Java Applet

Presentation · December 2019

DOI: 10.13140/RG.2.2.12247.62886

Hana Esmaeel
Al-Nahrain University
25 PUBLICATIONS **16** CITATIONS

Java Applet

BY

Senior lecturer Hana Rashied Esmaeel

Abstract: An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.
Applets are used to make the web site more dynamic and entertaining.

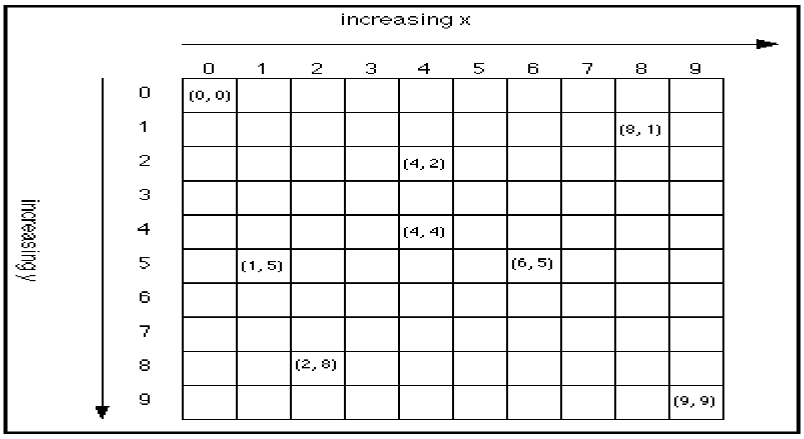
1-What is an Applet?

Definition: An applet is a small program that is run inside a web browser. And is a secure program that runs A subclass of java.applet.Applet .

The differences between an apple and an application are as follows:

Applications	**Applets**
Applications have the main( ) method	Applets don't have the main ( ) method.
Java application does not require an external viewer program. This means that you can execute a java application directly using the java interpreter.	Java applets require external viewer programs, so to use an applet; you need web browser or an applet viewer.
Application can access the local file system and resources.	Applet cannot access the local file system and resources
Java application generally refers to an application that is designed stand alone use.	Applets are designed to be embedded within an HTML page. (i.e.) designed for use on the world wide web (www)

2-The Coordinate System: Java uses the standard, two-dimensional, computer graphics coordinate system. The first visible pixel in the upper left-hand corner of the applet canvas is (0, 0). Coordinates increase to the right and down.



3-Graphics Objects: In Java all drawing takes place via a Graphics object. This is an instance of the class java.awt.Graphics.
Initially the Graphics object you use will be the one passed as an argument to an applet's paint ( ) method. Each Graphics object has its own coordinate system, and all the methods of Graphics including those for drawing Strings, lines, rectangles, circles, polygons and more. You get access to the Graphics object through the paint (Graphics g) method of your applet.

4-The Basic Applet Life Cycle

All applets have the following four methods:

1-public void in it ( ): The init () method is called exactly once in an applet's life, when the applet is first loaded. It's normally used to read PARAM tags, start downloading any other images or media files you need, and set up the user interface. Most applets have init () methods.

2-public void start ( ): The start () method is called at least once in an applet's life, when the applet is started or restarted. In some cases it may be called more than once. Many applets you write will not have explicit start () methods and will merely inherit one from their super class. A start () method is often used to start any threads the applet will need while it runs.

3-public void stop ( ): The stop () method is called at least once in an applet's life, when the browser leaves the page in which the applet is embedded. The applet's start () method will be called if at some later point the browser returns to the page containing the applet. In some cases the stop () method may be called multiple times in an applet's life. Many applets you write will not have explicit stop () methods and will merely inherit one from their super class. Your applet should use the stop () method to pause any running threads. When your applet is stopped, it *should* not use any CPU cycles.

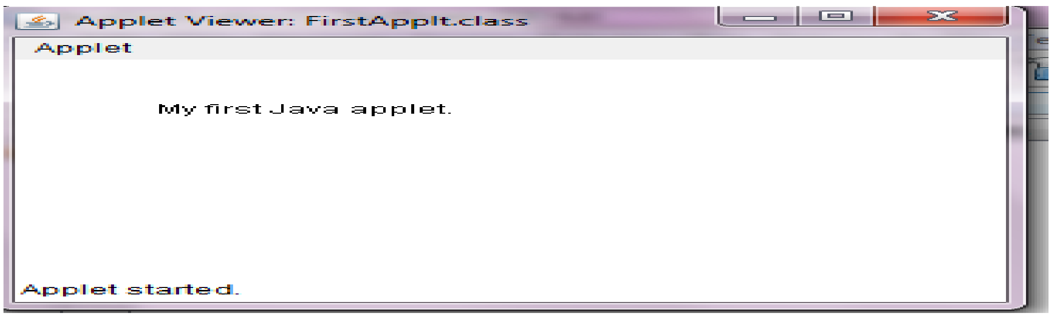
4- Public void destroy ( ): The destroy () method is called exactly once in an applet's life, just before the browser unloads the applet. This method is generally used to perform any final clean-up. Many applets will not have explicit destroy ( ) methods and just inherit one from their super class.
For example, in a video applet, the init () method might draw the controls and start loading the video file. The start () method would wait until the file was loaded, and then start playing it. The stop () method would pause the video, but not rewind it. If the start () method were called again, the video would pick up where it left off; it would not start over from the beginning. However, if destroy were called and then init (), the video would start over from the beginning.

5-Shapes: The *Graphics* class includes a large number of instance methods for drawing various shapes, such as lines, rectangles, and ovals. The shapes are specified using the (x,y) coordinate system described above. They are drawn in the current drawing color of the graphics context. The current drawing color is set to the foreground color of the component when the graphics context is created, but it can be changed at any time using the setColor() method.
Here is a list of some of the most important drawing methods. all these methods are in the *Graphics* class, so they all must be called through an object of type *Graphics*.
1-drawString(String str, int x, int y) - Draws the text given by the string str. The string is drawn using the current color and font of the graphics context. x specifies the position of the left end of the string. y is the y-coordinate of the baseline of the string .as shown in this example:

Example (1) : Write java applet program to display the string “My first Java applet” on the applet viewer with x=50,y=50, fig(1) show the output of the program.

Solution:

```
import java.applet.*;
import java.awt.Graphics;
public class FirstApplt extends Applet {
public void paint(Graphics g) {
      g.drawString("My first Java applet.",50,50);
  }
}
```



```
My first Java applet.
```

Fig(1) draw string on applet

2-Drawing Lines:Drawing straight lines with Java is easy. Just call

g.drawLine(x1, y1, x2, y2)

where (x1, y1) and (x2, y2)are the endpoints of your lines and g is the Graphicsobject you're drawing with.

Example (2): This program draws a line diagonally across the applet x1=25,y1=25,x2=75,y2=75 as shown in fig(2) below.

Solution:

```
import java.awt.Graphics;

public class MyLine extends java.applet.Applet {

    public void paint(Graphics g) {

        g.drawLine(25,25,75,75);  }}
```

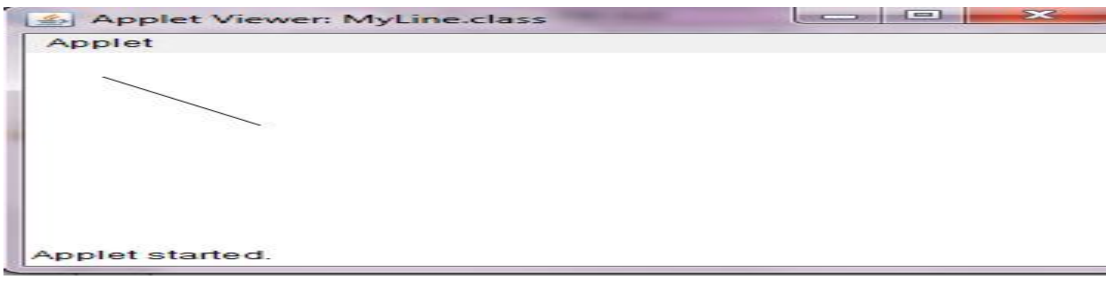


Fig (2) draw line on applet

3-Drawing Rectangles

Drawing rectangles is simple. Start with a Graphics object g and call its drawRect() method:

public void drawRect(int x, int y, int width, int eight)

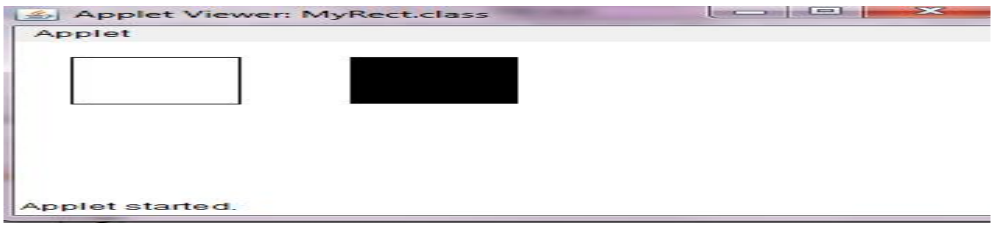
As the variable names suggest,
1-The first int is the left hand side of the rectangle.
2-The second is the top of the rectangle
3-The third is the width .
4-The fourth is the height.

4-Filling Rectangles

fillRect (int x, int y, int width, int height) -- Draws a filled-in rectangle.

Example (3):Write a program which draw rectangle and fill rectangle as shown in figure(3) below.

```
import java.awt.Graphics;
public class MyRect extends java.applet.Applet {
   public void paint(Graphics g) {
      g.drawRect(20,20,60,60);
      g.fillRect(120,20,60,60); }}
```



Fig(3) draw rectangle and fill rectangle in an applet

5- drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) To get an arc of a circle, make sure that width is equal to height.

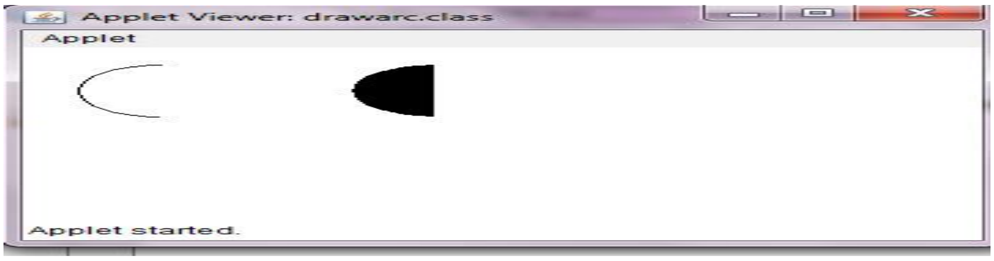
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)

Example (4): Write a program which draw arc and fill arc as shown in figure (4) below.

Solution:

```
import java.awt.Graphics;
public class drawarc extends java.applet.Applet {
   public void paint(Graphics g) {
      g.drawArc(20,20,60,60,90,180);
```

```
        g.fillArc(120,20,60,60,90,180) }}
```



Fig(4) draw arc and fill arc in an applet

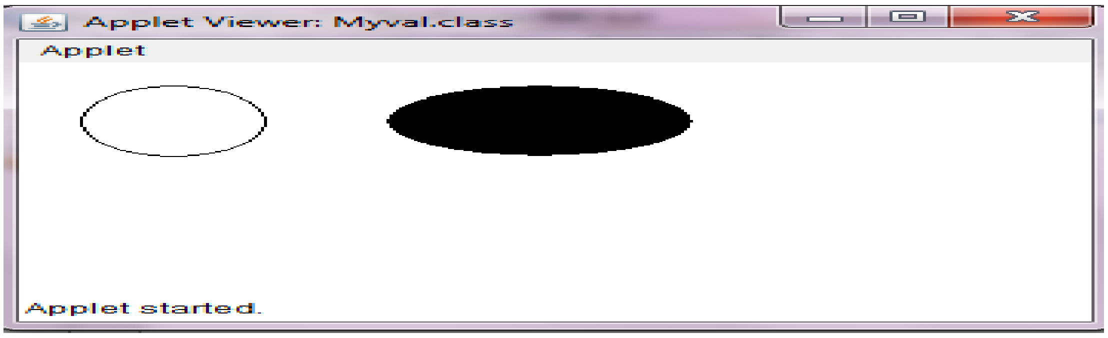
6-Ovals :Java has methods to draw outlined and filled ovals. these methods are called drawOval( )and fillOval( )respectively i.e :

```
public void drawOval(int left, int top, int width, int height)
public void fillOval(int left, int top, int width, int height)
```

Example (5): draw oval as shown in figure(5) below.

Solution:

```
import java.awt.Graphics;
public class Myval extends java.applet.Applet {
    public void paint(Graphics g) {
        g.drawOval(20,20,60,60);
        g.fillOval(120,20,100,60);
    }}
```

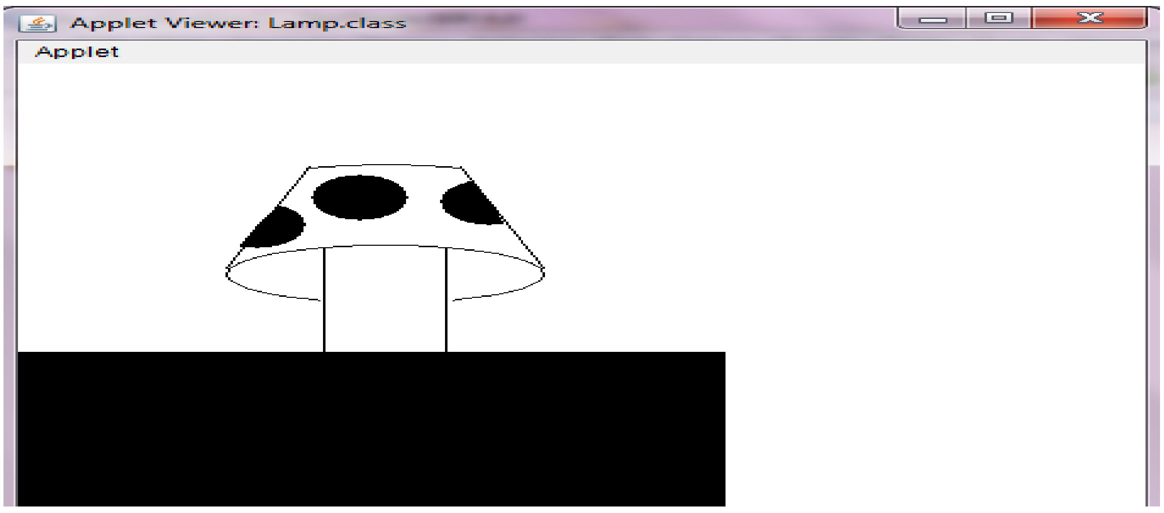


Fig(5) draw oval and fill oval in an applet

Example (6) write a program to draw the lamp as in figure (6) below:

Solution:

```
import java.awt.*;
  public class Lamp extends java.applet.Applet {
       public void paint(Graphics g) {
         // the lamp platform
         g.fillRect(0,250,290,290);
          // the base of the lamp
        g.drawLine(125,250,125,160);
        g.drawLine(175,250,175,160);
         // the lamp shade, top and bottom edges
         g.drawArc(85,157,130,50,-65,312);
         g.drawArc(85,87,130,50,62,58);
          // lamp shade, sides
         g.drawLine(85,177,119,89);
         g.drawLine(215,177,181,89);
          // dots on the shade
         g.fillArc(78,120,40,40,63,-174);
        g.fillOval(120,96,40,40);
         g.fillArc(173,100,40,40,110,180);
    } }
```



Fig(6) draw the lamp in an applet

6-Fonts :A font represents a particular size and style of text. The same character will appear different in different fonts.

In Java, a font is characterized by a **font name, a style, and a size**.

1-name: The available font names are system dependent, but you can always use the following four strings as font names:

"Times New Roman", "Arial", " Courier ", and "Dialog".

2-The style of a font is specified using named constants that are defined in the *Font* class. You can specify the style as one of the four values:

- Font.PLAIN,
- Font.ITALIC,
- Font.BOLD, or
- Font.BOLD + Font.ITALIC.

3-The size of a font is an integer. Size typically ranges from about 10 to 36, although larger sizes can also be used. The size of the default font is 12.

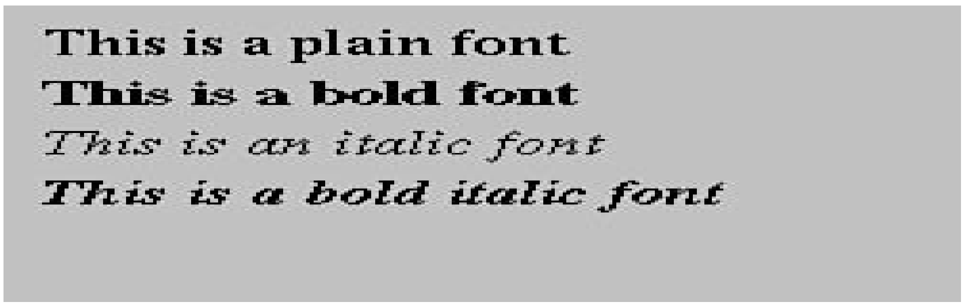
Java uses the class named **java.awt.Font** for representing fonts. You can construct a new font by specifying its font name, style, and size in a constructor:

```
Font  plainFont =  new Font("Serif", Font.PLAIN, 12);
Font  bigBoldFont = new Font("SansSerif", Font.BOLD, 24);
```

Every graphics context has a current font, which is used for drawing text. You can change the current font with the setFont() method. For example, if g is a graphics context and bigBoldFont is a font, then the command g.setFont(bigBoldFont) will set the current font of g to bigBoldFont.

Example (7) This Example Use Many Different Fonts As Shown In Figure (7) Below
Solution:

```
   import java.awt.Font;
   import java.awt.Graphics;
    public class ManyFonts extends java.applet.Applet {
    public void paint(Graphics g) {
    Font f = new Font("TimesRoman", Font.PLAIN, 18);
    Font fb = new Font("TimesRoman", Font.BOLD, 18);
    Font fi = new Font("TimesRoman", Font.ITALIC, 18);
    Font fbi = new Font("TimesRoman", Font.BOLD + Font.ITALIC, 18);
    g.setFont(f);
    g.drawString("This is a plain font", 10, 25);
    g.setFont(fb);
    g.drawString("This is a bold font", 10, 50);
    g.setFont(fi);
    g.drawString("This is an italic font", 10, 75);
    g.setFont(fbi);
    g.drawString("This is a bold italic font", 10, 100);
 } }
```



```
This is a plain font
This is a bold font
This is an italic font
This is a bold italic font
```

Fig (7) many different font

7- Using the Color Class: The simplest way to display a color in a Java program is to use one of the constant variables from the Color class. You can refer to the following constants:

black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, and yellow.

In a panel or applet window, you can set the background color of the component using these constants. The following is an example:

setBackground(Color.orange);

When you want to display text of a certain color or draw other graphics in different colors, you have to use a method that sets up the current color. Unlike the setBackground() method, which can be called directly on a container such as a panel or applet, the setColor() method must be used on an object that can handle a color change.

Example (8): Write java applet program that display the message "Hello again!" with red color and font TimesRoman size 36 ,as shown in fig(8) below:

Solution:

```
 import java.awt.Graphics;
 import java.awt.Font;
 import java.awt.Color;

 public class HelloAgainApplet extends java.applet.Applet
{

 Font f = new Font("TimesRoman",Font.BOLD,36);

 public void paint(Graphics g)
{
 g.setFont(f);
 g.setColor(Color.red);
 g.drawString("Hello again!", 5, 50);

}
}
```

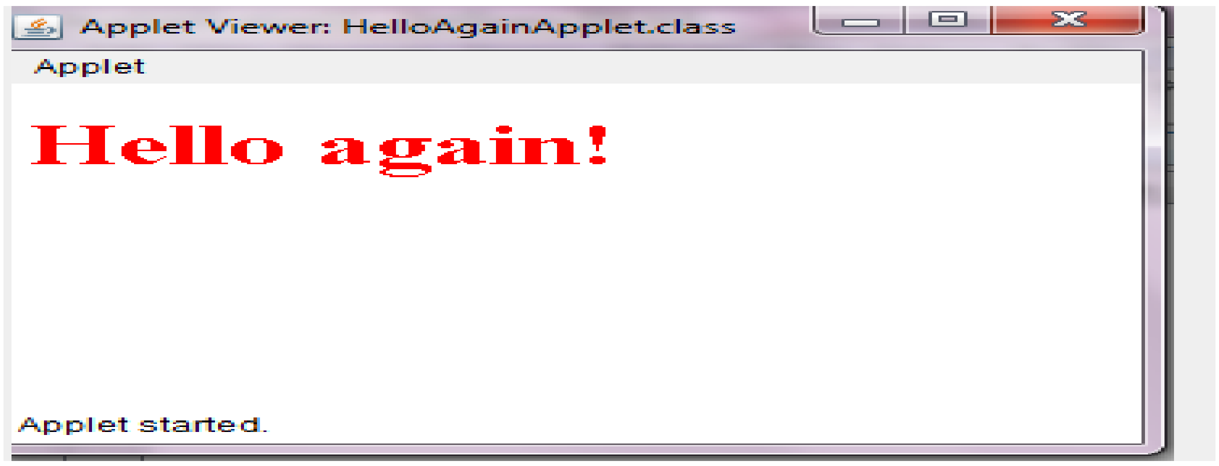


Fig (8) Display message with red color

8-Copying and Clearing

The copyArea method copies a rectangular area of the screen to another area of the screen. copyArea takes six arguments: the x and y of the top corner of the rectangle to copy, the width and the height of that rectangle, and the distance in the x and y directions to which to copy it.

For example, this line copies a square area 100 pixels on a side 100 pixels directly to its right:

g.copyArea(0,0,100,100,100,0);

To clear a rectangular area, use the clearRect method. clearRect, which takes the same four arguments as the drawRect and fillRect methods, fills the given rectangle with the current background color of the applet .
To clear the entire applet, you can use the size( ) method, which returns a Dimension object representing the width and height of the applet. You can then get to the actual values for width and height by using the width and height instance variables:
g.clearRect(0,0,this.size( ).width,this.height( ));

9-Including an Applet on a Web Page

After you create a class or classes that contain your applet and compile them into class files as you would any other Java program, you have to create a Web page that will hold that applet by using the HTML language. There is a special HTML tag for including applets in Web pages;
Java-capable browsers use the information contained in that tag to locate the compiled class files and execute the applet itself..

The <*APPLET*> Tag

To include an applet on a Web page, use the <APPLET> tag. <APPLET> is a special extension to HTML for including applets in Web pages.

Example (9): Assume you have the following applet program how you can embed it in html file:

Solution:

```
  import java.applet.Applet;
 import java.awt.Graphics;
public class  HelloWorldApplet  extends Applet {
  Public void paint (Graphics g) {
   g.drawString("Hello world!", 50, 25)  } }
```

solution:the following is a very simple example of a Web page with an applet included in it.

A simple HTML page.

```
 <HTML>
 <HEAD>
 <TITLE>This page has an applet on it</TITLE>
 </HEAD>
 <BODY>
 <P>My second Java applet says:
<APPLET CODE=”HelloAgainApplet.class” WIDTH=200 HEIGHT=50>
  </APPLET>
 </BODY>
 </HTML>
```

Exercise: Consider the program in fig (9) modify it to be as in fig (10) below:

Solution:

```
import java.awt.*;
 import java.applet.*;
import java.awt.Graphics;
public class Face extends Applet {
public void paint (Graphics g) {
   g.setColor(Color.yellow);
   g. fillOval (40, 40, 120, 150); //  Head.
   g.setColor(Color.black);
   g.fillOval (68, 81, 10, 10);          //  Left pupil.
   g.fillOval (121, 81, 10, 10); //  Right pupil.
   g.drawArc(60, 125, 80, 40, 180, 180);  //  Mouth.
   g.drawOval (25, 92, 15, 30);          //  Left ear. } }
```

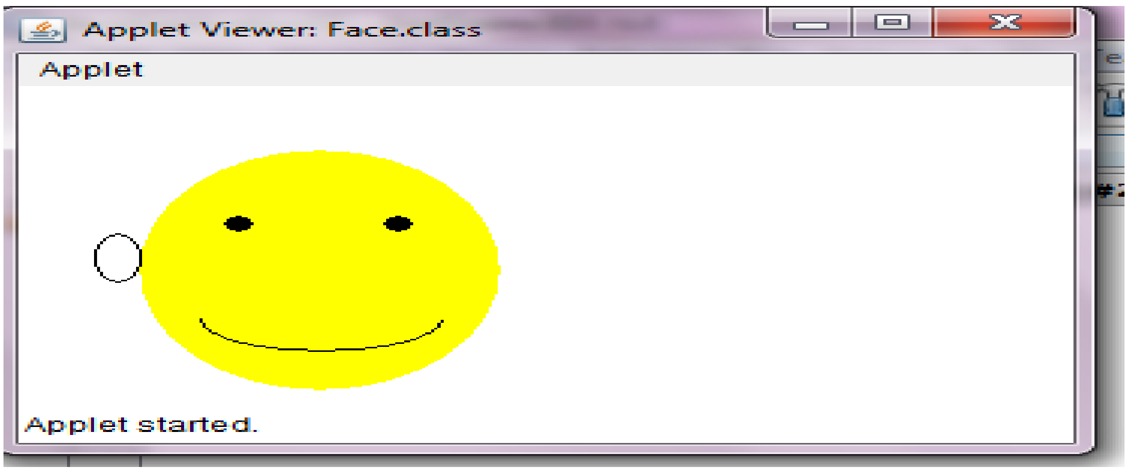


Fig (9) draws a face before

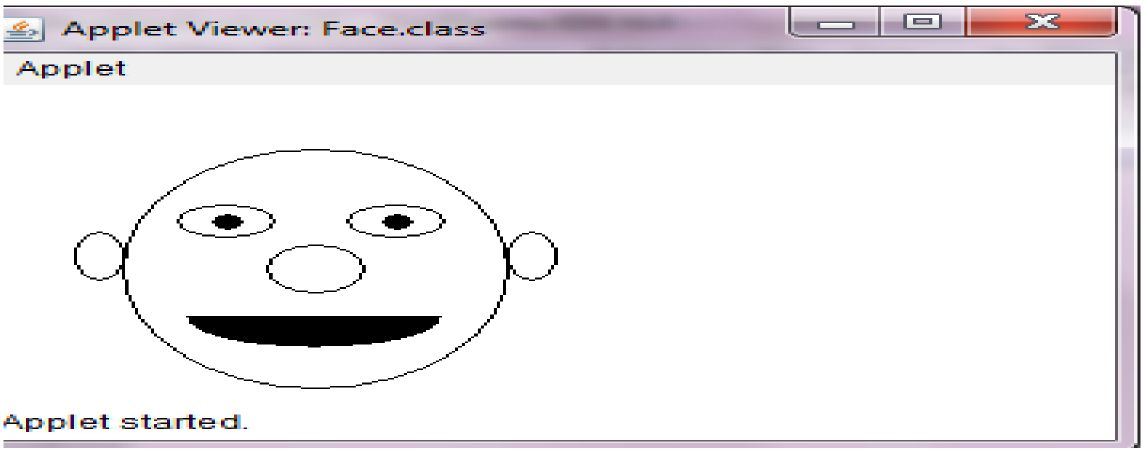


Fig (10) draws a face after

References:

1-"java How to program” , H.M.Deitel „ Sixth edition ,2006

2- https://www.geeksforgeeks.org/java-applet-basics/

3- https://www.tutorialspoint.com/java/java_applet_basics.htm