

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 #9
Name	Aiden Cox (aidencox)
Partners	Sean Mackay (snmackay) Ryan Gangwish (ryangang) Supratik Neupane (sneupane) Aiden Cox (aidencox)
Submitted	02/25/17 01:26PM, 10 hrs, 32 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/JuliaTest.java	0	0.0 100.0%
src/fractalsTests/MandelbrotTest.java	0	0.0 100.0%
src/fractalsTests/MultibrotTest.java	0	0.0 100.0%
Submit Results.pdf	0	0.0
Submit Results2.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	<code>package edu.buffalo.cse116;</code>
2	
3	<code>public class Main {</code>
4	
5	<code>public static void main(String[] args) {</code>
6	
7	}
8	
9	<code>}</code>

src/fractals/BurningShip.java

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

```
111     for (int j = 0; j < 512; j++) {
112         fractals[i][j] = getEscapeTime(getRangeValueX(i), getRangeValueY(j));
113     }
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  * Translates the column to the corresponding y-coordinate.
136  *
137  * If we treat it as an arithmetic sequence with common difference of rangeY,
138  * then the nth term will be startY + (n-1)*rangeY.
139  * But the index in our 2-d array starts from 0 so we disregard the (n-1) and just use n.
140  *
141  * @param j The column of the 2-d array
142  * @return The translated y-coordinate of the corresponding column.
143  */
144 public double getRangeValueY(int j) {
145     //type cast j to double
146     double b = (double) j;
147     return startY + (b * rangeY);
148 }
149 }
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      * Initialises 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
13 public class Julia extends Fractals {
14
15     /**
16      * Calls the constructor in the super class, assigns the starting and ending x and y
17      * coordinates and calculates the ranges
18      */
19
20
21     public Julia() {
22         super();
23         startX = -1.7;
24         endX = 1.7;
25         startY = -1.0;
26         endY = 1.0;
27         setRanges();
28     }
29
30     /** (non-Javadoc)
31      * @see Fractals.Fractals#getEscapeTime(double, double)
32      */
33     @Override
34     public int getEscapeTime(double x, double y) {
35         passes = 0;
36         double k = -0.72689;
37         double l = 0.188887;
38         xCalc = x;
39         yCalc = y;
```

| 54 |

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

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 Fractals.FractalTestsAbstractClass.java file
9  */
10
11 import static org.junit.Assert.*;
12 import fractals.FractalTestsAbstractClass;
13 import org.junit.Test;
14 import fractals.BurningShip;
15
16 public class BurningShipTest extends FractalTestsAbstractClass {
17
18
19     /* (non-Javadoc)
20      * @see FractalsTests.FractalTests#neverEscapeTest()
21      */
22 }
```

```

37 | public void arraySizeTest() {
38 |
39 |     BurningShip b = new BurningShip();
40 |     assertEquals(512, b.getFractals().length);
41 |     assertEquals(512, b.getFractals()[511].length);
42 |
43 | }
44 |
45 | /* (non-Javadoc)
46 |  * @see fractalsTests.FractalTests#xCoordinateTest()
47 |  */
48 | @Override
49 | @Test
50 | public void xCoordinateTest() {
51 |     BurningShip b = new BurningShip();
52 |     int z = r.nextInt(512);
53 |     double x = (z * b.rangeX) + b.startX;
54 |     assertEquals(x, b.getRangeValueX(z), 0.01);
55 |
56 | }
57 |
58 |
59 | /* (non-Javadoc)
60 |  * @see fractalsTests.FractalTests#yCoordinateTest()
61 |  */
62 | @Override
63 | @Test
64 | public void yCoordinateTest() {
65 |     BurningShip b = new BurningShip();
66 |     int z = r.nextInt(512);
67 |     double y = (z * b.rangeY) + b.startY;
68 |     assertEquals(y, b.getRangeValueY(z), 0.01);
69 |     assertEquals(b.endY, b.getRangeValueY(511), 0.001);
70 |
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 |             assertEquals(j==0);
83 |             assertEquals(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 evalustes
5 |  * the Juia 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 |     /* (non-Javadoc)
44 |     * @see fractalsTests.FractalTests#arraySizeTest()
45 |     */
46 |     @Override
47 |     @Test
48 |     public void arraySizeTest() {
49 |
50 |         Julia j = new Julia();
51 |         assertEquals(512, j.getFractals().length);
52 |         assertEquals(512, j.getFractals()[511].length);
53 |
54 |     }
55 |
56 |     /* (non-Javadoc)
57 |     * @see fractalsTests.FractalTests#xCoordinateTest()
58 |     */
59 |     @Override
60 |     @Test
61 |     public void xCoordinateTest() {
62 |         Julia j = new Julia();
63 |         int z = r.nextInt(512);
64 |         double x = (z * j.rangeX) + j.startX;
65 |         assertEquals(x, j.getRangeValueX(z), 0.01);
66 |         assertEquals(j.endX, j.getRangeValueX(511), 0.01);
67 |
68 |     }
69 |
70 |     /* (non-Javadoc)
71 |     * @see fractalsTests.FractalTests#yCoordinateTest()
72 |     */
73 |     @Override
74 |     @Test
75 |     public void yCoordinateTest() {
76 |         Julia j = new Julia();
77 |         int z = r.nextInt(512);
78 |         double y = (z * j.rangeY) + j.startY;
79 |         assertEquals(y, j.getRangeValueY(z), 0.01);
80 |
81 |     }
82 |

```

```
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 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 |     /**
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 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 |
17 | public class MultibrotTest extends FractalTestsAbstractClass{
18 |
19 |
20 |     /* (non-Javadoc)
21 |     * @see fractalsTests.FractalTests#neverEscapeTest()
22 |     */
23 |     @Test
24 |     @Override
25 |     public void neverEscapeTest() {
26 |         Multibrot m = new Multibrot();
27 |         assertEquals(255, m.getEscapeTime(0.5859375, 0.24375000000000108));
28 |     }
29 |
30 |
31 |     /**
32 |     * escapeAfterOneLoopTest runs getEscapeTime of Multibrot and confirms that
33 |     * a point escapes after one run through the algorithm
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()[511].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 |         assertEquals(m.startX,m.getRangeValueX(0),0.01);
66 |         assertEquals(m.endX,m.getRangeValueX(511),0.01);
67 |     }
68 | }
```

82		
83		
84		1

[< Back to Summary](#)