B.Sc (Computer Science)
Programming in Java
Unit-III

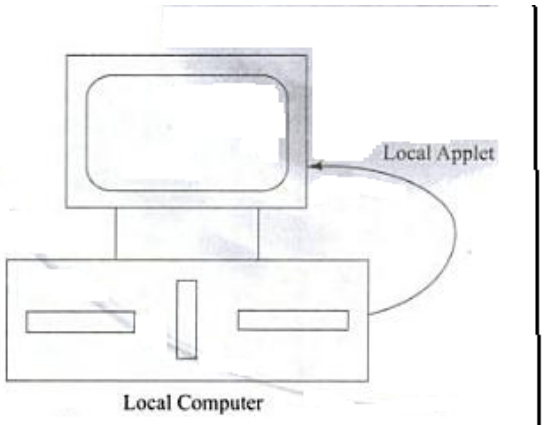
1. What is applet? How applet differs from application programs. (Mar 2011)

Applets are small java programs that are 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. Applet can perform arithmetic operations, display graphics, play sounds, accept user input, create animation and play interactive games.

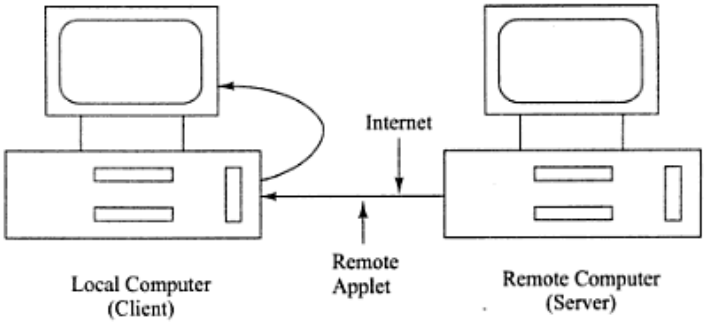
Local and Remote applet: We can embed applet into web pages in two ways. We can write our own applets and embed them into web pages. We can download an applet from a remote computer system and then embed into a web page.

An applet developed locally and stored in a local system is known as local applet. Local system does not require the internet connection. It simply searches the directories in local systems.

A remote applet is that which is developed by someone else and stored on a remote computer connected to the internet. If our system is connected to the internet, we can download the remote applet onto our system via internet and run it. In order to locate and load a remote applet we must know the applet's address on the web. This address is known as Uniform Resource Locator (URL).



Loading local applets



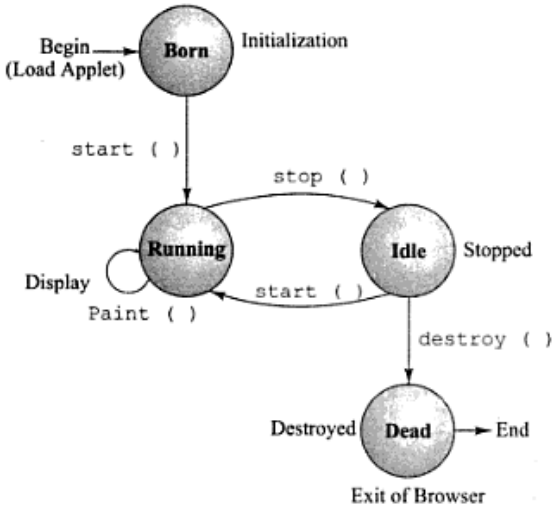
Loading a remote applet

Applets Differ from Applications: Both the applets and the stand alone applications are the java programs, there are significant differences between them. Applets are not full featured application programs. They are usually written to accomplish a small task or a component of a task. They are designed for use on the internet and have certain limitations and restrictions in their design:

- Applets do not use the main( ) method for initiating the execution of the code. when it loaded, automatically call certain method of applet class to start and execute the applet code.
- Unlike stand-alone applications, Applets cannot be run independently. They are run from inside a web page using a special feature known as HTML tag.
- Applets cannot read from or write to the files in the local computer.
- Applets cannot communicate with other serves on the network.
- Applets cannot run any program from the local computer.
- Applets area restricted from using libraries from other languages such as C or C++.

2. Write Life Cycle of an applet? (Mar 2010) (Mar 2011) (Oct 2012)

Every java applet inherits a set of default behaviours from the Applet class, when an applet is loaded it undergoes a series of changes in its state as shown below:



- Born on initialization state
- Running state
- Idle state
- Dead or destroyed state

Initialization state: Applets enters the initialization state when it is first loaded. This is achieved by calling the **init( )** method of applet class. The applet is born, we may do following:

- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

The initialization occurs only once in the applet life cycle.

```
public void init( )
{
….
….
}
```

Running state: Applets enters the running state when the system calls the **start( )** method of applet class. This occurs automatically after the applet is initialized. Starting can also occur if the applet is in stopped state. The start method may be called more than once. We may override start( ) method to create a thread to control the applet.

```
public void start( )
{
….
….
}
```

Idle or stopped state : An applet becomes idle when it is stopped from running. Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the **stop ( )** method explicitly. If we use a thread to run the applet, then we must use stop( ) method to terminate the thread.

```
public void stop( )
{
…..
…..
}
```

Dead state: An applet is said to be dead when it is removed from memory. This occurs automatically by invoking the **destroy ( )** method when we quit the browser. Destroying stage occurs only once in the applet's life cycle.

3. What are the steps involved in developing, running and testing an applet? (Mar 2012)

Preparing an Applet:

Before we try to write applets, we must make sure that java is installed properly and also ensure that either the java appletviewer or a java-enabled browser is available.

The steps involved are:

1. Building an applet code (.java file)
2. Creating an executable applet (.class file)
3. Designing a web page using HTML tags.
4. Preparing <APPLET> tag .
5. Incorporating <APPLET> tag into the web page.
6. Creating HTML file.
7. Testing the applet code.

Building Applet code:

The applet class from java.applet package provides life and behavior to the applet through its methods such as **init( )**, **start( )**, and **paint( )**. Java calls the main( ) method directly to initiate the execution of the program, when an applet is loaded , java automatically calls a series of applet class methods for starting , running, and stopping the applet code. The applet class therefore maintains the lifecycle of an applet.

The paint( ) method of the applet class displays the result of the applet code when it is called. The output may be text, graphics, or sound. The paint( ) method which requires a Graphics object as an argument is defined as follows:

public void paint (Graphics g)

Syntax:

```
import java.awt.*;
import java.applet.*;
….
…..
public class appletclassname extends Applet
{
……
……
        public void paint (Graphics g)
        {
        ……
        ……
```

```
        }
……
……
}
```

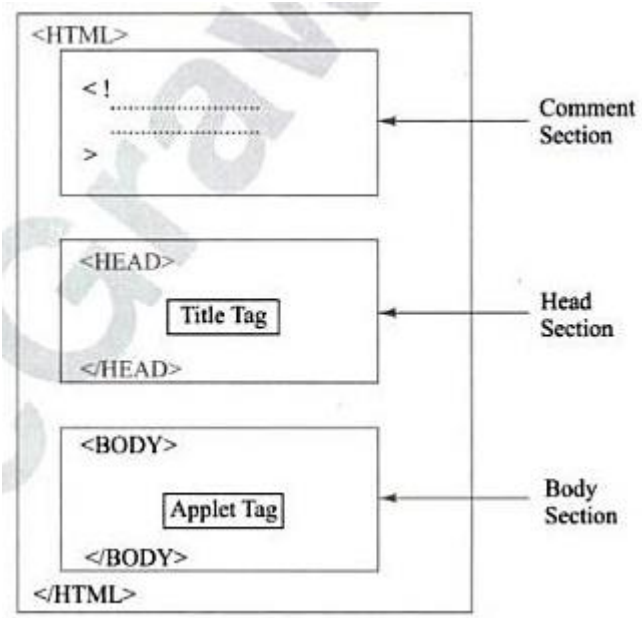
Example:

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet
{
        public void paint (Graphics g)
        {
                g.drawString("Hello java", 10, 100);
        }
}
```

Creating an Executable Applet: Executable applet is nothing but the .class file of the applet, which is obtained by compiling the source code of the applet. The steps required for compiling the Hellojava applet.

- Move to the directory containing the source code and type the following commands:
  javac Hellojava.java
- The compiled output file called **Hellojava.class** is placed in the same directory as the source.
- If any error message is received then we must check for errors, correct them and compile the applet again.

Designing a web page: (Mar 2012) Java applets are programs that reside on Web pages. In order to run a java applet it is first necessary to have a web page that references that applet. Web page is basically made up of text and HTML tags that can be interpreted by a web browser or an applet viewer. Java source code can be prepared using any **ASCII text editor** (e.g. Notepad). A web page is also known as HTML page or HTML documents. Web pages are stored using a file extension **.html** such as **MyApplet.html**.



Web pages include both text that we want to display and HTML tags (commands) to Web browsers. A Web page is marked by an opening HTML tag < HTML> and a closing HTML tag and is divided into the following **three sections (Oct 2010)**:

1. Comment section (Optional)
2. Head section (Optional)
3. Body section

Comment Section: This section contains comments about the Web page. A comment line begins with <! And ends with >. Web browsers will ignore the text enclosed between them. Comments are optional and can be included anywhere in the Web page.

Head Section: The head section is defined with a starting **<HEAD>** tag and a closing **</HEAD>** tag. This section usually contains a title for the Web page. The text enclosed in the tags will appear in the title bar of the Web browser when it displays the page. The head section is also optional.

Body Section: After the head section, comes the body section. We call this as body section because this section contains the entire information about the Web page and its behaviour. We can set up many options to indicate how our page must appear on the screen (like colour, location, sound, etc.). Shown below is a simple body section:

Preparing <APPLET> tag: The <APPLET…> tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires. The ellipsis in the tag <APPLET….> indicates that it contains certain attribute that must specified.

```
<APPLET
        CODE = hellojava.class
        WIDTH = 400
        HEIGHT = 200 >
</APPLET>
```

The HTML code tells the browser to load the compiled **java applet hellojava.class** which is in the same directory as the html file. Also specifies area for the applet output as **400 pixels width** and **200 pixels height**.

Incorporating <APPLET> tag into the web page: Now we can put together the various components of the web page and create a file known as HTML file. Insert the <APPLET> tag in the page at the place where the output of the applet must appear.

Creating HTML file:

```
<HTML>
        <HEAD>
                <TITLE>
                Welcome to java applet
                </TITLE>
        </HEAD>
        <BODY>
                <CENTER>
                        <H1> welcome to the world of applet </H1>
                </CENTER>
                <BR>
                <CENTER>
                        <APPLET
                                CODE = hellojava.class
                                WIDTH = 400
                                HEIGHT = 200>
                        </APPLET>
                </CENTER>
```

```
</BODY>
</HTML>
```

Testing (Running) the applet code: To run the applet we require one of the following

Tools (Oct 2012):

1. Java-enabled web browser (such as HotJava, Netscape, Internet Explorer)
2. Java appletViewer

If we use java-enabled web browser we will see entire web page containing the applet. If we use the appletviewer tool we will only see applet output. The appletviewer is available as a part of the Java Development Kit. We can use it to run or applet as follows :

appletviewer hellojava.html

4. Why do applet classes need to be declared as public? (Mar 2011)

This **applet class** is declared as **public** so that the class can be accessed when the applet is run in a **Web browser** or in the **appletviewer** application. If we fail to declare the applet class as public, the code will compile fine, but the applet will refuse to run. In other words, all applet classes must be public.

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet
{
        public void paint (Graphics g)
        {
                g.drawString("Hello java", 10, 100);
        }
}
```

5. How do we pass parameters to applets?(Mar 2012) (Oct 2012)

We can supply user-defined parameters to an applet using <PARAM…> tags. Each <PARAM…> tag has a name attribute such as color, and a value attribute such as red. Inside the applet code, the applet can refer to that parameter by name to find its value.

We can change the color of the text displayed to red by an applet by using a <PARAM…> tag as follows:

```
<APPLET>
<PARAM= color value = "red">
<APPLET>
```

Similarly, we can change the text to be displayed by an applet by supplying new text to the applet through a <PARAM…> tag as shown below:

<PARAM NAME=text VALUE="JAVA">

Passing parameters to an applet code using <PARAM> tag is something similar to passing parameters to the main() method using command line arguments. To set up and handle parameters, we need to do two things.

1. Include appropriate <PARAM..> tags in the HTML Document.
2. Provide Code in the applet to parse these parameters.

Parameters are passed on an applet when it is loaded. We can define the init() method in the applet to get hold of the parameters defined in the <PARAM> tags.

```
import java.awt.*;
import java.applet.*;
public class HelloJavaParam extends Applet
{       String str;
        public void init()
        {
                str= getParameter ("string");        //Receiving parameter value
                if( str==null)
                        str="Java";
                str="hello"+str;          //Using the value
        }
        public void paint(Graphics g)
        {
                g.drawString("str",10, 100);
        }
}
```

6. Explain align attribute. (Mar 2010)

We can align the output of applet using the ALIGN attribute. This attribute can have one of the nine values : **LEFT, RIGHT, TOP, TEXT, TOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM, ABSBOTTOM**

```
<html>
<head>
        <title>
                Welcome to Java Applets
        </title>
</head>
<body>
        <center>
                <h1>Welcome to the world of Applets </h1>
        </center>
        <br/>
        <center>
                <applet
                        code = HelloJava.class
                        width=400
                        height=200
                        align=right>
                </applet>
        </center>
</body>
</html>
```


7. List HTML Tags their functions. (Oct 2010)

<html></html>	Defines the beginning and end of HTML file
<head></head>	Defines the header of an HTML document
<title></title>	Defines the title of an HTML document. The title appears in the title bar of your browser window.
<body></body>	Defines the body of an HTML document. All of the content that appears on your page goes between these two tags.
. . .	Defines section headings. <h1> is the largest section heading and <h6> is the smallest section heading.
	Text between these tags will be displayed as a paragraph. Most browsers separate paragraphs with a blank line.
	The text will usually be displayed in **bold.**
	The text will usually be displayed in **italics**
<u></u>	The text will usually be displayed with **underline**
	Defines a hyperlink. Text between these tags will be highlighted. This highlighted text is a link to another web page.
 	It is used to change the face, color & size of text.
<img src="/somepicture.gif">	Inserts a picture into the document
<!-- This is a comment. -->	Inserts a comment (i.e., text not interpreted or displayed by the web browser)

8. Develop an applet to find sum two numbers.

```
import java.awt.*;
import java.applet.*;
public class UserIn extends Applet
{
        TextField text1, text2;
        public void init()
        {
                text1=new TextField(8);
                text2=new TextField(8);
                add (text1);
                add (text2);
                text1.SetText ("0");
                text2.SetText ("0");
        }
```

```
    public void paint(Graphics g)
    {
        int x=0, y=0, z=0;
        String s1, s2, s;
        g.drawString("Input a number in each box",10,15);
        try
        {
            s1=text1.getText();
            x= Integer.parseInt (s1);
            s2=text2.getText();
            y= Integer.parseInt (s2);
        }
        catch (Exception ex) { }
        z= x + y;
        s= String.valueOf (z);
        g.drawString ("The Sum is: ",10,75);
        g.drawString ("s ",100,75);

    }
}
```