**G00364753 – Patrick Moore – Multi-Paradigm Programming – Shop Report**

As part of the *Multi Paradigm Programming* module for the *Higher Diploma in Data Analytics*, we were assigned a project to model the behaviour of a shop. The objective of the exercise was to model the shop in both C and Java and to compare the 2 solutions. In the C, the shop was modelled using a *procedural paradigm*, where as in Java an *object oriented approach* was employed. The aim of this document is to explain both approaches to the problems by discussing the similarities and differences between both programming styles.

An appropriate place to start is to define what a *procedural paradigms* and o*bject oriented paradigms* are. A *procedural paradigm* is one which is based on calling procedures. When using this paradigm, the problem to be solved is broken down into smaller and smaller problems that can be solved using functions (or procedures/sub-routines), and these smaller solutions are combined to solve the problem. In procedural programming, the program is often written in one file, and the commands in the script are executed in order, from the top if the file to the end of the file.

An *object oriented paradigm* is one that is designed around the concept of objects. Objects are data structures that are modelled on real world things. They are defined by a template called a *class.* The class definition contains all of the information pertaining to an object including the data that it holds (for the shop this would include the cash and the stock list), how its created (c*onstructor methods* that determine how object instances of a class are created) and how the object data fields are updated (these are functions know as *methods* that are called by an object to interact with the data fields in that object).

The two paradigms will be discussed under the following headings:

- Project layout & code maintainability
- Data types
- Creating new items
- Updating items

**Project Layout & Code Maintainability**

For the procedural paradigm the project layout was quite straightforward. The entire project consists of one program script (shop.c), a csv file for the shop and stock information and a few test orders used to test the functionality of the code. The script was written from start to finish, with care taken to ensure that any procedures required were defined first in the file before they are called – otherwise the program won't compile. The layout of the object oriented project was much more complex. Each class required its own file, this needed to be named exactly the same as the class name. However the methods in each class can be written and called in any order without affecting the compilation of the program. Even though the object oriented layout is a bit more complex – it is much easier to navigate when writing code. For example methods relating to the Customer are in the customer file and are very easy to locate and update when maintaining the code, whereas the functions in the C program don't explicitly relate to any one data structure but must be located in the file in such a way that they are defined before they are called. It can be difficult to find them and update them when maintaining the code.

**Data Types**

In Java everything (except primitive data types – ints, doubles, chars etc.) are modeled as objects. This means that it is possible to create methods to interact with them. The benefit of this is that there is some built in functionality in Java using these methods. In C there are no objects, so instead structs are used to model data types. They are similar to classes in that they define something that has many attributes (like a shop or a customer), but they don't have dedicated methods for performing dedicated operations on the fields in that data.

**Creating New Items**

When using the object oriented approach a special method is created for each class called a *constructor.* This defines how to create an instance of the class and set iinitial values for the attributes. So in the shop program we created a constructor for the shop class that took a csv filename as a parameter and then read in the initial values for the object from the file. We also created a similar one in the customer class that was used for creating customers by reading data in from a csv file. One of the benefits to using an object oriented approach is that we can create many different constructors for the same class that are called depending on the data passed as a parameter. This process is known as overloading. Overloading was used in our program to create customer objects in a different manner depending on whether or not we were processing a csv file or live order. As a privacy conscious individual, I don't believe that a shop needs to know a customers name or budget when they go into a shop to buy items. They just need to be able to process the shopping list and determine the total cost. In Java this was a trivial problem to solve by using an extra constructor, however in the C program a new struct (liveOrder) and 2 new functions (createLiveOrder and processLiveOrder) were required. Overall this was much easier to accomplish using the object oriented approach.

Another issue encountered regarding creating new items was related to the way strings are handled in C. As stated above there are no strings in C, instead there are 'character arrays''which are lower-level are require the programmer to manually allocate memory of sufficient length for the character array. This is quite challenging to do dynamically as it requires clever algorithms anytime a user is going to enter a string to ensure that enough memory will be allocated to that string. This was also easier to accomplish in Java due to having both native String data types and automatic memory allocation.

**Updating Items**

In C, in order to update the fields of an instance of a struct, we need to create a function to do this and pass pointers to the memory location of the particular struct to the function, so that the method can be update. This was done in the processOrder function where pointers to a customer and shop are passed so they can be updated, and also processLiveOrder where a pointer to a shop is passed.  This is a little bit tricky as it is very 'computer focused' - taking the programmer back into the memory of the computer. In the Java program, the objects are modeled on 'real world' phenomenon such as customers and shops, and there are methods defined to both fetch values from the objects, and also to update values in them while abstracting the memory location away from the programmer. For this reason I believe that the object oriented approach is more intuitive for even problems of reasonable complexity such are our shop example.

**Conclusion**

This project was carried out in order to gain an insight into the difference between object oriented programming and procedural programming. While both paradigms have their places, and individual strengths I believe that that the object oriented approach was a much more appropriate paradigm for modelling the shop. This is mainly due to its superiority in the following areas:

- Ease of code navigation and maintainability
- Intuitive data types modelled on real world phenomena
- Use of overloading in constructors (this is available in other methods too)
- Automatic memory allocation in Java vs. manual memory allocation in C
- Getting and setting object attributes using methods without resorting to pointers