TCP1201 Object-Oriented Programming and Data Structures

Lab03 Object-Oriented Thinking

Exercise 1: Implementing Student Class

Define the Student class in the following UML diagram.

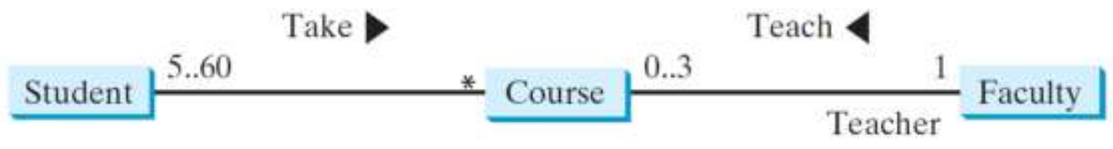
```
                  Student
-id: int
-name: String
-courses: String[]
+Student(id: int, name: String)
+addCourse(course: String): void
+dropCourse(course: String) : void
+toString(): String
```

Sample run:

```
Enter student id  : 111
Enter student name: Ali
Enter courses to add: TCP1101 TCP1201 TMA1101 TMA1201
id = 111, name = Ali, courses = [TCP1101, TCP1201, TMA1101, TMA1201]
Enter courses to drop: TCP1101 TMA1101
id = 111, name = Ali, courses = [TCP1201, TMA1201]
```

Exercise 2: Implementing Student and Course Classes

Lecture 3 has the following UML diagram.



Implement the Student and Course class es based on the the following UML Class Diagram.

```
                  Student
- id: int
- name: String
- courses: Course[]
+ Student (id: int, name: String)
+ getId(): int
+ addCourse (course: Course): void
+ dropCourse (course: Course): void
+ toString():String
```

```
                      Course
- code: String
- teacher: String

+ Course (code: String, teacher: String)
+ getCode():String
+ toString():String
```

Your program shall produce the following output.

Sample output 1: Courses and students are empty.

```
Courses
No. Code/Teacher
-
```

```
Students
No. ID   Name    Course
-

Menu
1. Create a course
2. Create a student
3. A student adds a course
4. A student drops a course
0. Exit
> 1
Enter course code : TCP1101
Enter teacher name: Tan
...
```

Sample output 2: After creating 3 courses and 2 students.

```
> 2
Enter student id  : 222
Enter student name: Bob

Courses
No. Code/Teacher
1.  TCP1101/Tan
2.  TMA1101/Lim
3.  TMA1201/Tong

Students
No. ID   Name    Course
1.  111  Ali     []
2.  222  Bob     []
```

Sample output 3: After Ali added 3 courses and Bob added 1 course.

```
> 3
Enter student id   : 111
Enter course to add: TMA1201

Courses
No. Code/Teacher
1.  TCP1101/Tan
2.  TMA1101/Lim
3.  TMA1201/Tong

Students
No. ID   Name    Course
1.  111  Ali     [TCP1101/Tan, TMA1101/Lim, TMA1201/Tong]
2.  222  Bob     [TMA1101/Lim,]
```

Sample output 4: After Ali dropped 1 course.

```
> 4
Enter student id   : 111
Enter course to add: TCP1101

Courses
```

```
No. Code/Teacher
1.  TCP1101/Tan
2.  TMA1101/Lim
3.  TMA1201/Tong

Students
No. ID   Name    Course
1.  111  Ali     [TMA1101/Lim, TMA1201/Tong]
2.  222  Bob     [TMA1101/Lim,]
```

Exercise 3: Relationship between Objects of Same Class

The UML Class Diagram for a Person class is provided below.

```
                 Person
- name: String
- spouse: Person
+ Person (name: String)
+ setName (name: String): void
+ setSpouse (spouse: Person): void
+ toString(): String
```

Define the Person class:

- Note that a person may have another person as spouse. If the person does not have a spouse, the value of spouse data field is null.
- The `toString()` method returns both the names of the person and the spouse. You shall check whether the person has a spouse. If the person does not have a spouse, then returns “none” as the spouse’s name.

Write a test program that:

1. Create 2 persons. Give each of them a name. Display the name and spouse of each person.
2. Set the spouse of person to the other person. Display the name and spouse of each person.
3. Change the name of one of the 2 persons. Display the name and spouse of each person.

Sample run:

```
Name = Ali, spouse = none
Name = Siti, spouse = none
Ali and Siti get married.
Name = Ali, spouse = Siti
Name = Siti, spouse = Ali
Change Ali's name to Abu.
Name = Abu, spouse = Siti
Name = Siti, spouse = Abu
```