JAVA PROGRAMMING

COURSE NAME: MCA – 5TH SEMESTER

COURSE CODE: MCA18501CR

Teacher Incharge: *Dr. Shifaa Basharat*
Contact: *fazilishifaa@gmail.com*

APPLETS

Applets in Java are small and dynamic internet-based programs. A Java Applet can be only executed within the applet framework of Java. Generally, the applet code is embedded within an HTML page. The applet codes are executed when the HTML pages get loaded into the Java-compatible web browsers. Applets are mainly downloaded on remote machines and used on the client side.

A Java applet can be a fully functional Java application as well since it can utilize a complete Java API at its own accord. But still, there is a thin line between an applet and an application in Java.

Java Applet vs Java Application

Java Application	Java Applet
Java Applications are the stand-alone programs which can be executed independently	Java Applets are small Java programs which are designed to exist within HTML web document
Java Applications must have main() method for them to execute	Java Applets do not need main() for execution
Java Applications just needs the JRE	Java Applets cannot run independently and require API's
Java Applications do not need to extend any class unless required	Java Applets must extend java.applet.Applet class
Java Applications can execute codes from the local system	Java Applets Applications cannot do so
Java Applications has access to all the resources available in your system	Java Applets has access only to the browser-specific services

Note:

1. All applets are sub-classes (either directly or indirectly) of *java.applet.Applet* class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

Applet Hierarchy in Java

- class java.lang.**Object**
  - class java.awt.**Component**
    - class java.awt.**Container**

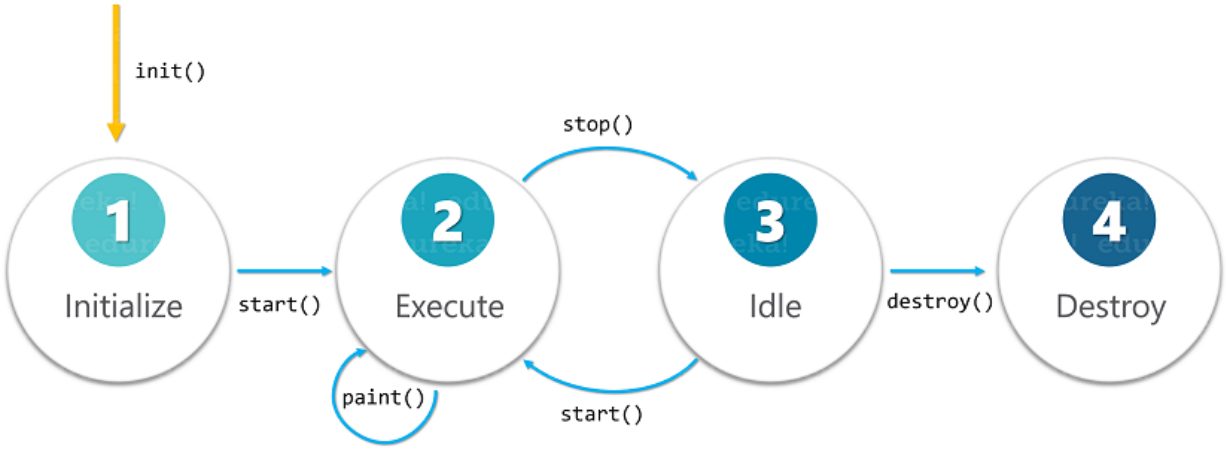
- class java.awt.**Panel**
  - class java.applet.**Applet**

The Java Applet class which is a class of applet package extends the Panel class of awt package. The Panel class is a subclass of the Container class of the same package. The Container class is an extension of Component class belonging to the same package. The Component class is an abstract class and derives several useful classes for the components such as Checkbox, List, buttons, etc.

Lifecycle of Java Applet

Following are the stages in Applet

1. Applet is initialized.
2. Applet is started
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.



There are 4 main methods which are mandatory for any Java Applet to override as discussed below:

1. **public void init():** This is the very first method to be invoked during the life cycle of an applet. In this method, the variable that will be used further in the applet is initialized. One thing you must note here is that this method can be invoked only once per applet life cycle.
2. **public void start():** This is the second method that is invoked just after the init() method is called by the browser. Each time a user revisits the web page containing the applet, start() method is invoked and the applet is started.
3. **public void stop():** This method is invoked whenever a user leaves the web page containing applet. In other words, the stop() method is used to suspend the threads which are not required when the applet is in the background or is not visible on the screen. These can be easily resumed using the start() method.

4. **public void destroy():** Finally, we have the destroy() method which is invoked in order to completely remove an applet from the memory. This method is invoked only once per applet life cycle and all the engaged resources must be freed up before this method is called.

One more method that is mostly used along with the above four is paint().

- **public void paint(Graphics g):** This method is invoked whenever an applet needs to be redrawn or repainted in the browser, irrespective of the cause. The paint() method takes one Graphic object as a parameter that contains the graphics context in which the applet is being executed. Also, this method is invoked each time output is expected from the applet.

The use of above functions has been demonstrated below by a simple example:

```
import java.applet.*;

public class AppletLifeCycle extends Applet{
public void init(){
System.out.println("Applet is Initialized");
}
public void start(){
System.out.println("Applet is being Executed");
}
public void stop()
{
System.out.println("Applet execution has Stopped");
}
public void paint(Graphics g)
{
System.out.println("Painting the Applet...");
}
public void destroy()
{
System.out.println("Applet has been Destroyed");
}
}
```

Creating a HelloWorld Applet:

```
// A Hello World Applet
// Save file as HelloWorld.java

import java.applet.Applet;
import java.awt.Graphics;
```

```
// HelloWorld class extends Applet
public class HelloWorld extends Applet
{
   // Overriding paint() method
   @Override
   public void paint(Graphics g)
   {
      g.drawString("Hello World", 20, 20);
   }
  }
```

Explanation:

1. The above java program begins with two import statements. The first import statement imports the Applet class from applet package. Every AWT-based(Abstract Window Toolkit) applet that you create must be a subclass (either directly or indirectly) of Applet class. The second statement imports the Graphics class from AWT package.
2. The next line in the program declares the class HelloWorld. This class must be declared as public because it will be accessed by code that is outside the program. Inside HelloWorld, **paint( )** is declared. This method is defined by the AWT and must be overridden by the applet.
3. Inside **paint( )** is a call to *drawString( ),* which is a member of the Graphics class. This method outputs a string beginning at the specified X,Y location. It has the following general form:

   void drawstring(String message, int x, int y)

   Here, message is the string to be output beginning at x,y. In a Java window, the upper-left corner is location 0,0. The call to *drawString( )* in the applet causes the message "Hello World" to be displayed beginning at location 20,20.

Running the HelloWorld Applet :

After you enter the source code for HelloWorld.java, compile in the same way that you have been compiling java programs(using *javac* command). However, running HelloWorld with the *java* command will generate an error because it is not an application.

```
java HelloWorld
Error: Main method not found in class HelloWorld, please define the main method as:
  public static void main(String[] args)
```

There are two standard ways in which you can run an applet :

1. Executing the applet within a Java-compatible web browser:
   To execute an applet in a web browser we have to write a short HTML text file that contains a tag that loads the applet. We can use APPLET or OBJECT tag for this purpose. Using APPLET, here is the HTML file that executes HelloWorld :

```
<applet code="HelloWorld" width=200 height=60>
</applet>
```

The width and height statements specify the dimensions of the display area used by the applet. The APPLET tag contains several other options. After you create this html file, you can use it to execute the applet.

2. Using an applet viewer, such as the standard tool, applet-viewer. An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet. To execute HelloWorld with an applet viewer, you may also execute the HTML file. For example, if the preceding HTML file is saved with RunHelloWorld.html,then the following command line will run HelloWorld :

```
appletviewer RunHelloWorld.html
```

3. appletviewer with java source file : If you include a comment at the head of your Java source code file that contains the APPLET tag then your code is documented with a prototype of the necessary HTML statements, and you can run your compiled applet merely by starting the applet viewer with your Java source code file. If you use this method, the HelloWorld source file looks like this :

```
// A Hello World Applet
// Save file as HelloWorld.java

import java.applet.Applet;
import java.awt.Graphics;

/*
<applet code="HelloWorld" width=200 height=60>
</applet>
*/

// HelloWorld class extends Applet
public class HelloWorld extends Applet
```

```
{
   // Overriding paint() method
   @Override
   public void paint(Graphics g)
   {
      g.drawString("Hello World", 20, 20);
   }

}
```

With this approach, first compile HelloWorld.java file and then simply run below command to run applet :

```
appletviewer HelloWorld
```

Restrictions imposed on Java applets

Due to security reasons , the following restrictions are imposed on Java applets:

1. An applet cannot load libraries or define native methods.
2. An applet cannot ordinarily read or write files on the execution host.
3. An applet cannot read certain system properties.
4. An applet cannot make network connections except to the host that it came from.
5. An applet cannot start any program on the host that's executing it.

Graphics in Applet

In Applet, java.awt.Graphics class provides methods for using graphics. Below are the Methods of the Graphics class.

1. ***public abstract void drawString(String str, int x, int y) :*** Used to draw specified string.
2. ***public void drawRect(int x, int y, int width, int height) :***Used to draw a rectangle of specified width and height.
3. ***public abstract void fillRect(int x, int y, int width, int height):*** Used to draw a rectangle with a default colourof specified width and height.
4. ***public abstract void drawOval(int x, int y, int width, int height):*** Used to draw oval of specified width and height.
5. ***public abstract void fillOval(int x, int y, int width, int height):*** Used to draw oval with a default colour of specified width and height.
6. ***public abstract void drawLine(int x1, int y1, int x2, int y2) :*** Used for drawing lines between the point (x1, x1) and (x2, y2).

7. ***public abstract booleandrawImage(Image img, int x, int y, ImageObserver observer):*** Used for drawing a specified image.
8. ***public abstract void drawArc(int x, int y, int width, int height, intstartAngle, intarcAngle)*** :Used for drawing a circular arc.
9. ***public abstract void fillArc(int x, int y, int width, int height, intstartAngle, intarcAngle):*** Used for filling circular arc.
10. ***public abstract void setColor(Color c) :***Used to set a colour to the object.
11. p***ublic abstract void setFont(Font font)***: Used to set font.

Example: GraphicsDemo.java

```
import java.applet.Applet;
import java.awt.*;
public class GraphicsDemo1 extends Applet
{
 public void paint(Graphics g)
 {
  g.setColor(Color.black);
  g.drawString("Welcome to studytonight",50, 50);
  g.setColor(Color.blue);
  g.fillOval(170,200,30,30);
  g.drawArc(90,150,30,30,30,270);
  g.fillArc(270,150,30,30,0,180);
  g.drawLine(21,31,20,300);
  g.drawRect(70,100,30,30);
  g.fillRect(170,100,30,30);
  g.drawOval(70,200,30,30);
 }
}
```

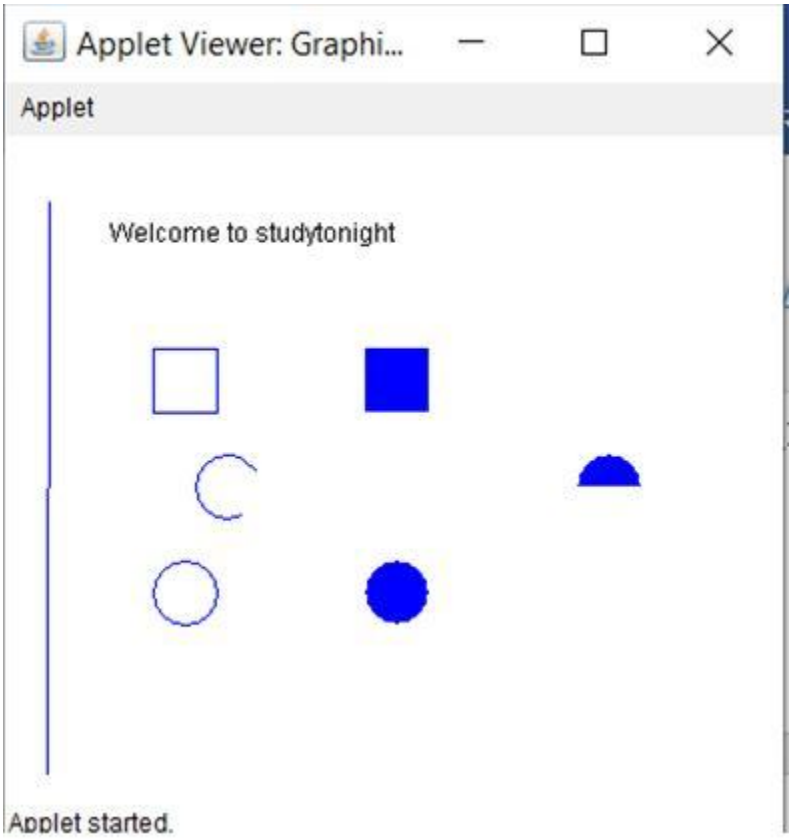
GraphicsDemo.html

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

Execution steps:

```
javac GraphicsDemo.java
appletviewer GraphicsDemo.html
```

Output:



Displaying image in an applet

Applets are widely used in animation and games. The Graphics class of Java, provides a drawImage() method to display an image. The syntax of the drawImage function to draw the specified image is:

```
public abstract boolean drawImage(Image image, int a, int b, ImageObserver observer)
```

Java provides support for two common image formats: GIF and JPEG. An image that is in one of these formats can be loaded by using either a URL, or a filename. The java.applet.Applet class provides a method getImage() that returns the image object. The syntax of the getImage funtion to draw the specified image, is:

```
public Image getImage(URL url. String image)
```

Some other required methods are:
1. public URL getCodeBased() returns the base URL.

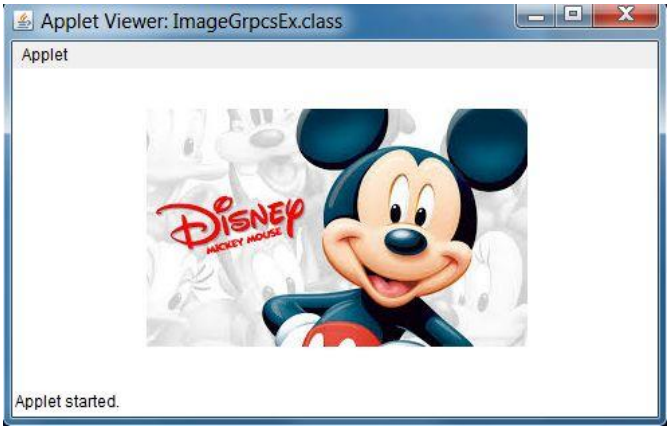
2. public URL getDocumentBase() returns the URL of the document in which the Applet is embedded.

In most of the applets, it is required to load text and images explicitly. Java enables loading data from two directories. The first one is the directory which contains the HTML file that started the applet (known as the **document base).** The other one is the directory from which the class file of the applet is loaded (known as the **code base).** These directories can be obtained as URL objects by using getDocumentBase ()and getCodeBase ()methods respectively. You can concatenate these URL objects with the string representing the name of the file that is to be loaded.

Example: Displaying Image in an Applet

```
import java.applet.*;
import java.awt.*;
public class ImageGrpcsEx extends Applet
{
   Image pic;
   public void init()
   {
      pic = getImage(getDocumentBase(), "images.jpeg");
   }
   public void paint(Graphics grp)
   {
      grp.drawImage(pic, 100, 30, this);
   }
}
```

Output:



Example: Displaying Animation in an Applet

```
import java.awt.*;
import java.applet.*;
public class AnimationEx extends Applet
{
   Image pic;
   public void init()
   {
      pic = getImage(getDocumentBase(), "images.jpeg");
   }
   public void paint(Graphics grp)
   {
      for (int i = 50; i < 600; i++)
      {
         grp.drawImage(pic, i, 30, this);
         try
         {
            Thread.sleep(400);
         } catch (Exception e) {}
      }
   }
}
```