**README - Bank**

**Structure:**

The program is made so that there are several functions in a menu, which are available in the case that the user successfully logs in. The program starts by requesting a username and loops until it is given. Progressing, the password is required to continue to the menu and if the password is not entered before three tries the program shuts off.

**The functions in the program, which are available from the menu, are (in Swedish):**

**Method** - ShowAccountsAndMoney(seKonton)

**Method** – Login(Inloggning)

**Method** - Transfer(transfer)

**Method** – TakeOutMoney(taUt)

**Method** – Menu(Meny)

**Non-methodous-function** -Log out(LoggaUt)

**Sequentially the program could be used in this order:**

Login, Menu, ShowAccountsAndMoney, Transer, TakeOutMoney, LogOut

**Main**:

In Main I created several string 2D-arrays: username(holding usernames and passwords), Accounts(holding equal usernames and several accounts. Although every user has the same amount of accounts, they do have different values on their accounts; some have empty accounts which are unnamed but usable, and later displayed as “Empty Account”. I did this since it is possible to have empty accounts. I could have alternatively made the program not write out the accounts in the case that they were empty.

The decimal 2D-array is created with the same number of values as the other arrays, so that it would be simpler to loop through the arrays at the same time. The decimal values, which of some are 0, are converted to properly show the correct local currency, only when it is supposed to be written for the user.

I also created variables, which are used in methods, in Main so that I could run the program, which starts by the Login method.

**Reasoning**:

The reasoning behind the way the program is built is solely based upon the code I believe was most logical to use since the functions demand certain properties, and I feel there are certain obvious options to these demands.

**The Login** starts an infinity loop which is run until you enter one of the users. Another loop starts which in you can enter your password, and if you don’t succeed within three tries, Environment.Exit(0) will exit the program.

**The Menu** is also looping infinitely and is only passed by entering one of the methods.

**ShowAccountsAndMoney** solely uses a for-statement to display accounts and currency, which are related to the user. It is possible since the user has an array index, which is declared and saved so that it can be used for the loop. This is something I declared several times in the code so that the correct users accounts and money are displayed.

**Transfer** displays the accounts, and money, as well as allowing the user to choose an account to transfer money to, an account to transfer the money from, and the amount of money to transfer. All these variables are saved so that simple calculations later can be declared to transfer the money and change the values in the arrays. All user input is secured by do-while-loops while invalid input it given and try-catch-statements to inform the user to change the input.

**TakeOutMoney** displays the accounts and money and allows option to take out money from the chosen account, which later has the chosen sum removed from it.

**LogOut** returns the user to the Inlog where a new user can log in, leaving all changes from the previous users saved.

I used a lot of for-statements and if-if else:s to display the accounts and money.

To properly display the correct accounts and money in the arrays, and not also repeating the belonging username, I had to declare variables which were created depending on the user input, so that it would be easier to write down the specific users properties: If the users index is 2,0 then the 2 is saved so that it can display the values following 2,0, on the 2-row in the array.

I built the program starting with the methods, as a skeleton. Putting in necessary code to make the methods work, as muscles on the bones. I wanted to make it usable first, and handle the possible errors lastly, which I did, with some time.

If I had to do it again, I would probably write down the skeleton, muscles and identify the possible errors on paper first, before writing (and rewriting) the code.

I experimented by creating a method to display the accounts and money but decided not to use that specific method.

I would have rather used classes to build the program.