Exercise 9
– Modeling with specifications –
(Exam question from 2006)

Your client gives you the following description of the problem domain.

Universities have unique names, for instance "ETHZ". Universities offer courses. At every university, each course has a unique identification number. Courses are worth a certain number of credit points for students who complete them. Courses can be mandatory or non-mandatory for students. Each mandatory course is worth more than five credit points. A course can be offered at multiple universities.

Universities have enrolled students. Students can enroll in multiple universities. They take courses offered by one of the universities they are enrolled in. Students can have courses which they have already completed and courses they are currently attending. A course is only given in a certain semester if at least five students are attending it. A student is registered with a unique identification number at a given university. Furthermore, students' names and addresses are stored.

Universities have employed lecturers who teach courses. A course is taught by exactly one lecturer and a lecturer can teach at most four different courses. Lecturers hold unique identification numbers (unique even considering student identification numbers), furthermore their names, addresses, and office numbers are stored. Lecturers can be employed by multiple universities.

Your tasks:

1. Create a UML class diagram that models the above description. The class diagram has to use exactly the following classes: University, Course, Person, Student, and Lecturer. You do not have to include any methods.

2. Provide OCL invariants for the classes in your class diagrams. The invariants have to express the constraints in the above description. General well-formedness conditions of the data structures do not have to be specified.
   You can use operation **allInstances** which is applied on a class and yields the set of all instances of the class in hand. For instance, University.**allInstances**() yields the set of all objects of type University.
   Furthermore, you can use "<>" for inequality and **not** for logical negation.

3. Assume a method `readCourse(c: Course): void` in class Student with the intuitive meaning of a student registering for course `c`. Specify this behavior by an OCL contract.