# ICT373 Assignment 1 – Question 2

Callum Peel – 34217062 10/04/2022

Supplement Services – A Java program that tracks newspaper supplements details for customers.

## Requirements/Specification

### Description:

The program should keep track of the details relating to:

* Magazines
* Supplements
* Customers
* Paying customers
* Payment methods
* Cards
* Accounts

It should create the text for weekly and monthly emails respective to their paying status.

Paying customers should get a breakdown of their own supplements as well as the customers supplements that they are paying for.

### Assumptions

* + The user knows how to use command lines.
  + The user can input data.
  + The user speaks English and is literate.
  + If customers were to change their subscription midway through the month, then the monthly emails will be inaccurate.
  + The customers cannot change their subscriptions mid-month
  + Not expected to implement “time”
  + It is assumed that supplements that the customer is interested in are the supplements they are subscribed to.
  + Subscriptions are either valid or invalid for each whole month.
  + Along the same vein I have also implemented a formula for working out the monthly cost.
  + Up to date NetBeans*(8.2 or later)* has been installed on user’s machine.

## User Guide

### How to compile/run:

In the program directory navigate to src > SupplementServices

And double click on Main.java (below)

Text

Description automatically generated

Once NetBeans has loaded and the Application tab is selected click the run button*(circled in red below).*

Graphical user interface

Description automatically generated

### How to use:

Text

Description automatically generated

In the console below the program should be printing a menu in the console.

Entering a letter from A-F will choose a menu option, for example A will add a customer to the database.

Follow the prompts and enter data to build a database of customers or supplements, etc.

Text

Description automatically generated

Once you are finished with the program simply enter Q in the main menu to exit.(below)

**Text

Description automatically generated**

## Structure

### Outline:

In the Main class there are 2 global variables, a FrontEndConsole and a BackEnd.

This can easily be modified to hold an array of BackEnds and FrontEnds for multiple applications.

Not doing this will mean that user input handling will be mixed up in the same class that holds the information.

This seems messy, confusing, and not scalable.

There is a customer class and a Paying Customer class that inherits from customer.

I utilize polymorphism in building the email strings.

Similarly, a Parent class PaymentMethod and child classes for Cards and Accounts.

A magazine class will hold a list of Supplement objects.

I tried to think about the problem in terms of object and implement a solution that groups like-classes into parent/child classes.

Other classes that would have been nice to implement but were not in my MVP were Classes that delt with formatting, i.e., an ExpiryDate class to handle the formatting of dates (getters and setters for day, month, year, String builders for returning that data in a particular format etc.).

Limitations:

This program has no way of stopping excessive memory allocation.

This could be easily fixed by limiting the list’s size.

### Structure Chart:

## Diagram Description automatically generated

## Testing

### Strategy:

My strategy is to build a database automatically of customers, paying customers, supplements etc,.

Once there is already a database, I will then go through each menu option and test the functionality further.

I want to add incorrect values and see how the program handles it.

I want to add correct values and see that everything works as expected.

### Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Data | Expected Output | Result | Pass/Fail |
| 1 | Test case conversion works | a | “Please eneter a name: | “Please eneter a name: | Pass |
| 2 | Test that menu redisplays when inopoput is incorrect | k | menu | menu | Pass |
| 3 | See if program handles symbol input | # | “Invalid input. Enter Y or N.  Do you want to enter another customer? (Y or N)” | “Invalid input. Enter Y or N.  Do you want to enter another customer? (Y or N)” | Pass |
| 4 | Add a customer  Input customer data  Exit  Print customer list | a  Callum Peel  [Callum.epeel@gmail.com](mailto:Callum.epeel@gmail.com)  Supplement = Three  D | Callum Peel printed in customer list | Callum Peel printed in customer list N)” | Pass |
| 5 | Add a Paying Customer and print names of customers | B  Boris  [B@gmail.com](mailto:B@gmail.com)  Two  Exit  West Bank  1234 1234 1234 1234  12/03/2023  123  Tim  Phil  Exit  F | Boris added to customer list | Boris added to customer list | Pass |
| 6 | Add a Paying Customer then get monthly emails | B  Boris  [B@gmail.com](mailto:B@gmail.com)  Two  Exit  West Bank  1234 1234 1234 1234  12/03/2023  123  Tim  Phil  Exit  F | Monthly email:  Boris: Supplement 2  Tim’s and Phil’s supplement breakdown | Monthly email:  Boris: Supplement 2  Tim’s and Phil’s supplement breakdown | Pass |
| 7 | Delete a customer with correct name | D  C  Dom  D | Name removed from the list | Name removed from the list | Pass |
| 8 | Delete a customer with correct name | D  C  Hans  D | Customer not found | Customer not found | Pass |
| 9 | Delete a customer with case insensitive name pass for a name | D  C  dom  D | “Please enter something.  Please enter the name of the person:” | “Please enter something.  Please enter the name of the person:” | Pass |
| 10 | Check for new line entry on main menu | \n | Redisplay menu | Redisplay menu | Pass |
| 11 | Number entry when char is expected in main menu | 1 | Menu redisplayed | Menu redisplayed | Pass\* |
| 12 | Check for new line entry on change input | \n | “Incorrect input, please try again...  Please enter the coin value for the person:” | “Incorrect input, please try again...  Please enter the coin value for the person:” | Pass |

**Test Data:**

**Supplements:**

**One, 2.4**

**Two, 6.3**

**Three, 7**

**Four, 8**

**Five, 11**

**Six, 3.4**

**Regular customers:**

**“Callum", "callum@gmail.com", supplementList1**

**“Maddie", "Maddie@gmail.com", supplementList2**

**"Dom", "Dom@gmail.com", supplementList3**

**“Tim", "Tim@gmail.com", supplementList4**

**“Sally", "Sally@gmail.com", supplementList2**

**“Fin", "Fin@gmail.com", supplementList4**

**Customer Lists for Paying Customers:**

**customerList1 = Callum, Maddie, Dom**

**customerList2 = Callum, Dom, Tim**

**customerList3 = Dom, Tim, Sally**

**customerList4 = Sally, Fin, Matthew**

**Paying Customers:**

**"Matthew", "Matthew@gmail.com", supplementList1**

**new PaymentMethod("", new Card("Matthew", "1234 1234 1234 1234", "12/24", 232)), customerList1)**

**"Steven", "Steven@gmail.com", supplementList2),**

**new PaymentMethod("", new Card("Steven", "1234 4444 1234 2222", "10/24", 513)), customerList2)**

**Mark", "Mark@gmail.com", supplementList3),**

**new PaymentMethod("", new Card("Steven", "6666 4444 3333 2222", "11/25", 765)), customerList3)**

**"Phil", "Phil@gmail.com", supplementList4),**

**new PaymentMethod("", new Card("Phil", "3233 1313 1111 4344", "12/23", 748)), customerList3)**

### Testing Output:

1

Graphical user interface

Description automatically generated with low confidence

2

Text

Description automatically generated

3

Text

Description automatically generated

4.

Text

Description automatically generated

Text

Description automatically generated

5.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

6.

Text

Description automatically generated

7.

Text

Description automatically generated

Text

Description automatically generated

8.

Text

Description automatically generated

9.

A screenshot of a computer screen

Description automatically generated with medium confidence

10.

Text

Description automatically generated

11.

Text

Description automatically generated