Note: For all draw methods tested, the DrawFractal class was used by creating a method in DrawFractal that calls the draw method of the input shape/fractal/fractal picture.

Testing Point Class

To test the Point class, the getter and setter methods must be tested, the rotate method must be tested, and the exception must be tested. To test the getter and setter methods, the point (0,0) should be input and the getX() and getY() should be used to return the respective values. Then using the setter methods, the point should be changed to non-zero values and then using the getter methods should return the new values just set. To test the exception, an instance of Point must be created where either the x or the y value of the point is negative.

Testing BaseShapes Class and Line Class

The BaseShapes Class can be tested through the Line class because Line is a subclass of BaseShapes. Thus, it inherits all of the methods from BaseShapes.

To test the Line class, the two constructors, the getFirstPoint() method, the getSecondPoint() method, the setFirstPoint() method, the setSecondPoint() method, the getLines() method, the getNumSides() method, and the draw method should be tested. To test the getter and setter methods, an instance of Line should be created, where the getter methods are first used to check the two end points, then the setter methods are used to change them, and lastly the two ends should be checked using the getter methods. The getNumSides() can be tested as soon as the instance of Line is created. To test the draw method, requires using DrawFractal class to create the canvas where the line will be drawn using a helper method in DrawFractal. The draw method is tested by checking whether the output image is correct. The exceptions for the setter methods and constructors can be tested by creating cases where the end points of the line are the same.

Special cases can be made to test Line, like when the endpoints involve negative values or when the endpoints involve a zero in one of the coordinates.

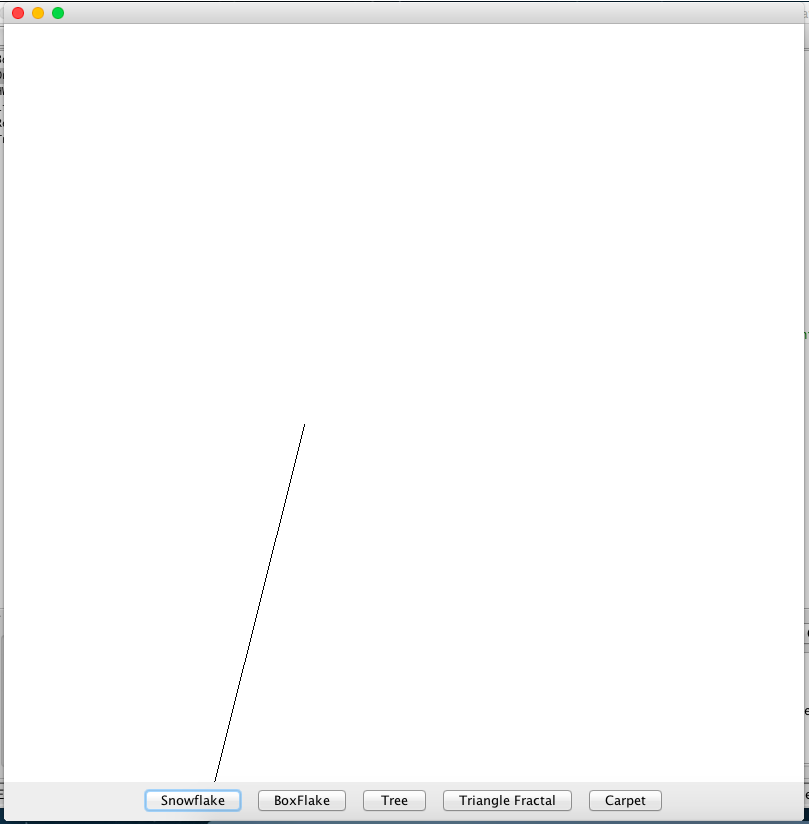
Testing draw(Graphics graphics):

> DrawFractal drawTest = new DrawFractal(800, 800)

> drawTest.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> drawTest.drawBaseShape(new Line(300, 400, 200, 800))

Produces:

This image is correct because the window was created as 800x800 and the coordinates of the two endpoints were (300, 400) and (200, 800). Thus we can assume that the draw method works for BaseShapes class since the draw method is inherited by Line class.

Testing Rectangle Class

To test the Rectangle Class, the two constructors, getReferencePoint(), getWidth(), getHeight(), getLines(), and the draw() method should be tested. To test the two constructors, they should be tested so that their exceptions are thrown, one where some points are zero and negative, and regular cases where the coordinates are positive. The getWidth() and getHeight() should be tested so that the outputs are correct for cases where the values are positive, where some coordinates are zero, and where come coordinates are negative. The getLines() method should be tested such that the lines are in the proper order in the array and there are the correct number of elements in the array.

The draw method is tested here by checking whether the output image matches the input.

Testing draw(Graphics graphics):

> DrawFractal drawTest = new DrawFractal(800, 800)

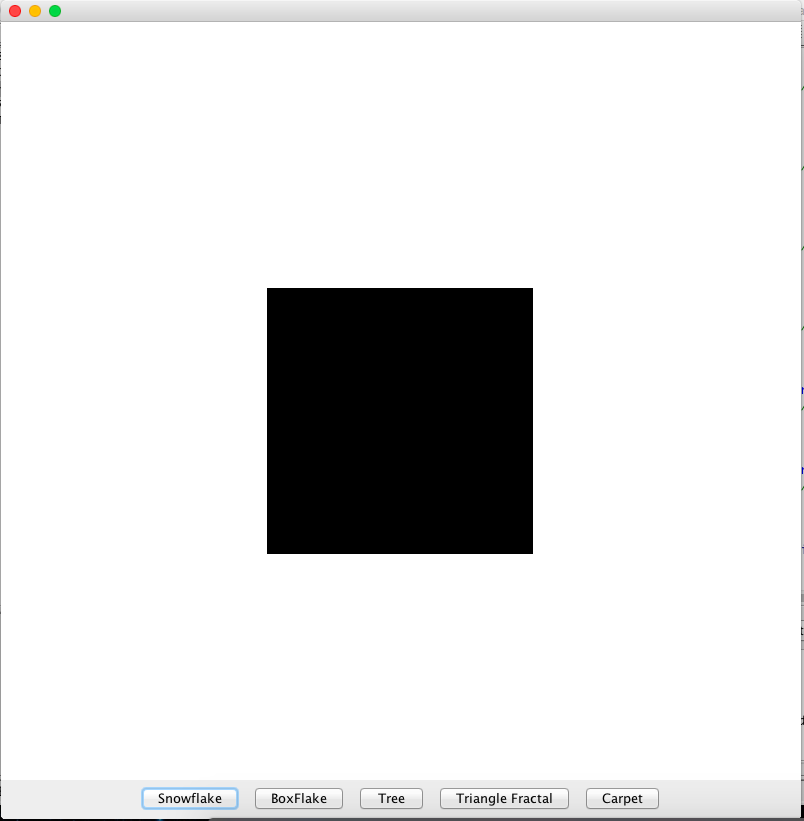
> Rectangle draw = new Rectangle(new Point(400, 400), 800)

> drawTest.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> drawTest.drawBaseShape(draw)

Produces:



This image is correct because the window was made 800x800; the image drawn is supposed to be the center rectangle of a rectangle with length 800, centered around the point (400, 400). Thus, the draw method works for the Rectangle class.

Testing Triangle class:

To test the Triangle, all that needs to be tested are the two constructors and the getLines() method. To test the two constructors and getLines() method, three instances of Triangle should be created where they test the cases of positive coordinates, negative coordinates, and coordinates containing zero. The exceptions that exist in the constructors should also be tested by creating cases where the exceptions will be thrown.

Testing Fractal class and FlakeFractal class

The Fractal class can be tested through the FlakeFractal class. This is because the draw method from Fractal class is inherited by the FlakeFractal class. The exception in the draw method must be tested by creating a case where the exception will be thrown but this can’t be tested by JUnit testing. This will have to be tested by attempting to draw the case using the interaction pane. This is due to the second parameter of the draw method, Graphics. To fully test the draw method, the input number of layers should be varied. For the purpose of base testing, zero layers and 1 layer will be tested. If these images are correct then it can be assumed that the method works correctly. The getBaseShapes() method can be tested by using the FlakeFractal class and insuring that it returns the correct base shapes.

To test the FlakeFractal class, the subFractals() method can be tested by getting the lines that form the FlakeFractal for 1 layer.

Test draw:

> Line testLine = new Line(400,400, 600, 600)

> FlakeFractal testFlake = new FlakeFractal(testLine)

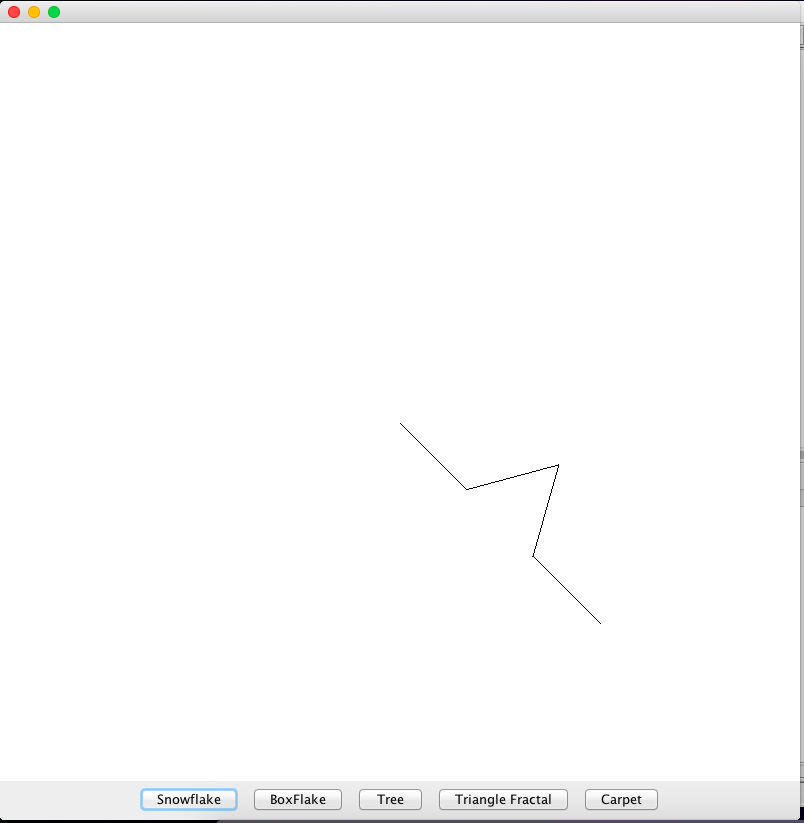
> DrawFractal testDraw = new DrawFractal(800, 800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(testFlake, 1)

Produces:



This is the correct image because the window was made to be 800x800 and the original line has the endpoints (400, 400) and (600, 600).

> Line line = new Line(200, 200, 600, 500)

> FlakeFractal testFlake = new FlakeFractal(line)

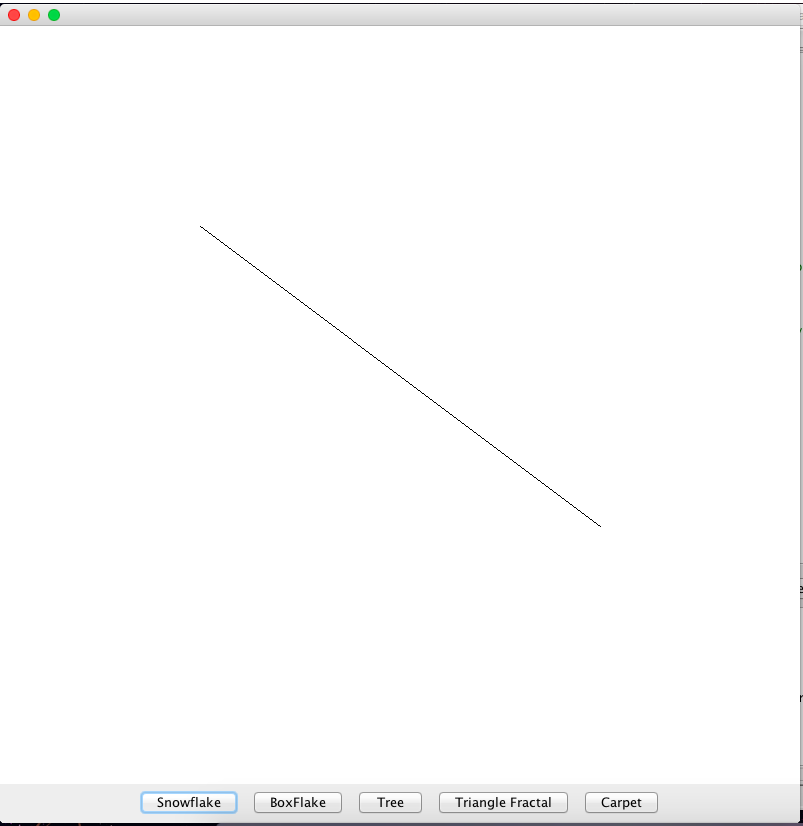
> DrawFractal test = new DrawFractal(800, 800)

> test.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> test.drawBaseFractal(testFlake, 0)

Produces:



This is the correct output because endpoints match of the original line on an 800x800 window and there are no layers on the fractal.

Testing the exception in draw:

> Line testLine = new Line(400,400, 600, 600)

> FlakeFractal testFlake = new FlakeFractal(testLine)

> DrawFractal testDraw = new DrawFractal(800, 800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(testFlake, -5)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at Fractal.draw(Fractal.java:39)

at DrawFractal.drawBaseFractal(DrawFractal.java:135)

Testing TreeFractal class:

To test the TreeFractal class, its subFractals() method and its draw method must be tested. The subFractals() method can be tested by creating an instance of TreeFractal and checking that it returns the correct lines for the first layer. The exception in the draw method must be tested as well.

Testing draw method:

> Line testLine = new Line(0, 200, 400, 400)

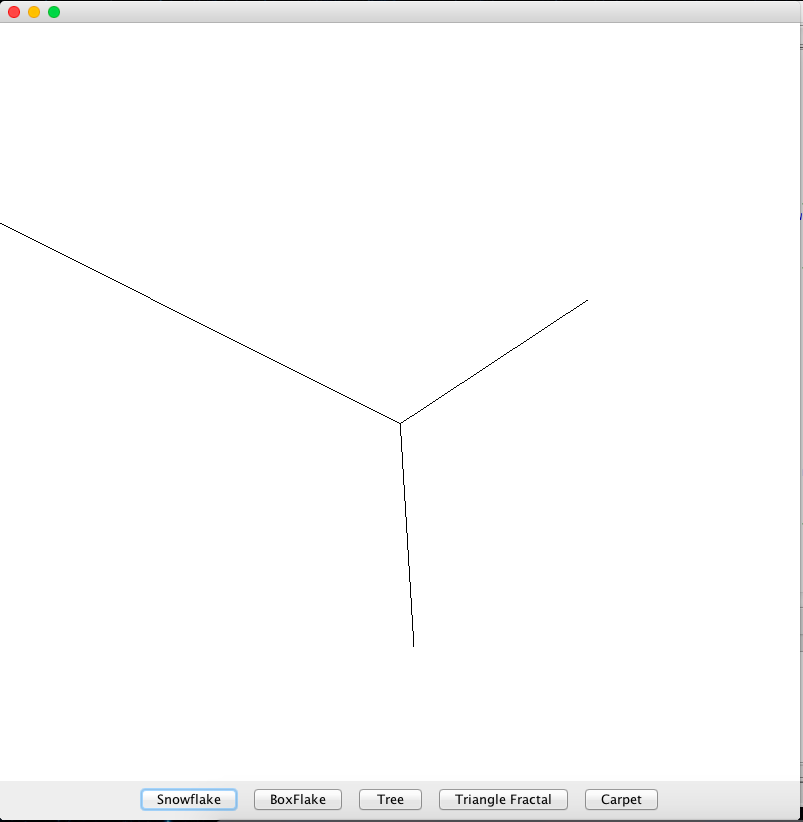
> TreeFractal testTree = new TreeFractal(testLine)

> DrawFractal testDraw = new DrawFractal(800, 800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(testTree, 1)

Produces:

This output image is correct because it draws the original line and 1 layer of the tree fractal with lengths of half of the original line.

> Line line = new Line(200, 200, 600, 500)

> TreeFractal treeFract = new TreeFractal(line)

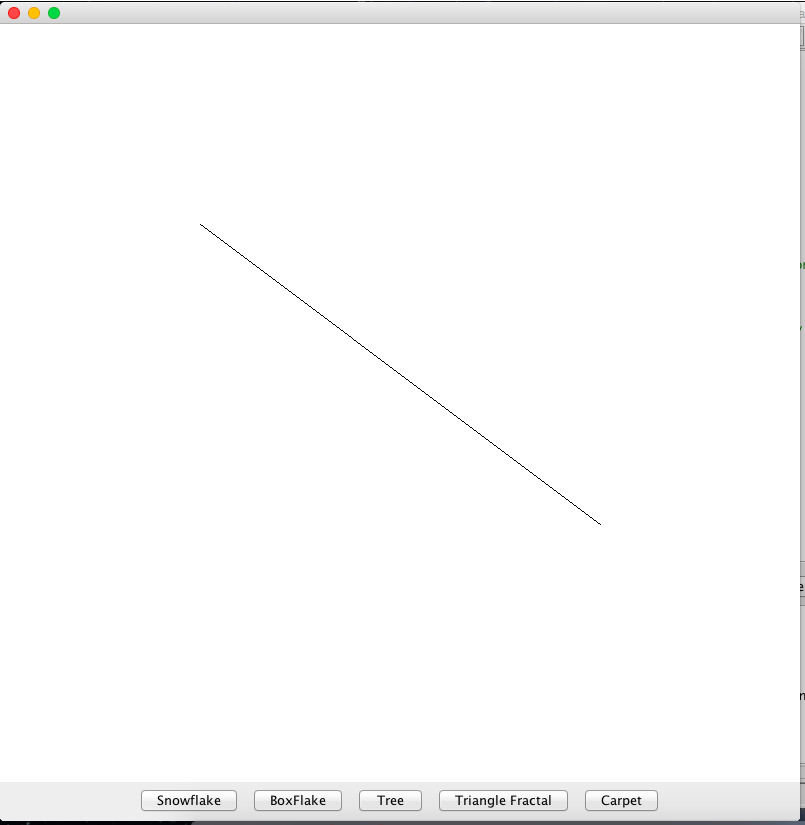
> DrawFractal test = new DrawFractal(800,800)

> test.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> test.drawBaseFractal(treeFract,0)

Produces:



This image is correct because there aren’t any layers and the image has the original endpoints.

Testing draw method exception:

> Line testLine = new Line(0, 200, 400, 400)

> TreeFractal testTree = new TreeFractal(testLine)

> DrawFractal testDraw = new DrawFractal(800, 800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(testTree, -5)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at TreeFractal.draw(TreeFractal.java:61)

at DrawFractal.drawBaseFractal(DrawFractal.java:135)

Testing TriangleFractal Class

To test the TriangleFractal class, only the subFractals() method needs to be tested. To test this method, an instance of TriangleFractal needs to be tested and compared against an array containing triangles with the same dimensions. This is to insure the triangles appear in the correct order with the correct coordinates.

Testing RectFractal Class

To test the RectFractal class, the subFractals() method, the draw method, and the draw method exception must be tested. The subFractals() method can be tested by creating an instance of RectFractal and testing its subFractals() method to insure the rectangles appear in the correct order with the correct coordinates.

Testing draw method:

> Rectangle rect = new Rectangle(new Point(400, 400), 800)

> RectFractal rectFract = new RectFractal(rect)

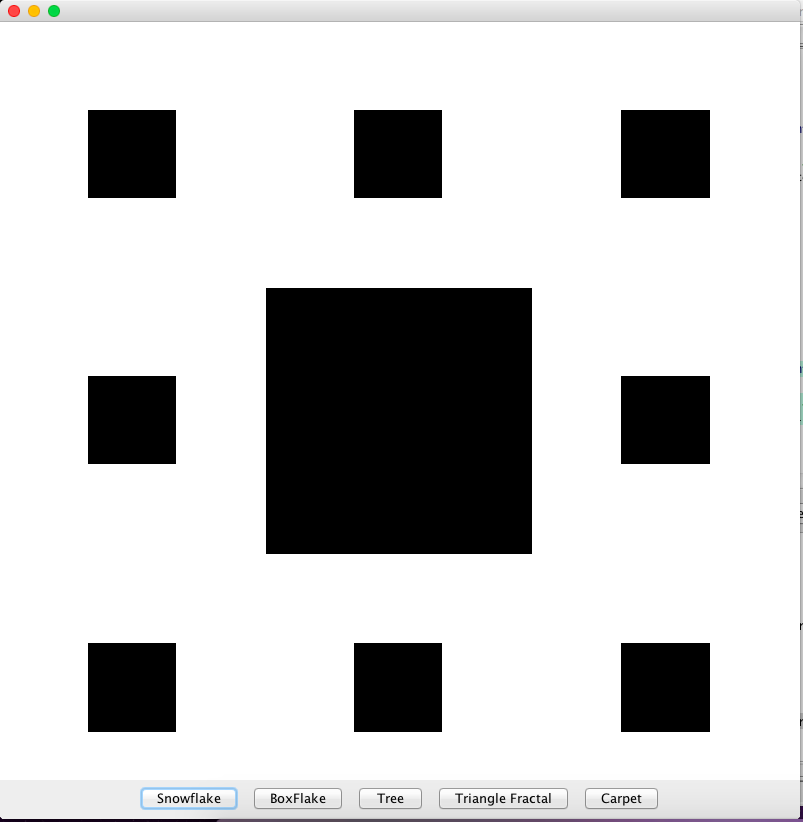
> DrawFractal testDraw = new DrawFractal(800, 800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(rectFract, 1)

Produces:

  
  
This image is correct because it draws the center of a square with dimensions fit to be the size of the window. This is what is expected from 1 layer of the fractal.

> Rectangle rect = new Rectangle(new Point(400, 400), 400)

> RectFractal testRectFract = new RectFractal(rect)

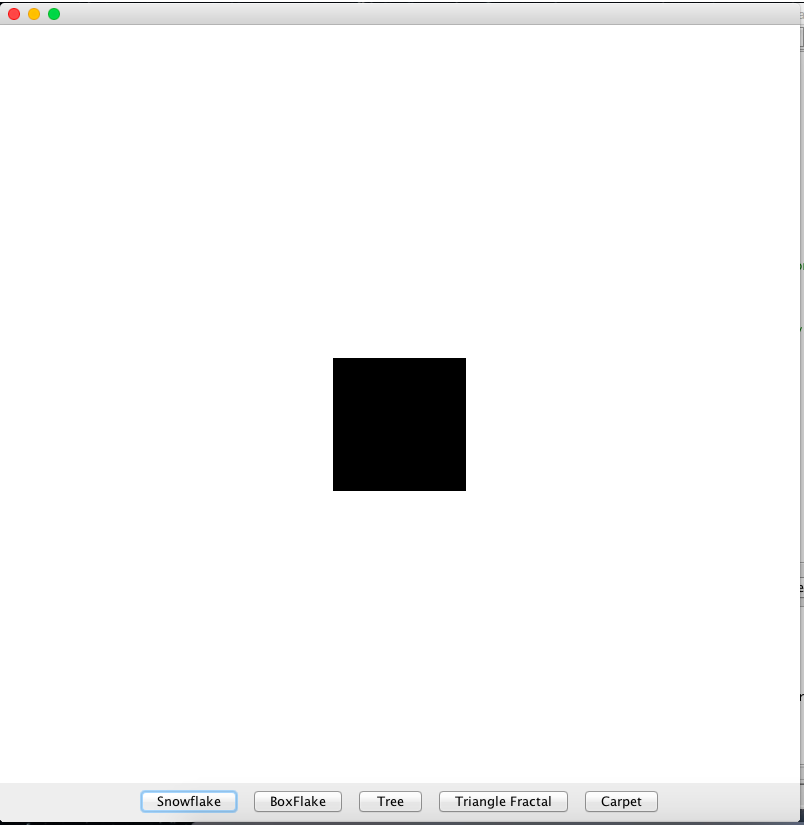
> DrawFractal test = new DrawFractal(800, 800)

> test.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> test.drawBaseFractal(testRectFract, 0)

Produces:



This output image is correct because 0 layers were input, so only the center rectangle of the original rectangle was drawn.

Testing exception in draw method:

> Rectangle square = new Rectangle(new Point(90, 90), 90)

> DrawFractal testDraw = new DrawFractal(800, 800)

> RectFractal testRectFract = new RectFractal(square)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawBaseFractal(testRectFract, -2)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at RectFractal.draw(RectFractal.java:85)

at DrawFractal.drawBaseFractal(DrawFractal.java:135)

Testing FractalPicture class and TreePicture class

The FractalPicture class can be tested through the TreePicture class. This is because all non-abstract methods in FractalPicture are directly inherited by TreePicture. To test the draw method of FractalPicture, a fractal picture can be tested with 0 objects in the input array (test 0), 1 object in the input array(test 1), and 2 objects in the input array (test many). The number of levels should be varied by 0 (test 0), 1 (test 1), 2 (test many), and -5 (test exception). The draw method will be tested using a method from the DrawFractal that calls the draw method of FractalPicture. The constructor should be tested as well to insure that the exception is thrown when there is an empty array input. The getBaseFractals() will be tested through the TreePicture class. To test the setFractal() method, a TreePicture instance will be created and one of the base fractals will be changed using the setFractals() method. For this method, the length of the original array will be varied by 0 (test 0) and 1 (test 1 and many). Then the first line of each base fractal will be changed(tests first), then middle of each base fractal (test middle), and then the last (test last).

To test the TreePicture class, all that needs to be tested is the getBaseFractals() method. It will be tested by creating TreePicture with 2 elements and checking that the return is correct for 0 layer of fractals. If the getBaseFractals() works with 2 elements, then it should work with 1 element as input as well. Zero elements will not be tested because the constructor does not accept 0 elements in the input array, so this exception is tested with the constructor.

The setFractal() fails because once the fractal has changed, it is not stored anywhere in the parent class. Therefore, it can’t be drawn because the draw method will simply call getBaseFractals() and the change that was made will be over-written.

Testing draw

> Triangle tri = new Triangle(new Point(400, 400), 400)

> TreePicture testTree = new TreePicture(new Triangle[]{tri})

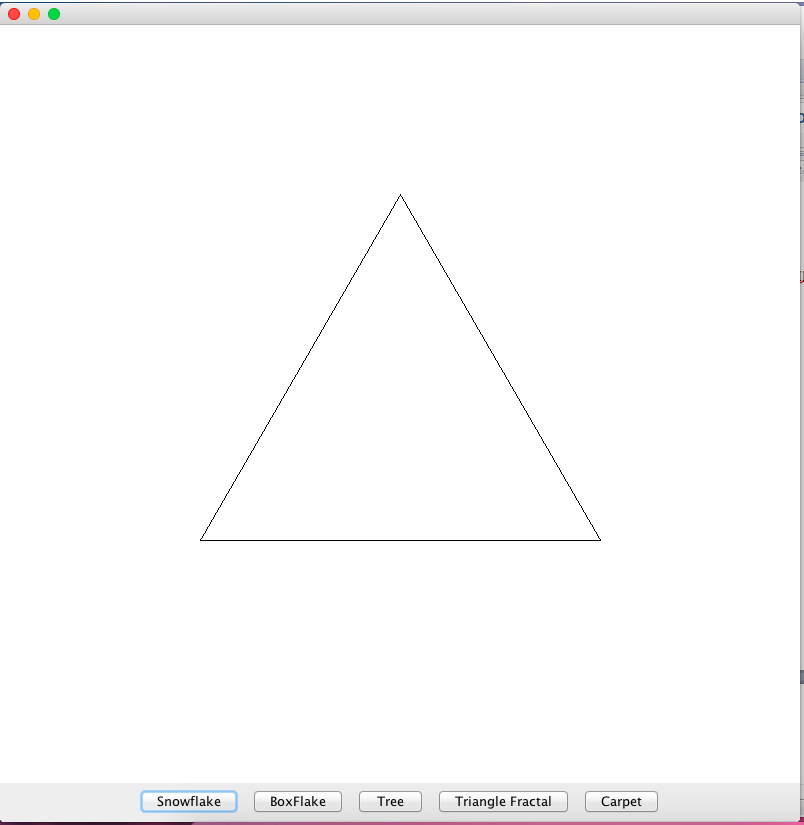
> DrawFractal testDraw = new DrawFractal(800,800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawFractal(testTree, 1)

Produces:



This is not the expected image because there should be 1 layer of fractals.

> Triangle tri = new Triangle(new Point(400, 400), 400)

> TreePicture testTree = new TreePicture(new Triangle[]{tri})

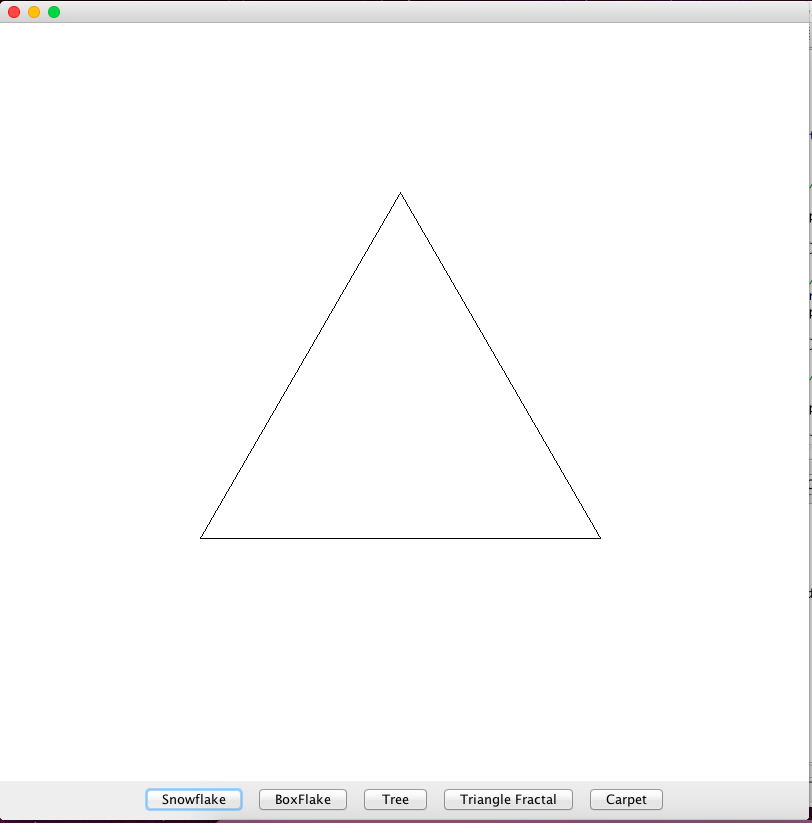
> DrawFractal testDraw = new DrawFractal(800,800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawFractal(testTree, 0)

Produces:



This image is correct because it produces the base shape.

Testing draw exception

> Triangle tri = new Triangle(new Point(400, 400), 400)

> TreePicture testTree = new TreePicture(new Triangle[]{tri})

> DrawFractal testDraw = new DrawFractal(800,800)

> testDraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testDraw.drawFractal(testTree, -5)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at FractalPicture.draw(FractalPicture.java:35)

at DrawFractal.drawFractal(DrawFractal.java:145)

Testing SnowflakePicture:

To test the SnowflakePicture class, the draw method exception must be tested, the draw method with level = 0, 1 must be tested, and the getBaseFractals() method of it must be tested. The draw method is tested such that level = 0 is test 0 and level = 1 is test 1 and many.

Testing draw method:

> Line line1 = new Line(200,200,400,400)

> Line line2 = new Line(500,500,600,600)

> SnowflakePicture testSnow = new SnowflakePicture(new Line[]{line1, line2});

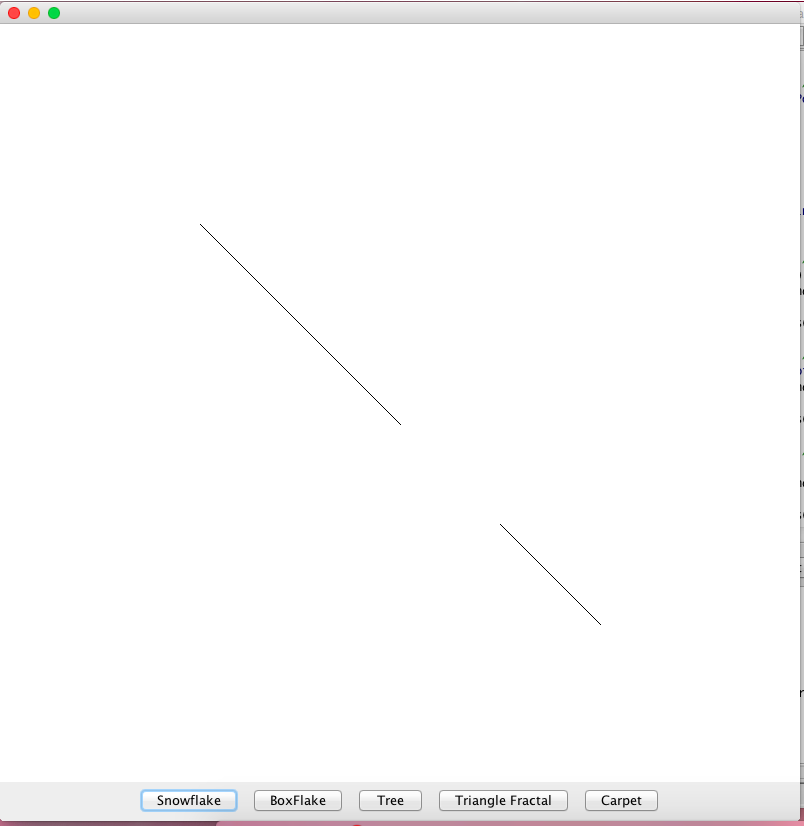
> DrawFractal drawTest = new DrawFractal(800,800)

> drawTest.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> drawTest.drawFractal(testSnow,0)

Produces:



This image is correct because the draw method produces the base shapes with the proper end points on an 800x800 window.

> Line line1 = new Line(200,200,400,400)

> Line line2 = new Line(500,500,600,600)

> SnowflakePicture testSnow = new SnowflakePicture(new Line[]{line1, line2});

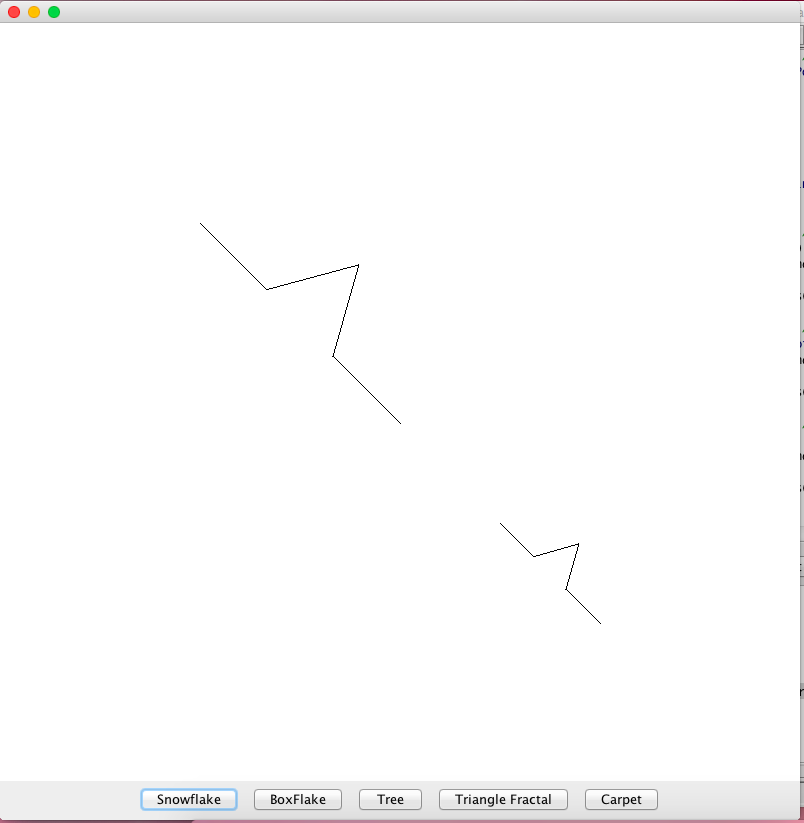
> DrawFractal drawTest = new DrawFractal(800,800)

> drawTest.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> drawTest.drawFractal(testSnow,1)

Produces:



This image is correct because it produces 1 layer of fractals with the proper endpoints on an 800x800 window.

Testing exception in draw:

> Line line1 = new Line(200,200,400,400)

> Line line2 = new Line(500,500,600,600)

> SnowflakePicture testSnow = new SnowflakePicture(new Line[]{line1, line2});

> DrawFractal drawTest = new DrawFractal(800,800)

> drawTest.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> drawTest.drawFractal(testSnow,-5)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at SnowflakePicture.draw(SnowflakePicture.java:35)

at DrawFractal.drawFractal(DrawFractal.java:145)

Testing TrianglePicture class:

To test the TrianglePicture class, all that needs to be tested is the getBaseFractals() method. It should be tested by creating an instance of TrianglePicture and checking it against an array of expected return lines. This can be done with 1 element in the input array(1 and many). For the getBaseFractals() test, only the first fractal triangle will be tested. If this test passes, then it can be assumed the rest of the array is correct since the method calls the TriangleFractal class, which has already been tested.

Testing CarpetPicture class:

To test the CarpetPicture class, the getBaseFractals() method and the draw method will be tested. The getBaseFractals() method will be tested with an input array of 1 element to test 1 and many. For the getBaseFractals() test, only the first fractal rectangle will be checked against an array of expected lines. If this test passes, then it can be assumed the rest of the array is correct since the method calls the RectangleFractal class, which as already been tested. To test the draw method, the method will be called through the DrawFractal class with varying levels of -5 (test exception), 0 (test 0), and 1 (test many and 1).

Test draw:

> Rectangle square = new Rectangle(new Point(400,400), 400)

> CarpetPicture testCarpet = new CarpetPicture(new Rectangle[]{square})

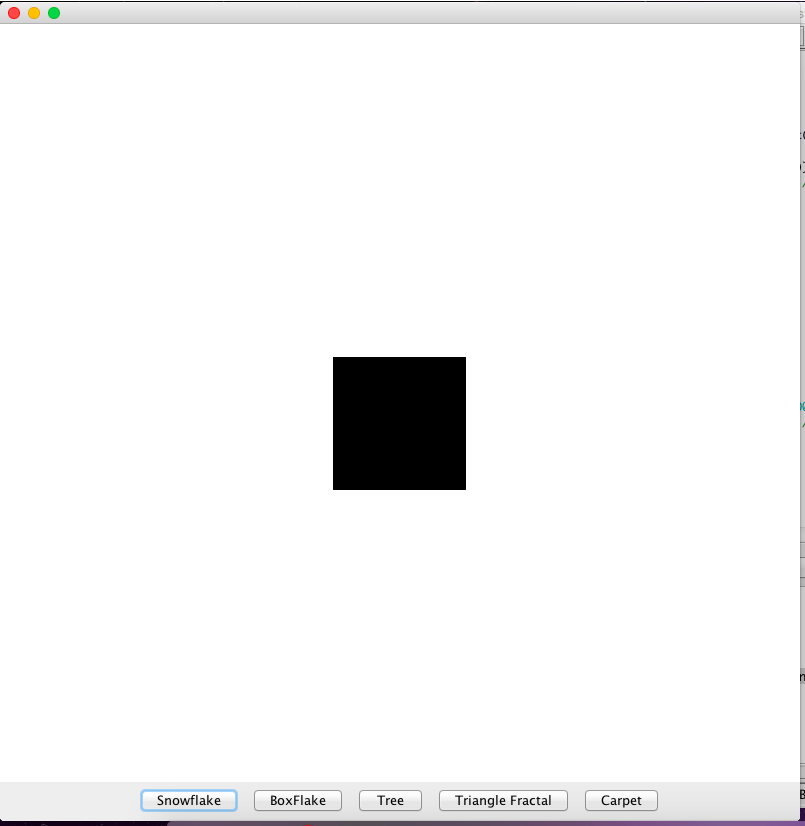
> DrawFractal testdraw = new DrawFractal(800,800)

> testdraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testdraw.drawFractal(testCarpet,0)

Produces:



This is the correct image because when there are 0 layers, only the center rectangle of the original rectangle is filled.

> Rectangle square = new Rectangle(new Point(400,400), 400)

> CarpetPicture testCarpet = new CarpetPicture(new Rectangle[]{square})

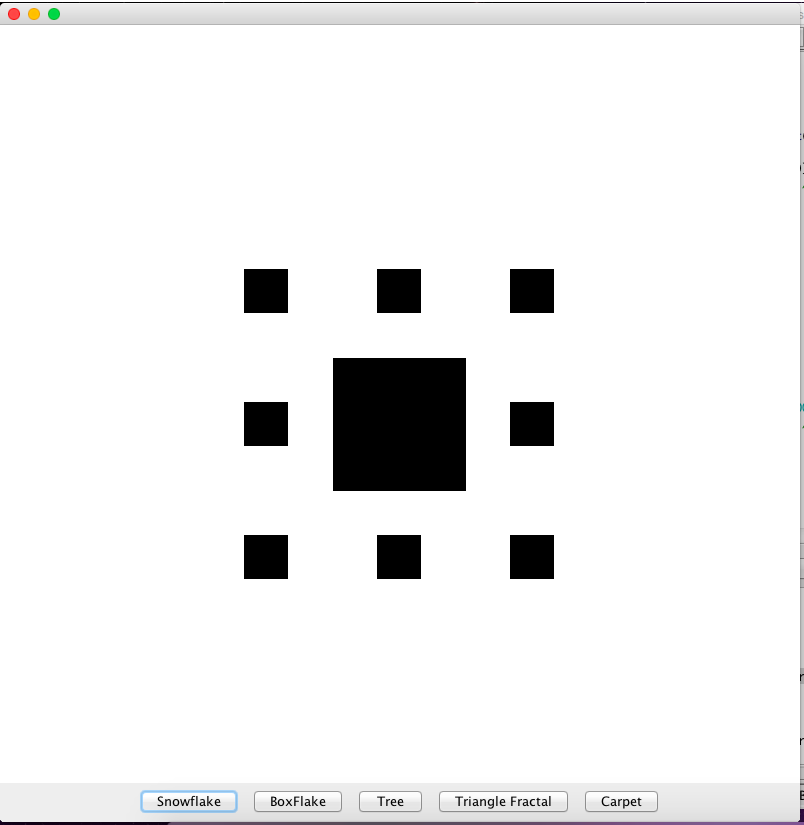
> DrawFractal testdraw = new DrawFractal(800,800)

> testdraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testdraw.drawFractal(testCarpet,1)

Produces:



This is the correct image because it produces 1 layer of fractals.

Test exception in draw:

> Rectangle square = new Rectangle(new Point(400,400), 400)

> CarpetPicture testCarpet = new CarpetPicture(new Rectangle[]{square})

> DrawFractal testdraw = new DrawFractal(800,800)

> testdraw.getGraphics()

sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> testdraw.drawFractal(testCarpet,-5)

java.lang.IllegalArgumentException: Fractal levels cannot be negative.

at CarpetPicture.draw(CarpetPicture.java:49)

at DrawFractal.drawFractal(DrawFractal.java:145)

Testing DrawFractal Class:

To test the DrawFractal class, the class should be compiled and the “Run” button should be clicked. The buttons can be tested by clicking them and changing the screen size to erase the canvas of the window. The exception in the constructor can be tested by creating an instance where the exception is thrown. All of the draw methods have been tested in the previous tests. The getGraphics() has also been tested in previous tests.

In order, the buttons produce:

