Elisha Parslow

CS 273

HW04 Banking I

**Project Specification:**

The banking project is meant to simulate common actions one would take at a local bank. The program should open up to a menu where the user gets to select the options for which they would like to take: add\_account(), list\_account(), make\_desposite(), make\_withdrawl() and exit. The user can create an account which will begin by asking them their name. The user will then select if they would like to open up a checking or savings account which are classes that inherit their properties from the base class account. If the name entered is new to the program, it will begin to as them questions about their identity; age, phone number, address and assign them a customer number. It will also ask them what type of customer they are: senior, adult or student. These are all classes that inherit from the base class customer and have polymorphic attributes and inherit customer functions.

After the user has created their account, they now can access the other options in the menu to list account and make a deposit. The list account function will permit the user to view their accounts after they enter a corresponding name to that accounts. The make deposit function will ask the user to enter their account ID which was created in the account class through the add account option. The user then asked to input the amount they would like to deposit and it is added to that account. The withdraw option does the same type of activity except it removes money from the respective account.

After going through the process of each menu item, the user is always asked if they would like to exit the program. The user then has the option to exit or return to the main menu to go a new menu option or complete the same option as previous.

**Use Cases:**

**#1:**

            System prompts user to a menu of 5 options.

User enters command to add an account.

                                System prompts user to enter a name.

User enters a name.

                                System prompts user to select type of account to build: checking or savings

User enters command for savings.

System tells user that they are new user. Prompts user for more information, first address.

User enters address.

System prompts user for telephone number.

User enters telephone number.

System prompts user for age.

User enters age.

System prompts user to enter type of customer: student, adult or senior.

User selects senior.

System creates savings account for senior user. Asks user if they would like to perform another transaction.

User enters n.

System displays goodbye message and program exits.

**#2,3:**

System prompts user to select from the menu of 5 options.

User enters command to list accounts

System prompts user for name.

User enters name.

System searches and displays total number of accounts for user entered name. System asks user if they want to perform another transaction or exit program.

User enters to perform another transaction.

System returns to main menu and prompts user to input an option from the 5 on the menu.

User enters to make deposit.

System prompts user for account ID.

User enters account ID.

System prompts for amount the user would like to deposit.

User enters the amount to be deposited.

System deposits to account and asks user if they would like to complete another transaction.

User enters to perform another transaction.

System returns to main menu and prompts user to input an optio from the 5 on the menu.

User enters command to list accounts.

System prompts user for name.

User enters name.

System searches and displays total number of accounts for user entered name. System asks user if they want to perform another transaction or exit program.

User enters to leave program.

System displays goodbye message and exits program.

**#4**

System prompts user to select from the menu of 5 options.

User enters command to make withdraw.

System prompts user for account ID.

User enters account ID.

System prompts for amount to withdraw.

User enters amount to withdraw.

System checks for account and withdraws user entered amount from that account. System prompts user if user wants to perform another transaction.

User enters to exit program.

System displays goodbye message and program exits.

**#5**

System prompts user to select from the menu of 5 options.

User enters command to exit program.

System displays goodbye message and exits program.

**Pseudocode**:

Add\_Account(); in Banking\_Application.cpp

1. Display menu and prompt for command.
2. Read command “0” to access Add\_Account().
3. Display enter name message to user and prompt user for command.
4. Read the command.
5. Display enter type of account message to user and prompt user for another command.
6. Read the command.
7. Call read\_int function
8. If command is ‘0’, assign string to correlating option.
9. Else, assign string to the second option.
10. Declare a pointer type of Account to a function taking the first read command and string.
11. If the string is NULL, display appropriate message for new user information to create an account.
12. Display menu\_string to prompt user for type of customer command.
13. Read in command.
14. If command is ‘0’, assign cust\_type\_str to adult.
15. Else if command is ‘1’, assign cust\_type\_string to senior.
16. Else assign cust\_type\_str to student and set acct variable to add\_account function in Bank.h
17. If acct is created, display prompt to user’s account ID
18. Else display prompt that account could not be created

Make\_deposit(); in Bank.h and Banking\_Application.cpp

1. Else return add\_account
2. Create function of Account type with parameters name, address, telephone and age
3. Delcare pointer of Customer type
4. Push cust onto the stack for customers
5. Return add\_account
6. Create void function to make deposit and void function to make withdraw

Make\_withdraw(); in Banking\_Application.cpp

add\_account(); overloaded in Bank.h

get\_account() in Bank.h

1. Declare vector of ints with name parameter
2. Return function to find accounts with name parameter ‘
3. Declare pointer of Account type with int parameter
4. For size of account, if account of I is less than get\_account, equal to int parameter passed through
5. Return accounts at i
6. Return NULL

**Bank Data Storage Description:**

The account numbers and customer ID numbers will be generated by creating variables for each that will be incremented every time the add\_account and Add\_Account functions are used. Account number will start at two decimal points higher than customer ID to keep the numbers differentiated. The numbers will be stored into vectors in the account vector. The accounts will be linked to customers based on the parameters entered. If both do not match what is in the vector, then the customer cannot access that account. Transactions will be linked to customers through the customer ID number that is taken in as a parameter for the transaction functionality.

**UML Diagrams:**

