AP/ITEC 2610 Fall 2020 Sections C/D

Assignment 1

Task Description:

Task 1: BetterHealth is a membership-based fitness club in Toronto. Customers can purchase a membership that provides a fixed number (e.g. 100) of free visits to the club upon registration. When the initial number of visits runs out, they can purchase additional visits as needed. All types of fitness equipment and all fitness classes offered in the club are open for this membership. Please write a Java program (**Membership**.java) for membership management.

The Membership class should have the following public interface:

1. `public Membership(String name, int visits)`: A constructor to set up the member's name and the initial number of visits allowed. The parameter `visits` is supposed to be positive (if not, initialize the number of visits to 0).
2. `public String getName()`: An accessor to return the member's name.
3. `public int getRemainingVisits()`: An accessor to return the number of remaining visits.
4. `public boolean isValid()`: A method to decide if one's membership is still valid (`true` if the number of remaining visits is greater than 0; `false` otherwise).
5. `public boolean topUp (int additionalVisits)`: A method to add additional visits (represented by `additionalVisits`) to the `remainingVisits`. Returns `true` if the top-up succeeds and `false` otherwise. The top-up succeeds only If the given `additionalVisits` is non-negative.
6. `public boolean charge()`: A method that deducts 1 from the number of remaining visits when the membership is valid. Returns `true` if the charge succeeds and `false` otherwise. The charge succeeds only if the membership is valid before charging.
7. `public boolean equipmentAllowed()`: A method that indicates whether the equipment at the gym is available to this kind of membership. Returns `true` for now; to be updated in Task 2.
8. `public boolean classesAllowed()`: A method that indicates whether the fitness classes at the gym are available to this kind of membership. Returns `true` for now; to be updated in Task 2.

Write a tester program **Task1MembershipTester** to test the class you have written. Follow the examples given in class (e.g., BankAccountTester), and create an object

from the class **Membership** above. Test the public interface of your class thoroughly (each public method must be tested at least once). For accessor methods, print out the expected return value and the return value of your method (see the BankAccountTester example) to compare. This tester class is not to be submitted for marking, but it is beneficial for you to make sure your Membership class works as intended.

Task 2: The club now introduces a new type of membership called **PremiumMembership**. It is valid within one year from the date of purchase for unlimited visits. Such members can use fitness equipment, but cannot attend fitness classes. In addition, pool access is available for this type of membership.

Since now we have two types of memberships (and imagine down the road we might have more), you should implement a hierarchy of classes to represent the different types of membership. At the top of the hierarchy, you must define an abstract class called **BaseMembership** defining the methods common to different types of memberships. BaseMembership has the following public interface:

1. `public BaseMembership(String name, String type)`: A constructor to set up the member's name and member's type.
2. `public String getName()`: An accessor to return the member's name.
3. `public String getType()`: An accessor to return a string that represents the type of membership.
4. `public abstract boolean isValid()`: An abstract method that should be overridden in the subclasses.
5. `public boolean equipmentAllowed()`: Set an initial return value and override in the subclasses.
6. `public boolean classesAllowed()`: Set an initial return value and override in the subclasses.

You must then define classes StandardMembership and PremiumMembership that inherit from BaseMembership, defining additional methods as needed. **StandardMembership** should be implemented to model the type of membership in Task 1.

The class PremiumMembership should have the following public interface:

1. `public PremiumMembership(String name, String startDateString)`: A constructor to set up the member's name and his start date. To transform `startDateString` to a Date type, a method `stringToDate` is provided in the hint below.

2. `public boolean isValid()`: A method to decide if one's membership is still valid (If the current time is less than one year from the start time, it is valid). Please see the hint below.
3. `public boolean equipmentAllowed()`: Return true.
4. `public boolean classesAllowed()`: Return false.
5. `public boolean poolAllowed()`: Return true.

Hint:

Java uses the number of milliseconds since January 1, 1970, 00:00:00 GMT to represent time/date. Use 'new java.util.Date()' to create a Date object of the current time in milliseconds. 'System.currentTimeMillis()' is used to get the current time in milliseconds; 'getTime()' is used to return the long milliseconds of a Date object; One year in milliseconds can be presented: *long MILLIS_YEAR = 360 * 24 * 60 * 60 * 1000L*;

```
/**
* Utility method to transform a string represented date to Date object.
* E.g., "2010-01-02" will be transformed to Sat Jan 02 00:00:00 GMT 2010
*/
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
public static Date stringToDate(String dateString) {
   DateFormat f = new SimpleDateFormat("yyyy-MM-dd");
   Date date = null;
   try {
      date= f.parse(dateString);
   } catch (ParseException e) {
       e.printStackTrace();
   }
   return date;
}
```

Similarly, write a tester program **Task2MembershipTester** to test the classes you have written in Task 2. Create objects from the classes **StandardMembership** and **PremiumMembership** above. This tester class is not to be submitted.

What to submit:

Please complete the Task 1 and Task 2 according to the requirements, documenting them properly using JavaDoc. **Please define the classes using the exact names given here.** Compile and test your code under the command line environment (e.g., by executing Terminal under macos or cmd under Windows).

- ➔ Submit a single zipped archive named **A1.zip** containing all your **4** source files (containing **Membership**.java, **BaseMembership**.java, **StandardMembership**.java, and **PremiumMembership**.java). Nothing else (including the tester programs) should be submitted.
- ➔ **DO NOT submit .class files! .java files ONLY. DO NOT use other archival formats such as .RAR or .7z.**

Important Note:

(1) Your assignment **will be given a zero** mark if only the compiled files (.class files) are submitted. Please make sure to submit the source files (.java files).

(2) Please make sure your code compiles under the command line (i.e., without an IDE). Do not put any package statement at the beginning of your source file. If you are using an IDE, this is especially important because some IDEs put your code under a particular package for your project. Any code that does not compile under the command line can only receive 20/100. Remarking requests like "...but the code works on my computer/in my IDE" will **not** be entertained.

Marking Scheme:

```
Style (variable naming, indentation, & Layout)  _____/10

JavaDoc Comments                                _____/10

Code Compiles?                                  _____(yes/no)

Successful Execution of Test Cases              _____/80

Total                                           _____/100
```

According to this marking scheme the maximum mark you can get for code that does

not compile is 20/100.