POLYGONS

The Java API class `java.awt.Graphics` declares two methods that draw polygons.

<table>
<thead>
<tr>
<th>java.awt.Graphics Methods to Draw Polygons</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void drawPolygon( int [] x, int [] y, int n )</code></td>
</tr>
<tr>
<td><code>void fillPolygon( int [] x, int [] y, int n )</code></td>
</tr>
</tbody>
</table>

The first two arguments are parallel arrays holding the (x, y) coordinates of the polygon’s corners. The third argument is the number of corners. The methods trace out a polygon by connecting the points specified by arrays x and y – first point to second, second to third, and so on finishing by connecting the last point with the first.

**Example**

This code draws a polygon from (150, 50) to (225, 125) to (150, 200) to (75, 125) and back to (150, 50).

```java
int [] x = { 150, 225, 150, 75 };  
int [] y = { 50, 125, 200, 125 };  
g.drawPolygon( x, y, 4 );
```

drawPolygon traces the polygon’s outline; fillPolygon fills it with color. Both methods draw onto the applet’s graphics context in whatever color it is currently set to.

**Example**

```java
int [] x = { 150, 225, 150, 75 };  
int [] y = { 50, 125, 200, 125 };  
g.fillPolygon( x, y, 4 );
```
Example

```java
int [] x = { 150, 150, 225,  75 };
int [] y = {  50, 200, 125, 125 };
g.drawPolygon( x, y, 4 );
```

Example

```java
int [] x = { 150, 150, 225,  75 };
int [] y = {  50, 200, 125, 125 };
g.fillPolygon( x, y, 4 );
```

The `drawPolyline` method has the same arguments as `drawPolygon`. It draws lines from one corner to the next but does not connect the last corner with the first.

```java
void drawPolyline( int [] x, int [] y, int n )
```

Example

```java
int [] x = { 150, 225, 150,  75 };
int [] y = {  50, 125, 200, 125 };
g.drawPolyline( x, y, 4 );
```
**Example**

Write an applet that draws a pentagon and a 5-pointed star as shown. The pentagon has a black border and a light gray interior. The star is red. The corners of the two figures are, starting at the top and moving clockwise around the figure, (260, 10), (498, 183), (407, 462), (113, 462), and (22, 183).

To draw and fill the pentagon, we can call the polygon methods sending the corner coordinates in the order listed.

```java
int [] pentagonX = { 260, 498, 407, 113, 22 };
int [] pentagonY = { 10, 183, 462, 462, 183 };
```

You must fill the polygon before drawing the border; otherwise the fill will cover the border.

```java
g.fillPolygon( pentagonX, pentagonY, 5 );
g.drawPolygon( pentagonX, pentagonY, 5 );
```

One way to draw the star is to connect the top point (260, 10) to that on the lower right (407, 462) then to the upper left (22, 183) then across to the upper right (498, 183) then down to the lower left (113, 462) and finally back to the start (260, 10). Here’s the array setup for that:

```java
int [] starX = { 260, 407, 22, 498, 113, 260 };
int [] starY = { 10, 462, 183, 183, 462, 10 };
```

And the call to `drawPolyline`:

```java
g.drawPolyline( starX, starY, 6 );
```

The complete applet is shown below.
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Color;

public class StarApplet extends Applet
{
    int [] pentagonX = { 260, 498, 407, 113, 22 };
    int [] pentagonY = { 10, 183, 462, 462, 183 };

    int [] starX = { 260, 407, 22, 498, 113, 260 };
    int [] starY = { 10, 462, 183, 183, 462, 10 };

    public void paint( Graphics g )
    {
        // always call the Applet paint method first
        super.paint( g );
        // draw light grey pentagon
        g.setColor( Color.LIGHT_GRAY );
        g.fillPolygon( pentagonX, pentagonY, 5 );
        // give it a black border
        g.setColor( Color.BLACK );
        g.drawPolygon( pentagonX, pentagonY, 5 );
        // draw a red star
        g.setColor( Color.RED );
        g.drawPolyline( starX, starY, 6 );
    }
}
## Exercises

1. Write a Java applet that uses the `drawPolyline` method to draw a 3-dimensional appearing pyramid as shown.

![Pyramid](image1.png)

2. Write a Java applet that fills the background with a blue triangle that has a red border as shown. Use the `fillPolygon` and `drawPolygon` methods to draw it. The triangle must resize so that it always fills the background whenever the applet window is resized.

![Triangular Pattern](image2.png)

3. Write a Java applet that fills the background with a yellow diamond that has a red border as shown. Use the `fillPolygon` and `drawPolygon` methods to draw it. The diamond must resize so that it always fills the background whenever the applet window is resized.

![Diamond Pattern](image3.png)
4. Modify your solution to problem 3 so that it follows these specifications:
   - Use a `JOptionPane` input dialog to read the user’s name, as shown below:

   ![JOptionPane Input Dialog]

   - Write a welcome message to the user on the applet, as shown here:

   ![Welcome Message]

   - A horizontal line must be drawn through the diamond, centered vertically.
   - The message must be printed just above the line.
   - The message must be horizontally centered within the applet.