Unit -1: Java Applets

1.1 Concept of Applet Programming:

- Local and remote applets
- Difference between applet and application
- Preparing to write applets
- Building applet code
- Applet life cycle
- Creating an Executable Applet

1.2 Designing a Web page:

- Applet tag
- Adding Applet to HTML file
- Running the Applet
- Passing parameter to applet

Introduction

- Java supports two types of programming :

1) **Application program:**
   Application is a program that runs on your computer under the operating system of that Computer.
2) **Applet program:**
   An **applet** is an application designed to be transmitted over the Internet and executed by a Java-compatible Web browser.

Compare Web Applet and Stand-alone application

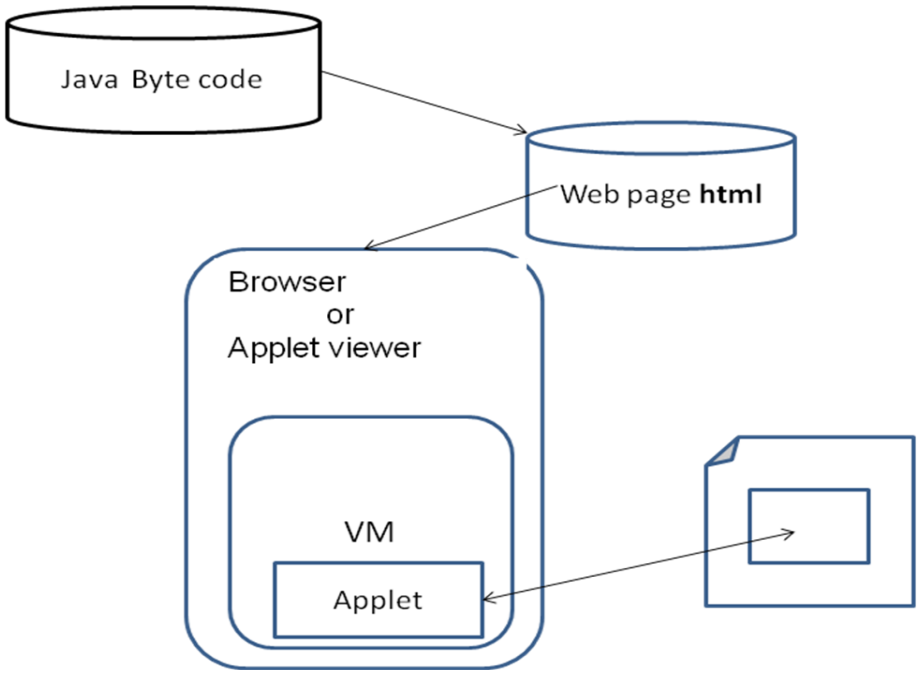
Or

Applet program and application program

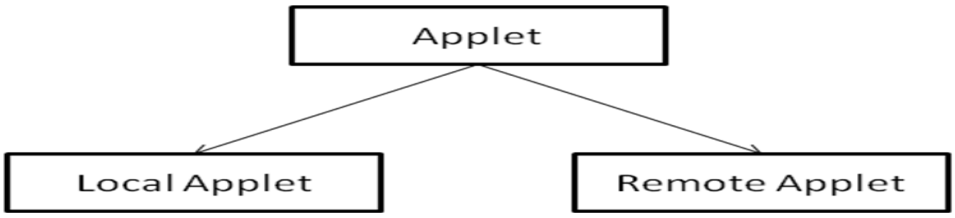
NO	Applet (web applet)	Application Program (Stand-alone application)
1	An **applet** is an application designed to be transmitted over the Internet and executed by a Java-compatible Web browser.	Application is a program that runs on your computer under the operating system of that Computer.
2	Applet is dynamic program which is run on browser.	Application is static program which run using java interpreter.
3	Applet do not use the main () method for initiating the execution of the code.	Application uses main () method for execution of the code.
4	Applets cannot be run independently. They must be run under an applet viewer or a java compatible web browser.	Application runs independently using javac compiler.
5	Applets cannot read or write files on the web user's disk.	Application can read write files on the web user's disk.
6	Applet cannot communicate with other server on the network.	Application can communicate with other servers on the network.
7	Applet cannot run any program from the local computer.	Application program can run from local computer.
8	Applets are restricted from using libraries from other languages such as c, c++.	Application program can use methods of c, c++ libraries.

Applet concept

- **Applet** is small programs that are primarily used in internet computing, they can be transported over the internet from one computer to another and run using applet viewer or java compatible web browser.
- **Java applet is a java class that you embed in an HTML page and is downloaded and executed by a web browser.**
- Applet can't be executed directly.
- For running an applet, **HTML file must be created** which tells the browser what to load and how to run it.
- applet begins execution via loading of a HTML page "containing it" after that java enabled web browser or "applet viewer" is required to run an applet
- Now, Web pages not only contain static text or simple image but it can also perform arithmetic operation, displays graphics, play sounds and moving Images.
- We can embed applets into web pages in two ways:
  1. We can write our own applets and embed them into web pages.
  2. We can download an applet from a remote computer system and then embed it into a web page.

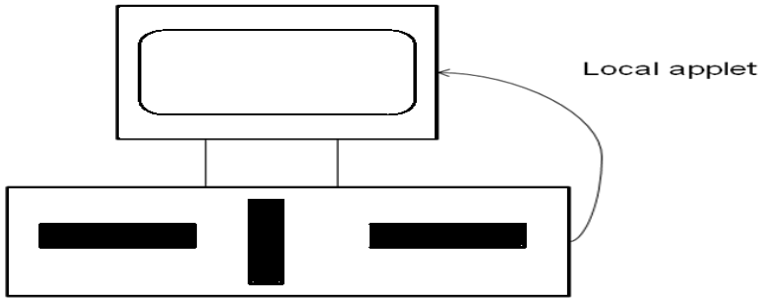


Types of applet:



1) Local Applet

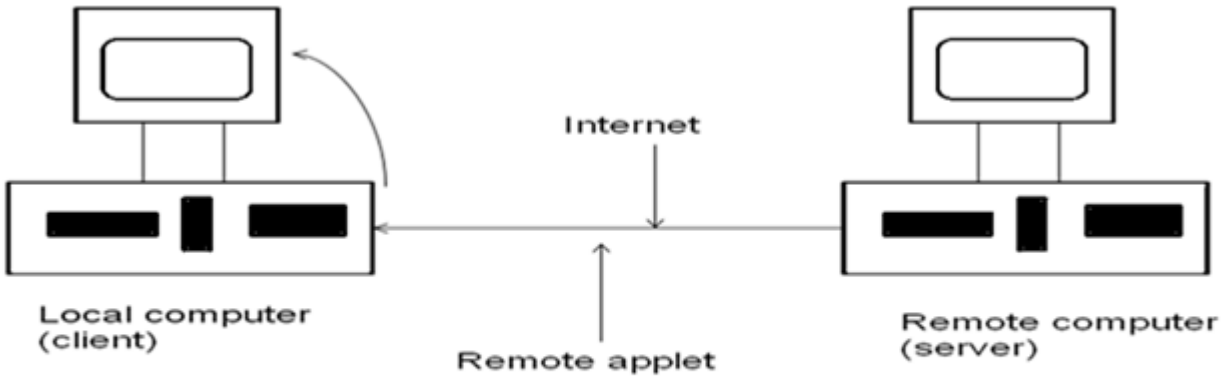
- An applet developed locally and stored in a local system is known as local applet.
- When a web page is trying to find a local applet, it does not need to use the Internet and therefore the local system does not require the Internet Connection.
- It simply searches the directories in the local system and locates and loads the specified applet.
- User can write his own applets and embed them into web pages.



Local computer

2) Remote Applet

- An applet which is developed by someone else and stored on a remote computer connected to the Internet is known as remote Applet.
- If our system is connected to the Internet, we can download the remote applet onto our system via Internet and then run it.
- User can thus download an applet from a remote computer sys
- In order to locate and load a remote applet, we must know the applet's address on web. This address is known as **URL (Uniform Resource Locator)** and must be specified in the **applet's HTML document as the value of the CODEBASE attribute.**
  EX: **CODEBASE= http: // www. Netserve.com /applets**
- In case of local applets, CODEBSAE may be absent or may specify a local directory.



Loading a remote applet

Limitations of applet

- All the restriction and limitations are placed in the interest of security of systems. These ensure that an applet cannot do any damage to local system.
- Applets allow neither to execute any application nor to load any DLL s on the local system.
- Applets do not need a main method.
- Applet runs under an applet viewer or a java compatible web browser.
- Applets can't read or write files on the web user's disk. If information must be saved to disk as an applet is executing, the storage of information must be done on the disk from which the web page is served.
- Applet cannot make network connection to a computer other than the one from which the web page is served, expect to direct the browser to a new location.
- Applets are restricted from using libraries from other languages such as C,C++.
- Applet cannot run any programs on the web user's system, including browser plug-ins, activeX controls or other browser related items.
- Some of Java's functionality (like removal of pointers, verification of byte code and restricted remote and local file access) blocked for applets because of security concerns.

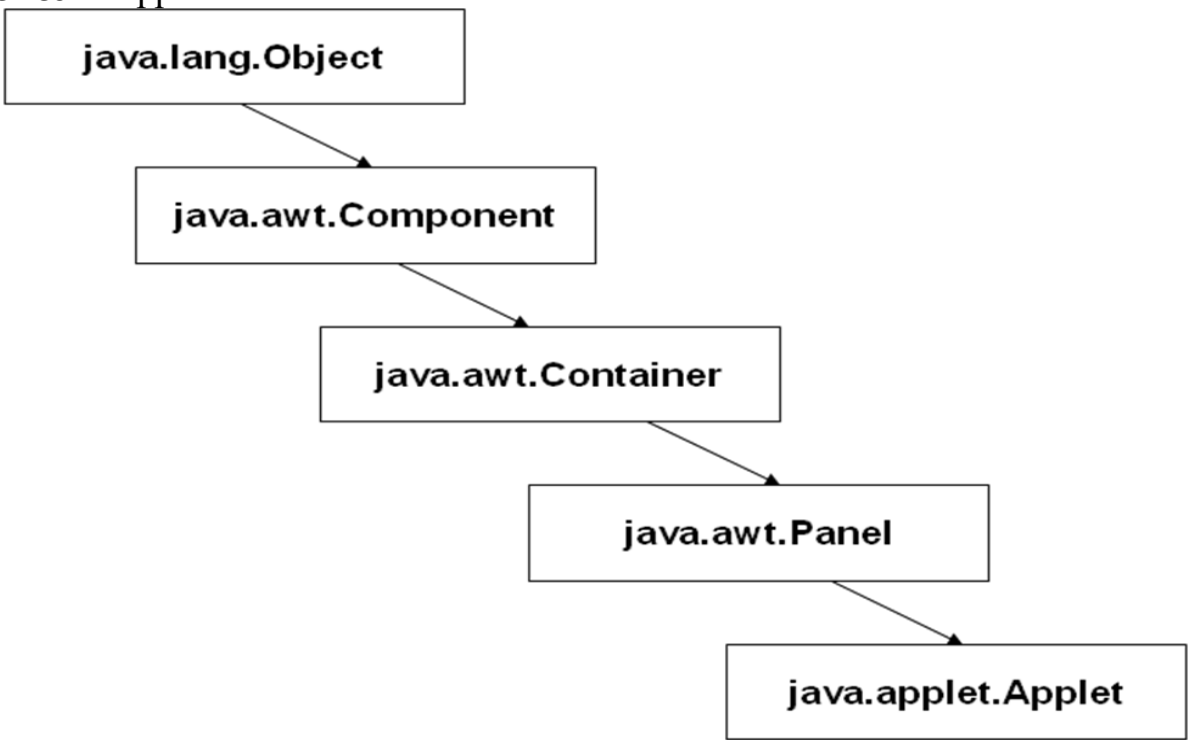
When to use Applet?

- When we need something dynamic to be included in the display of a web page.
- When we require some flash outputs. For example, applets that produce sounds, animations or some special effects would be used when displaying certain pages.
- When we want to create a program and make it available on the internet for us by others on their computers.

The Applet class

- **java.applet.Applet** class is actually a subclass of **java.awt.Panel**.
- **java.applet is smallest package** in Java API.
- It consists of a **single class and three interfaces**:
  - **Class** : Applet class
  - **Three Interface**: Applet Context, AppletStub and Audio clip.
- Applet class contains only a single default parameter less constructor, which is not used generally.
- Applets are constructed by the run time environment when they are loaded, they do not have to be explicitly constructed.

- Applet class contains 21 methods that are used to display images, play audio files, respond to events and obtain information about applet's execution environment, referred as applet's context.



Applet class Hierarchy

Methods of Applet class

No	Methods	Action
1	**Image getImage(URL url):**	Used to retrieve an Image identified by URL
2	**AudioClip getAudioClip(URL url)**	Used to retrieve Audioclip object that is identified by URL.
3	**Void play(URL url):**	If an audioThe play method is used to play an audio clip
4	**void init( ), void start() void stop(), void destroy( ) :**	Used to implement each of four life cycles stages of an applet.
5	**boolean isActive() :**	Return true if applet has been started else return false if it has been stopped
6	**AppletContext getAppletContext():**	Used to obtain AppletContextObject associated with an applet.

7	**String getAppletInfo():**	Returns a string object that provides information about applet. this includes version, copyright and ownership data as well as applet specific data.
8	**URL getCodeBase() :**	Returns base URL specifying the applet's location.
9	**URL getDocumentBase():**	returns the URL of the HTML document in which the applet is contained
10	**String getParameter(String paramname):**	Used to obtain parameter data that is passed to an applet in an HTML file.returns null if parameters not found.
11	**String [ ] [ ] getParameterInfo() :**	Returns array that describe all the parameters used by an object
12	**void resize(Dimension dim):**	Used to resize an applet according to the dimensions specified by dim.
13	**void showStatus(String str) :**	Used to display a status message in the status window of the browser or appletviewer.
14	**void setStub(AppletStub stubObj):**	Used to set the AppletStub associated with the applet. It should not be used unless you are constructing your own custom applet viewer.

Stub:

A stub is a small piece of code that provides the linkage between your applet and the web browser.

AppletContext Interface

- It defines methods that allow an applet to access the context in which it is being run.
- AppletContext interface is accessed using getAppletContext() method of Applet class.
- It provides 7 methods that allows an applet to obtain information about and manipulate its environment.

    1. Enumeration getApplets()

2. Applet getApplet(String name)
3. AudioClip getAudioClip(URL *url*)
4. Image getImage(URL *url*)
5. void showDocument(URL *url*)
6. void showDocument(URL *url* ,String *str*)
7. void showStatus(String *str*)

AudioClip interface

It defines three methods:

1. void play(URL url) :to play an audio clip
2. void stop(): to terminate the playing of an audio clip.
3. void loop(): to start and play an audio clip in a continuous loop.

Simple Applet display Method

No.	Method	Description
1	void drawString(String ***message*, int x,int y)** ***e.g. drawString("hi",10,100)***	To output a string to an Applet.
2	void setBackground(Color *colorname*) ***e.g. setBackground(Color.red)***	To set the back ground of a applet window
3	void setForeground(Color *colorname*) ***e.g. setForeground(Color.pink)***	To set the foreground color of an applet window.
4	Color getBackground()	To obtain the current settings for the background color.
5	Color getForeground()	To obtain the current settings for the foreground.
6	Applet getApplet(String name)	To obtain the applet specified by given name from the current applet context.
7	void showStatus(String status)	To display the status message in the status bar of applet window.
8	URL getDocumentBase()	To obtain the directory of the current browsers page.
9	URL getCodeBase()	To obtain the directory from which the applet's class file was loaded.

Color class:

- we can fill colors in applet using **Color** class of **awt** package.
- **Pre-defined colour in java:**

Color.black	Color.blue	Color.cyan	Color.mangenta
Color.darkgray	Color.gray	Color.green	Color.yellow
Color.lightgray	Color.red	Color.white	Color.orange
Color.pink			

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

setBackground(Color.green); or
setForeground(Color.red); or
setColor(Color.red);

Color constructor:

- We can create any new color bu using constructor of Color class.
- It takes three integer parameters and uses RGB combination for creating new colors.

Color colorobj=new Color (int red ,int green, int blue);

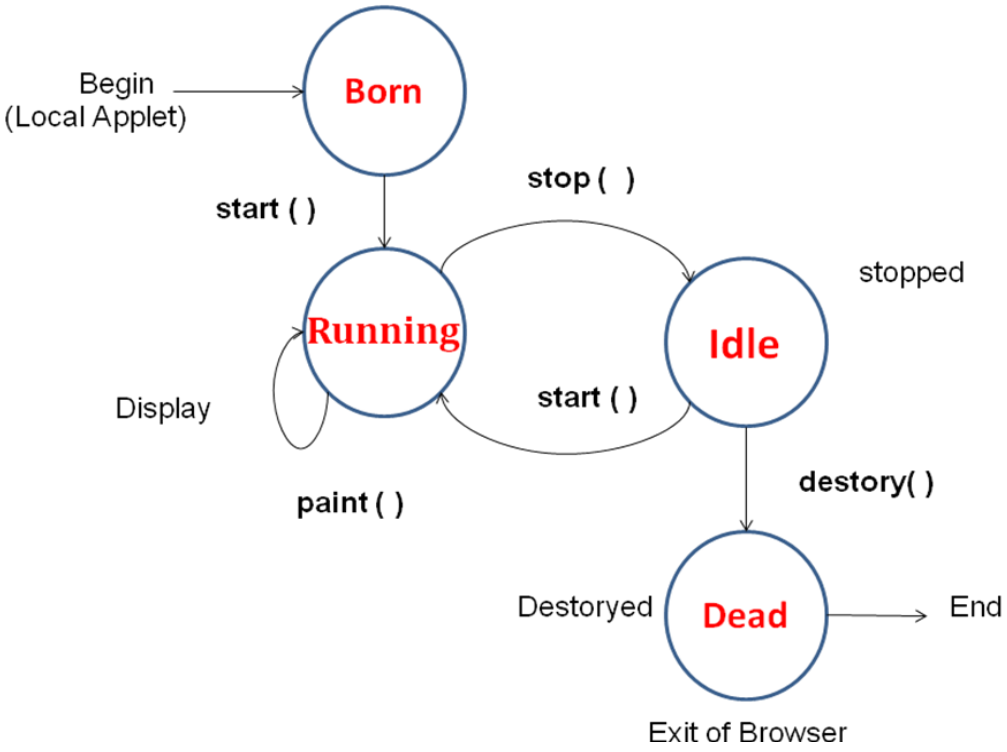
E.g: **Color** c=new **Color**(255,100,100);
g.**setColor**(c);

Color Obtained	Red Value	Green Value	Blue Value
White	**255**	**255**	**255**
Black	**0**	**0**	**0**
Lightgray	**192**	**192**	**192**
darkgray	**128**	**128**	**128**
Red	**255**	**0**	**0**
Green	**0**	**255**	**0**
Blue	**0**	**0**	**255**
Yellow	**255**	**255**	**0**
purple	**255**	**0**	**255**

Applet life cycle

- Every java applet inherits a set of default behaviours from the Applet class.so when an applet is loaded ,it undergoes a series of changes in its state.
- The applet states are:

1. **Born or Initialization state**
2. **Running state**
3. **Idle state**
4. **Dead or destroyed state**



1) Initialization state

- Applet enters the initialization state when it is first loaded.this is achieved by calling the init() method of Applet class.The applet is born.
- Initialization occurs only once in the applet's life cycle.
  - We do following at this stage,
    - create objects needed by the applet
    - set up initial values,initailize variables
    - Load images or fonts
    - set up colors
  - To provide any of behaviour mentioned above,we must overrride **init() method:**

```
public void init()
{
```

```
.....(Action)
.....
}
```

2) Running state

- Applet enters the ***running*** state when the system calls **start()** method of Applet class.
- This occurs automatically after applet is initialized.
- Starting of applet can also occur if the applet is already in "stopped" (idle) state.
- Unlike **init() method,**the **start()** method may be called more than once.
- We may override the **start()** method to create a thread to control the applet.

```
public void start( )
{
.....(Action)
.....
}
```

3) Idle or stopped state

- An applet becomes *idle* when it is stoppped from running.
- Stopping occurs automatically when we leave the page containing the currently running applet.
- We can also stop applet by calling the **stop()** method explicitly.
- If we use thread to run the applet then we must use **stop()** method to terminate the thread.
- To do this override **stop() method:**

```
public void stop()
{
.....(Action)
.....
}
```

4) Dead state

- An applet is said to be dead when it is removed from memory .this occurs automatically by invoking the **destroy()** mehtod when we quit the browser.
- Destroying stage occurs only once in the applet's life cycle.
- If the applet has created any resources,like threads we may override **detstroy()** method to clean up these resources .

```
public void destroy()
{
.....
.....
}
```

5) Display state

- Applets moves to the display state ,whenever it has to perform some output operations on the screen.
- This happens immidiately after the applet enters into the running state.
- The **paint()** method is called to accomplish this task.
- Almost every applet will have **paint()** method .deault version of **paint()** method does nothing.
- We must have to override **paint()** if we want anything to be displayed on the screen.
- Display state is not considered as a part of the applet's life cycle.
- The **paint()** method is defined in the applet class.it is inherited from the **Component** class ,a super class of Applet.

```
public void paint()
{
.....(Display statements)
.....
}
```

Steps to developing and testing an applet:

1) Building an applet code(.java file)
2) Creating an executable applet (.class file)
3) Designing a web page using HTML tags
4) Preparing <APPLET> tag
5) Incorporating <APPLET> tag into the web page
6) Creating HTML file
7) Testing the applet code

1. Building an applet code

- Applet code uses the services of two classes,namely **Applet** and **Graphics** from java class.
- Applet class is contained in **java.applet** package ,which provides life and behaviour to the applet through its methods such as **init() ,start()** and **paint().**
- **Applet class maintains the *life cycle* of an applet.**
- When an applet is loaded ,java automatically calls a series of Applet class methods for the starting,running,stopping the applet code.
- When **paint()** method is called ,it will actually display the result of the applet code on the screen.The output may be text,graphics or sound.
- The **paint()** method requires a **Graphics** object as an argument.

public void **paint** (**Graphics** g)

- We have to import java.awt package that contains **Graphics** class.
- All output operation of an applet are performed using the methods defined in the **Graphics** class.
- The *appletclassname* is the main class for the applet.when the applet is loaded ,java creates an instance of this class and then create Applet class methods are called on that instance to execute the code.
- *appletclassname* should be declared **public** because it is main applet class.

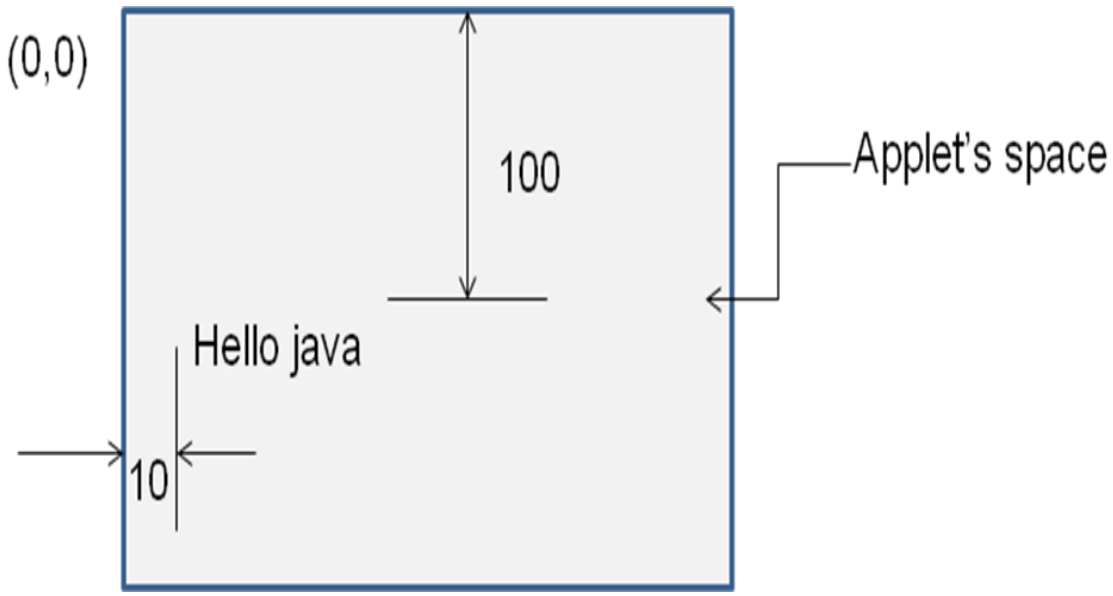
General format of applet code:

```
import  java.awt.*;
import  java.applet.*;
.....
.....
public  class appletclassname extends Applet
{
.....
.....
    public void paint(Graphics g)
    {

        //Applet operation code
    }
.....
.....
}
```

//save file with name ***Hellojava.class*** in a java subdirectory

```
import  java.awt.*;
import  java.applet.*;
public class Hellojava extends Applet
{
public void paint(Graphics g)
    {
        g.drawString( "Hello java" ,10,100);
    }
}
```

Output:



2. Creating an executable applet

- Executable applet is **.class** file of the applet which is obtained by compiling the source code of the applet.
- Compiling an applet is same as compiling an application.

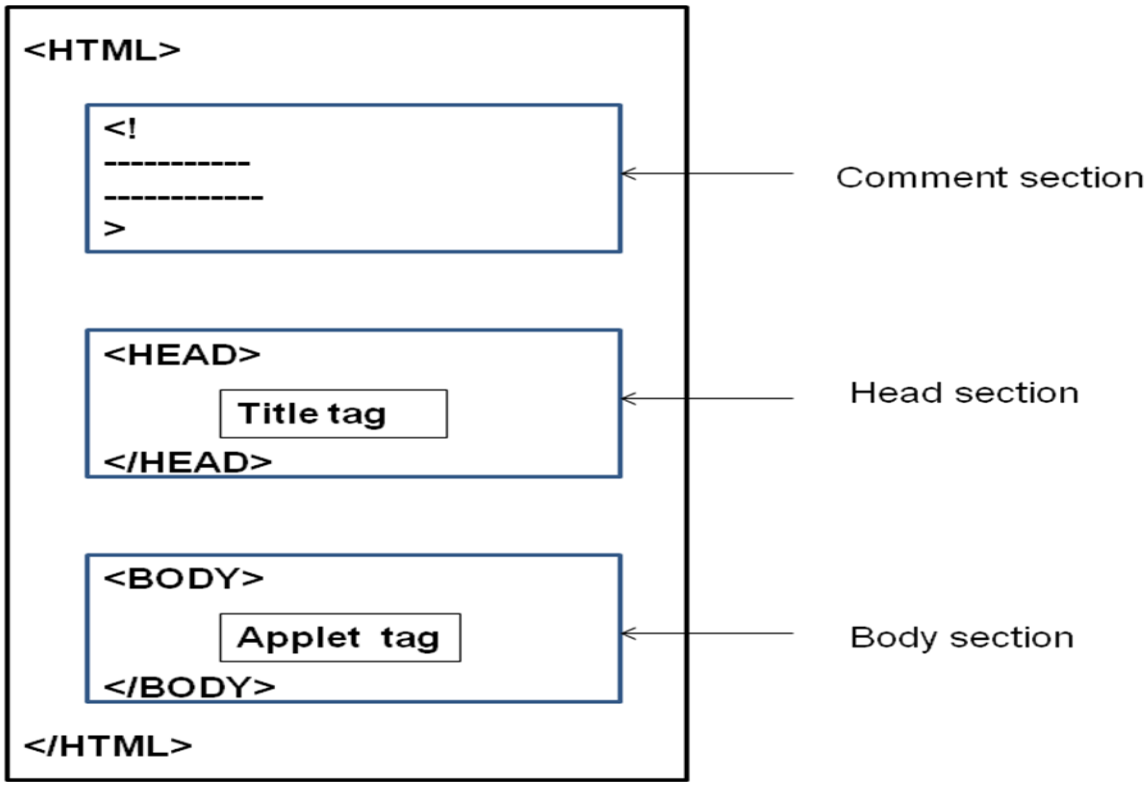
Ex : Hellojava.java

1) Move to directory containing the source code and type command:
   javac *Hellojava***.java**
2) The compiled output file called Hellojava.class is placed in same directory as the source .
3) If any error message is received ,then we must check for errors ,correct themand compile the applet again.

3. Designing webpage using HTML

- Java programs resides on web pages.to run a java applet ,it is necessary to have a web page that reference thta applet.
- A web page is made up of text and HTML tags that can be interpreted by a web browser or appletviewer.
- A web page is also known as *HTML page* or *HTML document.*
- Web pages are stored using a file extension **.html**
- HTML files should be stored in the same directory as the compiled code of the applets.
- Web page is divided in to three major sections:

  1) Comment section(optional) 2) Head Section(optional) 3) Body section



Head section:

<HEAD>

<TITLE> Welcome to Java Applets **</TITLE>**

</HEAD>

Here,text enclosed in <TITLE> and </TITLe> will apper in the title bar of the web browser when it displays the page.

Body Section:

After Head section comes the body section.this section contains the entire information about the web page and its behaviour.

```
<BODY>
    <CENTER>
        <H1> WELCOME </H1>
    </CENTER>
    <APPLET .....>

    </APPLET>
</BODY>
```

HTML tags

- **<HTML>.....</HTML>**
- <HEAD>.....</HEAD>
- <TITLE>.....</TITLE>
- **<BODY>.....</BODY>**
- <H1>.....</H1>.....<H6>.....</H6>
- <CENTER>.....</CENTER>
- **<APPLET...>**
- **<APPLET...>.....</APPLET>**
- **<PARAM....>**
- <B>.....</B>
-
- <P>
- <IMG.....>
- <HR>
- <A.....></A>
- <FONT>.....</FONT>
- <!..... (comment line)

4. Preparaing Appplet tag

- **<APPLET ....>** tag supplies the name of applet to be loaded and tells the browser how much space the applet requires.
- The ellipsis in the tag <APPLET **....**> indicates that it contains certain attributes that must be specified.
- Minimum requirement of <APPLET> TAG:

  <APPLET *CODE*=Hellojava.class *WIDTH*=200 *HEIGHT*=400 >

  </APPLET>

This HMTL code tells the browser to load the compiled java applet Hellojava.class ,which is in the same directory as this HTML file .and also specify display area for the applet output.

5. Adding applet to HTML file

<HTML>
<! This page includes a welcome title in title bar and also display a welcome message**.** **>**
<HEAD>
<TITLE> Welcome to Java Applets **</TITLE>**
</HEAD>

```
<BODY>
    <CENTER>
        <H1> WELCOME </H1>
    </CENTER>
    <APPLET CODE=Hellojava.class  WIDTH=200 HEIGHT=400 >
    </APPLET>
</BODY>
</HTML>
```

6. Running an applet

We must have following files in our current directory:

1) Hellojava**.java**
2) Hellojava**.class**
3) Hellojava.**html**

- To run an applet ,we require one of following tools:

1) **Java enabled web browser** :if we use it,we will be able to see the entire web page containing the applet.
2) **Java appletviewer:**

- if we use it,we will only see the applet output.
- appletviewer is not full fleged web browser and therfore it ignores all of the HTML tags except the part which runs applet.

Syntax:

appletviewer Hellojava.html

Appplet tag

- **<APPLET ....>** tag supplies the name of applet to be loaded and tells the browser how much space the applet requires.
- The ellipsis in the tag <APPLET **....**> indicates that it contains certain attributes that must be specified.
- Minimum requirement of <APPLET> TAG:

  <APPLET *CODE*=Hellojava.class *WIDTH*=200 *HEIGHT*=400 >

  </APPLET>

This HMTL code tells the browser to load the compiled java applet Hellojava.class ,which is in the same directory as this HTML file .and also specify display area for the applet output.

Attributes of Applet tag

< **APPLET**

[CODEBASE= codebase_URL**]**
CODE=AppletfileName.class
[ALT = Alternate_text **]**
[Name = applet_instance_name **]**
WIDTH = Pixels
HEIGHT = Pixels
[ALIGN= Alignment**]**
[VSPACE = Pixels **]**
[HSPACE = Pixles**]**

>

[<PARAM NAME=name1 **VALUE**= value1 >**]**
[<PARAM NAME=name2 **VALUE**= value2 >**]**
.....

</APPLET>

Everything in [ ] indicates the option can be used when integrating an applet into a webpage.

Attributes of Applet tag:

NO	Attribute	Meaning
1	**CODE=**AppletfileName**.class (necessary)**	**Specifies the name of the applet class to be loaded.** It is the name of already compiled **.class** file in which the executable java byte code for the applet is stored.
2	CODEBASE= codebase_URL **(**Optional)	Specifies URL of the directory in which the applet resides.(must used in remote applet)
3	**WIDTH =** Pixels **HEIGHT =** Pixels**(necessary)**	**Specify width and height of the space** on the HTML page that will be reserved for applet.
4	Name = applet_instance_name (Optional)	A name for the other applet on the page may refer to this applet. This facilitates inter applet communication.
5	[ALIGN= Alignment] (Optional)	Alignment of applet on page: Values for alignment are: (TOP, BOTTOM, LEFT, RIGHT,

		MIDDLE,ABSMIDDLE,ABSBOTTOM, TEXTTOP,BASELINE)
6	[VSPACE = Pixels] (Optional)	It specifies amount of vertical blank space the browser should leave surrounding the applet. it is Used only when ALIGN some vertical alignment is specified with the ALIGN attribute
7	[HSPACE = Pixels] (Optional)	It is Used only when ALIGN is set to LEFT or RIGHT, It specifies amount of horizontal blank space the browser should leave surrounding the applet.
8	[ALT = Alternate_text] (Optional)	Non _java browser will display this text where the applet would normally go.

Steps to adding an applet to HTML page

1. Insert an **<APPLET>** tag at an appropriate place in web page(in HTML file).
2. Specify the name of the applet's **.class** file.(CODE=Appclass.**class**)
3. If the **.class** file is not in the current directory ,uses the codebase parameter to specify
   - The relative path if file is on the local system
   - The URL of the directory containing the file if it is on a remote computer.
4. Specify the space required for display of the applet in terms of width and height in pixels.
5. Add any user defined parameters using <PARAM> tags.
6. Add alternate HTML text to be displayed when a non java browser is used.
7. Close the applet declaration with the </APPLET> tag.

passing parameters to applet

- **<PARAM ...> tag is used to supply user defined parameters to applet.**
- Passing parameter to an appplet code using <PARAM> tag is similar to passiing parameters to the **main()** method using command line argument.
- To pass and handles parameter ,do following:
  1. Include appropriate <PARAM...> tag in HTML document.
  2. Provide code in the applet to parse these parameters.
- **Parameters are passed to an applet when applet is loaded**.
- We can define **init()** method in applet to get hold of the parametes defined in <PARAM> tags.
- **getParameter() method** ,takes one string argument representing the name of program.

- Each <PARAM...> tag has a ***name*** **attribue** and **value attribute** .
- Inside the applet code ,the applet can refer to that parameter by name to find its value.
- We can also change text to be displayed by an applet by supplying new text to the applet through <PARAM....> tag.
  - E.g. :

```
<APPLET .....>
    <PARAM name=color value="red">
    <PARAM name=text value="I like java!">
</APPLET>
```

Example:

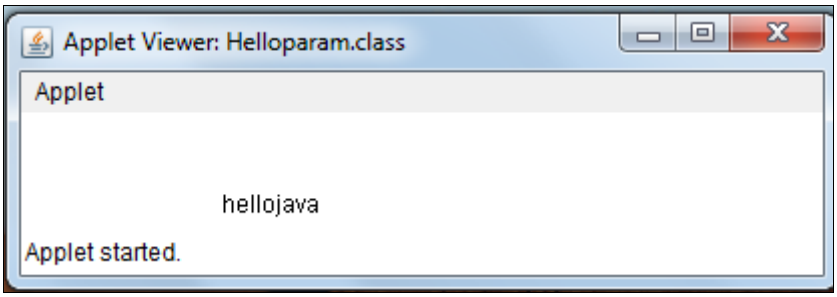
```
import    java.applet.*;
import    java.awt.*;
public class Helloparam extends Applet
{
    String str;
public void init()
    {
        str=getParameter("string");
        if(str==null)
            str="java";
        str="hello" + str;
    }
public void paint(Graphics g)
    {
        g.drawString(str,10,100);
    }
}
```

//save file in path "D:\javapro\Helloparam.java"

```
<html>
        <! parameterized HTML file>
<HEAD>
        <TITLE>Welcome to java Applets</TITLE>
</HEAD>
<BODY>
        <APPLET CODE=Helloparam.class width=400 height=400 >
        <PARAM NAME="string" VALUE= "Applet !">
        </APPLET>
</BODY>
</html>
```

//save file in path "D:\javapro\Helloparam.html"

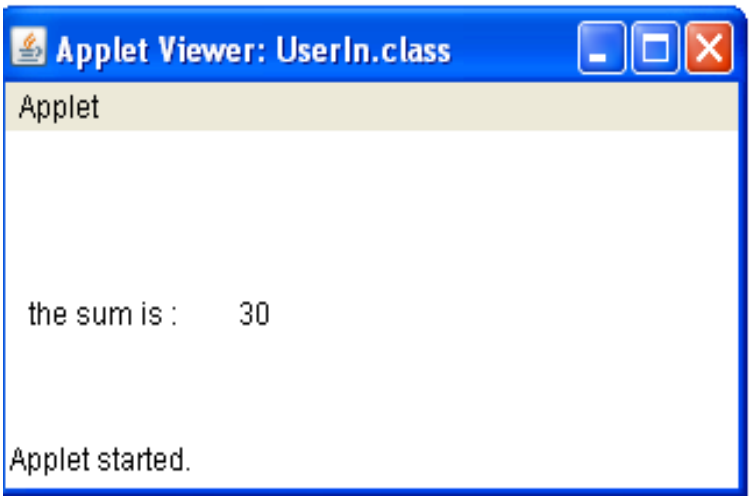
Output:



Displaying a numeric value

- In applet,we can display numerical values by first converting them into **strings** and then using **drawString()** method of Graphics class.
- **Example:**

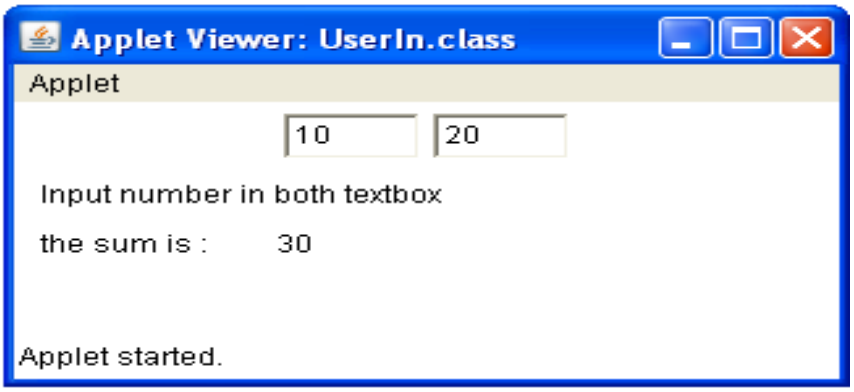
```
import   java.awt.*;
import  java.applet.*;
/*<applet code=NumValues.class height=400 width=400>
</applet> */
public class NumValues extends Applet
{
public void paint(Graphics g)
{
        int   val1=10;
        int   val2=20;
        int sum=  val1 + val2;
        String s = "sum :" + String.valueOf(sum);
        g.drawString(s,10,100);
}
}
```



Write an applet program which takes two input values from user and calculate sum of two values.

```
/*      <html><body>
        <applet code=UserIn.class width=400 height=400>                    </applet>
        </body> </html>
 */
import java.awt.*;
import java.applet.*;
public class UserIn extends Applet
{
        TextField text1,text2;
public void init()
{
        text1 = new TextField(8);              //creates Textfield (text box) object
        text2 = new TextField(8);
        add (text1);                           //add textField to our applet
        add (text2);
        text1.setText("0");                    //set default value  "0" in TextFeield
        text2.setText("0");
}
public void paint(Graphics g)
{
        int x=0,y=0,z=0;
        String s1,s2,s;
        g.drawString("Input number in both textbox ",10,50);
        try
        {
                s1=text1.getText();            //to get String object from TextField
                x=Integer.parseInt(s1);        //convert String object to integer value
                s2=text2.getText();
                y=Integer.parseInt(s2);
        }
        catch(Exception e) { }
        z = x + y;
        s=String.valueOf(z);            //converts integer sum into String objecttype
        g.drawString("the sum is :" ,10,75);
        g.drawString(s,100,75);
}
    public boolean action(Event event ,Object obj)
```

```
    {
    repaint();
    return true;
    }
}
```



Painting an applet

(Update, paint, repaint method)

- All three methods are defined in **Component** class.
- All the various controls and applets inhrit these methods.

1) paint ( ) method :

- It determines that what is done when any item is redrawn.
- It is called when anything needs to be redrawn.
- All displayble entities have paint() method that they inherit from Component.
- Interacive elements such as buttons ,are redrawn automatically no need to implement a paint method for them.
- paint() method is called whenever applet window is minimized or maximized or overwritten.

2) repaint() method

- If in the middle of some other method you realize that appearance of the component should change,then you should change the values of hte instance variables and call the components repaint() method.
- This method tells that it should redraw the component as soon as as it gets chance(by calling the paint() method)
- repaint() method requests a redraw of an item or an entire interface.It then calls **update()** method.
- An applet can call **repaint()** method directly to update the window whenever necesary.

repaint() method has four forms :

1. public void **repaint() :**
   This causes the entire window to be repainted.
2. public void **repaint() (**int **left** ,int **top** , int **width** ,int **height):**
   Repaints only the rectangular part of the screen defined by x,y width and height .
3. void **repaint**(long mililseconds)**:**
   - Tries to call update for the number of milliseconds in the input argument.
   - If update can't be called by the end of the specified time ,repaint gives up.No error is thrown if the method gives up.
   - This version of method used primarily for animations.
4. void **repaint(**long **time,**int **x,**int **y,**int **width,**int **height) :**
   - It combines previous two repaint() versions so that only a portion of hte screen is updated.
   - If it can't be done before the specified time elapses (in milliseconds),it is skipped.
   - When repaint() method is invoked on container ,such as an applet,the AWT takes care of invoking update on all items in container.

3) update() method:

- The system does not actually call the paint() method,actually update() called directly by the system.
- The built in update procedure first fills in the entire component with a background color.
- then it calls the paint() method ,the paint() method draws on a rectanglar area that has already been filled with the background color.
- update() method controls what happens when repaint is called;this method can be overriden.
- update() method improves drawing performance along with paint method.
- update() method clears the current display and calls the paint() method.
- Syntax:

```
public void update( Graphics g)
{
paint ( g) ;
}
```

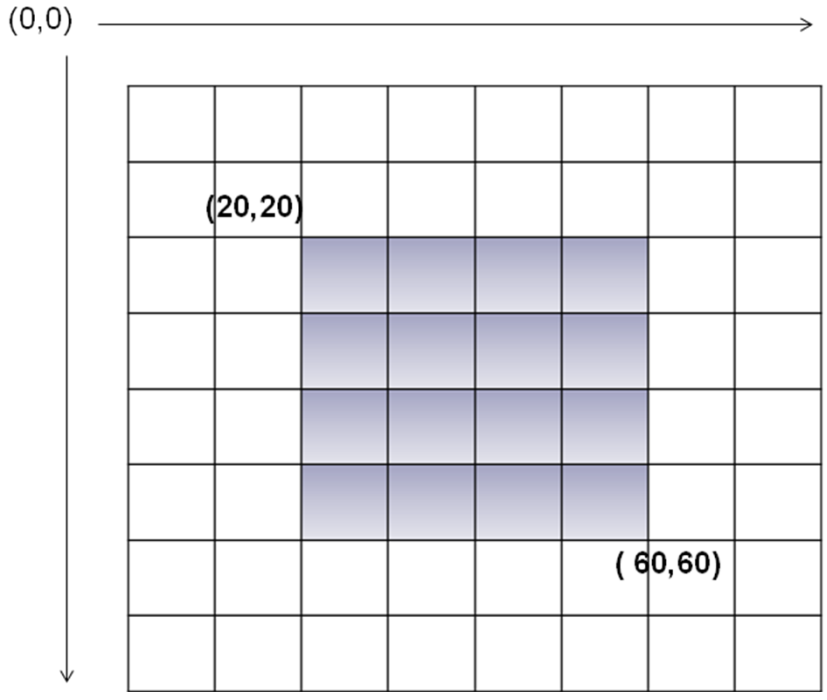
Example: update() ,patint() and repaint() method

```
import java.awt.*;
import java.applet.*;
/*<applet code=PaintMethod.class width=200 height=400>
</applet>*/

public class PaintMethod extends Applet implements Runnable
{
	int counter;
	Thread t;
	public void init()
	{
		counter=1;
		t=new Thread(this);
		t.start();
	}

	public void run()
	{
	try
		{
	while(true)
		{
		repaint();
		Thread.sleep(1000);
		counter=counter+1;
		setBackground(Color.red);

		}
		}
	catch(InterruptedException e)
		{
		}
	}
	public void update(Graphics g)
	{
		paint(g);
	}

	public void paint(Graphics g)
	{
		setBackground(Color.pink);
		showStatus("paint and update Method Demo");
		if(counter%2==0)
```

```
                    {
                    g.drawString( "Hello world ! ",100,counter);
                    }
                    else
                    {
                    g.drawString( "Hello java ! ",10,counter);
                    }
                }
        }
```

Graphics class

- **The Graphics class is part of AWT.**
- **Graphics class is contained in java.awt.Graphics** and **it includes methods for drawing types of shapes or text in variety of fonts.**
- Every applet has its own area of the screen known as **canvas.** where it creates its display.
- A Java applet draws graphical images inside its space using the coordinate system.
- Java's coordinate system has the origin **(0, 0)** in the upper-left corner.
- **Positive x** values are to the **right** and **positive y** values are to the **bottom.** The values of coordinates x and y are in pixels.
- Using drawing methods of class we can draw a shape on the screen.
- All the drawing methods have arguments representing end points, corners or starting location of shape as values in applet's coordinate system.



Canvas

Methods of Graphics class :

We have to create object of **Graphics** class to call these methods.

Like public void paint(Graphics g)

1. **clearRect**( ): Erases a rectangular area of canvas

Syntax: **g.clearRect (**int **x**, int **y**, int **width**, int **height**)

This method cut the portion specified in parameters

2. **copyArea( ):** Copies a rectangular area of the canvas to other area.

Syntax: **g.copyArea (**int **x,** int **y,** int **width, int height,** int **new_x,** int **new_y)**

Copies rectangular area defined by **x, y, width, height t**o a rectangle defined by **new_x, new_y**(new location) width and height in the same graphics objects.

3. **drawLine( ):** Draws a straight line.

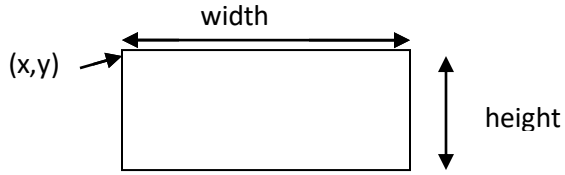
Syntax: **g.drawLine (**int **x1,** int **y1,** int **x2,** int **y2)**

It draws a line at specific location.

(X1,y1) (x2,y2)

4. **drawRect( ):** Draws a hollow rectangle.

Syntax: **g.drawRect (**int **x,** int **y**. int **width**, int height)



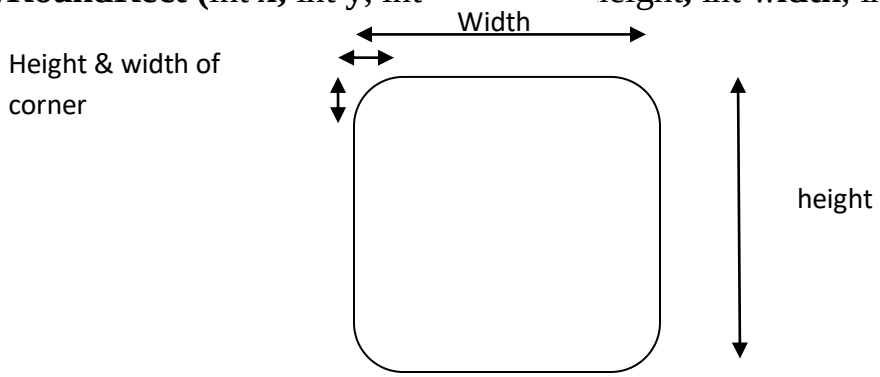
5. **fillRect( ):** fills a rectangle with default black colour

Syntax: **g.fillRect (**int **x,** int **y**, int **width**, int height)

6. **drawRoundRect( ):** Draws a hollow rectangle with round corner

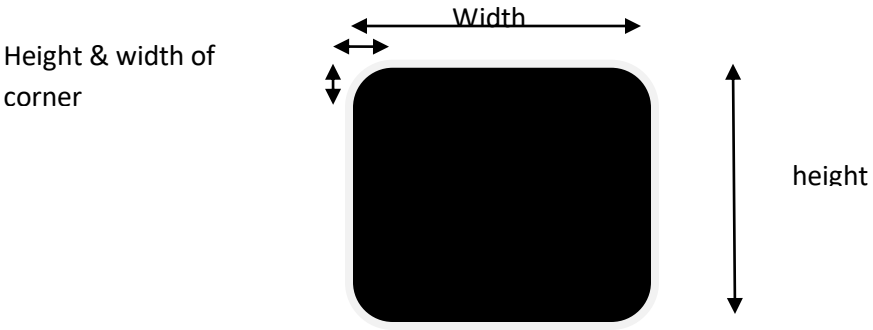
Syntax:

g.drawRoundRect (int **x,** int **y**, int **width**, int height**,** int **width**, int **height)**



7. fillRoundRect( ): Draws a hollow rectangle with round corner

Syntax: **g.fillRoundRect (**int **x,** int **y**, int **width**, int height**,** int **width**, int **height)**

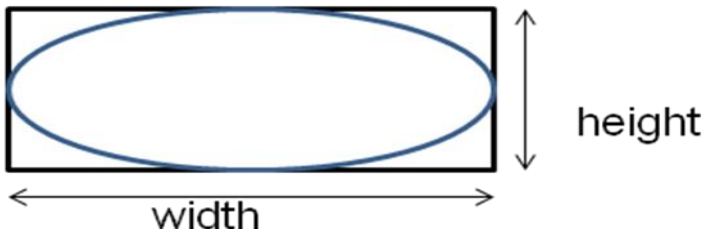


8. drawOval: Draws a hollow oval

Syntax: **g.drawOval (int x,int y,int width, int height)**

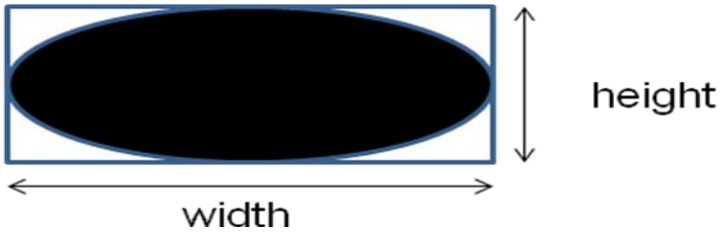
- Graphics class does not have any method for circles or ellipse.
- **drawOval ( )** method can be used to draw a circle or an ellipse.
- **To draw Circle:**If width and height both are same in draw oval method

Then it will draw a circle.



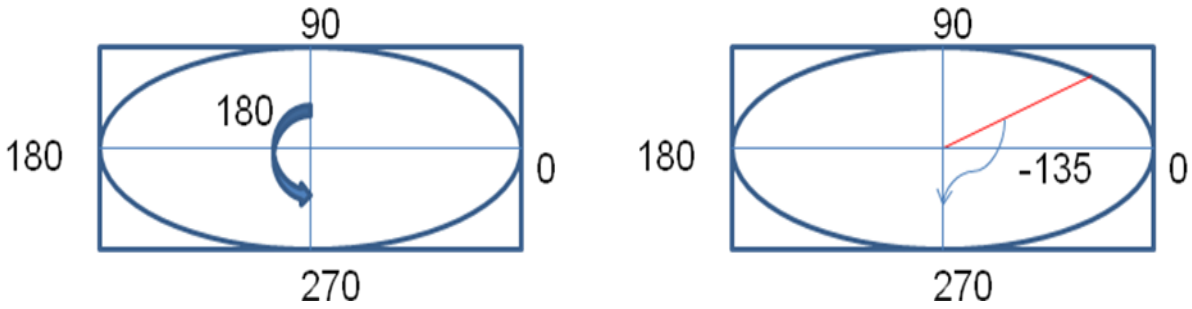
9. fillOval( ) : draws a filled Oval

Syntax: **g.fillOval(int x,int y,int width, int height)**



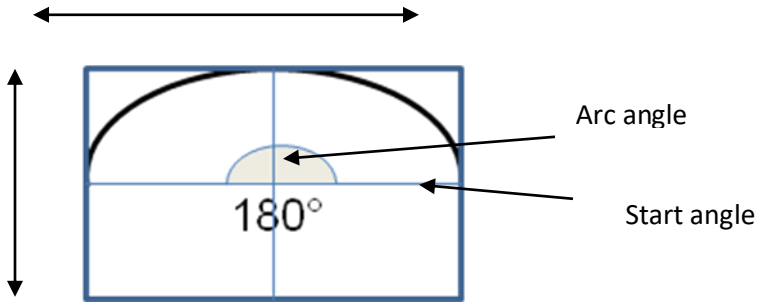
- An arc is a part of an oval. We can say oval is an series of arcs that are connected together in an orderly manner.
- **drawArc ()** is designed to draw arc which has six parameter. In which last two presents starting angle of the arc and the number of degrees (sweep angle) around the arc.
- Java considers the three O'clock as zero degree position and degree increases in anticlockwise direction.(see fig(a))
- We can draw in backward direction by specifying

E.g : If last argument is -135° and starting angle is 45° then arc is as in fig.(b)



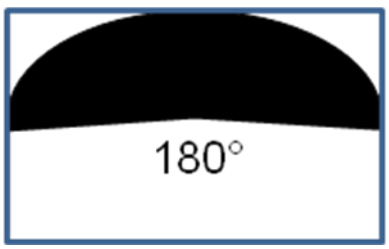
10. drawArc : draws a hollow arc

Syntax: **g.drawArc**(int **x**,int **y**,int **width**,int **height**,int **startangle**,int **arcangle**)



11. fillArc( ): draws a filled arc.

Syntax: g.**fillArc(**int **x**,int **y**,int **width**,int **height**,int **startangle**,int **arcangle**)



12. drawString(String msg, int x ,int y) : it draws /display string in output.

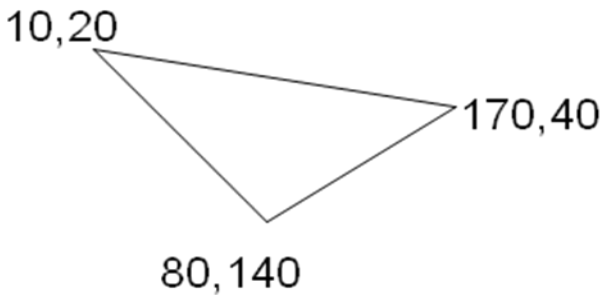
g.drawString(" hello ",10 ,100)

Draws a polygon:

- Polygons are shape with many sides. It is a set of lines connected together.
- Te end of the first line is the beginning of another line,the end of second line is the beginning of third and so on.
- We can draw polygon using **drawLine( ) ,drawPolygon()** and **fillPolygon()** method.

Polygon Using drawLine Method

```
public void paint( Graphics g)
 {
 g.drawLine(10,20,170,40);
 g.drawLine(170,40,80,140);
 g.drawLine(80,140,10,20);
 }
```



12. **drawPolygon(int x[ ],int y[ ], int n) :**
13. **fillPolygon(int x[ ],int y[ ], int n) :**

Here,

x[ ] is an array of integers containing ***x coordinates***

y[ ] is an array of integers containing **y *coordinates***

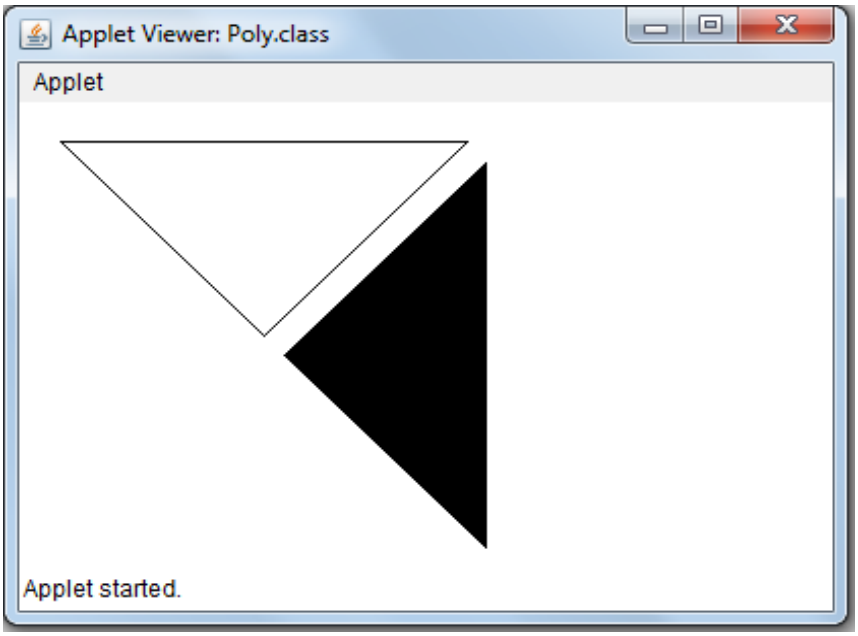
n is an integer for the total number of point.

(x and y arrays should be of same size and we must repeat the first point at the end of the array for the closing the polygon)

Example:

```
import java.applet.*;
import java.awt.*;
/*  <applet  code=Poly.class  width=400 height=400>
    </applet>        */
```

```
public class Poly extends Applet
{
    int x1[ ]={20,120,220,20};
    int y1[ ]={20,120,20,20};
        int n1=4;
    int x2[ ]={130,230,230,130};
    int y2[ ]={130,30,230,130};
        int n2=4;
public void paint(Graphics g)
{
    g.drawPolygon(x1,y1,n1);
    g.fillPolygon(x2,y2,n2);
}
}
```



We can use **Polygon** object to draw polygon.

Polygon class is useful if we want to add points to the polygon.

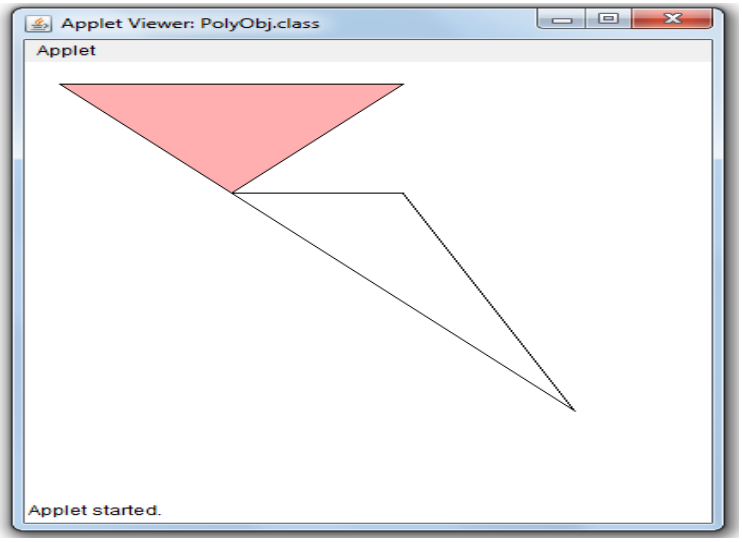
If we have polygon object existing then we can add point as follows:

```
poly.addPoint(20,20);
```

Example:

```
import java.applet.*;
import java.awt.*;
/*
<applet  code=PolyObj.class  width=400 height=400>
</applet>              */
public class PolyObj extends Applet
{
public void paint(Graphics g)
```

```
{
        int x[]={20,120,220,20};
        int y[]={20,120,20,20};
        int n= x. length;
        Polygon poly=new Polygon(x , y , n);
        g.setColor(Color.pink);
        g.fillPolygon(poly);
        g.setColor(Color.black);
        poly.addPoint(120,120);
        poly.addPoint(320,320);
        poly.addPoint(220,120);
        poly.addPoint(120,120);
        g.drawPolygon(poly);
    }
    }
```



14. **getColor( )** :get color of current shape
15. **setColor(Color c)** :sets Color c for next drawing shape
16. **getFont ( ) :**gets font type of your applet if you have set that.
17. **setFont( )** :to set fonts in your applet

    Font f=new Font(String fontname,int style ,int size);

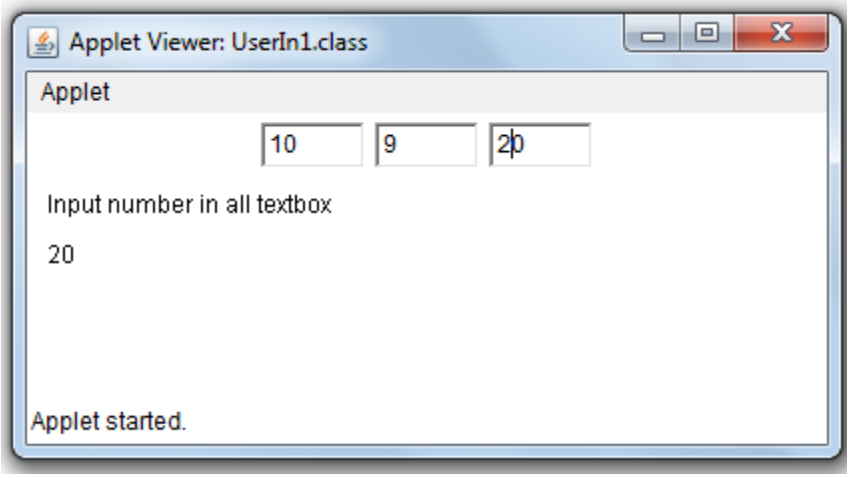
    Ex : **Font f=new Font("Times new Roman" , Font.BOLD, 20);**
18. **getFontMetrics( )** :to get all details related to fonts.

Example1:

Write a an applet which takes three input from user and finds bigger out of them.

```
/*<applet code=UserIn1.class width=400 height=600>
</applet>*/
import java.awt.*;
import java.applet.*;
public class UserIn1 extends Applet
{
TextField text1,text2,text3;

public void init()
{
    text1 = new TextField(4);
    text2 = new TextField(4);
    text3 = new TextField(4);
    add (text1);
    add (text2);
    add (text3);
    text1.setText("0");
    text2.setText("0");
    text3.setText("0");
}

public void paint(Graphics g)
{
int x=0,y=0,z=0;
String s1,s2,s3,s;
g.drawString("Input number in all textbox ",10,50);

        try
        {
            s1=text1.getText();
            x=Integer.parseInt(s1);

            s2=text2.getText();
            y=Integer.parseInt(s2);
            s3=text3.getText();
            z=Integer.parseInt(s3);
        }
        catch(Exception e)
        {
```

```
            }
        if(x>y && x>z)
        {
            s=String.valueOf(x);
        g.drawString(s,10,75);
        }
        else if( y>x && y>z)
        {
            s=String.valueOf(y);
            g.drawString(s,10,75);
        }
        else
        {
            s=String.valueOf(z);
            g.drawString(s,10,75);
        }
    }
    public boolean action(Event event ,Object obj)
    {
    repaint();
    return true;
    }
}
```

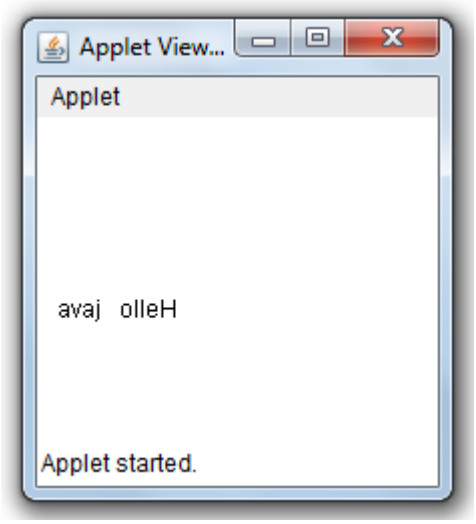


Example-2:

Write a applet program which reverse the string.

```
import java.awt.*;
import java.applet.*;
/*<applet code =reverse_str.java width=200 height=400>
</applet>*/
public class reverse_str extends Applet
 {
```

```
String str,s;
public void init()
{
        str=getParameter(" ");
        if(str==null)
              str="java";
              str="Hello   " + str;
        StringBuffer s=new StringBuffer( str);
        s=s.reverse();
        str=String.valueOf(s);
}
public void paint(Graphics g)
{
    g.drawString(str,10,100);
}
}
```

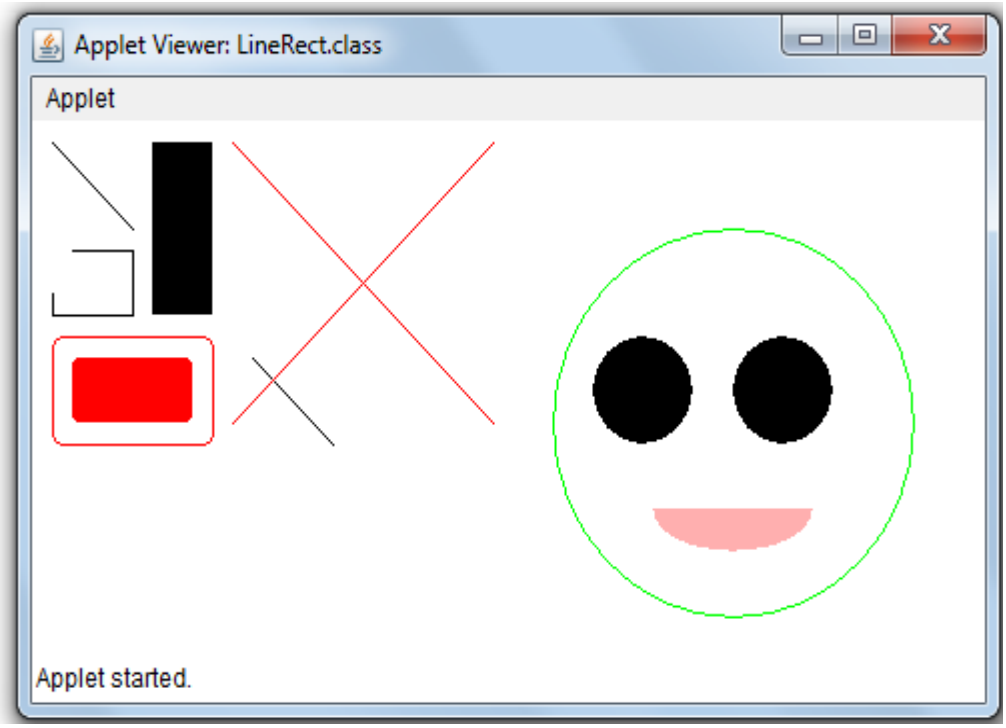


Example :3

Write an applet program which uses Graphics class methods.

```
import java.awt.*;
import java.applet.*;
/*<applet  code=LineRect.class  width=400 height=400>
</applet>
*/
public class LineRect extends Applet
{
public void paint(Graphics g)

{
```

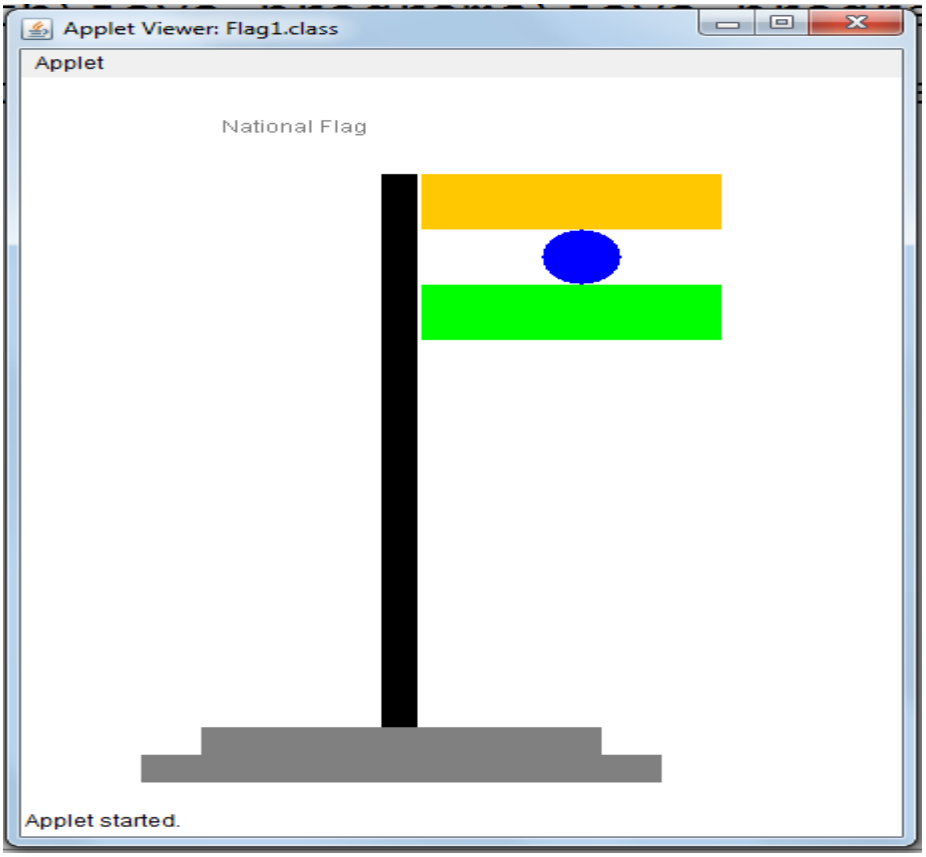
```
        g.drawLine(10,10,50,50);
        g.drawRect(10,60,40,30);
        g.clearRect(10,60,10,20);
        g.copyArea(10,10,50,50,100,100);
        g.fillRect(60,10, 30,80);
        g.setColor(Color.red);
        g.drawRoundRect(10,100,80,50,10,10);
        g.fillRoundRect(20,110,60,30,5,5);
        g.drawLine(100,10,230,140);
        g.drawLine(100,140,230,10);
        g.setColor(Color.green);
        g.drawOval(260,50,180,180);
        g.setColor(Color.black);
        g.fillOval(280,100,50,50);
        g.fillOval(350,100,50,50);
        g.setColor(Color.pink);
        g.fillArc(310,160,80,40,180,180);
}
}
```



Example 4:

Write an applet which draws an national flag.

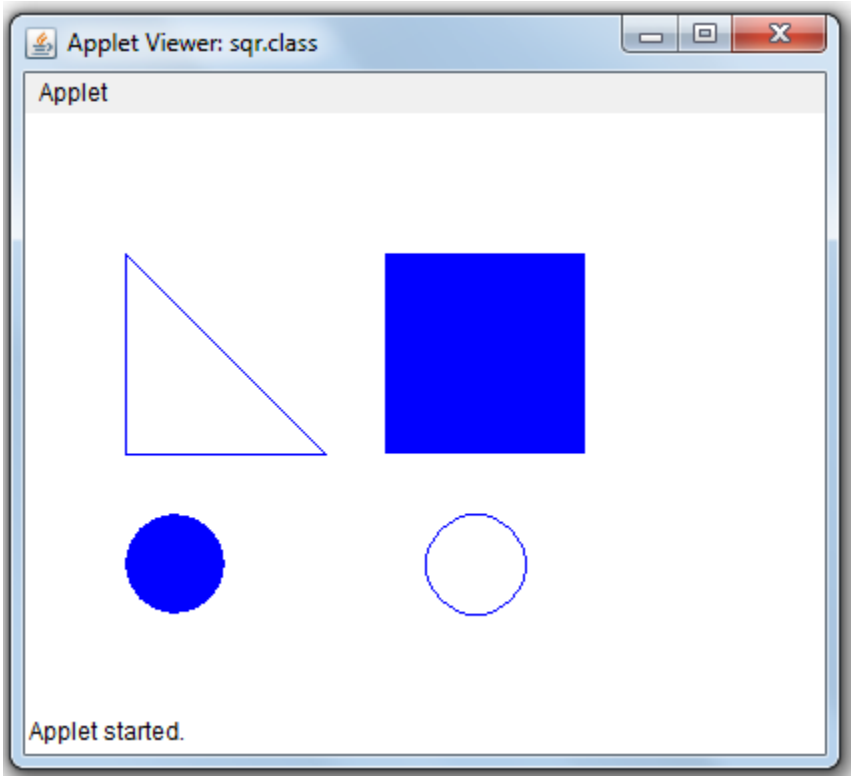
```
import java.awt.*;
import java.applet.*;
/*<applet code=Flag1.class width=800 height=500>
</applet>*/
public class Flag1 extends Applet
{
public void paint(Graphics g)
{
        g.fillRect(180,70,18,400);
        g.setColor(Color.orange);
        g.fillRect(200,70,150,40);
        g.setColor(Color.white);
        g.fillRect(200,120,150,40);
        g.setColor(Color.green);
        g.fillRect(200,150,150,40);
        g.setColor(Color.blue);
        g.fillOval(260,110,40,40);
        g.setColor(Color.gray);
        g.fillRect(90,470,200,20);
        g.fillRect(60,490,260,20);
        g.drawString("National Flag",100,40);
}
}
```



Example :5

Write an applet which draws circle ,square and right angle triangle.

```
import java.awt.*;
import java.applet.*;
/*<applet code=sqr.class width=800 height=800>
</applet>*/
public class sqr extends Applet
{
public void paint(Graphics g)
{
        g.setColor(Color.blue);
        g.fillRect(180,70,100,100);                //draws square

             //draws right angle triangle

        g.drawLine(50,70,50,170);
        g.drawLine(50,170,150,170);
        g.drawLine(150,170,50,70);

        //draws a circle
        g.drawOval(200,200,50,50);
        g.fillOval(50,200,50,50);
}


}
```



Example:

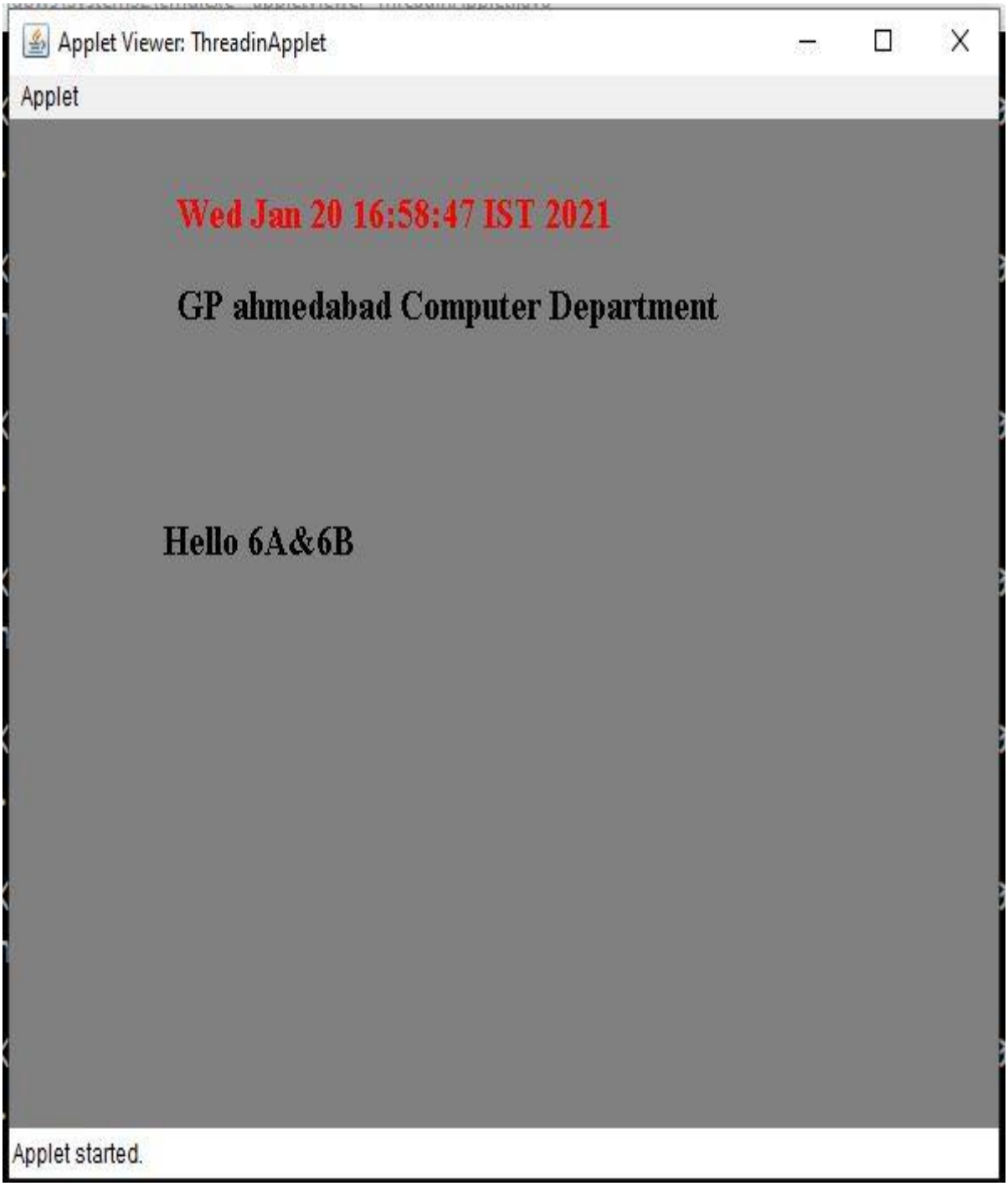
Write an applet program which add two values using <param> tag.

```
import     java.applet.*;
import     java.awt.*;
public class Paramsum extends Applet
{
String s1,s2,s;
int a,b,sum;
public void init()
{
s1=getParameter("value1");
s2=getParameter("value2");

a=Integer.parseInt(s1);
b=Integer.parseInt(s2);



}
public void paint(Graphics g)
{
     sum=a+b;
     s=String.valueOf(sum);
     g.drawString(s,10,100);
}
}


/*
<html>
<! parameterized HTML file>
<HEAD>
        <TITLE>Welcome to java Applets</TITLE>
</HEAD>
<BODY>
        <APPLET CODE=Paramsum.class width=400 height=400 >
                <PARAM NAME="value1" VALUE= "10">
                <PARAM NAME="value2" VALUE= "20">
        </APPLET>
</BODY>
</html>
*/
```

Write an applet code to display digital clock in applet using thread.

```
/*
<applet  code="ThreadinApplet" width=400 height=400>
</applet>
*/
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;
import java.util.Date;

public class ThreadinApplet extends Applet implements Runnable
{
    //Thread -->init(),start(),stop(),destroy(),suspend(),resume(),sleep()
    // init() start() stop() destroy() run()
    int x=0,y=0;
    Font ft;
    Thread th;
    int counter=0;
    Date dt;
    public void init()
    {
       ft=new Font("TimesRoman",Font.BOLD,20);
    }
    public void start() //applet method
    {
       if(th==null)
       {
       th=new Thread(this);
       th.start(); //thread method
       }
    }
    public void run()
    {
       while(true)
       {
              dt=new Date();
              repaint();
              try
              {
              Thread.sleep(1000);
              }
              catch(InterruptedException e)
              { }
       }
```

```
    }

    public void paint(Graphics g)
    {
       counter++;
       g.setFont(ft);
       g.drawString("Hello 6A&6B",x++,200);
       g.drawString("GP ahmedabad Computer Department",100,y++);
       g.setColor(Color.red);
       g.drawString(dt.toString(),100,50);
       g.setColor(Color.red);
       //g.fillRect(x++,y++,50,50);
       if(x>400)
       {
                    x=0;
       }
       if(counter>100)
       {
            if(counter%5==0)
            setBackground(Color.pink);
            else
            setBackground(Color.gray);
       }
       else
       {
            if(counter%5==0)
            setBackground(Color.green);
            else
            setBackground(Color.gray);
       }
    }
}
```


Applet Viewer: ThreadinApplet
Applet
Wed Jan 20 16:58:47 IST 2021
GP ahmedabad Computer Department
Hello 6A&6B
Applet started.

Question list

1. What is applet? Give difference between Applet and application.
2. Explain Local and Remote Applet.
3. Explain life cycle of an applet.
4. Explain structure of simple applet code with applet tag.
5. Explain applet tag with its attributes.
6. Explain passing parameter in <param> tag with example.
7. Explain methods of Graphics class.
8. Explain Applet class with any three methods.
9. Give advantages of applet.