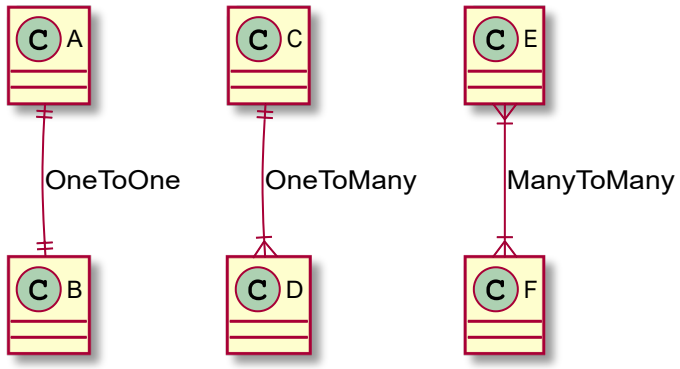
School's out

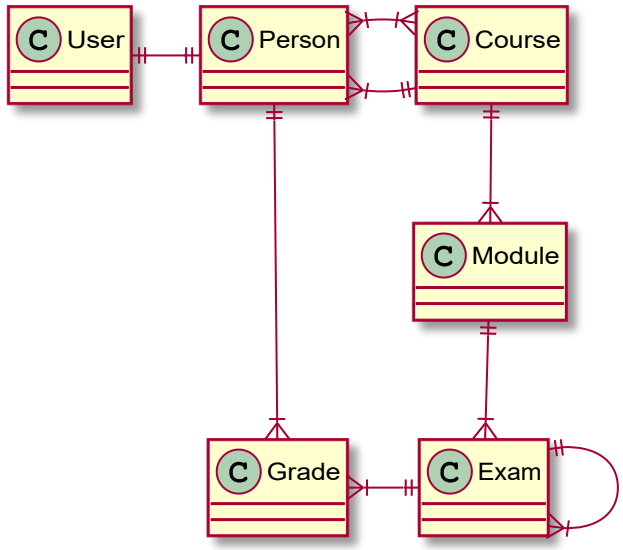
Part 0: Overview en uitleg over UML

UML beschrijving



Overzicht oefening

De onderstaande UML is een overzicht van de volledige oefening en dient als referentie. Het is niet de bedoeling dat deze (mag maar moet niet) compleet wordt uitgewerkt vanaf de eerste dag.



Zoals je misschien al kan afleiden van het UML diagram gaan we in ons project een school administratie applicatie schrijven. In de onderstaande tabel vind je een beetje meer uitleg over de verschillende tabellen/entiteiten.

Het project zal bestaan uit meerdere delen. Het is de bedoeling dat je elk deel commit en deelt via github met je instructeurs.

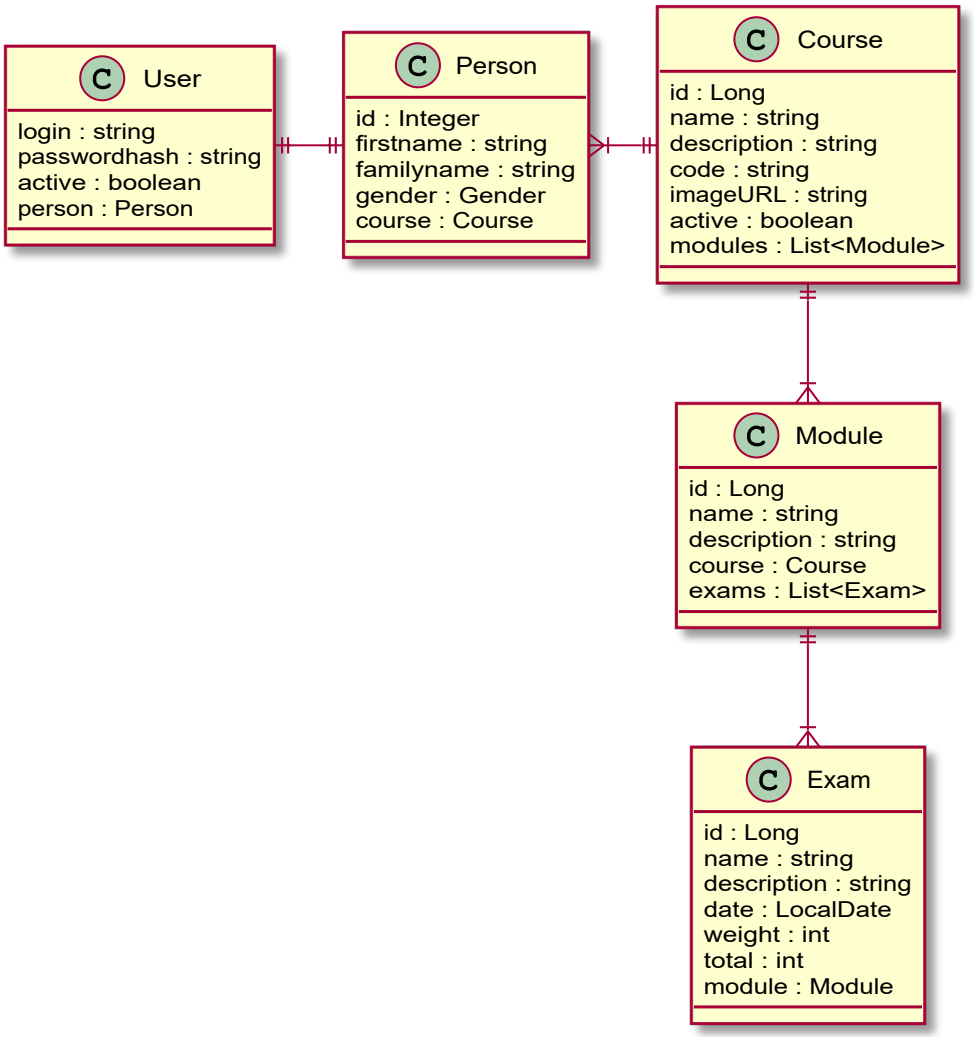
Table	Description
Person	De Person entity gaat onze persoon gegevens bijhouden
User	De User entity houdt alle informatie over de user logins bij. 1 login per persoon
Course	De Course entity beschrijft de cursus die een Persoon volgt. In eerste instantie gaan we enkel bijhouden wat de huidige cursus is. Later zullen we ook een 'geschiedenis' toevoegen
Module	De Module entity beschrijft de verschillende modules waaruit een cursus bestaat.
Exam	De Exam entity houdt de verschillende examens of testen bij die afgenomen worden. Initieel zullen we 'simpele' examens bijhouden in ons programma. Later gaan we examens toevoegen die kunnen bestaan uit meerdere delen.
Grade	De Grade entity houdt de individuele scores bij van de testen en de personen die ze afleggen.

Part 1: Descend into madness

Zet een nieuw JPA/Hibernate project op. Gebruik je persoonlijke database (dezelfde als voor de SQL/JDBC oefeningen) om de connectie op te zetten.

Maak de onderstaande entiteiten aan en voorzie de nodige classes om instances van deze classes naar de database te schrijven en te lezen. Voorlopig voorzien we enkel de CRUD operaties.

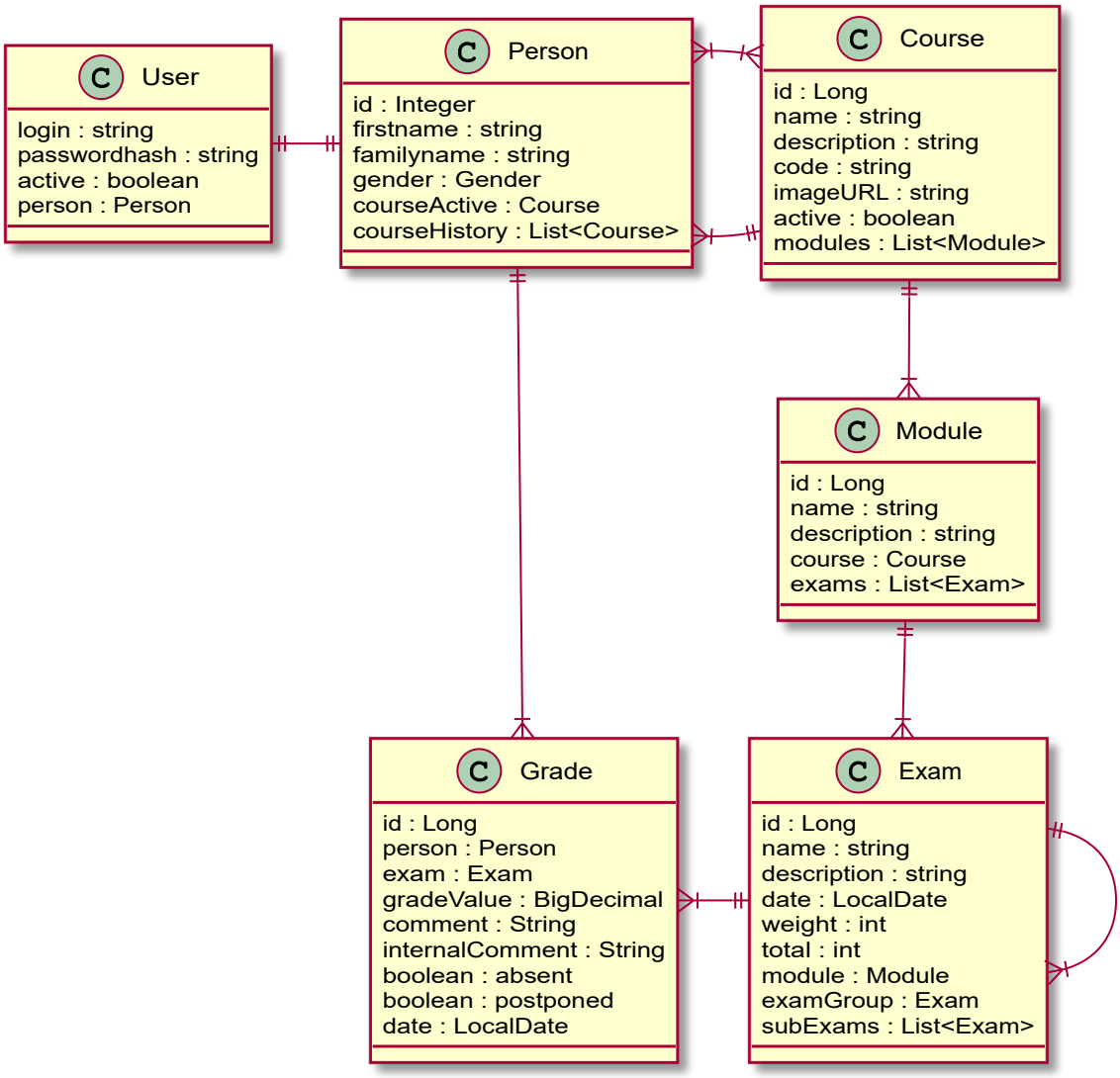
- Person
- Course
- Module
- Exam



Extra information:

- De database zou 5 tabellen moeten tellen als je Hibernate deze tabellen laat aanmaken.
- Alle numerieke id's zijn autonumbered
- De PK voor de User class is login
- Descriptions voor elke moeten minimaal 2000 chars kunnen bevatten

Part 2: Answerring the call of Cthulhu



- new class: Grade
- new relationships:
  - exam -> exam : Examens kunnen bestaan uit meerdere 'sub'-examens. De 'parent' exam zal geen grades bevatten.. (dit moet niet enforced worden)
  - person -> course : een persoon kan meerdere examens gevolgt hebben in het verleden..
- Voorzie ook een repository voor de Grade class
- Schrijf een ExamService met 1 methode: void outputExam(Long id)
  -> de outputExam method schrijft een examen naar de console MET zijn subentities.
  -> dus als een examen grades heeft schrijf je ook de grades naar de console (zonder de person)
  -> als een examen subexamens bevat, druk je het examen af en de subexamens met de grades..