ICT167 - Laboratory Three Notes

Object-Oriented (O-O) Concepts:

Note that this description is by no means exhaustive – read the lecture notes and textbook for a fuller understanding.

- A **Class** definition is a description of an entity (eg: a physical object, identity, concept, etc.). The implementation of such a class is coded in a Java file. A Java file, and the class defined in it, must have exactly the same name, with the filename also including the .java extension. Case sensitivity is important. For example, a class called HelloWorld will be defined in the file named HelloWorld.java

- Components of a class definition are:

    1. **Instance variables** (often called member or class variables, or data members), which store data to describe the characteristics of a class object using primitive data types (int, float, double, etc.) and possibly complex data types (arrays and objects of other classes).
    2. **Method definitions** (class or member methods), which define operations that can be performed on the class instance variables; i.e. they allow the manipulation of the values of the class instance variables. In fact, when a class is defined correctly, its class methods would typically be the primary way (if not the only way) to interact with its instance variables. This is sometimes described as defining class behaviour.

- An **Object** of a class is an instance or occurrence of a class definition, that is typically declared or instantiated in an application/program (but can also be an instance variable of another class).

  If we declare an int variable, that variable can be used to store any integer value; we can manipulate the value stored in that variable via its defined or allowed operations (i.e. +,-,/,* etc.).

  If we declare a variable as an instance or occurrence of a class, we say that we create an object of that class; we can create many objects of a class. Each class object has memory allocated to it, which allows it to maintain its own 'copy' of the defined class instance variables. Two objects of the same class have their own copy of the class instance variables, but they would typically have different values stored in them.

Both objects will have access to the same class methods which allow for the manipulation of the values stored in their own copies of their instance variables (not those of another object of that class). Thus, an object exhibits behaviour because it can call the class methods to manipulate the values of its own instance variables.

- When attempting to design a solution to a given problem involving classes, it is best to do a UML diagram before attempting to define classes or algorithms. It can help you think more clearly about the data that needs modelling in the class, the required class definitions (including member variables and methods), and how they will interact.

- In O-O design input/output methods do not typically go in user defined class methods, even though you may see this in the lecture notes and the textbook. These are there to teach some basic concepts in earlier topics but will not do that in later chapters. It is important to learn the right design from the beginning and that's why this lab exercise discourage you to do that. So, if I/O is not done in a class, where should it be done? In O-O design, input/output methods are typically defined in the client class.

- However, it is not uncommon for a class to provide public validation methods for client classes to utilize. After all the designer of the class knows what input values are acceptable. This can be done via static methods or methods that require a class 'calling object'.

For this unit (unless you are familiar and comfortable with O-O), you are advised to code each class definition in a separate Java file. Only one file (of the possible many Java files you use for a particular project) will contain the main method; that file will be the client class or application/test/demo program. Therefore, other classes in a project will NOT have a main method. A client class program may declare many instances of many class objects; user defined or library classes. For simplicity, make sure all user-defined Java class files are in the same directory/folder as the client class.

IMPORTANT: You should learn how to prepare your lab exercises and understand the concepts that the lab exercises want you to learn. This will be the last notes provided to you. You should learn the skill of writing your own lab notes. For the next lab, you should prepare your own first before reading the lab notes provided.