**Introduction**

A company has contacted you and asked you to develop an ATM machine that can be used from the command line. The ATM will be used by a number of different users, each of which have their own accounts and different amounts of money stored in the bank.

When the command line application is loaded, the user should first be welcomed to the bank’s ATM machine and then told to enter in their customer ID number followed by their four-digit pin number.

If these are correct the user should be brought to a second screen in the application which will present each of the different options available to the user. These menu items include:

1. Check the current account balance
2. Withdraw money from their account
3. Change their current password
4. Check the latest stock prices for the bank
5. Logout of the system.

The company has outlined that they want this system to be **robust**, due to the fact that often users do not enter in the correct information when they are asked to. As Java has no storage memory after the program has finished, all data should be stored inside of **text files** and loaded into the program once the user has started and saved when the user is finished. Each user will have their own text file, e.g., user 500 will have a text file called 500.txt.

The customer has also specified that they would like you to use good programming practices during development, using methods to break down each core element of the program, making it easier to reuse these again in the future.

For a complete breakdown of the specification for the system, please reference the system specification below.

**System Specification (20%)**

* When the program is first opened a welcome message should be printed to the screen welcoming the user to the system.
* After the welcome message has shown, the user should be prompted to type in their user ID number and their four-digit password.
* If the user ID and password are valid, the user should then be brought to the menu screen in the program.
* A global variable should be created in the system to store the user ID that the user has logged in with (this will be used to read in their account information in later)
* The user should be able to enter the number of the five options available to them:  
  1. Check the current account balance
  2. Withdraw money from their account
  3. Change their current password
  4. Check the latest stock prices for the bank
  5. Logout of the system.

* If the user types in 1, they will be shown their account balance. This account balance should be read from a text file in your system for that user ID. e.g., if user 500 logged into the system, then the text file titled 500.txt should be read into the system. The first line in this text file should be the current amount that they have in the account.
* If the user selects the option 2, the text file for that user account should be read in and if funds are available, the user should be prompted to enter in the amount they would like to withdraw from the system. This amount should be subtracted from their account balance and the text file should be updated.
* If the user selects option 3, the user should be asked to enter in the new four-digit password that they would like to use. They should then be asked a second time to verify that they did not make a mistake when entering in the password. Both the first password they entered and the second password should be validated against each other to ensure that they match. If they do, the second line in the text file should be updated with their new password.
* If the user chooses option 4, a file titled “stocks.txt” should be read into the system. All the information should be printed to the console. At the very end of this process, the user should then be asked do they want to return to the main menu. If they enter **Y** then they will be, if they enter **N** then the program will close.
* If the user chooses option 5, then the program should say goodbye and close the program.
* At every state in the system, whenever the user is asked to enter any input, the user should **not ever be able** to proceed to the next step unless they have entered the validated information.
* The code should all be correctly indented and comments should be added to explain at each step what is happening, allowing future developers to easily work on the code.

**Distinction Grade Additional Work (5%)**

* If you are looking to get a distinction for this assignment, create an additional menu item (item 6) called “Bank Summary”
* This menu item should call a custom method in the system that will read in all the account balances of the user’s in the system and create a total for the amount of money currently stored in the bank.
* A calculation should also be performed to break the accounts into 4 different categories, and count how many accounts are in each group:
  + small (less than 100 euro)
  + medium (101 – 200 euro)
  + large (201-300 euro)
  + extra large (more than 301 euro)

**Notes**

* This is an **individual assignment**. If two assignments are the same both will be marked zero.
* Late submissions will have 5% of the grade taken for each day late.
* All submissions must be made through Moodle. E-mailed assignments will not be graded.

Grading Scheme

|  |  |
| --- | --- |
| **Description** | **Marks: 20%** |
| Welcome message printed to the screen by custom method | 1 |
| Each core element of the program has been broken down into smaller methods, allowing for easy reusability. | 2 |
| User ID and password can be entered, and validated if left blank | 2 |
| Menu presented to the user, allowing the user to select which sub menu they would like to enter. Including validation on the input only allowing to choose 1,2,3,4,5. | 2 |
| User can withdraw money from their account and the amount they have selected is take away from the amount they have stored in the text file for their user ID | 5 |
| Option to change their password working correctly, the user will enter their new password twice and they will be validated to ensure they both match. The result is stored in the second line of their text file. | 2 |
| If the user selects option 4, all of the information about the current stock prices for the bank will be listed. The user should then be prompted to return back to the welcome screen if they type Y or exit the program if they type N | 2 |
| All input which is typed into the system should be validated to ensure only the correct options have been typed. The user should never progress with invalid input | 2 |
| Overall code quality and comments added into the code explaining at each step what is being performed. | 2 |
| **Distinction** | **Marks 5%** |
| Total account balance for the entire bank is calculated and printed when option 6 is selected | 2 |
| The total number of accounts in each category are printed to the screen, in format of small account, medium, large and extra large. | 3 |
| **Total:** |  |