Unit 3 cs 1102 Learning Journal

Programming 1 (University of the People)

In the past week, we focused on Static Variables and Member variables. From the learning resources and activities, I have learned helpful concepts surrounding the aforementioned focus.

Firstly, I understood that static methods can be called without creating an object for the class where the method resides. The class name is enough to call the method, that is, the caller must first to the class then using the dot (.) operator, point to the method name. For example, ClassName.method(), unlike ClassName object = new ClassName, then object.method() – which is typical with void methods. I also observed that static methods require a return type (statement).

Like other methods, I have come to understand that static methods have to be defined before they are called. Method definition requires specification of the access modifier (public or private), static or void, method name, and parentheses with or without parameters. For example, public static int add(int a, int b){} – between the two curly braces (method body) goes the implementation (function code). About parameters, I have learned that there are formal and actual parameters. Considering the add(int a, int b) method, int a and int b are formal parameters in that they are set to accept values passed by actual parameters (literal, expression or variables) to use in the implementation of the method. Note that if the method is defined to accept integers in the parameter, then the caller must supply integers. Additionally, parameters can be static member variables, these are like constants declared in the class, depending on their declaration, they can be public or private.

For the method to be used, the caller (a class, object, another method, or recursively, the same method) calls the method to action. Since this is a static method concept, the method is called by pointing the class where it resides. This necessitates the concept of Black-Box, my understanding of this concept is that the code or implementation of the method is hidden from the caller. The caller only gets to see the impact of the implementation and is not interest in the code. For example, when operating a printer (machine), the user is less or not interested in how the printer prints, but is interested in the output. The understanding here is that the Black-box received input and then it processes it (how it does it, it does not matter) and then output. This concept gave me a lot of ideas as I will be working on projects in Java.

Comprehensively, methods are subroutines that make the implementation of a program more efficient. Coupled with the skills I have gained in the previous Units, I feel more comfortable meaning code in Java Programming Language, especially the Object-oriented aspect. In fact, the assignment, was one of such motivations.

This week I have received positive feedback from peers, at the same time, going through and accessing my peers' work was very good and challenging.