Corey Parks – Final Project Documentation

CS240

12/11/2016

Table of Contents:

Project Summary – 2

Use Case Analysis – 2

Data Design – 3

UI Design - 4

Algorithm - 4

User Instructions - 4

**Project Summary:**

* **Revised Title: C++ Bank Account**
* **Longer Description of Project:**
  + Command line driven bank account/ATM. Allows users to create new accounts.
  + Username and Password: The user creates a username and password that is stored externally. When a user logs in, they are prompted with this information. If the information provided is incorrect, access is denied.
  + Once logged in, the user is able to manipulate their funds.
* **Intended User:**
  + Anyone who would like to streamline their bank account.
* **What problem does the project solve?**
  + Efficiency in banking software
* **What technologies were used?**
  + C++ programming language:
    - Object Oriented Paradigm
    - File Streaming
    - File Removing/Renaming
    - A generous amount of Exception handling
    - String Stream and type conversion

**Use Case Analysis:**

A majority of the use case analysis remained the same in my final project. The only thing I really changed from the concept to the final program was removing the “transfer funds” function. The scope for the project became quite large quickly, and I wanted to ensure the basic functions worked in a OOP design.

What would you like to do?

1.Login

2.Create an Account

3. Quit

> 1

Username: >

Password: >

//If username and password are incorrect or do not exist, give an error message.

Username or password is incorrect.

//If username and password are correct, it logs them in to their account.

Welcome back! You have a current balance of $xxx

Please choose from the following:

1. Deposit
2. Withdraw
3. ~~Transfer Funds~~
4. Logout

>1

Please insert funds (type integer value)

>2

Please remove funds(Type integer value)

//If the user tries to take more than available, give an error message.

>3

~~Please choose from the following:~~

1. ~~Transfer from account~~
2. ~~Transfer to account~~
3. ~~Return to Previous Menu~~

~~Note: In order to transfer funds, username and password of second account is required.~~

~~>1.~~

~~Account 2 Username:~~

~~Account 2 Password:~~

~~How much will you be transferring out?~~

~~//If the user tries to transfer more than available, give and error message. If the password and username are incorrect, create an error message.~~

~~>2.~~

~~Account 2 Username:~~

~~Account 2 Password:~~

~~How much will you be transferring in?~~

~~//If the user tries to transfer in more than is available, print an error message. If the password and username are incorrect, give an error message.~~

~~//We are now returning to the first screen and creating a new account.~~

~~Please enter a username:~~

~~Please enter a password:~~

~~Thank you! Please login with you information to access your account. //Returns to the login screen.~~

~~//If the username already exists, give an error message.~~

**Data Design**

What data is the program really about?

* **The data this program deals with are integer values and strings**

Will the data need to be persistent? How did you make that happen?

* **The data is persistent. My original plan was to create a database using a SQL extension of C++. I ended up parsing the data to a text file that re-writes itself every time there is interaction with it. This allows the user to leave and come back from their “account”, and their balance, username, and password are the same.**

Was the data aggregated into a larger structure?

* **Yes. The bank “has” accounts. We can add parts to the program as member variables, which is done by passing reference values to other methods inside of classes.**

***Note: See unzipped file for UML Diagram***

**UI Design:**

The program is command line driven.

**Algorithm:**

Define Data Members: What are the key data element of the class?

**The Data Members I used are the variables for username, password, balance, newBalance (After withdraws), checkUser, and checkPass. The two “check” variables are used to temporarily store and compare username and passwords when a user is logging in.**

Describe the initializer – what will create and populate the data members? Will you read in parameters? Have default values?

**The initializer is the first menu method in the “Menu” class. This can be thought of like the bank. It is called in main. It then prompts the user with a few different options. If the user decides to create a username and password, it creates an instance of the “Account” class. Once the account is finished, it returns to the Menu1 method. If the user decides to log in from there, it gets the username and password and checks them against the information in the text file. If the username and password are correct, it passes the username and password by reference to Menu2, where the user has the ability to manipulate the information in their account (Withdraw, Deposit, and Check Balance). All constructors for both classes are null.**

**User Instructions:**

I have given two files. One is a functionally driven version, while the other is completely OOP, and is given with a makefile.

Once the program is running you will be at the first menu. It does not allow you to log in if a username and password have not been created.

A username and password can be whatever you want. Any string is accepted.

Once you have created a username and password, you may log in.

Your default balance will be 0.

Please only use whole “int” data types for deposits and withdraws. Change is not accepted at my bank.

There is a bug in the withdraw process. If you try to take out more than is in the account, it displays an error message, as it should. However, it then deletes the account of the user that is currently logged in.

Professor Ryan also suggested that I consider security for the project. The idea I had was to run the usernames and passwords through an encryption method (similar to the cryptography project done in CS230). Then output the encrypted text to the file. That way if someone comes upon the text file containing all of the information, it would be worthless unless they knew the key. I did not implement this on this version of the project, as I wanted to get the basics working. Scope becomes very large quickly. It could have been a separate class that gets the username and password by reference and then returns them, or it could have been a method within the same class. A new class would have been better for ease of code reusability.