

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
MOCK PRACTICAL ASSESSMENT I FOR
Semester 2 AY2022/2023

CS2030S Programming Methodology II

March 2023

Time Allowed 90 minutes

INSTRUCTIONS TO CANDIDATES

1. This mock practical assessment consists of **one** question. The total mark is 0: 0 marks for design; 0 for style; 0 for correctness. Style and correctness are given on the condition that reasonable efforts have been made to solve the given tasks.
2. This is an OPEN BOOK assessment. You are only allowed to refer to written/printed notes. No online resources/digital documents are allowed, except those accessible from the PE nodes (peXXX.comp.nus.edu.sg) (e.g., man pages are allowed).
3. You should see the following in your home directory.
 - The files `Test1.java`, `Test2.java`, and `Test3.java` for testing your solution.
 - The directories `inputs` and `outputs` contains the test inputs and outputs.
 - The file `GymManagement.java` for you to improve upon, and the file `pristine/GymManagement.java` for your reference.
 - The files `checkstyle.jar`, `cs2030_check.xml`, and `test.sh` are given to check the style of your code and to test your code.
 - You may add new classes/interfaces as needed by the design.
4. Solve the programming tasks by editing `GymManagement.java` and creating any necessary files. You can leave the files in your home directory and log off after the assessment is over. There is no separate step to submit your code.
5. Only the files directly under your home directory will be graded. Do not put your code under a subdirectory.
6. Write your student number on top of EVERY FILE you created or edited as part of the `@author` tag. Do not write your name.
7. To compile your code, run `javac -Xlint:unchecked -Xlint:rawtypes *.java`. You can also compile the files individually if you wish.
8. To run all the test cases, run `./test.sh`. You can also run the test individually. For instance, run `java Test1 < inputs/Test1.1.in` to execute `Test1` on input `inputs/Test1.1.in`.
9. You can check the style of individual file with `java -jar checkstyle.jar -c cs2030_checks.xml <FILENAME>`.

IMPORTANT: If the submitted classes or any of the new files you have added cannot be compiled, 0 marks will be given. Make sure your program compiles by running

```
javac -Xlint:unchecked -Xlint:rawtypes *.java
```

before submission.

Overview

You are building a system to help with the management and administration the newly opened SoC gym. This system will need to keep track of the equipment and people in the gym. A gym has a capacity, that is the number of people that can be in the Gym. If a person comes to the Gym and it is at capacity, they will not be allowed to enter. A trainer can train a customer using some equipment.

The Class GymManagement

The class GymManagement has a constructor which takes in no arguments. It can be used in the following way:

```
GymManagement gym = new GymManagement();
```

Types of Gym Equipment

The class can handle two different types of gym equipment.

- Type 0: A Treadmill which has a speed (a double)
- Type 1: A Dumbbell which has a weight (a double)

Types of People

The class can handle two different types of people.

- Type 0: A Trainer who can train other Customers
- Type 1: A Customer who can be trained

Types of Actions

The class can handle three different types of actions.

- Type 0: A Customer entering the gym
- Type 1: A Trainer training a Customer with a piece of gym equipment
- Type 2: A Trainer finishing training

Input File

The equipment, people, and actions of people at the gym are stored in a file format. This file format is loaded into GymManagement through the constructor and has the following format.

- The first line of the file contains an integer n_1 , which is the number of pieces of equipment in the gym.
- The next n_1 lines contain information about the equipment. Each of these n_1 lines contains two fields, separated by a space.
- The first field is always an integer and indicates the types of equipment (either 0 for Treadmill, or 1 for Dumbbell)
- The second field is a double which is the property of equipment, either a speed or a weight.
- The next line of the file contains an integer n_2 , which is the number of people in the gym.
- The next n_2 lines contain information about these people. Each of these n_2 lines contains two fields, separated by a space.
- The first field is always an integer and indicates the types of person (either 0 for Trainer, or 1 for Customer)
- The second field is a string which is the name of the person.
- The next line of the file contains an integer n_3 , which is the capacity of the gym.
- The next line of the file contains an integer n_4 , which is the number of actions.
- The next n_4 lines contain information about these actions. Each of these n_4 lines contains either two fields or 4 fields, separated by a space.

- The first field is always an integer and indicates the types of action (either 0 for Entering, 1 for Training, 2 for Finishing Training)
- The second field is either the index of the customer (for Entering action), or the index of the trainer for other actions. This field is an integer that indicates the index of the customer that is being trained.
- The third field applies only if the action is Starting Training. This field is an integer that indicates the index of the customer that is being trained.
- The fourth field applies only if the action is Starting Training This field is an integer that indicates the index of the equipment that is being trained with.

You can assume that the given test data follows the input format correctly.

Listing Equipment

There is a method to print out the equipment list of the gym `printEquipment`.

The method `printDescriptions` takes in no arguments and returns nothing. It prints the description of each Equipment in enumerated order in the same order as they appear in the input files.

Consider the following input file, which specifies a gym with a capacity of two, with two equipment (A Treadmill and a Dumbbell), and zero people and actions.

```
2
0 3.0
1 2.5
0
2
0
```

Using the following code excerpt:

```
GymManagement gym = new GymManagement();
gym.printEquipment();
```

Will print out:

```
Equipment: Treadmill at speed 3.0
Equipment: Dumbbell with weight 2.5
```

Listing People

There are two types of people in the gym, customers and trainers.

There is a method to print out the people list `printPeople`.

The method `printPeople` takes in no arguments and returns nothing. It prints the description of each person in enumerated order in the same order as they appear in the input files.

Consider the following input file, which specifies a gym with a capacity of two, with two equipment, and four people (two trainers and two customers) and zero actions.

```
2
0 3.0
1 2.5
4
0 Frank
0 Sam
1 Bob
1 Sally
2
0
```

Using the following code excerpt:

```
GymManagement gym = new GymManagement();
gym.printPeople();
```

Will print out:

```
Trainer: Frank
Trainer: Sam
Customer: Bob
Customer: Sally
```

Customers Entering the Gym

Customers can enter the gym until capacity is reached. After which customers are no longer allowed in. Consider the following input file, which specifies two equipment, and three people (two trainers and three customers) and three actions (all customers entering).

```
2
0 3.0
1 2.5
5
0 Frank
0 Sam
1 Bob
1 Sally
1 Tim
2
3
0 2
0 3
0 4
```

Using the following code excerpt:

```
GymManagement gym = new GymManagement();
```

Will print out:

```
Customer: Bob can enter
Gym Capacity: 1/2
Customer: Sally can enter
Gym Capacity: 2/2
Customer: Tim cannot enter
Gym Capacity: 2/2
```

Customers being Trained

A trainer can only train one customer at a time. A piece of gym equipment can only be used by one customer at a time.

Consider the following input file, which specifies two equipment, and four people (two trainers and two customers) and four actions. These four actions are entering the gym, beginning training, and then finishing training (twice).

```
2
0 3.0
1 2.5
4
0 Frank
0 Sam
1 Bob
1 Sally
2
4
0 2
```

```
1 0 2 0
2 0
2 0
```

Using the following code excerpt:

```
GymManagement gym = new GymManagement();
```

Will print out:

```
Customer: Bob can enter
Gym Capacity: 1/2
Trainer: Frank training Customer: Bob using Equipment: Treadmill at speed 3.0
Trainer: Frank has finished training
Trainer: Frank is not training
```

Your Task

The following Java files are provided to you:

- **GymManagement.java** This is the main file that you should edit for this exam.
- **Test1.java** to **Test3.java** contain simple test cases that correspond to the examples above.

You may create new classes if needed. Additionally, sample test inputs and outputs can be found under the `inputs` and `outputs` directories. You can test your program by running the bash script `bash test.sh` which will test your program with all tests. You can test each test individually by:

```
javac TestX.java
java TestX < inputs/TestX.1.in
```

where X is a number between 1 and 3. Compare with the relevant output file `outputs/TestX.1.in`. Before submitting, make sure everything compiles with `javac *.java`.

Task: Rewrite this program using OOP principles

You should read through the file `GymManagement.java` to understand what it is doing. The given implementation applies minimal OO principles, your task in this exam is to rewrite `GymManagement.java` with new classes to apply the OO principles you learned.

To achieve this, create new classes to encapsulate the relevant attributes and methods. Create subclasses as necessary. Use polymorphism to simplify the code in `GymManagement` and make your code extensible to possible new types of equipment, actions, and people. Make sure all OO principles, including LSP, tell-don't-ask, information hiding, are adhered to.

END OF PAPER