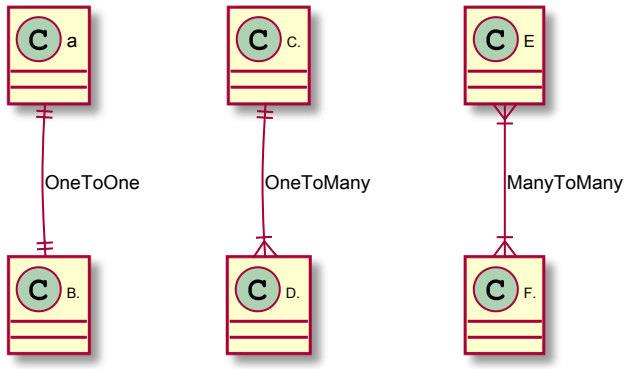
School's out

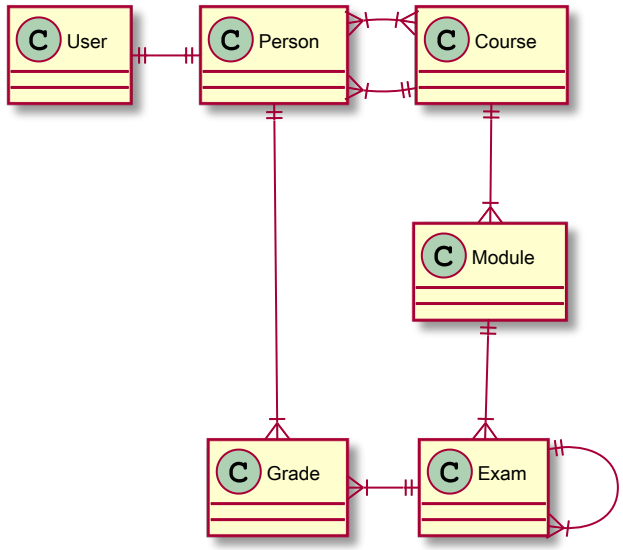
Part 0: Overview and explanation about UML

UML description



Exercise overview

The UML below is an overview of the entire exercise and serves as a reference. It is not the intention that this (may but must not) be fully worked out from the first day.



As you may already deduce from the UML diagram, we are going to write a school administration application in our project. In the table below you will find a little more explanation about the different tables / entities.

The project will consist of several parts. You are supposed to commit and share each part with your instructors via github.

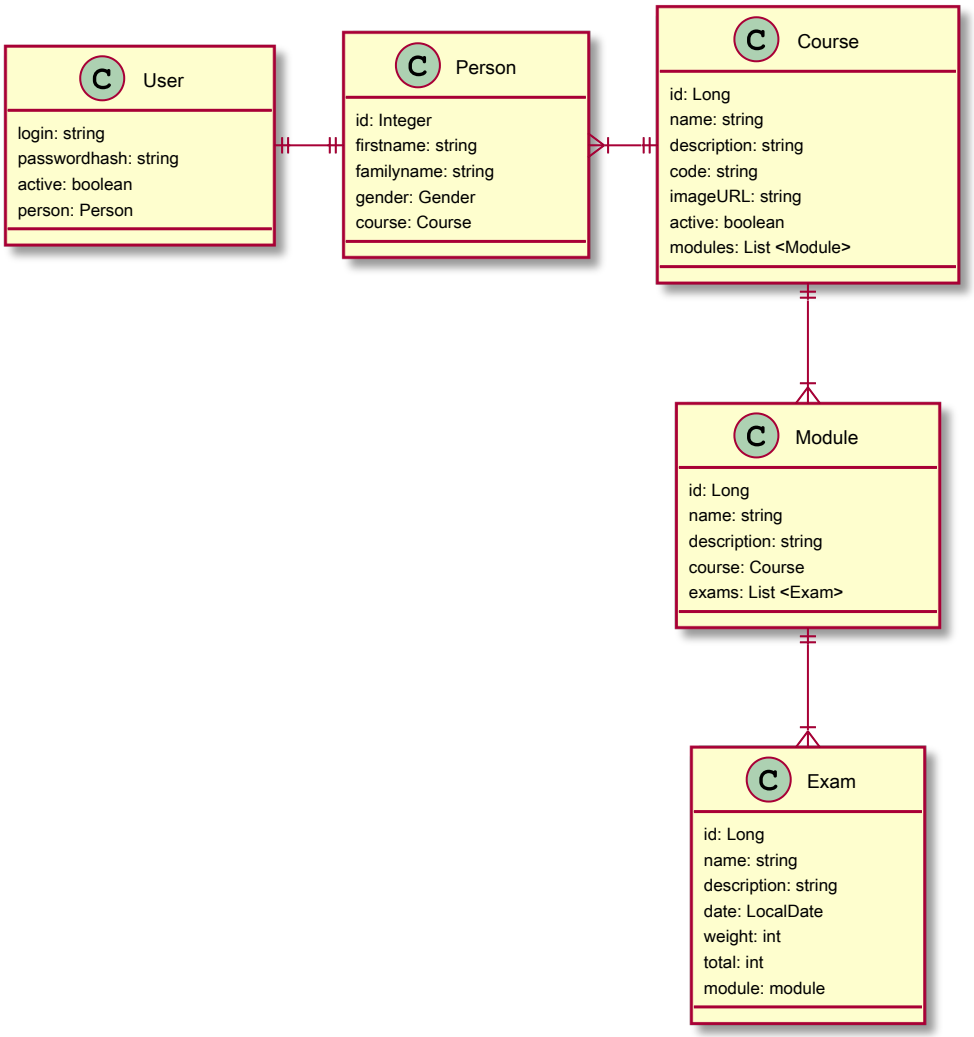
Table	Description
Person	The Person entity will keep our personal data
User	The User entity keeps track of all information about the user logins. 1 login per person
Course	The Course entity describes the course that a Person follows. At first we will only keep track of what the current course is. Later we will also add a 'history'
Module	The Module entity describes the different modules that make up a course.
Exam	The Exam entity keeps track of the various exams or tests that are taken. Initially we will keep 'simple' exams in our program. Later we will add exams that can consist of several parts.
Grade	The Grade entity keeps track of the individual scores of the tests and the people who take them.

Part 1: Descend into madness

Set up a new JPA / Hibernate project. Use your personal database (the same as for the SQL / JDBC exercises) to set up the connection.

Create the entities below and provide the necessary classes to write and read instances of these classes to the database. For now, we only provide CRUD operations.

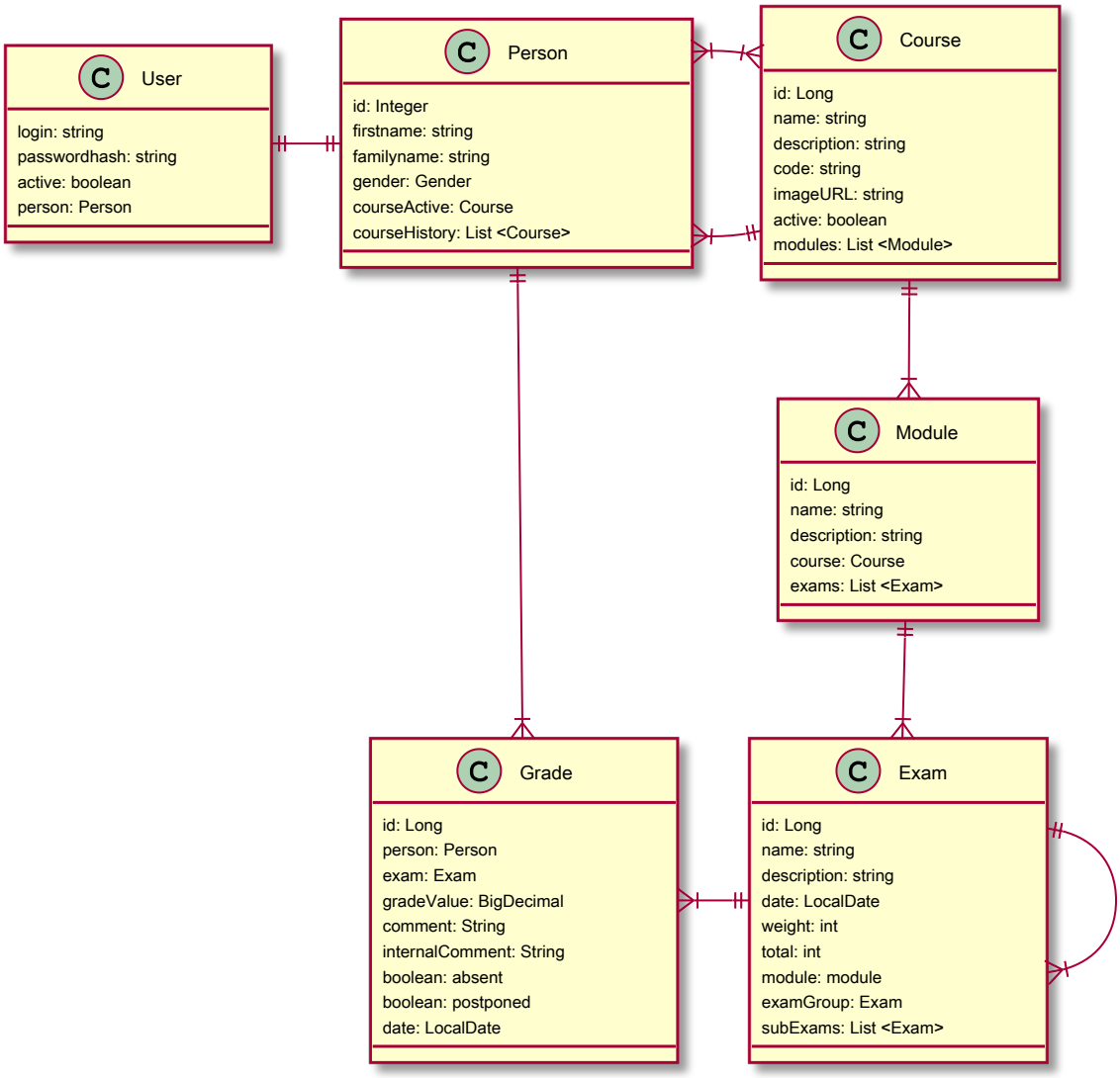
- Person
- Course
- Module
- Exam



Extra information:

- The database should have 5 tables if you let Hibernate create these tables. All numeric IDs are autonomous
- numbered
- The PK for the User class is login
- Descriptions for each must contain a minimum of 2000 chars

Part 2: Answerring the call of Cthulhu



- new class: Grade
- new relationships:
    - exam -> exam: Exams can consist of several 'sub' exams. The 'parent' exam will not contain grades. (This should not be enforced)
    - person -> course: a person can have taken several exams in the past. Also provide a repository for the Grade class
-
- Write an ExamService with 1 method: void outputExam (Long id)

  -> the outputExam method writes an exam to the console WITH its subentities.

  -> so if an exam has grades you also write the grades to the console (without the person)

  -> if an exam contains sub-exams, print the exam and the sub-exams with the grades.