CSE 116 (1-Mon 2000-2200 - A10): Semester-Long Project, by Aiden Cox

< Back to Summary

Assignment	CSE 116 (1-Mon 2000-2200 - A10): Semester-Long Project try #6
Name	Aiden Cox (aidencox)
Partners	Supratik Neupane (sneupane) Ryan Gangwish (ryangang) Sean Mackay (snmackay) Aiden Cox (aidencox)
Submitted	02/24/17 04:22PM, 1 day, 7 hrs, 36 mins early
Total Score	**0.0**/100.0

File	Remarks	Deductions	
MeetingMinutes/MinutesTemplate.txt	0	0.0	
src/edu/buffalo/cse116/Main.java	0	0.0	0.0%
src/fractals/BurningShip.java	0	0.0	100.0%
src/fractals/Fractals.java	0	0.0	100.0%
src/fractals/Julia.java	0	0.0	100.0%
src/fractals/Mandelbrot.java	0	0.0	100.0%
src/fractals/Multibrot.java	0	0.0	100.0%
src/fractalsTests/AllTests.java	0	0.0	0.0%
src/fractalsTests/BurningShipTest.java	0	0.0	100.0%
src/fractalsTests/FractalTests.java	0	0.0	100.0%
src/fractalsTests/JuiaTest.java	0	0.0	100.0%
src/fractalsTests/MandelbrotTest.java	0	0.0	100.0%
src/fractalsTests/MultibrotTest.java	0	0.0	100.0%

MeetingMinutes/MinutesTemplate.txt

```
Meeting Minutes for _____:

Meeting Attendance:
  * Ryan  U
  * Sean  M
  * Supratik
  * Aiden

Tasks completed since last meeting:
  * Completed first part
  * Not yet optimized

Tasks started, but not completed, since last meeting:
  * Optimization
  * Cleanup

Tasks to be worked on (and by which team members) for the next meeting:
  * Clean up code
  * Optimization

Schedule for the next week's set of pair programming meetings:
  * In Recitation next week
  * Wednesday, 5pm, Davis Hall
```

src/edu/buffalo/cse116/Main.java

```
package edu.buffalo.cse116;

public class Main {

  public static void main(String[] args) {




  }

}
```

```
 */


public class BurningShip extends Fractals {

    /**
     *Calls the constructor in the super class, assigns the staring and ending x and y
     * coordinates and calculates the ranges
     *
     */

    public BurningShip() {
        super();
        startX = -1.8;
        endX = -1.7;
        startY = -0.08;
        endY = 0.025;
        setRanges();
    }

    /* (non-Javadoc)
     * @see fractals.Fractals#getEscapeTime(double, double)
     */
    @Override
    public int getEscapeTime(double x, double y) {
        passes = 0;
        xCalc = x;
        yCalc = y;
        dist = distanceFromOrigin(xCalc, yCalc);

        while (dist <= 2.0 && passes < 255) {
             // tempX stores the value of xCalc before it's updated and used to update yCalc.
            double tempX = new Double(xCalc);
            xCalc = xCalc * xCalc - yCalc * yCalc + x;
            yCalc = (Math.abs(2 * tempX * yCalc) + y);

            dist = distanceFromOrigin(xCalc, yCalc);
            passes++;
        }
        escapeTime = passes;
        return escapeTime;
    }

}
```

src/fractals/Fractals.java

```
package fractals;

/**
 * Super class that provides the base for generating fractals and
 * calculating escape times for each point in the fractals.
 *
 * @author Supratik Neupane
 * @author Aiden Cox
 * @author Ryan Gangwish
 * @author Sean Mackay
 *
 */
public abstract class Fractals {

    /**
     * startX stores the starting value of the range of x-coordinates.
     */
    /**
     * endX store the final value of the range of x-coordinates.
     */
    /**
     * startY stores the starting value of the range of y-coordinates.
     */
    /**
     * endY stores the final value of the range of y-coordinates.
     */
    /**
     * rangeX stores the distance between two x-coordinates.
     */
    /**
     * rangeY stores the distance between two y-coordinates.
     */
    /**
     * dist stores the distance between a point and the origin at any given time in the program.
     */
    /**
     * xCalc is a variable used the update the value of the x-coordinate in the getEscapeTime method.
     */
    /**
     * yCalc is a variable used the update the value of the y-coordinate in the getEscapeTime method.
     */
    public double startX, endX, startY, endY, rangeX, rangeY, xCalc, yCalc, dist;
    /**
     * passes is a counter that increases by one every time the loop is entered in the getEscapeTime method;
     */
    /**
     * escapeTime is the final value of passes after it exits the loop.
     */
    protected int passes, escapeTime;
    /**
     * fractals store the values of the escape times of all the points in the fractal.
     */
    protected int[][] fractals;

    /**
     * Initializes fractals to a new 512 by 512 2-d array of type int.
     */
    public Fractals() {

        fractals = new int[512][512];
    }

    /**
     * Calculates the distance between the point and the origin using the
     * Pythagorean theorem.
     *
     * @param x
     *            The point's x-coordinate
     * @param y
     *            The point's y-coordinate
     * @return the distance between the point and the origin
     */
    public double distanceFromOrigin(double x, double y) {

        return Math.sqrt(Math.pow(x - 0, 2) + Math.pow(y - 0, 2));

    }

    /**
     * Calculates the distance between any two equally spaced x-coordinates and y-coordinates.
     */
    public void setRanges() {

        rangeX = (endX - startX) / 511;
        rangeY = (endY - startY) / 511;
    }

    /**
```

```
     *
     * @return a 2-d array that holds all the escape times for the 512 by 512 combination of
     *         rows and columns
     */
    public int[][] getFractals() {
        //loop through the fractal's rows
        for (int i = 0; i < 512; i++) {
            //loop through the fractal's columns
            for (int j = 0; j < 512; j++) {
                fractals[i][j] = getEscapeTime(getRangeValueX(i), getRangeValueY(j));
            }

        }
        return fractals;
    }

    /**
     * Translates the row to corresponding x-coordinate.
     *
     * @param i The row of the 2-d array
     * @return The translated x-coordinate of the corresponding row
     */
    public double getRangeValueX(int i) {
        //type cast i to double
        double a = (double) i;
        return startX + (a * rangeX);

    }


    /**
     * Translates the column to the corresponding y-coordinate.
     *
     * @param j The column of the 2-d array
     * @return The translated y-coordinate of the corresponding column
     */
    public double getRangeValueY(int j) {
        //type cast j to double
        double b = (double) j;
        return startY + (b * rangeY);
    }
}
```

src/fractals/Julia.java

```
package fractals;

/**
 * The Julia class is apart of the fractals package
 * Julia uses an equation to return the escape time
 * for a specific coordinate
 * @author Ryan, Aiden, Supratik & Sean
 * @see Fractals.java file
 */


public class Julia extends Fractals {

    /**
     * Calls the constructor in the super class, assigns the staring and ending x and y
     * coordinates and calculates the ranges
     */


    public Julia() {
        super();
        startX = -1.7;
        endX = 1.7;
        startY = -1.0;
        endY = 1.0;
        setRanges();
    }

    /* (non-Javadoc)
     * @see fractals.Fractals#getEscapeTime(double, double)
     */
    @Override
    public int getEscapeTime(double x, double y) {
        passes = 0;
        double k = -0.72689;
        double l = 0.188887;
        xCalc = x;
        yCalc = y;
        dist = distanceFromOrigin(xCalc, yCalc);

        while (dist <= 2.0 && passes < 255) {
            // tempX stores the value of xCalc before it's updated and used to update yCalc.
            double tempX = new Double(xCalc);
            xCalc = xCalc * xCalc - yCalc * yCalc + k;
            yCalc = 2 * tempX * yCalc + l;
            dist = distanceFromOrigin(xCalc, yCalc);
            passes++;
        }
        escapeTime = passes;
        return escapeTime;
    }

}
```

src/fractals/Mandelbrot.java

```
package fractals;

/**
 *The Mandelbrot class is apart of the
 * fractals package Mandelbrot uses an equation to return the escape time
 * for a specific coordinate
 *
 * @author superkid Ryan, Aiden, Supratik & Sean
 * @see Fractals.java file
 */
public class Mandelbrot extends Fractals {



    /**
     * Calls the constructor in the super class, assigns the staring and ending x and y
     * coordinates and calculates the ranges
     */
    public Mandelbrot() {
        super();
        startX = -2.15;
        startY = -1.3;
        endX = 0.6;
        endY = 1.3;
        setRanges();
    }


    /* (non-Javadoc)
     * @see fractals.Fractals#getEscapeTime(double, double)
```

```
44 
45             dist = distanceFromOrigin(xCalc, yCalc);
46             passes++;
47 
48         }
49 
50         escapeTime = passes;
51         return escapeTime;
52     }
53 
54 }
```

src/fractals/Multibrot.java

```
1  package fractals;
2  /**
3   * The Multibrot class is apart of the fractals package
4   * Multibrot uses an equation to return the escape time
5   * for a specific coordinate
6   * @author Ryan, Aiden, Supratik & Sean
7   * @see Fractals.java file
8   */
9 
10 public class Multibrot extends Fractals {
11     /**
12      *Calls the constructor in the super class, assigns the staring and ending x and y
13      * coordinates and calculates the ranges
14      */
15     public Multibrot() {
16         super();
17         startX = -1.0;
18         endX = 1.0;
19         startY = -1.3;
20         endY = 1.3;
21         setRanges();
22     }
23 
24     /* (non-Javadoc)
25      * @see fractals.Fractals#getEscapeTime(double, double)
26      */
27     @Override
28     public int getEscapeTime(double x, double y) {
29         passes = 0;
30         double xCalc = x;
31         double yCalc = y;
32         double dist = distanceFromOrigin(xCalc, yCalc);
33 
34         while (dist <= 2.0 && passes < 255) {
35             // tempX stores the value of xCalc before it's updated and used to update yCalc.
36             double tempX = new Double(xCalc);
37             xCalc = Math.pow(xCalc, 3) -(3*xCalc*yCalc*yCalc)+x;
38             yCalc = (3 * tempX *tempX* yCalc) -Math.pow(yCalc, 3)+ y;
39             dist = distanceFromOrigin(xCalc, yCalc);
40             passes++;
41         }
42         escapeTime = passes;
43         return escapeTime;
44     }
45 
46 }
```

src/fractalsTests/AllTests.java

```
1  package fractalsTests;
2 
3  import org.junit.runner.RunWith;
4  import org.junit.runners.Suite;
5  import org.junit.runners.Suite.SuiteClasses;
6 
7  /**
8   * Test suite that runs all the of the tests together.
9   *
10  *@author Supratik Neupane
11  * @author Sean Mackay
12  * @author Ryan Gangwish
13  * @author Aiden Cox
14  *
15  */
16 @RunWith(Suite.class)
17 @SuiteClasses({ BurningShipTest.class, JuiaTest.class, MandelbrotTest.class, MultibrotTest.class })
18 public class AllTests {
19 
20 }
```

src/fractalsTests/BurningShipTest.java

```
1  package fractalsTests;
2 
3      /**
4       * 'BurningShipTest' is a JUnit test that evaluates
5       * the BurningShip file
6       * 'FractalTests' is the superclass for this JUnit.
7       * @author Ryan, Aiden, Supratik & Sean
8       * @see FractalTests.java file
9       */
10 
11 
12 import static org.junit.Assert.*;
13 
14 import org.junit.Test;
15 import fractals.BurningShip;
16 
17 public class BurningShipTest extends FractalTests {
18 
19 
20     /* (non-Javadoc)
21      * @see fractalsTests.FractalTests#neverEscapeTest()
22      */
23     @Override
24     @Test
25     public void neverEscapeTest() {
26         BurningShip b = new BurningShip();
27         assertEquals(255, b.getEscapeTime(-1.7443359374999874, -0.017451171875000338));
28 
29     }
30 
31 
32     /* (non-Javadoc)
33      * @see fractalsTests.FractalTests#arraySizeTest()
34      */
35     @Override
36     @Test
37     public void arraySizeTest() {
38 
39         BurningShip b = new BurningShip();
40         assertEquals(512, b.getFractals().length);
41         assertEquals(512, b.getFractals()[511].length);
```

```
        assertEquals(x, b.getRangeValueX(z), 0.01);

    }


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#yCoordinateTest()
     */
    @Override
    @Test
    public void yCoordinateTest() {
        BurningShip b = new BurningShip();
        int z = r.nextInt(512);
        double y = (z * b.rangeY) + b.startY;
        assertEquals(y, b.getRangeValueY(z), 0.01);

    }

    /**makes sure no fractals escape time is equal
     * to 0 or 1
     */
    @Test
    public void noOneOrZeroEscapeTimeTest() {
        BurningShip b = new BurningShip();
        for (int i[] : b.getFractals()) {
            for (int j : i) {
                //assertNotEquals(0, j);
                //assertNotEquals(1, j);
                //

            }
        }
    }

}
```

src/fractalsTests/FractalTests.java

```
package fractalsTests;


import java.util.Random;


/**
 * Main class that has method headers for all the fractal set tests.
 * @author Supratik Neupane
 * @author Sean Mackay
 * @author Ryan Gangwish
 * @author Aiden Cox
 *
 */
public abstract class FractalTests {
    /**
     *  Random used in testing translation of x and y coordinates.
     */
    protected Random r;

    /**
     * Initializes random
     */
    public FractalTests() {
        r = new Random();
    }

    /**
     * Test if the point has an escapeTime of 255 i.e. the maximum number of passes into the loop
     */
    public abstract void neverEscapeTest();

    /**
     * Test if the method to create the 2-d array has 512 rows and 512 columns
     */
    public abstract void arraySizeTest();

    /**
     * Test if the row is translated to the corresponding x-coordinate
     */
    public abstract void xCoordinateTest();

    /**
     * Test if the row is translated to the corresponding y-coordinate
     */
    public abstract void yCoordinateTest();



}
```

src/fractalsTests/JuiaTest.java

```
package fractalsTests;

/**
 * 'JuliaTest' is a JUnit test that evaluates
 * the Julia file
 * 'FractalTests' is the superclass for this JUnit.
 * @author Ryan, Aiden, Supratik & Sean
 * @see FractalTests.java file
 */

import static org.junit.Assert.*;

import org.junit.Test;

import fractals.Julia;

public class JuiaTest extends FractalTests {


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#neverEscapeTest()
     */
    @Override
    @Test
    public void neverEscapeTest() {
        Julia j = new Julia();
        assertEquals(255, j.getEscapeTime(1.0492187499999897, -0.234375));

    }

    /**
     * Checks the escape time for a specific X and Y Value after one full loop
     * of the Julia class values.
     * checking to see that the escape time is equal to 1.
     */
    @Test
    public void escapesAfterOneLoopTest() {
        Julia j = new Julia();
        assertEquals(1, j.getEscapeTime(1.6933593749999853, 0.9765625));

```

```
54  
55      }
56  
57      /* (non-Javadoc)
58       * @see fractalsTests.FractalTests#xCoordinateTest()
59       */
60      @Override
61      @Test
62      public void xCoordinateTest() {
63          Julia j = new Julia();
64          int z = r.nextInt(512);
65          double x = (z * j.rangeX) + j.startX;
66          assertEquals(x, j.getRangeValueX(z), 0.01);
67          assertEquals(j.endX - j.rangeX, j.getRangeValueX(511), 0.01);
68  
69      }
70  
71      /* (non-Javadoc)
72       * @see fractalsTests.FractalTests#yCoordinateTest()
73       */
74      @Override
75      @Test
76      public void yCoordinateTest() {
77          Julia j = new Julia();
78          int z = r.nextInt(512);
79          double y = (z * j.rangeY) + j.startY;
80          assertEquals(y, j.getRangeValueY(z), 0.01);
81  
82      }
83  
84  }
```

src/fractalsTests/MandelbrotTest.java

```
1   package fractalsTests;
2   
3   import static org.junit.Assert.*;
4   
5   import org.junit.Test;
6   
7   import fractals.Mandelbrot;
8   /** @author Ryan,Aiden,Supratik & Sean
9   * 'MandelBrotTest' is a JUnit test that evaluates
10  * the MandelBrot file
11  * 'FractalTests' is the superclass for this JUnit test.
12  * @see FractalTests.java file
13  */
14  public class MandelbrotTest extends FractalTests {
15  
16      /* (non-Javadoc)
17       * @see fractalsTests.FractalTests#neverEscapeTest()
18       */
19      @Override
20      @Test
21      public void neverEscapeTest() {
22          Mandelbrot m = new Mandelbrot();
23          assertEquals(255, m.getEscapeTime(0.3207031250000001, -0.07109374999999386));
24  
25      }
26  
27      /**
28       * Test if the given point has the escape time of 1
29       */
30      @Test
31      public void escapesAfterOneLoopTest() {
32          Mandelbrot m = new Mandelbrot();
33          assertEquals(1, m.getEscapeTime(0.5946289062500001, 1.2949218750000122));
34      }
35  
36      /* (non-Javadoc)
37       * @see fractalsTests.FractalTests#arraySizeTest()
38       */
39      @Override
40      @Test
41      public void arraySizeTest() {
42  
43          Mandelbrot m = new Mandelbrot();
44          assertEquals(512, m.getFractals().length);
45          assertEquals(512, m.getFractals()[511].length);
46  
47      }
48  
49      /* (non-Javadoc)
50       * @see fractalsTests.FractalTests#xCoordinateTest()
51       */
52      @Override
53      @Test
54      public void xCoordinateTest() {
55          Mandelbrot m = new Mandelbrot();
56          int z = r.nextInt(512);
57          double x = (z * m.rangeX) + m.startX;
58          assertEquals(x, m.getRangeValueX(z), 0.01);
59  
60      }
61  
62      /* (non-Javadoc)
63       * @see fractalsTests.FractalTests#yCoordinateTest()
64       */
65      @Override
66      @Test
67      public void yCoordinateTest() {
68          Mandelbrot m = new Mandelbrot();
69          int z = r.nextInt(512);
70          double y = (z * m.rangeY) + m.startY;
71          assertEquals(y, m.getRangeValueY(z), 0.01);
72  
73      }
74  
75  }
```

src/fractalsTests/MultibrotTest.java

```
1   package fractalsTests;
2   
3   /**
4    * 'MultibrotTest' is a JUnit test that evaluates
5    * the Multibrot file
6    * 'FractalTests' is the superclass for this JUnit.
7    * @author Ryan, Aiden, Supratik & Sean
8    * @see Multibrot.java file
9    */
10  
11  import static org.junit.Assert.*;
12  import org.junit.Test;
13  
14  import fractals.Multibrot;
15  
16  
17  public class MultibrotTest extends FractalTests{
18  
19  
```

```
33      */
34
35     @Test
36     public void escapeAfterOneLoopTest(){
37         Multibrot m = new Multibrot();
38         assertEquals(1, m.getEscapeTime(0.9921875, 1.05625));
39     }
40
41
42     /* (non-Javadoc)
43      * @see fractalsTests.FractalTests#arraySizeTest()
44      */
45     @Test
46     @Override
47     public void arraySizeTest() {
48         Multibrot m = new Multibrot();
49         assertEquals(512, m.getFractals().length);
50         assertEquals(512, m.getFractals()[0].length);
51
52     }
53
54
55     /* (non-Javadoc)
56      * @see fractalsTests.FractalTests#xCoordinateTest()
57      */
58     @Test
59     @Override
60     public void xCoordinateTest() {
61         Multibrot m = new Multibrot();
62         int z = r.nextInt(512);
63         double x = (z * m.rangeX) + m.startX;
64         assertEquals(x, m.getRangeValueX(z), 0.01);
65
66     }
67
68     /* (non-Javadoc)
69      * @see fractalsTests.FractalTests#yCoordinateTest()
70      */
71     @Test
72     @Override
73     public void yCoordinateTest() {
74         Multibrot m = new Multibrot();
75         int z = r.nextInt(512);
76         double y = (z * m.rangeY) + m.startY;
77         assertEquals(y, m.getRangeValueY(z), 0.01);
78
79     }
80
81
82
83 }
```

< Back to Summary