David Villalobos
CSCI 200 - Section A
Clark Scholten
Colorado School of Mines

Expense Tracking System

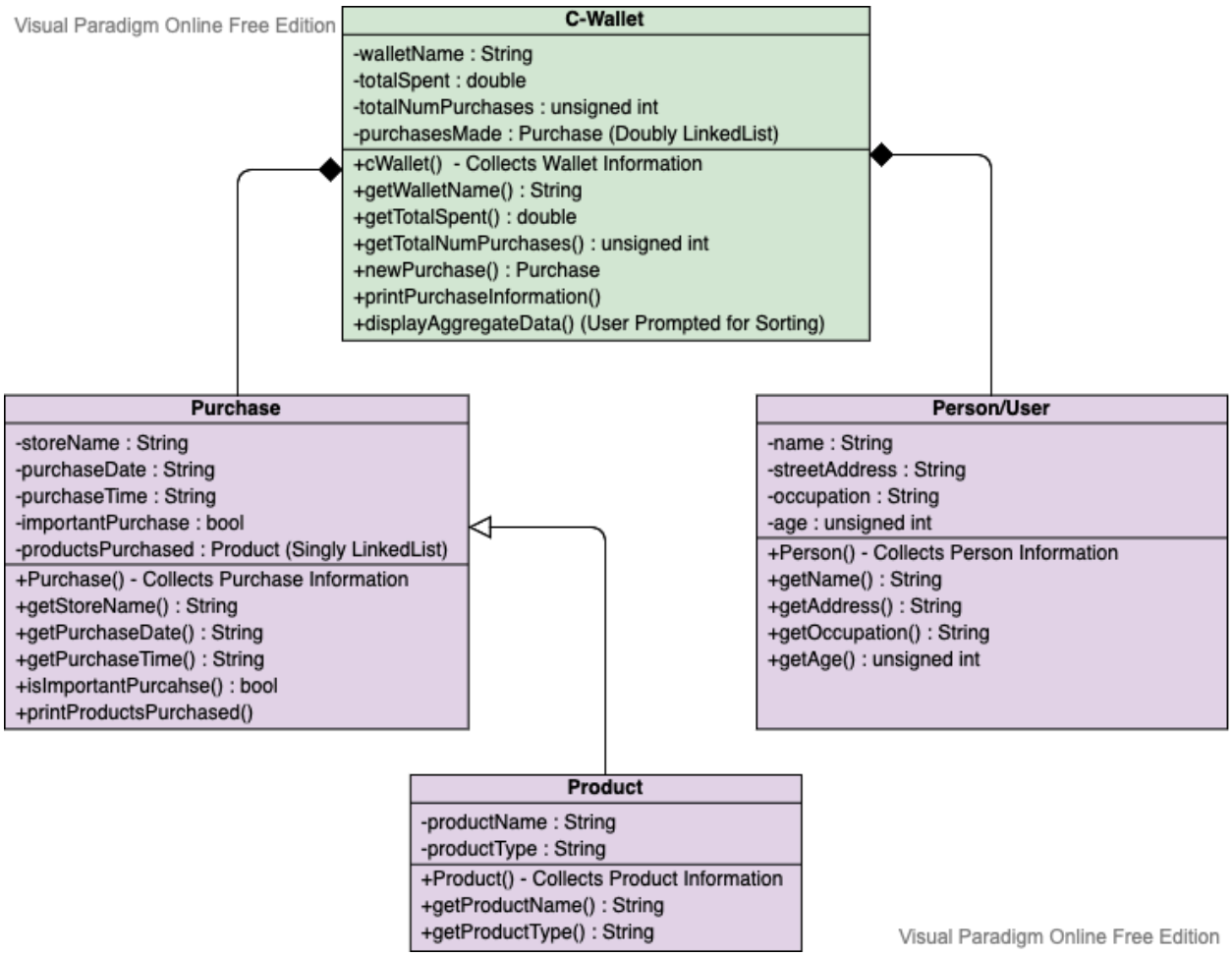
25th October 2022

OVERVIEW/PROBLEM DESCRIPTION

As college students, we are always trying to find a way to limit our spending and make smart money decisions. Unfortunately, our quick reactions to temptations make purchases hard to resist; this immediate response however can be reduced with our expense tracking system named C-Money. C-Money is a very fitting name because not only can you “see” the money spent, but you’ll be in a sea of money after implementing our system. Each time a purchase is made, you will go on this system and input a new purchase which will prompt you with information such as kind of purchase, product name, amount, store name, date, form of payment, if it was an important purchase, recurring purchase, and a few other questions. As this system is used, enough data will be collected to highlight poor spending habits and make the user aware of unnecessary purchases. As each purchase is collected, it will be logged internally through the use of text files and the user can ask the system for a copy of all purchases. All the purchases can be outputted for the user in a text file and displayed on the console with different sorting algorithms for example showing a specific store’s purchases. Other features include demonstrating the most common purchases, most expensive ones, most common stores, most unnecessary spending at a specific store, and other similar features. These reports will ensure that we as college students are aware of dangerous spending habits and develop us as conscious spenders.

DATA DESCRIPTION

UML Class Diagrams



UML Diagram Please Note Below

1. Each class constructor will be responsible for collecting user input and verifying it before assigning it to its instance variables. In short, each class is completely responsible for managing their own objects. For example, if the user prompts the C-Wallet class to print out all purchases, a static method in the class Purchase will be called to perform that action. This is so there are limited conflicts with access modifiers and to follow proper programming practice of encapsulation.
2. While this is not clearly demonstrated in the UML diagrams, each class will use destructors as necessary to ensure there are no memory leaks.
3. There are no setters in the classes as the constructor will be setting and verifying all input. Data input must not be modified once inputted as it's stored in several locations.

Data Structures

The 3 major data structures that will hold most of the information include a doubly linked list, a singly linked list, and a text file. The doubly linked list will be housed in the C-Wallet class and it will contain every single purchase made throughout the history of the wallet. We decided to use a doubly linked list since it would ensure that chronological order is maintained and the data can be manipulated in several ways to collect statistics about the purchases; this will be used when displaying the summary of purchases with the help of the methods in the Purchase class. This doubly linked list will be composed of Purchase objects which will allow us to store a lot of information in a single list which is very convenient and efficient. Next, the singly linked list will be stored in the Purchase class. Since each purchase usually has more than one product, the user will be given the opportunity to manually input each product purchased into this singly linked list. While it may seem complex to input every single item, this will be made optional to each user; it will also provide more insight into the overall purchases and ultimately be more effective as our system will provide the user with a more precise money management summary. When the user is ready to review all their data, they will ask C-Wallet to prompt a summary of their data. The summary will be provided in a text file format which will provide each transaction separated by store and item type. Hand-written sorting algorithms and output streams will be utilized in the creation of this output.

PROCEDURAL DESCRIPTION

The entire program will begin in the main with the creation of a C-Wallet object. Since we previously stated that all class constructors will be in charge of collecting and checking their input data, the program execution will follow the constructor's steps, then once all input for the Person object has been created, an overall menu will be displayed. This menu will provide options such as "Add new Purchase" and "Display All Purchase Stats". From here, when a new purchase is created, its constructor will take charge prompting for valid input and creating the new object. Each time, input will be validated using a try-catch by throwing proper exceptions.

Main creates C-Wallet ➔ C-Wallet Constructor Prompts New Wallet Information ➔ New Person is created and the Person constructor collects and validates Person ➔ Once C-Wallet object has been instantiated, menu will populate terminal ➔ User can select options such as "Create new purchase" or "Display Purchase Information" ➔ Creating a new purchase will call it's constructor and purchase information will be collected and verified ➔ From here it's just a loop in which the console is waiting for what the user may want to do next ➔ At the very end, a summary will populate the program and a text file will be create which will contain a summary of each purchase and some statistics while will help the user regulating excessive spending habits.

MAJOR CONCEPTS IMPLEMENTED

Classes ➜ Class Instances (Objects) ➜ Doubly LinkedList ➜ Singly LinkedList ➜ Try and Catch (Exception Handling) ➜ Exception Throwing ➜ File Output ➜ Output Formatting (iomanip) ➜ Accessor/Mutator Methods ➜ Constructors ➜ Pass By Reference & Pass by Pointer ➜ Private/Public Data Types.