SENG2200 Assignment 1 Report

Designing the project was quick and simple. The initial design consisted of a main class (*PA1*) that serves as an interface; a *MyPolygons* class that stores a series of polygons as a circular doubly-linked list; a *Polygon* class that consists of a closed series of points; a *Point* class that contains a set of Cartesian coordinates; and, as per the specifications, a *ComparePoly* interface which contains a single abstract method, *ComesBefore().* Designing these classes, their functions, their variables and their relations to other classes, took about 30 minutes.

With a design in place, I set up the CLASSPATH and started to write the *PA1* class, just enough so that it could produce output. The next step was to create the *Point* class. Considering the class consists of 2 variables, 2 getters and the *ToString()* function, writing the entire class took around 15 minutes. Most of that time was spent revising the basics of the *String.format()* function, and there were zero errors.

After creating a few sample *Points* in the main class to make sure the *ToString()* function worked, I wrote the *Polygon* class. This class took much longer – my initial design had 3 variables (1 point array and 2 polygons) and 10 functions. I later added the *pointCount* integer after realising how much easier it made iterating through the list properly. As for errors, the *Area()* function caused a major problem – chiefly, the design specification gave a version of the area formula that was wrong. Even after fixing that problem, I forgot to calculate the absolute value of the formula, instead calculating the actual value.  
Along with a few minor bugs in *DistanceFromOrigin()* and *ComesBefore()*, I spent around 20 minutes writing the code and 2 hours bug-fixing.

I wrote more of the *PA1* class, this time reading a single polygon from a file and then outputting it properly to the console. There were a few formatting and syntax errors, but writing and debugging combined took about 10 minutes.

Next was the *MyPolygons* class – the problem child. The initial design consisted of 3 variables (2 Polygons as a sentinel and current node, and an integer count of total polygons), the functions specified in the assignment, a constructor and 2 getters. I created a nigh-unfathomable amount of bugs by trying to write and test the *Append(), Prepend() and Insert()* functions together. Errors included new Polygons I accidentally added to the stack and references to other Polygons added and updated incorrectly.   
Compounding these issues, I insisted on writing the main function such that it focused on following the task. In other words, I tried to reach a goal working with unpredictable functions, rather than focusing on debugging those functions, making them reliable and saving a lot of time in the long term. Between finishing *PA1*, writing *MyPolygons* and debugging both, I spent nearly 90 minutes writing and over 7 hours debugging.

According to my log, around 10% were design errors and around 90% were implementation/coding errors. I made very few alterations to the overall design – with the exception of 1 variable in the *Polygon* class, all the design errors were in the output section of the main class. In my initial design, I neglected to consider that the *current* Polygon in the list may not always start at the *sentinel* node unless I declared it as such.  
By contrast, I made a lot of coding errors in the *MyPolygons* class. Each function had at least one misplaced reference and on 2 separate occasions my program was caught in an infinite loop. There were almost no coding errors in any of the other classes, surprisingly enough.

By the principles of Inheritance, *Rectangles* and *Squares* could be extensions of the *Polygon* class. The *Area()* function would become abstracted since the area formula for those classes is much more straightforward – for example, the area function for *Square* could be represented as:  
area = sqrt((x1 – x0)2+(y1 – y0)2)