Programming Exercise

CSC140 Foundations of Computer Science

09 February 2015

Exhalation.java

```
package exhalation;

import java.awt.Component;
import java.awt.Container;
import javax.swing.JFrame;

public class Exhalation extends JFrame {

  private static final int WINDOW_WIDTH = 512;
  private static final int WINDOW_HEIGHT = 512;
  private static final String WINDOW_TITLE = "Exhalation";

  public Exhalation() {
    this.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    this.setTitle(WINDOW_TITLE);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Container pane = this.getContentPane();
    Component panel = new ExhalationPanel();
    pane.add( panel );

    this.setVisible(true);
  } // Exhalation()

  public static void main(String[] args) {
    System.out.println("Guten Tag!");
    Exhalation exhalatation = new Exhalation();
  } // main( String [] )

} // Exhalation
```

ExhalationPanel.java

```
package exhalation;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JPanel;

public class ExhalationPanel extends JPanel {

  private static final Color BG_COLOR = new Color(112, 132, 206);
  private static final Color FG_COLOR = new Color(206, 224, 160);

  public ExhalationPanel() {
    this.setBackground(BG_COLOR);
  } // ExhalationPanel()

  @Override
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2D = (Graphics2D) g;

    int w = this.getWidth();
    int h = this.getHeight();

    AffineTransform scale = new AffineTransform();
    scale.setToScale(w / 2, h / 2);

    AffineTransform translate = new AffineTransform();
    translate.setToTranslation(1.0, 1.0);

    AffineTransform transform = new AffineTransform();
    transform.concatenate(scale);
    transform.concatenate(translate);

    BasicStroke stroke = new BasicStroke(4);
    g2D.setStroke(stroke);
```

```
    g2D.setColor(Color.WHITE);

    Rectangle2D bounds = new Rectangle2D.Double(-1.0, -1.0, 2.0, 2.0);
    Shape transformedBounds = transform.createTransformedShape(bounds);


    List<Circle> bubbles = new ArrayList<>();

    double meanRadius = 0.1;

    int n = 8192;
    for (int i = 0; i < n; i++) {
      double x = 2 * Math.random() - 1.0;
      double y = 2 * Math.random() - 1.0;
      double r = -meanRadius * Math.log(Math.random());
      if( r > 1.0 ) {
        r = 1.0;
      } // if

      Circle circle = new Circle(x, y, r);
      Shape shape = circle.getShape(transform);
      if (transformedBounds.contains(shape.getBounds2D())) {
        boolean overlapping = false;
        int j = 0;
        while (!overlapping && j < bubbles.size()) {
          Circle otherCircle = bubbles.get(j);
          overlapping = circle.intersects(otherCircle);
          j++;
        } // while
        if( !overlapping) {
          bubbles.add( circle );

          g2D.setColor( FG_COLOR );
          g2D.fill(shape);

          g2D.setColor( Color.WHITE );
          g2D.draw(shape);
        } // if
      } // if

    } // for

  } // paintComponent( Graphics )

} // ExhalationPanel
```

Circle.java

```
package exhalation;

import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;


public class Circle {
  private final double x;
  private final double y;
  private final double r;

  public Circle( double x, double y, double r ) {
    this.x = x;
    this.y = y;
    this.r = r;
  } // Circle( double, double, double )

  public Shape getShape( AffineTransform t) {
    double ulx = this.x - this.r;
    double uly = this.y - this.r;
    double dia = 2 * this.r;
    Ellipse2D circle = new Ellipse2D.Double( ulx, uly, dia, dia );
    return t.createTransformedShape(circle);
  } // getShape( AffineTransform )

  public boolean intersects( Circle c ) {
    double sumOfRadii = this.r + c.r;
    double xDiff = this.x - c.x;
    double yDiff = this.y - c.y;
    double distance = Math.sqrt( xDiff * xDiff + yDiff * yDiff );
    return distance < sumOfRadii;
  } // intersects( Circle )

} // Circle
```