CHAPTER 30

APPLETS

When a HTML (Hyper Text Markup Language) page wants to communicate with the user on Internet, it can use a special Java program, called 'applet' to communicate and respond to the user. The user can interact by typing some details or by clicking the components available in the applet program. The program then processes and displays the results.

We can understand an applet as a Java byte code embedded in a HTML page, generally for the purpose of achieving communication with the user. We can think of an applet as:

```
Applet = Java byte code + HTML page.
```

Important Interview Question

What is an applet?

An applet represents Java byte code embedded in a web page.

Creating an Applet

To create an applet, we need to write a Java program and compile it to get byte code. Then we should embed (include) it into a HTML page on a particular location wherever we want it to be displayed. This page is then stored in the web server. A client machine communicates with the web server, the server then sends the HTML page that contains the applet. The page is then transmitted to the client where the applet is executed on the client's web browser. Thus applets are executed at client side by the web browser. Thus applets travel thousands of kilometers of distance on Internet and reach the client machines before their execution on the client.

To create an applet, we have Applet class of `java.applet` package and JApplet class of `javax.swing` package. These classes use the following methods, which are automatically run by any applet program. So, these methods should be overridden in an applet program.

- `public void init()`: This method is the first method to be called by the browser and it is executed only once. So, the programmer can use this method to initialize any variables, creating components and creating threads, etc. When this method execution is completed, browser looks for the next method: start().
- `public void start()`: This method is called after init() method and each time the applet is revisited by the user. For example, the user has minimized the web page that contains the applet and moved to another page then this method's execution is stopped. When the user

comes back to view the web page again, start() method execution will resume. Any calculations and processing of data should be done in this method and the results are also displayed.

- `public void stop()`: This method is called by the browser when the applet is to be stopped. If the user minimizes the web page, then this method is called and when the user again comes back to this page, then start() method is called. In this way, start() and stop() methods can be called repeatedly. For example, an applet with some animation might want to use the start() method to resume animation, and the stop() method to suspend the animation.
- `public void destroy()`: This method is called when the applet is being terminated from memory. The `stop()` method will always be called before destroy(). The code related to releasing the memory allocated to the applet and stopping any running threads should be written in this method.

Executing init(), start(), stop() and destroy() methods in that sequence is called 'life cycle of an applet'. Note that none of these methods are compulsory while writing an applet.

Important Interview Question

What is applet life cycle?

An applet is born with init() method and starts functioning with start() method. To stop the applet, the stop() method is called and to terminate the applet completely from memory, the destroy() method is called. Once the applet is terminated, we should reload the HTML page again to get the applet start once again from init() method. This cyclic way of executing the methods is called applet life cycle.

Please note that the 'public static void main(String args[]) ' method is not available in case of applets. This means, we can compile the applet code but we can not run it using a JVM. Now the question is, where are the applets run?

Once the applet is created, we compile and obtain its byte code. This byte code is embedded in HTML page and the page is sent to the client computer. The client machine contains a browser like Internet explorer, Netscape Navigator or Mozilla Firefox where the HTML page is viewed by the user. The same browser will execute the applet of the HTML page. The browser contains a small virtual machine called 'applet engine' which understands and runs the applet code.

Important Interview Question

Where are the applets executed?

Applets are executed by a program called applet engine which is similar to virtual machine that exists inside the web browser at client side.

Applets can show images and animation, play sounds, take user input and send it to the server ,etc. Applets cannot interact and spoil the resources of the client system and hence they are harmless. This restricted environment where the applets are executed is called 'sandbox'. The sandbox provides an applet with some amount of memory to get executed and does not allow the applet to do any illegal operations on the client system. For example:

- An applet cannot run any executable program in the client system.
- An applet cannot communicate with any server other than the server from which it was downloaded.
- An applet cannot read from a file or write into a file that belongs to the client computer.
- An applet cannot find sensitive data like user's login name, email-address, etc.

Uses of Applets

`Applets` can be used for multiple purposes. Some of the uses of applets are:

- ❑ Applets are used on Internet for creating dynamic web pages. There are two types of web pages: Static and Dynamic. Static web pages provide some information to the user but the user cannot interact with the web page other than viewing the information. Dynamic web pages interact with the user at the runtime. For example, a student can type his hall ticket number in a text field and click the retrieve button to get back his results from his University server. Applets are useful to provide such interaction with the user at runtime.
- ❑ Another use of applets is for creating animation and games where the images can be displayed or moved giving a visual impression that they are alive.

In fact, applets are one of the main reasons for the popularity of Java on Internet. In 1995, when JavaSoft people demonstrated some animation using an applet on HotJava browser, developers started believing that it is possible to perform animation in the browsers.

Important Interview Question

What is HotJava?

HotJava is the first applet-enabled browser developed in Java to support running of applets.

Now-a-days most of the websites on Internet are dependent on other softwares like PhotoShop, Flash, DreamWeaver, etc. for creating, editing, and providing animation to the images. But for validating the form data, scripting languages like JavaScript and PHP are better. Hence the use of applets is diminishing rapidly.

<APPLET> tag

<APPLET> tag is useful to embed an applet into an HTML page. It has the following form:

```
<APPLET  CODE= "name of the applet class file"
                    CODEBASE= "path of the applet class file"
                   HEIGHT= maximum height of applet in pixels
            WIDTH= maximum width of applet in pixels
            ALIGN= alignment (LEFT,RIGHT,TOP,BOTTOM,MIDDLE)
            ALT= alternate text to be displayed>

                     <PARAM NAME = parameter name VALUE= its value>
</APPLET>
```

The <PARAM> tag is useful to define a variable (parameter) and its value inside the HTML page which can be passed to the applet. The applet can access the parameter value using getParameter() method, as:

```
String value = getParameter("pname");
```

Here, pname is the parameter name and its value is retrieved by the above method into String type variable: value.

Important Interview Question

Which tag is used to embed an applet into a HTML page?

<APPLET> tag is used to insert an applet into HTML page.

A Simple Applet

Let us create an applet that displays 'Hello applet' in the app
take the help of paint() method of Component class of
methods of the applet and the applet class itself should be
available to the browser to execute.

Program 1: This is Java program that creates an apple
message "Hello Applets!".

```
        //A simple applet
        import java.awt.*;
        import java.applet.*;

        public class MyApp extends Applet
        {
              //set a background color for the frame
              public void init()
              {
                      setBackground(Color.yellow);
              }

              //display message in applet window
              public void paint(Graphics g)
              {
                      g.drawString("Hello Applets!", 5
              }
        }
```

Output:

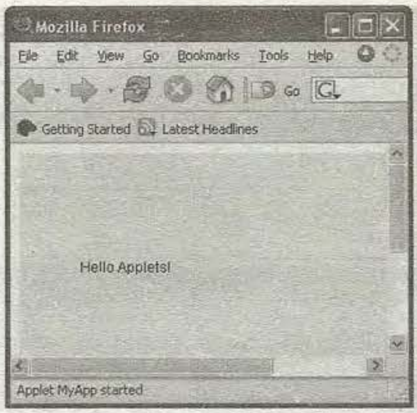
```
        C:\> javac MyApp.java
        Now, MyApp.class is created. This byte code s
using <APPLET> tag, as shown below:

        <! MyApp.html that embeds MyApp applet>
        <html>
        <applet code="MyApp.class" height=300 width=4
        </applet>
        </html>
```

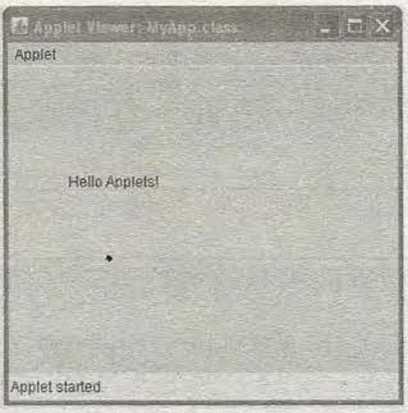
Save the above code with the name: MyApp.html. This HTM
opened in the browser, or an applet viewer supplied by the
test the applet. For this purpose, open any browser and in t
name along with the directory path. The applet opens in
system prompt as:

```
        C:\> appletviewer MyApp.html
```

to open the applet in the applet viewer.

Opening of the applet in the browser also in the applet view


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Getting Started Latest Headlines
Hello Applets!
Applet MyApp started



Let us write another applet using the methods: init(), start(), stop() and destroy() along with paint() method to track and display the execution sequence of these methods. Remember, none of these methods are compulsory for creating an applet.

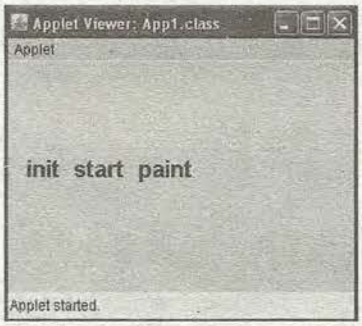
Program 2: This Java program creates an applet with some background color and foreground color with a message. The message string is stored in msg and is displayed in paint() method.

```
//applet creation
import java.awt.*;
import java.applet.*;

public class App1 extends Applet
{
     //vars
     String msg="";

     //this method is executed when an applet is loaded
     public void init()
     {
             //set backround color for applet frame
             setBackground(Color.yellow);

             //set foreground for text in frame
             setForeground(Color.red);

             //set font for text in applet
             Font f = new Font("Arial", Font.BOLD, 20);
             setFont(f);

             //store method name in msg
             msg+=" init ";
     }

     //this method is executed after init()
     public void start()
     {
             //add this method name to msg
             msg+=" start ";
     }


     //to stop the applet
     public void stop()
     {
             //add this method name to msg
             msg+= " stop ";
```

```
        }

        //to remove applet from memory
        public void destroy()
        {
                //add this metho name to msg
                msg+= " destroy ";
        }
```

Output:

```
C:\> javac App1.java
C:\>

Now, create a HTML page to embed the App1 applet into it, as shown below:

<! MyApp.html that embeds App1 applet>
<html>
<applet code="App1.class" height=200 width=300>
</applet>
</html>

Open the above MyApp.html file in the applet viewer, as:
C:\> appletviewer MyApp.html
```



```
init start paint
```

See the output of Program 2. Minimize the applet frame and you will see stop() method executed, then maximize it to see if the start() method is executed. When ever the applet frame is resized `paint()` method is again executed, thus showing the updated contents of the applet frame.

An applet with Swing Components

Let us see how to use swing components in an applet. We know how to create components using `javax.swing` package. In Program 3, we create a text field for receiving the name of the user, a text area to receive the address and a list box to receive the user selected items. Two push buttons OK and Cancel are created additionally at the bottom of the form. When the user enters his data and clicks the OK button, the data is retrieved from the form and again displayed in the text area. In actual application, the same data is sent to a server on the network where it is stored, processed and a reply to the customer is sent. When the Cancel button in the form is clicked, the data typed in the form will be cleared.

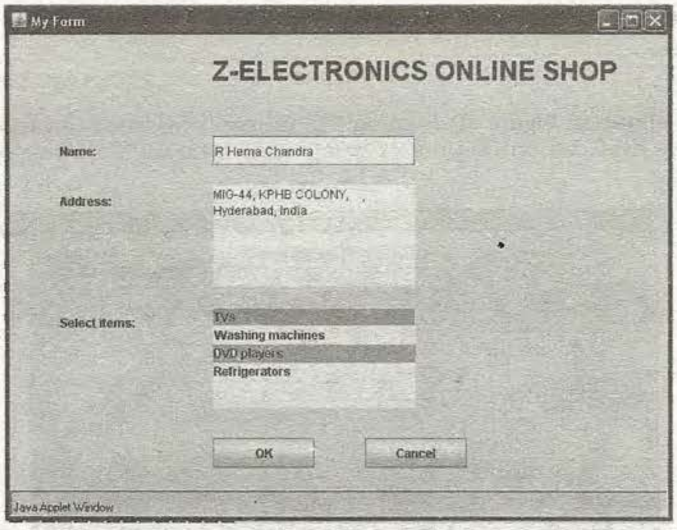
Program 3: In this program, we create a form where the user can type his details and select items according to his requirement.

```
        //Online shopping form
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyForm extends JApplet implements ActionListener
{
        //vars
        String str="", str1="", str2="";
        Object x[];
        JLabel n,a,i,lbl;
        JTextField name;
        JTextArea addr;
        JList lst;
        JButton b1,b2;
        Container c;

        public void init()
        {
                //create JFrame and container
                JFrame jf = new JFrame();
                c = jf.getContentPane();

                //display yellow background color in container
                c.setBackground(Color.yellow);

                //do not set any layout to c
                c.setLayout(null);

                //set the size and title for frame
                jf.setSize(500,400);
                jf.setTitle("My Form");

                //display the frame
                jf.setVisible(true);

                //Display heading in the frame using a label
                Font f = new Font("Dialog",Font.BOLD,26);
                lbl = new JLabel();
                lbl.setFont(f);
                lbl.setForeground(Color.red);
                lbl.setText("Z-ELECTRONICS ONLINE SHOP");
                lbl.setBounds(200,10,500,50);
                c.add(lbl);

                //TextField and a label for entering name
                n = new JLabel("Name: ", JLabel.LEFT);
                name = new JTextField(30);
                n.setBounds(50,100,100,30);
                name.setBounds(200,100,200,30);
                c.add(n);
                c.add(name);


                //TextArea and a label for entering address
                a = new JLabel("Address: ", JLabel.LEFT);
                addr = new JTextArea(5,50);
                a.setBounds(50,150,100,30);
                addr.setBounds(200,150,200,100);
                c.add(a);
                c.add(addr);


                //List box for multiple selection
                i = new JLabel("Select items: ", JLabel.LEFT);
                String[] data = {"TVs", "Washing machines", "DVD players",
                "Refrigerators"};
                lst = new JList(data);
                i.setBounds(50,270,100,30);
                lst.setBounds(200,270,200,100);
                c.add(i);
                c.add(lst);

                //add Two push buttons: OK and Cancel
                b1 = new JButton("OK");
                b2 = new JButton("Cancel");
                b1.setBounds(200,400,100,30);
                b2.setBounds(350,400,100,30);
                c.add(b1);
                c.add(b2);

                //add listeners to buttons
                b1.addActionListener(this);
                b2.addActionListener(this);
        }

        //this method is executed when the buttons are clicked
        public void actionPerformed(ActionEvent ae)
        {
                //know which button is clicked
                str = ae.getActionCommand();

                //if the button label is OK then
                if(str.equals("OK"))
                {
                        //retrieve data from text field, text area and list boxes
                        str1= name.getText()+"\n";
                        str1+= addr.getText()+"\n";
                        x = lst.getSelectedValues();
                        for(int i=0;i<x.length; i++)
                        str2 +=(String)x[i]+"\n";

                        //display the data in text area
                        addr.setText(str1+str2);

                        //make the strings empty
                        str1="";
                        str2="";
                }
                else{
                        //if Cancel button is clicked, clear the data in the form
                        name.setText("");
                        addr.setText("");
                        lst.clearSelection();
                }
        }
}
```

Output:

```
C:\> javac MyForm.java
C:\>
Now embed the byte code MyForm.class generated above in a HTML page, as shown
below:

 <! This MyForm.html contains MyForm applet>
<html>
<applet code="MyForm.class" width=400 height=400>
</applet>
</html>

 Open the above MyForm.html in an appletviewer as:
C:\> appletviewer MyForm.html
```



R Hema Chandra

MIG-44, KPHB COLONY,
Hyderabad, India

TVs
Washing machines
DVD players
Refrigerators

In the above output, the user can select any one item or several items from the list box. To select a group of items which are in sequence, the user can use `SHIFT+CLICK` on the items. To select the items randomly, the user can use `CONTROL+CLICK` on the items.

Animation in Applets

One of the uses of applets is in performing animation and developing games. Animation represents moving the objects from one place to another so that the objects look alive. To animate an object, we should first load it into Image class object, as:

```
        Image img = getImage(getDocumentBase(),"plane.gif");
```

Image class belongs to java.awt package and getImage() method belongs to Applet class. getDocumentBase() is a method of Applet class that gives the directory path where the image is located. This directory path may change because, after creating an applet the applet is loaded into a specific directory of a web server software. If the images are also loaded along with the applet in the same directory, getDocumentBase() method returns that directory path. If the image is available in some other directory, its path should replace the method. Here we have used an image by the name of 'plane.gif' that we want to load into img object.

To display the image in the applet frame, we can use drawImage() method of Graphics class, as:

```
        g.drawImage(img,x,y, obj);
```

Here, img represents the Image class object where the image is found. X and y represent the coordinates starting from where the image should be displayed. obj represents the ImageObserver object which stores the history of how the Image is loaded into memory. Since this is not needed by us, we can pass null in its place.

In program 4, we create an applet to load an image 'plane.gif' into Image object and display it in the applet frame using drawImage(). For this purpose, we can draw an image with an aero plane in blue background using paint software and save it as 'plane.gif' using the option:

- ❑ File -> Save As...

And typing the filename as "plane.gif" as shown here:

- ❑ Filename: plane
- ❑ Save as type: GIF.

The image that is shown in Figure 30.1. using GIF file can be created in various ways. You can use any of the .gif files available on Internet or create your own .gif file using softwares like Flash or Dreamweaver, etc.

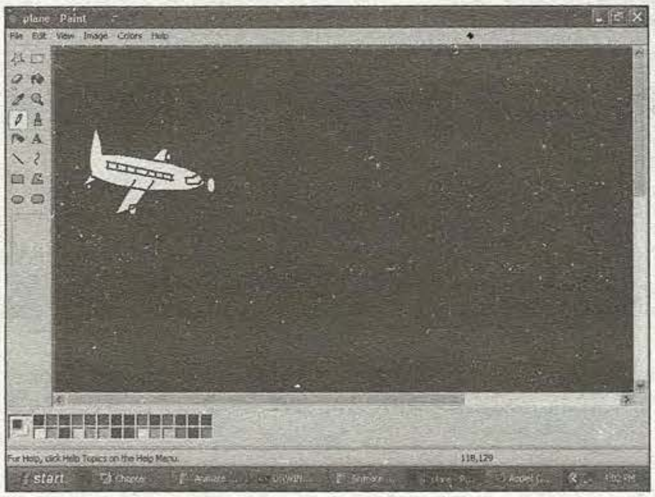

plane - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.
118,129
start

Figure 30.1 A plane image drawn in paint utility

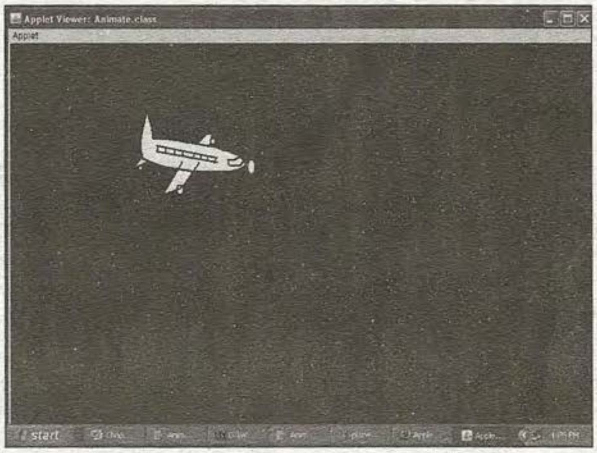
The location of the image is changed by changing the x coordinate from 0 to 800px. So the image looks as if it is moving from left to right. Between each move, we insert a time delay of 20 milliseconds using Thread.sleep() method. If this time is increased, the speed of the movement will be reduced. If the time is decreased, the speed can be increased.

Program 4: Moving an aero plane picture from left to right.

```
//Animation in applets
import java.awt.*;
import java.applet.*;
public class Animate extends Applet
{
        public void paint(Graphics g)
        {
                //load the image into Image object img
                Image img = getImage(getDocumentBase(),"plane.gif");

                //move the image from left to right by changing
                //x coordinates from 0 to 800px and take y coordinate as 0
                for(int x=0; x<800; x++)
                {
                        g.drawImage(img,x,0,null);
                        try{
                                Thread.sleep(20);//delay for 20 milliseconds
                        }catch(InterruptedException ie){}
                }
        }
}
```

Output:

```
C:\> javac Animate.java
C:\>
<!Animate.html>
<html>
<applet code= "Animate.class", height=600 width=800>
</applet>
</html>
C:\> appletviewer Animate.html
```



Here is another program which animates a group of images. First, we create four images fig1.gif, fig2.gif, fig3.gif, fig4.gif in 'Paint' that are almost similar, except the hand positions as shown in Figure 30.2. The hands of these images are drawn slightly different. Of course, the second image is repeated as fourth image. These images can be displayed one by one with some time gap between each one, so that the image appears waving its hands from bottom to top and again from top to bottom.

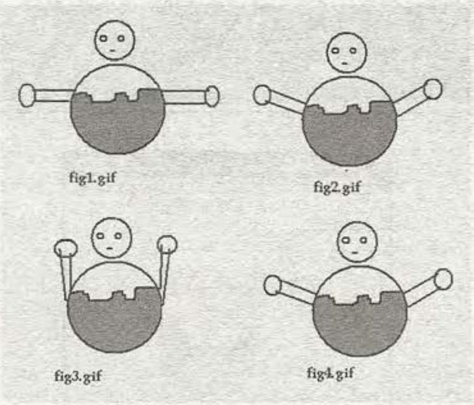


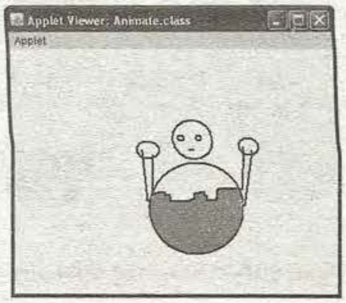
Figure 30.2 Four images for animation

Program 5: This program demonstrates animation using several images. By displaying all the images continuously one by one we get an illusion that the same image is moving. Press Control+C at System prompt to terminate the applet.

```
//Continuous animation
import java.awt.*;
import java.applet.*;
public class Animate extends Applet
{
      public void paint(Graphics g)
      {
              //load the images into Image objects
              Image img1 = getImage(getDocumentBase(), "fig1.gif");
              Image img2 = getImage(getDocumentBase(), "fig2.gif");
              Image img3 = getImage(getDocumentBase(), "fig3.gif");

              //continuous animation
              for(;;)
              {
                      //display images one by one with time gap of 200milli
                      seconds
                      try{
                              g.drawImage(img1,50,50,null);
                              Thread.sleep(200);

                              g.drawImage(img2,50,50,null);
                              Thread.sleep(200);

                              g.drawImage(img3,50,50,null);
                              Thread.sleep(200);

                              g.drawImage(img2,50,50,null);
                              Thread.sleep(200);

                      }catch(InterruptedException ie){}
              }
      }
}
```

Output:

```
C:\> javac Animate.java

<!Animate.html>
<html>
<applet code= "Animate.class", height=300 width=400>
</applet>
</html>

C:\> appletviwer Animate.html
```



A simple Game with an Applet

One of the important uses of applets is for developing games. Let us write a simple game where we display a push button and ask the user to click it. When the user tries to place the mouse pointer on the button, the button will jump to a new location. This new location will be decided using a random() method so that no one can guess to what location the button will jump.

In this game, we create a push button with an image that uses a smiling face. When the user clicks the button, the image is changed to a crying face. For this purpose, two images (fig.gif and fig1.gif) can be drawn in paint as shown in Figure 30.3.

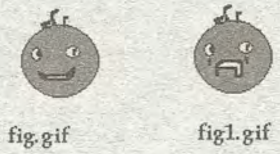


Figure 30.3 Figures useful in our game

To prevent the user from placing the mouse on the button, we should receive mouse events so that we can understand what the user is doing. For this purpose, we should attach two listeners to the button: `MouseMotionListener` and `MouseListener`. `They got methods to catch any mouse event`. In any case, the button location is changed as:

```
int x = (int)(600*Math.random());
int y = (int)(500*Math.random());
b.setBounds(x,y,250,75);
```

The `setBounds()` method positions the button at a random location specified by x and y coordinates. It is almost impossible to click the button and get some score in this game.

Program 6: In this program, we create a push button. When the user tries to place the mouse on the button, the button jumps to a new location randomly. When the user clicks on the button, the score is increased by 100.

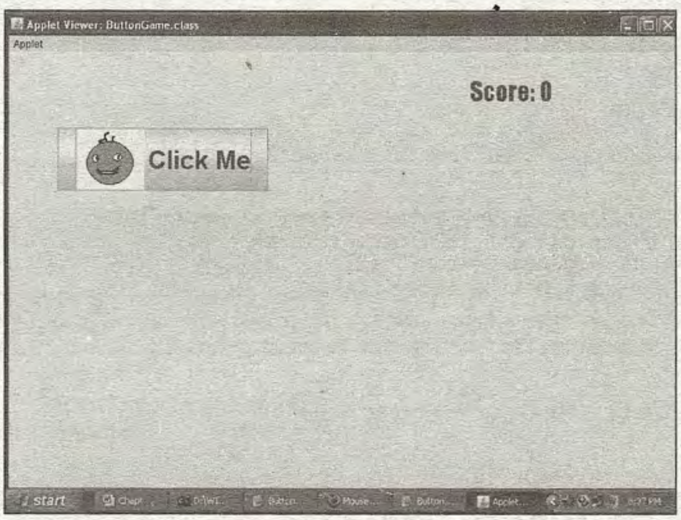
```
//An applet where a button is displayed for the user to click.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ButtonGame extends JApplet implements MouseMotionListener,

MouseListener
{
      //vars
      JButton b;
      JLabel lbl;
      static int score=0;
      public void init()
      {
              //goto content pane
              Container c = getContentPane();
              c.setLayout(null);

              //create the button with laughing image
              ImageIcon ii = new ImageIcon("fig.gif");
              b = new JButton("Click Me", ii);
              b.setFont(new Font("Helvetica", Font.BOLD, 30));
              b.setBounds(400,300,250,75);
              c.add(b);
```

```
                //create a lable to display score
                lbl= new JLabel();
                lbl.setFont(new Font("Impact", Font.PLAIN, 30));
                lbl.setText("Score: "+score);
                lbl.setBounds(550,20,150,50);
                c.add(lbl);

                //add listeners to button
                b.addMouseMotionListener(this);
                b.addMouseListener(this);
        }

        public void mouseDragged(MouseEvent me)
        {
                //change the button coordinates randomly when the mouse is dragged
                int x = (int)(600*Math.random());
                int y = (int)(500*Math.random());
                b.setBounds(x,y,250,75);
        }

        public void mouseMoved(MouseEvent me)
        {
                //change the button coordinates randomly when mouse on it
                int x = (int)(600*Math.random());
                int y = (int)(500*Math.random());
                b.setBounds(x,y,250,75);
        }
        public void mouseClicked(MouseEvent e)
        {
                //when user clicks on the button change the image
                //as crying image and and add 100 to score
                ImageIcon ii = new ImageIcon("fig1.gif");
                b.setIcon(ii);
                lbl.setForeground(Color.red);

                score+=100;
                lbl.setText("Score: "+score);

        }

        public void mouseEntered(MouseEvent e)
        {
                //change the button coordinates randomly when mouse entered it
                int x = (int)(600*Math.random());
                int y = (int)(500*Math.random());
                b.setBounds(x,y,250,75);
        }


        public void mouseExited(MouseEvent e)
        {
                //when mouse is exited from the button, display laughing image
                ImageIcon ii = new ImageIcon("fig.gif");
                b.setIcon(ii);

        }

        public void mousePressed(MouseEvent e){}

        public void mouseReleased(MouseEvent e){}

    }
```

Output:

```
    C:\> javac ButtonGame.java
```

```
C:\>

<! ButtonGame.html that embeds ButtonGame applet>
<html>
<applet code="ButtonGame.class" width=800 height= 600>
</applet>
</html>

C:\> appletviewer ButtonGame.html
```



Applet Parameters

Along with applet tag, we can use <param> tag to pass information to applets. For example, to pass name and salary of an employee to an applet from HTML page, we can write the applet and param tags as:

```
<applet code="Tax.class" width=300 height=300>
<param name="t1" value= "Nag">
<param name="t2" value= "150000.50">
</applet>
```

<param> tag has two attributes 'name' and 'value'. 'name' represents the name of the parameter and 'value' indicates its value. For example, in the above code, we have two parameters: t1 and t2 and their values are "Nag" and "150000.50" respectively.

To receive the values of the parameters, an applet uses getParameter() method. This method takes the parameter name and returns its value as a string.

```
name = getParameter("t1");
str = getParameter("t2");
```

where name and str and String type variables. `getParameter()` method should be used inside the init() method of the applet.

<param> tag is useful for passing information to the applets. This eliminates the need for modifying the applet even if we want to pass different data. The following program demonstrates how to use

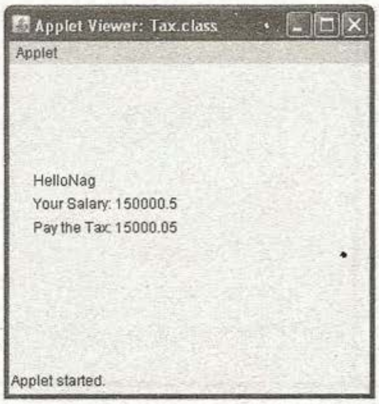
<param> tag to pass name and salary of an employee to an applet called 'Tax.class' and how the applet receives the data using `getParameter()` method and processes the data.

Program 7: Program to pass name and salary to an applet and get the tax calculated.

```
//Calculating tax by taking name and salary from param tags
import java.awt.*;
import java.applet.*;
public class Tax extends Applet
{
     //vars
     String name,str;
     float sal;
     float tax;

     public void init()
     {
            //accept name from t1 parameter
            name = getParameter("t1");

            //accept salary into str from t2 parameter
            //and convert str into float type sal
            str = getParameter("t2");
            sal = Float.parseFloat(str);

            //call calculateTax() method and pass sal to it.
            calculateTax(sal);
     }

     public void calculateTax(float sal)
     {
            //calculate tax value based on salary.
            if(sal<=100000)
            tax = 0.0f;
            else if(sal <= 200000)
            tax = sal*0.1f;
            else tax = sal*0.2f;
     }
     public void paint(Graphics g)
     {
            //display the tax details
            g.drawString("Hello"+ name, 20, 100);
            g.drawString("Your Salary: "+sal, 20,120);
            g.drawString("Pay the Tax: "+ tax, 20, 140);
     }
}
```

Output:

```
C:\> javac Tax.java
C:\>

<!Accept.html - this page sends name and salary to the applet >
<html>
<h1> INCOME TAX CALCULATOR </h1>

<! param tag takes name and its value as strings>
<applet code="Tax.class" width=300 height=300>
<param name="t1" value= "Nag">
<param name="t2" value= "150000.50">
</applet>

</html>

C:\> appletviewer Accept.html
```



```
HelloNag
Your Salary: 150000.5
Pay the Tax: 15000.05
```

Applet started.

Conclusion

An applet represents Java byte code that is embedded in an HTML page, so that when the page is sent to the client, the applet is executed in the client-side browser and the results are displayed. Applets are useful to provide dynamic nature to the web pages by communicating with the user at runtime and also by providing animation.