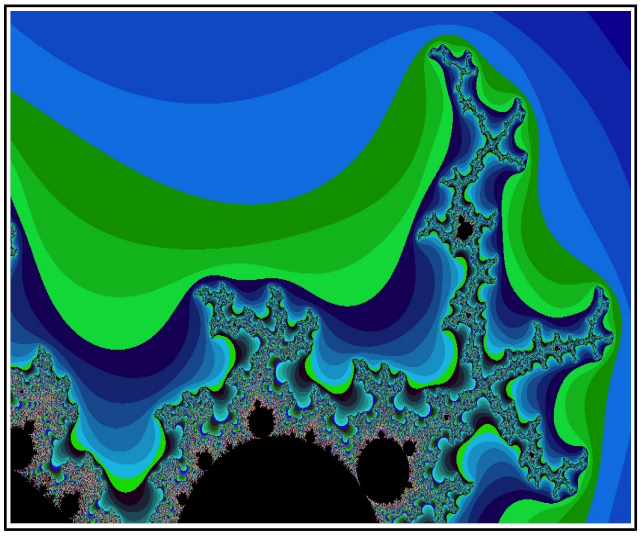
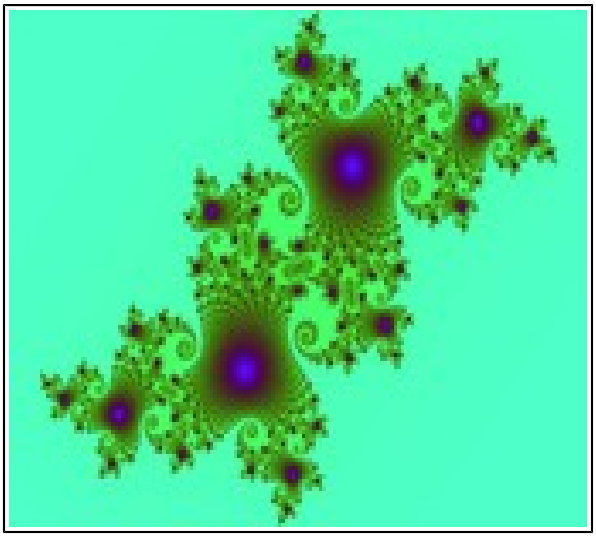
CO225: Software Construction
Project 1: Fractals (Individual project)

Aim: The aim of this project is to build a somewhat complex piece of software using the techniques that we have discussed in class. In particular inheritance. Make sure to put sufficient comments to your code because you will be using this code again later (for a different project).

You will need more than one class to implement this and look at the sample code provided and documentation links. **Do a proper design before coding.** You are also required to submit a digram depicting how classes are derived so do this first.



Mandelbrot set



Julia Set

Introduction: You are required to design and implement a software that would plot two of the most popular *fractals*: Mandelbrot set and Julia set.

Fractals are infinity many, self-similar shapes formed by some simple mathematical computations. The computation varies from fractal set to set but are generally based on complex numbers.

The Mandelbrot set: In mathematics Mandelbrot set is defined as the set of complex numbers C such that: $\mathbf{Z_{n+1} = Z_n^2 + C}$, starting with $Z_0 = 0$ remains bounded when *n* reach infinity. In other words if for some C the above equation remains bounded for after many iterations then that C is in the Mandelbrot set. There is a mathematical proof which shows that if $ABS(Z_n) > 2$ the above equation will diverge hence C is not in the Mandelbrot set. Some complex numbers will be in the Mandelbrot set and some are not.

To plot the set, in the basic case one can use two color (say black and white) to mark the points corresponding to the complex numbers that are and are not in the Mandelbrot set.

In any case all Mandelbrot numbers are within a region of the complex plane; -1 < real part < 1 and -1< complex part<1. We call this the **region of interest** which you should be able to set.

You would want to make a plot in a canvas of a given size (say 800x800). A point on the canvas first

needs to be mapped onto a point within the region of interest. Once that is done use that value as C and perform the above computation $\mathbf{Z_{n+1} = Z_n^2 + C}$. Perform 1000 iterations and then see if $ABS(Z_n) > 2$ for any n < 1000. If so C is not a Mandelbrot number so assign some colour to it based on value of n when $ABS(Z_n) > 2$. Else assign black. Repeat this process for all points on the canvas.
You should be able to set the iterations and the region of interest from the command-line by passing arguments; java Fractal ***Mandelbrot -0.5 0.5 -0.1 1 1000*** means the region of interest for the image should be from -0.5<real<0.5 and -0.1<complex<1 and for each point you need to do 1000 iterations before deciding that it is in the set. Note that *Fractal* is the name of the application.

The Julia set: The Julia set is similar to the Mandelbrot set in that it uses the same equation $\mathbf{Z_{n+1} = Z_n^2 + C}$ but Z_0 is the point in the complex plane corresponding to the pixel and C is a constant. The rest of the computation is the same; including the way you map a point in the canvas to that in the complex plane. If one types; java Fractal ***Julia -0.5 0.156 1000*** then you should plot the Julia set for C = -0.5 + 0.156i with 1000 iterations for each point. You may take the region of interest in the complex plan as 1 < real part < 1 and -1< complex part<1 which cannot be modified.

Generating the image: We have a area (with some dimension) to plot the pattern. This area will contain pixels and will be fixed for 800x800 pixels. For each point you need to see if the point is in the set specified by the user and assign colour accordingly.

Program: Your program should accept arguments; at least one which would specify what set to plot. If the set selected is Mandelbrot the user should give either 0, 4 or 5 arguments. If there are 0 arguments then use the default values as specified in the table below. 4 arguments will be the region of interest in the complex plane and the 5th one is the number of iterations to do for a point.

For the Julia set the user should give 0 or 2 arguments. If there are no arguments one should use the default arguments and 2 arguments will be the real and complex part for C.

Item	Default value	Note
Region of interest	-1<real<1 -1<complex<1	Always use default for Julia
Number of iterations	1000	
C	-0.4+0.6i	Only for Julia set

Programming: You are **not given** a skeleton code for this project. You may use fragments of code and/or classes from lectures/sample code. See how you can minimize the code by using Java inheritance and other techniques we discussed.

You are advised to do a design before coding. You need to submit a PDF of your class diagram with the code.

Attention: You might want to consider the following:

1. Precision of the complex numbers and the data type for the complex and real parts
2. Run time. Since there are lots of computation one might think of using techniques to reduce the computation time. For example instead of testing $ABS(Z_n) > 2$ one might use $ABS(Z^2_n) > 4$.

This make sense since the equation gives you $Z^2_{n.}$

3. As an advanced part one might consider using Java threads. This is not expect from you but you may try that for additional 10marks.

Submission: You should submit all code files and a PDF containing your class diagram as a **single zip** file via Moodle before the deadline. As always you will be given marks for coding logic, correctness and neatness.