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

< Back to Summary

Assignment	CSE 116 (1-Mon 2000-2200 - A10): Semester-Long Project try #8
Name	Aiden Cox (aidencox)
Partners	Sean Mackay (snmackay) Ryan Gangwish (ryangang) Supratik Neupane (sneupane) Aiden Cox (aidencox)
Submitted	02/25/17 01:16PM, 10 hrs, 42 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/FractalTestsAbstractClass.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/BurningShipTest.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	83.3%
Submit Results.pdf	0	0.0	

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

```java
package edu.buffalo.cse116;

public class Main {

    public static void main(String[] args) {

    }

}
```

src/fractals/BurningShip.java

```java
package fractals;

/**
```

```java

     */

    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

```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 using the formula
     * for arithmetic sequence. common difference =(last-first)/(n-1)
     */
    public void setRanges() {

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

    /**
     * Calculates the number of times a point goes through a loop before it's
     * distance from the origin exceeds the escape distance or if it never
     * exceeds the escape distance.
     *
     * The while loop checks if the the dist is<=2.0 or if the passes<255
     *
     * @param x The point's x-coordinate
     * @param y The point's y-coordinate
     * @return The number of times the point enters a loop before it escapes
     */
```

```java

        }
        return fractals;
    }

    /**
     * Translates the row to corresponding x-coordinate.
     * If we treat it as an arithmetic sequence with common difference of rangeX,
     * then the nth term will be startX + (n-1)*rangeX.
     * But the index in our 2-d array starts from 0 so we disregard the (n-1) and just use n.
     *
     * @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.
     *
     * If we treat it as an arithmetic sequence with common difference of rangeY,
     * then the nth term will be startY + (n-1)*rangeY.
     * But the index in our 2-d array starts from 0 so we disregard the (n-1) and just use n.
     *
     * @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/FractalTestsAbstractClass.java

```java
package fractals;


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 FractalTestsAbstractClass {
    /**
     *  Random used in testing translation of x and y coordinates.
     */
    protected Random r;

    /**
     * Initializes random
     */
    public FractalTestsAbstractClass() {
        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/fractals/Julia.java

```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) {
```

src/fractals/Mandelbrot.java

```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)
     */
    @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 = (2 * tempX * yCalc) + y;

            dist = distanceFromOrigin(xCalc, yCalc);
            passes++;

        }

        escapeTime = passes;
        return escapeTime;
    }

}
```

src/fractals/Multibrot.java

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

public class Multibrot extends Fractals {
    /**
     *Calls the constructor in the super class, assigns the staring and ending x and y
     * coordinates and calculates the ranges
     */
    public Multibrot() {
        super();
        startX = -1.0;
        endX = 1.0;
        startY = -1.3;
        endY = 1.3;
        setRanges();
    }

    /* (non-Javadoc)
     * @see fractals.Fractals#getEscapeTime(double, double)
     */
    @Override
    public int getEscapeTime(double x, double y) {
        passes = 0;
        double xCalc = x;
        double yCalc = y;
        double 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 = Math.pow(xCalc, 3) -(3*xCalc*yCalc*yCalc)+x;
            yCalc = (3 * tempX *tempX* yCalc) -Math.pow(yCalc, 3)+ y;
            dist = distanceFromOrigin(xCalc, yCalc);
            passes++;
        }
        escapeTime = passes;
        return escapeTime;
    }

}
```

src/fractalsTests/BurningShipTest.java

```java
package fractalsTests;

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


import static org.junit.Assert.*;
import fractals.FractalTestsAbstractClass;
import org.junit.Test;
import fractals.BurningShip;

public class BurningShipTest extends FractalTestsAbstractClass {


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#neverEscapeTest()
     */
    @Override
    @Test
    public void neverEscapeTest() {
        BurningShip b = new BurningShip();
        assertEquals(512, b.getFractals().length);
        assertEquals(512, b.getFractals()[511].length);

    }


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#xCoordinateTest()
     */
    @Override
    @Test
    public void xCoordinateTest() {
        BurningShip b = new BurningShip();
        int z = r.nextInt(512);
        double x = (z * b.rangeX) + b.startX;
        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);
        assertEquals(b.endY,b.getRangeValueY(511),0.001);
    }

    /**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) {
                assertFalse(j==0);
                assertFalse(j==1);


            }
        }

    }

}
```

src/fractalsTests/JuiaTest.java

```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 fractals.FractalTestsAbstractClass.java file
 */

import static org.junit.Assert.*;
import fractals.FractalTestsAbstractClass;
import org.junit.Test;

import fractals.Julia;

public class JuiaTest extends FractalTestsAbstractClass {


    /* (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));

    }


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#arraySizeTest()
     */
    @Override
    @Test
    public void arraySizeTest() {

        Julia j = new Julia();
        assertEquals(512, j.getFractals().length);
        assertEquals(512, j.getFractals()[511].length);

    }

    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#xCoordinateTest()
     */
    @Override
    @Test
    public void xCoordinateTest() {
        Julia j = new Julia();
        int z = r.nextInt(512);
        double x = (z * j.rangeX) + j.startX;
        assertEquals(x, j.getRangeValueX(z), 0.01);
        assertEquals(j.endX , j.getRangeValueX(511), 0.01);

    }

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

    }

}
```

```java
* 'MandelBrotTest' is a JUnit test that evaluates
* the MandelBrot file
* 'FractalTests' is the superclass for this JUnit test.
* @see fractals.FractalTestsAbstractClass.java file
*/
public class MandelbrotTest extends FractalTestsAbstractClass {

    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#neverEscapeTest()
     */
    @Override
    @Test
    public void neverEscapeTest() {
        Mandelbrot m = new Mandelbrot();
        assertEquals(255, m.getEscapeTime(0.3207031250000001, -0.07109374999999386));

    }

    /**
     * Test if the given point has the escape time of 1
     */
    @Test
    public void escapesAfterOneLoopTest() {
        Mandelbrot m = new Mandelbrot();
        assertEquals(1, m.getEscapeTime(0.5946289062500001, 1.2949218750000122));
    }

    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#arraySizeTest()
     */
    @Override
    @Test
    public void arraySizeTest() {

        Mandelbrot m = new Mandelbrot();
        assertEquals(512, m.getFractals().length);
        assertEquals(512, m.getFractals()[511].length);

    }

    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#xCoordinateTest()
     */
    @Override
    @Test
    public void xCoordinateTest() {
        Mandelbrot m = new Mandelbrot();
        int z = r.nextInt(512);
        double x = (z * m.rangeX) + m.startX;
        assertEquals(x, m.getRangeValueX(z), 0.01);

    }

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

    }

}
```

src/fractalsTests/MultibrotTest.java

```java
package fractalsTests;

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

import static org.junit.Assert.*;
import org.junit.Test;
import fractals.FractalTestsAbstractClass;
import fractals.Multibrot;


public class MultibrotTest extends FractalTestsAbstractClass{


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#neverEscapeTest()
     */
    @Test
    @Override
    public void neverEscapeTest() {
        Multibrot m = new Multibrot();
        assertEquals(255, m.getEscapeTime(0.5859375, 0.24375000000000108));
    }

    /**
     * escapeAfterOneLoopTest runs getEscapeTime of Multibrot and confirms that
     * a point escapes after one run through the algorithm
     */

    @Test
    public void escapeAfterOneLoopTest(){
        Multibrot m = new Multibrot();
        assertEquals(1, m.getEscapeTime(0.9921875, 1.05625));
    }


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#arraySizeTest()
     */
    @Test
    @Override
    public void arraySizeTest() {
        Multibrot m = new Multibrot();
        assertEquals(512, m.getFractals().length);
        assertEquals(512, m.getFractals()[0].length);

    }


    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#xCoordinateTest()
     */
    @Test
    @Override
    public void xCoordinateTest() {
        Multibrot m = new Multibrot();
        int z = r.nextInt(512);
        double x = (z * m.rangeX) + m.startX;
        assertEquals(x, m.getRangeValueX(z), 0.01);
        assertEquals(m.startX,m.getRangeValueX(0),0.01);
        assertEquals(m.endX,m.getRangeValueX(511),0.01);
    }

    /* (non-Javadoc)
     * @see fractalsTests.FractalTests#yCoordinateTest()
```

< Back to Summary