Final Project Report

# Objective

At the begging of the project my object was to create a quick organizer with non-volatile memory to keep track of some financial transactions with a few friendly messages. I learned while working on this that starting a project with broad vague requirements can amount to frustration quickly, but I think I caught myself early enough to have my main ideas put on to it. I cannot say I did not wish I could have done more, I wanted to add a type of text suggestion based on comparing stored items name summed up character values to avoid misspelling but that went over my time constraints I really needed to make some clean up time.

# Design

My project has 3 main menus with several submenus, there are 8 classes; class Driver extends MessagesToUser, class Expenses extends CreateFile, class Menu extends Calculator, class WishList extends CreateFile, class CheckingAccount extends CreateFile,

The data flows from the menu to the validations made in the Calculator class then they are tested against other data instances in MessagestoUser to generate feedback, after that the data is sent to their respective classes initialized into objects and written to a text file.

# Output

I was happy with the overall look. It did what I expected with some bugs on the way but they got fixed eventually. The data expected will be a string item and a double value. The data is first taken as a string separately; then separated into a string name for the item; the value is sent to be tested in the Calculator class to a testForNumber method that recursively asks for the right format and returns it back to the class that requested it.

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \*

\*

WELCOME TO YOUR PERSONAL FINANCIAL ORGANIZER AND ADVISER

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter 1 for payday 2 for expenses 3 to quit: 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EXPENSES MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter 1 for entries 2 for view: 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ENTRY MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter 1 for expenses 2 for wishes: 1

Enter item name: ffff

Enter price: 5.66

Congratulations you have achieved your goal! It will be deleted from your wish list

This is it would reflect on your checking 3675.7900002002716

if you would like to make the purchase type Y(yes) to adjust your checking account: y

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter 1 for payday 2 for expenses 3 to quit: 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PAYDAY MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter 1 for deposit 2 for balance 3 for statement: 3

9.0 2020/12/11 -9.0

500.0 2020/12/11 491.0

-8.0 2020/12/11 483.0

-7.0 2020/12/11 476.0

This is the sample run. It does not reflect all the capabilities of the project I think I went a little overboard, but mainly it is asking for three different types of lists. The checking account only takes a value, it saves as a value, date, and the sum of previous values. Expenses asks for values, and for two lists an expense list and a wish list. The expense list records the values as an item and a value. It also automatically updates the checking account to reflect the expense; the wish list records an item and a value of something you wish to purchase it is sorted so that the least expensive item is at the top of the list. It is checked after every purchase and deposit to tell the user he can realize that wish. It is then deleted from the list and if the user feels he still wants to buy that item he can chose so that it is accounted in the checking list. There is also a list that keeps only unique items with their respective summed up values (that was tricky).

# Challenges

The most challenging part was making it non-volatile so I could retrieve the information. I am glad I decided to use that even though it was not necessary because it gave me a chance to really dive into the data structures needed, and see how they can be used in different ways for different types of situations and how the wrong structure can be a menace. I learned to let go instead of forcing things to work there are many structures for a reason.

# References

I used many references I truly believe that we are nothing but the reference of past experiences and guiding figures, <https://www.geeksforgeeks.org/sorting-a-hashmap-according-to-values/> this the only direct reference only because I was running out of time and I really needed to sort it to meet my requirements if I had more time I perhaps could have find a way to sorted on my own, but I think the solution it’s giving by this sample code was very good although I don’t like the single letter variables it makes it difficult to understand. I used other references like the professor’s lectures, the book, and of course the oracle documentation for the java classes I used.