CS 101 - Algorithms & Programming I

Fall 2025 - Lab 3

Due: Week of October 13, 2025

*Remember the **honor code** for your programming assignments.*
*For all labs, your solutions must conform to the CS101 style **guidelines**!*
All data and results should be stored in variables (or constants) with meaningful names.

The objective of this lab is to learn how to program simple and complex decisions by using if/else statements. Remember that analyzing your problems and designing them on a piece of paper *before* starting implementation/coding is always a best practice.

In this particular lab, do **not** use a switch-case statement instead of an if-else statement!

0. Setup Workspace

Start VSC and open the previously created workspace named `labs_ws`. Now, under the labs folder, create a new folder named `lab3`.

In this lab, you are to have three Java classes/files (under `labs/lab3` folder) as described below. A fourth Java file containing the revision should go under this folder as well. We expect you to **submit a total of 4 individual files,** including the revision, **without compressing** them. Do *not* upload other/previous lab solutions in your submission.

The user inputs in the sample runs are shown in blue.

1. Student Grade Evaluation System

Create a Java program named `Lab03_Q1.java` in the `lab3` folder. This program will ask the user to enter their midterm and final exam scores and then calculate their overall course average. The average is calculated using a weighted formula where the midterm contributes 40% and the final exam contributes 60% to the overall grade. It is calculated using the formula: **Average = (Midterm × 0.40) + (Final × 0.60)**. Based on the average value and specific conditions, the program will determine the student's letter grade and whether they passed or failed the course.

Important rule: If the final exam score is below 50, the student automatically fails with an FF grade, regardless of their average.

The letter grade categories and their ranges are:

- AA: Average ≥ 90
- BA: 85 ≤ Average < 90
- BB: 80 ≤ Average < 85
- CB: 75 ≤ Average < 80
- CC: 70 ≤ Average < 75
- DC: 60 ≤ Average < 70
- DD: 50 ≤ Average < 60
- FF: Average < 50

Sample runs:

```
Enter your midterm score: 75
Enter your final exam score: 82
Your average is: 79.20
Your letter grade is: CB
You passed the course successfully. Congratulations!
```

```
Enter your midterm score: 60
Enter your final exam score: 45
Your average is: 51.00
Your letter grade is: FF
You failed the course due to insufficient final exam score.
```

```
Enter your midterm score: 80
Enter your final exam score: 95
Your average is: 89.00
Your letter grade is: BA
You passed the course successfully. Congratulations!
```

```
Enter your midterm score: 45
Enter your final exam score: 52
Your average is: 49.20
Your letter grade is: FF
You failed the course.
```

2. Credit Card Application Evaluator

Create a new/empty file under the `lab3` folder named `Lab03_Q2.java` with a class with the same name that evaluates a person's eligibility for a credit card application based on multiple parameters. The program should take the following inputs:

1. The applicant's age (an integer value).
2. The applicant's annual income (a double value).
3. The applicant's credit score (an integer value).
4. The applicant's number of existing credit cards (an integer value).
5. The applicant's monthly rent/mortgage payment (a double value).

The program should calculate the applicant's eligibility score using these parameters. The rules for eligibility score calculation are as follows:

- Base eligibility score: 100
- For age: Add 10 points if age is between 18-25, add 20 points if age is between 26-35, add 25 points if age is between 36-50, add 15 points if age is between 51-65.
- Add 5 points for every $10,000 of annual income (up to a maximum of 200 points).
- For credit score: Add 0 points if score is 300-579, add 50 points if score is 580-669, add 100 points if score is 670-739, add 150 points if score is 740-799, add 200 points if score is 800-850.
- Subtract 10 points for each existing credit card.
- Subtract 5 points for every $500 of monthly rent/mortgage payment.

After calculating the applicant's eligibility score, the program should determine eligibility for the credit card. If s/he is not eligible, then the reason must be printed. The rules for credit card eligibility are as follows:

- The applicant's age must be at least 18.
- The applicant's annual income must be at least $15,000.
- The applicant's credit score must be at least 580.
- The applicant's number of existing credit cards must not exceed 5.
- The applicant's calculated eligibility score must be at least 250.

Sample runs:

```
Enter the applicant's age: 30
Enter the applicant's annual income: 60000
Enter the applicant's credit score: 750
Enter the number of existing credit cards: 2
Enter the monthly rent/mortgage payment: 1200
Total eligibility score: 287
The applicant is approved for the credit card.
```

```
Enter the applicant's age: 17
Enter the applicant's annual income: 25000
Enter the applicant's credit score: 700
Enter the number of existing credit cards: 1
Enter the monthly rent/mortgage payment: 800
Total eligibility score: 242
The applicant is not approved for the credit card. Reason:
Age is below 18.
```

```
Enter the applicant's age: 22
Enter the applicant's annual income: 12000
Enter the applicant's credit score: 720
Enter the number of existing credit cards: 0
Enter the monthly rent/mortgage payment: 500
Total eligibility score: 221
The applicant is not approved for the credit card. Reason:
Annual income is below $15000.
Total eligibility score is below 250 points.
```

```
Enter the applicant's age: 28
Enter the applicant's annual income: 45000
Enter the applicant's credit score: 550
Enter the number of existing credit cards: 1
Enter the monthly rent/mortgage payment: 900
Total eligibility score: 266
The applicant is not approved for the credit card. Reason:
Credit score is below 580.
```

```
Enter the applicant's age: 35
Enter the applicant's annual income: 80000
Enter the applicant's credit score: 800
Enter the number of existing credit cards: 6
Enter the monthly rent/mortgage payment: 1500
Total eligibility score: 325
The applicant is not approved for the credit card. Reason:
Number of existing credit cards exceeds 5.
```

```
Enter the applicant's age: 24
Enter the applicant's annual income: 20000
Enter the applicant's credit score: 620
Enter the number of existing credit cards: 3
Enter the monthly rent/mortgage payment: 600
Total eligibility score: 144
The applicant is not approved for the credit card. Reason:
Total eligibility score is below 250 points.
```

Note: If an applicant fails due to multiple conditions, the program should output all reasons for the failure. For example, if an applicant is 17 years old and also has insufficient income, the program should output:

The applicant is not approved for the credit card. Reason: Age is below 18. Annual income is below $15000.

3. Simple Library Management System

Create a new/empty file under the `lab3` folder named `Lab03_Q3.java` with a class with the same name. Your program will be a simple library management system where users can log in, manage their library records, and log out.

The username and password must be stored as `String` objects. Assume that the username is "librarian" and the password is "books2024". The user will have a list of members, stored as a `String` object. Also assume that the user starts with 3 predefined members: "JohnSmith, EmilyDavis, ZeynepDemir, " (include a comma and space after each item).

In addition to that the user will have books which are stored in a `String`.. Books are stored with their ISBN numbers. Assume that the user has 2 predefined books and these are stored as "ISBN9781:JavaProgramming ISBN9782:DataStructures ", where ISBN numbers are followed by a colon and then a short name.

The user must be able to perform the following operations in the application:

- **Log in** using a username and a password. If both are correct, display the operations listed below. Otherwise, display “Username not found! Goodbye!” if the username is incorrect, and “Incorrect password! Goodbye!” if the password is wrong, then exit.

```
Enter your username: librarian
Enter your password: books2024
1- Add member
2- Delete member
3- Add book
4- Delete book
5- Logout
Select an operation:
```

Enter your username: student Username not found! Goodbye!	Enter your username: librarian Enter your password: books2023 Incorrect password! Goodbye!

- **To add a member**, the user must enter 1. Display the chosen operation "-- Add Member --". After that, the name of the member who will be added is asked of the user. If the name is in the members list, then display "This member is already in your list!"; if not, display "New member <memberName> is added!", where <memberName> is the name of the newly added member. At the end, display all of the members.

-- Add Member -- Enter member name: AyşeYılmaz This member is already in your list! Your members: (AyşeYılmaz, MehmetKaya, ZeynepDemir,)	-- Add Member -- Enter member name: SarahWilson New member SarahWilson is added! Your members: (JohnSmith, EmilyDavis, MichaelBrown, SarahWilson,)

- **To delete a member**, the user must enter 2. Display the chosen operation "-- Delete Member --". After that, the ~~member~~ name of the member who will be deleted is asked of the user. If the member name is in the members list, then delete the member and display "<memberName> is deleted successfully from members!"; if not, display "You don't have any member whose name is <memberName>!", where <memberName> is the entered member name. At the end, display all of the members.

-- Delete Member -- Enter member name which you want to delete: DavidJones You don't have any member whose name is DavidJones! Your members: (JohnSmith, EmilyDavis, MichaelBrown,)	-- Delete Member -- Enter member name which you want to delete: MichaelBrown MichaelBrown is deleted successfully from members! Your members: (JohnSmith, EmilyDavis,)

- **To add a book**, the user must enter 3. Display the chosen operation "-- Add Book --". After that, the name of the book that will be added is asked of the user. Name of the book must be a `String`. After that, the user enters the name of the book, and a random number is generated for the ISBN. If the generated ISBN exists already in the earlier books, then display "There is a book with the ISBN <isbn>, you cannot add a new book with the same ISBN!"; if not, display "New book with ISBN <isbn> is added!", where <isbn> is the randomly generated ISBN. At the end, display all of the books.

-- Add Book -- Enter book name: Algorithm Design There is a book with the ISBN 9781, you cannot add a new book with the same ISBN! Your books: ISBN9781:JavaProgramming ISBN9782:DataStructures	-- Add Book -- Enter book name: Algorithm Design New book with ISBN 9856 is added! Your books: ISBN9781:JavaProgramming ISBN9782:DataStructures ISBN9856:AlgorithmDesign

- **To delete a book**, the user must enter 4. Display the chosen operation "-- Delete Book --". After that, the ISBN that will be deleted is asked of the user. If the entered ISBN is in the books list, then delete the book and display "The book with the ISBN <isbn> is deleted successfully!"; if not, display "You don't have any book with the ISBN <isbn>!", where <isbn> is the entered ISBN. At the end, display all of the books.

-- Delete Book -- Enter ISBN which you want to delete: 5000 You don't have any book with the ISBN 5000! Your books: ISBN9781:JavaProgramming ISBN9782:DataStructures	-- Delete Item – -- Delete Book -- Enter ISBN which you want to delete: 9782 The book with the ISBN 9782 deleted successfully! Your books: ISBN9781:JavaProgramming

- **To logout**, the user must enter 5. Display "Logged out successfully!".

You can consider this as a one-session application where you run it from scratch each time. In other words, the username, password, members or books are not updated according to the previous run. However, you should be able to change these values and re-run your program to get different results.