Nate Esrey

HW 5

Requirement Specs:

The software must act like a bank. It must serve its customers with a user-friendly display that is not just simple and effective, but secure as well. The software must be able to keep track of returning users if a returning user chooses to return, and it must be able to add in new users as the program grows and becomes more popular. Each customer could choose from different account types, such as investment accounts that accumulate interest over time, checking accounts that allow the user to make transactions and withdrawals at any time with certain restrictions, and savings accounts. With the certain restrictions, customers should not be able to withdrawal money from an account with insufficient funds unless an additional charge would be added to the account. The program should also be able to look at account transaction history, keeping track of the amount of money taken and withdrawn. There should be a programmer view of the software to ensure security and accessibility in case of emergency as well.

**Use Cases:**

Use Cases for the banking application:

* Create New Account
  + New Checking
  + New Savings
* Create New Transaction
  + Deposit to Savings
  + Withdraw from Savings
  + Deposit to Checking
  + Withdraw from Checking
* Check Balance
* Update Customer Information
  + Name
  + Address
  + Age
  + Telephone Number
  + Customer (ID) Number
* Get Account Information
  + Customer (Owner)
  + Balance
  + Account Number
  + Transactions (Deposit and Withdrawal)

**Create new Checking account:**

*Step 1:* User issues command to create new checking account

* System prompts for user to add checking account to the name and creates a unique ID for the individual.

*Step2:* User enters name into the program

* System creates an account and sets a starting balance to $0.00

**Create new Savings account:**

*Step 1:* User Issues command to create new savings account

* System prompts for user to add savings account to the name and creates a unique ID for the individual.

*Step 2:* User enters a name into the program

* System creates an account and sets a starting balance to $0.00

**Create new Deposit to Savings:**

*Step 1:* User issues command to create deposit into savings account

* System asks user for Id number and for an amount to deposit into the savings account

*Step 2:* User enters Id number and the amount for the deposit into the savings account.

* If user cancels the entry, process halts and disregards any future entry into the program
* System checks for a valid savings deposit amount. If the amount is appropriate, then the amount is added to the user’s previous account amount and updates the new account amount to the user.
* Appropriate deposit fees will then be applied if necessary.

**Create new Withdrawal from Savings Account:**

*Step 1:* User issues command to create a savings account withdrawal.

* System asks for the customer ID number and amount to withdrawal from the account.

*Step 2:* User enters their ID number and amount to withdrawal

* If user cancels the command, process halts and disregards any future entry into the program.
* System checks to ensure that an amount worth taking out is possible. If so, withdrawal undergoes, subtracting withdrawal amount from the savings account and updating the account balance.

**Create New Withdrawal from Checking Account:**

*Step 1:* User issues command to create new withdrawal from checking account

* System asks user for Id number and amount to withdrawal from checking account

*Step 2:* User enters ID number and amount to withdrawal

* If user cancels the command, process halts and disregards and future entry into the program
* System checks the checking account balance to ensure an amount worth depositing is possible. If so, undergoes the withdrawal, subtracting withdrawal amount from the current checking account. Then updates account balance.

**Check Balance:**

*Step 1:* User issues command to check account balances

* System asks for user ID number

*Step 2:* User enters ID number.

* If user cancels the command, process halts
* System goes and retrieves account balances for all accounts that the user has.

**Update Customer’s Name:**

*Step 1:* User issues command to update customer name

* System asks for customer number and user name

*Step 2:* User enters ID number and new name for the accounts

* If user cancels the command, process halts
* System looks up customer ID and updates the name if there is account with such ID number.

**Update Customer Address:**

*Step 1:* User issues command to update customer address

* System asks for user ID number

*Step 2:* User enters customer ID number and new address

* If user cancels command, process halts
* System looks up user ID number. If it exists, command updates user’s address.

**Update Customer Age:**

*Step 1:* User enters command to update Customer age.

* System asks for user ID and new age

*Step 2:* User enters customer ID number and new age

* If user cancels the command, process halts
* System looks up user ID number. If it exists, age of customer is updated

**Update Customer Telephone Number:**

*Step 1:* User Issues command to update customer telephone number

* System asks for user ID number and new telephone number

*Step 2:* User enters customer ID number and new telephone number

* If user cancels the command, process halts
* System looks for user ID. If it exists, user telephone number is updated

**Update Customer ID Number:**

*Step 1:* User issues command to Update Customer (ID) Number

* System asks for old ID number and a new ID number

*Step 2:* User enters customer ID number and new customer ID number

* If user cancels the command, process halts
* System looks up customer ID number and if the number exists, updates to new customer ID number

**Get Customer Name:**

*Step 1:* User Issues Command to Get Customer Name

* System asks for user ID number

*Step 2:* User enters customer number

* If user cancels the command, process halts
* System looks up ID number, returns customer name if it is found in the system

**Get Customer Balance:**

*Step 1:* User issues command to get customer balance

* System asks for user ID number

*Step 2:* User enters customer number

* If user cancels command, process halts
* System looks up user ID number. If ID exists, gets customer balance of all accounts

**Get Customer Account Number:**

*Step 1:* User issues command to get a customer account number

* System asks user for ID number

*Step 2:* User enters ID number

* If user cancels the command, process halts
* System looks for entered ID number, and returns user ID number if found

**Get Customer Account Transactions:**

*Step 1:* User issues command to get customer account transactions

* System account asks for user ID

*Step 2:* User enters customer ID number

* If user cancels command, process halts
* System looks up customer number and returns customer account transactions if the ID is found.

Macintosh HD:Users:nesrey25:Downloads:Homework 5.pdf

**Pseudocode:**

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

1. Ask user for name, phone number, address, age

2. Read information from the user

3. Ask user for customer type (senior, adult, student)

4. Read in user info

5. Call constructor to create new customer in the program

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

1. Ask user for customer\_number and account\_number

2. Read in user info

3. Ask for how much to be deposited into the appropriate account

4. Call deposit() function

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

1.Ask user for customer\_number and account\_number

2. Read in user info

3. Ask for how much to be withdrawn from appropriate account

4. Call withdrawal fuction()

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

Adding to a pre-existing user:

* 1. Ask user for customer\_number and account type
  2. Read in user info
  3. Ask user if it will be a checking or saving’s account
  4. Read in user info
  5. Call get\_account() to get accounts associated with user – verify valid operation to add account of the specified type – based upon user input
  6. Get a new account\_number
  7. Call constructor and set balance to 0

To add a completely new user to the program

* 1. Call constructor to create new customer
  2. Get a new account\_number
  3. Ask user if it will be a checking or saving’s account
  4. Read in user info
  5. Call constructor and set balance to 0

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

1. Ask user for the customer\_number

2. Loop through Accounts\* to find associated user accounts

Return results

**Bank Data Storage Description:**

* *Account and Customer Numbers:*
* The customer identification number begins with 0001. An incremented private variable would be utilized to assign values to each new customer that decides to create an account. The customer identification number will be stored in the customer object when it is created. The Account numbers will take the customer number and add one digit to the end, a 0 for checking and a 1 for savings. The account number will be stored in the account object when it is created.
* Accounts Linked to Customers:
* Accounts will be linked to customers by having the unique customer identification number as a part of the account number. This will allow the customer to be identified by the first four digits of the account number. Linking the two will allow for efficiency of the unique customer identification number.
* Transactions Linked to Customers:
* Transactions will be linked to the customer by having the account number stored with the transaction. Again this will allow the customer to be identified by the first four numbers of the transaction to allow efficient processing of the account/ID numbers.