CS 102 - Project

ÖZYEĞİN ÜNİVERSİTESİ

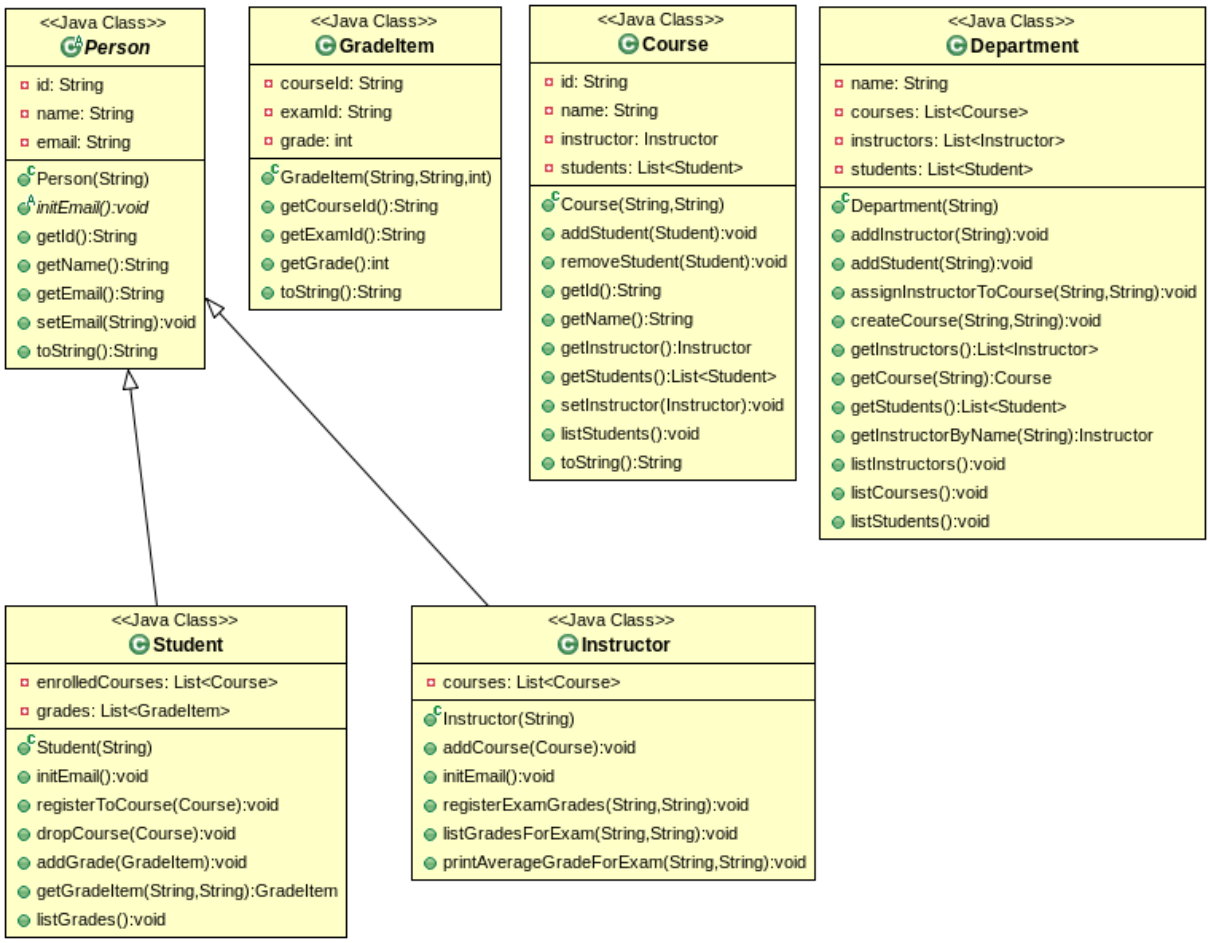
Project Due Date: Friday, December 29, 2017, 23:55

(Late submissions will not be accepted)

Part 1

For the term project you are going to implement a simplified student and course management application similar to LMS or SIS. This is the first part of your project. In this part you will implement the model part of your project. In the second part of the project you are going to integrate it to a GUI. The second part of the project will be uploaded to LMS on November 27, 2017. You should start working on the project as soon as possible. This will be a practice for Midterm 2 as well. There won't be an extension. Late submission won't be accepted.

You will be implementing the classes given in the UML diagram in the figure below. The description of the classes and their methods is briefly summarized. A sample main class and its output is provided for your convenience.



Class Explanations

Usual methods such as *get*, *set*, *add*, *remove*, *toString* and *constructors* should have the usual behaviour unless explicitly stated. *toString* formattings should be similar to given example output. For the printing utilize *toString* methods.

All methods should print error messages whenever necessary, this will be tested exhaustively.

Person

- *Constructor* takes the name parameter as String. You can assume that name will always have two words (no middle names).
- *initEmail()* needs to be implemented by subclasses.
- *id* fields are generated when a Person object is created. It will be a String with two letters followed by 6 digits. Similar to LMS system, letters correspond to the initials of the person's name and number part is generated starting from 0001 and increases whenever a Person is created to guarantee the uniqueness. (Hint: For taking the initials you can use String.split, to generate the id use a static method or variable similar as a counter).

Instructor

- *initEmail()* creates and sets an email address for the instructors. Instructors have the email address with <firstname>.<lastname>@ozyegin.edu.tr, e.g.: reyyan.yeniterzi@ozyegin.edu.tr
- *registerExamGrades(String courseId, String examId)* function is used to enter student grades for exam with id *examId* of the course with id *courseId*. In this function, a *GradeItem* object having a random grade between 0-100 is created for each student enrolled to the course with given *courseId* for exam with given *examId*. Later this *GradeItem* is added to student's grades.
- *listGradesForExam(String courseId, String examId)* prints the grade list of students for the given *courseId* and *examId*.
- *printAverageGradeForExam(String courseId, String examId)* prints the average grade for the given *courseId* and *examId*.

Student

- *initEmail()* creates and sets an email address for the students. Students have the email address with <firstname>.<lastname>@ozu.edu.tr, e.g.: ali.yilmaz@ozu.edu.tr
- *registerToCourse(Course course)* student registers to the given course. Note that, even though a course exists, a student cannot register to that course unless an instructor is assigned to the course. For simplicity, you can assume that the student will not register to a course after an exam is administered by the instructor.
- *dropCourse(Course course)* student drops the course. (Hint: Do not forget to update other objects)
- *getGradeItem(String courseId, String examId*) returns the *GradeItem* object with the given parameters.
- *listGrades()* list all exam grades for all the enrolled courses.

Course

- *Constructor's* parameter order is courseId and courseName
- *listStudents()* prints the names of the enrolled students

GradeItem

- *Constructor*'s parameter order is courseId, examId, grade

Department

- *assignInstructorToCourse(String instructorName, String courseId)* assign an instructor to the course with the courseId. Only the assigned instructor has permission to perform operations related with that course. You can assume there will not be any reassignment for the course.
- *createCourse(String courseId, String courseName)* creates a course. The course will not allow any enrolments until an instructor is assigned.
- *getCourse(String st)* returns the course with id or name equal to the given String st
- list* methods simply print the list similar to other classes

Submission

Submission instruction will be provided in the second part of the project.

Important Notice

- You can add extra classes, extra methods or fields to your classes if you need any. You can also change, add, remove any existing private methods or private fields if you think it makes sense (remember that Main class can only interact with your classes through public methods, so it will not be a problem doing so).

- Provided Main class does not contain all test cases. Unless stated otherwise, you need to think of and handle the edge cases and give proper error messages when necessary.

- Plagiarism will not be tolerated. All plagiarised projects will receive -100. Do all the implementation by yourself, do not share any part of your code for any reason.