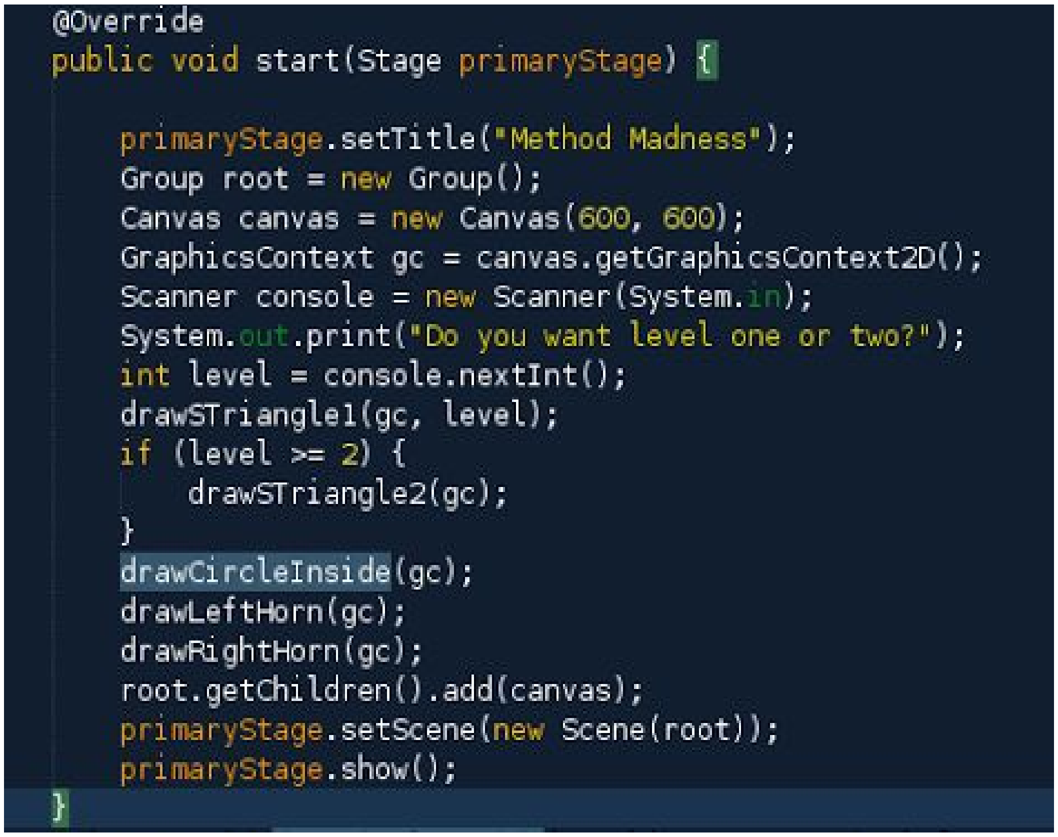
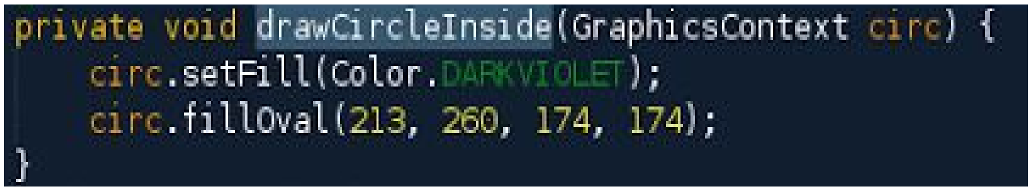
Method Madness Summative Essay

My Method Madness program, “Shen_7_MethodMadness.java”, first draws a one or two level fractal triangle, depending on the what the user enters. It then adds a purple circle in the center and horns on either side. It isn’t just a pattern, or just a collection of shapes like many other Method Madness creations, but combines both. It draws the fractal part through a simplified formula, and draws the rest with the code for one shape at a time.

My program demonstrates encapsulation because I call many private methods in a public method. A specific example is where the public method *drawCircleInside(gc);* is used to call the private method *private void drawCircleInside(GraphicsContext circ) {*.

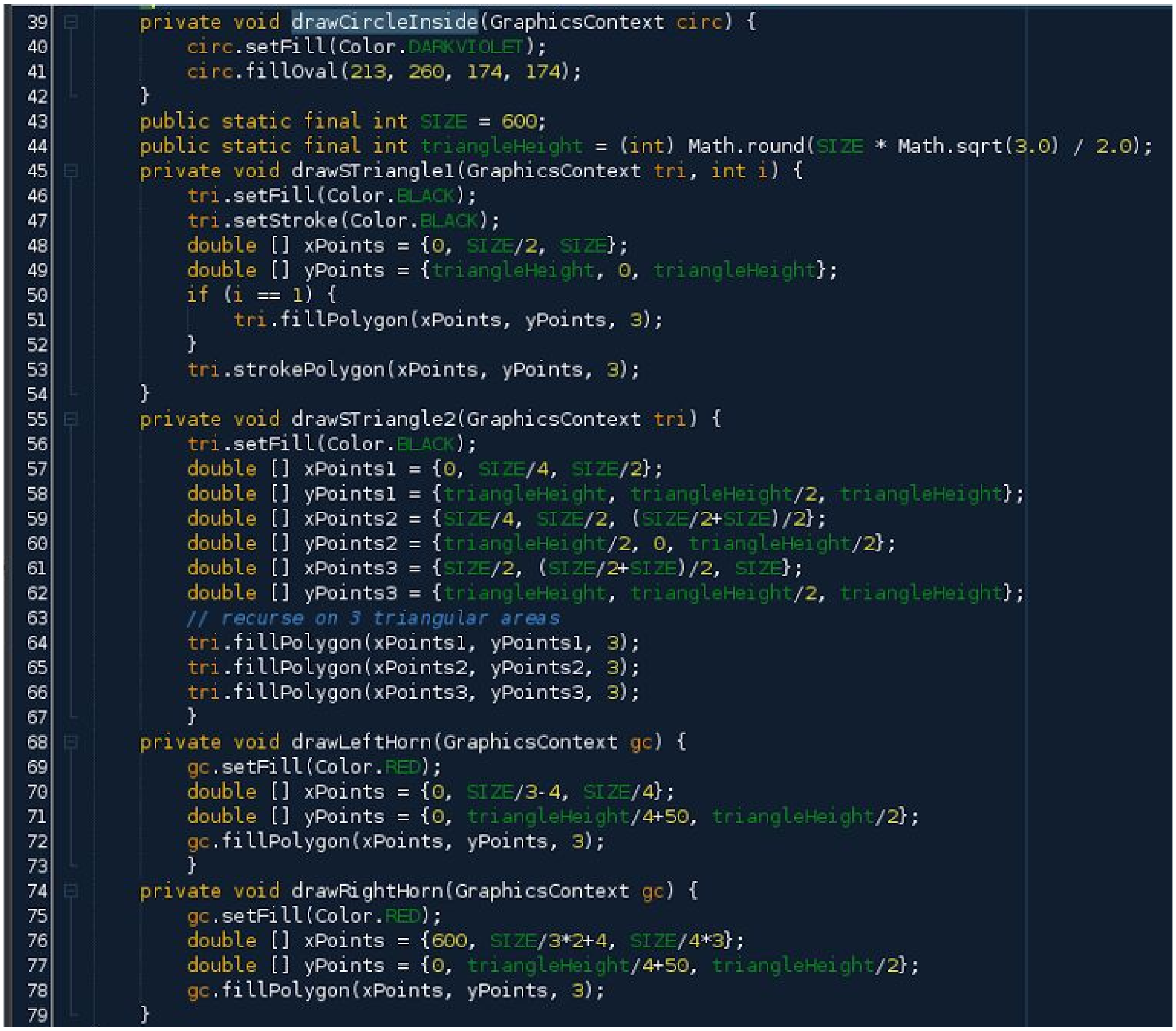


```
@Override
public void start(Stage primaryStage) {

    primaryStage.setTitle("Method Madness");
    Group root = new Group();
    Canvas canvas = new Canvas(600, 600);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    Scanner console = new Scanner(System.in);
    System.out.print("Do you want level one or two?");
    int level = console.nextInt();
    drawSTriangle1(gc, level);
    if (level >= 2) {
        drawSTriangle2(gc);
    }
    drawCircleInside(gc);
    drawLeftHorn(gc);
    drawRightHorn(gc);
    root.getChildren().add(canvas);
    primaryStage.setScene(new Scene(root));
    primaryStage.show();
}
```



```java
private void drawCircleInside(GraphicsContext circ) {
    circ.setFill(Color.DARKVIOLET);
    circ.fillOval(213, 260, 174, 174);
}
```

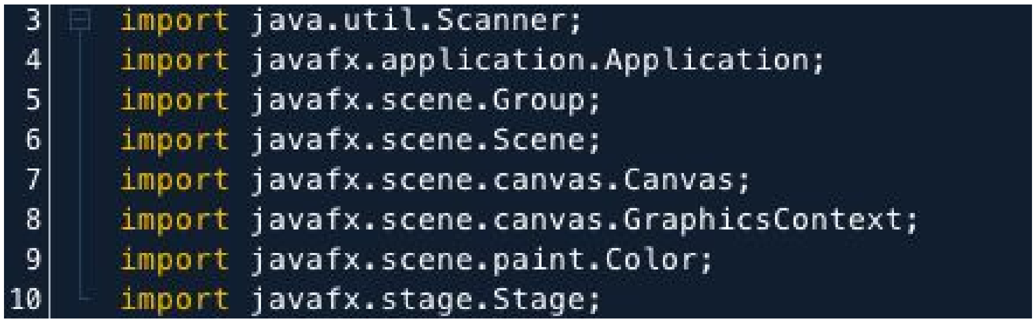
These are the methods I used:



```java
    private void drawCircleInside(GraphicsContext circ) {
        circ.setFill(Color.DARKVIOLET);
        circ.fillOval(213, 260, 174, 174);
    }
    public static final int SIZE = 600;
    public static final int triangleHeight = (int) Math.round(SIZE * Math.sqrt(3.0) / 2.0);
    private void drawSTriangle1(GraphicsContext tri, int i) {
        tri.setFill(Color.BLACK);
        tri.setStroke(Color.BLACK);
        double [] xPoints = {0, SIZE/2, SIZE};
        double [] yPoints = {triangleHeight, 0, triangleHeight};
        if (i == 1) {
            tri.fillPolygon(xPoints, yPoints, 3);
        }
        tri.strokePolygon(xPoints, yPoints, 3);
    }
    private void drawSTriangle2(GraphicsContext tri) {
        tri.setFill(Color.BLACK);
        double [] xPoints1 = {0, SIZE/4, SIZE/2};
        double [] yPoints1 = {triangleHeight, triangleHeight/2, triangleHeight};
        double [] xPoints2 = {SIZE/4, SIZE/2, (SIZE/2+SIZE)/2};
        double [] yPoints2 = {triangleHeight/2, 0, triangleHeight/2};
        double [] xPoints3 = {SIZE/2, (SIZE/2+SIZE)/2, SIZE};
        double [] yPoints3 = {triangleHeight, triangleHeight/2, triangleHeight};
        // recurse on 3 triangular areas
        tri.fillPolygon(xPoints1, yPoints1, 3);
        tri.fillPolygon(xPoints2, yPoints2, 3);
        tri.fillPolygon(xPoints3, yPoints3, 3);
        }
    private void drawLeftHorn(GraphicsContext gc) {
        gc.setFill(Color.RED);
        double [] xPoints = {0, SIZE/3-4, SIZE/4};
        double [] yPoints = {0, triangleHeight/4+50, triangleHeight/2};
        gc.fillPolygon(xPoints, yPoints, 3);
        }
    private void drawRightHorn(GraphicsContext gc) {
        gc.setFill(Color.RED);
        double [] xPoints = {600, SIZE/3*2+4, SIZE/4*3};
        double [] yPoints = {0, triangleHeight/4+50, triangleHeight/2};
        gc.fillPolygon(xPoints, yPoints, 3);
    }
```

The data I passed to my methods was mostly just the Graphics Context or gc, but I also passed a value concerning whether the user wanted the second level in the fractal. I then checked to see if they did. For the methods that drew the shapes, such as *circ.fillOval(213, 260, 174, 174);* determine the location and exact dimensions of the oval, or in this case, circle. This is similar to the *tri.fillPolygon(xPoints, yPoints, 3);* method, where the first two variables are lists of the coordinates of the points, and the three tells the computer that I want a shape with three points, a triangle. Other methods such as *gc.setFill(Color.RED);* have a color passed to them instead of a variable. This determines the color of all the drawings after it is defined. These methods return shapes drawn on a canvas that uses a Cartesian coordinate system. The origin (0,0) is in the top left corner and the plane expands downwards and right to the defined extent, which in my case was 600 by 600.

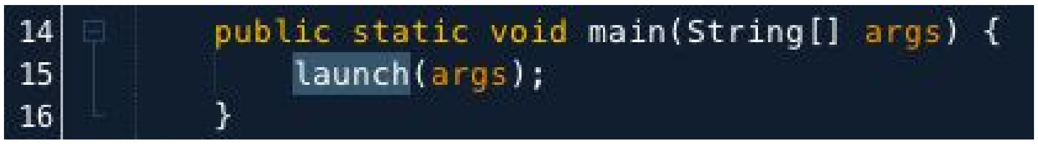
These methods are all imported at the beginning of the program, such as



```
import java.util.Scanner;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
```

Many of these are also classes that I used in the program, the primary two being gc, and canvas.

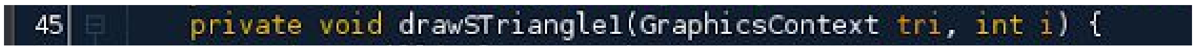
I have five different methods in my program, there is only one called in There is only one method called in main(): *launch(args);*. This is due to the fact that my program creates drawings on a canvas, and does not print out values.



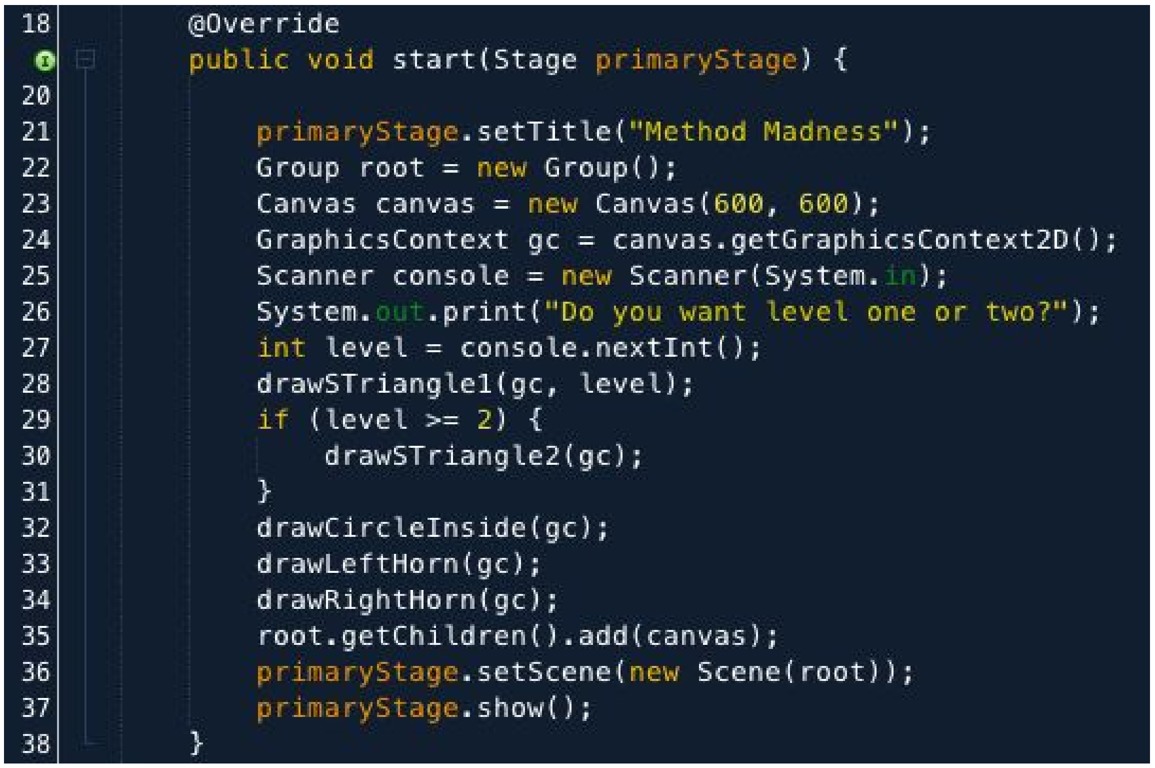
```
    public static void main(String[] args) {
        launch(args);
    }
```

I handled errors by making sure I didn't divide by zero

I used both public and private access modifiers in this program. An example of this is when the private method *drawSTriangle1(gc, level);* is called inside the public method *public void start(Stage primaryStage) {*.



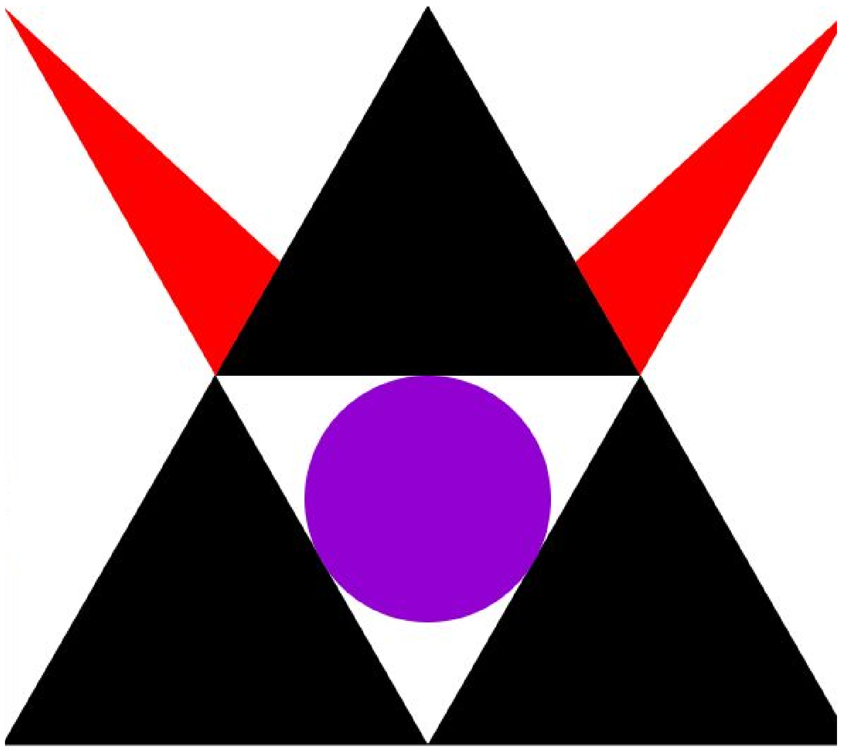
```java
    private void drawSTriangle1(GraphicsContext tri, int i) {
```



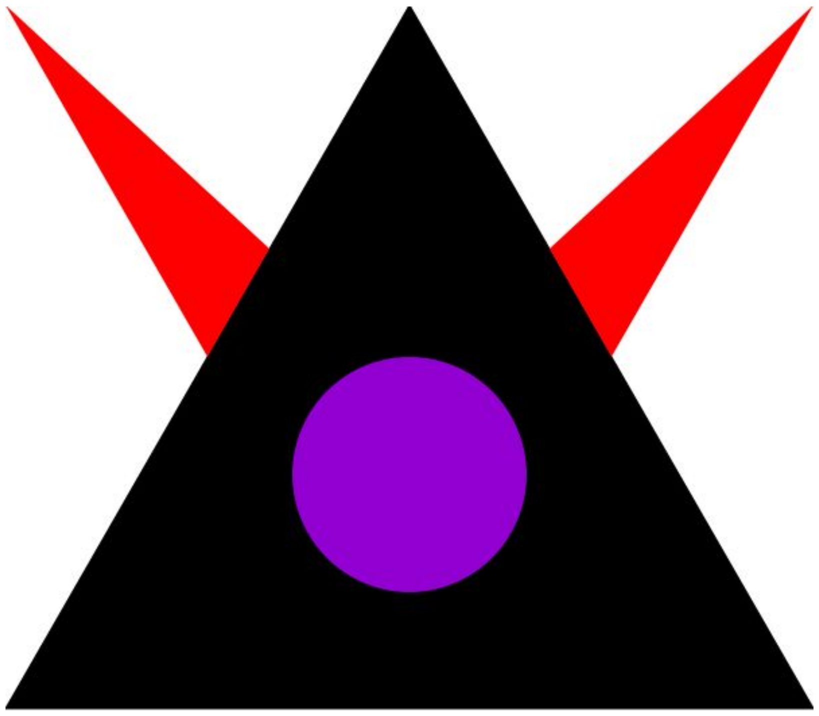
```java
    @Override
    public void start(Stage primaryStage) {

        primaryStage.setTitle("Method Madness");
        Group root = new Group();
        Canvas canvas = new Canvas(600, 600);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        Scanner console = new Scanner(System.in);
        System.out.print("Do you want level one or two?");
        int level = console.nextInt();
        drawSTriangle1(gc, level);
        if (level >= 2) {
            drawSTriangle2(gc);
        }
        drawCircleInside(gc);
        drawLeftHorn(gc);
        drawRightHorn(gc);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
```

I also used class constructors such as *drawLeftHorn* and *drawRightHorn*. All of my constructors simply let me pass variables into my methods and do not return a value, but instead draw a shape on the canvas.

Altogether, my program creates this image if you want two levels:



And it creates this if you only want one:



My project embarked on a shaky road from the very beginning. I was unable to work on it at first, and consulted various websites on how to create a fractal. I eventually found some old Java code that I converted into what you see now. I gradually understood how each part of the Graphic Context worked, and now I have my great program. I had had to gradually trim off ideas because they were too complicated and lengthy for the time and knowledge I had, but now, after this project, I feel experienced enough to keep on coding.