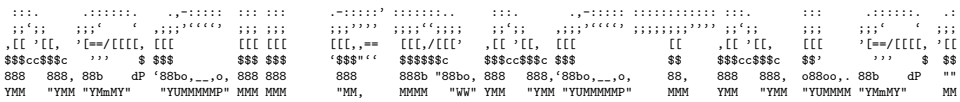
ASSIGNMENT 3



COMP-202, Summer 2010

Due: Thursday, May 27th, 2010

1 Introduction

In this assignment we will draw pretty pictures. But instead of drawing them in pixels, we will do so using characters on the terminal. Our object is going to be the mandelbrot set, which is a fractal. A fractal is a complicated geometric figure, whose boundary is not smooth - and when you zoom into the boundary, you will see structures that are similar to the whole fractal itself (this is called 'self similar'). The fractal is generated from a formula, given a set of coordinates in the 2d plane (i.e. an x- and y-coordinate), which produces a new set of coordinates:

$$x_{new} = x_{old} \cdot x_{old} - y_{old} \cdot y_{old} + x_{original}$$
$$y_{new} = 2 \cdot x_{old} \cdot y_{old} + y_{original}$$

Note how in order to compute the above, we needed the original coordinates that we started with.

When we repeatedly apply the formula to the coordinate, there are two possible things that can happen:

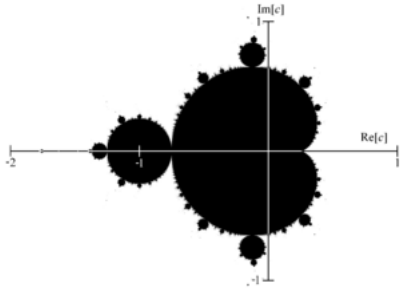
- Eventually we reach a coordinate whose absolute value is larger than 2. After that, when applying the formula, the new coordinates will always have a larger absolute value ($\sqrt{(}x^2 + y^2)$) than before and we will 'go to infinity'. The original coordinate is then considered not to be in the mandelbrot set
- It doesn't matter how often we repeat the formula, the absolute value of all coordinates will always stay smaller than 2. The original coordinate is then considered part of the mandelbrot set.

The above formula generates the Mandelbrot set. If we want to generate different fractals, we can use different formulas. But the idea of repeatedly applying a function and checking for divergence is the same.

Since we can not compute indefinitely, we will use an approximation. We will set a maximum number of times we will apply the formula, i.e. a maximum number of iterations. If after that the produced coordinate has an absolute value smaller than 2, then we will consider it part of the mandelbrot set. Of course the moment we reach a coordinate whose absolute value is larger than 2, we don't need to continue computing.

Our program will print out an area of the 2d plane, where it will find out for every displayed coordinate whether it is part of the Mandelbrot set. If it is, we will represent the coordinate with a `' '`. If it's not, we will display a character based on how many iterations of the formula it took to receive a coordinate whose absolute values is larger than 2.

Please follow the below specification to write your program. Refer to assignment 2 for Do's and Don'ts.



2 Specification

2.1 `getIterationCount`

Create a class `Fractal` with a constant global `MAX_ITERATIONS` set to 1000. Add a method `getIterationCount` taking two doubles as parameters, an x- and y-coordinate. The method should return (as an integer) how many iterations it took before either reaching a coordinate whose absolute value is larger than 2, or `MAX_ITERATIONS` if that doesn't happen within that many iterations. Hints:

- Be careful not to overwrite the old x coordinate with the new x coordinate before you have computed the new y coordinate.
- Note that $\sqrt{v^2} < 2$ is equivalent to $v \cdot v < 4$.
- Always test a new method you add using a main method.
- example values:

x	0.0	100.0	0.3	1.0	0.001643	0.0016437	0.00164372
y	0.0	0.0	1.0	0.3	0.82246	0.822467	0.8224673
iterations	1000	0	2	1	58	67	91

2.2 `iterationCountToChar`

Add a method, that given an integer iteration count, will return a character to represent it. If the given value is equal to `MAX_ITERATIONS` it should return `' '`. Otherwise it should return the following, based on the last digit of $(\text{int})\frac{i}{ln(i)+1}$ (where i is the iteration count):

digit	0	1	2	3	4	5	6	7	8	9
return	.	,	:	;	?	&	8	S	H	@

2.3 `computeFractal`

Add a `String` method `computeFractal`, taking four doubles. It should return a string representing the fractal in the given area of the 2d-plane. The area is given by the parameters in the following order: the x-coordinate of the center, the width of the area (dx), the y-coordinate of the center and the height of the area (dy). Add two integer constants called `ROWS` and `COLUMNS`, initialized to 40 and 120. Note that every character in the String will represent a 2d-coordinate. The string should have `ROWS` · `COLUMNS` letters, and `ROWS` newlines. The first character should be the representation of the top left corner of the area, the last two should be a character denoting the bottom right corner of the area and a new line. The method should work regardless of what the constants are (So you could test the method with something like 4 and 6 to check for off-by-one errors).

2.4 `printVideo`

Add a void method `printVideo` that starts at the coordinates $(0.001643721971153, 0.822467633298876)$, zoomed out ($d_x = 6$,$d_y = 4$), and prints out a video zooming in. it should zoom in with a factor of 0.9 from one frame to the next. Note that we don't need to pause for every frame, because the computation takes some time, anyway. Zoom until you see the last 'baby mandelbrot'. The method should exit the program after displaying the last frame.

2.5 `main`

Add a main method. It should repeatedly print the fractal, then prompt the user for an action with

```
what next? (i)up, (k)down, (j)left, (l)right, (e)zoom in, (d)zoom out, (v)ideo+quit, (q)uit:
```

until q is entered (or the program exits because of the video). The actions are specified by the characters, other characters get ignored. We start in the area $x_{center} = -.5, d_x = 6$, $y_{center} = 0$, $d_y = 4$. We move by adding or subtracting $0.3 \cdot d_x$ or $0.3 \cdot d_y$ (why can't we 'move' by a constant amount, but have to move relative to the width/height of the visible area?). We zoom in by multiplying d_x and d_y by 0.6, and zoom out by dividing by that same number.