Multi Paradigm Programming
Shop Project – G00411347 – Jamie Roche

Objective:
The objective of this project is to create three identical programs across multiple paradigms and programming languages, then compare and contrast the programs and the implementation.

The first program is python using procedural programming.

The second program is C using procedural programming.

The third and final program is python using objective orientated programming.

Interpretation of the Shop Program:
The programs must have the same user experience across all three programs, so the user can not determine which language or paradigm is used.

The program is a simulation of a shop, my implementation uses menus for the user to navigate to select the functions to avail of.

The initial menu allows the user to select the one of four options:

```
Menu
====
1 - Buy With Shopping List (CSV)
2 - Buy Live
3 - Check Shop Stock
4 - Check Shop Cash Balance
x - Exit application
Select:
```

Option 1 allows users to make a purchase using an existing shopping list.

The user is then asked if they are an existing customer, 1 for yes, 2 for no.

```
Are you an existing customer with a shopping list?
Type 1 for yes, 2 for no
Select:
```

If the user selects 1, for an existing customer they are then presented with a list of customers which is pulled from the files in the customers folder.

The user types their name and their order is completed if they have enough in their budget, and enough stock is available. Their shopping list is printed and the transaction is completed, updating the stock and cash stored in the stock csv file.

```
What is your name from the list above?dom
Name: Dom cash: 15.0

Shopping list for Dom
Name: Coke Can, Quantity: 1
Name: Bread, Quantity: 2
Name: Bin Bags, Quantity: 1
Total cost to Dom = 5.0
Dom new balance = 10.0
Welcome to the python procedural shop
----------

Menu
====
1 - Buy With Shopping List (CSV)
2 - Buy Live
3 - Check Shop Stock
4 - Check Shop Cash Balance
x - Exit application
Select:
```

If the user opts for option 2 at the main menu, the user is presented with the current stock in the shop, prices, and quantity of items available.

```
Select: 2
Menu
=========================
Name: Coke Can, Price: 1.1, Quantity: 79
Name: Bread, Price: 0.7, Quantity: 77
Name: Spaghetti, Price: 1.2, Quantity: 100
Name: Tomato Sauce, Price: 0.8, Quantity: 87
Name: Bin Bags, Price: 2.5, Quantity: 20

=========================
Enter Your name:
```

The user enters their name and budget, and the items and quantities they would like to buy.

```
=========================
Enter Your name: j
Enter your budget: 555
Please enter the itmes you would like to buy, when ready to submit order enter y, or x to exit

Select: Bread
enter quantity: 4

Please enter additional itmes you would like to buy, when ready to submit order enter y, or x to exit

Select: Coke Can
enter quantity: 3

Please enter additional itmes you would like to buy, when ready to submit order enter y, or x to exit
```

When ready to buy the users enters y or x to exit.
The program then generates a temporary csv file called live basket to reuse the function to complete the purchase in the same manner as set shopping lists.

The option 3 at the main menu prints the current stock in the store.

```
Select: 3
Current Stock in Shop

Name: Coke Can, Price: 1.1, Quantity: 76
Name: Bread, Price: 0.7, Quantity: 73
Name: Spaghetti, Price: 1.2, Quantity: 100
Name: Tomato Sauce, Price: 0.8, Quantity: 87
Name: Bin Bags, Price: 2.5, Quantity: 20
Welcome to the python procedural shop
----------
```

Option 4 prints the current cash in the store:

```
Menu
====
1 - Buy With Shopping List (CSV)
2 - Buy Live
3 - Check Shop Stock
4 - Check Shop Cash Balance
x - Exit application
Select: 4
Shop Current Balance is €914.9
```

All three programs use the same customer files and the same stock file, they also all use the same temporary Live Basket file to facilitate the live buying function.

Implementation:

The implementation of the procedural programs in C and Python are linear each function completes the task or calls the next function in line to complete or progress the task. The data is passed through the functions, but the data and functions are separate entities.

The Object Orientated Python program uses objects that contain data structures and functions which process the data. Object orientated programs use encapsulation to ensure the unnecessary alteration of the data. Inheritance allows classes to inherit properties from established classes.

The C procedural program and the Python Procedural program are implemented very similarly as they use the same paradigm. Straight forward methods that are called to perform specific tasks, in a specific order. I used dictionaries and lists to store the data in the Python Procedural as opposed to data classes to keep the structure as far from Object Orientated as possible.
The program displays a menu, from there the user selects what option they want to use.

If the user selects, they are an existing customer the existing customer function is called, the user is then presented with a list of existing customers taken from the customer folder. If the user verifies, they are an existing customer by typing their name and it matches a name in the buying csv is called, this calls the read_customer function to read in the users shopping list, and budget. The readInshopCSV function is called to read in the shop details, current cash value and stock price and quantities.

If the transaction is deemed acceptable, the user can afford the transaction and there is enough stock in the shop to perform the transaction the transaction is complete, and the stock and cash balance of the shop is updated. If the transaction is not a valid transaction a flag is set to indicate

where the transaction failed. Whether it was due to a lack of stock or inability for a customer to afford the transaction. The error is then relayed to the customer. When the user selects live buying option, the user enters their name and budget, and the items they wish to purchase. The program then takes the information and creates a csv file in the same format as existing customer files and process the temporary LiveBasket CSV file in the same way as the existing customer files. This allows the reuse of the functions to read in the shop values and update these values on a successful purchase.

The Python Object Orientated Program uses objects and classes to store the data allowing the program more flexibility and modularity. Within these classes are functions that handle the operation of the program. The all the functions in the objects orientated program are in classes except for the function that prints the menu as there is no data associated with this function, it was left to be a standalone function not in a class. The program exhibits in the same manner as the procedural program, the same menus, and the same outputs to the screen for failing transactions and data retrieval, i.e. stock values and shop cash values.
The data in the classes can only be edited if explicitly edited through a function in the class ensuring protection against accidental value changes this is the idea of encapsulation, The data in the product and product stock classes can not ne edited outside of functions where the classes are explicitly used. The livebuying and existing customer functions were adapted to fit within the shop class. The Object orientated projects uses inheritance, both Shop and Customer classes uses instances of the productstock class, which in turn stores instances of the product class.

Conclusion:

In this instance, this is a short and simple program. Both paradigms can effectively complete the task. If the program required was more complex the Object orientated Programming paradigm would be more suitable as the modularity and reusability of the program would lend itself to an easier implementation and structure to the program.

Both paradigms have different use cases, the ability to create objects or classes makes the object orientated paradigm more robust, with better capabilities of handling more complex programs.
The idea of encapsulating the data within the classes, products and product stock ensure the data is protected and only altered when specifically required.

Allowing both the shop and customer classes to use the product stock class displays the use of inheritance, where both shop and customer classes uses the same child class product stock to store the shop stock items and the customer shopping list items respectively.

Although using different paradigms and languages, all three programs use similarly structured functions, and handle the data processing in a similar manner.
Each program reads in a csv file with the user shopping lists and alters the original stock csv.
Or uses a temporary Live basket csv file to store the users' items and process the order using the same function that is used to process the static user order files.