Coding Bootcamp 3

Project 1 – Java Stream

Definition

You are required to build a Windows console application where you will be asked to read various inputs from the keyboard.

These inputs will be used as login details and actions in order to control the internal banking system of a cooperative company.

Each input will be used for directing the various subsystems of this private banking system.

The output of the various subsystems will be displayed to the screen or it will be written to simple text files.

A. Logical Units of the application

1. Main application
2. Login Screen
3. Application's menus
4. Database's access
5. Files' access
6. Internal Bank Accounts

B. Description of the Logical Units of the application

1. The Main Application
   It holds all the class variables that will be consumed from the rest of the classes. It checks for proper database connectivity and displays via Logical Unit 2 the menus.
2. Login Screen
   You are required to build a class that will act as the login screen of the application. This login screen will be asking for the **username / password** of the user.
   The user should be able to login with the following credentials (username / password):

- admin / admin, this is the super admin account
- user1 / password1, this is a simple member's account
- user2 / password2, this is a simple member's account

The credentials should be checked for validity against the stored values in the database.

3. The Application's menus are defined based on the level of the user.
   The **super admin account** should have the following menus:

   - View Cooperative's (super admin) internal bank account
   - View Members' bank accounts
   - Deposit to Member's bank account
   - Withdraw from Member's bank account
   - Send to the **statement_admin_dd_mm_yyyy.txt** file today's transactions where **dd_mm_yyyy** are today's day, month, year and Exit
   - Exit the application

   The **simple member's account** should have the following menus:

   - View his bank account
   - Deposit to Cooperative's internal bank account
   - Deposit to another Member's bank account
   - Send to the ***statement_user_x_dd_mm_yyyy.txt*** file today's transactions where **dd_mm_yyyy** are today's day, month, year
   - Exit the application

4. The database should be created and initialized using the accompanied file:

   afdemp_java_1.sql

   The database must be a local MySQL database running on the same machine with the application.

   From the Application's menu each user should be able, depending on his level to:

   - View his account (all levels)
   - View all accounts (only super admin)
   - Deposit to Cooperative's internal bank account (simple member)
   - Deposit to a Member's bank account (all user levels)
   - Withdraw from Member's bank account (only super admin)
   - Send today's statement to a file (all user levels)

5. All user levels while they are logged in, must keep a memory buffer with all today's transactions. When the user selects from the menu ***Send Today's Statement*** to the file ***statement_user_x_dd_mm_yyyy.txt*** for the simple users or ***statement_admin_dd_mm_yyyy.txt*** for the super admin the application should write to the file all of the today's transactions, then log out the user and terminate.

6. The Internal Bank Accounts are used for the following operations:

    - Display the user's account balance (all user levels)
    - Display all of the users' account balances (only super admin)
    - Deposit to a member's account by using the member's username and the desired amount (all user levels) from the current user's bank account
    - Withdraw from a member's bank account to the cooperative's account using the member's username and the desired amount (only super admin)

C. Deliverables

You are requested to produce a Java console application that has the following:

1. Six (6) .java files, the main application and one file per Logical Unit of the application as described above on A. **[9 marks]**
2. The Login Screen should be displayed first and it should let the user view the application's main menu after correct input of the username / password combination which are checked against the values stored in the database table *users* **[12 marks]**
3. The application's main menu should be changed depending the level of the user as described above on B.2. **[4 marks]**
4. The application should let the super admin to view all the accounts and deposit or withdraw from the simple users' bank accounts while keeping all actions to memory and write to statement file via ***Send Today's Statement*** **[27 marks]**
5. The application should let the simple users to deposit to the cooperative's bank account an amount that is available to his bank account while keeping all actions to memory and write to statement file via ***Send Today's Statement*** **[25 marks]**
6. Use the class BankAccounts as an intermediate storage for database's data. Override the toString() of the class BankAccount in order to show the user's **username, transaction date, amount**. Format all currency data to be displayed and written to file with Locale("el-GR"). Format all dates to the form "yyyy-MM-dd HH:mm:ss.SSS". **[23 marks]**

Total marks for the Project = 100
Oral Examination marks = 20

A **Detailed Marking Scheme** Follows.

Deliverables / Marks	TASKS							Conditions for Obtained Marks
1 / 9 Marks	a. Use at least 6 class files (as the Logical Units)	b. Main file consumes the others via private class variables	c. ability to clear the Console before menu redisplay	d. checks connectivity with the database				a. **0.5 per file**, **3 in total** b. **2** for private class variables c. **2** for clearing the console d. **2** for database connectivity
2 / 12 Marks	a. the password field is not prompted	b. after 3 false consecutive tries the app exits	c. change the db to use encrypted password					a. **3** for hidden password from prompt! b. **3** for exit after false consecutive tries c. **6** for password encryption
3 / 4 Marks	a. current user level is checked and menu changes accordingly	b. the user input is checked if an inappropriate input is given						a. **2** for dynamic menu change b. **2** for checking user input
4 / 27 Marks	a. super admin views his account	b. super admin views members' accounts	c. super admin withdraws from members' accounts (loop)	d. super admin deposits to members' accounts (loop)	e. all a-d tasks use appropriate setters and getters	f. check if the amounts to be withdrawn or deposited are available	g. keep transactions to memory and send to statement file	a. **2** for viewing his account b. **2** for viewing other accounts c. **2** for withdrawing from other accounts d. **2** for depositing to other accounts e. **3 X 4** (a-d)**, 12 in total**

								f. **2** for correct amounts g. **2** for memory usage and **3** for sending to statement file
5 / 25 Marks	a. member views his account	b. deposit to super admin's account	c. deposit to another member's account via menu selection	d. all a-c tasks use appropriate setters and getters	e. check if the amounts to be withdrawn or deposited are available	f. keep transactions to memory and send to statement file		a. **2** for viewing his account b. **2** for depositing to super admin's account c. **5** for depositing to other accounts d. **3 X 3** (a-d)**, 9 in total** e. **2** for correct amounts f. **2** for memory usage and **3** for sending to statement file
6 / 23 Marks	a. class BankAccount is used as an intermediate storage	b. Override of the toString() in order to show the user's **username, transaction date, amount**	c. Format displayed and written currency data with Locale("el-GR")	d. Format all the dates to the form "yyyy-MM-dd HH:mm:ss.SSS"				a. **10** b. **3** c. **5** d. **5**
Total 100 Marks								