# APPLETS

A Java applet is a special kind of Java program that a browser enabled with Java technology can download from the internet and run. An applet is typically embedded inside a web page and runs in the context of a browser. An applet must be a subclass of the **java.applet.Applet** class. The Applet class provides the standard interface between the applet and the browser environment.

The **Applet** class is contained in the **java.applet** package.**Applet** contains several methods that give you detailed control over the execution of your applet.

In addition,**java.applet** package also defines three interfaces: **AppletContext**, **AudioClip**, and **AppletStub**.

## Applet Basics:

All applets are subclasses of **Applet**. Thus, all applets must import **java.applet**. Applets must also import **java.awt**. **AWT** stands for the Abstract Window Toolkit. Since all applets run in a window, it is necessary to include support for that window by importing java.awt package.

Applets are not executed by the console-based Java run-time interpreter. Rather, they are executed by either a Web browser or an applet viewer.

Execution of an applet does not begin at **main( ).** Output to your applet's window is not performed by **System.out.println( )**. Rather, it is handled with various AWT methods, such as **drawString( )**, which outputs a string to a specified X,Y location. Input is also handled differently than in an application.

Once an applet has been compiled, it is included in an HTML file using theAPPLET tag. The applet will be executed by a Java-enabled web browser when it encounters the APPLET tag within the HTML file.
To view and test an applet more conveniently, simply include a comment at the head of your Java source code file that contains the APPLET tag.
Here is an example of such a comment:

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

This comment contains an APPLET tag that will run an applet called **MyApplet** in a window that is 200 pixels wide and 60 pixels high. Since the inclusion of an APPLET command makes testing applets easier, all of the applets shown in this tutorial will contain the appropriate APPLET tag embedded in a comment.

## The Applet Class:

**Applet** extends the AWT class **Panel**. In turn, **Panel** extends **Container**, which extends **Component**. These classes provide support for Java's window-based, graphical interface. Thus, **Applet** provides all of the necessary support for window-based activities

## Applet Architecture:

An applet is a window-based program. As such, its architecture is different from the so-called normal, console-based programs .

First, applets are event driven. it is important to understand in a general way how the event-driven architecture impacts the design of an applet.

Here is how the process works. An applet waits until an event occurs. The AWT notifies the applet about an event by calling an event handler that has been provided by the applet. Once this happens, the applet must take appropriate action and then quickly return control to the AWT.

## An Applet Skeleton:

Four methods—**init( )**, **start( )**, **stop( )**, and **destroy( )**—aredefined by **Applet**. Another, **paint( )**, is defined by the AWT **Component** class. All applets must import **java.applet**. Applets must also import **java.awt.**

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
  }
}
```

## Applet Initialization and Termination:

It is important to understand the order in which the various methods shown in theskeleton are called. When an applet begins, the AWT calls the following methods, 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( )**

**init( ):init( )** method is called once—the first time an applet is loaded. The **init( )** method is the first method to be called. This is where you should initialize variables.

**start():**The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped(i.e start() method is called every time, the applet resumes execution).

**Paint():**The **paint( )** method is called each time your 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**.

**stop( ):**The stop() method is called when the applet is stopped(i.e for example ,when the applet is minimized the stop method is called).

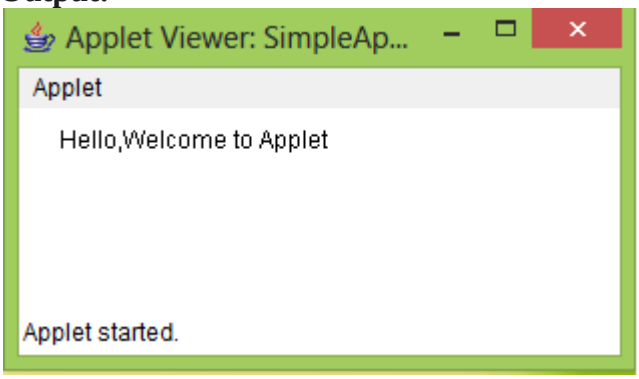
**destroy( ):**The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory(i.e destroy() method is called when the applet is about to terminate).The **stop( )** method is always called before **destroy( )**.

# Simple Applet programe:

**SimpleApplet.java**

```
import java.awt.*;
import java.applet.*;
/*
<applet code="SimpleApplet" width=300 height=100>
</applet>
*/

public class SimpleApplet extends Applet
{
   String msg="";
   // Called first.
  public void init()
  {
     msg="Hello";
  }
   /* Called second, after init().
   Also called       whenever  the applet is restarted. */
  public void start()
  {
     msg=msg+",Welcome to Applet";
  }

  // whenever the applet must redraw its output, paint( ) is called.
  public void paint(Graphics g)
  {

     g.drawString(msg,20,20);
  }
}
```

**Output:**



# How To Run an Applet Programe:

There are two ways in which you can run an applet:
■ 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 window. This is generally the fastestand easiest way to test your applet.

**Using an applet viewer to run applet(demonstrates you to run SimpleApplet.java):**

Place the applet tag in comments in java source code.
Note:Code attribute value must be equal to name of class which extends Applet class.

Compiling: javac SimpleApplet.java
Run: AppletViewer SimpleApplet.java

**Executing the applet within a Java-compatible Web browser(demonstrates you to run SimpleApplet.java):**

Compiling: javac SimpleApplet.java
Create an Html file and embeded Applet tag in html file.
**Attributes in applet tag:**
**Code(attribute):**specify name of applet class to load into browser.
**Width(attribute):**width of an applet.
**Height(attribute):**height of an applet.

SimpleApplet.html

```
<html>
  <body>
    <applet code="SimpleApplet" width=300 height=100></applet>
  </body>
</html>
```

When you open SimpleApplet.html , SimpleApplet.class applet is loaded into browser.

Note: The Browser must be java enabled to load applet programe.

## Simple Applet Display Methods:

As we've mentioned, applets are displayed in a window and they use the AWT to perform input and output.To output a string to an applet, use **drawString( )**, which is a member of the **Graphics** class.Graphics class is defined in **java.awt** package.

**void drawString(String *message*, int *x*, int *y*)**

Here, *message* is the string to be output and x and y are x-coordinate ,y-coordinate respectively. In a Java window, the upper-left corner is location 0,0.

To set the background color of an applet's window, use **setBackground( )**. To set the foreground color (the color in which text is shown, for example), use **setForeground( )**. These methods are defined by **Component**, and they have the following general forms:

**void setBackground(Color *newColor*)**
**void setForeground(Color *newColor*)**

Here, *newColor* specifies the new color. The class **Color** defines the constants shown here that can be used to specify colors:

Color.black Color.magenta
Color.blue Color.orange
Color.cyan Color.pink
Color.darkGray Color.red
Color.gray Color.white
Color.green Color.yellow
Color.lightGray

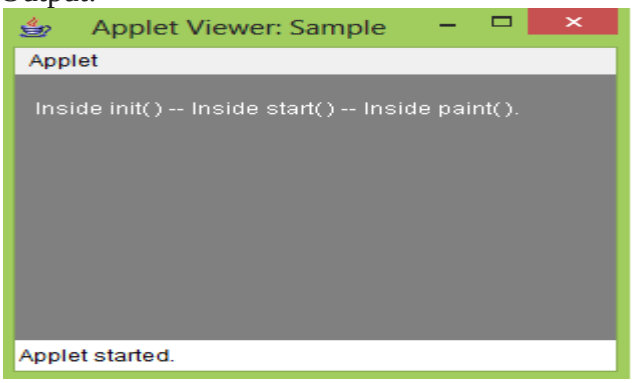
For example, this sets the background color to green and the text color to red:

**setBackground(Color.green);**
**setForeground(Color.red);**

Sample.java

```
/* A simple applet that sets the foreground and background colors and outputs a string. */
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample" width=300 height=200>
</applet>
*/
 public class Sample extends Applet
 {
    String msg;
      public void init()
    {
      setBackground(Color.gray);
      setForeground(Color.white);
      msg = "Inside init( ) --";
    }
    // Initialize the string to be displayed.
    public void start()
    {
      msg += " Inside start( ) --";
    }
    // Display msg in applet window.
   public void paint(Graphics g)
   {   msg += " Inside paint( ).";
       g.drawString(msg, 10, 30);
   }
}
```

Output:




## Requesting Repainting:

Whenever your applet needs to update the information displayed in its window, it simply calls **repaint( )**.The **repaint( )** method is defined by the AWT. It causes the AWT run-time system to execute a call to your applet's **update( )** method, which, in its default implementation, calls **paint( )**.

The simplest version of **repaint( )** is shown here:

**void repaint( )**

This version causes the entire window to be repainted. The following version specifies a region that will be repainted:

**void repaint(int *left*, int *top*, int *width*, int *height*)**

Here, the coordinates of the upper-left corner of the region are specified by *left* and *top*, and the width and height of the region are passed in *width* and *height*. These dimensions are specified in pixels. You save time by specifying a region to repaint.

The other two versions of repaint():

**void repaint(long *maxDelay*)**

**void repaint(long *maxDelay*, int *x*, int *y*, int *width*, int *height*)**

Here, *maxDelay* specifies the maximum number of milliseconds that can elapse before **update( )** is called.

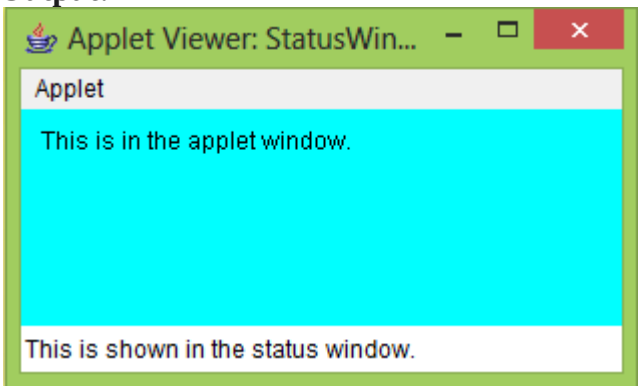
## Using the Status Window:

In addition to displaying information in its window, an applet can also output a message to the status window of the browser or applet viewer on which it is running. To do so, call **showStatus( )** with the string that you want displayed.

```
// Using the Status Window.
import java.awt.*;
import java.applet.*;
/*
<applet code="StatusWindow" width=300 height=300>
</applet>
*/
public class StatusWindow extends Applet
{
  public void init()
  {
```

```
    setBackground(Color.cyan);
  }
  // Display msg in applet window.
  public void paint(Graphics g)
  {
    g.drawString("This is in the applet window.", 10, 20);
    showStatus("This is shown in the status window.");
  }
}
```

**Output:**




## The HTML APPLET Tag:

```
< 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>
```

Let's take a look at each part now.

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).
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 understands 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.

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.The 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.

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

## Passing Parameters to Applets:

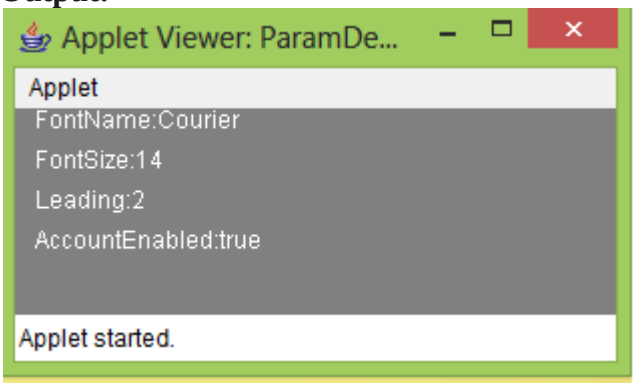
The APPLET tag in HTML allows you to pass parameters to your applet. To retrieve a parameter, use the **getParameter( )** method. It returns the value of the specified parameter in the form of a **String** object. Here is an example that demonstrates passing parameters:

ParamDemo.java

```
import java.awt.*;
import java.applet.*;
/*
<applet code="ParamDemo" width=300 height=300>
<param name=fontName value=Courier>
<param name=fontSize value=14>
<param name=leading value=2>
<param name=accountEnabled value=true>
</applet>
*/

public class ParamDemo extends Applet
{
   String fn=null,fz=null,l=null,ae=null;
   public void init()
   {
     setBackground(Color.gray);
     setForeground(Color.white);
   }
```

```
    public void start()
    {
      fn=getParameter("fontName");
      fz=getParameter("fontSize");
      l=getParameter("leading");
      ae=getParameter("accountEnabled");
      repaint();
    }

    public void paint(Graphics g)
    {
       g.drawString("FontName:"+fn,10,10);
       g.drawString("FontSize:"+fz,10,30);
       g.drawString("Leading:"+l,10,50);
       g.drawString("AccountEnabled:"+ae,10,70);
    }

}
```

**Output:**




## getDocumentBase( ) and getCodeBase( ):

Java will allow the applet to load data from the directory holding the HTML file that started the applet (the *document base*) and the directory from which the applet's class file was loaded (the *code base*). These directories are returned as **URL** objects by **getDocumentBase( )** and **getCodeBase( )**.

Bases.java

```
import java.awt.*;
import java.applet.*;
import java.net.*;
/*<applet code="Bases" width=300 height=50></applet>*/
public class Bases extends Applet
{
  // Display code and document bases.
   public void paint(Graphics g)
  {
     String msg;
     //URL class is defines in java.net package.
     URL url = getCodeBase(); // get code base
```

```
    msg = "Code base: " + url.toString();
    g.drawString(msg, 10, 20);
    url = getDocumentBase(); // get document base
    msg = "Document base: " + url.toString();
    g.drawString(msg, 10, 40);
  }
}
```

**Sample output from this program is shown here:**


## AppletContext and showDocument( ):

To allow your applet to transfer control to another URL, you must use the **showDocument( )** method defined by the **AppletContext** interface.

ACDemo.java

```
import java.awt.*;
import java.applet.*;
import java.net.*;
/*<applet code="ACDemo" width=300 height=50></applet>*/
public class ACDemo extends Applet
{
  URL u;
  public void start()
  {
    AppletContext ac = getAppletContext();
    URL url = getCodeBase(); // get url of this applet
    try
    {
     u=new URL(url+"Test.html");
     ac.showDocument(u);
    }
    catch(MalformedURLException e)
    {
     showStatus("URL not found");
    }
  }
}
```

**Explanation:**In this programe,the control is transfered from ACDemo.class (i.e applet) to Test.html file.

## Two Types of Applets:

There are two varieties of applets. The first are those based directly on the **Applet** class. These applets use the Abstract Window Toolkit (AWT) to provide the graphic user interface (or use no GUI at all). This style of applet has been available since Java was first created.

The second type of applets are those based on the Swing class **JApplet**. Swing applets use the Swing classes to provide the GUI. Swing offers a richer and often easier-to-use user interface than does the AWT. Thus, Swing-based applets are now the most popular. **JApplet** inherits **Applet**, all the features of **Applet** are also available in **Japplet**.