# Object-Oriented Programming Lab#8, Spring 2024

## Today's Topics

- Inheritance
- encapsulation
- method override
- method overload
- subclass polymorphism
- abstract class
- Add project reference

## A Fitness Tracking System

Create a **fitness tracking system**. The goal of system is to **store data about the trainers, trainees** and **track the activities**. These include the workout plan a trainee and his/her progress such number of steps taken, push-ups, distance ran, and other fitness metrics. To make it easy for users to monitor progress, create a fitness tracking system. To keep it simple we will work with minimal functionalities. There will **be 3 types of users** of this system; **admin** (to do administrative job), **trainer**, and **trainee**. There will be different functionalities for different users.

The system will have the following functionalities.

1. There are 3 types of users for this system; **admin, trainer and trainee**. A user can log in as an admin, trainer or trainee. For simplicity, we can **skip** the log-in part and add an option or button for logging in as different types of users.
2. The system will have the following functionalities.

   Admin:
   a. Login/Logout
   b. Add new trainer info
   c. Add new trainee info
   d. View the list of trainers
   e. View the list of trainees

   Trainer:
   a. Login/Logout
   b. Set workout plan for a trainee under him
   c. Add a workout item from the plan
   d. View the list of trainees under him
   e. View the progress of a trainee.
   f. View the trainee requests

g. Accept a request

Trainee

a. Login/Logout

b. View the list of trainers

c. Send a request to a trainer.

d. View the workout plan and progress.

e. Start a specific workout

f. Complete a specific workout

## *What you need to do: (Note: Do not use default package)*

**You need 2 projects for this Lab.**

Create a Project name **FitnessLibrary (and do the following)**

1. Create the following User-defined Exception

```
public class InvalidUserException extends Exception {
   public InvalidUserException(String id, String userType) {
         super(String.format("%s with ID:%s is not a valid user.", 
userType, id));
   }
}
```

2. Create a **WorkOut** class:

   *a.* Add 3 private instance variables: ***name, type, status (Planned / InProgress/ Completed)***

   b. Add a parameterized constructor and pass parameters for **name** and **type**. Initialize the respective attributes and set **status** to "**Planned**".

   c. Create **getter/setter** methods for all attributes. and **toString**() method.

   d. Add the following methods

      i. `public void startWorkOut()`

      Inside the method set **status** to "**In Progress**"

      ii. `public void completeWorkOut()`

      Inside the method set **status** to "**Completed**"

      iii. Override the **equals** method and return true if both name and type match.

      `public boolean equals(Object obj)`

3. Create an **abstract** class named **User**.

   *a.* Add following **private** instance variables:

```
   private String name, fitnessId;
```

```
  private int age;
  private float weight, height;
private LocalDate joiningDate; // under java.time package
```

a. Add a parameterized constructor and pass parameters for all attributes except ***fitnessId, and joiningDate***. Initialize all attributes with the respective parameters. Generate and assign a **4 digits** random number to ***fitnessId.*** Assign LocalDate.now() to joiningDate.

b. Add the following methods:

i. `public void addPrefixToId(String prefix)`
   – inside this method prepend the **prefix** with **fitnessId**.

ii. Add getter/setter for all attributes. Setter method of joining date will be little different, you can use the code below.

```
public void setJoiningDate(String joiningDate) {
DateTimeFormatter format=
DateTimeFormatter.ofPattern("dd/MM/yyyy"); LocalDate date =
LocalDate.parse(joiningDate, format); this.joiningDate = date;
 }
```

iii. Override toString() – return values of all attributes as a concatenated string format.

**Code to convert LocalDate to String**

DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");
String joinDate = joiningDate.format(formatter);

iv. Add the following abstract method
   **public abstract** String toString(**boolean** details) ;

v. Override the **equals** method and return true if both **fitnessId** matches.
   `public boolean equals(Object obj)`

4. Create a **Trainer** class:

a. Make this class a subclass of **User** class.

b. Add additional private instance variables; **yearOfExperience, ArrayList<Trainee> myTrainees**, **ArrayList<Trainee> myTraineeRequests**.

c. Add parameterized constructor as below
   **public** Trainer(String name, **int** age, **float** weight, **float** height, **int** yearOfExperience) - Call the parent's constructor, initialize ***yearOfExperience***. Add "11-" as the prefix of the id that is generated from parent class using the ***addPrefixToId*** method. Instantiate the **myTrainees** and **myTraineeRequests** object.

d. Add the following methods.

i. Add **getter** methods for all attributes and override toString().

ii. `public Trainee findTrainee(String traineeId)`

inside the method, loop through the **myTrainees** list and look for trainee with matching id. If the trainee is found return the object. If the trainee is not found return **null**.

iii. `public Trainee findTraineeRequest(String traineeId)`

inside the method, loop through the **traineeRequests** list and look for trainee with matching id. If the trainee is found return the object. If the trainee is not found return **null**.

iv. `public void addTrainee(Trainee trainee)`

inside the method, add the trainee parameter to **myTrainees** list.

v. `public void addTraineeRequest(Trainee trainee)`

inside the method, add the trainee parameter to **myTraineeRequests** list.

vi. `public void acceptTraineeRequest(String traineeId)`

Call **`findTraineeRequest`** method with the **traineeId**. If the trainee is found in the list, remove the object from the **traineeRequests** list and add to the **trainees** list.

vii. `public void addWorkoutForTrainee(Trainee trainee, WorkOut workOut)`

if the **trainee** parameter is in the **trainees** list (use findTrainee methos), call the ***addWorkOutItem***(…) method using the **trainee** variable. Otherwise throw an Exception with message "[Trainer] doesn't have the authority to assign work out item." where [Trainer] is the name of the Trainer.

**`Override following 2 methods.`**

viii. `public String toString()`

Call the toString() method of parent class and concatenate the value of **yearOfExperience** varaiable. Return the concatenated String

ix. `public String toString(boolean details)`

if details is false, call toString() method. If it is true, call toString() and concatenate the values of remaining attributes

5. Create a **Trainee** class:

a. Make this class a subclass of **User** class.

b. Add additional private instance variables; **Trainer myTrainer, ArrayList<WorkOut> workOutPlan**.

c. Add parameterized constructor

**public** Trainee(String name, **int** age, **float** weight, **float** height) - Call the parent's constructor. Add "22-" as the **prefix** of the **fitnessId** that is generated from parent class using the ***addPrefixToId*** method. Also initialize the **workOutPlan** ArrayList.

d. Add getter method for all attributes and setter method for **myTrainer**.

e. Add the following methods.

i. `public void addWorkOutItem(String name, String type)`

Inside the method, Create an ***WorkOut*** object using the workout parameter and add the object to ***workOutPlan*** ArrayList.

ii. `public void startWorkOut(String name, String type)`

Inside the method, loop through the ***workOutPlan*** ArrayList and if the matching workout is found set the status to "In Progress"

iii. `public void completeWorkOut(String name, String type)`

Inside the method, loop through the ***workOutPlan*** ArrayList and if the matching workout is found set the status to "Complete"

**Override following 2 methods.**

iv. `public String toString()`

Call the toString() method of parent class and concatenate the value of myTrainer varaiable. Return the concatenated String

v. `public String toString(boolean details)`

if details is false, call toString() method. If it is true, call toString() and concatenate the values of remaining attributes

6. Create a class name "**FitnessCenter**""

a. Add the following private attributes

```
private String name;
private     ArrayList<Trainer>     trainers     =     new
ArrayList<>();  private  ArrayList<Trainee>  trainees  =
new ArrayList<>(); private User loggedInUser = null;
```

b. Create a parameterized constructor as pass the name of the centre.

Inside the constructor, initialize the **name** attribute.

c. Add getter methods for attributes and setter method for **loggedInUser**.

d. Add the following methods

i. `public String addTrainer(String name, int age, float weight, float height, int yearOfExperience)`

Create an object of **Trainer** using the parameters and then add the object to **trainers** list. Also return the **fitnessId** of the **Trainer**.

ii. `public String addTrainee(String name, int age, float weight, float height)`

Create an object of **Trainee** using the parameters and then add the object to **trainees** list. Also return the **fitnessId** of the **Trainee**.

iii. `public Trainer findTrainer(String id) throws InvalidUserException`

Inside the method, loop through the list of the **Trainer** (***trainers*** *instance variable*) and find the Trainer whose **fitnessId** match with the parameter ***id***. If the **Trainer** is found return the object otherwise throw ***InvalidUserException*** and pass the id and "Trainer" as parameter.

iv. `public Trainee findTrainee(String id) throws InvalidUserException`

Inside the method, loop through the list of the **Trainee** (***trainees*** *instance variable*) and find the Trainee whose **fitnessId** match with the parameter ***id***. If the **Trainee** is found return the object otherwise throw ***InvalidUserException*** and pass the id and "Trainee" as parameter.

v. `public void requestForTrainer(String trainerId)`

If loggedInUser is a Trainee, do the following
Call FindTrainer and pass the trainerId.
If the trainer is found, cast the loggedInUser object to Trainee and call requestForTrainer() using the trainee object.

-

vi. `public void acceptTraineeRequest(String traineeId)`

If loggedInUser is a Trainer, cast the loggedInUser object to Trainer and call acceptTraineeRequest () using the trainer object.

## Now create a new project FitnessApplication (and do the following).

1. Add project reference of the previous project.
2. Create an **application class** (that has the main method) named "**FitnessApp**" which will have the **main** method. Create an object of the **FitnessCenter** class and assign to **myFitness** variable. ○ In the main method, ask if the user is **admin, trainer** or a **trainee**.

   Ask to enter his/her id. For now, as we do not have any admin user, you can use a hardcoded id for admin. For Trainer or Trainee, once he/she enters the id call ***findTrainer(…)*** or ***findTrainee(…)*** depending on the role. If the Trainer/Trainee is found, call the setLoggedInUser(User loggedInUser) setter method to set the loggedInUser. You can use this variable to access different functionalities.

   ○ If the user is an admin show the following menu. Depending on the menu take necessary user input and call appropriate method using the **myFitness** variable.

   - Login/Logout
   - Add new trainer info
   - Add new trainee info
   - View the list of trainers
   - View the list of trainees

- If the user is a trainer or trainee, ask for his/her id. If the id is valid, show the following employee Menu. Depending on the menu take necessary user input and call appropriate method using the **myFitness** variable.
  - Trainer:
    - Login/Logout
    - Set workout plan for a trainee under him
    - Add a workout item from the plan
    - View the list of trainees under him
    - View the progress of a trainee.
    - View the trainee requests
    - Accept a request
  - Trainee
    - Login/Logout
    - View the list of trainers
    - Send a request to a trainer.
    - View the workout plan and progress. •
    Start a specific workout
    - Complete a specific workout

**Problem 3: Implement User-defined Exception in your project.**

1. Create a user-defined exception **InvalidUserException** will take a String type parameter **userId** and set the exception message to "user with **userId** is not a valid user.", here **userId** is the parameter passed to the constructor which represents the id of an user.

2. Update the **Trainer** Class.
   a. Update the "findMyTrainee" methods. Instead of returning null, throw the **InvalidUserException** when the Trainee is not found.
   b. Update all methods which are calling the findTrainee(String id). Add throws in those method headers.

3. Update the **FitnessCenter** Class.
   a. Update the following methods. Instead of returning null, throw the **InvalidUserException** when the Trainer/Trainee is not found.
      i. findTrianer(String id)
      ii. findTrainee (String id)

4. Update the **FitnessCenterApp** Class.
   a. Use try/catch wherever an exception (**InvalidUserException or any other Exception**) is thrown.