Multi Paradigm Programming – Shop Assignment

Comparing Programming Paradigms

Zoe Cahill – G00244816

11/12/2019

Repo Link: https://github.com/zoecahill1/multi-paradigm-programming

Comparing Programming Paradigms – Procedural vs Object Oriented

Introduction

In this report, we will discuss some of the benefits of using both paradigms as well as some of the difficulties using the example of building a shop. The language we will use to examine procedural is C, and our object oriented example is Java.

We will also look at some important concepts that you need to be aware of in using these paradigms and how they compare. After this, we will look at the design and development process for each language. We will also look at the advantages and disadvantages of each paradigm, and examine some important issues such as reusability and maintainability of code which are also important factors when choosing what approach to use. And finally we will look at some of the key differences noticed between Java and C.

Procedural Programming

Procedural programming is an excellent start to learning to program for most people - it is simple and fairly straightforward. With a good program design, it can allow for good isolation and containment for variables when properly scoped with functions and control loops [2].

Procedural programming takes a more “top-down” approach to programming. So looking at our shop problem, starting out was relatively simple – where the problems really begin with design is when the functionality begins to expand. Where OOP uses classes and objects, procedural takes on problems by solving it from the top of the program down to the bottom – meaning you keep breaking it down to sub-problems until that sub-problem becomes simple enough to solve[3].

Despite the logical simplicity of this approach, it soon became apparent when building the shop program - the more lines of code you add, the harder it will be to maintain as you will have to locate and edit everything related in various parts of the program. Updating a particular variable used by many methods may involve combing through lines and lines of code. This is a big advantage of OOP over procedural.

Object Orientated Programming

Object Orientated programming uses classes and objects to create models based on the real world environment [1]. For example, in our shop program we have a collection of objects relating to the representation of a customer. This will pass messages when called to - for example, we might request a summary of information about the customer like their name and budget. These objects are able to pass, receive or process information in the form of data [2].

Initially, as humans and how we process data, the procedural approach might make more sense - at least it did to me. However as the program expands, the real benefits of OOP start to become apparent. It is much easier to maintain and modify existing code as new objects can be created inheriting characteristics from existing ones [2]. For example, when building the live mode of the shop, all we have to do is make a method to create a new csv to feed into the existing architecture. This csv is simply feed into the program to create new instance of a customer which is much easier to manage and alter than its C counterpart. This meant that the development time and adjusting the shop program was much simpler. It was also much easier to follow and understand the flow of the program, which I imagine would make adding to the program for a new developer much easier.

Another OOP benefit is memory management. In this particular example, in the C implementation we assigned memory and once the variables were initialised they stayed until the program finished (I'm sure a more accomplished C programmer would handle this a little better than me as C gives you total control of this). In the Java implementation, the program will dump unused classes or objects which will free up more system memory [4]. Therefore, I think this would make it the faster and more efficient implementation and also one less thing for the developer to worry about. Depending on what you are trying to do this could be an important factor in choosing a paradigm.

Comparing Paradigms

The design process in each of these paradigms in quite different – with OOP, the designs can be reused and recycled through the program. In procedural we can't generally do this. Also the planning process will be different between the two paradigms. In a procedural approach, instead of thinking about classes or objects, we will be thinking about how the program should execute as the code is executed [2].

Having coded the shop assignment, I can imagine that the Java implementation would have been 10 times harder if it had been the first one! Doing the C version first meant we had already figured out a lot of the logic of the program. Much of OOP development will involve careful planning of both current and future design considerations [3]. When planning, a developer usually starts with a list of classes that are needed in the program. For example, ours had classes for customers, products, orders etc. I found this difficulty with tracking the customer's orderTotal (found in Shop class). As we didn't actually plan out the design of the shop at all, I ended up having to make this orderTotal static which defeats the purpose of OOP, but I was limited on time. We should have used encapsulation with getter and setter methods instead. OOP can add a lot of time to development, procedural is much easier to just "jump into". Having many use cases before starting development would help you to decide if it is appropriate to use this approach. This additional time in the planning phase can also help us to uncover any problems in the design - thus making it much easier to fix in the beginning rather than later in the process [2].

Although the reusability of OOP is very helpful when programming, it is also very easy to override a method that should not be overwritten, or re-implement a class in a way that causes problems in subclasses [3]. This is also an important consideration when choosing your paradigm. So even though there are many benefits to OOP, sometimes budget, time and complexity will force one to choose principles when and not follow them consistently due to over complexity and time. I certainly felt at times when getting used to the idea of OOP that it was needlessly complex! However OOP was much easier to debug, once you got your head around the object oriented concept.

Java vs C

We will now look at some of the differences encountered between these two languages. The first and most relevant – Java is object-oriented while C is procedural. We have already examined some of the differences between these paradigms.

Another key different between these two is that java is an interpreted language, while C is compiled [5]. This means that Java code is more portable. This is relevant for this project as the environment was much easier to set up for Java than C comparatively. C is a much lower level language than Java, meaning that it was a little bit more difficult to interpret [6]. I found Java a little easier to understand because it is a higher level language and had a little more relevance to shop problem.

When it comes to the Java implementation of the shop, we did not need to use *'s and &'s for pointers as there are no pointers in Java. Java will manage this behind the scenes, whereas we needed to explicitly handle these in C [4]. This added an extra layer of complexity to the program in the C implementation. In the same vein, Java also handles memory management behind the scenes using a garbage collector. In C, using malloc, we had to manage the memory ourselves. The garbage collector in Java simply deletes the objects that are no longer referenced [6].

Another huge difference between Java and C how they both handle errors and crashes – this is the difference which, as a novice programmer, had the biggest significance for me. When an error happens in a Java program, it will throw an exception which we can then handle using different techniques [4]. It will also give you some information about the error. In C if you have an error, you have an error. I found it incredibly difficult to debug and handle errors in C compared to Java.

Conclusion

It is a difficult choice one which particular paradigm one should use. If time is not a factor and the complexity of the program is higher, I would say OOP is the way to go. If you need a very quick solution for example you need to patch a critical fix for a production server, then procedural is the better choice as you can implement and deploy the fix much quicker. However there is no rule in choosing…it really does depend on the situation which concepts or methodologies you use. In conclusion however, for the implementation of the original shop problem, I think the OOP in Java is the better solution, particularly when thinking ahead of adding functionality etc. I would however take a lot more time to plan the OOP architecture were I to work on a similar project.

References

1. Programming Languages: Principles and Paradigms (Undergraduate Topics in Computer Science) - Maurizio Gabbrielli, Simone Martini.
2. https://www.codementor.io/learn-programming/comparing-programming-paradigms-procedural-programming-vs-object-oriented-programming
3. https://neonbrand.com/website-design/procedural-programming-vs-object-oriented-programming-a-review/2/
4. https://www.geeksforgeeks.org/difference-between-java-and-c-language/
5. https://www.educba.com/c-vs-java/
6. https://www.slideshare.net/Ankit92Chitnavis/procedural-programming-30623196
7. https://www.quora.com/What-is-difference-between-Java-and-C