CHAPTER

12

APPLETS

Chapter Includes :

- INTRODUCTION
- LOCAL & REMOTE APPLETS
- APPLET VS APPLICATIONS
- WRITING APPLETS
- LIFE CYCLE OF AN APPLET
- CREATING SOURCE CODE OF APPLET
- CREATING AN EXECUTABLE APPLET
- CREATING APPLET TAG
- ADDING APPLET TAG TO HTML
- RUNNING THE APPLET
- DETAILED FORM OF APPLET TAG
- PASSING PARAMETERS TO APPLET
- ALIGNING THE DISPLAY
- HTML TAGS
- GETTING INPUT FROM USER

INTRODUCTION

Applets are small programs that are run on Internet and transported over Internet. They are installed automatically and always run as a part of web document. Applet requires Java Runtime Environment with them. They are popular because of security feature provided by them. Applets can't access user system resources and they can't even do file handling. That's why, they contains no malicious infections with them and hence do not harm our computers.

Applets interact with the user through AWT (Abstract Window Toolkit, provides support for window-based Graphical User Interface) and not using console based Input/Output .Classes.

Java defines Applet class which provides support for applets. This class provides a variety of methods in order to control the execution of Applets. Applet class is itself a sub-class of Panel. Let's have a look on hierarchy of classes:

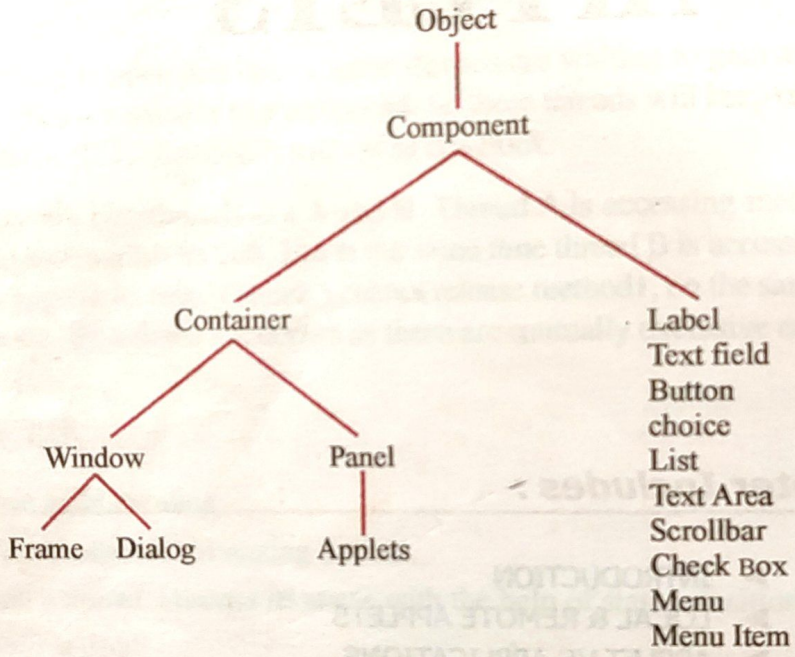


Figure : Hierarchy of classes

So in this chapter, you will learn about applets, how to write them, how to execute them and many important concepts about them.

Let's begin with types of applets :

- Local Applets
- Remote Applets

LOCAL AND REMOTE APPLETS

As you know, applets enable us to create fully interactive multimedia web documents. By the use of applets we can now create dynamic web pages.

There are two ways how we can embed applets. Either we can write our own applets or we can download them from webserver (remote computer system).

Local Applet

When we create our own applets and store them in a local system then this type of applet is termed as local applet. So, if a web page wants to access this applet, it will only need to search directories of local system and no internet is required. Web page will search the directories and on finding the applet it loads the specified applet.

Remote Applets

When an applet is created by someone else and we can access it only when we are connecting with Internet. This type of applet which is stored on remote computer, is known as remote applet. We can download the remote applet from webserver to client system (our system) provided internet connection is there and can execute it. For searching a particular remote applet, we are required to add its web address (known as URL- Uniform Resource Locator) into the applet's HTML document as the value of CODEBASE attribute (You will study about this shortly in this chapter).

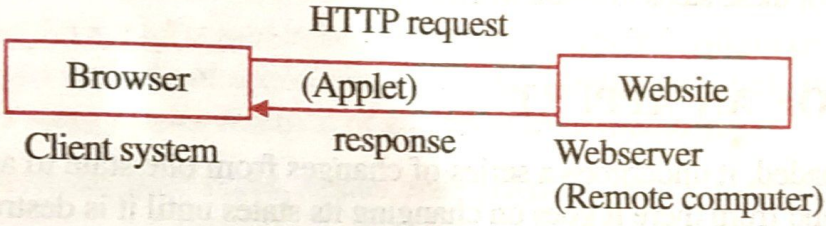


Figure : Loading Remote Applet

In case of local applet, you can specify CODEBASE as name of local directory if HTML file and applet reside in different directories. If both resides in same directory then you can omit specifying CODEBASE attribute..

APPLETS VS APPLICATIONS

As you know, Java provides us two ways to write a program. i.e. writing applets and writing application programs. An application program is a simple Java program that runs on our computer under control of operating system of the computer. While applets are applications that are designed in order to use for internet and executed by Java - compatible web browser. These two applications of Java differ in some aspects; such as:

- Applets do not use main( ) method for initiating execution of the code, while application programs cannot be executed if they not contain main( ) method.
- If we want to run an application program, then we can simply run it by invoking its members with the help of its class object in main( ) method. While if we want to run applets, then we required to create a HTML document that initiates its execution.
- Applets cannot take input from user by using console input/output classes, instead they interact with users through AWT.
- Applets are restricted to communicate with other servers on the networks. They are not supposed to run any program from local computer.
- Applets cannot use libraries of other languages such as C or C++.

WRITING APPLETS

When we write a simple Java program, then we follow some structure (defined in Chap2) i.e. we create class, define methods and variables, then we create main method, create objects and invoke methods in order to execute them. Similarly, for writing applets there is also a procedure. It includes following steps. :

- Creating source code of applet.
- Compiling source code of applet [Creating an executable code].
- Designing a Webpage using HTML tag.
- Preparing <APPLET> tag.
- Adding <APPLET> to HTML file.
- Running the applet by using Java compatible Web browser.

We will discuss each of these steps in detail in this chapter. But before that let's discuss the life cycle of an Applet.

LIFE CYCLE OF AN APPLET

When an applet is loaded, it undergoes a series of changes from one state to another. It starts its life from born state and from there it goes on changing its states until it is destroyed. Its life cycle and states are shown below with the help of diagram.

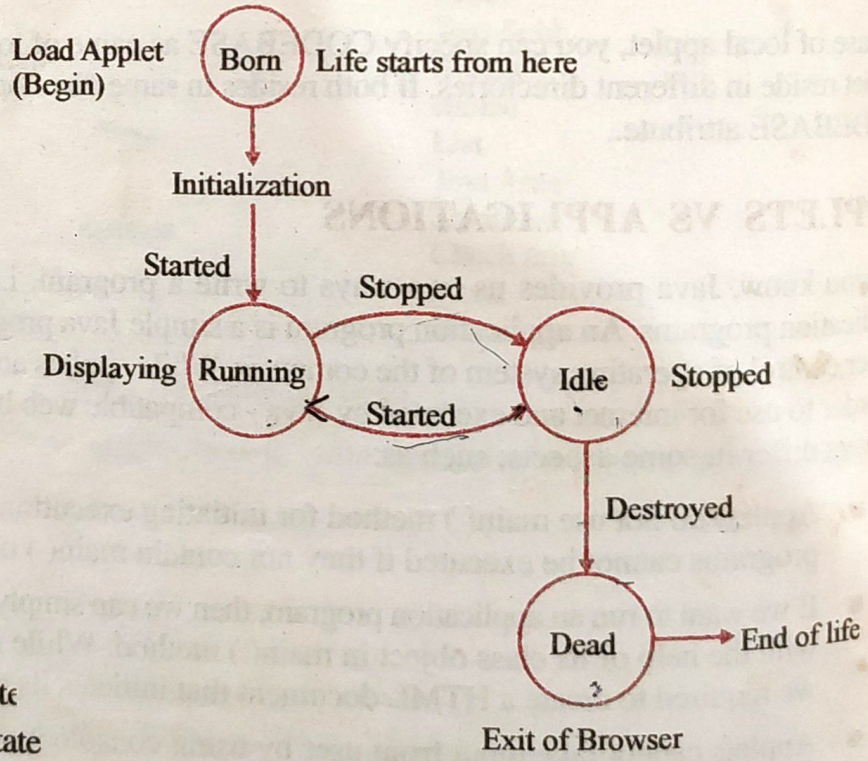


Figure : Life cycle of an Applet

So an applet encounters 5 state

1. Born or Initialization state
2. Running state
3. Display state
4. Idle or Stopped state
5. Dead state

1. **Born or Initialization state :** When we load an applet, then it is in initialization state. When we initialize the applet by declaring variables then it is born. This state occurs only once in the life cycle of an applet. At this stage , you can perform many things like setting the background color, changing the font, loading the images etc.

2. **Running state :** After reaching initialization state, it goes to running state and starts performing its work. At this stage it can be stopped for a while for some reasons. But a stopped applet can be restarted again. So this stage occurs for many times in the life cycle of an applet.

3. **Display state :** When applet has some task that is to be displayed on screen then applet enters into display state. When an applet reaches running state then it automatically switch over to display state when it has to perform some operations on screen.

4. **Idle or Stopped state :** This stage comes into focus if an applet becomes idle for sometime. It can happen when we leave the page that contains currently running applet.

5. **Dead state :** An applet is said to be in dead state when it finishes executing its work. Then it is destroyed and life of applet end here. This stage occurs only once in lifetime of an applet. This stage causes removal of an applet from memory.

To transit applets between these states, Applet class provides various methods that we will discuss in next section.

CREATING SOURCE CODE OF APPLET

As you know applets are subclasses of Applet. Applet class is contained in the package java.applet. As applets run in a window and interact with the user through the AWT, so they require support of this package also. So an applet begins with importing two important packages. They are:

1. java.awt.*;
2. java.applet.*;

Before going into detail of writing source code of Applet, let's have a look over Applet class and Abstract Window Tool kit (AWT).

Applet Class

Applet class is a superclass for all applets and is itself a subclass of Panel class. The inheritance hierarchy that it follows, is:

Object
↓
Component
↓
Container
↓
Panel
↓
Applet

So Applet class contains all features of its superclasses. Applet class provides various methods that control the execution of an applet. It includes various methods which are used to initialize an applet, start an applet, stop or start an applet. Let's discuss methods given by Applet class briefly.

- **void init( )** – This method is called when an applet begins execution. This method is used for initialization. It is the first method which is called when an applet begins its execution.
- **void start( )** – This method is called by browser automatically after init( ). This tells when an applet should start execution.
- **void stop( )** – This method is called by browser to stop execution of the applet. This is used to suspend execution for some time and applet can resumed its execution by calling start( ).
- **void destroy( )** – This method is called by browser just before an applet is about to terminate. You can override this method if you want to perform some action like any cleanup prior to its destruction.
- **URL getCodeBase( )** – This method returns you URL associated with the invoking applet.
- **URL getDocumentBase( )** – This method returns the URL of the HTML document that is invoking the applet.
- **Image getImage(URL url)** – This method returns you an image object that encapsulates the image found at the location specified by url.
- **Image getImage (URL url, String imgName)** – This method returns you an image object that encapsulates the image found at location specified by url and having the name specified by imgName.
- **boolean isActive( )** – This method returns true if the applet has been started i.e. it is active. It returns false if the applet has been stopped.
- **void resize(Dimension d)** – By using this method, you can resize your applet by using dimensions specified by d Dimension is a class of java. awt package that contains two interger fields width & height.
- **void resize(int width, int height)** – This method is used to resize your applet by specifying width & height of applet.

AWT

The main purpose of AWT is to support applet window. It provides lots of classes and methods that help us to create and manage windows. As applets are event driven, i.e. an applet waits for an event to occur. AWT helps applet by notifying it that event has occured by calling an event handler (provided by applet) Once AWT notifies about event, applet performs its task and return control to AWT. So AWT servers as very important aspect to applets. Some commonly used classes that AWT provides are :

- **BorderLayout** – Border Layout manager. This class uses 5 components North, South, East, West & Center.
- **Button** – This class is used to create push button controls.
- **Checkbox** – This class is used to create check box controls
- **Choice** – To create pop-up list.
- **Color** – To manage colours.

- **Event** – Encapsulates event.
- **Font** – Encapsulates a type font.
- **Frame** – Creates a window that includes title bar, resizes corners and a menu bar.
- **Graphics** – This class is used to add graphics.
- **Image** – To add images.
- **Label** – To add label that displays string.
- **Menu** – To create a pull-down menu.
- **Menubar** – To create a menubar.
- **List** – To create a list.
- **Text Area** – It creates a multiline edit control.
- **Text Field** – It creates a single line edit control.
- **Window** – It creates a window with no frame, no menu bar and no title.

So now it's time to discuss how to write source code of an applet. Mostly when we write an applet, it overrides 5 methods. These are :

init( )
start( )
stop( )
destroy( )
paint( )

Let's look more closely at these methods.

- **init( )** – This method is used to initialize variables. When an applet is loaded, this is the first method to be called. This method is called only once during the runtime of applet.

General form :

```
void init( )
  {
      Write code here
  }
```

- **start( )** – This method is called automatically after init( ). This method is used to start execution of applet. This method is used to resume an applet after it has been stopped. This method is called each time when you want to restart your applet.

General form :

```
void start( )
  {
      Write code here
  }
```

- **paint( )** – This method is called when an applet begins its execution. This method is used to redraw output of an applet.

General form :

```
void paint (Graphics g)
{
    Write code here
}
```

Here paint contains one parameter g of type Graphics. This g will contain graphic context which describes the graphics environment in which the applet is running.

- **stop( )** – stop( ) is called to suspend applet for some time. Sometimes it can happen that a browser leaves the HTML document containing the applet. When it goes to another page, then at that time browser calls stop( ) method of applet. When browser comes back, then it resumes the execution of applet by calling start( ).

General form :

```
void stop( )
{
    Write code here
}
```

- **destroy( )** – This method is called when we want to remove applet from memory. stop() method is always called before destroy( ) method.

General form :

```
void destroy( )
{
    Write code here
}
```

Let's see program structure to write source code.

```
import java.awt.*;
import java.applet.*;
public class appletname extends Applet
  {
    public void init( )
    {
        initialization
    }
        public void start( )
    {
        start or resume execution
    }
        public void paint (Graphics g)
```

```
    {
        Display contents
    }
        public void stop( )
    {
        suspend execution
    }
        public void destroy( )
    {
        to perform destruction of applet
    }
}
```

Program : Creating a simple applet.

```
import java.awt.*;
import java.applet.*;
public class FirstApplet extends Applet
{
    Button b1;
    public void init( )
    {
        b1 = new Button ("Hello");
        add (b1);
        setBackground(Color.pink);
    }
}
```

add this element to the applet → It refers to the current applet

First two lines begin with importing two important packages awt and applet. Second line declares the class named as FirstApplet. This class is extending Applet, so Applet class is super class of First Applet. This class is given access as public, as this class will be used outside the program. Inside FirstApplet, we create a variable b1 of Button type. Then we are calling init( ) method, in which we are allocating memory to b1 and assigning it value "Hello". Then we are adding b1 to frame (As stated earlier, applet runs under frame). Button & Frame classes are classes provided by AWT. Then we are calling method setBackground( ). This method is defined by AWT to set the colour of background. So the statement setBackground (color.pink) will set the background colour as pink.

This applet will be saved as name FirstApplet.java.

So after writing source code let's see how to create executable file of applet.

CREATING AN EXECUTABLE APPLET

Creating an executable applet means compiling the source code of applet and creating class file. An applet is compiled in the same way we compile an application program.

So in order to compile applet that was discussed in last example we will write as :

javac FirstApplet.java

FirstApplet is the name of applet.

Designing a Webpage

As we know, applets are small Java programs that reside on webserver. So in order to execute an applet in a web browser, we should develop webpage that will refer to Applet.

In order to design webpage, we will use HTML. HTML is a markup language used to develop webpages. For creating a HTML file we will follow the following structure :

HTML structure or section of Webpage

```
<HTML>
< !
    comment      } comment section
>
<HEAD>
    <TITLE>
        Write title here      Head Section
    </TITLE>
</HEAD>
<BODY>
    Applet tag will be added here      Body Section
</ BODY>
</ HTML>
```

A HTML document begins with <HTML> and ends with </HTML>. It contains three section.

- Comment Section
- Head Section
- Body Section

In the comment section we include comments. Comments are added in order to make a program easy to understand. Comments are used to describe nature of program, explain meaning of some methods. Comment in HTML file begins with < ! and end with>. All lines written inside these symbols are ignored by browser. This section is optional.

Head section is used to give title to our program. This section begins with <HEAD> and end with </HEAD>. If we want to specify a title to our HTML file (webpage) then we enclosed title in the tags <TITLE> and </ TITLE>. The text written in <TITLE> and </TITLE> bar is displayed on the title bar of webpage. This section is also optional.

Body section includes the body of HTML file. This includes all information that will be displayed on webpages. This section can also be used to design our web page on the screen. In this section, you can define color, font, images, sound anything that HTML can support. This section also includes applet code which causes reference to specified applet.

The HTML file is saved as .html extension. HTML is not case-sensitive means you can write this file in any case you want.

Program : Showing creating of simple web page.

```
<! This prgram shows creation of simple web page>
<HEAD>
   <TITLE>
      SIMPLE WEBPAGE
   </TITLE>
</HEAD>
<BODY bgcolor = "Green">
   <H1>
      WEBPAGE USING HTML
   </H1>
   </BODY>
</HTML>
```

On running this file browser, it will look like :

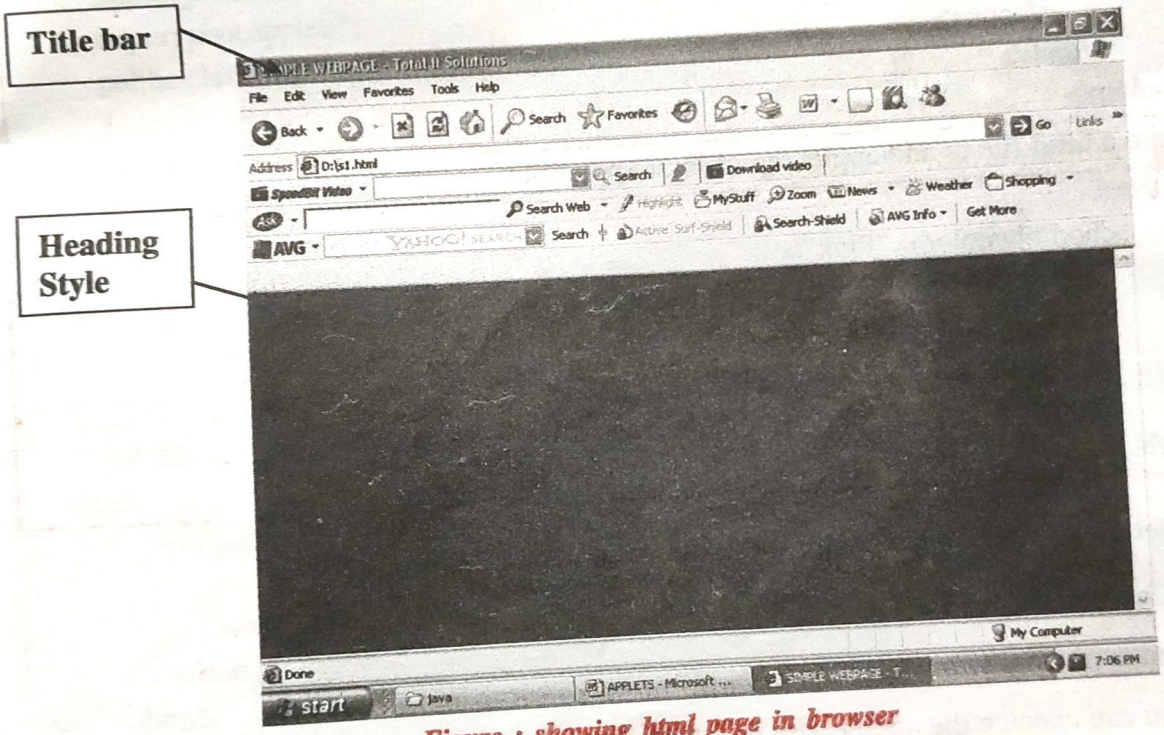


Figure : showing html page in browser

CREATING APPLET TAG

In order to execute an applet in a web browsers you need to create applet tag. An applet tag starts from <applet> and ends with </applet>. Inside this tag name of applet, width and height of applet are specified. The applet tag for FirstApplet file is created as :

```
<applet code = "FirstApplet.class" width = 100 height = 60>
</applet>
```

This tells that name of applet is FirstApplet and its height is 60 pixels and width is 100 pixels.

So in applet tag three things are included :

1. Code → Name of Applet file
2. Width → Width of Applet
3. Height → Height of Applet

Width and height determine of what size applet will appear in browser.

ADDING APPLET TAG TO HTML FILE

Now we can put the Applet tag inside the HTML file in order to execute the applet. We can insert Applet tag at any place in body section.

General form :

```
<html>
    <body>
        <applet>
    </applet>
  </body>
</html>
```

Let's write a html file to add applet tag of FirstApplet file.

```
<html>
  <body bgcolor = "Pink">
    <applet code = "FirstApplet.class"
        height = 100 width = 70>
    </applet>
  </body>
</html>
```

Now applet tag has been added to html file. Name this file first.html.

RUNNING THE APPLET

We can run applets in two ways :

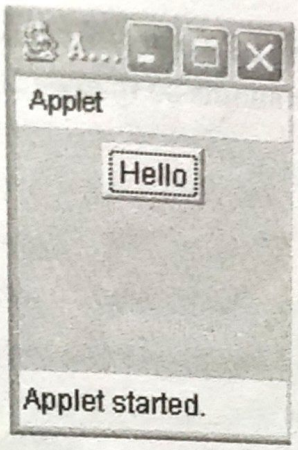
(a) You can execute the applet with in a Java - compatible web browser. When you execute a web page in Java - compatible web browser, then you will be able to see the entire web page that you have designed.

(b) By using an applet-viewer : When we use an applet - viewer then output of applet is shown. This tool ignores all other tags of HTML except applet and shows the output of applet. But it is the simplest way to run applets.

So, for example in order to run preceding HTML file named as first.html, we will write as :

appletviewer first.html

So it will display a window like:



Program : Displaying welcome on applet screen

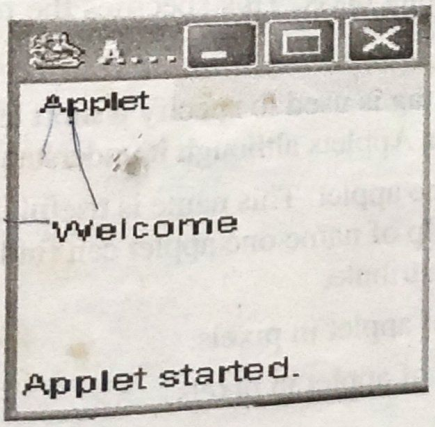
Applet class file

```
import java.awt.*;
import java.applet.*;
public class WelApplet extends Applet
{

    public void paint( Graphics g)
    {
        g.drawString("Welcome",10,50);
    }
}
```

Html file for WebApplet

```
<html>
    <applet code = "WelApplet.class"
            height = 60 width = 200>
        </applet>
    </body>
</html>
```

Output :



DETAILED FORM OF APPLET TAG

All the programs we have made till now include simple form of Applet tag. It only tells the name of applet class and its size. But actual syntax of applet tag contains so many attributes. Let's have a detailed look on syntax of Applet Tag.

```
<APPLET
    [CODEBASE = codebase URL]
    CODE = Appletclassname.class
    [ALT = alternate Text]
    [NAME = applet Instance name]
    WIDTH = pixels
    HEIGHT = pixels
    [ALIGN = alignment]
    [VSPACE = pixels]
    [HSPACE = pixels]
                    >
    [<PARAM NAME = Attribute name1 VALUE = value 1>]
    [<PARAM NAME = Attribute name2 VALUE = value 2>]
    [<PARAM NAME = Attribute name3 VALUE = value 3>]
    ..............................................
    ..............................................
    </APPLET>
```

All attributes written in [ ] are optional i.e. applet tag provides you facility to define them but if you don't want to use them then it will not cause any error. Three attributes that are always required are :

CODE = Appletclassname.class

WIDTH = Pixels

HEIGHT = Pixels

Now, let's understand meaning of each attribute one by one.

CODEBASE = Codebase URL CODEBASE attribute provides you the address (URL) of the directory where you will find the applet. If the applet code resides in the same directory as that of HTML file, then you can omit the CODEBASE attribute. This attribute is optional.

CODE = This attribute specifies the name of applet class. This specifies the name of already compiled class and stored as .class file.

- **ALT :** This is an optional attribute. This ALT tag is used to specify a short text message that a browser will display if it is unable to run java Applets although it understand the applet tag.
- **NAME :** This tag specifies a name given to the applet. This name is useful if there are more applets running on same page. So with the help of name one applet can find other and communicate with each other. This is an optional attribute.
- **WIDTH :** This tag is used to specify width of applet in pixels.
- **HEIGHT :** This tag is used to specify height of applet in pixels.

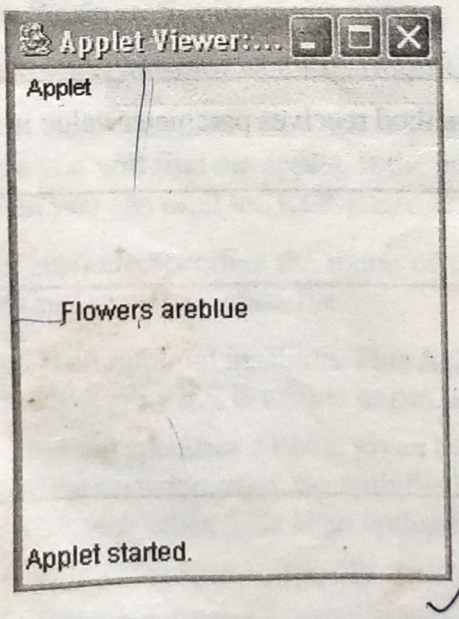
```
{
    String s1;
    public void init( )
    {
        s1 = getParameter ("Color");
        s1 = "Flowers are" + s1;
    }
public void paint (Graphics g)
    {
        g.drawString (s1, 20, 100);
    }
}
```

HTML file

```
Para.html
<HTML>
    <BODY bg Color = "red">
        <Applet Code = ParamApplet.class
            Width = 200
            Height = 200 >
        <PARAM name="color"
                value = "blue" >
        </APPLET>
    </BODY>
</HTML>
```

appletviewer Para.html

OUTPUT :



ALIGNING THE DISPLAY

This tag is used to align the applet using the ALIGN attribute. This attribute can have one of the following 9 values :

Left, Right, Top, TextTop, Middle, AbsMiddle, BaseLine, Bottom, Abs Bottom

HTML file

```
<HTML>
    <BODY>
            <APPLET CODE = ParamApplet.class
            Width = 200
            Height = 200
            Align = Top >
                    <param name="color" value="red">
    </APPLET>
</BODY>
</HTML>
```

This will cause the alignment of ParamApplet to the top.

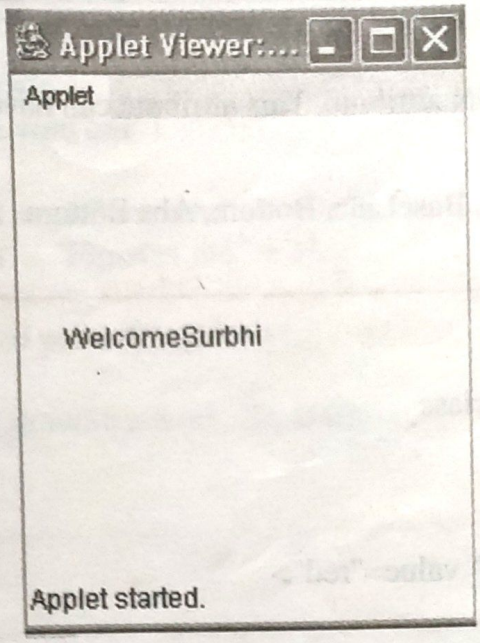
Program : Taking your name as parameter and displaying welcome your name Most

Applet Code

ParamApplet.Java

```
import java.awt.*;
    import java.applet.*;
public class ParamApplet extends Applet
{
    String s1;
    public void init( )
    {
            s1 = getParameter ("name");
            if(s1==null)
                    s1="Welcome Guest";
            else
                    s1="Welcome"+s1;
    }
public void paint(Graphics g)
    {
            g.drawString (s1,20,100);
    }
}
```

Output :



HTML file

```
Para.html
<HTML>
        <BODY bg Color = "red">
                <Applet Code = ParamApplet.class
                        Width = 200
                        Height = 200 >
                <PARAM name="name"
                        value="Surbhi" >
                </APPLET>
        </BODY>
</html>
```

HTML TAGS

Let's have a overlook of the tags supported by HTML.

<HTML>.....</HTML>	– It signifies beginning and end of HTML file.
<HEAD>.....</HEAD>	– This tag signifies beginning and end of Head section.
<TITLE>.....</TITLE>	– This is used to define title to the web page. The text enclosed between these tags appears on the title bar.
<BODY>.....</BODY>	– This tag signifies beginning and end of body text.
<H1> ..... </H1>	– This isthe heading style. There are six heading styles that are
<H6 ..... <H6>	– Defined in HTML. H1 creates the largest font and <H6> creates the smallest font.
<CENTER> ..... </CENTER>	– The text enclosed in this tag is placed at the center.
<B>	– To print the text in bold.

 	– To give a line break.
<P>	– To start a paragraph.
<IMG.SRC>	– To declare attributes of an image to be displayed.
<HR>	– To draw a horizontal rule.
<A> ..... </A>	– Anchor tag, used to add hyperlinks.
<FONT> ..... </FONT>	– Used to change font of the text. This tag contains attributes color and size which are used to set color and size of text.

Program : Applet displaying numerical value

Applet Code

ProductApplet.Java

```
import java.awt.*;
import java.applet.*;
public class ProductApplet extends Applet
{
    public void paint (Graphics g)
    {
        String s1;
        int v1= 20;
        int v2 = 50;
        int product = v1 * v2;
        s1 = "Product:"+ String.valueOf(product);
        g.drawString (s1,20,100);
    }
}
```

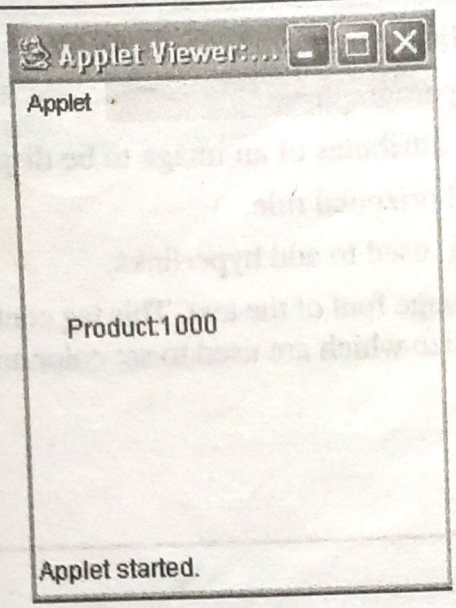
HTML file

Prod.html

```
<HTML>
    <BODY bg Color = "red">
        <Applet Code = ProductApplet.class
            Width = 200
            Height = 200 >
            </APPLET>
    </BODY>
</HTML>
```

Here numerical value product is first converted to a string by using valueOf( ) method and then it is displayed.

Output :



GETTING INPUT FROM USER

Applet treats input as text strings as they work in graphical environment. In order to get input from user, we create an graphical applet that can type and edit input items. This is done by using TextField class. So we create Textfield to receive input, then we retrieve these items from the fields. As text fields contain input in string form so they are converted to relevant form, then only they are used for calculations. Then as textfield contain data in string form, so the result produced by calculation is converted back to string form. So that text field can display it correctly.

Program : Applet receiving two numbers and displaying their product

Applet Code

ProductApplet.Java

```
import java.awt.*;
import java.applet.*;
public class ProductApplet extends Applet
  {
      TextField t1,t2;
      public void init()
        {
          t1 =  new TextField(10);
          t2 =  new TextField(10);
           add(t1);
           add(t2);
          t1.setText("0");
          t2.setText("0");
        }
```

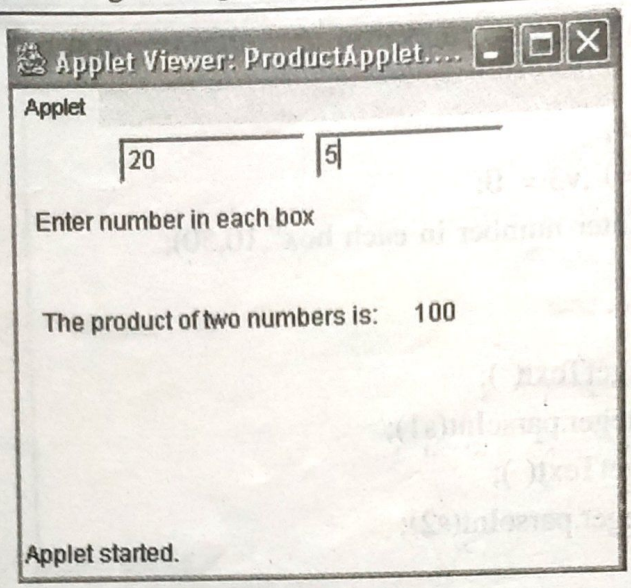
```
public void paint (Graphics g)
{
        String s1,s2,s;
        int v1= 0,v2=0 ,v3 = 0;
      g.drawString ("Enter number in each box",10,50);
       try
            {
              s1 = t1.getText( );
              v1 = Integer.parseInt(s1);
              s2= t2.getText( );
              v2= Integer.parseInt(s2);
            }
             catch ( Exception a) { }
              v3 = v1 * v2;

        s=String.valueOf(v3);
    g.drawString (" The product of two numbers is:",10,100);
       g.drawString(s,200,100);
  }
public boolean action(Event event,Object object)
{
    repaint ();
    return true;
  }
}
```

HTML file

prod.html

```
<HTML>
   <BODY bg Color = "red">
        <Applet Code = ProductApplet.class
                Width = 300
                Height = 200 >
        </APPLET>
   </BODY>
</HTML>
```

Output :



Program : Applet taking three numbers and displaying largest of three

Applet Code

LargestApplet.Java

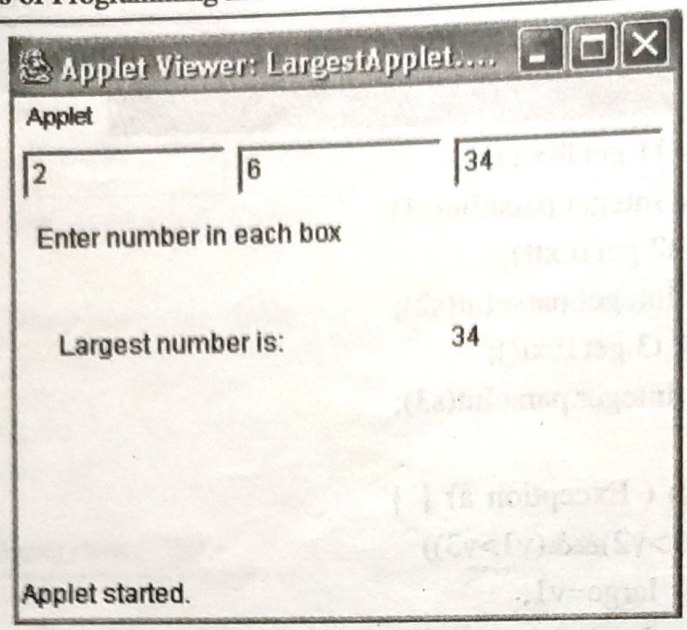
```
import java.awt.*;
import java.applet.*;
public class LargestApplet extends Applet
{
    TextField t1,t2,t3;
    public void init( )
      {
        t1 =  new TextField(10);
        t2 =  new TextField(10);
        t3 =  new TextField(10);
         add(t1);
         add(t2);
         add(t3);
         t1.setText("0");
        t2.setText("0");
        t3.setText("0");
      }
    public void paint (Graphics g)
      {
          String s1,s2,s3,s;
        int v1= 0,v2=0 ,v3 = 0,large=0;
        g.drawString("Enter number in each box",10,50);
```

```
            try
                {
                  s1 = t1.getText();
                  v1 = Integer.parseInt(s1);
                  s2= t2.getText();
                  v2= Integer.parseInt(s2);
                   s3= t3.getText();
                  v3= Integer.parseInt(s3);
                 }
                  catch ( Exception a) { }
                 if(( v1>v2)&&(v1>v3))
                        large=v1;
                  if(( v2>v1)&&(v2>v3))
                        large=v2;
                  if(( v3>v1)&&(v3>v2))
                        large=v3;
              s = String.valueOf(large);
                g.drawString("Largest number is:",20,100);
            g.drawString(s,200,100);
        }
public boolean action(Event event,Object object)
   {
        repaint ();
        return true;
   }
}
```

HTML file

```
prod.html
<HTML>
   <BODY bg Color = "red">
        <Applet Code = LargestApplet.class
              Width = 300
              Height = 200 >
        </APPLET>
   </BODY>
</HTML>
```

Output :



Program : Applet taking your name as input and displaying its length

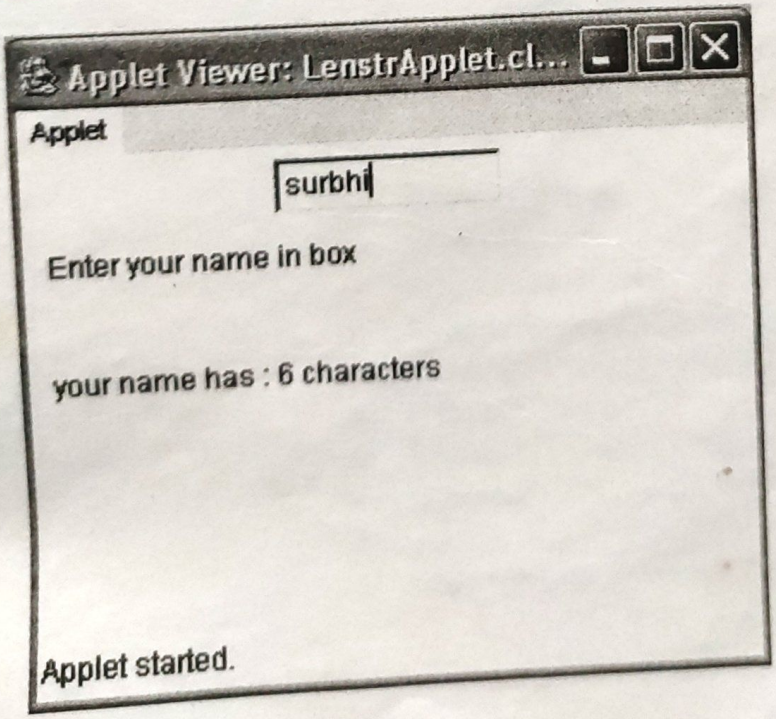
Applet Code

LenstrApplet.Java

```
import java.awt.*;
import java.applet.*;
public class LenstrApplet extends Applet
    {
        TextField t1;
        public void init( )
            {
                t1 =  new TextField(10);
                  add(t1);
                t1.setText("your name");
            }
        public void paint (Graphics g)
          {
                  String s1,s,str;
                 int v1=0;
              g.drawString ("Enter your name in box",10,50);
                 s1=t1.getText();
                  v1 = s1.length();
                s=String.valueOf(v1);
                  str="your name has : "+s+" characters";
```

```
                g.drawString(str,10,100);
        }
public boolean action(Event event,Object object)
        {
                repaint ();
                return true;
        }
}
```

HTML file

```
prod.html
<HTML>
    <BODY bg Color = "red">
        <Applet Code = LenstrApplet.class
            Width = 300
            Height = 200 >
        </APPLET>
    </BODY>
</HTML>
```

OUTPUT :