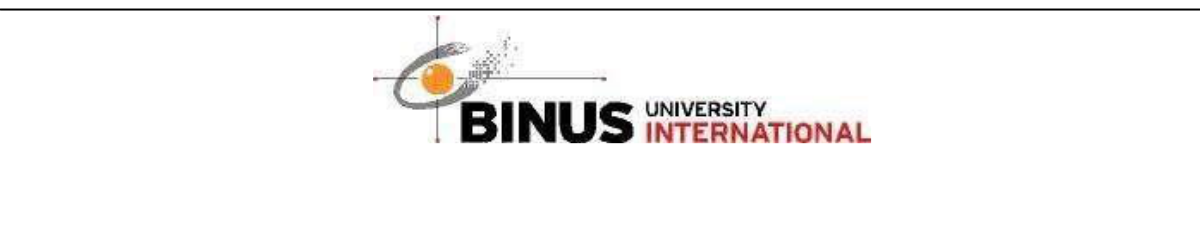
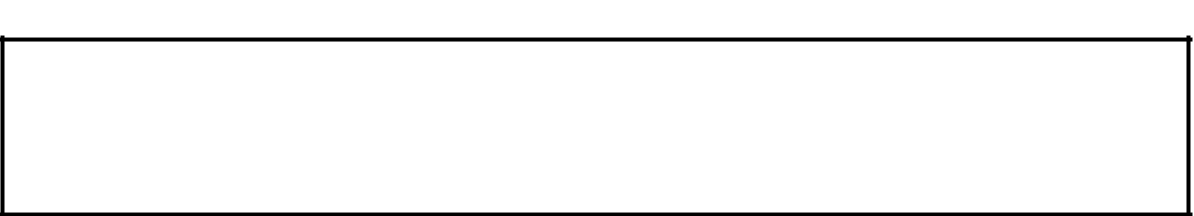
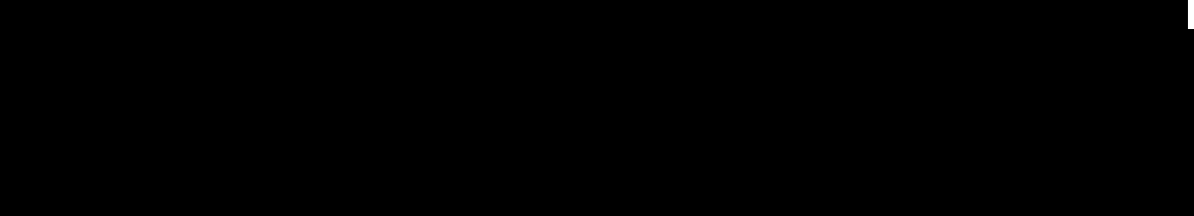
Even Semester (2024)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Individual Work)**

**Student Information**:

**Surname**

**Given Names**

**Student ID Number**

1.

Alvin

Nathaniel

2440042430

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Code** | **:** COMP6056 | **Course Name** | **:** Program Design Methods |
| **Class** | **:** L1AC | **Name of Lecturer(s) :** Jude Joseph Lamug Martinez | |
| **Major** | **:** Computer Science |  |  |

**Title of Assignment** : Simple Banking System

**Type of Assignment :** Final Project

**Submission Pattern**

**Due Date** **:** 13-01-2021 **Submission Date** **:**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Nathaniel Alvin

**“Simple Banking System”**

**Name : Nathaniel Alvin**

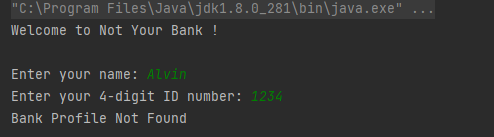
**ID** **: 2440042430**

1. **Program Description**

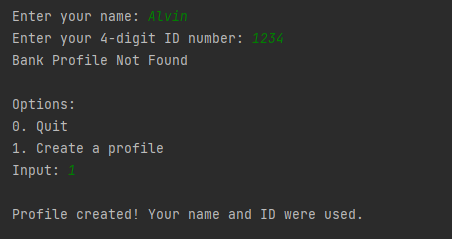
This application as the name suggest, is a banking program that have all the common actions that all banks have. In this application, the user can create multiple Profiles and Accounts which are saved on a text file. It is a command line application.

1. **How the program works**

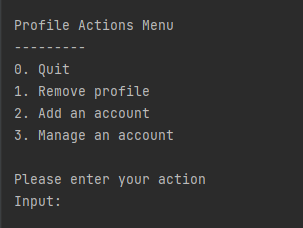
First of all, there will be a prompt asking for our name and ID number. This will be used to create a unique profile.



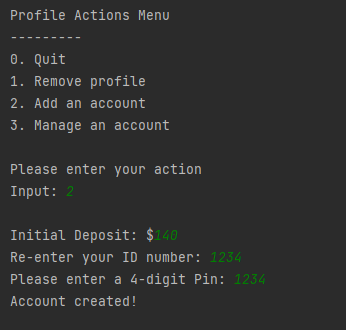
Next, it will check if the Profile exist. If the profile doesn’t exist, it will create a new profile with the name and ID number that has been inputted before. The user then have a couple of choices. If they are a new user, they can only create a profile.



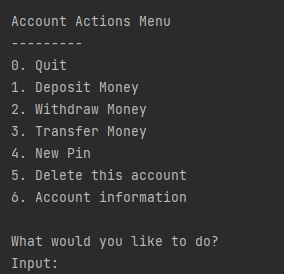
From there the user can choose on what to do. They can remove this profile, they can add an account, or they can manage an account if they already have at least one account.



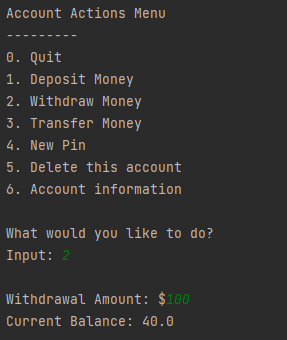
Here the user usually adds a new account to the profile. This account like all other account have an account number, initial balance, and pin. The user needs to input the balance and pin, while the account number is created automatically.



Then the Account actions menu pop up. Here there are some actions that the user can do. Deposit, withdraw, and transfer is some of the example. The user also have the ability to update the pin of this account or to even delete the account.



In this example, the user choose to withdraw some money from the account.



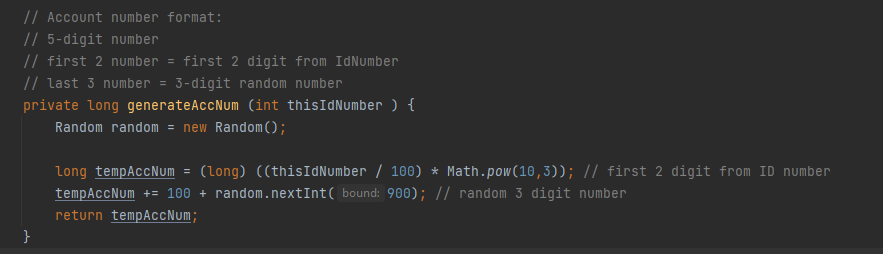
1. **Lessons that have been learned**

From this project, I now understand deeper about the way java works and its workflow. Although this project is far from perfect, I feel like it has taught me a lot on how OOP works.

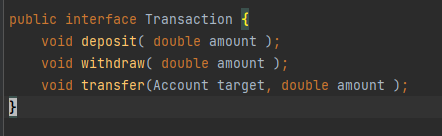
**IV.** **Code explanation**

There is not much algorithm used in this project. All of them seems to be self-explanatory.

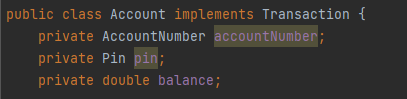
But here are some functions that needs a little bit more explanation.

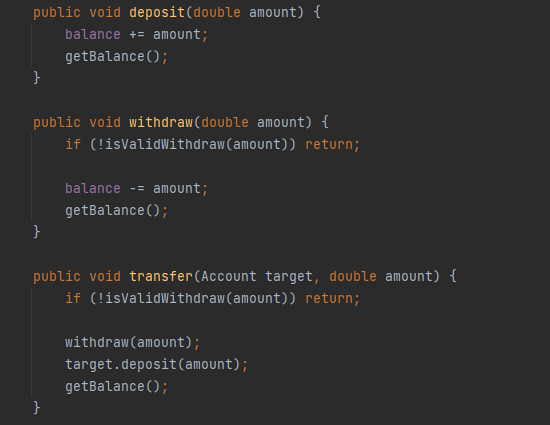


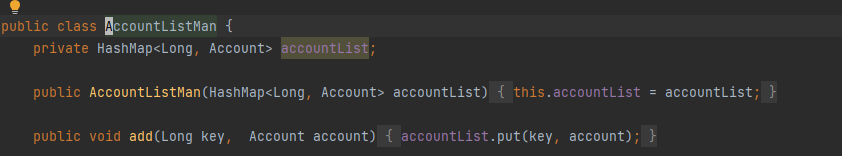
This is a method to generate an account number. I use the long data type to ensure the numbers will not be limited as I was worried integer data type will not be enough. This method take the ID number inputted by the user. So the format that I want for the account number is as follows. It will be a 5 digit number. The first 2 digits will be the first 2 digit from the ID number. And the last 3 digits will be a randomly generated number. So in this method, I created a temporary account number. It will divide the ID number by 100 and then add three zeros in the back by multiplying it with math.pow 10^3. Then the random number is added. The temporary account number is added with a random 3 digit number as shown. Thus a 5 digit number is produced as desired.



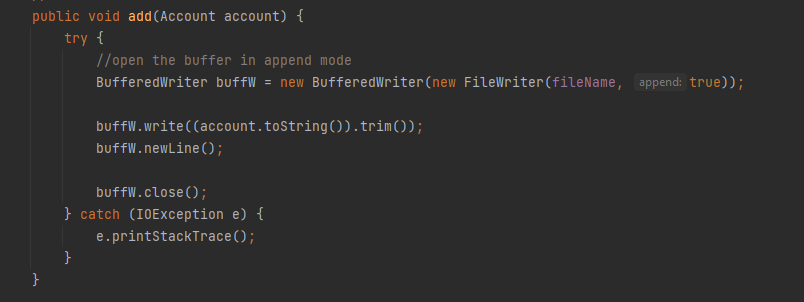
This is a transaction interface. This contains the methods for the accounts. Thus the Account class implements this transaction interface as shown below.





All the accounts are saved inside an account.txt file. They are saved in a hashmap as shown below. The hashmap contains the account number and the account object. 

To store an account inside a file, bufferwriter is used.



To load the account from the file, bufferedreader is used.



This is true for both account and profile. They are both saved inside a file. And the same technology is used.

**V.** **Project Link + Video demo**

https://github.com/nathaniel-alvin/Final-Java-OOP