We will create a simple java application for management of university courses, where we can create, update and delete a course,
Register new students to a course, register lectors for each course.
All the data will be saved in the java heap (in arrays or lists)

The application should have next functionalities:

- adding of a new course
- deleting of an already existing course
- sigh a lector to a course
- adding of a student to a course
- deleting of a user (lector/student) from a course

When you finish the task make a test class with main method
and also you will have the challenge to add functionality with passing command line arguments via which to manage the courses and student for the university

NOTE: for implementation you will use everything learned until now.

Also, we are assuming that each student has unique ID, each professor, lector has unique ID and each assistance has unique ID, and each course has unique course name.

So, it's possible to have a student with ID 1 and professor with ID 1, but no another student with ID 1!

SUPPORT - steps for implementation

1. Create new java application

2. Create class Course with next variables:
- name
- array of students
- assistance - it's a class of Lector, represents the professor assistance
- lector - it's a class of Lector, represents the docent or professor

Define 2 constructors as:
- one constructor with passing the name of the course as arguments to the constructor
- another constructor (overloaded) with passing the name, the lector, and the assistance as arguments to the constructor

Define next methods:

- one method for adding of a Student
- one method for delete of a student
- one method for setting the assistance
- one method for setting the lector

When you are adding any of the above, you need to pass all required argumetns in order to have any of the instances created.
Maximum students per course are 30!

3. Create class with name User with next variables: Java Bean
- id
- firstName
- lastName

Define constructors and getters

4. Create class Student extends class User. Add next variables to class Student:
- facNumber
- array of courses

Define constructors - pass with super keyword to the constructor in the base class User

Define next 2 methods:
- add a Course for the student
- delete a Course for the student

Maximum courses per student are 10!

5. Create class Lector extends class User. Add next variables to class Lector

- enum lectorType (with predefined values - DOCENT, PROFESSOR, ASSISTANCE)
- array of courses

Define constructors - pass with super keyword to the constructor in the base class User

Define next 2 methods:
- add a Course for the Lector
- delete a Course for the Lector
Maximum courses per Lector are 4!

6. Create interface **UniManagement** which will define the contract, or all functions, which will be used for management of the university courses as:

```
public interface UniManagement {

    /**
    * Crete a new course with name courseName and return it
```

```
        *
        * @return new instance of course with the passed name if it's created or
        * null in another case. A course will be created only if already does not exists
        * another course with the same courseName
        */
        public Course createCourse(String courseName);


        /**
        * Delete a course with name courseName
        *
        * @return <code>true</code> only in case the course with passed name was
removed
        */
        public boolean deleteCourse(String courseName);

        /**
        * Creat and return new instance of Student with the passed arguments and initial
state of the student
        *
        * @return new instance of a student identified with the passed ID, if already does
not exists,
        * and the other arguments as initial state if it's cerated or
        * null in another case
        */
        public Student createStudent(int id, String firstName, String lastName, String
facNumber)


        /**
        * Delete a student with the passed ID
        *
        * @return <code>true</code> only in cae the student was remvoed
        */
        public boolean deleteStudent(int id);


        /**
        * Create a new assistance in the univertity withthe passed arguemtns as initial state
        *
        * @return new created professor assistance identified withthe passed ID, if already
does not exists with that ID
        */
        public Lector createAssistance(int id, String firstName, String lastName);


        /**
```

```
        * Delete an professor asisstance with the passed ID, if such exists
        *
        * @return <code>true</code> ONLY in case the assistance was removed
        */
        public boolean deleteAssistance(int id);


        /**
        * Aighn an assistance to a course
        *
        * @return <code>true</code> ONLY n case the assistance was succesfully assighed
to the course
        */
        public boolean asighAssistanceToCourse(Lector assistance, Course course);


        /**
        * Aighn a professor to a course
        *
        * @return <code>true</code> ONLY n case the professor was succesfully assighed
to the course
        */
        public boolean asighProfessorToCourse(Lector professor, Course course);


        /**
        * Add a studnt to a course
        *
        * @return <code.true</code> ONLY inca se the student is successfully added to the
course
        */
        public boolean addStudentToCourse(Student student, Course course);


        /**
        * Add all students to a course
        *
        * @return <code>true</code> ONLY inc ase all students are added to a course
        */
        public boolean addStudentsToCourse(Student[] students, Course course);


        /**
        * Remvoe a student from a course
        *
        * @return <code>true</code> only in case the student was succesfully removed
from a course
```

```
        */
        public boolean removeStudentFromCourse(Student student, Course course);
}
```

6. Create class UniManagementImpl implements the interface UniManagement which will manage the univerity courses and students as:

Define an array of Students - it will save all students in the course. Max 1000!
Define an array of Courses - it will save all courses. Max 10!
Define an array of Lectors - it will save all assistancences
Define an array of Lectors - it will save all docents and professors

Also, for each array define an index for pointing the last used index for the array, whcih could be used for insetting of a new item.
This index will be used for next purposes:
- when you add a new item (a course, a student or lector) to the relevant array, you will also increment the index with one
- when you remove an item, you will decrement that index
Eg the indexes could be inserted as:
- int private lastUsedStudentIndex;
- int private lastUsedCourseIndex;
- int private lastUsedAssistanceIndex;
- int private lastUsedLectorIndex;

- create a new Course via next method signature:

public Course createCourse(String courseName);

this method makes next operations:
1. in for loop cycle check if there is an already existing course with the passed courseName.
2. if there is no such a course, create new instance of Course AND save the returned Course instance in the array with courses.
In another case, throws an exception (or print in the console an error message).
3. increment lastUsedCourseIndex with 1. so, the index will point next index in the array, where a new student could be inserted

- delete a Course via next method signature:

public boolean deleteCourse(String courseName);

this method makes next operations:
1. in a for look cycle check if there is a course with the passed courseName. If there is then remove that course from the array of courses.
2. Also, when you remove a course, shift all remains course with one index behind it. This practise will be for all arrays in the application

3. Decrement the lastUsedCourseIndex with 1. So, to point the index where next course could be inserted in the array.

An example: you have array with next filed indexes |0|1|2|3| and if you remvoe the element in index 1, you will shift the elements for indexes 2 and 3 with one behind (at place 1 and 2)
And the end you will have next array with indexes |0|1|2| and the lastUsedCourseIndex will point 3 (so, next free index where an element could be inserted)

- create a new Student via next method signature:

public Student createStudent(int id, String firstName, String lastName, String facNumber)

Again - add the new created student to the array of students

- delete a student via next method signature:

public boolean deleteStudent(int id);

Delete the student from the array of students with the passed ID and again, as it was done with courses array, shift the students

- create an assistance with next method signature:

public Lector createAssistance(int id, String firstName, String lastName);

Add the new assistance to the array of assistances as well

- delete an assistance with next method signature:

public boolean deleteAssistance(int id);

- asigh a course to an assiatnce as:

public boolean asighAssistanceToCourse(Lector assistance, Course course);

- assign a course to an professor as:

public boolean asighProfessorToCourse(Lector professor, Course course);

- add a student to a course as:

public boolean addStudentToCourse(Student student, Course course);

NOTE: when you add a student to a course, also have in mind to update the array of courses for in Student instance, also the array of students in Course instance

- add many students to a course as:

public boolean addStudentsToCourse(Student[] students, Course course);

Please for all above operations for adding of a student to a course, check the maximum number of students per course.
Throw an exception if the max exceed, to print error message in the console

- remove a student from a course as:

public boolean removeStudentFromCourse(Student student, Course course);

- Create a test class where you create students, create sources, lectoers and asigh lectors to courses, add and remove stiudents in;form courses and etc

---

Command line augments.

The program could be run via simply main method or via .jar executable file. Either of the both are fine. You need to implement the command pattern here – where we will accept command from the user input (console based), the program will recognize the command, extract the arguments and then execute the requested command.

Wee can create students, courses, lectors, to a sigh students to a course, to remove and etc via passing command line arguments to the .jar file or via the main method

Let's define next commands with arguments of the program as:

- createStudent <user_id> <userFirstName> <userLastName> <facNumber>

Example commands: createStudent 1 Pesho Peshev 34565
This command will create a new student

- createCourse <courseName>

Example commands: createCourse java
This command will create a new course in the university

- createAssistance <user_id> <userFirstName> <userLastName>

Example: createAssistance 5 Marya Petkova

- createProfessor <user_id> <userFirstName> <userLastname> <lectorType>

Example: createProfessor 10 Dran Drankov DOCENT

- asighStudentToCourse <student_id> <courseName>

Example: asighStudentToCourse 1 java
Register a student with passed ID to the course with passed name

- asighStudentToCourse <assistance_id> <courseName>

Example: asighAssistanceToCourse 5 java
Register an assistance with passed ID to the course with passed name

- asighProfessorToCourse <lectort_id> <courseName>

Example: asighProfessorToCourse 10 java
Register a lector with passed ID to the course with passed name

Please in order to implement the arguments line command use another class ArgumentParser, which will
define as constant all possible commands as:

publc static final String CREATE_STUDENT = "createStudent";

and all the others ...

Use these constants to check if the user input current command as argument and only then check the arguments for each command
and use these arguments for execution for the command (creation of a course, a sighing student to a course and etc.)