applet.*;

 /*<applet code ="DisplayImage" width=400 height=400>

   </applet>*/


public class DisplayImage extends Applet
{

  Image picture;

  public void init()
{
    picture = getImage(getDocumentBase(),"Koala.jpg");
  }

  public void paint(Graphics g)
 {
    g.drawImage(picture, 30,30, this);
  }

  }
```