**Declaration of plagiarism**

**Name(s):** Stephen Mc Donagh

**Programme:** Case3

**Module Code:** CA341

**Assignment Title:** Comparing imperative and object-oriented programming

**Submission Date:** 20-11-2017

**Module Coordinator:** David Sinclair

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

**Introduction**

An Imperative programming (also known as procedural programming) style is one that uses statements to change what is needed within the program. By writing in an imperative style you are telling the computer exactly what you want it to do step by step to accomplish the required goal. Imperative programming is of such importance as the hardware implementation of most computers is imperative. From a low level perspective (x86 assembly language) the program is defined by the memory contents and the statements tell the computer what to do, example of this is “ mov eax, 3”. In higher level languages variables are used to do the same thing “ x = 3”

In comparison Object oriented programming paradigm organised around objects rather than actions. Languages that support OOP use inheritance for code to be reused. One major advantage of an OOP style over an imperative style is that it allowed developers to create modules that don’t need to be changed when a new type of object is created. The developer can just create a new object which inherits many of its features from other classes for example with a chess application the class ‘Queen’ will inherit many of its features from class ‘Piece’. This makes it much easier to change.

I decided to do my object oriented style in java as it is the language I am most comfortable with and had planned to do the same with the imperative style until I was informed that java is not purely imperative so therefore I chose to do this part in python.

**Comparison**

**Classes:**

In my object oriented style of this assignment I broke up my project into 4 different classes which were ToDoList, Queue, Task and Event. I did it this way as I was able to break everything up and it makes it much easier than having everything in the one class. In my queue, task and event classes I had to make constructors for each of these objects in order to create them. I also added in any methods necessary for example my enqueuer and dequeue methods into the queue class. My ToDoList class acted as my main class and this is where I created my Event, Task and queue objects and initialised values to each of them in order to test that my program acted as a queue should. Inheritance (creating classes that come from other classes) is a huge advantage of OOP over imperative programming

In contrast, my imperative style which had the exact same function as the OO style used no classes. This is why I could not use java as you need at least one class in java making it not purely imperative. I have everything in one “class” which in my opinion makes the program a bit messy and if it was a much larger program with thousands of lines it would become increasingly difficult for someone who didn’t write the code to understand and make any changes necessary.

**Order Of execution:**

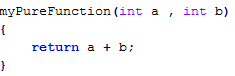
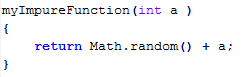
In my OO style it is clear that the order is not important. In my classes queue, Task and Event I can have my code in any order that I want and it wouldn’t affect how my program works. The only place where order is important is in my main class as this is where I tested that my application works. So with OOP the order of execution is of low importance.

In my imperative style the order of execution is of high importance. This is more relevant with low level languages such as assembly as you’re changing variables and registers each line and by having even one line in the wrong place could result in the program giving the wrong output.

**Methods/Function:**

In an object oriented style there is no restrictions to what you can and cannot use. You’re able to use library functions in your own methods or over write them and create your own versions. An example of a type of function which is not imperative would be Math.random() which gets a random number or Date.now() which gets todays take, an example is below. These are called impure functions and are something which cannot be used in an imperative style of programming.

Imperative style is the exact opposite as you can only create pure functions. Pure functions depend on its arguments; therefore if you call a pure function with the same arguments you will always get the same output. An example of a pure function is below. You cannot use library functions which is something I didn’t realise until after I made my imperative program. For example in my isEmpty method I used the built in function length which I then had to modify after finding out that this is not an imperative style.

**Objects:**

In the object oriented style I am able to create instances of my Event, Task and Queue objects by doing something such as ‘Event myEvent = new Event();’ after this line is executed the constructor is called. This is the main concept of object oriented programming. The advantage of this over using the imperative style is that code can be reused and recycled, thus saving time. Another huge advantage of OOP is polymorphism which is the ability for an object to take on different forms.

As can be seen in my imperative part of this assignment I don’t create any instances of objects and don’t call any constructors from other classes as this is not an imperative style of programming. Instead the task and event “objects” I created are final and no other instances of them can possibly be created. This is an inconvenience as if this program was much larger and more Task and Events were needed it could get messy and hard to understand.

**Design:**

With the object orientated style I found little difficulties