Name: Derek Aguirre
Date: August 28, 2020
CS 3331 – Advanced Object-Oriented Programming – Fall 2020
Instructor: Daniel Mejia
Assignment: Programming Assignment 1

I confirm that the work of this assignment is completely my own. By turning in this assignment, I declare that I did not receive unauthorized assistance. Moreover, all deliverables including, but not limited to the source code, lab report and output files were written and produced by me alone.

## 1. Program Explanation

In this programming assignment I am tasked with programming a bank for a hypothetical company. This assignment requires reading and storing a csv file with user accounts, handling deposits, withdraws, money transfers, balance inquiries, and logging all transactions between users. Initially I was overwhelmed with where to start in the assignment, but I broke down the instructions and completed them in small parts to eventually complete the full assignment.

## 2. What did I learn?

I learned a lot from this assignment, it was more of a refresher but it helped me understand how to manipulate objects better which is something that previous CS courses did not delve into too much.

My solution can be improved in numerous ways, but I tried making it as functional as possible. Here are a few things I noticed that could be improved on while I was working on it:

- I should save a user's updated balance instead of saving their initial balance before making any transactions, then carrying it over to the next transaction
- Users should be able to authenticate better with a pin or a password
- Case sensitivity exception
- More efficient way to look up the users, especially for looking up a second user
- Minimum deposit/withdraw/sending amounts
- Program should not stop when an exception is caught
- Sending money needs to be more secure, need a better solution to verifying if recipient exists (I can ask for full name on top of the account number.)
- I do not use all of the information given in the .csv file

I have a couple solutions for some issues I was having when solving this problem. I believe for the issue I had with looking up multiple users at the same time, I can use a hash table to store the users for a second request to save on run time. As for the authentication of users, I also could have added more questions and authentication methods whenever a user is logging in.

In total, it took me roughly 4 days to complete this assignment, most of the time spent working on the assignment was figuring out the logic on how to look up a second person while also still keeping track of the first person in memory.

### 3. Solution Design

I started with the skeleton and worked my way through the problem in pieces. After the skeleton was completed I focused on reading the file and storing it. Now that the csv file was stored in my program I worked on prompts for the users by utilizing a switch to handle a user inputting in their account number to log in along with choosing a number associated with a service they needed. After some exception handling with the user inputs it was time to work on the functions that deal with depositing, withdrawing, and transferring money.
I used an ArrayList because it is easy to work with and has a multitude of methods that can be utilized over an array. However, I am considering using a hash table as well for a quicker lookup when the program verifies the existence of a second person or above. During the process, I assumed that the users will be making transactions only on a checking account, so I did not specify if they need to use a checking or savings. I also infinite funds to deposit, and that they are not going to abuse sending $0.00 to each other.

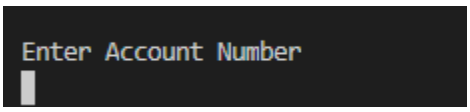
### 4. Testing

My testing method would be considered white-box testing since I developed the program up to completion, testing included condition testing for user input, and loop testing for correct data retrieval. I tested my program in different ways depending on what needed to be tested. For accessing the users, I printed out the account number that the user inputted along with the account numbers that were retrieved from the ArrayList using the enhanced for loop. The purpose of the printing is to verify that the information was retrieved properly and that the account number that a user inputted matched the account number in the system, if everything worked properly, the program would log the user in. The other main way of testing the program was during all the transactions that a user must make. I selected a few users to test, the first, middle, and last were the main ones to test for the purpose of verifying that retrieval from the data structure was working. I entered the account number of a user and tested depositing, withdrawing and an inquiry of their balance. The most testing involved sending money from one user to another. For money sending, I would log in with a random user and send money to a second random user. Afterwards, I logged in with the second user to send money to the first user.

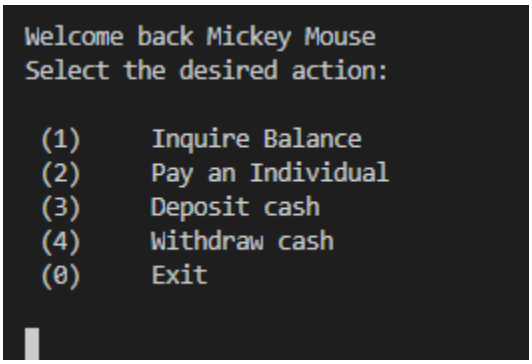
I tested my user input code greatly and made sure to break the program by any means necessary, as a result my exceptions take care of all the possible user error that I can think of. I believe that the tests on my solutions are enough that they can hold up solidly with regular use. However, I feel that I could have tested more efficiently, especially during the money transfer section of the program. I could have tested less on that section because as long as the first, middle, and last accounts are in working order then it is safe to assume that the accounts in between are functioning properly.

### 5. Test results

I will include the results of my testing, along with a brief explanation of what the console shows:

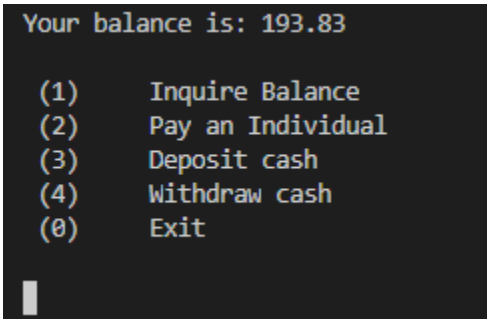



The user is greeted with a prompt for entering their account number at launch. If the user inputs an account number that matches with one stored in the data structure then it continues onwards, otherwise, it stops the program.



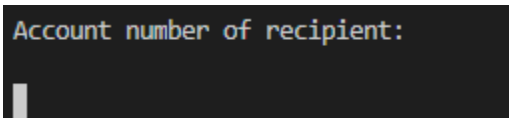

The user is then shown a menu of what options they have for their account.

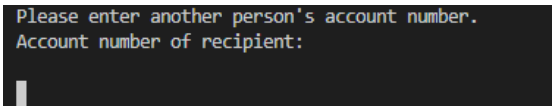
---

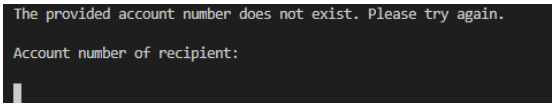


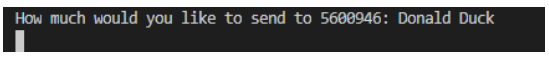

Entering in “1” will provide the user’s balance.

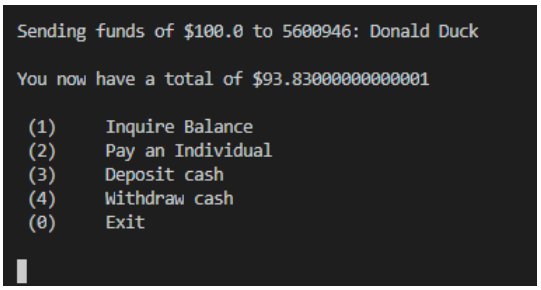
---




Entering in “2” will prompt the user to provide the account number of whomever is receiving the money.

If the user enters in their own account number, or an account number that does not exist, the user will be asked to try again

If the user enters correctly, they will be allowed to send money that does not exceed their own balance.

After the transaction is complete, the program will report how much money the user has after the transaction and will be brought back to the menu.

```
How much would you like to deposit?
```

Entering in “3” will ask the user how much money they would like to deposit.

```
Depositing $500.74
You now have a total of 594.57
 (1)     Inquire Balance
 (2)     Pay an Individual
 (3)     Deposit cash
 (4)     Withdraw cash
 (0)     Exit
```

After entering a non-negative amount, it will report back to the user with an updated amount and brings the user back to the menu.

```
How much would you like to withdraw?
```

Entering in “4” will ask the user how much money they would like to withdraw.

```
Withdrawing $350.64
You now have a total of 243.93000000000006
 (1)     Inquire Balance
 (2)     Pay an Individual
 (3)     Deposit cash
 (4)     Withdraw cash
 (0)     Exit
```

If the user enters an amount that does not exceed their account balance, it will report back the user’s updated account balance and bring them back to the menu.

```
Successfully logged out.
```

Entering in “0” prompts the user of a successful logout and exits the program.

Below is a screenshot of the log after this test of the program.

```
Mickey Mouse has logged in.

Mickey Mouse has requested to see their balance of $193.83

Mickey Mouse has transferred $100.0 to Donald Duck.

Mickey Mouse now has 93.83000000000001 from a balance of 193.83

Donald Duck now has 1713.96 from a balance of 1613.96

Mickey Mouse with an initial balance of $93.83000000000001 deposited $500.74 totaling $594.57

Mickey Mouse with an initial balance of $594.57 withdrew $350.64 totaling $243.93000000000006

Mickey Mouse has logged out.
```