Stars.java

```java
package exercises;
import java.awt.*;
import java.applet.*;
import java.awt.Color;
import java.util.Random;

public class Stars extends Applet {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public void paint(Graphics g) {
        Random rand = new Random();
        g.fillRect(0, 0, 1250, 650);
        Color[] colors = new Color[10];
        for (int i = 0; i < 10; i++)
            colors[i] = new Color(rand.nextInt(256),rand.nextInt(256),rand.nextInt(256));
        int[] hVals = new int[10];
        int[] kVals = new int[10];
        int[] rVals = new int[10];
        boolean overlapping = true;
        int r = 0;
        int h = 0;
        int k = 0;
        for (int i = 0; i < 10; i++) {
            g.setColor(colors[i]);
            overlapping = true;
            r = rand.nextInt(121) + 10;
            while (overlapping) {
                overlapping = false;
                h = rand.nextInt(1250 - 2 * r) + r;
                k = rand.nextInt(650 - 2 * r) + r;
                for (int j = 0; j < 10; j++) {
                    for (int x = 0; x < 1250; x++) {
                        for (int y = 0; y < 650; y++) {
                            if (Math.pow(x - h, 2) + Math.pow(y - k, 2) <= Math.pow(r, 2) &&
Math.pow(x - hVals[j], 2) + Math.pow(y - kVals[j], 2) <= Math.pow(rVals[j], 2))
                                overlapping = true;
                        }
                    }
                }
            }
            hVals[i] = h;
            kVals[i] = k;
            rVals[i] = r;
            double x1 = h;
            double y1 = k - r;
            double c = Math.sqrt(2 * Math.pow(r, 2) - 2 * Math.pow(r, 2) *
Math.cos(Math.toRadians(72)));
            double x2 = h - c * Math.cos(Math.toRadians(36));
            double y2 = k + c * Math.sin(Math.toRadians(36)) - r;
            double xm = h - r * Math.cos(Math.toRadians(36)) * Math.sin(Math.toRadians(72));
            double ym = k + r * Math.cos(Math.toRadians(36)) * Math.cos(Math.toRadians(72));
            double x3 = 2 * xm - x2;
            double y3 = 2 * ym - y2;
            double x4 = 2 * h - x3;
            double y4 = y3;
            double x5 = h + c * Math.cos(Math.toRadians(36));
            double y5 = y2;
            //Naming convention for inside points: the point between (x1,y1) and (x2,y2) will be called pt12 as it is the point between 1 and 2
            int[] pt12 = systemSolver(x1,y1,x3,y3,x2,y2,x5,y5);
            int[] pt23 = systemSolver(x1,y1,x3,y3,x2,y2,x4,y4);
            int[] pt34 = systemSolver(x2,y2,x4,y4,x3,y3,x5,y5);
            int[] pt45 = systemSolver(x3,y3,x5,y5,x4,y4,x1,y1);
            int[] pt51 = systemSolver(x4,y4,x1,y1,x2,y2,x5,y5);
            int[] xVals = {(int) x1, pt12[0], (int) Math.round(x2), pt23[0], (int) Math.round(x3), pt34[0], (int) Math.round(x4), pt45[0], (int) Math.round(x5), pt51[0]};
            int[] yVals = {(int) y1, pt12[1], (int) Math.round(y2), pt23[1], (int) Math.round(y3), pt34[1], (int) Math.round(y4), pt45[1], (int) Math.round(y5), pt51[1]};
            g.fillPolygon(xVals, yVals, 10);
        }
    }

    //parameters are the two points on one line as ordered pairs, then the two points on the other line in the same format
    public static int[] systemSolver(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {
        double x = (y3 - ((y4 - y3) / (x4 - x3)) * x3 - y1 + ((y2 - y1) / (x2 - x1)) * x1) / ((y2 - y1) / (x2 - x1) - (y4 - y3) / (x4 - x3));
        double y = ((y2 - y1) / (x2 - x1)) * x + y1 - ((y2 - y1) / (x2 - x1)) * x1;
        int[] coords = {(int) Math.round(x), (int) Math.round(y)};
        return coords;
    }
}
```