Design Document for Console ATM

Name: Luong Trac Duc Anh

Student ID: 103488117

Introduction

This document is used to demonstrate my program for the High Distinction task and how it implements its functionalities. My program is an ATM console that can be used as a machine to insert your credit card and make transactions, including check balance, place a deposit, make a withdrawal, and perform the transfer (within accounts in the bank).

Program Functionalities

The ATM program is structured by creating interfaces that are in correlation with the primary features displayed in the Secure Menu. Each interface will contain a method that can later be used in various parts of the program since the program has features that require the integration of many methods from different interfaces.

Login

On initial run time, the main menu will appear with 2 options – Insert Credit Card or Exit. When inserting a card, you will enter the card number and pin code, which will then be verified via the CheckPassword() method. You have a total of 3 tries before your account status is changed to locked (IsLocked = true). If the details you entered are correct, then the Secure Menu will show up, where you can do transactions with your account balance.

Exit

Breaks from the program and terminates the process, returning an exit code to the system.

Check Balance

This feature will print out the account's current amount of money, using the format method that is declared in the Utility class.

Place Deposit

This feature allows you to insert money into the ATM and your account balance will also be incremented by that amount of money. A guard clause is used here, for example, you cannot deposit a number that is equal to or less than 0, or you cannot deposit an odd number (has to be a multiplier of 10). This feature is necessary as, in real life, banknotes will have a set of currency, and certain odd numbers are unobtainable. Because of this, I have also added the PreviewBankNotes() method in the LTDAbankATM class. This method declares 3 integer values

of 10, 20, and 50 like values on the Australian banknotes. Each value will be the remainder of the previous larger value, except for the biggest value (50), in which the previous remainder is 0.

Sample code snippet:

```
int fifty = (int)amount / 50;
int twenty = ((int)amount % 50) / 20;
int ten = ((int)amount % 20) / 10;
```

After that, you can get a preview of which notes you have inserted and confirm whether you want to continue with the transaction or not. Finally, the transaction will be recorded for later view.

Make Withdrawal

The option allows you to withdraw money from your bank account. First, you have to insert a number. Another guard clause will be used, to eliminate errors from the program. For example, the entered number cannot be less than 0, or it has to be a sufficient amount within the money balance that you have. The withdrawal will also be added to your transaction list.

Perform Transfer

This feature allows you to send money between bank accounts that are listed in the LTDA banking system (accounts that are created on initial runtime). You will have to enter the recipient's account number, name, and amount of transfer. If any of these inputs are incorrect, you can re-enter at your convenience. The transaction will be recorded and added to the transaction list, which you can later view in the secure menu. There are common rules that you have to follow, for example, the account that you use to send money cannot be the recipient itself or the transaction will be blocked if the recipient's account is blocked at transfer time.

View Transaction

After executing many transactions, the user would want to view their transaction history, which will be listed in the "Transactions" option in the secure menu. A table of 5 columns will appear, which displays the type of transfer, the transfer sender and recipient, the amount of money for each transaction, and the date (which will be recorded in real-time using the DateTime struct).

To create this table, use strings of hyphens "-" with Console.WriteLine (or Console.Write) method will be time-consuming and prone to errors. Therefore, I have used the ConsolesTable package to create properly aligned tables for easier viewing. The command for installation is as follows:

- dotnet add package ConsoleTables –version 2.4.2

```
PS C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM> dotnet add package ConsoleTables --version 2.4.2
  Determining projects to restore...
  Writing C:\Users\Pro\AppData\Local\Temp\tmpB3BF.tmp
info : Adding PackageReference for package 'ConsoleTables' into project 'C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\ATM.csproj'.
info : Restoring packages for C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\ATM.csproj...
info :   GET https://api.nuget.org/v3-flatcontainer/consoletables/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/consoletables/index.json 831ms
info :   GET https://api.nuget.org/v3-flatcontainer/consoletables/2.4.2/consoletables.2.4.2.nupkg
info :   OK https://api.nuget.org/v3-flatcontainer/consoletables/2.4.2/consoletables.2.4.2.nupkg 52ms
info : Installed ConsoleTables 2.4.2 from https://api.nuget.org/v3/index.json with content hash OSRFOWYCgFQORfAMof2DsGEJHI2YKquCA4X26kKyuG+MG+/2R3NRgRVEqXDMU7
eVP69MGDzb1eg4i3/APlIfQg==.
info : Package 'ConsoleTables' is compatible with all the specified frameworks in project 'C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\ATM.c
sproj'.
info : PackageReference for package 'ConsoleTables' version '2.4.2' added to file 'C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\ATM.csproj'.
info : Writing assets file to disk. Path: C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\obj\project.assets.json
log  : Restored C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM\ATM.csproj (in 2.13 sec).
PS C:\Users\Pro\Documents\Swinburne\COS20007\Tasks\6.6HD\ATM\ATM>
```

Figure 1: Adding ConsoleTables package

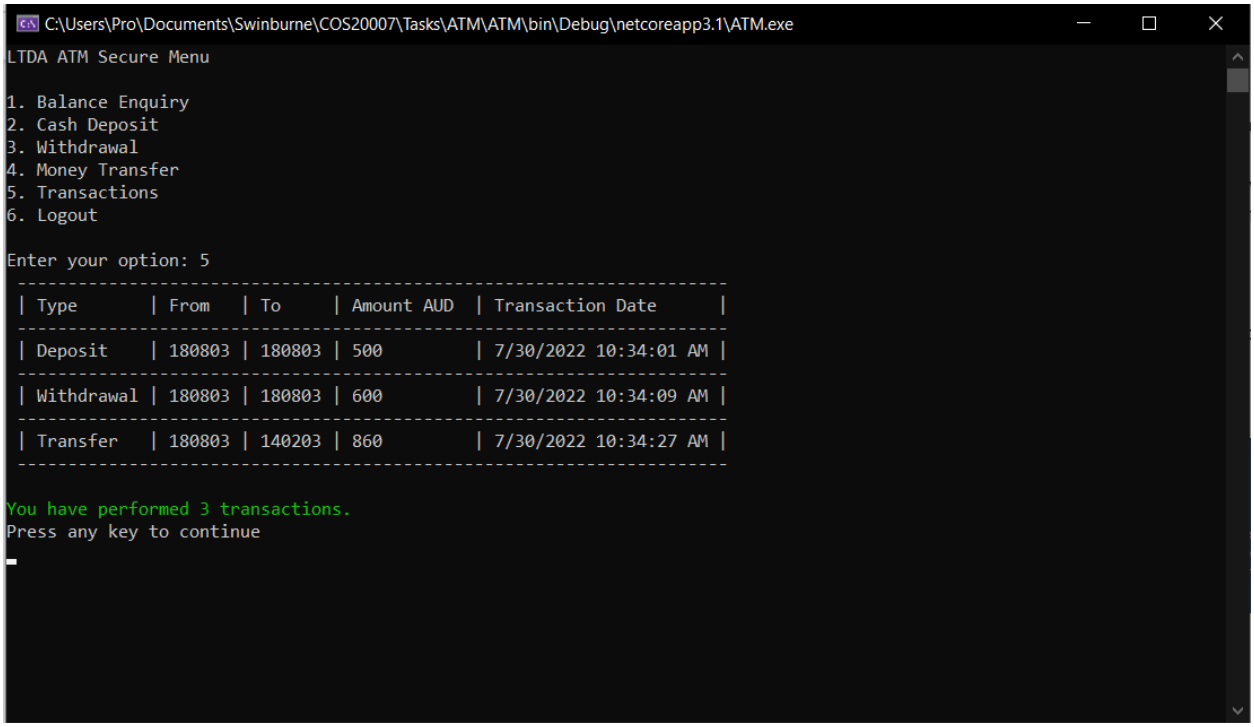

Figure 2: Sample output

Logout

This function will log you out of your bank account and return you to the Main Screen where you can enter another account or exit the program.

Error validation with Utility Class

There are 3 validation methods used to parse in a hidden console input (the one used for your PIN input), a decimal input, and an integer input. If the input is incorrect, the program will return an error and print it out to the console.

Animation

An animation was implemented in the program via the printDotAnimation() method in the Utility class, which prints a sequence of 10 dots every time the user logins or exits the program.