Practical project

420-D02-SU

Introduction to structured programming

AEC Programming, Networks and Security (LEA.5F)

AEC Programming and Web Technologies (LEA.5G)

AEC Video Game Programming (LEA.CU)

**Evaluation weight:**

25% of the final grade

**Submission specifications:**

* **Due date: morning of the final exam**
* Name the program correctly: pw2Bank + *name1* + *name2* [ + *name3* ]
* The code of the program and any dependencies
* The compressed file deposited in Dropbox in the folder designated for this purpose

**Description:**

Create a prototype of a program that simulates a bank. More precisely, the program should be able to store the family name and given name of a client, their unique account number (a number between 10000 and 10099), and the balance of their account.

The program should display a menu with the following options:

1. Add a bank account.
2. Remove a bank account. (optional)
3. Display the information of a particular client’s account. (by account number)
4. Apply a deposit to a particular account. (by account number)
5. Apply a withdrawal from a particular account. (by account number)
6. Sort and display the list of clients according to their balance, family name and given name, in ascending or descending order.
7. Display the average balance value of the accounts.
8. Display the total balance value of the accounts.
9. Exit the application.

You must implement each of these functionalities.

**Remarks:**

* The maximum number of accounts for this application is 100 accounts.
* All input must be validated.
* Each task out of the various functionalities should be delegated to an appropriate function.
* This project must be done in teams of two or three.

**Teamwork Validation**

This is a team project and must be worked together as a group, and not copied from any external sources.To verify that all members are familiar with the code, after submission each team member will be seperated and questioned about the project. The questions will be along the lines of “Show me the code that does X”, “What will this breakpoint reveal if I do…” or “If I change this code here, what will happen?”. If a member is found being unfamiliar with the code, their total mark will be multiplied by the amount they know. For example, if a member is shown to understand only 30% of the code, and the project received a 90%, their total mark will be 27%.

**Evaluation:**

The project will be graded according to the following criteria:

* **Exactitude of the program:**Does the program do what it is supposed to do?
* **Visual appearance:**All text displayed to the screen should be well arranged and written in proper English.
* **Input validation:**The program should properly handle errors that can occur during data input.
* **Structure of the code:**The program should be split up into functions according to the relevant needs.
* **Arrangement and clarity of the code:**The code should be properly indented, with relevant comments where needed, and it should respect programming conventions (variables in camelCase, constants in UPPER\_CASE\_SEPARATED\_BY\_UNDERSCORES). Hard-coded values should be put into constants.

|  |  |
| --- | --- |
| **Tasks to accomplish** | **Points** |
| Respect submission specifications | 1 |
| Program compiles and executes | 1 |
| Structure of the code | 2 |
| Menu | 2 |
| Add an account | 2 |
| Remove an account | 2 |
| Display information for all accounts | 2 |
| Display information for a particular account | 2 |
| Make a deposit into an account | 2 |
| Make a withdrawals from an account | 2 |
| Display the list of clients by alphabetical order | 5 |
| Display the average value of all the accounts | 1 |
| Display the total value of all the accounts | 1 |
| Exit the application | 1 |
| Input validations | 3 |
| Arrangement and clarity of the code | 2 |
| Global evaluation (teamwork, effort) | 3 |
| **Total** (with bonus in parentheses) | **34** |