PS1

In Java I we had to create a class for a cube. Our cube classes had the instance variable length and methods like base area and volume.

In this problem sheet we were asked to do something similar, create a class for a shape. The difference this time was that we were not allowed to just start from scratch and create a completely new class with new instance variables and methods.

Instead, we were given a class for one shape and asked to assume that this is the parent class. We were then asked to make a class for another shape that was the child of that parent class. We were only allowed to write new instance variables and methods when we absolutely needed to. Everything else had to be inherited.

The first question gave us a parent class called Rectangle. From that we were asked to create a child class called Cube. This was fairly straightforward. For cube’s instance variables, we inherited length and width but created a new one called height. We created a new method called volume for the cube but we used the method from the rectangle’s class called area to work this out. In other words, height x area = volume.

You can see the code here:

The second question was similar to the first but required a bit more thought. We were given a parent class of an isosceles triangle. We were then asked to create a child class that was a regular polygon. The idea was that if you put lots of polygons next to each other with them all pointing to the centre that they would form a regular polygon. For example, four isosceles triangles could be used to create a square. It was this fact that we had to use in order to inform us on what to inherit and what not to inherit in order to create a class for a regular polygon.

You can see the code here:

PS2

In Java I assignment one, we were asked to use an equation of motion to work out where a bullet lands when it is fired from a gun. We were asked to do this for a range of speeds and angles that the gun was held at. This data had to be output as a table.

In this assignment, we had to do something similar. We had to produce a table for how long it takes a car to stop after it brakes, for a range of speeds. The main difference was that instead of being provided a range of values to work with as we had in the Java I problem, this time we had to produce a program that asked for the user to provide the data.

The user had to provide the smallest and lowest speeds they were interested in calculating the stopping speed for. They were also given an increment button to choose which speeds they would like to work with between those two upper and lower speeds.

In order to create the user interface we had to work with Java’s Panel and GridLayout.

You can see the code here:

PS3

The third problem set had two problems to solve for one data set. The data set was a list of names and exam marks in a text document.

For the first problem we were asked to take this text file and print the list in alphabetical order. We were asked to do this using the data structure array.

You can see the code here:

For the second problem we were asked to print the same list descending order of exam mark. We were asked to use the data structure ArrayList for this problem.

You can see the code here:

PS4

Everyone remembers those cheap plastic tile games you got free with things as a child. They typically had 4 x 4 rectangular tiles organized in a grid. You had to move the pieces up, down, left or right until you revealed the unjumbled image.

That was what we had to make in Java II’s last problem sheet. We had to create a 4 x 4 grid of squares numbered 1 to 15 (where the 16th square had no tile so that the other tiles could move around).

One interesting note to make about this game is that when you open it up the tiles are arranged from 1 to 15 is the correct order. In other words, the game is complete when you open it. In order to make the game playable we had to make a shuffle button that jumbled the tiles up for the user to solve.

You can see the code here: