SHOP PROJECT REPORT

MODULE: Multi-Paradigm Programming

AUTHOR: Ante Dujic

This is the report created as a second part of the Multi-Paradigm Programming module project on ATU. First part of the project was to create three Shop applications with same functionalities and same user experience – two using procedural approach and one using object-oriented approach. The aim of this report is to discuss the solutions achieved using those approaches. The Procedural application was written in Python and C programming language, while Object-orientated application was written in Python.

1. INTRODUCTION

"**Programming paradigm** is an approach to solve problem using some programming language; to solve a problem using tools and techniques that are available to us following some approach (GeeksForGeeks, 2018)". To simplify, programming paradigm is the way the programmer writes the program. There are multiple different programming paradigms and different programming languages allow usage of one or multiple paradigms. Ones used for this project are procedural paradigm and object-orientated paradigm.

Procedural Programming Paradigm is the approach where the program code gets divided into procedures (Isaac Computer Science, 2022), often called functions or methods. Each procedure carries out single computational task. Any procedure can be called at any time by itself or by another procedure. This paradigm tends to follow top-down approach. Some of the advantages of the Procedural paradigm are that this paradigm is very good for a general-purpose usage, its simplicity, reusability within the program and the memory usage. The disadvantages are that the code is often not reusable in another program and the difficulty to relate the code to the real-world objects (www.computingatschool.org.uk, 2022).

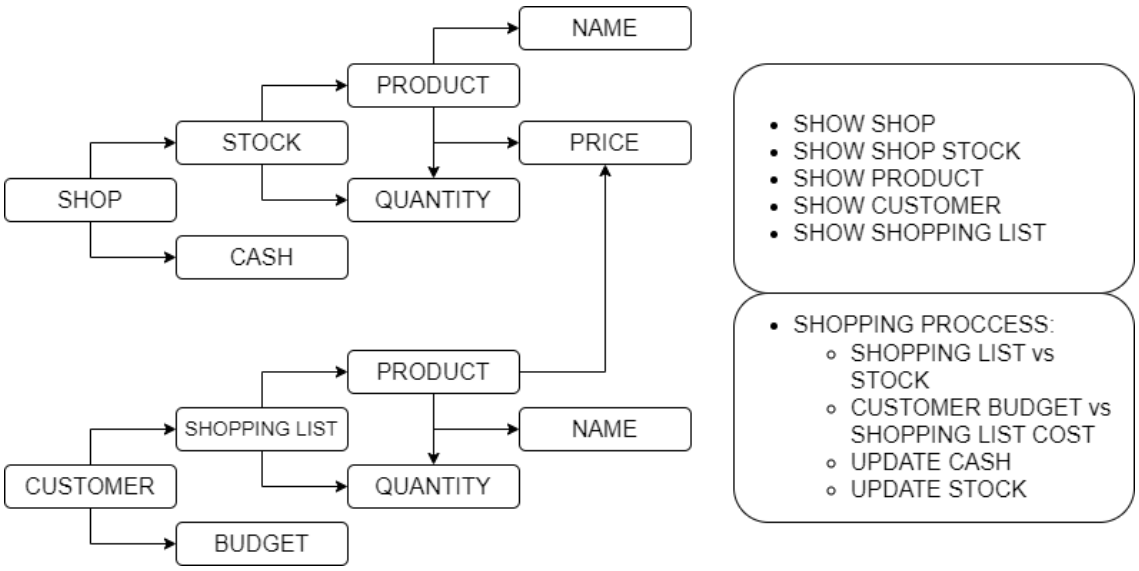
Object-Oriented Paradigm is the approach where the program is designed around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour (Gillis, 2021). It contains the code to perform certain operations on its data fields and attributes. The structure of this paradigm are classes, objects, methods and attributes. There are three important properties of using this paradigm: inheritance, encapsulation and polymorphism. Inheritance means that one class can be created using (inheriting) attributes and methods of another class. Encapsulations means that attributes can be protected and that other objects don't have access to this class. Polymorphism allows usage of the same methods in the subclass but with different functionality. Object-oriented paradigm follows the bottom-up approach. The advantages of

this approach are reusability, flexibility, real world relation and maintenance. The disadvantages are the program size, performance and the complexity of the paradigm (www.computingatschool.org.uk, 2022).

2. APPLICATIONS OVERVIEW

The applications written for this project all have the same functionalities. They are a representation of a shopping experience. There is a shop with certain cash balance and stock balance. Stock is built of the products that each have a name and a price, and the quantity of each product. There is also a customer with a certain budget and their shopping list. Same as the shop stock, the shopping list is built of the products and their quantity. In this instance, only the name of the product is included, as the customer doesn't know the shop prices before entering the shop. This price is "collected" from the shop stock. The shopping list can contain items that are not on stock, or more items then available on stock. Only available items get added to the customer cart. If customer has enough money, they can purchase the products. The shop cash balance increases for the cost of the items paid and the customer budget decreases for the same amount. Also, the stock balance on the shop gets decreased for the purchased product quantity. Shopping can be done by either existing customer (csv file) or in real time by a new customer (application user). The applications have a menu for the user to choose one of the options: to see the current shop stock and cash balance, to shop with the existing customer and live shopping. Further below is the explanation of how the applications functionalities are designed using the two previously mentioned paradigms and what the main differences and similarities are.

SHOP VISUALISATION



The diagram above was used as a base for creating the applications. It is only a visual representation to help deciding what functionalities the applications should have.

Procedural C

There are four different structures (structs) created to hold the data: Shop, ProductStock, Product and Customer. They are used to group together variables related to them. Those variables can be of various types, including another struct. Having the above diagram in mind, Shop holds a Cash which is a double and a stock which is another struct - ProductStock. ProductStock holds Product struct and an integer Quantity, etc. Structs don't contain any functions. All the functions exist outside the data structures and any modification to the state of the data containers is done by those functions. Functions are designed following the logic of the shopping experience and the activities that would occur in one. Thus, there are functions to create the shop and a customer, print the shop, print the customer or product details, to count the cost, update shop cash, etc. This demonstrates how all the tasks get broken down into individual functions. Certain functions are being reused. The best example is using the same functions for shopping with the existing customer, or doing the live shopping. While one is loaded from the existing csv file, and the other is done by taking user inputs, both functionalities use the same functions to print the customer, print the shopping list, calculate cost, update stock, etc. To access the variables in the struct and make any changes an instance of a struct needs to be created. This is done in the menu, when the user chooses the wanted option. Those instances are then passed into different functions to perform certain action.

Procedural Python

Procedural Shop in Python follows very similar logic to the one in C. Although, in C application, there are structures created to hold the data, this has been done slightly different in Python. Classes could have been used for the same purpose, but instead dictionaries have been used. They are defined and populated within the functions. Both procedural applications are designed around the functions, and using dictionaries within the functions makes this even more obvious. Same as in the C version of the shop, each task is broken down into the functions and they are been called in the menu, when the application user chooses the wanted option, and by the other functions. The biggest difference in the two procedural applications is that designing Python application is less rigid. All the code in C has to be in the correct order, while this is not necessarily the case in Python. Also, not having experience with C language made writing this code laborious. The simplest task as defining a variable or changing its state require some sort of low-level thinking.

Object-Oriented Python

Using the Object-Oriented approach allows easier translation of the real-world objects into the code. Having the above diagram in mind, there are classes defined for each real-world object: Shop, Customer (and LiveCustomer), ProductStock and Product. These classes are object constructions, and each contain object properties. Therefore, Shop class holds the budget and the stock, stock holds products and the quantity, product holds name and the price, etc. Each property can also be another object, so the object Product gets passed as a product property in ProductStock, and the ProductStock as object in the Shop, etc. This logic is very similar to Procedural approach. As we stated before, classes could have been used as

data holders in the Procedural Python application. The main difference is that the classes in Object-oriented approach can also hold the functions. Each class has the functions related to that class. So, the Shop class has the function to update cash or budget, class Customer has the function to print the shopping list or create the cart cost, etc. Classes in Object-oriented application also have the built in *repr* method, which is a string representation of that class. Printing an object accesses this method. Classes are responsible of updating its own state. Considering the idea of recreating a real-world environment, I found this approach to be more suitable. Same as the Procedural approach, reusability is a big focus in Object Oriented approach. Aside of reusing same functions, Object-oriented paradigm allows the previously mentioned inheritance. An example of inheritance is the class LiveCustomer, which is a child (subclass) of a class Customer. Polymorphism is also present in the same class. So, the LiveCustomer class inherits the functions of the class Customer but has its own *init* method defined, to suit the purpose of a live shopping - taking values from input instead from a csv file. The instances(objects) of the classes are created in the Menu when the user chooses the wanted option.

Menu is created in a similar manner in all three applications. Same was done with the functions with the same purpose where same comparators or same loops were used. For example, same logic was applied to create the cost in all applications, or to assign the price to the customer shopping list products, etc. This “regular” code hasn’t been discussed in this report, but is commented in the applications code.

3. CONCLUSION

While Object-oriented approach might be harder to grasp at the start, I found it to be more suitable to create a Shop application. The main reason being the easier translation of the real-world environment into the code. This made accessing certain variables more logical. Apart of that, I found both approaches to be very similar in terms of code navigation and maintenance. This is likely due to the applications size, which is not too big. As the size of the application would become bigger, I think using Object oriented approach would prove to be a better choice. Both approaches allowed for code reusability with this being even further extended using Object Oriented paradigm, due to the ability of inheritance. Using C language proved to be challenging at times, but this is because of the way I am used to think, using mainly Python prior to this project. This, however is not connected to the paradigm but the fact that C is a different programming language.

4. REFERENCES

- GeeksForGeeks (2018). *Introduction of Programming Paradigms - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/introduction-of-programming-paradigms/.
- Isaac Computer Science. (2022). Isaac Computer Science. [online] Available at: https://isaaccomputerscience.org/concepts/prog_pas_paradigm?examBoard=all&stage=all.
- www.computingatschool.org.uk. (2022). Object Oriented vs Procedural Programming paradigms. [online] Available at: https://www.computingatschool.org.uk/news-and-blogs/2022/may/object-oriented-vs-procedural-programming-paradigms [Accessed 13 Dec. 2022].
- Gillis, A. (2021). What is Object-Oriented Programming (OOP)? [online] SearchAppArchitecture. Available at: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP.