CS 101 - Algorithms & Programming I

Fall 2024 - Lab 3

Due: Week of October 14, 2024

*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 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 color.

1. BMI (Body Mass Index) Calculator

Create a Java program named `Lab03_Q1.java` in the `lab3` folder. This program will ask the user to enter their weight (in kilograms) and height (in meters) and then calculate their Body Mass Index (BMI). The BMI is a measure used to determine whether an individual has a healthy body weight for a given height. It is calculated using the formula: BMI = weight / (height * height). Based on the BMI value, the program will categorize the user as underweight, normal weight, overweight, or above the recommended range.

The BMI categories and their ranges are:

- Underweight: BMI < 18.5
- Normal weight: 18.5 ≤ BMI ≤ 24.9
- Overweight: 24.9 < BMI ≤ 29.9
- Above the recommended range: BMI > 29.9

Sample runs:

```
Enter your weight in kilograms: 45
Enter your height in meters: 1.75
Your BMI is: 14.69
You are underweight.
```

```
Enter your weight in kilograms: 68
Enter your height in meters: 1.75
Your BMI is: 22.20
You are of normal weight.
```

```
Enter your weight in kilograms: 85
Enter your height in meters: 1.75
Your BMI is: 27.76
You are overweight.
```

```
Enter your weight in kilograms: 95
Enter your height in meters: 1.70
Your BMI is: 32.87
Your BMI indicates that you are above the recommended range.
```

2. Scholarship Eligibility Checker

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 scholarship based on multiple parameters. The program should take the following inputs:

1. The student's GPA (a double value).
2. The student's total number of completed credit hours (an integer).
3. The student's extracurricular activity hours (an integer).
4. The student's number of academic honors received (an integer).
5. The student's total annual family income (a double value).

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

- Base scholarship score: 200
- Add 15 points for each credit hour completed (up to a maximum of 600 points).
- Add 10 points for each hour of extracurricular activity (up to a maximum of 300 points).
- Add 20 points for each academic honor received (up to a maximum of 100 points).
- Subtract 5 points for every $5,000 of total annual family income

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

- The student's GPA must be at least 3.5.
- The student's total number of completed credit hours must be at least 30.
- The student's extracurricular activity hours must be at least 20.
- The student's total annual family income must not exceed $75,000.
- The student's calculated scholarship score must be at least 900.

Sample runs:

```
Enter the student's GPA: 3.5
Enter the total number of completed credit hours: 40
Enter the extracurricular activity hours: 50
Enter the number of academic honors received: 10
Enter the total annual family income: 50000
Total scholarship score: 1150
The student is eligible for the scholarship.
```

```
Enter the student's GPA: 3.2
Enter the total number of completed credit hours: 40
Enter the extracurricular activity hours: 50
Enter the number of academic honors received: 5
Enter the total annual family income: 50000
Total scholarship score: 1150
The student is not eligible for the scholarship. Reason:
GPA is below 3.5.
```

```
Enter the student's GPA: 3.8
Enter the total number of completed credit hours: 28
Enter the extracurricular activity hours: 50
Enter the number of academic honors received: 5
Enter the total annual family income: 50000
Total scholarship score: 970
The student is not eligible for the scholarship. Reason:
Fewer than 30 completed credit hours.
```

```
Enter the student's GPA: 3.9
Enter the total number of completed credit hours: 40
Enter the extracurricular activity hours: 10
Enter the number of academic honors received: 5
Enter the total annual family income: 50000
Total scholarship score: 950
The student is not eligible for the scholarship. Reason:
Fewer than 20 extracurricular activity hours.
```

```
Enter the student's GPA: 3.9
Enter the total number of completed credit hours: 40
Enter the extracurricular activity hours: 50
Enter the number of academic honors received: 6
Enter the total annual family income: 100000
Total scholarship score: 1100
The student is not eligible for the scholarship. Reason:
Family income exceeds $75000.
```

```
Enter the student's GPA: 3.6
Enter the total number of completed credit hours: 30
Enter the extracurricular activity hours: 20
Enter the number of academic honors received: 0
Enter the total annual family income: 50000
Total scholarship score: 800
The student is not eligible for the scholarship. Reason:
Total scholarship score is below 900 points.
```

Note: If a student fails due to multiple conditions, the program should output **all reasons** for the failure. For example, if a student has a GPA of 3.2 and also has insufficient credit hours, the program should output:

```
The student is not eligible for the scholarship. Reason:
GPA is below 3.5.
Fewer than 30 completed credit hours.
```

3. Simple Inventory 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 inventory management system where users can log in, manage their inventory, and log out.

The username and password must be stored as `String` objects. Assume that the username is "manager" and the password is "inventory". The user will have a list of customers, stored as a `String` object. Also assume that the user starts with 3 predefined customers: "PrimeTech, Peak, EcoGoods, " (include a comma and space after each item).

In addition to that the user will have items which are stored in a `String`. Items are stored with their item IDs. Assume that the user has 2 predefined items and these are stored as "Item104:Laptop Item125:Monitor ", where IDs are followed by a column 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: manager
Enter your password: inventory
1- Add customer
2- Delete customer
3- Add item
4- Delete item
5- Logout
Select an operation:
```

```
Enter your username: employee
Username not found! Goodbye!
```

```
Enter your username: manager
Enter your password: archive
Incorrect password! Goodbye!
```

- To add a customer, the user must enter 1. Display the chosen operation “-- Add Customer --”. After that, the name of the customer who will be added is asked to the user. If the name is in the customers list, then display "This customer is already in your list!"; if not, display "New customer <customerName> is added!", where <customerName> is the name of the newly added customer. At the end display all of the customers.

```
-- Add Customer --
Enter customer name: PrimeTech
This customer is already in your list!
Your customers: (PrimeTech, Peak, EcoGoods, )
```

```
-- Add Customer --
Enter customer name: CrestviewCorporation
New customer CrestviewCorporation is added!
Your customers: (PrimeTech, Peak, EcoGoods,
CrestviewCorporation, )
```

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

-- Delete Customer -- Enter customer name which you want to delete: AlexCorp You don't have any customer whose name is AlexCorp! Your customers: (PrimeTech, Peak, EcoGoods,)	-- Delete Customer -- Enter customer name which you want to delete: EcoGoods EcoGoods is deleted successfully from customers! Your customers: (PrimeTech, Peak,)

- To add an item, the user must enter 3. Display the chosen operation “-- Add Item --”. After that, the name of the item that will be added is asked to the user. Name of the item must be a `String`. After that, the user enters the name of the item, and a random number is generated for the item ID. If the generated id exists already in the earlier items, then display "There is a item with the <itemId> id, you cannot add a new item with the same id!"; if not, display "New item with <itemId> id is added!", where <itemId> is the randomly generated ID. At the end display all of the items.

-- Add Item -- Enter item name: Television There is an item with the id 104, you cannot add a new item with the same id! Your items: Item104:Laptop Item125:Monitor	-- Add Item -- Enter item name: Television New item with id 243 is added! Your items: Item104:Laptop Item125:Monitor Item243:Television

- To delete an item, the user must enter 4. Display the chosen operation “-- Delete Item --”. After that, the item ID which will be deleted is asked to the user. If the entered ID is in the items list, then delete the item and display "The item with the id <itemId> is deleted successfully!"; if not, display "You don't have any item with the id <itemId>!", where <itemId> is the entered ID. At the end display all of the items.

-- Delete Item -- Enter itemId which you want to delete: 190 You don't have any item with the id 190! Your items: Item104:Laptop Item125:Monitor	-- Delete Item -- Enter itemId which you want to delete: 125 The item with the id 125 is deleted successfully! Your items: Item104:Laptop

- 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, customers or items 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.