```java
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Color;
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.util.Random;
public class Stars extends JPanel {
	// Unique version ID for this class to ensure saved objects can be loaded safely
	private static final long serialVersionUID = 1L;
	// Initial width of height of the window
	private static int width = 1000;
	private static int height = 650;
	// main method to launch the program as a standalone application - no need to modify
	public static void main(String[] args) {
		Stars panel = new Stars();
		panel.setPreferredSize(new Dimension(width, height)); // content size window dimensions
		JFrame frame = new JFrame("Stars"); // Title of frame

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.add(panel);
		frame.pack();
		frame.setVisible(true);

	}
	/**
```

```
        * Draws 10 stars (or more) of random sizes in random
locations.  The stars do not have to be fully contained within the
panel.
        * @param g the Graphics object used for drawing shapes,
text, and images
        */
       public void drawStars(Graphics g) {
              Random randy = new Random();
              double num = 100; //number of stars
              for (int z = 0; z < num; z++) {
                     double x = randy.nextDouble(1000); //The starting
x-coordinate
                     double y = randy.nextInt(650); //The starting
y-coordinate
                     double r2 = randy.nextDouble(60); //Outer radius;
distance from the center of the star to the tips.
                     double r1 = (double) r2 * 0.38; //Inner radius;
distance from the center to the inner indentations of the star.
                     int n = 5; //Number of outer points in the star.
                     int vertices = 2*n; //Number of total points in the
star
                     double theta = (Math.PI * 2)/vertices; //
                     double theta0 = -(Math.PI / 2);
                     int [] xc = new int [vertices];
                     int [] yc = new int [ vertices];

                     Color color = new Color (
                                   randy.nextInt(256),
                                   randy.nextInt(256),
                                   randy.nextInt(256)
```

```
                                );
                        g.setColor(color);
                        //This for loop runs that code below until the
number of times the program had looped is equal to the number
of vertices.
                        for(int i = 0; i < vertices; i++) {
                        //The if-else statement uses a concept called
Modular Arithmetic to determine whether the number of turns that
the program looped is an even number or an odd number. If it's
an even number, then an outer point will be made. If it's an odd
number, then an inner point will be made.
                        if(i % 2 == 0) {
                        xc[i] = (int)(x + (r2 * Math.cos(theta0 + i * theta)));
                        yc[i] = (int)(y + (r2 * Math.sin(theta0 + i * theta)));
                        }
                        else {
                                xc [i] = (int)(x + (r1 * Math.cos(theta0 + i *
theta)));
                                yc [i] = (int)(y + (r1 * Math.sin(theta0 + i *
theta)));
                        }
                        }
                        g.fillPolygon(xc, yc, vertices);
                }



        }
        /**
```

```
        * Overrides JPanel's paintComponent method to perform
custom drawing.
        * @param g the Graphics object used for drawing shapes,
text, and images
        */
        @Override
        protected void paintComponent(Graphics g) {
                super.paintComponent(g); // Clears the panel before
drawing
                drawStars(g);
        }
}
```