

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

[< Back to Summary](#)

<b>Assignment</b>	CSE 116 (1-Mon 2000-2200 - A10): Semester-Long Project try #8
<b>Name</b>	Aiden Cox (aidencox)
<b>Partners</b>	Sean Mackay (snmackay) Ryan Gangwish (ryangang) Supratik Neupane (sneupane) Aiden Cox (aidencox)
<b>Submitted</b>	02/25/17 01:16PM, 10 hrs, 42 mins early
<b>Total Score</b>	<b>0.0</b> /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/JuliaTest.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

```

1 Meeting Minutes for _____:
2
3 Meeting Attendance:
4   * Ryan U
5   * Sean M
6   * Supratik
7   * Aiden
8
9 Tasks completed since last meeting:
10  * Completed first part
11  * Not yet optimized
12
13 Tasks started, but not completed, since last meeting:
14  * Optimization
15  * Cleanup
16
17 Tasks to be worked on (and by which team members) for the next meeting:
18  * Clean up code
19  * Optimization
20
21 Schedule for the next week's set of pair programming meetings:
22  * In Recitation next week
23  * Wednesday, 5pm, Davis Hall

```

### src/edu/buffalo/cse116/Main.java

```

1 package edu.buffalo.cse116;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7     }
8
9 }

```

### src/fractals/BurningShip.java

```

1 package fractals;
2
3 /**

```

```
17 |  
18 |  
19 |  
20 | public BurningShip() {  
21 |     super();  
22 |     startX = -1.8;  
23 |     endX = -1.7;  
24 |     startY = -0.08;  
25 |     endY = 0.025;  
26 |     setRanges();  
27 | }  
28 |  
29 | /* (non-Javadoc)  
30 | * @see fractals.Fractals#getEscapeTime(double, double)  
31 | */  
32 | @Override  
33 | public int getEscapeTime(double x, double y) {  
34 |     passes = 0;  
35 |     xCalc = x;  
36 |     yCalc = y;  
37 |     dist = distanceFromOrigin(xCalc, yCalc);  
38 |  
39 |     while (dist <= 2.0 && passes < 255) {  
40 |         // tempX stores the value of xCalc before it's updated and used to update yCalc.  
41 |         double tempX = new Double(xCalc);  
42 |         xCalc = xCalc * xCalc - yCalc * yCalc + x;  
43 |         yCalc = (Math.abs(2 * tempX * yCalc) + y);  
44 |  
45 |         dist = distanceFromOrigin(xCalc, yCalc);  
46 |         passes++;  
47 |     }  
48 |     escapeTime = passes;  
49 |     return escapeTime;  
50 | }  
51 |  
52 |  
53 |  
54 | }
```

## src/fractals/Fractals.java

```
1 | package fractals;  
2 |  
3 | /**  
4 | * Super class that provides the base for generating fractals and  
5 | * calculating escape times for each point in the fractals.  
6 | *  
7 | * @author Supratik Neupane  
8 | * @author Aiden Cox  
9 | * @author Ryan Gangwish  
10 | * @author Sean Mackay  
11 | */  
12 | public abstract class Fractals {  
13 |  
14 |     /**  
15 |     * startX stores the starting value of the range of x-coordinates.  
16 |     */  
17 |     /**  
18 |     * endX store the final value of the range of x-coordinates.  
19 |     */  
20 |     /**  
21 |     * startY stores the starting value of the range of y-coordinates.  
22 |     */  
23 |     /**  
24 |     * endY stores the final value of the range of y-coordinates.  
25 |     */  
26 |     /**  
27 |     * rangeX stores the distance between two x-coordinates.  
28 |     */  
29 |     /**  
30 |     * rangeY stores the distance between two y-coordinates.  
31 |     */  
32 |     /**  
33 |     * dist stores the distance between a point and the origin at any given time in the program.  
34 |     */  
35 |     /**  
36 |     * xCalc is a variable used to update the value of the x-coordinate in the getEscapeTime method.  
37 |     */  
38 |     /**  
39 |     * yCalc is a variable used to update the value of the y-coordinate in the getEscapeTime method.  
40 |     */  
41 |     public double startX, endX, startY, endY, rangeX, rangeY, xCalc, yCalc, dist;  
42 |     /**  
43 |     * passes is a counter that increases by one every time the loop is entered in the getEscapeTime method;  
44 |     */  
45 |     /**  
46 |     * escapeTime is the final value of passes after it exits the loop.  
47 |     */  
48 |     protected int passes, escapeTime;  
49 |     /**  
50 |     * fractals store the values of the escape times of all the points in the fractal.  
51 |     */  
52 |     protected int[][] fractals;  
53 |  
54 |     /**  
55 |     * Initializes fractals to a new 512 by 512 2-d array of type int.  
56 |     */  
57 |     public Fractals() {  
58 |         fractals = new int[512][512];  
59 |     }  
60 |  
61 |  
62 |     /**  
63 |     * Calculates the distance between the point and the origin using the  
64 |     * Pythagorean theorem.  
65 |     *  
66 |     * @param x The point's x-coordinate  
67 |     * @param y The point's y-coordinate  
68 |     * @return the distance between the point and the origin  
69 |     */  
70 |     public double distanceFromOrigin(double x, double y) {  
71 |  
72 |         return Math.sqrt(Math.pow(x - 0, 2) + Math.pow(y - 0, 2));  
73 |     }  
74 |  
75 |  
76 |     /**  
77 |     * Calculates the distance between any two equally spaced x-coordinates and y-coordinates using the formula  
78 |     * for arithmetic sequence. common difference =(last-first)/(n-1)  
79 |     */  
80 |     public void setRanges() {  
81 |  
82 |         rangeX = (endX - startX) / 511;  
83 |         rangeY = (endY - startY) / 511;  
84 |     }  
85 |  
86 |     /**  
87 |     * Calculates the number of times a point goes through a loop before it's  
88 |     * distance from the origin exceeds the escape distance or if it never  
89 |     * exceeds the escape distance.  
90 |     *  
91 |     * The while loop checks if the the dist is<=2.0 or if the passes<255  
92 |     *  
93 |     * @param x The point's x-coordinate  
94 |     * @param y The point's y-coordinate  
95 |     * @return The number of times the point enters a loop before it escapes  
96 |     */  
97 |  
98 |  
99 | }
```

```
114 |     }
115 |     return fractals;
116 | }
117 |
118 | /**
119 |  * Translates the row to corresponding x-coordinate.
120 |  * If we treat it as an arithmetic sequence with common difference of rangeX,
121 |  * then the nth term will be startX + (n-1)*rangeX.
122 |  * But the index in our 2-d array starts from 0 so we disregard the (n-1) and just use n.
123 |  *
124 |  * @param i The row of the 2-d array
125 |  * @return The translated x-coordinate of the corresponding row.
126 |  */
127 | public double getRangeValueX(int i) {
128 |     //type cast i to double
129 |     double a = (double) i;
130 |     return startX + (a * rangeX);
131 | }
132 |
133 |
134 |
135 |
136 | /**
137 |  * Translates the column to the corresponding y-coordinate.
138 |  *
139 |  * If we treat it as an arithmetic sequence with common difference of rangeY,
140 |  * then the nth term will be startY + (n-1)*rangeY.
141 |  * But the index in our 2-d array starts from 0 so we disregard the (n-1) and just use n.
142 |  *
143 |  * @param j The column of the 2-d array
144 |  * @return The translated y-coordinate of the corresponding column.
145 |  */
146 | public double getRangeValueY(int j) {
147 |     //type cast j to double
148 |     double b = (double) j;
149 |     return startY + (b * rangeY);
150 | }
151 | }
```

## src/fractals/FractalTestsAbstractClass.java

```
1 | package fractals;
2 |
3 |
4 | import java.util.Random;
5 |
6 |
7 | /**
8 |  * Main class that has method headers for all the fractal set tests.
9 |  * @author Supratik Neupane
10 |  * @author Sean Mackay
11 |  * @author Ryan Gangwish
12 |  * @author Aiden Cox
13 |  *
14 |  */
15 | public abstract class FractalTestsAbstractClass {
16 |     /**
17 |      * Random used in testing translation of x and y coordinates.
18 |      */
19 |     protected Random r;
20 |
21 |     /**
22 |      * Initializes random
23 |      */
24 |     public FractalTestsAbstractClass() {
25 |         r = new Random();
26 |     }
27 |
28 |     /**
29 |      * Test if the point has an escapeTime of 255 i.e. the maximum number of passes into the loop
30 |      */
31 |     public abstract void neverEscapeTest();
32 |
33 |     /**
34 |      * Test if the method to create the 2-d array has 512 rows and 512 columns
35 |      */
36 |     public abstract void arraySizeTest();
37 |
38 |     /**
39 |      * Test if the row is translated to the corresponding x-coordinate
40 |      */
41 |     public abstract void xCoordinateTest();
42 |
43 |     /**
44 |      * Test if the row is translated to the corresponding y-coordinate
45 |      */
46 |     public abstract void yCoordinateTest();
47 |
48 |
49 |
50 | }
```

## src/fractals/Julia.java

```
1 | package fractals;
2 |
3 |
4 | /**
5 |  * The Julia class is apart of the fractals package
6 |  * Julia uses an equation to return the escape time
7 |  * for a specific coordinate
8 |  * @author Ryan, Aiden, Supratik & Sean
9 |  * @see Fractals.java file
10 |  */
11 |
12 | public class Julia extends Fractals {
13 |
14 |     /**
15 |      * Calls the constructor in the super class, assigns the staring and ending x and y
16 |      * coordinates and calculates the ranges
17 |      */
18 |
19 |
20 |     public Julia() {
21 |         super();
22 |         startX = -1.7;
23 |         endX = 1.7;
24 |         startY = -1.0;
25 |         endY = 1.0;
26 |         setRanges();
27 |     }
28 |
29 |     /** (non-Javadoc)
30 |      * @see Fractals.Fractals#getEscapeTime(double, double)
31 |      */
32 |     @Override
33 |     public int getEscapeTime(double x, double y) {
34 |         passes = 0;
35 |         double k = -0.72689;
36 |         double i = 0.188887;
37 |         xCalc = x;
38 |         yCalc = y;
39 |         dist = distanceFromOrigin(xCalc, yCalc);
40 |
41 |         while (dist <= 2.0 && passes < 255) {
42 | }
```

## src/fractals/Mandelbrot.java

```
1 package fractals;
2
3 /**
4  * The Mandelbrot class is apart of the
5  * fractals package Mandelbrot uses an equation to return the escape time
6  * for a specific coordinate
7  *
8  * @author superkid Ryan, Aiden, Supratik & Sean
9  * @see Fractals.java file
10 */
11 public class Mandelbrot extends Fractals {
12
13
14
15     /**
16      * Calls the constructor in the super class, assigns the staring and ending x and y
17      * coordinates and calculates the ranges
18      */
19     public Mandelbrot() {
20         super();
21         startX = -2.15;
22         startY = -1.3;
23         endX = 0.6;
24         endY = 1.3;
25         setRanges();
26     }
27
28
29     /* (non-Javadoc)
30      * @see Fractals.Fractals#getEscapeTime(double, double)
31      */
32     @Override
33     public int getEscapeTime(double x, double y) {
34         passes = 0;
35         xCalc = x;
36         yCalc = y;
37         dist = distanceFromOrigin(xCalc, yCalc);
38
39         while (dist <= 2.0 && passes < 255) {
40             // tempX stores the value of xCalc before it's updated and used to update yCalc.
41             double tempX = new Double(xCalc);
42             xCalc = xCalc * xCalc - yCalc * yCalc + x;
43             yCalc = (2 * tempX * yCalc) + y;
44
45             dist = distanceFromOrigin(xCalc, yCalc);
46             passes++;
47         }
48
49         escapeTime = passes;
50         return escapeTime;
51     }
52 }
53
54 }
```

## src/fractals/Multibrot.java

```
1 package fractals;
2
3 /**
4  * The Multibrot class is apart of the fractals package
5  * Multibrot uses an equation to return the escape time
6  * for a specific coordinate
7  *
8  * @author Ryan, Aiden, Supratik & Sean
9  * @see Fractals.java file
10 */
11 public class Multibrot extends Fractals {
12
13     /**
14      * Calls the constructor in the super class, assigns the staring and ending x and y
15      * coordinates and calculates the ranges
16      */
17     public Multibrot() {
18         super();
19         startX = -1.0;
20         endX = 1.0;
21         startY = -1.3;
22         endY = 1.3;
23         setRanges();
24     }
25
26     /* (non-Javadoc)
27      * @see Fractals.Fractals#getEscapeTime(double, double)
28      */
29     @Override
30     public int getEscapeTime(double x, double y) {
31         passes = 0;
32         double xCalc = x;
33         double yCalc = y;
34         double dist = distanceFromOrigin(xCalc, yCalc);
35
36         while (dist <= 2.0 && passes < 255) {
37             // tempX stores the value of xCalc before it's updated and used to update yCalc.
38             double tempX = new Double(xCalc);
39             xCalc = Math.pow(xCalc, 3) - (3*xCalc*yCalc*yCalc)*x;
40             yCalc = (3 * tempX * tempX * yCalc) -Math.pow(yCalc, 3)+ y;
41             dist = distanceFromOrigin(xCalc, yCalc);
42             passes++;
43         }
44         escapeTime = passes;
45         return escapeTime;
46     }
47 }
```

## src/fractalsTests/BurningShipTest.java

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

```
40 |         assertEquals(512, b.getFractals().length);
41 |         assertEquals(512, b.getFractals()[511].length);
42 |     }
43 |
44 |
45 |
46 |     /* (non-Javadoc)
47 |     * @see fractalsTests.FractalTests#xCoordinateTest()
48 |     */
49 |     @Override
50 |     @Test
51 |     public void xCoordinateTest() {
52 |         BurningShip b = new BurningShip();
53 |         int z = r.nextInt(512);
54 |         double x = (z * b.rangeX) + b.startX;
55 |         assertEquals(x, b.getRangeValueX(z), 0.01);
56 |     }
57 |
58 |
59 |
60 |     /* (non-Javadoc)
61 |     * @see fractalsTests.FractalTests#yCoordinateTest()
62 |     */
63 |     @Override
64 |     @Test
65 |     public void yCoordinateTest() {
66 |         BurningShip b = new BurningShip();
67 |         int z = r.nextInt(512);
68 |         double y = (z * b.rangeY) + b.startY;
69 |         assertEquals(y, b.getRangeValueY(z), 0.01);
70 |         assertEquals(b.endY, b.getRangeValueY(511), 0.001);
71 |     }
72 |
73 |     /**makes sure no fractals escape time is equal
74 |     * to 0 or 1
75 |     */
76 |     @Test
77 |     public void noOneOrZeroEscapeTimeTest() {
78 |         BurningShip b = new BurningShip();
79 |
80 |         for (int i[] : b.getFractals()) {
81 |             for (int j : i) {
82 |                 assertFalse(j==0);
83 |                 assertFalse(j==1);
84 |             }
85 |         }
86 |     }
87 |
88 |
89 |
90 |
91 | }
```

## src/fractalsTests/JuiaTest.java

```
1 | package fractalsTests;
2 |
3 | /**
4 |  * 'JuiaTest' is a JUnit test that evaluates
5 |  * the Julia file
6 |  * 'FractalTests' is the superclass for this JUnit.
7 |  * @author Ryan, Aiden, Supratik & Sean
8 |  * @see fractals.FractalTestsAbstractClass.java file
9 |  */
10 |
11 | import static org.junit.Assert.*;
12 | import fractals.FractalTestsAbstractClass;
13 | import org.junit.Test;
14 |
15 | import fractals.Julia;
16 |
17 | public class JuiaTest extends FractalTestsAbstractClass {
18 |
19 |
20 |     /* (non-Javadoc)
21 |     * @see fractalsTests.FractalTests#neverEscapeTest()
22 |     */
23 |     @Override
24 |     @Test
25 |     public void neverEscapeTest() {
26 |         Julia j = new Julia();
27 |         assertEquals(255, j.getEscapeTime(1.0492187499999897, -0.234375));
28 |     }
29 |
30 |
31 |     /**
32 |     * Checks the escape time for a specific X and Y Value after one full loop
33 |     * of the Julia class values.
34 |     * checking to see that the escape time is equal to 1.
35 |     */
36 |     @Test
37 |     public void escapesAfterOneLoopTest() {
38 |         Julia j = new Julia();
39 |         assertEquals(1, j.getEscapeTime(1.6933593749999853, 0.9765625));
40 |     }
41 |
42 |
43 |
44 |     /* (non-Javadoc)
45 |     * @see fractalsTests.FractalTests#arraySizeTest()
46 |     */
47 |     @Override
48 |     @Test
49 |     public void arraySizeTest() {
50 |
51 |         Julia j = new Julia();
52 |         assertEquals(512, j.getFractals().length);
53 |         assertEquals(512, j.getFractals()[511].length);
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.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 | }
```

```
9 |  * 'MandelbrotTest' is a JUnit test that evaluates
10 |  * the Mandelbrot file
11 |  * 'FractalTests' is the superclass for this JUnit test.
12 |  * @see fractals.FractalTestsAbstractClass.java file
13 |  */
14 | public class MandelbrotTest extends FractalTestsAbstractClass {
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 |      * Test if the given point has the escape time of 1
28 |      */
29 |     @Test
30 |     public void escapesAfterOneLoopTest() {
31 |         Mandelbrot m = new Mandelbrot();
32 |         assertEquals(1, m.getEscapeTime(0.5946289062500001, 1.2949218750000122));
33 |     }
34 |
35 |     /* (non-Javadoc)
36 |      * @see fractalsTests.FractalTests#arraySizeTest()
37 |      */
38 |     @Override
39 |     @Test
40 |     public void arraySizeTest() {
41 |
42 |         Mandelbrot m = new Mandelbrot();
43 |         assertEquals(512, m.getFractals().length);
44 |         assertEquals(512, m.getFractals()[511].length);
45 |     }
46 |
47 |     /* (non-Javadoc)
48 |      * @see fractalsTests.FractalTests#xCoordinateTest()
49 |      */
50 |     @Override
51 |     @Test
52 |     public void xCoordinateTest() {
53 |         Mandelbrot m = new Mandelbrot();
54 |         int z = r.nextInt(512);
55 |         double x = (z * m.rangeX) + m.startX;
56 |         assertEquals(x, m.getRangeValueX(z), 0.01);
57 |     }
58 |
59 |     /* (non-Javadoc)
60 |      * @see fractalsTests.FractalTests#yCoordinateTest()
61 |      */
62 |     @Override
63 |     @Test
64 |     public void yCoordinateTest() {
65 |         Mandelbrot m = new Mandelbrot();
66 |         int z = r.nextInt(512);
67 |         double y = (z * m.rangeY) + m.startY;
68 |         assertEquals(y, m.getRangeValueY(z), 0.01);
69 |     }
70 |
71 | }
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 Synn, Alden, Supratik & Sean
8 |  * @see fractals.FractalTestsAbstractClass.java file
9 |  */
10 |
11 | import static org.junit.Assert.*;
12 | import org.junit.Test;
13 | import fractals.FractalTestsAbstractClass;
14 | import fractals.Multibrot;
15 |
16 | public class MultibrotTest extends FractalTestsAbstractClass{
17 |
18 |     /* (non-Javadoc)
19 |      * @see fractalsTests.FractalTests#neverEscapeTest()
20 |      */
21 |     @Override
22 |     @Test
23 |     public void neverEscapeTest() {
24 |         Multibrot m = new Multibrot();
25 |         assertEquals(255, m.getEscapeTime(0.5859375, 0.24375000000000108));
26 |     }
27 |
28 |     /**
29 |      * escapeAfterOneLoopTest runs getEscapeTime of Multibrot and confirms that
30 |      * a point escapes after one run through the algorithm
31 |      */
32 |     @Test
33 |     public void escapeAfterOneLoopTest() {
34 |         Multibrot m = new Multibrot();
35 |         assertEquals(1, m.getEscapeTime(0.9921875, 1.05625));
36 |     }
37 |
38 |     /* (non-Javadoc)
39 |      * @see fractalsTests.FractalTests#arraySizeTest()
40 |      */
41 |     @Override
42 |     @Test
43 |     public void arraySizeTest() {
44 |         Multibrot m = new Multibrot();
45 |         assertEquals(512, m.getFractals().length);
46 |         assertEquals(512, m.getFractals()[0].length);
47 |     }
48 |
49 |     /* (non-Javadoc)
50 |      * @see fractalsTests.FractalTests#xCoordinateTest()
51 |      */
52 |     @Override
53 |     @Test
54 |     public void xCoordinateTest() {
55 |         Multibrot m = new Multibrot();
56 |         int z = r.nextInt(512);
57 |         double x = (z * m.rangeX) + m.startX;
58 |         assertEquals(x, m.getRangeValueX(z), 0.01);
59 |         assertEquals(m.startX, m.getRangeValueX(0), 0.01);
60 |         assertEquals(m.endX, m.getRangeValueX(511), 0.01);
61 |     }
62 |
63 |     /* (non-Javadoc)
64 |      * @see fractalsTests.FractalTests#yCoordinateTest()
65 |      */
66 | }
```

[< Back to Summary](#)