48024 Applications Programming

Assignment 1

Topics: OO design, standard patterns, lists
Objectives: This assignment supports objectives 1 - 3
Due date: 5pm Monday 9th of May 2016
Weight: 30%

1. Individual work

All work is individual. You may discuss ideas, approaches and problems, but you should write every line of code yourself except for code copied from the lecture notes, lecture code or lab code. More information about Academic Misconduct can be found at: http://www.gsu.uts.edu.au/rules/student/section-16.html

2. Specification

UTS has asked you to create a new timetabling system for students. The university maintains a list of students and a list of subjects. In the initial rollout, the university has just two subjects:

Subject number	**Subject name**
48024	Applications Programming
31284	Web Services Development

Each subject has a number of activities that students can enrol in which are listed below:

Subject	Group	Activity	Day	Start	Duration	Room	Capacity	Enrolled
48024	Lec1	1	Wed	18	1	CB11.00.405	200	0
48024	Cmp1	1	Wed	19	2	CB11.B1.403	2	0
48024	Cmp1	2	Wed	19	2	CB11.B1.401	2	0
48024	Cmp1	3	Wed	19	2	CB11.B1.402	2	0
31284	Lec1	1	Tue	16	1	CB02.03.002	160	0
31284	Cmp1	1	Tue	9	2	CB11.B1.102	30	0

31284	Cmp1	2	Tue	9	2	CB11.B1.103	30	0
31284	Cmp1	3	Tue	14	2	CB11.B1.102	30	0
31284	Cmp1	4	Tue	14	2	CB11.B1.103	30	0

Lectures have the group “Lec1” and labs have the group “Cmp1”. For a particular subject, a student can enrol in at most one activity per group (i.e. one lecture and one lab). If a student tries to enrol in a second activity with the same group and subject as an activity that the student is already enrolled in, then the student is automatically withdrawn from the original activity before being enrolled into the second activity.

The last two columns of the activities table above indicate the number of students allowed to enrol in that activity and the number of students who are currently enrolled in that activity respectively. Initially there are zero students enrolled in each activity. When a student successfully enrols into an activity, the enrolled count for that activity goes up. A student can also choose to withdraw from an activity at any time, which causes the enrolled count to go down. A student cannot enrol in an activity if the number enrolled has already reached the capacity. A student can also auto-enrol into a particular group for a subject. For example, a student can auto-enrol into a lab (i.e. group Cmp1) for Applications Programming, and the system will enrol the student into the first Cmp1 activity in the list that has not yet reached capacity.

Students can be added to and removed from the university. Each student has a unique student number and a name as well as a list of activities that the student is currently enrolled in.

University menu

When the application starts, the university offers a main menu. Typing an unrecognised option, such as a question mark, will show help:

```
Choice (a/r/v/l/x): ?
University menu options
a = add a student
r = remove a student
v = view all students
l = login
x = exit
Choice (a/r/v/l/x):
```

All user input is highlighted in bold. All of the following I/O traces are a continuation of the same run.

Students can be added and removed as follows:

```
Choice (a/r/v/l/x): a
Number: 12345678
Name: Bianca Sladen
Choice (a/r/v/l/x): a
Number: 49287512
Name: Hugo Aitken
Choice (a/r/v/l/x): a
Number: 23232323
Name: Jessica Sneddon
Choice (a/r/v/l/x): a
Number: 11111111
Name: Dakota Cavill
Choice (a/r/v/l/x): v
12345678 Bianca Sladen
49287512 Hugo Aitken
23232323 Jessica Sneddon
11111111 Dakota Cavill
Choice (a/r/v/l/x): r
Number: 11111111
Choice (a/r/v/l/x): v
12345678 Bianca Sladen
49287512 Hugo Aitken
23232323 Jessica Sneddon
Choice (a/r/v/l/x):
```

But student numbers are unique so you cannot add the same student number twice:

```
Choice (a/r/v/l/x): a
Number: 12345678
Student number already exists
Choice (a/r/v/l/x):
```

And you cannot remove a student that doesn't exist:

```
Choice (a/r/v/l/x): r
Number: 00000000
No such student
Choice (a/r/v/l/x):
```

When the user logs in, you show an error if the student is not found:

```
Choice (a/r/v/l/x): l
Number: 89898989
No such student
Choice (a/r/v/l/x):
```

If the student is found, you show the student menu:

```
Choice (a/r/v/l/x): l
Number: 12345678
Choice (v/e/w/x):
```

Student menu

The student menu presents the following options:

```
Choice (v/e/w/x): ?
Student menu options
v = view my activities
e = enrol in an activity
w = withdraw from an activity
x = exit
Choice (v/e/w/x):
```

A student enrols by selecting a subject and then selecting an activity by inputting an activity code in the format `group:activity`

```
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 0/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Lec1:1
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1:2
Choice (v/e/w/x): v
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
```

```
Choice (v/e/w/x):
```

If a student enrols into an activity with the same group and subject as one already enrolled into, the student is automatically withdrawn from that activity first before being enrolled into the new activity:

```
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1:1
Choice (v/e/w/x): v
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
Choice (v/e/w/x):
```

Enrolling into either a non-existent subject or a non-existent activity shows an error:

```
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48023
No such subject
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1:8
No such activity
Choice (v/e/w/x):
```

A student can also withdraw from an activity by entering an activity code in the format `subject:group`

```
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 31284
Select an activity
31284 Lec1 1 CB11.00.405 16:00 1hrs 0/200
31284 Cmp1 1 CB11.B1.102 09:00 2hrs 0/30
31284 Cmp1 2 CB11.B1.103 09:00 2hrs 0/30
31284 Cmp1 3 CB11.B1.102 14:00 2hrs 0/30
31284 Cmp1 4 CB11.B1.103 14:00 2hrs 0/30
Activity code (group:number): Lec1:1
Choice (v/e/w/x): v
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
31284 Lec1 1 CB11.00.405 16:00 1hrs 1/200
Choice (v/e/w/x): w
Activity code (subject:group): 31284:Lec1
Choice (v/e/w/x): v
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
Choice (v/e/w/x):
```

Show an error if the student tries to withdraw from an activity that he/she is not enrolled in, or from an activity that does not exist. Use the same error message in both situations:

```
Choice (v/e/w/x): w
Activity code (subject:group): 31284:Cmp1
Not enrolled in activity
Choice (v/e/w/x): w
Activity code (subject:group): 48023:Cmp1
Not enrolled in activity
Choice (v/e/w/x):
```

Exiting the student menu returns to the main university menu:

```
Choice (v/e/w/x): x
Choice (a/r/v/l/x):
```

Multiple students

When more than one student enrols in an activity, you will need to cope with activities reaching capacity. The labs in subject 48024 have a capacity of only 2 students each, so they are quick to fill up:

```
Choice (a/r/v/l/x): l
Number: 49287512
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1:1
Choice (v/e/w/x): x
Choice (a/r/v/l/x): l
Number: 23232323
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 2/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1:1
No available seats
Choice (v/e/w/x):
```

If the student inputs just the group as the activity code (rather than group:number), the student is auto-enrolled into the first available activity for that group. If all activities for that group are full, you show the “No available seats” error.

```
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 2/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:number): Cmp1
Choice (v/e/w/x): v
```

```
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
```

Notice that Cmp1:1 was full and so the student was automatically enrolled into Cmp1:2.

Removing a student who is enrolled

If you remove a student who is enrolled in one or more activities, that student is automatically withdrawn from those activities before being removed.

For example, the current enrolments for subject 48024 are:

```
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 2/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
```

Bianca Sladen is currently enrolled into Lec1:1 and Cmp1:1. After removing Bianca, the enrolments should become:

```
48024 Lec1 1 CB11.00.405 18:00 1hrs 0/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
```

We can test this by trying to enrol a new user into the full activity Cmp1:1 before and after removing Bianca:

```
Choice (a/r/v/l/x): a
Number: 98761234
Name: Ryan Heise
Choice (a/r/v/l/x): l
Number: 98761234
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 2/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:activity): Cmp1:1
No available seats
Choice (v/e/w/x): x
Choice (a/r/v/l/x): r
```

```
Number: 12345678
Choice (a/r/v/l/x): l
Number: 98761234
Choice (v/e/w/x): e
Select a subject
48024 Applications Programming
31284 Web Services Development
Subject number: 48024
Select an activity
48024 Lec1 1 CB11.00.405 18:00 1hrs 0/200
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 1/2
48024 Cmp1 2 CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 CB11.B1.402 19:00 2hrs 0/2
Activity code (group:activity): Cmp1:1
Choice (v/e/w/x): v
48024 Cmp1 1 CB11.B1.403 19:00 2hrs 2/2
Choice (v/e/w/x): x
Choice (a/r/v/l/x): x
```

3. Design

Your solution must have 4 classes with the following fields:

```
public class University {
	private LinkedList<Subject> subjects = new LinkedList<Subject>();
	private LinkedList<Student> students = new LinkedList<Student>();
}

public class Student {
	private String number;
	private String name;
	private LinkedList<Activity> activities = new LinkedList<Activity>();
}

public class Subject {
	private int number;
	private String name;
	private LinkedList<Activity> activities = new LinkedList<Activity>();
}

public class Activity {
	private Subject subject;
	private String group;
	private int number;
	private String day;
```

```
        private int start;
        private int duration;
        private String room;
        private int capacity;
        private int enrolled;
}

public class In {
        private static final Scanner scanner;
}
```

Your application's main method must be defined in class University. PLATE will not accept your submission unless your solution uses exactly these classes and fields.

Your constructors must also obey the following rules:

- All fields are initialised from constructor parameters, with two exceptions:
    - Lists are **not** initialised from constructor parameters. The constructor for class University should create and add the two subjects 48023 and 31284 to the list of subjects, and should also create and add all of the activities to those subjects, according to the data presented at the top of this document.
    - The `enrolled` field of class Activity is always initialised to zero, so there should be no constructor parameter for it.
- Constructors do nothing more than initialise fields.

Your toString() functions should also meet the following requirements:

- The toString() function of Activity should return a string in the following format:
  `31284 Cmp1 2 CB11.B1.103 09:00 2hrs 0/30`
  including the subject number (31284), the group (Cmp1), the activity number (2), the room (CB11.B1.103), the start time (09:00), the duration (2hrs) and the number of students enrolled over the capacity (0/30 means 0 students are enrolled with a capacity of 30). Note: in this example, the start time is stored in class Activity's field `private int start` as the integer 9. Only the hour is stored. The string representation is `09:00`. If the start time is less than 10, insert a leading zero into the string.
- The toString() function of Student should return a string in the following format:
  `12345678 Bianca Sladen`
  including the student number and student name.
- The toString() function of Subject should return a string in the following format:
  `31284 Web Services Development`
  including the subject number and subject name.
- The University class does not require a toString() function.

Beyond this, you are free to make your own design decisions, however, a design guide will be posted to UTSOnline in week 5 at Assignments / 1 / design which will offer strong recommendations on how you should design and build your solution for best results.

4. Expected workload

The time to do the assignment has been estimated at 25 hours for a student of average ability who has completed all the tutorial and lab exercises.

5. Online support

The assignment 1 discussion board has been set up so that students can ask questions, and other students can reply. I will post a reply only if I think the student response was wrong, or in the case of correcting a mistake in the assignment specification.

You must not post Java code to the discussion board. The board is there to help you, not to provide the solution. Posting your code is academic misconduct and will reported. Each time this rule is violated, I will delete the code and post a comment of the form: “Strike 1: Posting code”. After 3 strikes, the discussion board will be deleted because it did not work.

FAQs (Frequently Asked Questions) and their answers are posted on UTSOnline in Assignments/1/faq. If you have a question, check the FAQ first; it may already be answered there. You should read the FAQ at least once before you hand in your solution, but to be safe check it every couple of days. Anything posted on the FAQ is considered to be part of the assignment specification. The FAQ will be frozen (no new entries) two days before the due date; no questions will be answered after it is frozen.

If anything about the specification is unclear or inconsistent, contact me and I will try to make it clearer by replying to you directly and posting the common questions and answers to the FAQ. This is similar to working on the job, where you ask your client if you are unsure what has to be done, but then you write all the code to do the task. Email Ryan.Heise@uts.edu.au to ask for any clarifications or corrections to the assignment.

6. PLATE marking

Your solution is marked for correctness by PLATE (https://plate.it.uts.edu.au/) by comparing the output of your system to the output of the benchmark system. You can submit a solution to PLATE many times; I urge you to do this, so you receive credit for your work.

PLATE will test the features of your program in a certain order: Classes and fields, then constructors, then goals from basic to advanced. PLATE cannot test the more advanced goals until the basic goals are working. For example, you must implement “add a student”

before “enrol”, because it is impossible to enrol without first having a student. To receive marks, you must pass PLATE’s test cases in the order in which PLATE tests them.

Your code is marked by software, so you can get a good mark by fooling or spoofing the software. If you spoof a task worth N marks, you receive a penalty of 2*N marks.

7. Submission and return

Your solution is to be submitted to PLATE at https://plate.it.uts.edu.au/ under Applications Programming / Assessments / Assignment 1. Your provisional mark and feedback is generated immediately each time you submit to PLATE. However, analysis of spoofing, plagiarism and collusion is done in the two weeks following the due date. If you are suspected of Academic Misconduct, I will forward your case to the Misconduct Committee and will notify you by email.

There is no scheduled late submission period. An extension of up to one week may be given by the subject coordinator before the due date; you have to supply documentary evidence of your claim. An extension CANNOT be given after the due date.

You may also apply for special consideration for reasons including unexpected health, family or work problems. More information about how to apply for special consideration can be found at http://www.sau.uts.edu.au/assessment/consideration.html.

8. Marking scheme

Task	**Mark**
Class and field declarations	5%
Constructors	5%
toString functions	5%
University menu help	5%
Add student	10%
View all students	5%
Remove student	10%
Login	5%
Student menu help	5%
Enrol	25%

View my activities	5%
Withdraw	5%
Auto enrol	10%