The following example is made simple enough to illustrate the essential use of Java applets through its java.applet package.

**Example:**

import java.awt.\*;

import java.applet.\*;

public class SimpleApplet extends Applet

{

public void paint(Graphics g)

{

g.drawString("My First Applet",40,40);

}

}

• Save the file as **SimpleApplet.java**

• Compile the file using **javac SimpleApplet.java**

Here is the illustration of the above example,

• In the first line we imports the Abstract Window Toolkit(AWT) classes as Applet interact with the user through the AWT, not through the console – based I/O classes. The AWT contains support for a window based graphical interface.

• In the second line we import the Applet package, which contains the class “Applet”. As every applet that we create is the subclass of Applet.

• The next line declares the class SimpleApplet. This class must be declared in public, because it will be accessed by code that is outside the program.

• Inside simpleApplet, paint() method is declared. This method is defined by the AWT and must be overridden by the Applet. Method paint() is called each time that the applet must redisplay its output.

This paint() method has parameter of type “ Graphics”. This parameter contains 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.

• Inside paint() method is a call to drawstring(), which is a member of the Graphics class. This method output a String beginning at specified X, Y locations. How to run an Applet?

HOW TO RUN AN APPLET?

• There are two ways in which one can run an applet, as follows

1) Executing the applet within a java-compatible web browser.

2) Using an applet viewer, such as the standard SDK tool, “appletviewer”. An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet.

• To execute an applet in a web browser, you need to write a short HTML text file that contains the appropriate APPLET tag.

For above example it is

<html>

<body>

<applet code="SimpleApplet.class" width=200 height=100>

</applet>

</body>

</html>

• Save this code in text file with extension .html say

Myapplet.html.

• Compile the file using javac SimpleApplet.java

• On successful compilation of SimpleApplet.java file, execute the this file using appletviewer Myapplet.html or just open this html file directly.

Insted of creating different text file for html code one can write above program as follows

import java.awt.\*;

import java.applet.\*;

/\* <applet code="SimpleApplet" width=200 height=100>

</applet>

\*/

public class SimpleApplet extends Applet

{

public void paint(Graphics g)

{

g.drawString("My First Applet",40,40);

}

}

• Save the file as SimpleApplet.java

• Compile the file using javac SimpleApplet.java

• On successful compilation, execute the this file using appletviewer SimpleApplet.java

The output remains same.

**Building an applet code:**

• Applet code uses the series of two classes, namely Applet and Graphics from java class library.

• Applet class which is contained in the java.applet package provides life and behavior to the applet through its methods such as init(), start(), and paint().

• 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, when it is called, actually display the result of applet code on the screen.

• 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)

• This requires that the applet code imports the java.awt package that contains the Graphics class.

• All output operations of an applet are performed using the methods defined in the Graphics class.

**APPLET TAGS**

The Applet tag is used to start an applet from both HTML document and form applet viewer.

An applet viewer will execute each Applet tag that it finds in a separate window, while web browsers like Netscape Navigator,Internet Explorer and HotJava will allow many applets in a single page.

The tag<applet…….> included in the body section of HTML file supplies the name of the applet to be loaded and tells the browser how much space the applet requires

The synatax for the standard Applet tag is as follows

Here is meaning of each piece of above code

• 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. The HTML document’s URL directory is used as the CODEBASE if this attribute is not specified. The CODEBASE if this attribute is not specified. The CODEBASE does not have to be on the host from which the HTML document was read.

• Code: code is required attribute that gives the name of the file containing the applets compiled .class file. This file is relative to the code base URL of the applet , which is the directory that the HTML file is in or the directory indicated by the CODEBASE if set.

• ALT : The ALT tag is an optional attribute used to specify a short text message that should be displayed if browser understand the APPLET tag but cant currently run java applet.

• 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 getAppet(), which is defined by the AppletContext interface.

• Param name and value : The PARAM tag allows us to specify applet specific arguments in an HTML page. Applets access their attributes with the getParameter() method.

**PASSING PARAMETERS TO APPLET**

One can supply user-defined parameters to an applet using<param…….> tag. 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. For e.g. the color of the text can be changed to red by an applet 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 tag as shown below.

<param name=text value= ”xyz” >

Passing a parameters to an applet is similar to passing parameters to main() method using command line arguments. To set up and handle parameters, we need to do two things.

1) Include appropriate tags in the HTML document.

2) Provide code in the applet to pass these parameters. Parameters are passed to an applet when it is loaded. We can define the init() method in the applet to get hold of the parameters defined in the tags. This is done using the getparameter() method, which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

**TYPES OF APPLETS**

As we can embed applet into web pages in two ways i.e. by writing our own applet and then embed into web pages. Or by downloading it from a remote computer system and then embed it into webpage.

An applet developed locally and stored in a local system is known as local applet. Therefore when webpage is trying to find local applet it doen not need the internet connection.

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 then we can download it from remote computer 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 Resourse locator(URL) and must be specified in applet’s document.

Example 1 // Example to illustrate Applet Lifecycle

import java.awt.\*;

import java.applet.\*;

/\*<applet code=”AppletTest” width=200 height=100>

</applet>

\*/

public class AppletTest extends Applet

{

public void init()

{

System.out.println("Applet Initialised...");

setBackground(Color.cyan);

}

public void start()

{

System.out.println("Applet Started....");

}

public void stop()

{

System.out.println("Applet Stoppen....");

}

public void destroy()

{

System.out.println("Applet Destryoed....");

}

public void paint(Graphics g)

{

g.drawString("Applet Text",200,400);

showStatus("This is shown in Status.");

}

}

• Save the file as AppletTest. Java

• Compile the file using javac AppletTest.java

• On successful compilation, execute the file using appletviewer AppletTest.java

Example 2 // Example to illustrate Applet Lifecycle

import java.awt.\*;

import java.applet.\*;

/\* <applet code="Sample" width=200 height= 100>

</applet>

\*/

public class Sample extends Applet

{

String msg;

public void init()

{

setBackground(Color.cyan);

setForeground(Color.red);

msg = "Inside init()-";

}

public void start()

{

msg += "Inside start()-";

}

public void paint(Graphics g)

{

msg +="Inside paint()-";

g.drawString(msg,10,30);

showStatus("This is shown at status");

}

}

•Save the file as Sample. Java

•Compile the file using javac Sample.java

•On successful compilation, execute the file using appletviewer Sample.java