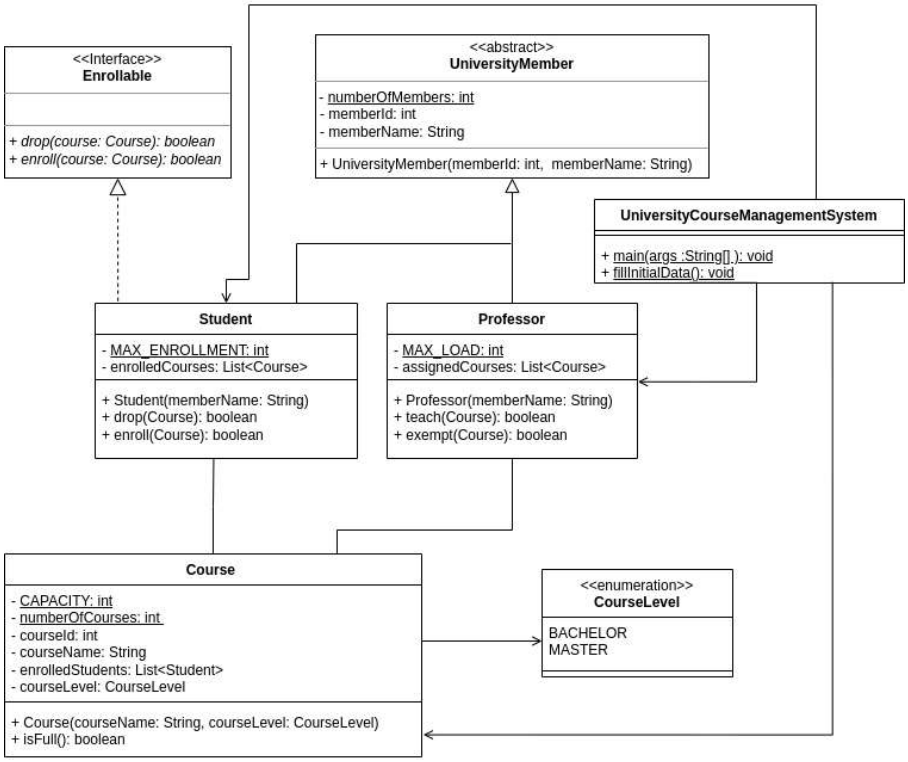
A. University Courses Management System

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

An IT university desires to develop a course management system. Courses have fixed capacities and students can only be enrolled in a limited number of courses if their capacity allows for it.



Initial Instances:

Courses to add:

courseId	courseName	courseLevel
1	java_beginner	bachelor
2	java_intermediate	bachelor
3	python_basics	bachelor
4	algorithms	master
5	advanced_programming	master
6	mathematical_analysis	master
7	computer_vision	master

Students to add:

memberId	memberName	enrolledCourses
1	Alice	java_beginner

		java_intermediate python_basics
2	Bob	java_beginner algorithms
3	Alex	advanced_programming

Professors to add:

memberId	memberName	enrolledCourses
4	Ali	java_beginner java_intermediate
5	Ahmed	python_basics advanced_programming
6	Andrey	mathematical_analysis

Note: The Ids of courses and university members are unique within their classes and assigned automatically by increasing *numberOfCourses* in case of courses and increasing *numberOfMembers* by 1 in case of professors and students. Professor and Student roles cannot be played simultaneously by any University Member.
For courses: the first courseId is 1, the next is 2, and so on with incrementation.

For University Members (professors and students): the first *memberId* is 1, the next is 2, and so on.

Methods:

- *fillInitialData():* load the provided instances with the arguments from the abovementioned tables.
- *teach(Course):* assign the professor to teach a course if not already teaching it and has not reached his/her course load limit.
- *exempt(Course):* exempt the professor from teaching a course if he/she is currently teaching it.
- *enroll(Course):* enroll the student in a course if the student is not already enrolled, did not reach the maximum number of enrollments, and if the course is not full of students.
- *drop(Course):* drop the student from a course if he/she is currently enrolled.
- *isFull():* check if the course has reached its enrollment capacity.
- *getters* and *setters* should be used whenever they are needed. Intentionally omitted in Class diagram.

Rules:

- Each course has a capacity of 3 students.
- A student can't be enrolled in the same course more than once.
- A student cannot be enrolled in more than 3 courses.
- A student cannot be enrolled in a course that is full (already 3 students enrolled this course).
- A student cannot drop a course, if not enrolled.
- Professor cannot be assigned to a course he/she is already teaching.
- Professor cannot be assigned to teach more than 2 courses.
- Professor cannot be exempted from the course, if not teaching it.

Input

The inputs are represented by standard input. The program should read the input line by line. It checks the correctness of the input line, if it is correct, it waits for the next line, otherwise it prints an error message and terminates the program. Whenever the empty line is met, the program should stop its execution. There can be multiple commands.

The input should be in the following format:

Command	Input format
Add course	course

		баллов
233553126	20.11.2023 14:21	Частичное решение: 53 баллов
233552979	20.11.2023 14:19	Частичное решение: 50 баллов
233349022	18.11.2023 21:45	Частичное решение: 52 баллов
233348397	18.11.2023 21:39	Частичное решение: 51 баллов
233348335	18.11.2023 21:39	Частичное решение: 53 баллов
233347351	18.11.2023 21:30	Частичное решение: 47 баллов

→ Набранные баллы

	Баллы
A	53
Всего	53

	courseName courseLevel
Add student	student memberName
Add professor	professor memberName
Enroll to course	enroll memberId courseId
Drop from course	drop memberId courseId
Assign to course	teach memberId courseId
Exempt from course	exempt memberId courseId

Constrains:

- Student and professor names should contain English characters only.
- Course name should contain English characters separated by underscore symbols, if needed. By both sides from underscore(s) there has to be at least one English character.
- Courses', students', professors' names and course level are a non-sensitive letter case, for example "Java" equals "java", "master" equals "MASTER".
- The course level should be either "bachelor" or "master".
- Student name, professor name and course level should not be having values equal to commands titles.

Output

The outputs are represented by standard output. Each output line should be generated for each read and performed input command. The first found error in inputs (if present) should lead to one of the output errors and stop further code execution.

For each valid input for command the appropriate functionality has to be performed and the following message has to be appended to the output:

Command	Message
course	Added successfully
student	Added successfully
professor	Added successfully
enroll	Enrolled successfully
drop	Dropped successfully
exempt	Professor is exempted
teach	Professor is successfully assigned to teach this course

The inputs may contain errors, which should lead to one of the error messages provided below. The errors are prioritized, which means that having the first error wouldn't allow considering further errors.

1. Course exists
2. Student is already enrolled in this course
3. Maximum enrollment is reached for the student

4. Course is full
5. Student is not enrolled in this course
6. Professor's load is complete
7. Professor is already teaching this course
8. Professor is not teaching this course
9. Wrong inputs

After the error message the program should finish its execution.

Examples

input Скопировать

```
course
Java_advanced
master
```

output Скопировать

```
Added successfully
```

input Скопировать

```
enroll
1
1
```

output Скопировать

```
Student is already enrolled in this course
```

input Скопировать

```
student
siba
enroll
7
5
enroll
7
6
enroll
7
4
enroll
7
2
```

output Скопировать

```
Added successfully
Enrolled successfully
Enrolled successfully
Enrolled successfully
Maximum enrollment is reached for the student
```

Note

Note that use of @SuppressWarnings for Checkstyle plugin will be considered as a cheating case.