2.1 Written Report

1. Keep track of how much time you spend designing, coding and correcting errors, and how many errors you need to correct.

The solution was designed, implemented, tested and refined over a 14 day period (03/03/2021 - 12/03/2021). Code was written on 7 of those days.

3rd March

- Initially 3-4h was spent reading and understanding the spec, as well as discussing and clarifying parts of the specification with fellow students.
- Created a GitHub repository for the assessment
- Created all Class files
- Finished implementing `Point` class
- Copied file reading code from previous COMP2240 Assignments

4th March

- Complete first draft of `Node` class
- Start initial work on `MyPolygons` class
- Modify `Node` class to setup sentinel node correctly
- Initial `Polygon` class
- Generate Polygons from file
- `append()`, `prepend()` and `insert()` methods all working in `MyPolygons`

Errors:

- Sentinel not initialized correctly, not sure why

6th March

- Fix initializing sentinel correctly in `MyPolygons` class
- Refactor all insertion methods in `MyPolygons` to use a single `add()` method

8th March

- Create branch off master branch to refactor code to not use ArrayList's
- Once all was working correctly with arrays instead of ArrayList's, opened Pull Request and merged branch back into master
- Cleaned up imports
- Initial `insertInOrder()` method in `MyPolygons` class using `ComesBefore()` method

Errors:

- Realized that we are not allowed to use `java.util.ArrayList`

9th March

- `area()` method in `Polygon` class
- `toString()` methods in Point, Polygon and `MyPolygons` classes

- Generated correctly formatted output for the unsorted list

Errors:

- Sorted list was throwing `java.lang.NullPointerException`, due to trying to print data from sentinel node

11th March

- Fixed sorted list, no longer throws exception.
- File, method and line level comments done

Errors:

- Sorted list is sorted from smallest to largest instead of largest to smallest

12th March

- Fix sorted list to be sorted from largest to smallest (swapped around comparator)
- Fix area threshold for `comesBefore()` to be 0.05% instead of 5%

Overall, approximately 14 hours were spent on the design, implementation, testing and refinement of my solution, as well as writing the associated report.

2. Keep a log of what proportion of your errors come from design errors and what proportion from coding/implementation errors.

The majority of errors that I encountered occurred because of poor design choices or lack of understanding of concepts or the specification. However, a few errors were due to poor implementation and understanding of Java as I have not produced Java code in many months.

It was calculated that approximately 80% of encountered errors were due to design errors.

A large issue was my lack of understanding of how a Circular Doubly-Linked List (CLL) operates with a sentinel node. The number of errors due to design choices could have been reduced by ensuring that I was completely up to date with the course content and verifying my understanding of CLL's before starting to implement my solution. Spending a short amount of time reviewing my knowledge on Java and design patterns would have also reduced the number of errors.

3. Given what we have covered in Topic 3 (Inheritance), how could you treat Rectangles and Squares as special cases of this assignment?

Given that I understand the concepts of Inheritance, I would change my code to be the following:

- An abstract Shape class
- A Polygon class that extends the Shape class

- A Rectangle class that extends the Polygon class, as a rectangle is just the special case of a polygon with 4 sides
- A Square class that extends the Rectangle class, as a square is just the special case of a rectangle with equal length sides
- An Ellipse class that extends the Shape class
- A Circle case that extends the Ellipse class, as a Circle is just the special case of an ellipse with zero eccentricity

This design can be seen in the UML diagram below.