Nate Idica

CS-273-OL: Data Structures

Professor Pete Tucker

12 June 2019

Homework 5 Write-Up

**PROJECT SPECIFICATION:**

The banking project and the program behind this project are aimed to represent what it would feel like to do regular transactions at a local bank. The program will keep record of the users or customers that have an account with the bank, or add them if their name is not recognized in their system and set them up with a new checking or savings account and will ask for their personal information such as name, phone number, age, address, and whether they are a senior, adult, or student. The software will also keep track of the user’s accounts at the bank, so for instance, every transaction of withdrawal and deposit into such accounts and adding interest onto the current balance in such accounts, as what normal banks do.

If a person is new and their name is not recognized when entered into the system, the system will assign the user an account ID number and customer ID number to verify that it is the user and not someone else. Other ideas that are associated with this software are the idea of overdraft penalties, unusual fees, processing transactions, setting the new amount of the account after withdrawal or deposit of a desired amount of money, and allow the user to have the accounts they have with the bank listed.

**USE CASES:**

#1:

* **SYSTEM:** Prompt user to choose from the 5 options posted on the menu.
* **USER:** Enters command that prompts the program to add a new account.
* **SYSTEM:** Prompts user to enter a name.
* **USER:** Enter their name.
* **SYSTEM:** Prompts user to select what account they are trying to access: savings or checking.
* **USER:** Enters command for a checking account.
* **SYSTEM:** Tells the user that they are new to the bank due to no previous record of their name, so prompts user to enter more information. Asks user to enter their address.
* **USER:** Enters their address.
* **SYSTEM:** Prompts user to add their telephone number.
* **USER:** Adds telephone number.
* **SYSTEM:** Asks the customer if they are either a senior, student, or adult.
* **USER:** Enters student.
* **SYSTEM:** Creates a checking account for the student. Asks the user if they would like another transaction.
* **USER:** Commands the program to say no.
* **SYSTEM:** Displays a thank you or goodbye message and program exits.

#2:

* **SYSTEM:** Prompts user to choose from the 5 options posted on the menu.
* **USER:** Enters the command to make a withdrawal.
* **SYSTEM:** Asks user to enter their account number.
* **USER:** Enters account number.
* **SYSTEM:** Looks for account and asks user to enter amount they would like to withdraw from the account.
* **USER:** Enters the amount that they would like to withdraw.
* **SYSTEM:** Subtracts amount that the user entered from the account specified from the account number entered. Asks the user if they would like another transaction.
* **USER:** Exits the program.
* **SYSTEM:** Displays a thank you or goodbye message and program exits.

#3:

* **SYSTEM:** Prompts user to choose from the 5 options posted on the menu.
* **USER:** Decides to not proceed with banking program and exits the program.
* **SYSTEM:** Displays a thank you or goodbye message and program exits.

#4:

* **SYSTEM:** Prompts user to choose from the 5 options posted on the menu.
* **USER:** Enters the command to make a deposit.
* **SYSTEM:** Asks user to enter their account number.
* **USER:** Enters account number.
* **SYSTEM:** Looks for account and asks user to enter amount they would like to deposit into the account.
* **USER:** Enters the amount that they would like to deposit.
* **SYSTEM:** Adds amount that the user entered from the account specified from the account number entered. Asks the user if they would like another transaction.
* **USER:** Exits the program.
* **SYSTEM:** Displays a thank you or goodbye message and program exits.

#5:

* **SYSTEM:** Prompts user to choose from the 5 options posted on the menu.
* **USER:** Enters the command to list the accounts associated with their name.
* **SYSTEM:** Prompts user to enter their name.
* **USER:** Enters their name.
* **SYSTEM:** Searches for their name, then displays all of the accounts associated with the name that was entered in the system. After a moment, the system asks the user if they would like another transaction.
* **USER:** Exits the program.
* **SYSTEM:** Displays a thank you or goodbye message and program exits.

**UML DIAGRAMS:**

A close up of text on a white background

Description automatically generated

**PSEUDOCODE:**

**Add\_Account in Banking\_Application.cpp:**

1. Asks user for the required information (name, telephone number, age, address, and if student, adult, or senior) to make a new account with the bank.
2. Examine the new information that was entered.
3. Creates new customer by calling the constructor.
4. Assigns the new customer to specific account ID and customer ID numbers to verify that it is the correct user.

**make\_deposit() in Bank.h and Banking\_Application.cpp:**

1. Asks the user to input their account and customer ID numbers.
2. Looks through information that was entered by the user.
3. If name is recognized by system, then asks how much user may want to deposit. If not recognized by the system, it returns add\_account, runs through processes to add new account, and then asks user how much they would like to deposit.
4. Uses deposit() function.

**make\_withdrawal() in Banking\_Application.cpp:**

1. Asks the user to input their account and customer ID numbers.
2. Looks through the information that was entered by the user.
3. If name is recognized by the system, then asks how much user may want to withdraw from the account.
4. Uses the withdrawal() function.

**Overloaded add\_account() in Bank.h:**

* **IF THE USER IS NEW (PUBLIC):**
  + Makes the new account by calling the constructor.
  + Assigns the customer a new account and customer ID number.
  + Prompts user if they would like to open a checking or savings account.
  + Prompts user to enter all information needed to make new account.
  + Goes through user’s information.
  + Set’s balance of account to 0, since new account.
* **IF THE USER ALREADY EXISTS(PRIVATE):**
  + Asks for user’s information such as name, account number, and type of account.
  + Reads and gets user’s information.
  + Prompts the user if they would like a checking or saving’s account.
  + Uses get\_account() function that is connected to the user to add the new account.
  + Sets the new account number for new checking or saving’s account created.
  + Sets the new account balance to 0, since new account.

**get\_account() in Bank.h:**

1. Obtains customer number.
2. Software searches through the vector of accounts to find the account that is synched or connected to the customer number obtained.
3. Displays the account.

**BANK DATA STORAGE DESCRIPTION:**

Account numbers and customer identification numbers will be generated by the use of variables and incrementing the functions add\_acount and Add\_Account whenever it is called to be used. To keep the account number and identification numbers different, the account number will be 1000 higher than the customer ID number, which will then be stored into vectors in the account vector. The accounts will be linked to the customer by the specifications entered and if these specifications that are entered do not match what was stored in the vector, then this user cannot access the information and the account itself. Transactions will be linked to customers through their ID number that is asked upon as a specification for a specific transaction a customer may want to do.