Multi-paradigm Programming

Shop Assignment

David Sheils – G00364696

## Introduction: Objectives of the assignment

The purpose of this assignment is to model a shop, based on the shop program(s) developed in the lecture series “Multi-Paradigm Programming”.

In particular, the aim is to extend the functionality as follows:

The csv from which the shop is to be created must have an opening cash balance

* There should be two methods of reading in a customers order:
* CSV mode. In this mode, customer details will be loaded from a CSV.
* Interactive mode, in which the customer order is entered via the console/terminal.

After processing the order, the cash balance of both the shop and the customer should be updated.

This functionality should be implemented (ideally identically) in C and in Java.

The key learning of this assignment should be demonstrated, comparing the approaches in the object-oriented solution (in Java) and in the procedural solution (in C).

## Business Logic and overall operation of programs

On running of the program, the **shop will be created and populated** from a csv file (“stock.csv”)

This is done by reading the first line, which is an integer into the variable that will contain the opening cash balance of the shop. This variable will be updated (i.e.) increased after each transaction.

The program will then **get the customer details**. The user is offered two modes by which this task may be accomplished:

1. “**CSV Mode**”. In this mode, the customer details are read in from a csv file (“orders.csv”). The first line contains two values, a string representing the name of the customer and the value (as a double) of the customers opening balance. The remaining lines will contain the names of the products and the quantity which they wish to order. These are read into the variable/object that will represent the customer details.
2. “**Interactive Mode**”. In this mode, the user plays the role of the customer, entering their name and their cash budget, when prompted, then entering the name of the product and the quantity of which they wish to purchase. The customer may enter up to 10 items; if they are requesting less than 10 items, they type “end” (all lowercase) to exit and to allow the program to process their order.

Please note that it is assumed that (a) the customer does not know the price of any item they wish to purchase or (b) whether the item is in stock. Where an item is in stock, however, it is essential that (in both modes) the entered product name must be spelled exactly as it is in the stock list (“stock.csv”).

The **processing of the customer’s order** is done as follows:

1. The customers opening cash balance is read.
2. The program then cycles through each item in the list of items the customer has ordered.
3. The program checks the name of the item against the name of each item in its stock (hence the necessity for an item to be spelled/entered exactly as it is spelled in the stock csv. If the item is found:
4. The program will then check that the quantity of the item the customer has sought is less than or equal to the quantity in stock. If the quantity sought is higher than the number in stock, the quantity that will be sold to the customer is capped at the amount in stock.
5. Then the program checks if the customer can afford the purchase of the item in the quantity required (capped where necessary), by checking the cost (i.e. the price of the item multiplied by the quantity sought) against the customers budget. If the total cost of the item exceeds the customers budget, the customer is told that they have insufficient funds for the transaction and the program moves to process the next item in the customers orders list.
6. After each item is processed, the customers budget is decreased by the cost of the item and the shops cash balance is increased by the same amount.
7. At the end of the processing cycle, the program will output the final shops cash balance and the final customers balance.

## Implementation of Functionality in C and in Java

The most fundamental difference between the two implementations is that the functionality of the C program is implemented in a single script, the functionality of the Java program is implemented as a collection of java classes which interact with each other.

The following is a list of the components of the program and how they are implemented in the respective implementation

There will be two input files.

Input files:

**Stock.csv**

* Holds initial cash value of shop
* Holds product name, product quantity, product price

**Orders.csv**

* Holds name of Customer and customer total budget
* Product required, quantity required
* While the shop cash balance and the customer budget will be updated in the operation of the program, the contents of the file will not be amended. Therefore, each time the program is run, the shop will start out with the same budget and stock list and the customer (at least in “CSV mode”) will have the same opening balance.

The components of the shop will be:

**Product:**

* Name – name of product (string)
* Price of product (double)

Product will be represented as a struct in C and declared at the start of the programming. In Java, it is a separate class.

In both implementations, a product is declared in the same manner: Product(name, price). In the Java implementation, the name and price are accessed outside the class by “getter methods”. There is no reason in either implementation to change the product name or price.

**Product Stock**

This represents the number of an individual product in stock. A product stock consists of the product (name and price) as well as the quantity in stock. Again, Product Stock will be represented as a struct in C as a class in Java. In the java implementation, a setter method is used to externally update the quantity in stock after each purchase.

The Product Stock data type (C) and class (Java) are both used to represent the item and quantity of the items in the customers order list. In this use, the quantity is the quantity of the item sought by the customer.

**Stock list/Shopping list**

In both implementations, the stock list and the customer shopping list is represented as an array of product stocks. There was no need in the Java Implementation to create a separate class.

In both the Java and C implementations, the **Customer** will have:

* Customer name (string)
* Customer budget (double)

Customer shopping list (implemented as an array composed of Product Stock struct/class)

The Customer will be represented by struct in C and as a class in Java. Note that an index will be required in the C implementation.

The **shop** itself:

* This shop an opening cash balance, a double(to be updated when each transaction is completed).
* The shop will have a **stock**. This will itself be a list of Product Stocks. This will be updated when each transaction is completed.

Shop will be represented as a struct in C and as a class in Java. Note that an index will be required in the C implementation.