148024 Applications Programming Assignment 1

Topics:

OO Design, Standard Patterns, Lists

Learning Outcomes:

This assessment task addresses the following subject learning objectives (SLOs): 1, 2 and 3

Due date:

11:59PM Monday 11 May

Weight:

35%

Platform:

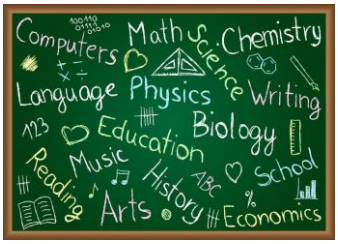
Edstem.org

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. You **MUST NOT** let another student see your solution code, and you **MUST NOT** look at another student's solution code. More information about Academic Misconduct can be found at:

http://www.gsu.uts.edu.au/rules/student/section-16.html

Overview



Timetable – A University Timetable System

UTS is setting up a new timetabling system, to help students affected by Covid-19 to reschedule their study plan. UTS has encouraged you and other capable programmers to write some software to help this process.

The aim is to simplify the process and makes the timetable more accessible to the students.

This requirement is urgent, as some capable programmers have heard the rumors of this state-of-the-art system and they have also started working on similar prototypes.

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 enroll 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 enroll in at most one activity per group (i.e. one lecture and one lab). If a student tries to enroll 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 enroll 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 enrolls 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 enroll 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 enroll 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.

An aside

While reading the first part of the specification, you will notice there is a lot going on.

- How many functions did you identify?
- How many classes did you identify?
- What are the fields in each class?
- How many goals did you identify?
- How many patterns did you think of that might be applicable?

This assignment will be challenging and you will probably want to manage your time well.

- How long do you think it will take you to code the functions?
- How long do you think it will take you to code each goal?

A good rule of thumb is to think of an estimate, and then multiply that number by 3 or 4!

To manage your time well, you may need to figure out which parts of the assignment you can start early.

- Which parts can you start now?
- Which parts can you start in week 6?

If you complete parts in the same week that you learn the topics (while they are fresh in your mind), they will take less time to complete.

The User Interface

Below are the sample I/O trace for different menus. The green text indicates user input (you are not expected to print color text). Not every conceivable scenario is shown below and you should submit your code to the platform to see what specific scenarios are tested. You should also implement your solution in the same order as test cases so that you can receive incremental feedback and marks as you progress.

University Menu

```
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): 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):
// 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):
// 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

```
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 Wed CB11.00.405 18:00 1hrs 0/200
48024 Cmp1 1 Wed CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 Wed CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 Wed CB11.B1.402 19:00 2hrs 0/2
Activity code (group:activity): 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 Wed CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 Wed CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 Wed CB11.B1.401 19:00 2hrs 0/2
48024 Cmp1 3 Wed CB11.B1.402 19:00 2hrs 0/2
Activity code (group:activity): Cmp1:2
Choice (v/e/w/x): v
48024 Lec1 1 Wed CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 2 Wed 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 Wed CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 Wed CB11.B1.403 19:00 2hrs 0/2
48024 Cmp1 2 Wed CB11.B1.401 19:00 2hrs 1/2
48024 Cmp1 3 Wed CB11.B1.402 19:00 2hrs 0/2
Activity code (group:activity): Cmp1:1
Choice (v/e/w/x): v
48024 Lec1 1 Wed CB11.00.405 18:00 1hrs 1/200
48024 Cmp1 1 Wed 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 Wed CB11.00.405 18:00 1hrs  1/200
48024 Cmp1  1 Wed CB11.B1.403 19:00 2hrs  1/2
48024 Cmp1  2 Wed CB11.B1.401 19:00 2hrs  0/2
48024 Cmp1  3 Wed CB11.B1.402 19:00 2hrs  0/2
Activity code (group:activity): 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 Tue CB11.00.405 16:00 1hrs  0/200
31284 Cmp1  1 Tue CB11.B1.102 09:00 2hrs  0/30
31284 Cmp1  2 Tue CB11.B1.103 09:00 2hrs  0/30
31284 Cmp1  3 Tue CB11.B1.102 14:00 2hrs  0/30
31284 Cmp1  4 Tue CB11.B1.103 14:00 2hrs  0/30
Activity code (group:activity): Lec1:1
Choice (v/e/w/x): v
48024 Lec1  1 Wed CB11.00.405 18:00 1hrs  1/200
48024 Cmp1  1 Wed CB11.B1.403 19:00 2hrs  1/2
31284 Lec1  1 Tue 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 Wed CB11.00.405 18:00 1hrs 1/200
48024 Cmp1  1 Wed 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 Student

When more than one student enroll 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. If the student inputs just the group as the activity code (rather than group:activity), 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 Wed CB11.00.405 18:00 1hrs  1/200
48024 Cmp1  1 Wed CB11.B1.403 19:00 2hrs  2/2
48024 Cmp1  2 Wed CB11.B1.401 19:00 2hrs  0/2
48024 Cmp1  3 Wed CB11.B1.402 19:00 2hrs  0/2
Activity code (group:activity): Cmp1
Choice (v/e/w/x): v
48024 Cmp1  2 Wed CB11.B1.401 19:00 2hrs 1/2
// Notice that Cmp1:1 was full and so the student was automatically
enrolled into Cmp1:2.
Choice (v/e/w/x):
```

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:

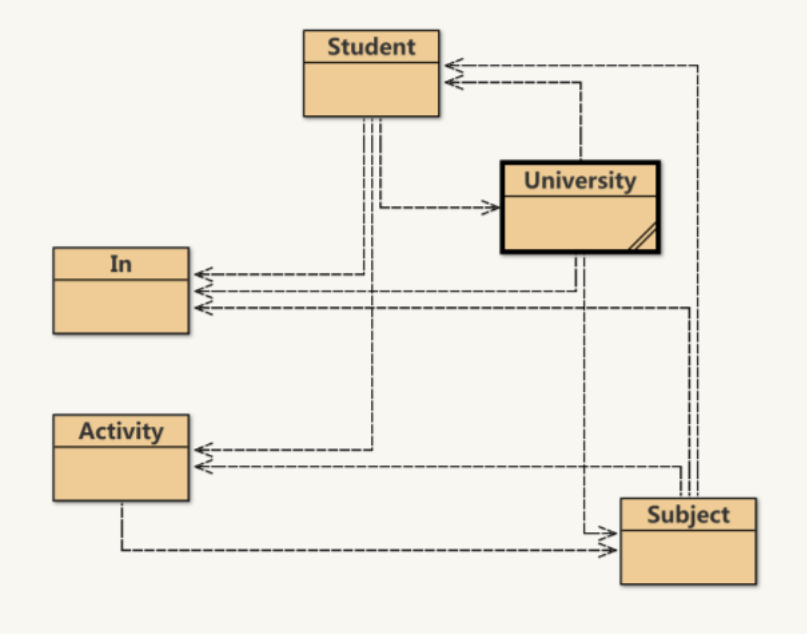
```
48024  Lec1  1  Wed   CB11.00.405   18:00   1hrs  1/200
48024  Cmp1  1  Wed   CB11.B1.403   19:00   2hrs  2/2
48024  Cmp1  2  Wed   CB11.B1.401   19:00   2hrs  1/2
48024  Cmp1  3  Wed   CB11.B1.402   19:00   2hrs  0/2
```

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

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

Requirements

- Your design will consist of exactly the following classes with the listed fields, declared as indicated. You may not add or remove classes or fields; however, you may add constructors, functions and procedures to complete your design (in fact, you will have to!). You should pay careful attention to the test cases, as these will help guide you with some (but not all) of these methods.
- To help visualize the design, a partial class diagram has been provided



- Classes – your design will consist of these 5 classes:
  1. University
  2. Student
  3. Subject
  4. Activity
  5. In (this is just the class you've been using throughout the labs to facilitate simpler I/O – just copy it over)

- Fields – the classes will have 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 String 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;
}
```

- Your application's main method must be defined in class University.
- Constructors – the constructors of the class have the following requirements:
  1. All fields are initialised from constructor parameters, with two exceptions:
     - The `enrolled` field of class Activity is always initialised to zero, so there should be no constructor parameter for it.
     - 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.
  2. Constructors do nothing more than initialize fields.
- Your toString() functions should also meet the following requirements:
  1. The toString() function of Activity should return a string in the following format:

     `31284 Cmp1 2 Tue CB11.B1.103 09:00 2hrs 0/30`

     including the subject number (31284), the group (Cmp1), the activity number (2), the day, 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.

2. The toString() function of Student should return a string in the following format:

```
12345678 Bianca Sladen
```

   including the student number and student name.

3. The toString() function of Subject should return a string in the following format:

```
31284 Web Services Development
```

   including the subject number and subject name.

4. The University class does not require a toString() function.

Expected Workload

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

Online Support

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

You must not post or share 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, the code will be removed and replaced with 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 will be posted in the platform. 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 the subject coordinator who 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 huan.huo@uts.edu.au to ask for any clarifications or corrections to the assignment.

Marking

The platform marks your solution for correctness by comparing the output of your system to the output of the benchmark algorithm. You can submit a solution to the platform many times; I urge you to do this, so you receive credit for your work.

The platform will test the features of your program in a certain order: Classes and fields, then constructors, then goals from basic to advance. The platform cannot test the more advanced goals until the basic goals are working. For example, you must implement “add a student” before “enroll”, because it is impossible to enroll without first having a student. To receive marks, you must pass the platform’s test cases in the order in which the platform 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.

Submission and Return

Your solution is to be submitted to the platform under Applications Programming / Assignment 1. Your provisional mark and feedback is generated immediately each time you submit to the platform. However, analysis of spoofing, plagiarism, collusion and general cheating 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.

Marking Scheme

The marks for the assignment are divided into the following functionality components (note that individual tests may test several functionality components, and a functionality component may be tested by several tests):

Functionality Component	Mark Allocation
Class and field declarations	5% (test3+test11)
Constructors	5% (test3+test11)
toString functions	5% (test3+test11)
University Help	5% (test1)
University Add student	10% (test2+test4)
University View all students	5% (test3)
University Remove student	10% (test5+test6+test19)
Login	5% (test7+test8)
Student Help	5% (test9)
Enroll	25% (test10,12,13,16)
Student View	5% (test11)
Withdraw	5% (test14+15)
Auto Enroll	10% (test17+test18)

This adds to a mark out of 100, at makes up 35% of your final assessment mark.