Assignment 1 - consumer object

Object-Oriented Programming II (Concordia University)

Concordia University
Computer Science & Software Engineering
Faculty of Engineering and Computer Science

Concordia University
Comp 249 – Fall 2021
Object-Oriented Programming II
Assignment 1 – Due 11:59 PM - Sunday October 03, 2021

Purpose: The purpose of this assignment is to help you review some of the foremost topics covered in the previous course, including classes, loops, arrays, arrays of objects, static attributes, and static methods.

Part I

For this part, you are required to design and implement a **Consumer** class based on the below mentioned specifications:

- A **Consumer** object has nine attributes, name (String), street number (int), street name (String), city (String), postal code (String), age (int), gender (String), marital status (Enum), education (Enum).
  - Upon creation of a **Consumer** object, it must immediately be initialized with valid values for all the nine attributes. (Hint: use constructors.).
  - The design should allow enough flexibility so that the value of any of these attributes can be modified later. For example, it should be possible to create a **Consumer** object with a given city and then change it later. The design should also allow a user to obtain value of any attributes. (Hint: use accessors & mutators.)
  - The design should allow all information of an object to be displayed at once through utilization of System.out.print() method. (Hint: use toString() method).
  - It is required to know how many **Consumer** objects have been created. For that, you need to add a method, called getNumberOfConsumers(), to the class. This method must return total number of **Consumer** objects created prior to invocation of this method. The method would simply return 0 if no **Consumer** has been created by the time the method is called. (Hint: use Static – You are allowed to add other attributes to the class.).
  - It is required to compare two **Consumer** objects for similarity. Two Consumer objects are similar if they have the same age, gender, and education. (Hint: use equals() method).
  - It is required to display any **Consumer** object (all info of that object) using System.out.println() method. (Hint: use toSting() method).

Part II

You are hired by a marketing company to write a software application that helps them keep track of their consumer base.

Write a driver program that will contain the **`main()`**`method` and will perform the following:
(Note: You can have the main function in a separate driver file, or in the same file if you prefer)

- Display a welcome message.
- Prompt the user for maximum number of consumers the company can handle. Create an empty array, called `consumerDatabase` that will keep track of created **Consumer** objects.
- Display a main menu (figure 1) with the following choices and keep prompting the user until they enter a number between 1 and 5 inclusive:

What do you want to do?
1. Enter a new Consumer (password required)
2. Change information of a Consumer (password required)
3. Display all Consumers similar to a given consumer
4. Display all Consumers with given age and location
5. Quit

Please enter your choice >

Figure 1. Main menu

- When **option 1** is entered:
  - Prompt the user for his/her password. (Make sure you have a constant variable containing the password `"password"` – do not use any other password as it will be easier for the grader to check your assignments). The user has a maximum of 3 attempts to enter the correct password. After the 3rd wrong attempt, the main menu in figure 1 is displayed again. Additionally, after this process is repeated 4 times (*i.e.* after consecutive 12 failed attempts), the program must display a message: `"Program has detected suspicious activity and will terminate immediately!"`, then the program must exit.

  - If the correct password is entered, ask the user how many consumers he/she wants to enter. Check to make sure that there is enough space in the `consumerDatabase` (array of `Consumer`) to add that many consumers. If so, add them; otherwise inform the user that he/she can only add the number of remaining places in the array. (How the consumer information will be input/entered by the user, is up to you).

- When **option 2** is entered:
  - Prompt the user for his/her password. (Make sure you have a constant containing the password `"password"` as a constant – do not use another password). Again, the user has 3 attempts to enter the correct password. However, after the 3rd wrong attempt entry, the program must exit with a message `"program terminated due to safety reasons"` (notice difference in behavior to the one above).

  - Once the correct password is entered, user is asked which consumer he/she wishes to update. The `Consumer` number is the index in the array `consumerDatabase`. If there is no `Consumer` at the specified index location, display a message asking user if he/she wishes to re-enter another consumer, or quit this operation and go back to the main menu. If the consumer is present, display the current information of that consumer in the following format:

    Consumer: # x (index of the consumer in the `consumerDatabase` array)
    Name: name of the consumer
    Location: Street number, Street name, City name, Postal Code
    Age: Age value
    Gender: Gender Value
    Marital Status: Marital Status Value

    Then ask the user which attribute they wish to change by displaying the following menu.

What information would you like to change?
1. Consumer name
2. Location
3. Age
4. Gender
5. Marital Status
6. Quit

Please enter your choice >

Figure 2. Update menu

- Once the user has entered a correct choice, make the changes to the attribute then display again all the attributes on the screen to show that the attribute has been changed. Keep prompting the user for additional changes until the user enters 6. Each time the user is prompted for a choice make sure that a number from 1 to 6 is entered, otherwise keep prompting until a valid number is entered. (Ensure that the user can change any of the choice 1 to 5 on figure 2).

➢ When **option 3** (in the main menu shown in figure.1) is entered, prompt the user to enter age, gender, and education values. You then need to display the information of all consumers similar to the one entered by the user. (Hint: You may use a static method, for instance called **`findConsumersBy`**, which accepts enter age, gender, and education values and then performs the needed search).

➢ When **option 4** (in the main menu shown in figure. 1) is entered, prompt the user to enter values for age and postal code. You then need to display all consumers that have same age and postal code as entered by the user. (Hint: You may use a static method, called **`findConsumerByAgeNGender`**, which accepts age and postal code, then performs the needed search).

➢ When **option 5** (in the main menu shown in figure. 1) is entered, display a closing message, and end the driver.

General Guidelines When Writing Programs:

- Include the following comments at the top of your source codes
  ```
  // -------------------------------------------------------
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // -------------------------------------------------------
  ```
- In a comment, give a general explanation of what your program does. As programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

JavaDoc Documentation:

Documentation for your program must be written in **javaDoc**.
In addition, the following information must appear at the top of each file:

```
Name(s) and ID(s)   (include full names and IDs)
COMP249
Assignment #        (include the assignment number)
Due Date            (include the due date for this assignment)
```

Assignment 1 Submission

- For this assignment, you are allowed to work individually, or in a group of a maximum of 2 students (*i.e.* you and one other student). You and your teammate must however be in the same section of the course. Groups of more than 2 students = zero mark for all members!
- Only electronic submissions will be accepted. Zip the source codes. (Please use WINZIP).
- Students will have to submit their assignments (one copy per group) using the Moodle system. Assignments must be submitted in the right folder of the assignments. **Assignments uploaded to an incorrect folder or submitted via email will not be graded and will result in a zero mark. No resubmissions will be allowed. Please make sure you submit on Moodle system making use of the Assignment1 submission link under the week of September 26.**
- **Naming convention for zip file:** Create one zip file, containing all source files and produced documentations for your assignment using the following naming convention:

- The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number respectively. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a1_Mike-Simon_123456.zip.* If working in a group, the name should look like: *a1_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*

- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded, and all others will be disregarded.
- If working in a team, only one of the members can upload the assignment. Do **NOT** upload the file for each of the members!

Evaluation Criteria for Assignment 1 (10 points)

Total	**10 pts**
Documentations	**1 pt**
JavaDoc documentations	1 pt
Part I (Class **Consumer**)	**3 pts**
Default & copy constructors	1 pt
Accessor/mutator method for static attribute	1 pt
equals, toString and static attributes/methods	1 pt
Part II (Driver & static methods & other methods)	**6 pts**
Handling of password	1 pt
Handling of option 1	1 pt
Handling of option 2	1 pt
Handling of option 3	1 pt
Handling of option 4	1 pt
Handling of option 5	1 pt