**How to draw polygon in java**

A **polygon** is a closed figure with many lines joined one to another. The ending point of one line is the starting point to another line and finally the last point is joins with the first point.

**java.awt.Graphics** class comes with two methods and one constructor to draw polygons.

1. **void drawPolygon(int x[], int y[], int numOfPoints):** Draws an **outline polygon** as per the coordinates specified in the **x[]** and **y[]** arrays. The **numOfPoints** gives the number of points (or to say, number of elements in the array) to join.
2. **void fillPolygon(int x[], int y[], int numOfPoints):** Draws a **solid polygon** as per the coordinates specified in the **x[]** and **y[]** arrays. The numOfPoints gives the number of points (or to say, number of elements in the array) to join.
3. **Polygon(int x [], int y [], int numOfPoints):** This constructor draws an **outline polygon** as per the coordinates specified in the **x[]** and **y[]** arrays. The **numOfPoints** gives the number of points to join.

All the above methods are used to draw polygons.

4 styles of drawing polygon are given.

1. Using drawPolygon() with arrays.
2. Using java.awt.Polygon class.
3. Using drawLine() method.
4. Using addPoint() method.

**1st style: Using drawPolygon() with arrays**

Here, **two arrays** comprising of x-coordinates and y-coordinates are created. These two arrays are passed to drawPolygon() method of **Graphics** class.

**Example prog:**

<applet code="DrawingPolygons.class" width="350" height="300">

       </applet>

import java.awt.\*

import java.applet.\*;

public class DrawingPolygons extends Applet

{

  public void paint(Graphics g)

  {

    int x[] = { 70, 150, 190, 80, 100 };

    int y[] = { 80, 110, 160, 190, 100 };

    g.drawPolygon (x, y, 5);

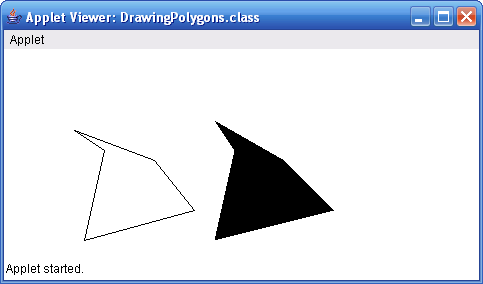
    int x1[] = { 210, 280, 330, 210, 230 };

    int y1[] = { 70, 110, 160, 190, 100 };

    g.fillPolygon (x1, y1, 5);

  }

}



Two arrays of **x** and **y** are created each with 5 elements. The array objects **x** and **y** are passed to **drawPolygon()** method with 5 points (marked by elements) to join. Alternatively, you can join 3 or 4 points also that joins first 3 or 4 becoming a **triangle** and **quadrilateral**. Attempting to join 6 points, which do not exist, throws ArrayIndexOutOfBoundException.

**2nd style: Using java.awt.Polygon class**

The above polygon can also be obtained with Polygon constructor as follows.

int x[] = { 50, 120, 150, 60, 75 };

       int y[] = { 60, 100, 150, 180, 100 };

       Polygon p1 = new Polygon(x, y, 5);

       g.drawPolygon(p1);

**3rd style: Using drawLine() method.**

Each and every line of the polygon is drawn separately (with drawLine()) while attaching end-points of one line to the starting points of another. If this is not done properly, you get a open polygon figure. This is a tedious approach compared to other 3 styles.

**Example prog:**

import java.awt.\*;

public class Style3Polygon extends Frame

{

  public Style3Polygon()

  {

    setTitle("Polygons by S N Rao");

    setSize(250, 350);

    setVisible(true);

  }

  public void paint(Graphics g)

  {

    g.drawLine(30, 50, 170, 110);

    g.drawLine(170, 110, 190, 220);

    g.drawLine(190, 220, 150, 320);

    g.drawLine(150, 320, 30, 50);

  }

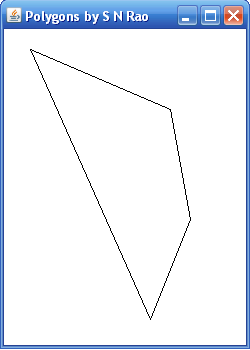
  public static void main(String args[])

  {

    new Style3Polygon();

  }

}



Observe the line coordinates carefully. One line **x1** and **y1** coordinates are another line's **x2**, **y2**. Again starting line **x1**, **y1** are last line's **x2**, **y2**; else, an open figure will be obtained.

**4th style: Using addPoint() method.**

Here, we use **addPoint()** method of **Polygon** class. addPoint() method takes a pair of coordinates that becomes automatically the vertex of the polygon. **drawPolygon()** method takes care of joining the first vertex with the last vertex.

Supporting method of Polygon class

* **void addPoint(int x, int y):** **drawPolygon()** mehod joints all the points of x and y specified in each **addPoint()** method.

**Prog example:**

public class Style4Polygon extends Frame

{

  public Style4Polygon()

  {

    setTitle("Polygons by S N Rao");

    setSize(400, 300);

    setVisible(true);

  }

  public void paint(Graphics g)

  {

    Polygon p1 = new Polygon();

    p1.addPoint(40,50);

    p1.addPoint(160,150);

    p1.addPoint(220,140);

    p1.addPoint(175,270);

    p1.addPoint(80,90);

    g.drawPolygon(p1);

  }

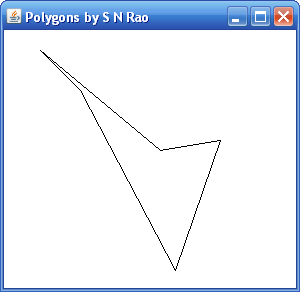
  public static void main(String args[])

  {

    new Style4Polygon();

  }

}



**HOW TO DRAW RECTANGLE BY USING APPLET CLASS**

Java comes with two methods to draw right-angled rectangles.

Supporting methods in Graphics class

* **void drawRect(int x, int y, int width, int height):** Draws an outline rectangle with the left-top coordinates of **x** and **y** and with the **width** and **height** specified.
* **void fillRect(int x, int y, int width, int height):** Draws a solid rectangle with the left-top coordinates of **x** and **y** and with the **width** and **height** specified.

Following program draws two right-angled rectangles – **outline** and **solid** in applet window.

**Applet file name: RectanglesDrawing.java**

**HTML FILE :** <applet code="RectanglesDrawing" width="400" height="300">

</applet>

**Java code:**

import java.awt.\*;

import java.applet.\*;

public class RectanglesDrawing extends Applet

{

  public void paint(Graphics g)

  {

    g.setColor(Color.blue);

    g.drawRect(50, 80, 150, 100);

    g.setColor(Color.magenta);

    g.fillRect(230, 80, 150, 100);

  }

}

the **setColor()** method of Graphics class sets the drawing color. In the above statement, the color is set to **blue** (one of the 13 predefined colors) and this color will be effective until changed again.

**drawRect()** method takes 4 parameters. 50 and 80 are the **x** and **y** positions that represents the top-left coordinates of the rectangle. 150 and 100 are the **width** and **height** of the rectangle. The method draws an outline rectangle in blue color.

**fillRect()** method draws a solid rectangle filled with **magenta** (one of the 13 predefined colors) color. 230, 80 are **x**, **y** coordinates and 150, 100 are **width** and **height** of the rectangle

**Drawing Square**

g.drawRect(50, 80, 150, 150);

The above statement draws a square. Observe, the width and height are same of 150 pixels.

**Java Draw Circles Graphics Applets**

No special or predefined method exists with Graphics class to draw circles. But still, circles can be drawn in three styles using other methods.

Using drawRoundRect()

Using drawOval() – most preferred

Using drawArc()

**Prog example:**

**Java code:**

import java.awt.Graphics;

import java.applet.Applet;

public class ThreeStyles extends Applet

{

  public void paint (Graphics g)

  {                  // using drawRoundRect()

    g.drawRoundRect(40, 50, 90, 90, 200, 200);

    g.fillRoundRect(40, 160, 90, 90, 200, 200);

                 // using drawOval()

    g.drawOval(150, 50, 90, 90);

    g.fillOval(150, 160, 90, 90);

                // using drawArc()

    g.drawArc(270, 50, 90, 90, 0, 360);

    g.fillArc(270, 160, 90, 90, 0, 360);

  }

}

**Html:**

<applet code="ThreeStyles" width="500" height="400">

</applet>

