Project #2 – 130 points

Due Date

Monday, October 17, 11:59pm.

Submission

1. Submit **a single zipped file** to Canvas. The zipped file MUST include the following grading items.
   - Source folder **src** [80 points]
     (a) Create a package to organize the source files *.java; use all lowercase letters for the package name
     (b) MUST include all team members' names using the **@author** tag in the comment block on top of EVERY Java class, or you will **lose 2 points**.
   - Class Diagram [10 points]
   - **JUnit** test classes
     (a) Date class, test the isValid() method. [10 points]
     (b) Premium class, test the membershipFee() [5 points]
     (c) FitnessClass class, test the methods for adding/removing members/guests. [20 points]
   - **Javadoc** folder, including all the files generated. [5 points]
2. The submission button on Canvas will disappear after **October 17, 11:59pm**. It is your responsibility to ensure your Internet connection is good for the submission. **You get 0 points** if you do not have a submission on Canvas. Submitting the project through the email will not be accepted.

Project Description

Your team will continue to work on the next phase of the software developed in Project 1. The fitness chain provides 3 membership options – standard, family, and premium. The software shall run the membership fee dues for all the members. The membership fees are listed in the table below.

Membership Type	One-Time Fee	Fee Schedule	Fitness Classes check-in
Standard	$29.99	$39.99, per month pay quarterly	Membership location only
Family		$59.99 per month, pay quarterly	• Any location • 1 family guest pass, membership location only
Premium	waived	One month free of Family membership rate, pay annually	• Same as Family, and • 3 family guest passes, membership location only

The three different fitness classes, Pilates, Spinning and Cardio, are now offered in person at different gym locations. Members are required to check in before they participate in the classes. To accommodate the increase of memberships, the fitness chain is adding more fitness classes and hiring additional instructors. However, members with the standard membership can participate in the classes held at the membership location only. Members with family and premium memberships can participate in the classes held at any locations. Family guest passes can only be used for the fitness classes held at the membership location.

There are currently 5 instructors teaching the classes, Jennifer, Kim, Denise, Davis and Emma. The fitness chain will provide the class schedule in a text file. This allows the change of schedule independent to the software and loading of the schedule to the database in batch. Currently, fitness classes are regularly offered during a day in the morning, afternoon, and evening, at 9:30, 14:00 and 18:30, respectively. Each fitness class can be uniquely identified by the

class name, instructor, and location. The software shall load the class schedule from an external text file to the system so the members can check in for the classes.

To run the fees, additional information is needed at the time of adding the members to the database. In addition to keeping the functionalities implemented in Project 1, you will make the changes to add or change the relevant commands to accommodate the new requirements in this project. The commands R, P, PD, PC, PD, S should work the same way as in Project 1. New commands or the commands that need to be changed are listed below.

- **LS** command, a new command to load the fitness class schedule from the file **classSchedule.txt** to the class schedule in the software system.
- **LM** command. There is historical member information needed to be imported to the database. This is a new command to load a list of members from the file **memberList.txt** to the member database.
- **A** command, to add a member with the standard membership to the member database. The expiration date should set to expire 3 months later. **NOTE:** the expiration date will be based on the date you add the member. The expiration dates in the file Project2_expectedOutput.txt I provided are based on the date I generated the output. The expiration dates you generated should be different.
- **AF** command, to add a member with the family membership to the member database. The expiration date should set to expire 3 months later. **NOTE:** see the NOTE above.
- **AP** command, to add a member with the premium membership to the member database. The expiration date should set to expire one year later.
- **PF** command, a new command to print the list of members with the membership fees. Let's assume the membership fees are for the next billing statement, regardless of the current membership expiration dates.
- **C** command, members check-in for a fitness class.
- **CG** command, family guest check-in for a fitness class; must keep track of the remaining number of guest passes.
- **D** command, a member is done with a fitness class and checking out.
- **DG** command, a guest is done with a fitness class, and checking out; must keep track of the remaining number of guest passes.

Project Requirement

1. You MUST follow the Coding Standard and Ground Rules posted on Canvas under Week #1 in the "Modules". **You will lose points** if you are not following the rules.
2. You are responsible for following the Academic Integrity Policy. See the **Additional Note #13** in the syllabus.
3. You must preserve all the functionalities developed in Project 1 to pass the test cases. There are test cases in the file **Project2_testCases.txt** for you to test the project. The associated output is in **Project2_expectedOutput.txt.** The graders will be using the sample test cases to test your project. Your project should be able to take the test cases from the console in the same order line by line without getting any exceptions and without terminating abnormally, including the empty lines. You will **lose 2 points** for each incorrect output or each exception causing the project to terminate abnormally.
4. Each Java class must go in a separate file. **-2 points** if you put more than one Java class into a file.
5. Your program MUST handle bad commands; **-2 points** for each bad command not handled, with a **maximum of losing 6 points**.
6. You are not allowed to import any Java library classes, EXCEPT the **Scanner**, **File, StringTokenizer, Calendar, DecimalForamt**, and necessary **Java Exception classes**. **You will lose 5 points** for each additional Java library class imported, with a **maximum of losing 10 points**.

7. You CAN use the Java library class **ArrayList** in the FitnessClass class for this project. You CANNOT use ArrayList in any other classes. You CANNOT use any other classes in the Java Collections. **You will get 0 points for this project** for any of the violations above.
8. When you import Java library classes, be specific and DO NOT import unnecessary classes or import the whole package. For example, `import java.util.*;`, this will import all classes in the `java.util` package. You will **lose 2 points** for using the asterisk "*" to include all the Java classes in the `java.util` package, or other java packages, with a **maximum of losing 4 points.**
9. You MUST create a **Class Diagram** for this project to document the software structure. The diagram is worth 10 points. You will **lose the 10 points** if you submit a hand-drawing diagram.
10. You are not allowed to have redundant code in this project; **reuse the code** as much as possible by using the **super** keyword, **-2 point** for each violation, and **-2 points** for each violation below.
    - Unused code: you write code that was never called or used.
    - Duplicate code segments for the same purpose in more than one places/classes.
    - Define common instance variables in each of the subclasses while they should be defined in the superclass.
    - Define specific instance variables in the superclass while they are not used by all the subclasses.
    - Define common methods in each subclass instead of defining it in the superclass.
    - Not defining instance variables as private.
11. You must define classes with the following inheritance relationships. **-5 points** for each class missing, or incorrect inheritance structure.
    - **Member** class from Project 1 defines the standard membership and the common instance variables and instance methods of the subclasses Family and Premium. You must include the following method or **lose 3 points.** You CANNOT change or add additional instance variables, or **-2 points** for each violation.

      ```
      public double membershipFee() { }
      ```

    - **Family** class extends the Member class and includes specific data and operations to a family membership; you must override the membershipFee() method, or **lose 3 points**.
    - **Premium** class extends the Family class and includes specific data and operations to a premium membership; you must override the membershipFee() method, or **lose 3 points**. You may need to override other public methods defined in Family class.
12. **Polymorphism is required**. You CANNOT use the getClass() method for checking the class type listed in #11 above, or you will **lose 10 points.** You can use **instanceof** keyword to check the actual type of an object. Common instance variables and public methods must be defined in the superclass. Override the public methods in the superclass if necessary. If you are overriding any methods, you must add the annotation **@Override** on top of the methods. **-2 points** for each violation.
13. In addition to the classes in #11, you must add or modify the classes listed below. **-5 points** for each class missing. You CANNOT perform I/O in all classes, EXCEPT the GymManager class, and the print methods in the MemberDatabase class and ClassSchedule class. **-2 points for each violation.** The floating-point numbers must be displayed with 2 decimal places. **-1 point for each violation.**

    (1) **ClassSchedule** class. You cannot change or add the instance variables listed below, or **-2 for each violation.** You should add constructors and methods. You CANNOT use **ArrayList** in this class, or you **will lose 10 points.** The public methods you defined in this class must either take no parameter or take a single parameter in FitnessClass type, such as (FitnessClass fitnessclass), or you will **lose 2 points** for each violation.

    ```
    public class ClassSchedule {
        private FitnessClass [] classes;
        private int numClasses;
    }
    ```

(2) **MemberDatabase** class. You cannot change or add instance variables in this class, or **-2 for each violation.** All the pubic methods in this class must either take no parameter or has a single parameter, which takes only an instance of Member class, such as (`Member member`), or you will **lose 2 points** for each violation. An instance of this class will hold a list of members with different memberships.

(3) **FitnessClass** class. You should make necessary changes to this class in order to accommodate the new features at this phase. You CAN use ArrayList in this class. This is the only class in Project 2 you can use the Java ArrayList class. The public methods you defined in this class must either take no parameter or take a single parameter in Member type, such as (Member member), except the equals() method, or you will **lose 2 points** for each violation.

14. **JUnit test class.** Write code to test isValid() method in the Date class, membershipFee() method in the Premium class and the methods you use to add/remove members, add/remove guests to/from the fitness class roster in FitnessClass class. You must use a descriptive name for all test methods and add proper comments, or **-2 points** for each violation.
15. **Class Diagram.** Create a class diagram to document your software structure for Project 2. Hand-drawing the diagram is not acceptable. You can follow the links below to create a class diagram online and save it to your device as a picture, which can be included in your submission. You can also use other UML tools if you like.
    - http://draw.io/ → Create New Diagram → UML → Class Diagram
    - https://online.visual-paradigm.com/app/diagrams/ → Create Something → Class Diagram
16. You are required to **generate the Javadoc** after you properly commented your code. Your Javadoc must include the documentations for the constructors, private methods, and public methods of all Java classes. Generate the Javadoc in a single folder "**doc**" and include it in the zip file to be submitted to Canvas. **Please double check your Javadoc after you generated it** and make sure the descriptions are NOT EMPTY. You will lose points if any description in the Javadoc is empty. You will **lose 5 points** for not including the Javadoc.