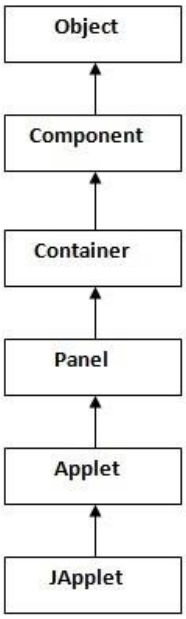
Unit 1: The Applet Basics

- Applet is a 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, and an applet class will not define main().
- Applets are designed to be embedded within an HTML page.
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
- A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.

Hierarchy Of Applet



Two Types of applets

1. AWT –based applet :
   → Uses the AWT (Abstract Window Toolkit) to provide the GUI (or no GUI at all)
   → Available since java was created.
2. Swing –based applet:
   → Uses swing class JApplet, which inherits Applet
   → Swing Applets use the swing class to provide the GUI

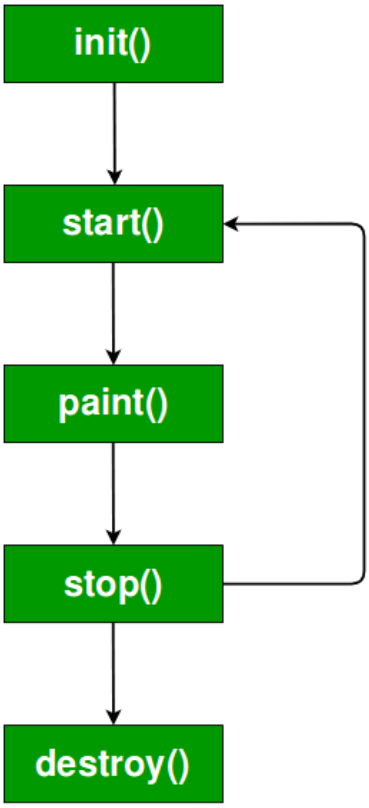
Swing offers a richer and often easier to use user interface than AWT. So, swing base applets are more popular.

Applet Basics

- 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.
- All applets are sub-classes (either directly or indirectly) of java.applet.Appletclass.
- Output of an applet window is not performed by **System.out.println()**. Rather it is handled with various AWT methods, such as **drawString().**
- In general, execution of an applet does not begin at main() method.

An Applet Skeleton

→ An applet has following methods (Applet life cycle methods)



→ When an applet begins, **init(),start()** and **paint()** methods are called.

→ When an applet is terminated, **stop()** and **destroy()** method are called.

→ paint() is method is declared in Component class. Other methods are the methods of Applet class.

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.
   → 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.
4. **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.
5. **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( ).

Note: Default implementation for init(), start(), stop() and destroy() are provided. Applets do not need to override the methods they do not use. AWT-based applets often override the paint method.

These five methods can be assembled into the skeleton shown here:

```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/

public class AppletSkel extends Applet {
  // Called first.
  public void init() {
    // initialization
  }

  /* Called second, after init().  Also called whenever
     the applet is restarted. */
  public void start() {
    // start or resume execution
  }

  // Called when the applet is stopped.
  public void stop() {
    // suspends execution
  }

  /* Called when applet is terminated.  This is the last
     method executed. */
  public void destroy() {
    // perform shutdown activities
  }

  // Called when an applet's window must be restored.
  public void paint(Graphics g) {
    // redisplay contents of window
  }
}
```

Although this skeleton does not do anything, it can be compiled and run. When run, it generates the empty window when viewed with appletviewer.

The HTML APPLET Tag

The APPLET tag of HTML is used to start an applet either from a web browser or an applet viewer. The HTML tag allows a Java applet to be embedded in an HTML document.

Syntax:

```
< APPLET
  [CODEBASE = codebaseURL]
  CODE = appletFile
  [ALT = alternateText]
  [NAME = appletInstanceName]
  WIDTH = pixels HEIGHT = pixels
  [ALIGN = alignment ]
  [VSPACE = pixels] [HSPACE = pixels]
>
[< PARAM NAME = AttributeName VALUE = AttributeValue>]
[< PARAM NAME = AttributeName2 VALUE = AttributeValue>]
. . .
[HTML Displayed in the absence of Java]
</APPLET>
```

CODEBASE : CODEBASE is an optional attribute that specifies the base URL of the applet code, which is the directory that will be searched for the applet's executable class file (specified by the CODE tag). The HTML document's URL directory is used as the CODEBASE if this attribute is not specified.

CODE: CODE is a required attribute that gives the name of the file containing your applet's compiled .class file. This file is relative to the code base URL of the applet, which is the directory that the HTML file was in or the directory indicated by CODEBASE if set.

ALT: The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser recognizes the APPLET tag but can't currently run Java applets.

NAME: NAME is an optional attribute used to specify a name for the applet instance. Applets must be named in order for other applets on the same page to find them by name and communicate with them. To obtain an applet by name, use **getApplet( ),** which is defined by the **AppletContext** interface.

WIDTH and HEIGHT: WIDTH and HEIGHT are required attributes that give the size (in pixels) of the applet display area.

ALIGN: ALIGN is an optional attribute that specifies the alignment of the applet. This attribute is treated the same as the HTML IMG tag with these possible values: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.

VSPACE and HSPACE : These attributes are optional. VSPACE specifies the space, in pixels, above and below the applet. HSPACE specifies the space, in pixels, on each side of the applet. They're treated the same as the IMG tag's VSPACE and HSPACE attributes.

PARAM NAME and VALUE : The PARAM tag allows you to specify applet specific arguments. Applets access their attributes with the **getParameter( )** method.

Other valid APPLET attributes include ARCHIVE, which lets you specify one or more archive files, and OBJECT, which specifies a saved version of the applet. In general, an APPLET tag should include only a CODE or an OBJECT attribute, but not both.

Example: Creating Hello World applet

Step 1: Creating Applet Class

```
// 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);
    }

}
```

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.

Syntax is : `void drawString(String message, int x, int y)`

Step 2: Creating HTML file which embeds the above applet.

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

</applet>
```

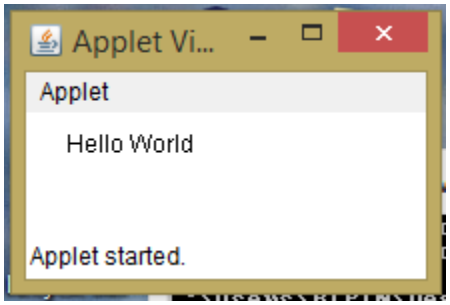
The width and height attributes specify the dimensions of the display area used by the applet. The APPLET tag contains several other options. After you create this html file, you can use it to execute the applet.

Save it as HelloWorld.html

→ We can open this file in web browser (but many browser do not support now!) or we can use appletviewer to run.

→ To run with appletviewer, in command prompt hit following command :

```
appletviewer RunHelloWorld.html
```



Passing Parameters to Applets

As just discussed, the APPLET tag allows you to pass parameters to your applet. To retrieve a parameter, use the **getParameter( )** method. It returns the value of the specified parameter in the form of a String object. Thus, for numeric and boolean values, you will need to convert their string representations into their internal formats

→ The applet tag helps to define the parameters that are to be passed to the applets

Programs to illustrate the method of passing parameters to an applet.

pp.html file contains :

```
<HTML>
<HEAD>
<TITLE>Example for Parameter passing to applet </TITLE>

</HEAD>

<BODY>

<H3>This is an example for passing parameters to Applet</H3> <p>

The parameters are<p>

message = Hello! We are students of TU <p>

p1 = 50 <p>

p2 = 50 <p>

<H5> THE TEXT PARAMETER WILL BE DISPLAYEDIN THE APPLET AT THE
 X, Y LOCATION </H5>

<APPLET code = "ParamApplet.class" WIDTH = 300 HEIGHT = 100>

<PARAM NAME = "message" VALUE = "Hello! We are students of TU">

<PARAM NAME = "p1" VALUE = "50">

<PARAM NAME = "p2" VALUE = "50">

</APPLET>

</BODY>

</HTML>
```

In this example the first parameter that is presented contains text (`Hello! We are students of TU`) whose location is specified by the other two parameters (p1 and p2).

ParamApplet.java file contains :

```
import java.applet.Applet;
import java.awt.Graphics;
public class ParamApplet extends Applet
{
        String s1, s2, msg;
        public void init(){
                //to get parametrs
                msg = getParameter("message");
                s1= getParameter("p1");
                s2 = getParameter("p2");
        }
   public void paint(Graphics g)
   {
                //to parse into integer
                int x = Integer.parseInt(s1);
                int y = Integer.parseInt(s1);
      g.drawString (msg, x, y);
   }
}
```