Keegan Maynard

Fall 2020

Abstract

GitHub repo Link: https://github.com/KeeganMaynard/SoloFinal.git  
email: kgm1328@live.missouristate.edu

CSC 232 Final project

Solo Bear Bank

Table of Contents

Program Features 2

System Administrators 2

Bank Officials 2

Bank Members 2

Data Structures 3

Users 3

Accounts 3

Dictionary 3

Abstract 3

Other 3

Summary Table 4

Compile Instructions and Link to Repository 5

Disclaimer: Due to the limit on time not all features were able to be implemented

Program Features

**System Administrators**

System administrator are able to do the following:

* Configure Bank Official accounts:
  + Create bank official accounts
  + Enable or disable bank official accounts, set the status of the accounts as good or bad
* Configure Accounts
  + Create, delete, and/or modify the account types available to bank users
* Provide Bank Member Support:
  + Retrieve any user’s login ID
  + Reset the password for any user

**Bank Officials**

Bank Official users are able to do the following:

* Configure Bank Member Accounts:
  + Open or close member accounts
  + Deposit or withdraw from a member account (with member login as permission)
* Search for Member Accounts:
  + Search for member accounts with any of the information saved to the account
    - Member name, ID, phone number, address, and account number

**Bank Members**

Bank member users are able to do the following:

* Configure Personal Accounts:
  + Enroll in the Bear Bank program (create first account)
  + Request to have multiple accounts
  + Reset their password (with assistance from a system administrator
  + Deposit or withdraw from their account
* Review Account Activity:
  + Log in with valid user ID and password to see account information
  + Provided recent activity updates
    - Last account login and time
  + Provided with account transaction review

Data Structures

**Users**

The Bear Bank program includes three variations of user accounts, with each variation having their own functionality and permission level. The parent class user (defined in Users.h) was used to define the commonalities between the system admin users, bank official users, and bank member users.

\*For all user types include complexity analysis if time allows and some description\*

**Accounts**

The Bear Bank program includes three default variations of bank accounts, with each variation having their own functionality and restrictions. System administrator users also have the ability to create a new account type, so a common account parent class was created to provide the framework for the default account types as well as new account types yet to be developed. In order to accomplish this, the account version class was created as the parent to the account class, which determines the properties of the account.

\*For all account types include complexity analysis if time allows and some description\*

**Abstract**

All the data for users and accounts are stored in abstract data structures. AVL trees were used to store information for similar accounts, because users could have similar information, a queue was used to implement a member’s ability to request an additional account, and an array-based dictionary was used to store account information.

\*For all abstract data types include complexity analysis if time allows and some description\*

**Other**

For the functionality of the program, the Time class (timer.h) and controller class (controller.h) were created to easily implement functions to gather the time of events and to manipulate data that is referenced in several classes. The Encryption class was also created to provide security to users by encrypting user information before storing it in text files and by hashing account IDs before inserting them into the dictionary.

\*For all user types include complexity analysis if time allows and some description\*

**Summary Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Summary | Reference |
| Node | User Defined | A node class to easily implement in different data structures | DataFiles/DataStructs/Node.h  DataFiles/DataStructs/AVL.h  DataFiles/DataStructs/AVL.cpp |
| LinkedList | Doubly Linked List | A linked list with pointers to the head and tail for access in other data structures | DataFiles/DataStructs/LinkedList.h |
| Queue | Dynamic Queue | Link-based queue to store user account requests | DataFiles/DataStructs/Queue.h |
| AVL Tree | BST, Dictionary | A self-balancing BST that is used to track all the accounts of a user | DataFiles/DataStructs/AVL.h  DataFiles/DataStructs/AVL.cpp |
| Dictionary | Array, Dictionary | An array-based dictionary that uses hashing and linked lists to store account data | DataFiles/DataStructs/Dictionary.h  DataFiles/DataStructs/Dictionary.cpp |
| DictEntry | User Defined | A struct that holds the key and data that will be inserted into the dictionary | DataFiles/DataStructs/Dictionary.h  DataFiles/DataStructs/Dictionary.cpp |
| AllFiles | User Defined | A group of AVL trees to search members by different parameters | DataFiles/DataStructs/Dictionary.h  DataFiles/DataStructs//Dictionary.cpp |
| Encryption | User Defined | A class designed to provide all users with security by encrypting and decrypting information. Also contains the hash function for the dictionary | DataFiles/DataStructs/Encryption.h  DataFiles/DataStructs/Encryption.cpp |
| Controller | User Defined | A static class designed to interpret data between the classes | DataFiles/DataStructs/Controller.h  DataFiles/DataStructs/Controller.cpp |
| Time | User Defined | A class designed to get and set time for recording account activity and interest | DataFiles/DataStructs/Timer.h  DataFiles/DataStructs/Timer.cpp |
| User | User Defined | Parent class for the derived variations of users | DataFiles/Users.h  DataFiles/Users.cpp |
| System Admin | User Defined | Child class of User that provides system admin functions to the user | DataFiles/SystemAdmin.h  DataFiles/SystemAdmin.cpp |
| Official | User Defined | Child class of User that provides bank official functions to the user | DataFiles/Official.h  DataFiles/Official.cpp |
| Member | User Defined | Child class of User that provides bank member functions to the user | DataFiles/Member.h  DataFiles/Member.cpp |
| Account Version | User Defined | An object created to allow system administrator users to create different account types in the future | DataFiles/Account/Account.h  DataFiles/Account/Account.cpp |
| Account | User Defined | Child of AccountVersion class that establishes the attributes of different account types | DataFiles/Account/Account.h  DataFiles/Account/Account.cpp |

Compile Instructions and Repository

All files needed to access the Bear Bank program are included in the SoloFinal.zip. To compile, place all the files and folders into the Cygwin directory, and ensure that directory is tiered the same as the .zip file. Once the files have been included in the Cygwin directory, open the Cygwin terminal and enter the following lines:

g++ BearBank.cpp -o bearbank -- press enter

./bearbank -- press enter

The terminal will create an executable of the bear bank program, which will provide and handler user interactions with the bank options.

GitHub Repository Link: <https://github.com/KeeganMaynard/SoloFinal.git>