Ny Pham

1. Testing Point class

a) Testing the getX() method, getY()method, and the Point class constructor. This method should return the x-coordinate of the point that was declared and initialized.

> Point test = new Point(1,2)

> test.getX()

1

> test.getY()

2

This is correct because the method returns what the constructor initialized when the Point was declared.

b) Testing the setX(int x) method. This method should set a new x-coordinate without changing the y-coordinate. I will be using the same test Point from a).

> test.setX(3)

> test.getX()

3

> test.getY()

2

This is correct because the method sets a new x-coordinate, which the getX() method should return from the constructor.

c) Testing setY(int y) method. This method should set a new y-coordinate without changing the x-coordinate. I will be using the same test Point from a).

> test.setY(4)

> test.getY()

4

> test.getX()

3

This is correct because the method sets a new y-coordinate, which the getY() method should return from the constructor.

d) Testing getAngle() method. This method should return the angle between the point and the positive x-axis in radians. The angle will be given as if the the vector from the origin to the point rotates counter-clockwise about the origin.

> Point test1 = new Point(1,2)

> test1.getAngle()

1.1071487177940904

> Point test2 = new Point(-1,2)

> test2.getAngle()

2.0344439357957027

> Point test3 = new Point(-1,-2)

> test3.getAngle()

4.2487413713838835

> Point test4 = new Point(1,-2)

> test4.getAngle()

5.176036589385496

> Point test5 = new Point(1,0)

> test5.getAngle()

0.0

> Point test6 = new Point(0,1)

> test6.getAngle()

1.5707963267948966

> Point test7 = new Point(-1,0)

> test7.getAngle()

3.141592653589793

> Point test8 = new Point(0,-1)

> test8.getAngle()

4.71238898038469

This returns are correct because the output radians fall within the proper quadrants that the points were assigned.

e) Testing rotate(double angle) method. This method should rotate the a point by the input angle without significant changes to the radius (due to truncation).

> Point test1 = new Point(1,0)

> test1.rotate(Math.PI)

> test1.getX()

-1

> test1.getY()

0

> test1.rotate((Math.PI)/2)

> test1.getX()

0

> test1.getY()

-1

> Point test2 = new Point(1,2)

> test2.rotate((Math.PI)/2)

> test2.getX()

-2

> test2.getY()

1

> test2.rotate((Math.PI)/2)

> test2.getX()

-1

> test2.getY()

-2

> test2.rotate((Math.PI)/2)

> test2.getX()

2

> test2.getY()

-1

> test2.rotate(Math.PI)

> test2.getX()

-1

> test2.getY()

1

The outputs are correct because they fall into the proper quadrants after rotation.

2. Testing Line class

a) Testing getFirstPoint(), getSecondPoint(), the draw(Graphics graphics), and the Line (Point start, Point end) constructor of Line using the Sketch class. If this method works then a line should be drawn from the first point to second point declared.

> Sketch test = new Sketch(500,500)

> Point start = new Point(1,1)

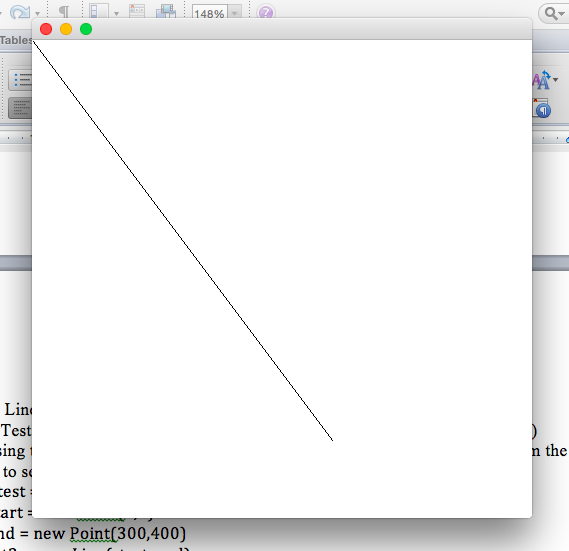
> Point end = new Point(300,400)

> Line test2 = new Line(start, end)

> 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.drawLine(test2)



These methods are correct because the draw(Graphics graphics) method calls the getFirstPoint() and getSecondPoint() to draw the line.

b) Testing the setFirstPoint(Point point) and setSecondPoint(Point point) methods. This should work if the line used in part a) is changed to new coordinates. This was added to the interaction pane used in part a).

> test.drawLine(test2)

> test2.setFirstPoint(new Point(300,200))

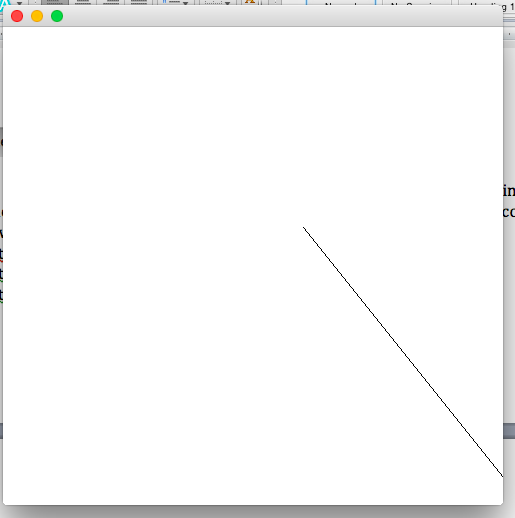
> test2.setSecondPoint(new Point(500,450))

> Sketch test = new Sketch(500,500)

> 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.drawLine(test2)



These methods are correct because new points are set for the line and then the line is drawn using the new points returned by getFirstPoint () and getSecondPoint() in the draw(Graphics graphics) method of Line.

c) Test of second constructor Line (int x1, int y1, int x2, int y2). If this constructor works then a line between (x1,y1) and (x2,y2) should be drawn using the draw(Graphics graphics) method of the Line class.

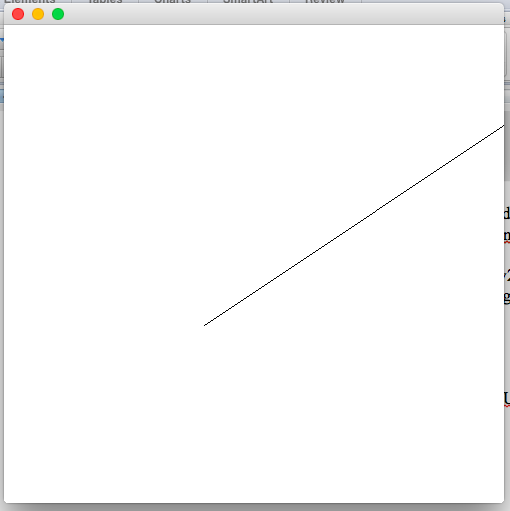
> Line test = new Line(200,300, 500,100)

> Sketch test2 = new Sketch(500,500)

> test2.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]]

> test2.drawLine(test)



3. Testing Fractal class

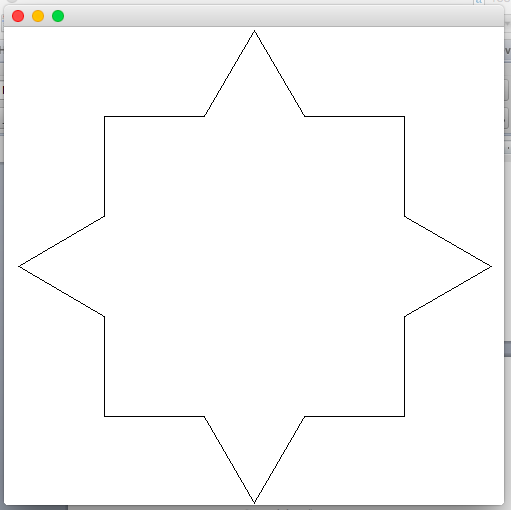
a) Testing the Fractal(Line line, int numberOfFractals) constructor, getLine(1-4)(), getFractal(1-4)(), and the draw(Graphics graphics) method of the Fractal class using the Sketch class. If these methods work then a fractal should be drawn if the number of layers designated is one or greater and if the number designated is zero or negative then a line will be drawn in place of the fractal.

>Sketch test = new Sketch(500,500)

> 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.drawBoxFlake(300,1)

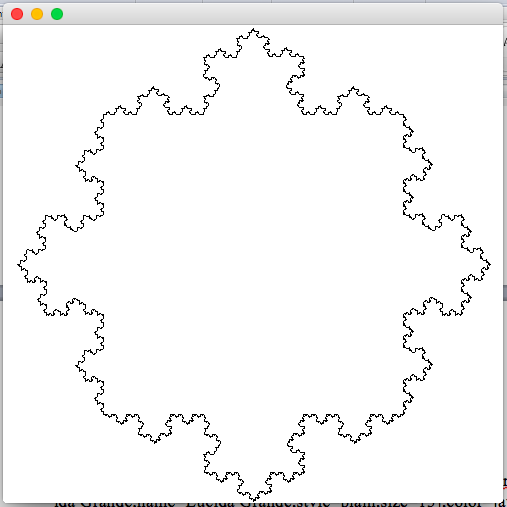


> Sketch test = new Sketch(500,500)

> 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]]

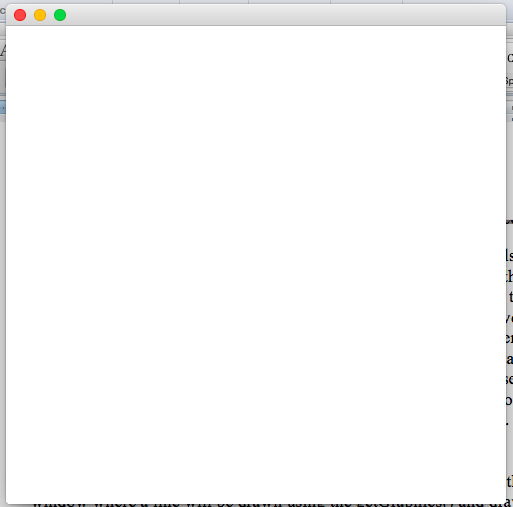
> test.drawBoxFlake(300,10)



The output images are correct because the Sketch method used calls the Fractal class and runs the conditional statements. For example, for the 1 layer of fractal, the constructor creates the fractal for all sides of the box and then terminates, meaning it runs the else statements in both the constructor and the draw method of Fractal. Since getLine(1-4)() worked in the else statements of the draw method, then the getLine methods worked. The 10 layered fractal created is correct because it run the Fractal class 10 times, creating 10 layers of fractals on each side of the original box. This shows the if statement of the constructor working because it constantly calls the Fractal within itself until numberOfFractals<1. The 10 layered fractals also showed that the draw method and getFractal(1-4) worked by creating the multiple layers.

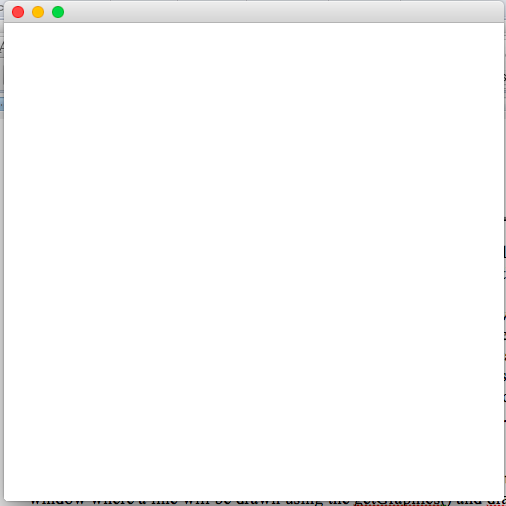
4. Testing Sketch method

a) Testing the constructor, getGraphics(), and drawLine. If these methods work, a window where a line will be drawn using the getGraphics() and drawLine method should open. > >Sketch test = new Sketch(500,500)



A new window with a white background opened, meaning the constructor worked.

> test.drawLine(new Line(200,300,400,100))

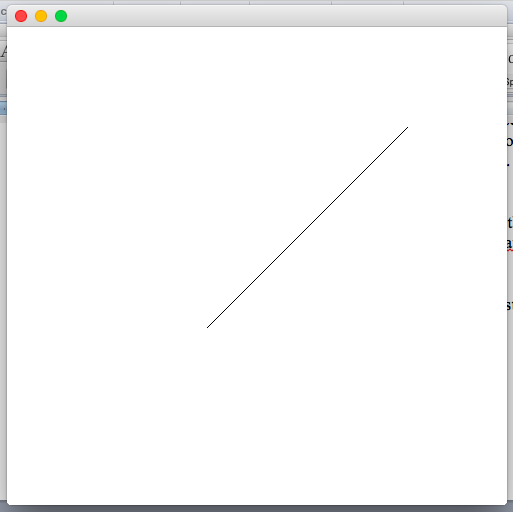


Nothing new appeared on the window, confirming that the if(graphics == null) condition of drawLine worked.

> test.getGraphics()

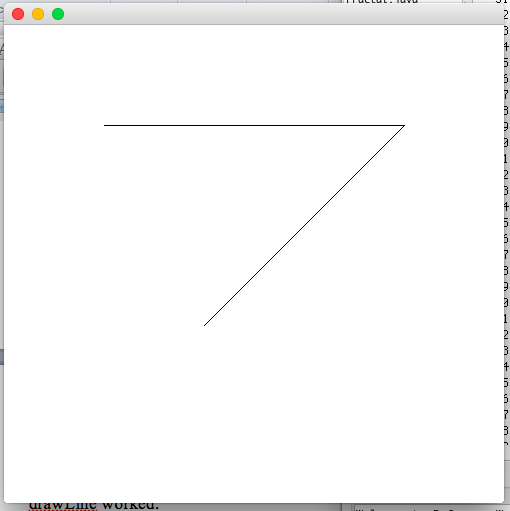
sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> test.drawLine(new Line(200,300,400,100))



The output image is correct because it showed the line from the drawLine method once the getGraphics method was called. Since graphics was null, it ran the else statement, storing the context to be used later on.

> test.drawLine(new Line(100,100,400,100))

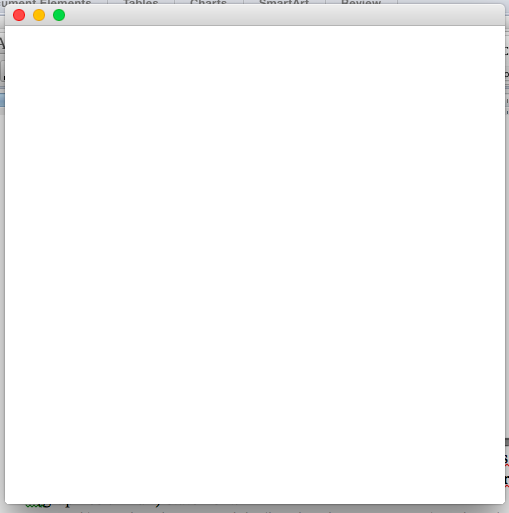


The output is correct because it runs the if statement of getGraphics when it is called inside of drawLine. Since getGraphics had already been called for the first drawLine, getGraphics ran the if(graphics != null) statement.

b) Testing drawBoxFlake(int size, int numLayers). What should happen is if getGraphics has not been called, then nothing is drawn. If getGraphics has been called then the method draws a square in the center of the canvas if numLayers is 0 or negative, if it is positive then it calls the Fractal class and draws the number of layers of fractals.

> Sketch test = new Sketch(500,500)

> test.drawBoxFlake(200,2)

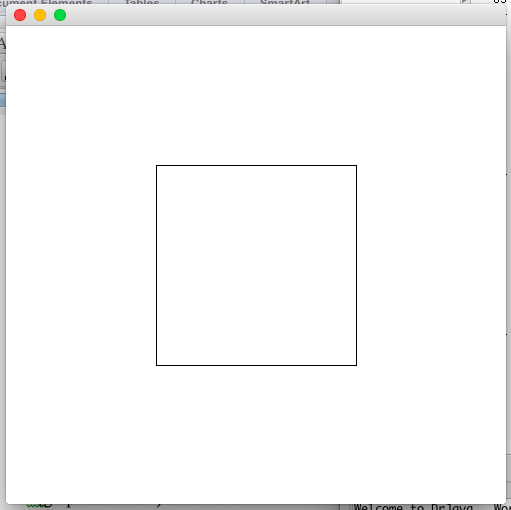


This is correct because getGraphics has not been called, therefore the method runs the if(graphics == null) statement.

> test.getGraphics()

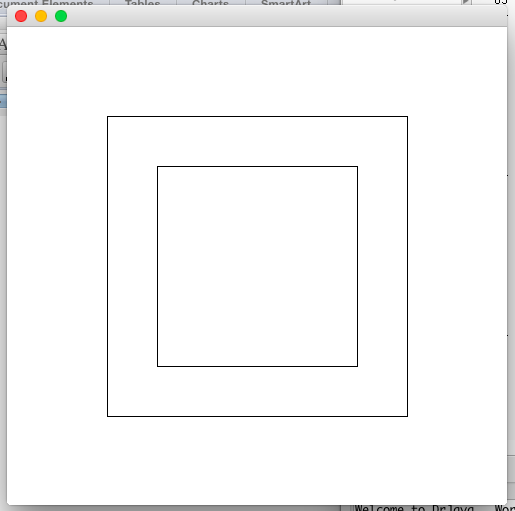
sun.java2d.SunGraphics2D[font=com.apple.laf.AquaFonts$DerivedUIResourceFont[family=Lucida Grande,name=Lucida Grande,style=plain,size=13],color=java.awt.Color[r=0,g=0,b=0]]

> test.drawBoxFlake(200,-5)



This is correct because the number of layers is negative, so it just draws a box, running the if (numLayers <= 0) statement.

> test.drawBoxFlake(300,0)



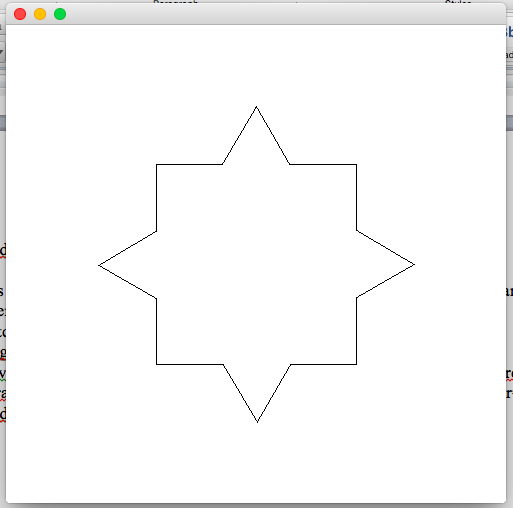
This is correct because size of the box increases while the shape is kept the same because the number of layers of fractals is 0, running the (numLayers <= 0) statement.

> Sketch test = new Sketch (500,500)

> 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.drawBoxFlake(200,1)



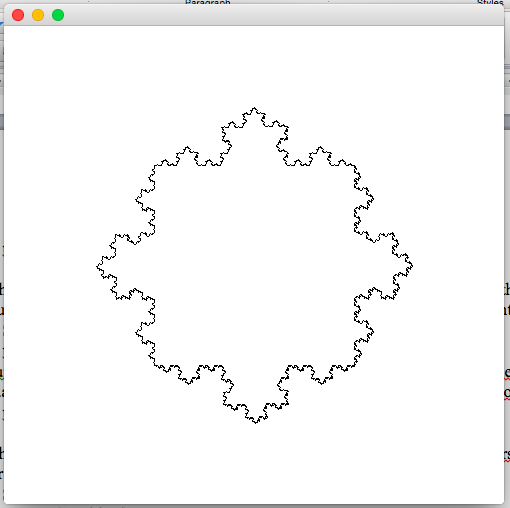
This output is correct because it runs the else statement where numLayers>0, thus calling the Fractal class, creating one layer of fractals, and drawing them.

> Sketch test = new Sketch (500,500)

> 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.drawBoxFlake(200,10)



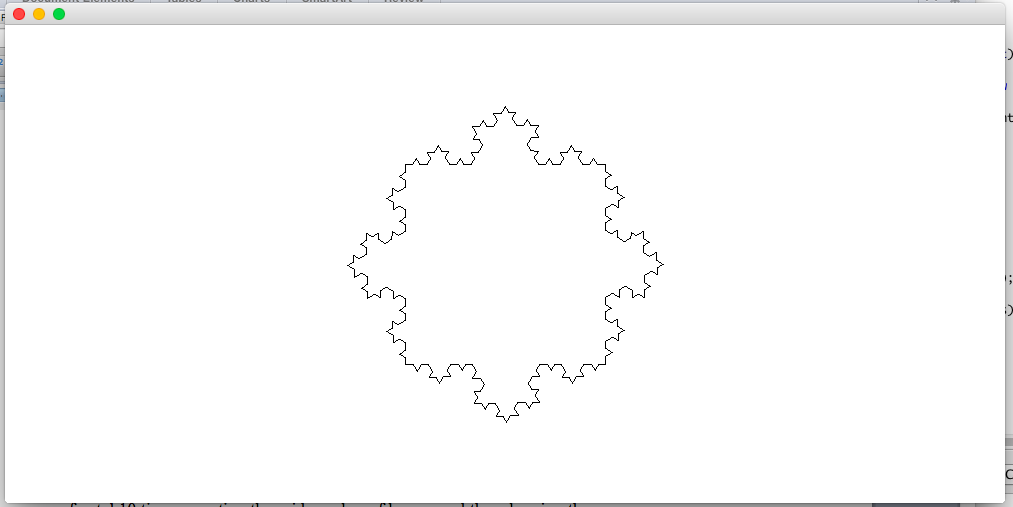
This output is correct because it runs the else statement where numLayers>0, thus calling the fractal 10 times, creating the said number of layers, and then drawing them.

> Sketch test = new Sketch(1000,500)

> 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.drawBoxFlake(200,3)

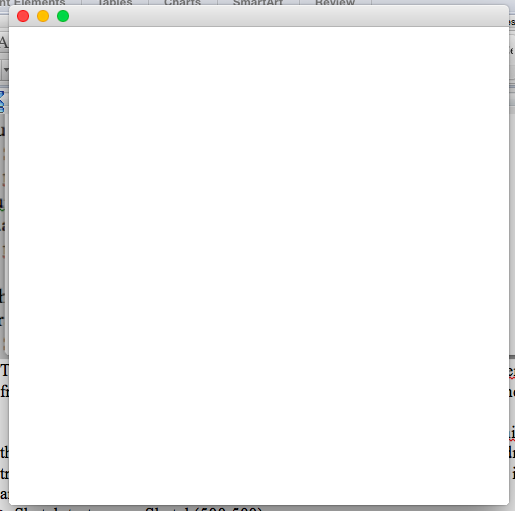


This output is correct because the shape of the canvas does not change the centering of box flake. This is because the dimensions of the canvas were taken into account when the method was written.

c) Testing drawSnowFlake().What should happen is if getGraphics has not been called, then nothing is drawn. If getGraphics has been called then the method draws an equilateral triangle in the center of the canvas(such that once its fractal is called, it is in the center of the canvas) if numLayers is 0 or negative, if it is positive then it calls the Fractal class and draws the number of layers of fractals.

> Sketch test = new Sketch(500,500)

> test.drawSnowFlake(200,1)



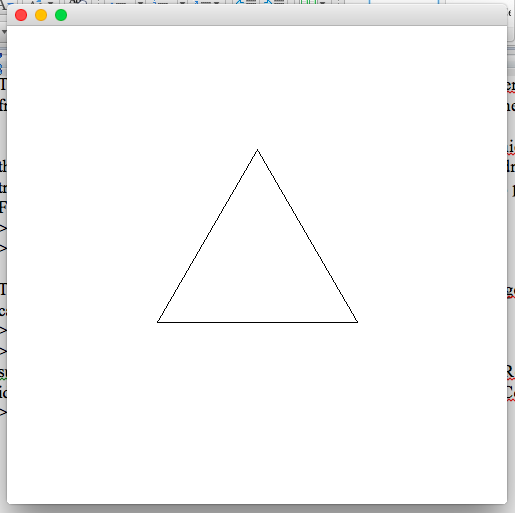
This is correct because runs the if(graphics == null) statement because getGraphics has not been called.

> Sketch test = new Sketch(500,500)

> 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.drawSnowFlake(200,-5)



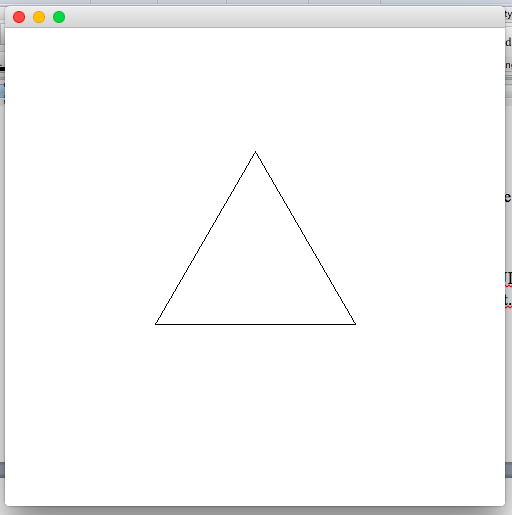
This is correct because the method runs the if (numLayers <= 0) statement because the number of layers is -5.

> Sketch test = new Sketch(500,500)

> 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.drawSnowFlake(200,0)



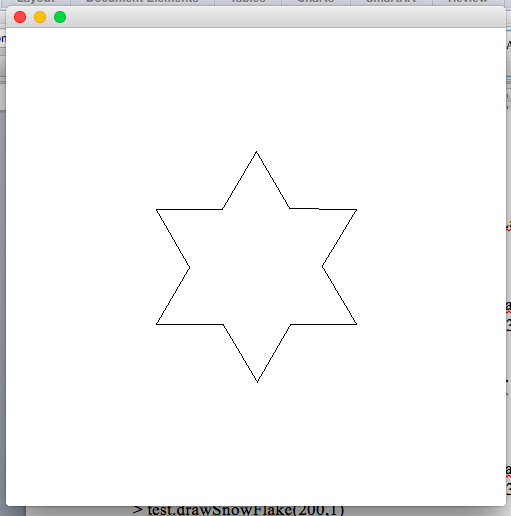
This output is correct because the method runs the if (numLayers <= 0) statement because the number of layers is 0.

> Sketch test = new Sketch(500,500)

> 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.drawSnowFlake(200,1)



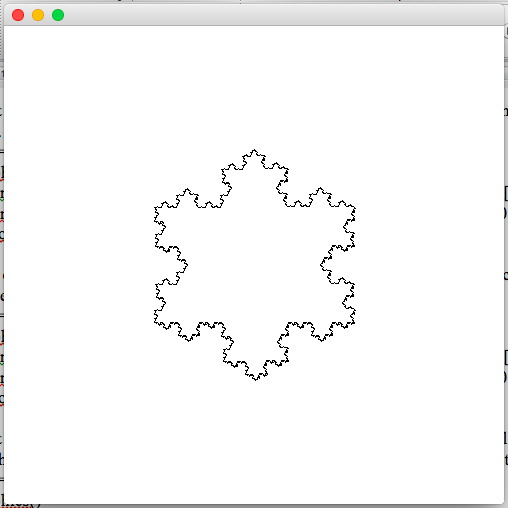
This is correct because the method runs the else statement because numLayers>0, calling the Fractal class the number of times assigned to numLayers. It only created 1 layer, so it is correct.

> Sketch test = new Sketch(500,500)

> 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.drawSnowFlake(200,5)



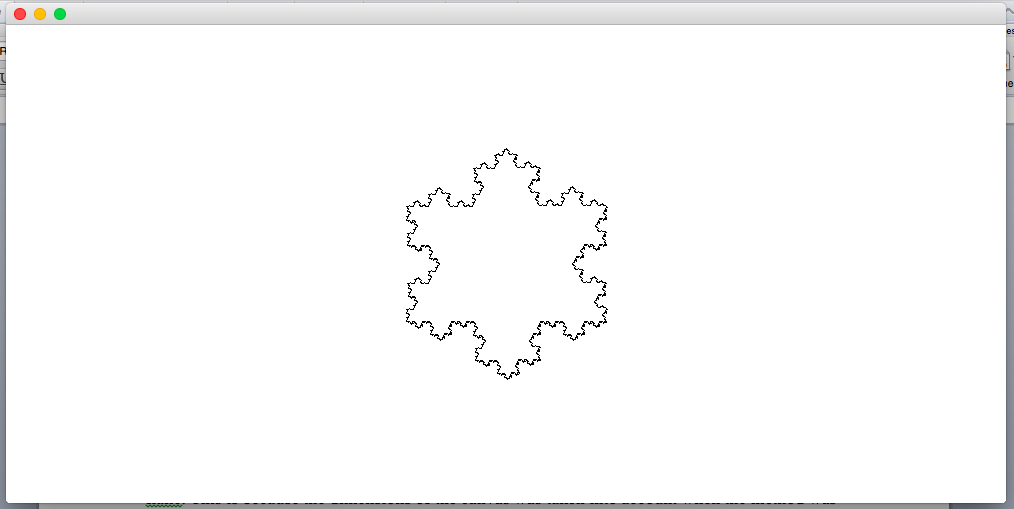
This output is correct because the method runs the else statement because numLayers>0, calling the Fractal class 5 times and drawing 5 layers of fractals.

> Sketch test = new Sketch(1000,500)

> 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.drawSnowFlake(200,5)



This output is correct because the shape of the canvas does not change the centering of snow flake. This is because the dimensions of the canvas were taken into account when the method was written.