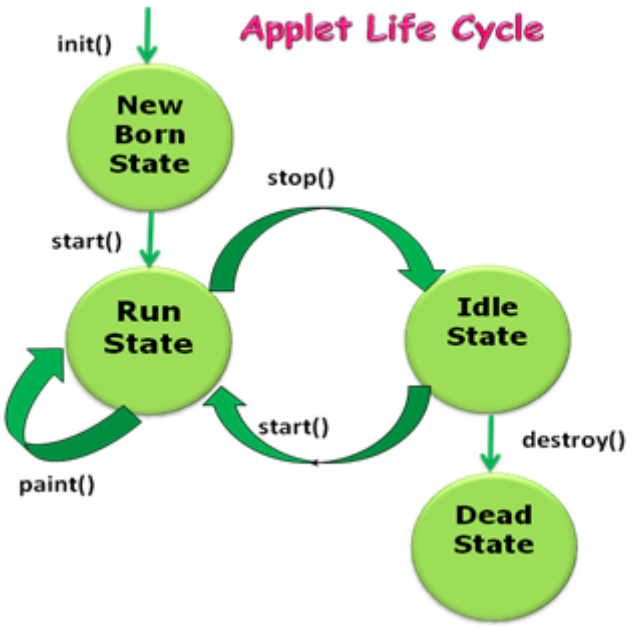
Applet

- An applet is a special Java program that runs in a browser enabled with Java technology (JRE) that can be downloaded from the internet and run.
- An applet is typically embedded inside a HTML page and runs in the context of a browser.
- An applet is designed to run remotely in the client browser, so there are some restrictions on it.The applet can't access system resources on the local computer.
- Applets are used to make the web site more dynamic and entertaining.
- An applet must be a subclass of the java.applet.Applet class. The Applet class provides the standard interface between the applet and the browser environment.

Life Cycle Of An Applet

These are the different stages involved in the life cycle of an applet.

- Initialization State
- Running State
- Idle or stopped state
- Dead State
- Display State



Initialization State

- The life cycle of an applet begins when the applet is first loaded into the browser and called the init() method.
- And the init() method is called exactly once in an applet's life when applet is first loaded.
- Basically the purpose of the init() method is to initialize the data fields and read the PARAM tag in the HTML file. It is retrieving the parameters in the param tag.

Syntax

```
public void init()
{
Statements
}
```

Running State

- This state is the second stage of the applet life cycle. After initialization, this state will automatically occur by invoking the start method of the applet class which again calls the run method and which calls the paint method.
- And it can occur from the Idle State when the applet is reloaded.
- This method runs multiple times.

Syntax

```
public void start()
{
Statements
}
```

stop State

- This state is the third stage of the applet life cycle; another name for it is Stopped state.
- There are two conditions when an applet moves to this state; when the currently executed applet is minimized or when the user switches to another page. At this point the stop method is invoked.
- It is called at least once when the browser leaves the web page when we move from one page to another.
- This method is called only in the Running State and can be called any number of times.

Syntax

```
public void stop()
{
Statements
}
```

Dead State

- This state is the fourth stage of the applet life cycle. When the applet program terminates, the destroy function is invoked which makes an applet to be in the Dead State.
- This stage occurs when the destroy() method is called. And this method is called only one time in the whole life cycle.

Syntax

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

Display State

- After the paint method is called the applet is said to be in the Display State.
- The applet is said to be in the Display State when the paint method is called. This method can be used when we want to display output in the screen paint() method.
- It is required in all applets when we want to draw something on the applet window. And the paint method takes a Graphics object as an argument.

Syntax

```
public void paint(Graphics g)
{
Statements
}
```

Advantages of Applet

- The biggest advantage of applets is Java's extensive detail to the user interface. Applets allow more complex user interface options than HTML combined with CGI.
- Applets are cross-platform and can run on the Windows, Mac OS and Linux platforms.
- Applets are to guaranteed to be safe; applets are downloaded into a "safe space" so they can't hurt the client PC.
- They are limited to certain operations on the browser, but can also use JDBC connections (to a database) or distributed objects.

Disadvantages of Java Applet

- All the Browser required Java plug-to run applet
- Java applets are compiled and run on the client side so the first time it takes a significant startup time.
- If an applet is not stored in the browser cached memory then it will be downloaded from the internet and that will take time.
- And the design of an applet is difficult compared to designing a good user interface in HTML.

```
//Applet program without using html file
import java.awt.*;
import java.applet.*;
public class MyApplet extends Applet
{
public void init()
{
```

```
System.out.println("Applet initialized");
}
public void start()
{
System.out.println ("Applet execution started");
}
public void stop()
{
System.out.println ("Applet execution stopped");
}
public void paint(Graphics g)
{
setBackground(Color.CYAN);
g.setColor(Color.BLUE);
g.drawString("welcome to my applet",20,20);
showStatus ("Painting...");
}
public void destroy()
{
showStatus ("Applet destroyed");
}
}
/* <applet code="MyApplet.class" height=300 width=300>
</applet>*/
```

- ✔ Import java.awt package and java.applet package
- ✔ The applet class must be public
- ✔ The class must be sub class of java.applet.Applet class
- ✔ Override paint() method to display

Applet class file must be included in html code

```
/* <applet code="MyApplet.class" height=300 width=300>
</applet>*/
```

How to run

Compile:

Javac MyApplet.java

Run:

Appletviewer MyApplet.java

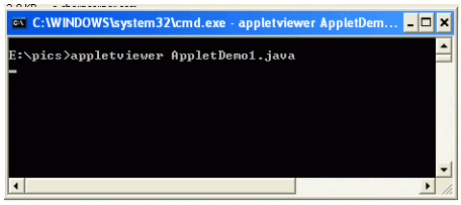
Example

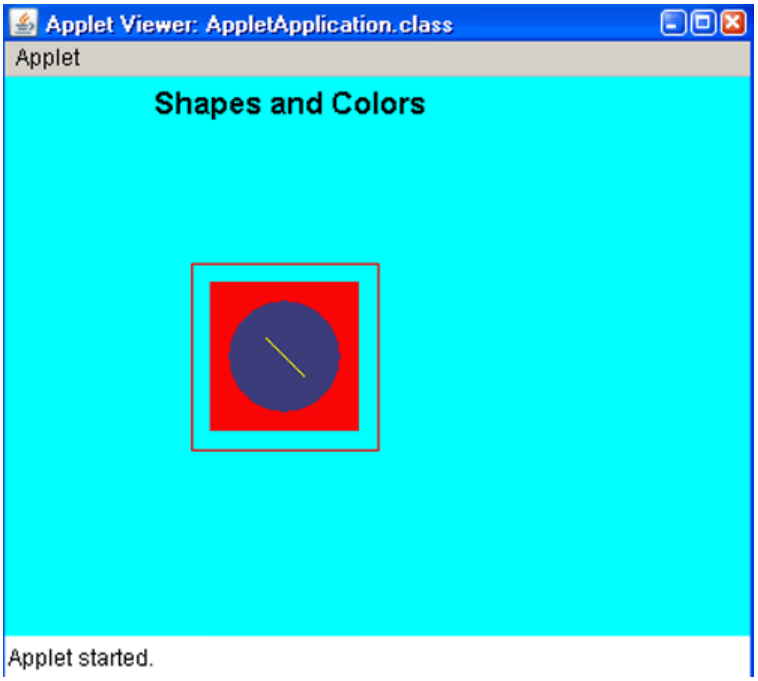
<applet code="AppletDemo1.class" width="450" height="450">

```
</applet>
import java.awt.*;
import java.applet.*;
public class AppletDemo1 extends Applet
{
   Font bigFont;
   Color redColor;
   Color weirdColor;
   Color bgColor;
   public void init()
   {
      bigFont = new Font("Arial", Font.BOLD, 16);
      redColor = Color.red;
      weirdColor = new Color(60, 60, 122);
      bgColor = Color.cyan;
      setBackground(bgColor);
   }
   public void stop()
   {
   }
   public void paint(Graphics g)
   {
      g.setFont(bigFont);
      g.drawString("Shapes and Colors", 80, 20);
      g.setColor(redColor);
      g.drawRect(100, 100, 100, 100);
      g.fillRect(110, 110, 80, 80);
      g.setColor(weirdColor);
      g.fillArc(120, 120, 60, 60, 0, 360);
      g.setColor(Color.yellow);
      g.drawLine(140, 140, 160, 160);
```

```
        g.setColor(Color.black);
    }
}
```

OUTPUT





To Change The Font Style Using Font Class In Applet Using Java Program

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

public class font extends Applet
 {
  Font f1,f2,f3;

  public void  init()
   {
     f1 = new Font("Arial",Font.BOLD,18);
     f2 = new Font("Forte",Font.PLAIN,24);
     f3 = new Font("Elephant",Font.ITALIC,28);
   }

  public void  paint(Graphics g)
```

```
  {
    g.drawString("Senthil",50,50);

    g.setFont(f1);
    g.drawString("Velan",50,80);

    g.setFont(f2);
    g.drawString("Vicky",50,110);

    g.setFont(f3);
    g.drawString("Vignesh",50,140);
  }
}

/* <applet code = "font.class" height = 500 width =500>
  </applet> */
```

Graphics in Applets

The java.awt.Graphics class provides many methods for graphics programming.

Some commonly used public methods in the Graphics class:

1. **abstract void setColor(Color clr)** sets the Graphics current color to the specified color.
2. **abstract void drawString(String strng, int a, int b)** draws the specified string.
3. **void drawRect(int a, int b, int wdth, int hgt)** draws a rectangle with the specified width and height.
4. **abstract void fillRect(int a, int b, int wdth, int hgt)** fills a rectangle with the default color and specified width and height.
5. **abstract void drawOval(int a, int b, int wdth, int hgt)** draws an oval.
6. **abstract void fillOval(int a, int b, int wdth, int hgt)** fills an oval with the default color.
7. **abstract boolean drawImage(Image img, int a, int b, int hgt, ImageObserver obsrvr)** draws the specified image.
8. **abstract void drawArc(int a, int b, int wdth, int hgt, int startAngle, intarcAngle)** draws a circular or elliptical arc.
9. **abstract void fillarc(int a, int b, int wdth, int hgt, int startAngle, int arcAngle)** fills a circular or elliptical arc.
10. **abstract void drawLine(int a1, int b1, int a2, int b2)** draws a line.
11. **abstract void setFont(Font fnt)** sets the current graphics font to the specified font.

Draw a simple string in Java

In this program, we show how to design a string in graphics programming.

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

```
 {
  grp.setColor(Color.blue);
  grp.drawString("welcome-to-appletworld-in-java", 150, 150);
 }
}
/*
<applet code="GrpStringEx.class" width="400" height="400">
</applet>
*/
```

Output

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

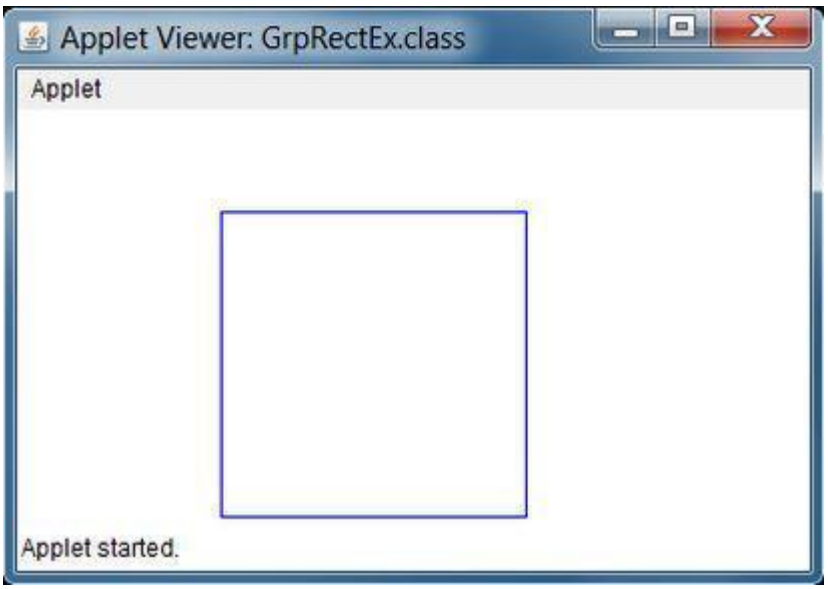
javac GrpStringEx.java
appletviewer GrpStringEx.java

Draw a rectangle in Java

In this example, we draw a simple rectangle.

```
import java.awt.*;
import java.applet.Applet;
public class GrpRectEx extends Applet
{
 public void paint(Graphics grp)
 {
  grp.setColor(Color.blue);
  grp.drawRect(100, 50, 150, 150);
 }
}
/*
<applet code="GrpRectEx.class" width="400" height="400">
</applet>
*/
```

Output

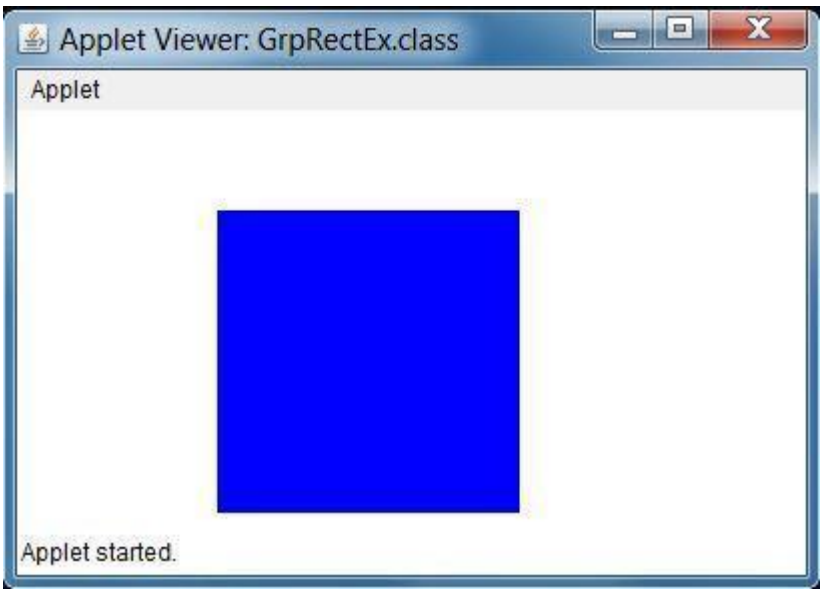


Fill color in a rectangle in Java

In this example, we fill in the color of our rectangle, usnig the grp.fillRect method.

```
import java.awt.*;
import java.applet.Applet;
public class GrpRectEx extends Applet
{
 public void paint(Graphics grp)
 {
  grp.setColor(Color.blue);
  grp.drawRect(100, 50, 150, 150);
  grp.fillRect(100, 50, 150, 150);
 }
}
/*
<applet code="GrpRectEx.class" width="400" height="400">
</applet>
*/
```

Output

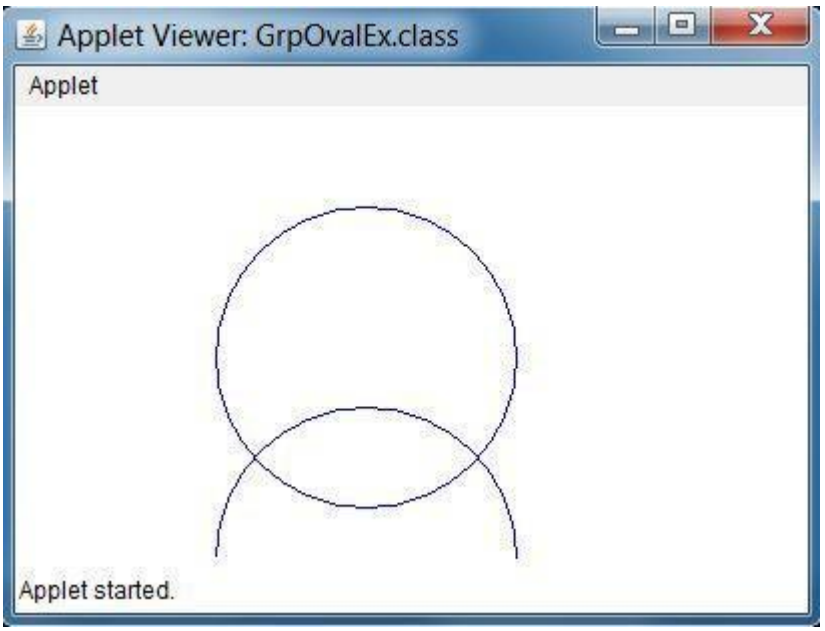


Draw an oval and an arc in Java

In this example, we create an oval and an arc with specified co-ordinates.

```
import java.awt.*;
import java.applet.Applet;
public class GrpOvalEx extends Applet
{
 public void paint(Graphics grp)
 {
  grp.setColor(Color.blue);
  grp.drawOval(100, 50, 150, 150);
  grp.drawArc(100, 150, 150, 150, 0, 180);
 }
}
/*
<applet code="GrpOvalEx.class" width="400" height="400">
</applet>
*/
```

Output

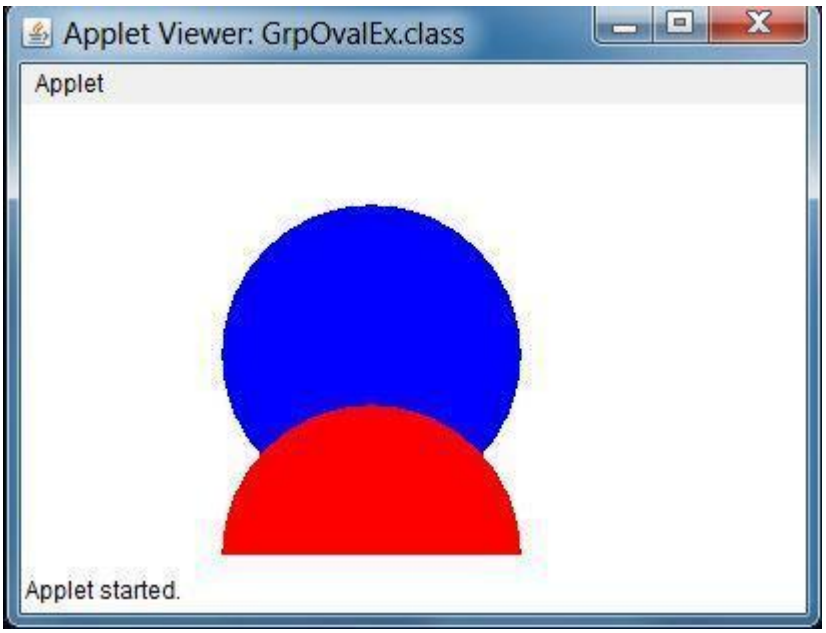


Fill color in oval and arc in Java

In this example, we fill an oval and an arc with a color, using the fillOval and fillArc methods.

```
import java.awt.*;
import java.applet.Applet;
public class GrpOvalEx extends Applet
{
 public void paint(Graphics grp)
 {
  grp.setColor(Color.blue);
  grp.fillOval(100, 50, 150, 150);
  grp.setColor(Color.red);
  grp.fillArc(100, 150, 150, 150, 0, 180);
 }
}
/*
<applet code="GrpOvalEx.class" width="400" height="400">
</applet>
*/
```

Output

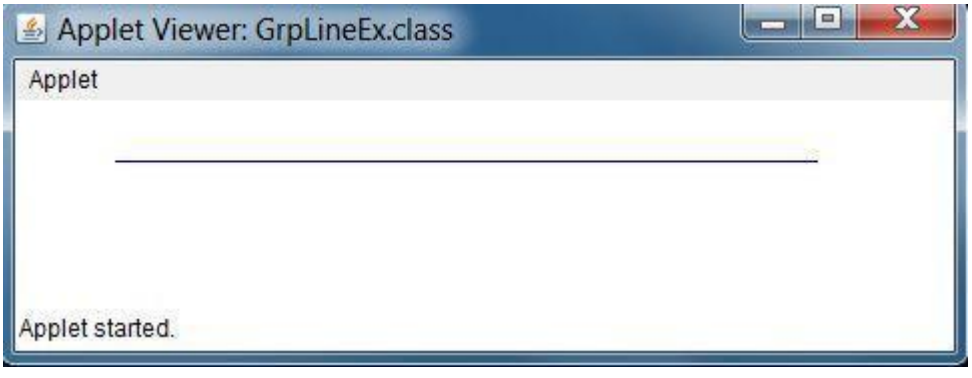


Draw a line in Java

In this example, we draw a simple line with specified co-ordinates.

```
import java.awt.*;
import java.applet.Applet;
public class GrpLineEx extends Applet
{
 public void paint(Graphics grp)
 {
  grp.setColor(Color.blue);
  grp.drawLine(50, 30, 400, 30);
 }
}
/*
<applet code="GrpLineEx.class" width="400" height="400">
</applet>
*/
```

Output



Displaying Image in Java

Applets are widely used in animation and games. For this, we require images to be displayed.

The Graphics class of Java, provides a drawImage() method to display an image.

Syntax

The syntax of the drawImage function to draw the specified image is:

public abstract boolean drawImage(Image image, int a, int b, ImageObserver observer)

Loading and Displaying Images

Images provide a way to augment the aesthetic appeal of a Java program. Java provides support for two common image formats: GIF and JPEG. An image that is in one of these formats can be loaded by using either a URL, or a filename.

How to get the object of the image

The java.applet.Applet class provides a method getImage() that returns the image object.

Syntax

The syntax of the getImage funtion to draw the specified image, is:

```
public Image getImage(URL url. String image)

{  }
```

Some other required methods are:

1. public URL getCodeBased() returns the base URL.

2. public URL getDocumentBase() returns the URL of the document in which the Applet is embedded.

Example

We must insert this image:

In this example, we show how to insert the image in an Applet. We use the drawImage() method to display the image.

```
import java.applet.*;

import java.awt.*;

public class ImageGrpcsEx extends Applet

{

   Image pic;
```

```
   public void init()
   {
      pic = getImage(getDocumentBase(), "Koala.jpeg");
   }
   public void paint(Graphics grp)
   {
      grp.drawImage(pic, 100, 30, this);
   }
}
/*
<applet code="ImageGrpcsEx.class" width="400" height="400">
</applet>
*/
```

Animation in Applet

Applets are widely used in animation and games, for this we need images that must be moved.

Example of animation

```
import java.awt.*;
import java.applet.*;
public class AnimationEx extends Applet
{
   Image pic;
   public void init()
   {
      pic = getImage(getDocumentBase(), "images.jpeg");
   }
   public void paint(Graphics grp)
   {
      for (int i = 50; i < 600; i=i+10)
      {
         grp.drawImage(pic, i, 30, this);
         try
         {
            Thread.sleep(400);
         } catch (Exception e) {}
      }
   }
}
/*
```

```
<applet code="AnimationEx.class" width="400" height="400">
</applet>
*/
```

Text animation

```
import java.awt.*;
import java.applet.Applet;
public class helloan extends Applet implements Runnable {
                    int x = 0;
                    int y = 0;
                    int width = 0;
                    Thread my_thread = null;
public void init() {
          x = size().width;
          y = size().height / 2;
          width = x;
          }
 public void start()
     {
      my_thread = new Thread(this);
      my_thread.start();
     }
  public void run()
     {
      while(true)
        {
          repaint();
          x -= 10;
          if(x < 0)
            x = width;

          try
            {
              Thread.sleep(100);
            }
          catch(InterruptedException e)
            {
            }
        }
     }
 public void paint(Graphics g)
     {
      g.drawString("Hello World!", x, y);
     }
}

/*<applet code="helloan.class" width=500 height=500>
</applet>*/
```