EXPERIMENT NO. -08

AIM: Edit, compile, execute and test Java applets.

THEORY:

Java Applet Basics

Let's understand first how many Package does GUI support:

1. AWT(Abstract Window Toolkit)
2. Swing

Throwback of making GUI application:
Java was launched on 23-Jan-1996(JDK 1.0) and at that time it only supported CUI(Character User Interface) application. But in 1996 VB(Visual Basic) of Microsoft was preferred for GUI programming. So the Java developers in hurry(i.e within 7 days) have given the support for GUI from Operating System(OS). Now, the components like button,etc. were platform-dependent(i.e in each platform there will be different size, shape button). But they did the intersection of such components from all platforms and gave a small library which contains these intersections and it is available in AWT(Abstract Window Toolkit) technology but it doesn't have advanced features like dialogue box, etc.

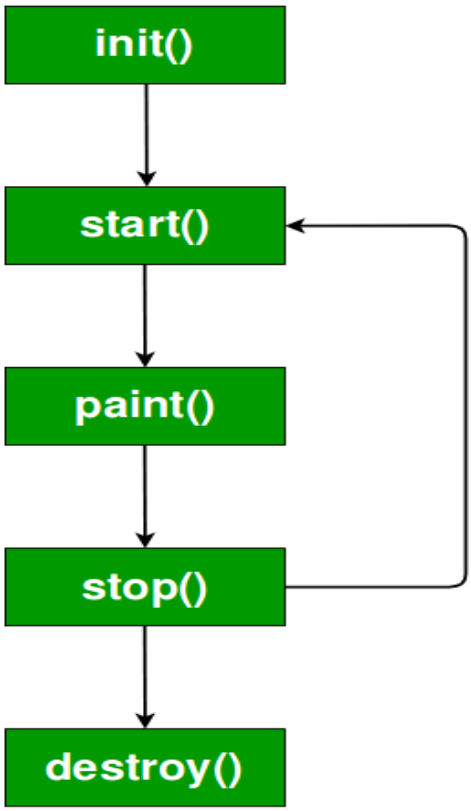
Now to run Applet, java needs a browser and at that time only "Internet Explorer" was there of Microsoft but Microsoft believes in monopoly. So "SUN Micro-System"(the company which developed Java) contracted with other company known as "Netscape"(which developed Java Script) and now the "Netscape" company is also known as "Mozilla Firefox" which we all know is a browser. Now, these two companies have developed a technology called "SWING" and the benefit is that the SWING components are produced by Java itself. Therefore now it is platform-independent as well as some additional features have also been added which were not in AWT technology. So we can say that SWING is much more advanced as compared to AWT technology.

What is Applet?
An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.
Applets are used to make the web site more dynamic and entertaining.

Important points :

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()*.

Life cycle of an applet :



It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:

1. stop( )
2. destroy( )

Let's look more closely at these methods.

1. **init( ) :** The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.
2. **start( ) :** The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )** is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.
3. **paint( ) :** The **paint( )** method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.

paint( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.

1. The **paint( )** method has one parameter of type Graphics. This parameter will contain 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.
   Note: This is the only method among all the method mention above, which is parametrised. It's prototype is
   public void paint(Graphics g)
   where g is an object reference of class Graphic.

Now the **Question Arises:**

Q. In the prototype of paint() method, we have created an object reference without creating its object. But how is it possible to create object reference without creating its object?

Ans. Whenever we pass object reference in arguments then the object will be provided by its caller itself. In this case the caller of paint() method is browser, so it will provide an object. The same thing happens when we create a very basic program in normal Java programs. For Example:

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

Here we have created an object reference without creating its object but it still runs because it's caller,i.e JVM will provide it with an object.

2. **stop( ) :** The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.
3. **destroy( ) :** The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy( )**.

Features of Applets over HTML

- Displaying dynamic web pages of a web application.
- Playing sound files.
- Displaying documents
- Playing animations

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.

EXPERIMENT NO. -08

AIM: Edit, compile, execute and test Java applets.

PROGRAM:

Creating Hello World applet :
Let's begin with the HelloWorldapplet :

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

importjava.applet.Applet;
importjava.awt.Graphics;

// HelloWorld class extends Applet
publicclassHelloWorld extendsApplet
{
    // Overriding paint() method
    @Override
    publicvoidpaint(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 import 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:

```
4. 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.

Notice that the applet does not have a **main( )** method. Unlike Java programs, applets do not begin execution at **main( )**. In fact, most applets don't even have a **main( )** method. Instead, an applet begins execution when the name of its class is passed to an applet viewer or to a network browser.

Running the HelloWorldApplet :
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.

```
javaHelloWorld



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.
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.

Each of these methods is described next.

1. **Using java enabled 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 :

```
2. <applet code="HelloWorld" width=200 height=60>
3. </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.

   NOTE : Chrome and Firefox no longer supports NPAPI (technology required for Java applets). Refer here

4. **Using appletviewer :** This is the easiest way to run an applet. To execute HelloWorld with an applet viewer, you may also execute the HTML file shown earlier. For example, if the preceding HTML file is saved with RunHelloWorld.html,then the following command line will run HelloWorld :

```
   appletviewer RunHelloWorld.html
```

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

importjava.applet.Applet;
importjava.awt.Graphics;

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

// HelloWorld class extends Applet
publicclassHelloWorld extendsApplet
{
    // Overriding paint() method
    @Override
    publicvoidpaint(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 :

```
appletviewerHelloWorld
```

CONCLUSION: Thus we have successfully studied about Editing, compiling, executing and testing Java applets**.**