CS5001 Object-Oriented Modelling, Design and Programming
Practical 3 – Graphics - Mandelbrot Set Explorer

School of Computer Science
University of St Andrews

Due 9pm Friday week 11, weighting 45%
MMS is the definitive source for deadlines and weightings

Objective

To gain experience in developing GUI-drive applications, To that end, in this assignment, you are going to develop a GUI-driven Fractal generator that allows the user to view and explore the Mandelbrot set graphically. # SetupAs usual, you may develop your code in any IDE or editor of your choice. Once you have created a suitable assignment directory (on the Linux Lab Machines) such as `~/Documents/CS5001-p3` in your network home space (your network home directory) on the Linux lab clients, you may wish to copy the starter code in the file

https://studres.cs.st-andrews.ac.uk/CS5001/Coursework/p3-graphics/code.zip

to your assignment directory. From a terminal window on the lab machines, you can do this using the following commands:

```
mkdir -p ~/Documents/CS5001-p3
cd ~/Documents/CS5001-p3
unzip /cs/studres/CS5001/Coursework/p3-graphics/code.zip
```

It contains a single public method `calcMandelbrotSet` which can compute the Mandelbrot set for given parameters as explained in the Javadoc and below. Feel free to use this, alter it or replace it with your own version.

Background

The Mandelbrot set is the set of complex numbers C for which iterative application of the equation

$Z_{n+1} = Z_n^2 + C$

with Z starting at 0 (i.e. at $0 + 0i$), remains bounded within a certain distance from the origin 0 in the complex plane. There are some resources on the Mandelbrot set at

https://en.wikipedia.org/wiki/Mandelbrot_set

http://mathworld.wolfram.com/MandelbrotSet.html

It is probably a good idea to have a look at these resources. However, you don't have to fully understand the underlying Mathematics and have been provided starter code that will compute the Mandelbrot set for you (see below).

Supplied Starter Code

The MandelbrotCalculator class in the `code.zip` on studres provides you with a `calcMandelbrotSet` method which permits you to calculate the Mandelbrot set for given input parameters. The class also defines a number of constants that may be of use as well, including default initial values for the minimum and maximum *real* and *imaginary* components for C (`INITIAL_MIN_REAL`, `INITIAL_MAX_REAL`, `INITIAL_MIN_IMAGINARY`, `INITIAL_MAX_IMAGINARY`), an initial cut-off limit after which boundedness may be assumed (`INITIAL_MAX_ITERATIONS`), and a default value for the squared distance from the origin in the complex plane (`DEFAULT_RADIUS_SQUARED`) which can be used to check whether Z is deemed to have remained *bounded* or not.

If you use the supplied code to generate the Mandelbrot set for an (X, Y) resolution of (20, 25), by calling

```
MandelbrotCalculator mandelCalc = new MandelbrotCalculator();
int[][] mandelbrotData = mandelCalc.calcMandelbrotSet(20, 25,
    INITIAL_MIN_REAL, INITIAL_MAX_REAL, INITIAL_MIN_IMAGINARY,
    INITIAL_MAX_IMAGINARY, INITIAL_MAX_ITERATIONS, DEFAULT_RADIUS_SQUARED);
```

you obtain a 2-D array `mandelbrotData[row][column]` with 25 rows and 20 columns which contains the iteration values shown on the left of Figure 1. below.

The top-left value of 2 (found in `mandelbrotData[0][0]`) signifies that the value of Z escaped its bounds after 2 iterations of applying the recurrence equation above with the value of the constant C set to $C = -2.0 - 1.25i$, or if you prefer thinking of C as an coordinate, with C set to (-2.0, -1.25).

As such, the top-left iteration value corresponds to setting C to the minimum default value for both real and imaginary parts (`INITIAL_MIN_REAL`, `INITIAL_MIN_IMAGINARY`).

Similarly, the top-right value of 3 (coming from `madelbrotData[0][19]`) signifies that the value of Z escaped its bounds after 3 iterations with C set to (0.7, -1.25), i.e. for C set to (`INITIAL_MAX_REAL`, `INITIAL_MIN_IMAGINARY`).

Values of 50 in the array (which is the value of `INITIAL_MAX_ITERATIONS`) indicate that the corresponding values of C are in the Mandelbrot set as Z remained bounded.

As such, the provided method `calcMandelbrotSet`, can be seen to calculate the Mandelbrot set and map the (real, imaginary) coordinates – for the initial default values, these range between top-left (-2.0, -1.25) and bottom-right (+0.7, +1.25) – onto the desired (X,Y) resolution. The method returns a 2-D integer array of values that can be used to represent colour. Executing the method with a higher X and Y resolution than 20 and 25 (but with the same default parameters as before), and then displaying (X, Y) coordinates with an iteration value >= 50 as black on

```
 2  2  2  2  3  3  3  3  4  4  4  4  4  4  4  3  3  3  3  3
 2  2  2  3  3  3  4  4  4  4  4  4  5  5  5  5  4  3  3  3
 2  2  2  3  3  4  4  4  4  4  4  5  5  7  7  5  5  4  3  3
 2  2  3  3  4  4  4  4  4  4  5  5  6  8 17  7  5  5  4  3
 2  2  3  4  4  4  4  4  4  5  5  6  7 48 50 17  6  5  4  4
 2  2  4  4  4  4  4  4  5  5  6  7  8 30 50 11  7  6  5  4
 2  3  4  4  4  4  4  5  5  6  8 14 40 50 50 45 19  9  6  4
 2  4  4  4  4  4  5  6  6  7  9 50 50 50 50 50 50 50  7  5
 2  4  4  4  4  6  7  6  7  8 50 50 50 50 50 50 50 50  8  5
 2  4  5  5  6  7 10 12  9  9 50 50 50 50 50 50 50 50 21  5
 2  5  5  5  6  8 25 50 50 13 50 50 50 50 50 50 50 50 12  5
 2  5  5  6  8 10 50 50 50 29 50 50 50 50 50 50 50 50  8  5
 2  7  8  8 11 50 50 50 50 50 50 50 50 50 50 50 50 50  7  5
 2  7  8  8 11 50 50 50 50 50 50 50 50 50 50 50 50 50  7  5
 2  5  5  6  8 10 50 50 50 29 50 50 50 50 50 50 50 50  8  5
 2  5  5  5  6  8 25 50 50 13 50 50 50 50 50 50 50 50 12  5
 2  4  5  5  6  7 10 12  9  9 50 50 50 50 50 50 50 50 21  5
 2  4  4  4  4  6  7  6  7  8 50 50 50 50 50 50 50 50  8  5
 2  4  4  4  4  4  5  6  6  7  9 50 50 50 50 50 50 50  7  5
 2  3  4  4  4  4  4  5  5  6  8 14 40 50 50 45 19  9  6  4
 2  2  4  4  4  4  4  4  5  5  6  7  8 30 50 11  7  6  5  4
 2  2  3  4  4  4  4  4  4  5  5  6  7 48 50 17  6  5  4  4
 2  2  3  3  4  4  4  4  4  4  5  5  6  8 17  7  5  5  4  3
 2  2  2  3  3  4  4  4  4  4  4  5  5  7  7  5  5  4  3  3
 2  2  2  3  3  3  4  4  4  4  4  4  5  5  5  5  4  3  3  3
```

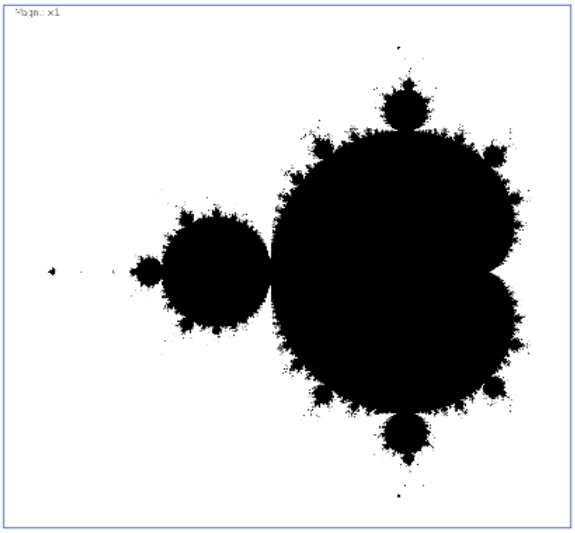

Figure 1: Figure 1. Textual (left) and Graphical (right) Representations of the Mandelbrot Set

a white background, yields the image on the right of Figure 1. Hopefully, you will see the resemblance between the left and right images in Figure 1.

You can zoom in, or rather re-calculate a close up of the Mandelbrot set, by calling the `calcMandelbrotSet` method with appropriate `minReal`, `minImaginary`, `maxReal`, `maxImaginary` values that lie between the initial default bounds outlined above (i.e. by increasing `minReal`, `minImaginary` and decreasing `maxReal`, `maxImaginary`).

Basic Requirements

You should write a GUI-driven Mandelbrot program which satisfies the following basic functional requirements.

- Basic display – display the Mandelbrot set in black and white for a sensible initial parameter setting.
- Zoom – permit the user to select a square (or rectangular area) **on the image** which will then be used as the bounds for a new calculation of the Mandelbrot set, thereby effectively zooming in on the image. Ideally, you should permit the user to select and view the zoom area super-imposed on top of the fractal image by dragging the mouse over the image.
- Pan – permit the user to select a distance and direction **on the image** which will then be used as the bounds for a new calculation of the Mandelbrot set, thereby effectively panning the image left/right/up/down by the given amount.
- Permit the user to choose to see an estimate of the zoom magnification super-imposed on the Mandelbrot image on screen.
- Permit the user to alter the value used for `maxIterations` so as to enhance the precision of the Mandelbrot image when zooming as they desire.
- Basic Undo, Redo – permit the user to undo / redo the last zoom or pan operation.

Enhancements

Possible extensions include:

- Different Colour Mappings – map the iteration values to different shades of a colour, such as shown for the colour blue and red in Figure 2. below, where higher iteration numbers are mapped to brighter/whiter shades of blue and red until the iteration limit is reached. Permit the user to switch between colour maps at the touch of a button.
- Full Undo, Redo, Reset – permit the user to zoom in on the image, pan around, change other parameter settings and subsequently undo and redo these operations or reset all parameter settings back to their defaults and display the corresponding image.
- Save and Load – permit parameter settings to be saved and loaded to/from file thereby permitting a saved image to be re-loaded and the user to continue exploring the Mandelbrot set from that position onward.
- Export – permit the computed image to be exported in a standard image format.
- Permit the user to define and view animations of zooming in on the Mandelbrot set.

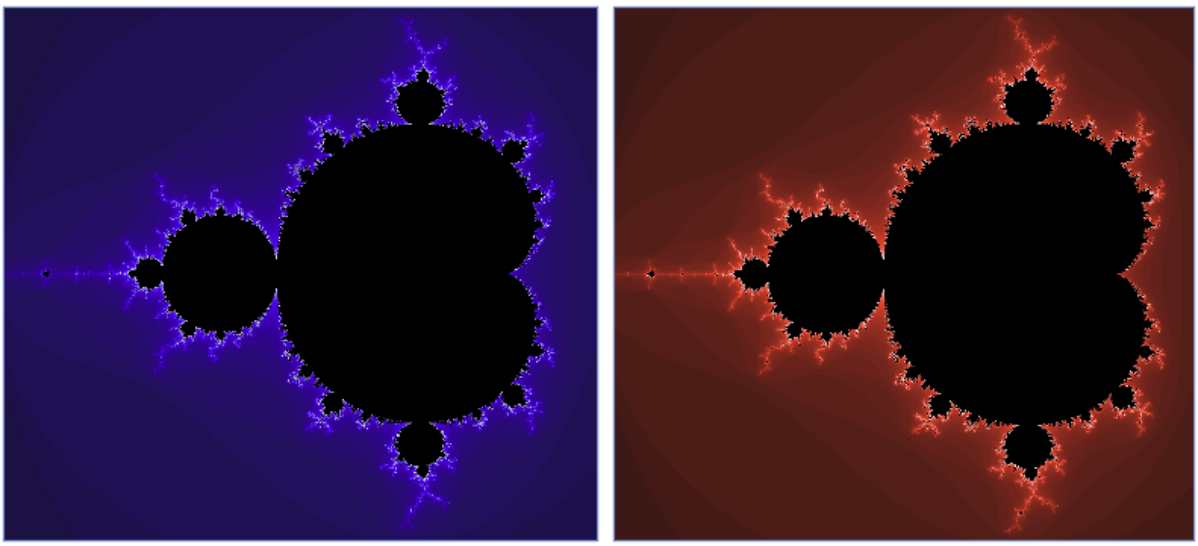

Figure 2: Figure 2. Blue (left) and Red (right) Colour Maps for Mandelbrot Set

Design / Implementation Considerations

For this practical you will have to think very carefully about the classes you will require for your program. The Model-Delegate (MD) design pattern is one possible program design to adopt. In any case, you should design a suitable set of classes to represent your Mandelbrot program and its various components, including Model and User-interface components, colour mappings, parameter settings etc.. You should think carefully about where to place the code which knows how to render the Mandelbrot set.

Adoption of MD or MVC patterns is recommended. You should aim to provide a Model component of your program comprising classes that model the calculation of the Mandelbrot set and the parameter settings which have been set based on user interaction. The Delegate should provide operations to permit a square or rectangular area (for zooming) or direction

and magnitude (for panning) to be selected and viewed and should be able to pass the relevant parameters (or coordinates) to the Model which can calculate the Mandelbrot set. The model would also provide methods to undo and redo various operations as well as methods to save and load the current settings and image to/from file, etc. depending on how many features you implement. The classes comprising the Delegate would translate button presses to the appropriate calls to methods defined in the Model. If you follow this design pattern, you would ideally link the Delegate to the Model by making the Delegate an Observer of the Model such that the drawing canvas can be redrawn when the model notifies its observers (loose coupling) by firing out PropertyChangeEvent objects.

When thinking about rendering the Mandelbrot set graphically, you may find it useful for a Swing GUI to consider that drawing pixels is much like drawing zero-length lines. For zooming, you may wish to think carefully how to map X,Y coordinates (of the zooming square/rectangle) and X,Y distances (for panning) to the lower and upper real and imaginary bounds for the Mandelbrot set computation. In order to display the zooming square or pan action smoothly while dragging, you may wish to investigate the use of Image objects such as `BufferedImage` and avoid a) re-computing the Mandelbrot set and b) re-applying the colour map while dragging the mouse over the image, however, you will probably have to re-draw parts or all of the image on your canvas in order to avoid smearing. When thinking about undo and redo, you may wish to consider suitable data structures (in `java.util`) which would permit you to easily restore the most recently used parameter settings.

Deliverables – Report and Software

For this practical, you should write a word-processed report. Your report should document your design decisions and the justification for those decisions. It should include a description of your implementation, how your system works, demonstrate and explain operation and testing of your system and any problems you had, and should be in your assignment folder as part of your submission. Hand in a .zip archive of your assignment directory (containing all your source code, any sub-directories, and PDF report) via MMS as usual.

Marking

A very good attempt at satisfying the basic requirements above in an object-oriented fashion with a very good design can achieve a mark of 14 - 16. This means you should produce very good, re-usable code which makes proper use of inheritance, association, and encapsulation where appropriate with very good method decomposition. To achieve a 17 or above, your submission should in addition make a very good attempt at one or more enhancements. See the standard mark descriptors in the School Student Handbook:

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness

The standard penalty for late submission applies (Scheme A: 1 mark per 24-hour period, or part thereof):

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

https://www.st-andrews.ac.uk/education/handbook/good-academic-practice/