Module IV **9 hours**

Applet Programming Introduction, how applets differ from applications, building applet code, applet life cycle, about HTML, designing a web page, passing parameters to applets, getting input from the user.

Applet Programming Introduction:

Applets are small java programs the are primarily used in internet computing. They can be transported over the internet from one computer to another and run using the **appletviewer** or any java enabled **web browser**.

An applet can perform arithmetic operations, display graphics, play sounds, accept user inputs, create animations and play interactive games.

Applets can be of two types:

Local Applets

Remote Applets.

An applet can be a fully functional Java application because it has the entire Java API at its disposal. There are some important differences between an applet and a standalone Java application, including the following:

- An applet is a Java class that extends the java.applet.Applet class.
- A main() method is not invoked on an applet, and an applet class will not define main().
- Applets are designed to be embedded within an HTML page.
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
- A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime. Applets have strict security rules that are enforced by the Web browser.
- The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.

How applets differ from applications:

Applets	Applications
Small Program	Large Program
Applets are not independent programs.i.e.they run from inside a web page or an appletviewer.	Applications are independent programs.Can be executed on stand alone computer system
Applets cannot read from or write to files in the local computer	Application can read from or write to files.
Applets cannot communicate with other servers on the network.	Applications can communicate with other servers on the network.
Applets are restricted from using the native methods	Applications can use the native methods.
Applet is portable and can be executed by any JAVA supported browser.	Need JDK, JRE, JVM installed on client machine.
Applet applications are executed in a Restricted Environment	Application can access all the resources of the computer
Applets are created by extending the java.applet.Applet	Applications are created by writing public static void main(String[] s) method.
Applet application has 5 methods which will be automatically invoked on occurance of specific event	Application has a single start point which is main method

Example:
import java.awt.*;
import java.applet.*;
public class Myclass extends Applet
{
    public void init() { }
    public void start() { }
    public void stop() {}
    public void destroy() {}
    public void paint(Graphics g) {}
}
	public class MyClass
	{
	public static void main(String args[]) {}
	}

Building Applet Code

To build the applet code two classes of java library are essential **Applet** and **Graphics**.The Applet class is contained in java.applet package.The **Applet** class maintains the life cycle of the applets with it's four methods – **init(),start(),stop()** and **destroy().**Another **paint()** is defined by the AWT **Component** class which requires a **Graphics** object as an argument.The output may be text, graphics or sound.The syntax of the paint() method is:

Public void paint(Graphics g)

The general format of building applet code is as follows:

```
import java.awt.*;
import java.applet.*;
……………………
……………………
public class appletclassname extends Applet
{
……………………………
…………………………… //statements
……………………………
public void  paint(Graphics g)
{
……………………
……………………//Applet operations code
……………………
}
…………………
…………………
}
```

Following is the chain of classes inherited by Applet class:

$$\begin{tikzcd}
\text{java.lang.Object} \arrow[d] \\
\text{java.awt.Component} \arrow[d] \\
\text{java.awt.Container} \arrow[d] \\
\text{java.awt.Panel} \arrow[d] \\
\text{java.applet.Applet}
\end{tikzcd}$$

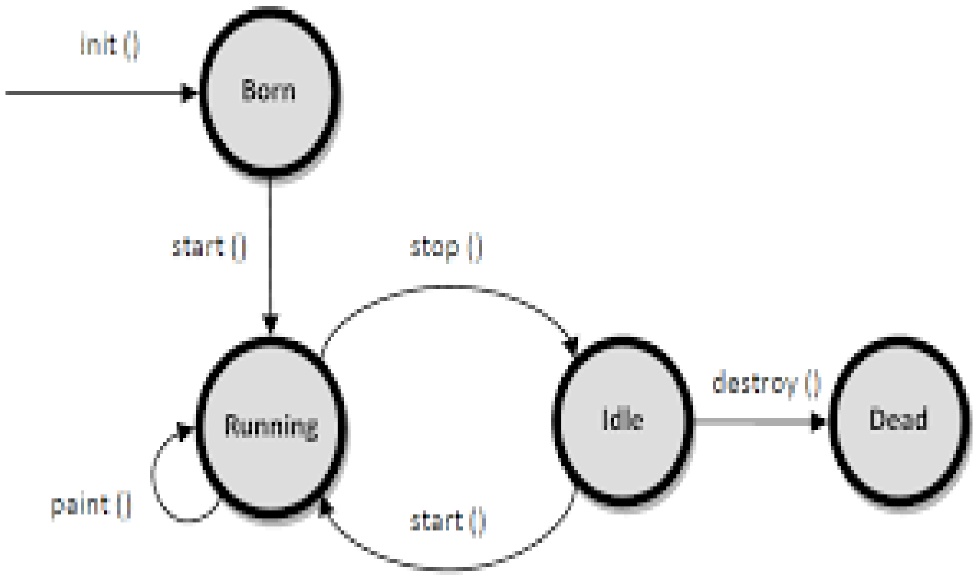
```
/*Example of building Applet Code
HelloApplet.java*/
import java.applet.Applet;
import java.awt.*;
  public class HelloApplet extends Applet
  {
    public void paint(Graphics g)
    {
      g.drawString ("Welcome to Applet world !",200, 100);
    }
  }
```

Life Cycle of an Applet:

Applet life cycle consists of 5 methods which will be automatically invoked by JRE when some action is done on the application/browser.

List of Methods:	**States:**
public void init()	Initialization
public void start()	Running
public void paint(Graphics g)	Display
public void stop()	Idle
public void destroy()	Dead/Destroyed

Applet's state Transition Diagram



Methods:

init(): This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.

- Invoked only once
- Invoked when the application is launched using appletviewer or browser
- Should include code for component definition, Object creation, Layout settings,set up initial values, load images or fonts,set up colors. Basic look and feel.

start(): This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.

- Invoked when application is maximized
- Default application state is Maximized window so this will also be invoked on launching
- Should include code for starting/resuming thread, starting/resuming Graphical User Interface, etc…

stop(): This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.

- Invoked when application is minimized
- invoked also when the window is terminated while its in maximizes state.
- Should include code for pausing/stopping thread, pausing/stopping Graphical User Interface, etc…

destroy(): This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.

- Invoked when application is about to terminate
- invoked when we close the application.
- Should include code for connection closing, file closing, etc…

paint(): Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

- Invoked when application is to be refreshed in terms of GUI

- invoked when we start, move or resize applet.
- Should include code for GUI design

Some method calling scenarios:

- On Launching an Applet application: init(), start(), paint()
- On Minimizing an Applet application: stop()
- On Maximizing an Applet application: start(), paint()
- On Moving/Re-sizing an Applet application: paint(), paint(), ..., paint()
- On Terminating an Applet application: stop(), destroy()

About HTML:

HTML stands for **H**yper **T**ext **M**arkup **L**anguage, which is the most widely used language on Web to develop web pages.

HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

```
<!DOCTYPE html>

<html>

  <body>

    <h1>Hello World!</h1>

  </body>

</html>
```

HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example **<html>**has its closing tag **</html>** and **<body>** tag has its closing tag**</body>** tag etc.

Tag	**Description**
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by **<head>...</head>** and document body which is represented by**<body>...</body>** tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The **<title>** tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.
<h1>	This tag represents the heading.
<p>	This tag represents a paragraph.

HTML Document Structure:

Document declaration tag

```
<html>

  <head>

     Document header related tags

  </head>


  <body>

     Document body related tags

  </body>

</html>
```

Designing Web Page:

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements **<h1>, <h2>, <h3>, <h4>, <h5>, and <h6>**. While displaying any heading, browser adds one line before and one line after that heading.

```
<!DOCTYPE html>

<html>

<head>

<title>Heading Example</title>

</head>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>
```

```
<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

</body>

</html>
```

<p> Paragraph Tag:

```
<!DOCTYPE html>

<html>

<head>

<title>Paragraph Example</title>

</head>

<body>

<p>Here is a first paragraph of text.</p>

<p>Here is a second paragraph of text.</p>

<p>Here is a third paragraph of text.</p>

</body>

</html>
```

<br> Line Break Tag:

```
<!DOCTYPE html>

<html>

<head>

<title>Line Break  Example</title>

</head>

<body>

<p>Hello<br />

This is your first paragraph <br />
```

```
Welcome to html<br />
All the best</p>
</body>
</html>
```

<center> Centering tag:

```
<!DOCTYPE html>
<html>
<head>
<title>Centering Content Example</title>
</head>
<body>
<p>This text is not in the center.</p>
<center>
<p>This text is in the center.</p>
</center>
</body>
</html>
```

<hr> Horizontal Line:

Horizontal lines are used to visually break up sections of a document. The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

```
<!DOCTYPE html>
<html>
<head>
<title>Horizontal Line Example</title>
```

```
</head>

<body>

<p>This is paragraph one and should be on top</p>

<hr />

<p>This is paragraph two and should be at bottom</p>

</body>

</html>
```

Steps to Create an Applet program

Step1:Write the applet code .(eg public class HelloJava Extends Applet{ })

Step2:Save the above code with .java extension.(HelloJava.java)

Step 3: Compile the above java file (javac HelloJava.java) to get HelloJava.class file.

Step 4: Design a web page using HTML tags and embed the .class file(HelloJava.class) created in step 3 and save the file with xxx.html extension

Example:

```
<html>

<body>

<applet code=HelloJava.class width=300 height=200)>

</applet>

</body>

</html>
```

Step 5: Test/Run the applet in 2 ways:

i) Open the HTML file(step4) in java enabled web browser(Java Plug-in installed and enabled properly)
ii) Use appletviewer tool(used only for testing purpose).

```
appletviewer  xxx.html
```

- **Simple example of Applet by html file:**

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code(.class file) in html file. Now click the html file.

```
//HelloJava.java

import java.applet.Applet;

import java.awt.Graphics;

public class HelloJava extends Applet

{

        public void paint(Graphics g)

        {

        g.drawString("welcome",150,150);

        }

 }
```

Note: class must be public because its object is created by Java Plugin software that resides on the browser.

<u>myapplet.html (embedding applet in HTML file)</u>

```
<html>

<body>

<applet code="HelloJava.class" width="300" height="300">

</applet>

        </body>

</html>
```

Note:Run myapplet.html from any java enabled web browser or use appletviewer tool

- **Simple example of Applet by appletviewer tool:**

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

```
//First.java

import java.applet.Applet;

import java.awt.Graphics;

 /*

        <applet code="HelloJava.class" width="300" height="300">

        </applet>

*/

public class HelloJava extends Applet

{

        public void paint(Graphics g)

        {

                g.drawString("welcome to applet",150,150);

        }

  }
```

To execute the applet by appletviewer tool, write in command prompt:

```
        c:\>javac First.java

        c:\>appletviewer First.java
```

Example2: Applet code , HelloJava.java

```
import java.awt.*;
import java.applet.*;
public class HelloJava extends Applet
{
public void init()
{
setBackground(Color.yellow);//defined in component class
setForeground(Color.red);
}
public void paint(Graphics g)
{
g.drawString(" in applet window", 10,100);
showStatus("this is shown instatus window");
}
}
```

Embedding HelloJava.class in HTML file(HelloJava.html)

```html
<html>
<head>
<title>applets program</title>
</head>
<body bgcolor=pink>
<center>
<h1> welcome to the world of applets</h1>
</center>
<applet code=HelloJava.class width=200 height=150 align=right>
</applet>
</body>
</html>
```

Note: HelloJava.java

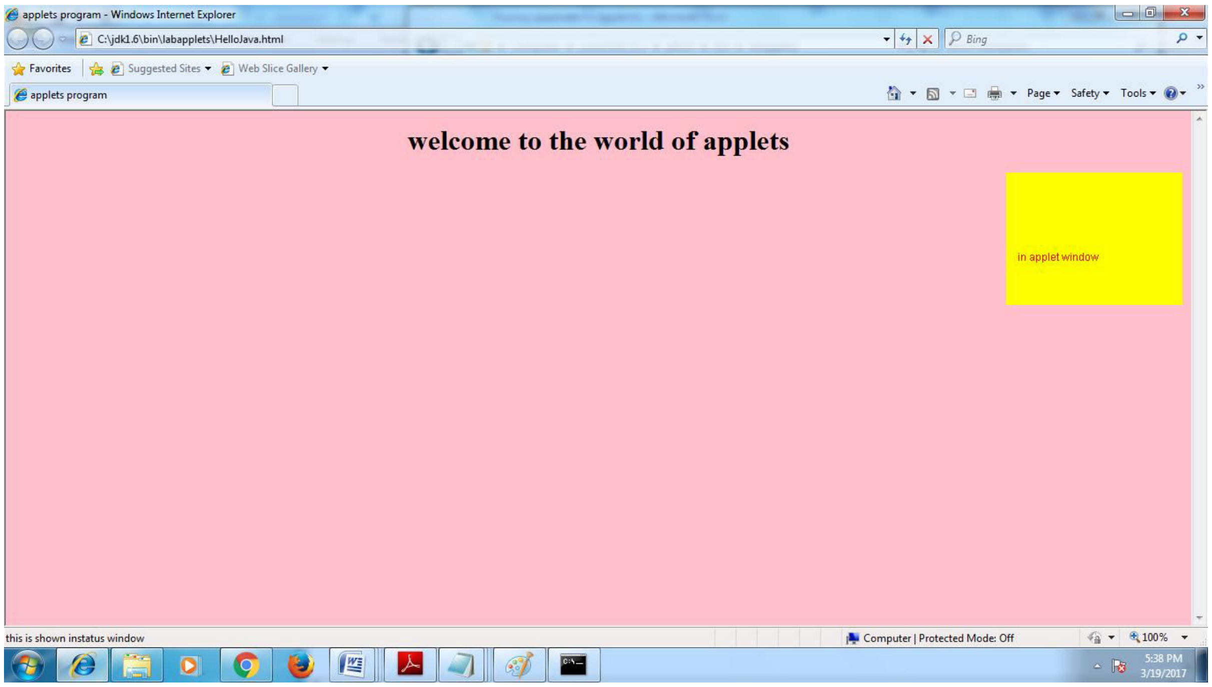
HelloJava.class

HelloJava.html

All the above three files are in the same directory.

Outputs

Running applet from web browser(say Internet Explorer)



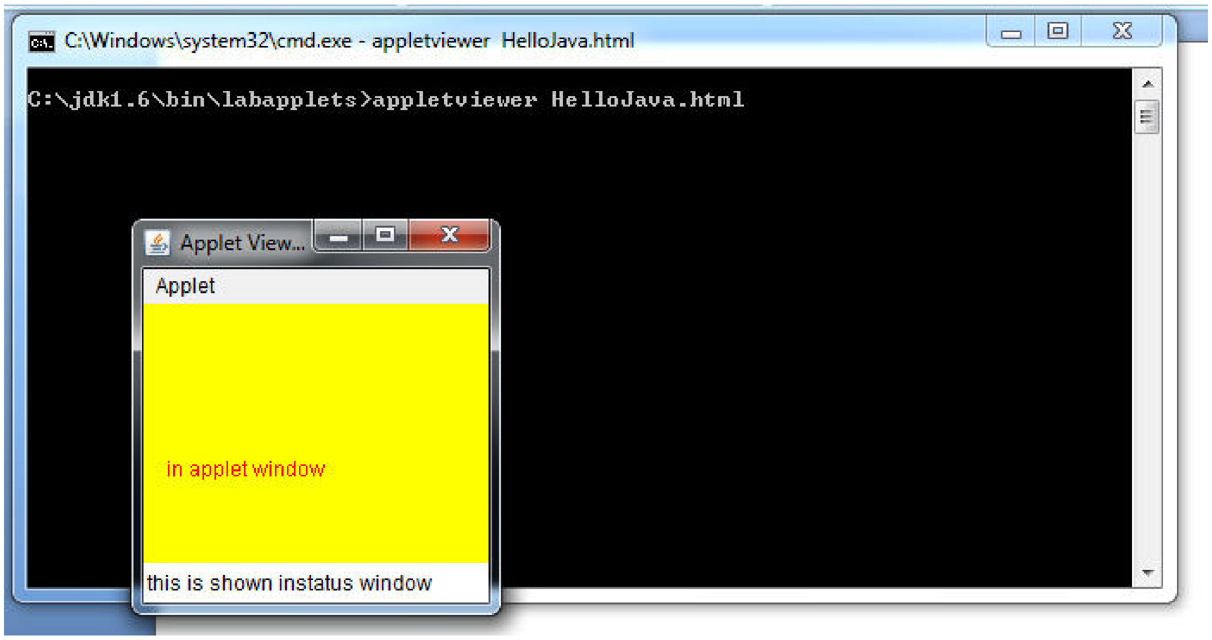
applets program - Windows Internet Explorer
C:\jdk1.6\bin\labapplets\HelloJava.html
Favorites
Suggested Sites
Web Slice Gallery
applets program
Page
Safety
Tools
Computer | Protected Mode: Off
100%
5:38 PM
3/19/2017

welcome to the world of applets

in applet window

this is shown instatus window

Running applets using appletviewer



```
C:\jdk1.6\bin\labapplets>appletviewer HelloJava.html
```

in applet window

this is shown instatus window

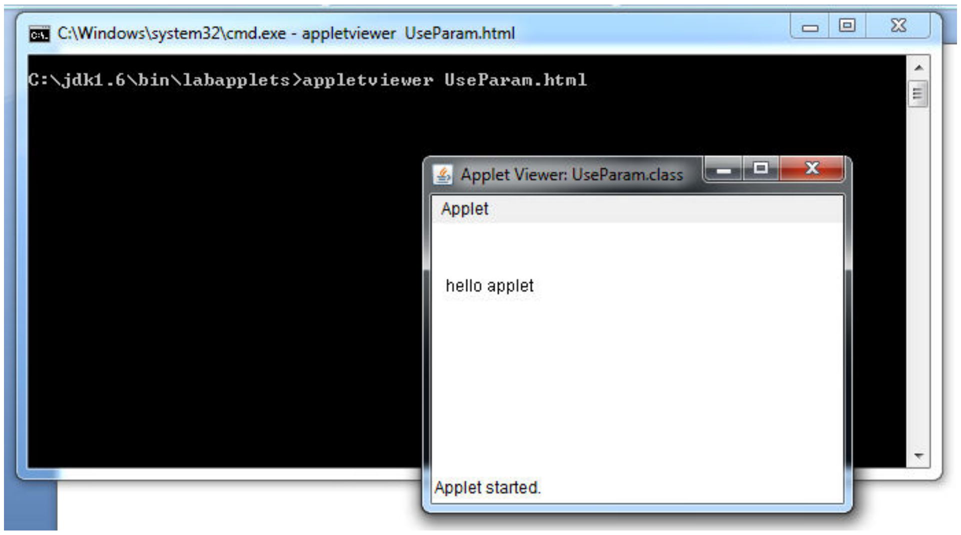
Passing parameters to applets

User-defined parameters can be passed to an applet using <PARAM…> tags. Each <PARAM..> tag has a **name** attribute and a **value** attribute.Inside the applet code, the applet can refer to the parameter by **name** to find it's value.Parameters are passed to the applet when it is loaded. We can define an **init()** method in the applet to hold the parameters defined in the <PARAM> 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.

Example program to illustrate <PARAM> tag

```
import java.applet.Applet;

import java.awt.Graphics;

 public class UseParam extends Applet

{

         String str;

        public void init()

        {

        str=getParameter("msg"); //receiving parameter value

        if(str ==null)

        str="JAVA";

        str="hello"+str;

        }

        public void paint(Graphics g)

        {

        g.drawString(str,10, 50);

        }
}
<html>
<body>
<applet code="UseParam.class" width="300" height="300">
<param name="msg" value=" applet">
</applet>
</body>
</html>
```

Output



```
C:\Windows\system32\cmd.exe - appletviewer  UseParam.html
C:\jdk1.6\bin\labapplets>appletviewer UseParam.html
```

```
Applet Viewer: UseParam.class
Applet
hello applet
Applet started.
```

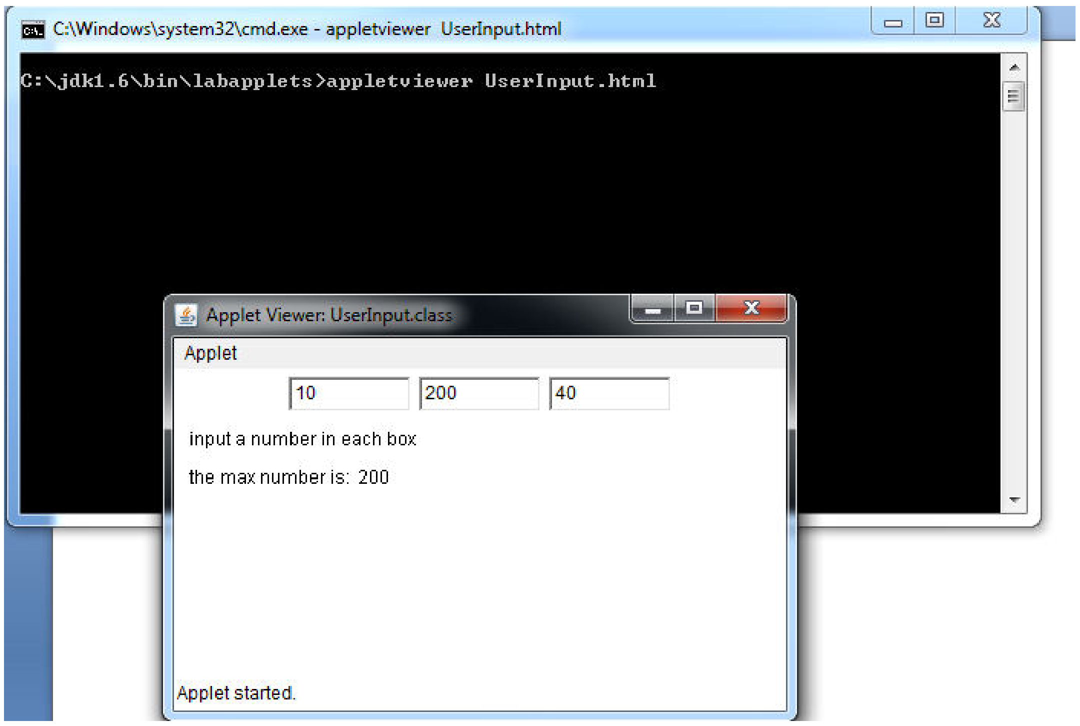
Getting input from the user

Applets work in a graphical environment. Applets treat inputs as text strings. First create an area on the screen in which the user can type and edit input items(of any type).For this **TextField** class of the applet package is used.Once the text fields are created for receiving input, we can type the values in the fields and edit them, if necessary.

Next step is to retrieve the items from the fields for display of calculations, if any.Text fileds contains items in string form. They need to be converted to the right form, before they are used in computations. The results are then converted back to strings for display.

```
//Getting input from user,applets treat inputs as text strings.
//UserInput.java


import java.awt.*;

import java.applet.*;

public class UserInput extends Applet

{

TextField text1,text2,text3;  //TextField is a class of awt package to receive

                                   user input of any datatype

public void init()

 {

 text1=new TextField(8);      //create a text field object to hold strings of 8

                                   character length

text2=new TextField(8);

  text3=new TextField(8);

add(text1);                      //add the objects to the applet's display area
add(text2);

add(text3);

text1.setText("0");                    //Initialize the contents of the objects to zero.

text2.setText("0");

text3.setText("0");

  }

public void paint(Graphics g)          //here in paint() all actions takes place

  {

int x=0,y=0,z=0,max;                   //integer variables

   String s1,s2,s3,s;                  // string variables

g.drawString("input a number in each box",10,50);

try

   {

    s1=text1.getText();                //input is in string form so getText() retrieves

                                               as strings

    x=Integer.parseInt(s1);            //for computation convert s1 string to integer

                                       using pareseInt() of Integer class

    s2=text2.getText();

    y=Integer.parseInt(s2);

    s3=text3.getText();

    z=Integer.parseInt(s3);

   }

catch(Exception e){ }

max=x;

if(y>max)max=y;
if(z>max)max=z;
  s=String.valueOf(max);                    //converting back to string for drawString()
g.drawString("the max number is:",10,75);
g.drawString(s,120,75);
 }
public boolean action(Event event,Object object)
{
repaint();
return true;
}
}


//UserInput.html file
<html>
<applet code=UserInput.class
width=400
height=200>
</applet>
</html>
```

Output


C:\Windows\system32\cmd.exe - appletviewer UserInput.html
Applet Viewer: UserInput.class
Applet

```
C:\jdk1.6\bin\labapplets>appletviewer UserInput.html
```

```
10    200    40
input a number in each box
the max number is:  200

Applet started.
```