Project Description Essay: Overview

The goal of this project is to make an expense tracker. In order to be effective, the expense tracker must have the ability to keep an accurate history of transactions entered, who the transactions belong to, transaction information, and a way for transactions to be entered. It has four fundamental elements, these being a system for keeping track of all metadata in single transactions, a system for keeping track of users, a main collection of all transactions, and an interface for the user to enter information into.

In order to achieve this goal, these elements had to be broken down into more manageable pieces. In this iteration of the task, these elements were separated into six different classes, namely Track Expense, Line Item, Unique Item, Budget, Category, and User. Track Expense is the class that contains the main method which runs the user interface and allows for a variety of user commands. It fills the role of allowing for transactions to be entered into the system. Line Item is a class that handles the default metadata of any transaction object, including instantiation time, and identification. Unique Item fills the role of specific transaction information. Together these two classes fill the role of transaction information system. Budget is the class that stores all the transactions made by the user. The Category Class references a subsection of the Budget for specific groups of transactions. Together these classes fill the role of keeping a history of transactions entered. The User class keeps a record of all users, and manages the storage, reading, and writing of the program files. This fills the role of tracking users, and also is a catchall for the file system.

In order to achieve functionality in transaction creation there needs to be a system for handling time. To achieve functionality in transaction records there needs to be a system for managing files. The Java language has built in tools for these two systems.

The in order to use time effectively, the first new tool to learn was Java.util.time.LocalDateTime . The LocalDateTime object contains all the data required for both the date (like era, year of era, month, day, and optionally weekday) and time (hour of day, minute, second, and nanosecond). The main utilities this provides to the project are instant time with the now method, allowing for the transactions to be organized chronologically and for every transaction to have a marked time of creation, allowing for the user to add a date to the transaction time.

In order to make the second a possibility, there needs to be a way for users to actually enter the time. This requires the LocalDateTimeFormatter. This object has a wide variety of optionality in how the time is implemented and can take a user string and parse it into a LocalDateTime. A date time format was chosen. Additionally, individual date and time formats as well as individual dates and times were created, so that a user could input the date, time or both, based on their knowledge and recollection of the transaction.

In order to save effectively, the information needed to be saved in a file at a specific location. java.nio.file.Files, java.nio.file.Path, java.io.BufferedReader, java.io.BufferedWriter, and others were implemented. These allow for the writing, reading, creation, and placement of files. Effectively utilizing these tools was not achieved in totality. Despite that, methods were implemented for creating files, saving files to a particular path, creating a txt file that contains a string of the information in the budget, and the necessary steps required to read that file and create an object based on the string read.

This implementation of the project has a great deal of room for improvement. Meeting the basic details of an expense tracker was barely achieved due to a lack of effective file management leading to a corruption of the expense trackers history. Information is saved to a file, and so are the users, but when the program is closed and opened again, this information is not always correctly retained.

After this fix is corrected, the budget and category need methods for the total and average that are updated after each entry is added. Then more complicated financial tools can be modeled, like routine, and expanding items.

In the outset, the goal was to simplify some elements of the personal wealth problem by tackling the challenge of keeping track of finances. Once the program effectively performs the full set goal functionality, the first step is in improving the quality of the interface from a text-based program to a GUI. User friendliness in finance tools is the first step to increasing financial literacy. This could ideally be done through mimicking other applications like the expense display tools standard with your banking app. A suite of graphic designs would also be necessary.

After the GUI is effectively created, security becomes the focus. As an expense tracker, the information is stored in a local file in plain text. All information would then need to be encrypted, and cybersecurity protocol would need to be learned and practiced. Then discussion of connection with finance applications can be discussed and coordinated, for something that can unite all finance applications available on your device, and promote an accurate, live picture of your financial health and goals.