# SENG2200 Report – Assignment Two

## Part One – Time Tracking

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Time** | **Hours** | **Description** |
| Sunday 8.8.18 | 2115 - 2215 | 1 | Relearning java + scope breakdown + repo setup |
| Monday 9.9.18 | 0900-1100 | 2 | Setting up repo on laptop + building simple classes |
| Tuesday 10.4.18 | 2200-2300 | 1 | Completing Point and Polygon + Reading about polymorphism |
| Monday 16.4.18 | 1630 - 1830 | 2 | Rewriting assignment one in Java |
| Tuesday 17.4.18 | 1030 - 1130 | 1 | Implementing LinkedList |
| Monday 23.4.18 | 1200-1500 | 3 | Theory + interface (ComparePolygons) + subclass (SortedPolygons) |
| Monday 23.4.18 | 1630-1730 | 1 | Getting comparePolygons to work |
| Saturday 28.4.18 | 2300-0100 | 3 | Insertion Sort |
| Sunday 29.4.18 | 1200-1730 | 5.5 | Insertion Sort Works |
| Sunday 29.4.1.8 | 2300-0100 | 2 | Comparable Interfaces |
| Saturday 5.5.18 | 1400-1600 | 2 | Iterator and Improved Reader |
| Saturday 5.5.18 | 1600-1700 | 1 | Shapes factory |
| Saturday 5.5.18 | 2100-2300 | 2 | Circle + Semicircle |
| Sunday 6.5.18 | 1200-1600 | 4 | Report and Polishing of code |
|  | **TOTAL** | **30.5** |  |

*Table 1: Time Investment for SENG2200 PA2*

## Part Two – Data Analysis

The time spent on errors was mostly spent correcting design flaws not compilation errors. Compilation errors are a lot easier to fix as the compiler tells you what is wrong. Design errors are something you have to solve on your own, though debugging tools are extremely useful in tracking the logic of your program.

A lot of time (approx. 8 hours) was spent getting insertion sort to work correctly as it was logic intensive. Time spent on compiler errors was light with notable cases being associated with insertion sort in which a number of nullPointerExceptions were thrown.

## Part Three – Program Extensibility

There are more shapes that could be children of PlanarShapes, any planar shape in fact. Squares and Triangles could appropriately inherit functionality from PlanarShapes. Squares and Triangles are more specifically, a type of polygon and therefore are appropriately it’s children. Conveniently, calculating the nearest point to the origin of these shapes and their area can be done using the methods of the Polygon class. They would not require any more member variables. When in the data file they are specified with “T” and “Q” there would simply be another switch case that read in the data in the same way as for a polygon, however instead of a polygon being created and stored it would be a square or triangle. See Figure 1 for hierarchy.

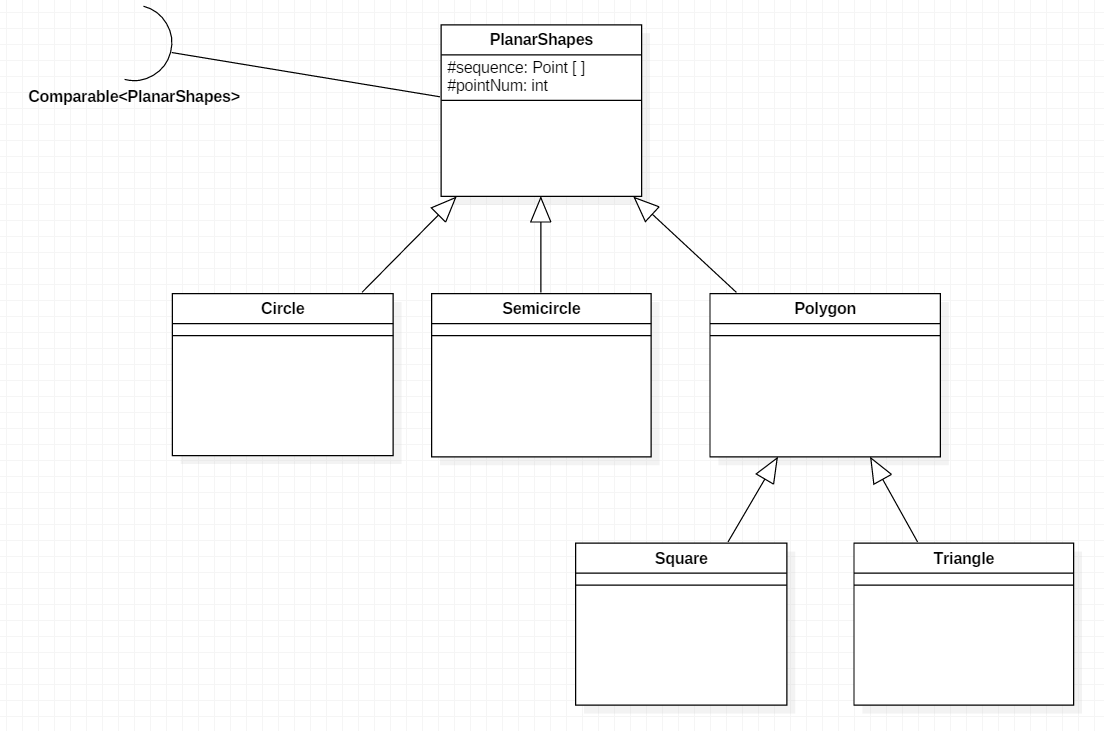
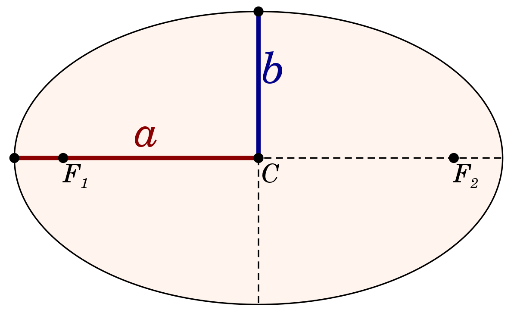


Table 2: Inheritance tree for PlanarShapes

### Part Four – The Ellipse

Figure 2: Minor and major axis of an ellipse

Where (h, k) is the ellipse center and ‘a’ and ‘b’ are the end points of the major and minor axis respectively (See Figure 3).

In order to incorporate the ellipse type into the data structure the preferred minimum information required would be the length of the minor and major axis and the center point and the gradient of one of the axes. Knowing axes lengths and gradient and the center point is sufficient to calculate all four extremity points of the ellipse. Using the following formulae to find extremity point (x, y):

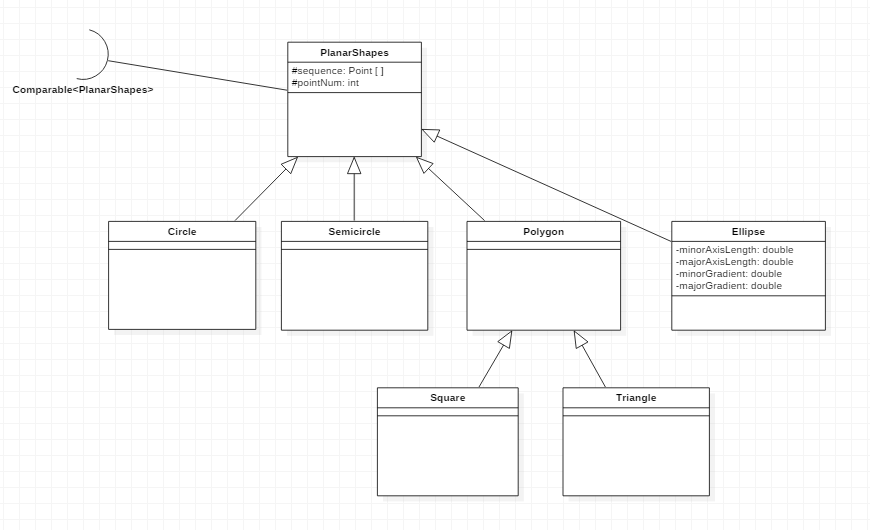
These points are then compared to find the ellipse’s distance from origin.

The area of the ellipse is calculated:

*Where a and b are the major and minor axes lengths.*

Additional attributes of the ellipse are required. Its point sequence would have only one Point in it, the center. Its inclusion in the hierarchy is as in Figure 3.

Figure 3:Heirachy of PlanarShapes with the inclusion of Ellipse



When printed to string it would read *“ELLI = [point0 , majorAxisLength , minorAxisLength]: theArea”.*