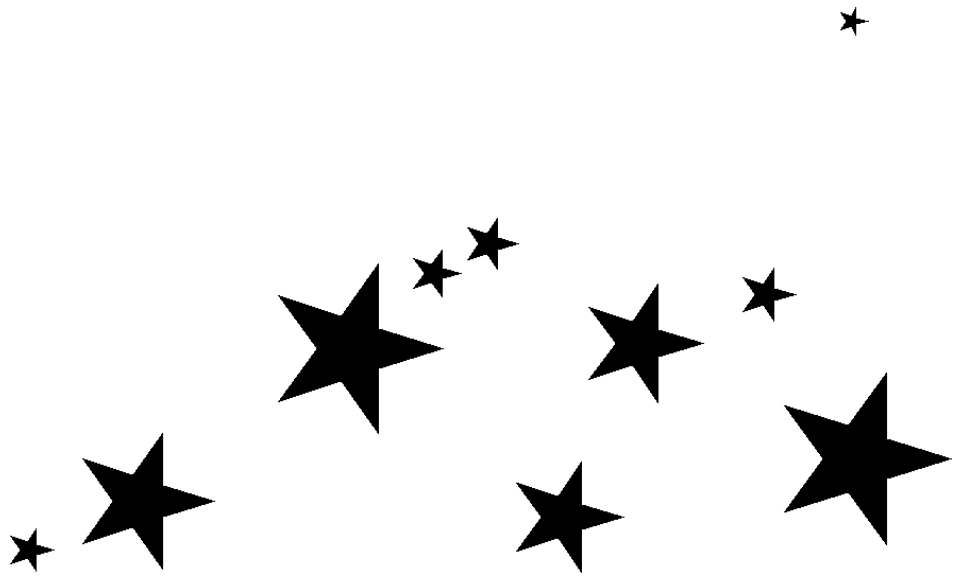

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

public class randomstars extends Applet {
//random location
//random size
      public void paint(Graphics g) {
            Random randomizer= new Random();
            int array_length = 10;
            for (int i = 0; i< array_length; i++) {
                  int initial_x = (randomizer.nextInt(1000) + 1);
                  int initial_y = (randomizer.nextInt(650) + 1);
                  //2.618 is ratio between the radii, as illustrated by this
resource:
https://math.stackexchange.com/questions/2135982/math-behind-creating-a-perfect
-star
                  double radii_ratio = 2.618;
                  int outer_pent_radius =
(int)(Math.round((randomizer.nextInt(100) + 1)));
                  int inner_pent_radius =
(int)(Math.round(outer_pent_radius/radii_ratio));
                  int rotation_factor = 36;
                  // This can be attained fairly easily:
                  //From the center of the perfect star and both outer/inner
pentagons, 5 kites, each split into two triangles,
                  //constitute a circle circumscribing the outer pentagon. This
creates 10 portions of the 360 space.
                  //In turn, this necessitates a rotation factor of the point
by 36 degrees (360/10)
```

```
                    int x_points[] = new int[array_length];
                    int y_points[] = new int[array_length];

                    // for the for loop constructing the start itself, there has
to be an interval of two,
                    // in order to get the two points, one for the
outer_pent_rad, other for inner.
                    //rotation factor is multiplied by the sub_i index, which is
at most 9, meaning 0 to (30 * 9) is performed, connecting back to the original
points, completing one rotation, and creating the star.
                    for (int sub_i = 0; sub_i < array_length; sub_i += 2) {
                          x_points[sub_i] = initial_x +
(int)(Math.cos(Math.toRadians(sub_i*rotation_factor))*outer_pent_radius);
                          x_points[sub_i+1] = initial_x +
(int)(Math.cos(Math.toRadians((sub_i+1)*rotation_factor))*inner_pent_radius);
                          y_points[sub_i] = initial_y +
(int)(Math.sin(Math.toRadians(sub_i*rotation_factor))*outer_pent_radius);
                          y_points[sub_i+1] = initial_y +
(int)(Math.sin(Math.toRadians((sub_i+1)*rotation_factor))*inner_pent_radius);
                    }
                    g.fillPolygon(x_points, y_points, array_length);
              }

      }

}
```