James Zafiri

03/10/23

CS6003A

Module 08: Final Project

Algorithm for Banking Program

Before starting with the Bank Manager class/main program, I will complete all the other classes that will be used in it first (Account, Bank, Bank Utility, and Coin Collector).

I will begin with the coin collector class since it didn’t have a lot to it. For this class, I will create one object for the total and set it to zero. I will then use the parse Change method to loop through a string that the user passes through, and depending on the character, add its assigned amount to the total.

I will then use the bank utility class to ask users for strings and positive numbers. The method that asks for positive numbers will make sure it isn’t negative and let the user know it can’t be if they try to. I will also have a method that creates a random integer in between a maximum and minimum value, as well as a method to convert dollars into cents by multiplying a float by 100. The given is numeric method is in here as well and that makes sure the user’s input is a number.

I will then work on the account class which will have a lot of useful methods. It will contain all the attributes for an account owner (number, first/last name, ssn, pin, and balance). There will be getter and setter methods for all these attributes.

In addition to this, there will be a method to deposit money and one to withdraw money into an account. The deposit takes an input from the user and adds it to the balance, while the withdraw takes an input and then makes sure there is enough money before taking it out of the balance. It will let the user know if there are no funds. There will also be a method to see if the pin a user inputs is equal to what is on file for that account. Also, the to string method will go through each line of a user’s account info and print that out using f-strings.

There will also be a method to create a random account number as well as a random pin that use boundaries to make sure it is the right number of digits. The method to change a pin will let the user do so if the one they enter is 4 digits and they enter it correctly twice (matches up). Lastly, the atm method will take an amount from the user and allow them to withdraw in 5s, 10s, or 20s if they enter a multiple of 5 that is less than 1000.

The bank class will start with an empty array of accounts and a constant that is a limit to for the number of accounts. There will be a method that adds accounts to the array, and this will also make sure there is no duplicate account numbers. It will add an account if there are spots available. The method that will remove accounts from the bank searches through the array to find the account and then removes it if is there. Also, there will be a find account method that returns the account if the number is in the array. Lastly, the add monthly interest method will ask the user for an interest amount and then add that to each account based on their balance.

It is now time for the bank manager class. I will first import all the other classes I created so I can use their methods. I will then create an object of the bank utility class so I can use it throughout the class.

The bank manager class will have an object for the menu that prints out all the options for the user, and no parameters for the constructor.

There will be a method that prompts the user for their account number and pin which will be used throughout the program. This will take two inputs from the user (acct num and pin) and test if they are valid. For the account number it will use the find account method from the bank class to make sure that the number is in the array. Then, if the pin matches with what is on file, it will return the account.

Now the main method will get to work. I will create an object for the bank class to use throughout the method, and one for the bank manager class so I can call the menu. I will then create a while loop that display the menu until the user decides to exit the program. This means after every transaction (valid or not) the menu will keep popping up. I will use exception handling to make sure the inputs are valid.

Lastly, using if and elif statements, I will create a scenario for each option (1 – 11) of the menu depending on what selection the user inputs. Each one (except for opening an account or adding interest) will ask the user for their account info and then proceed. Each different transaction will use methods from the various classes that I have described. When the user is done, they will enter 11 and exit the program.