COMPSCI 2S03, Principles of Programming
Assignment 2, Fall 2020

Hassan Ashtiani, McMaster University

Due date: Saturday, October 10, 11pm

- You should write your codes in Java. The input/output of the programs are all from/to standard-input/standard-output (i.e., reading from the keyboard, and writing to the text terminal).
- DOMJudge is used to automatically test your code against a number of testcases. To submit to DOMJudge, use the username and password that was given to you and log in to http://130.113.70.122/domjudge/team from the campus (or via VPN) and then submit your source files for each question.
- DOMJudge has both public test-cases (where you see the example which your code failed) and private test-cases (which you do not see the test cases). Your code will be graded based on both kinds of tests.
- **In addition to DOMJudge, submit your source codes on Avenue**. Upload a single zip file named `macID_assignment2.zip` (e.g., `zokaeiam_assignment2.zip`). The zip file should include only one folder, called `macID_assignment2`, and this folder should include all your .java source files and nothing else (see the example file uploaded on Piazza). This makes marking much faster for the TAs, so **any violations from this will be penalized**.
- Your source codes will be graded by the TAs; so **follow the style conventions** for coding from the class (naming, indentation, spacing, comments) to get full marks. For more info take a look at the slides or at https://google.github.io/styleguide/javaguide.html
- Use Piazza for discussions and clarifications about the assignment. This is an involved question and I expect that some questions may arise about different aspects of it. You can also post if you find a bug in the given code.
- This assignment has bonus points. If you get a grade above 100, then it can compensate for your other assignments (but not for the exams).
- Tip: the input to these programs can be quite long. Therefore, if you test your program by simply giving it the inputs in the terminal then you would spend a lot of time doing this each time you test your program. Therefore, you are strongly encouraged to create a number of tests and store them in a file. Then for testing purposes you can read from a file (as discussed in the tutorial) or at least copy-paste from a file to the terminal. Note that when submitting your code your program should read from the terminal input.

Download the source codes associated with this assignment from Avenue. The given classes include Course, Main, Registrar, Section, Student and TerminalPrinter. The big picture is that we can define courses, and we can define (multiple) sections with certain capacities for each course. We can also define students, and enroll or unenroll them from a certain section of a certain course. Take a look at the code and run it to see the results.

In the given code, we have hard-coded the commands (in runExampleCommands() method in Registrar class). In the first question we want to change the program so that it reads the input from the terminal as opposed to the hard-coded example. This is an example of input and the expected output which actually corresponds to the hard-coded runExampleCommands().

Input Sample

```
COURSE CS2XYZ
COURSE CS1ABC
STUDENT EMMA 23345
STUDENT DAVID 123345
SECTION CS2XYZ C01 10
SECTION CS1ABC C02 1
ENROLL 23345 CS2XYZ C01
ENROLL 123345 CS2XYZ C01
ENROLL 23345 CS1ABC C02
ENROLL 123345 CS1ABC C02
UNENROLL 23345 CS1ABC C02
UNENROLL 123345 CS1ABC C02
ENROLL 123345 CS1ABC C02
FINISH
```

Output Sample

```
EMMA WAS ENROLLED IN CS2XYZ SECTION C01
DAVID WAS ENROLLED IN CS2XYZ SECTION C01
EMMA WAS ENROLLED IN CS1ABC SECTION C02
DAVID WAS NOT ENROLLED IN CS1ABC SECTION C02
EMMA WAS UNENROLLED FROM CS1ABC C02
DAVID WAS NOT UNENROLLED FROM CS1ABC SECTION C02
DAVID WAS ENROLLED IN CS1ABC C02
```

As you can see, each line of the input is a command with multiple space-separated tokens:

- If a line starts with "COURSE", then it will be followed by the (unique) name of the course.
- If a line starts by "STUDENT" then it will be followed by the name of the student and their (unique) student ID.
- If a line starts by "SECTION" then it will be followed by the name of the course which it belongs to, the name of the section (which is unique among sections of that course) and the capacity of that section.
- If a line starts by "ENROLL" then it will include the id of the student that is going to be enrolled, the name of the course, and the name of the section. Depending on the remaining capacity of the section, this enrollment request may or may not be successful. Take a look at the sample output to see what should be printed in each case.
- If a line starts by "UNENROLL", it will be followed by three tokens exactly like the case for "ENROLL". If the student is actually enrolled in the specified section, then she/he will be removed from the section and a message is printed. Otherwise, no action is made and another message is printed. See the sample output to understand the format of the expected printed message.
- If a line starts by "FINISH" then it indicates the end of the input. There will be no more commands after this line.

1. [**70 points: 15 points for public test cases, 15 points for private test cases, 20 points for source code, 20 for styling**] Make the required changes to the code so that it reads the input from the terminal. Your code should be able to handle an arbitrary but "valid" input (e.g, if we replace the order of the first two lines of the above example we still have a valid input. But if we have repetitive course names, or a typo in the command – like COURRSE – then the input is not valid and we don't check your code for such cases). Your style marks for this question includes writing comments even for the given source codes (No need to add comments for every line; only add those comments that help understanding of the code and the context of the problem). You are allowed to change the code, but you are strongly encouraged to keep the overall structure of the given code.

2. [**40 points: 10 for public tests, 20 for private tests, 10 for source code, 10 for styling**]

   In this question we would like to add new functionality to our program. Namely,

- If a line starts with “REQUIREMENT”, then it will be followed by the name of a course (Say A) and then the name of its “required” course (Say B). This means that in order to enroll in A, the student should be first enrolled in B. Otherwise enrollment will be unsuccessful.
- Note that we can have a potential loophole that a student first enrolls in B in order to be able to enroll in A; but then (after enrolling in A) he unenroll from B. To avoid this, any unenrollment request will be unsuccessful if the student is enrolled in a course that requires the specified course (in the mentioned example, unenrolling from B will be unsuccessfull unless the student first unenrolls from A)

This is an example.

Input Sample

```
COURSE CS2XYZ
COURSE CS1ABC
REQUIREMENT CS2XYZ CS1ABC
STUDENT EMMA 23345
SECTION CS2XYZ C01 10
SECTION CS1ABC C02 1
ENROLL 23345 CS2XYZ C01
ENROLL 23345 CS1ABC C02
ENROLL 23345 CS2XYZ C01
UNENROLL 23345 CS1ABC C02
UNENROLL 23345 CS2XYZ C01
UNENROLL 23345 CS1ABC C02
FINISH
```

Output Sample

```
EMMA WAS NOT ENROLLED IN CS2XYZ SECTION C01
EMMA WAS ENROLLED IN CS1ABC SECTION C02
EMMA WAS ENROLLED IN CS2XYZ SECTION C01
EMMA WAS NOT UNENROLLED FROM CS1ABC SECTION C02
EMMA WAS UNENROLLED FROM CS2XYZ SECTION C01
EMMA WAS UNENROLLED FROM CS1ABC SECTION C02
```