CSC111 **Lab 5 – Visualizing the Mandelbrot Set** **Fall 2017**

In this lab we will compute a visualization of the Mandelbrot set. See the document "Lab 5 Supplement" for background material on complex numbers and a description of the Mandelbrot set.

Suggested Organization

Classes Provided for You:

In this lab, we will use several classes. Three classes are provided for you and can be downloaded from our web server menehune.opt.wfu.edu. These are:

Img.class – A simple class to visualize a 2-D array.

Img$1.class – A supporting class for `Img.class`

Ctab.class – A class to implement a color map for use with `Img.class`

Using the `Img` class is exceptionally easy. If A is a 2-D array containing non-negative integers, it can be displayed using two lines of code:

```
Img vis = new Img() ;   // Create a 'vis' object.
vis.display( A )    ;   // Call the 'display' method with 2-D array 'A'
```

Classes You Write:

Complex.java – A class to implement complex numbers. An object of class `Complex` represents one complex number. When implementing your class, avoid excessive memory use. Techniques relating to memory will be discussed in class.

Data: Use two `double` data members to represent the real and imaginary parts of a complex number. These values can be private if you prefer.

Methods:

- Write a constructor to initialize the current object to $0 + 0\imath$.
- Write a non-static method `add` to add two complex numbers. Header is:

  ```
  void add( Complex z1, Complex z2 )
  ```

  Store the result in the current object. This allows the caller to control the use of complex variables without constantly allocating new memory.
- Write a non-static method `mult` to multiply two complex numbers. Header is:

  ```
  void mult( Complex z1, Complex z2 )
  ```

  Again, store the result in the current object.
- Write a non-static method `abs`. This method should return the absolute value of the current Complex object. Header is:

  ```
  double abs( )
  ```

- Write a non-static method `set`. This method takes two double values and sets the real and imaginary parts of the current Complex object. Header is:

  ```
  void set( double a, double b )   // For  a + bi
  ```

- Write a non-static method `real`. This method returns the real part of the current object. Header is:

  ```
  double real( )
  ```

- Write a non-static method `imag`. This method returns the imaginary part of the current object. Header is:

  ```
  double imag( )
  ```

When you complete class `Complex` it is time for **unit testing**. Download a file named `test_complex.java` from menehune. Run it with your class Complex.

View.java – A class to hold the array of escape counts.

Data:

- A data member `N` is the size of the square 2-D array.
- A data member `A` is a square 2-D array of integers.
- Data members `a_min` and `b_min` represent the lower left corner of a corresponding square region in the complex plane.
- Data member `L` represents the length of one of the equal sides of the square region in the complex plane.

Methods:

- A constructor which accepts an input n and does the following:
  1. Sets `N = n`
  2. Allocates an `N x N` 2-D array `A`.
  3. Sets every entry in the array to zero.
  4. Sets the data members `a_min` and `b_min` to default values: -2.0 and -2.0 respectively. Sets `L` to the default value 4.0
- A method named `set_lower_left_corner`. Header is:

  ```
  void set_lower_left_corner( double a, double b )
  ```

- A method named `set_length` that sets the value of `L` to the given input.
- A method named `mapi` that maps a row index i to the real component of a point in the complex plane (as previously described).
- A method named `mapj` that maps a column index j to the imaginary component of a point in the complex plane (as previously described).
- A method named `set` that accepts a row-column position i, j in the array and a value v. It sets `A[i][j] = v`
- A method named `render` that creates an object of class `Img` and passes the array `A` to the `display` method.

main.java – A class to contain the main program and put all the parts together.

Data:

- An (constant) integer `ASIZE` to define the size of the array when creating a `View` object. The number of pixels in the displayed image will be `ASIZE*ASIZE`. Recommended value is `ASIZE = 896`

Methods:

- Write a static method `escape_time`. This method accepts a complex number c and returns the escape time (number of iterations) as defined by the Mandelbrot set. Header is:

  ```
  static int escape_time( Complex c )
  ```

- `public static void main( String [] args )`
  The main method will:
  1. Loop over all rows and columns in your 2-D array. Let i denote a row index and j denote a column index.
  2. Use your `View` object to map i and j to the numbers a and b. These define a complex number $c = a + b\imath$.
  3. Use your static method `escape_time` to compute the escape time for this point c in the complex plane.
  4. Use the method `set` in the `View` object to record the escape time.
  5. After completing the 2-D array, use the `render` method to display the Mandelbrot set.

Turn In;

Save all your work in a directory named "Lab5". Change to your home directory (the parent directory of "Lab5"), and create a file named "lab5.tar" using the command:

```
tar cf lab5.tar Lab5
```

Use `sftp` to upload the file "lab5.tar" to your account on telesto.