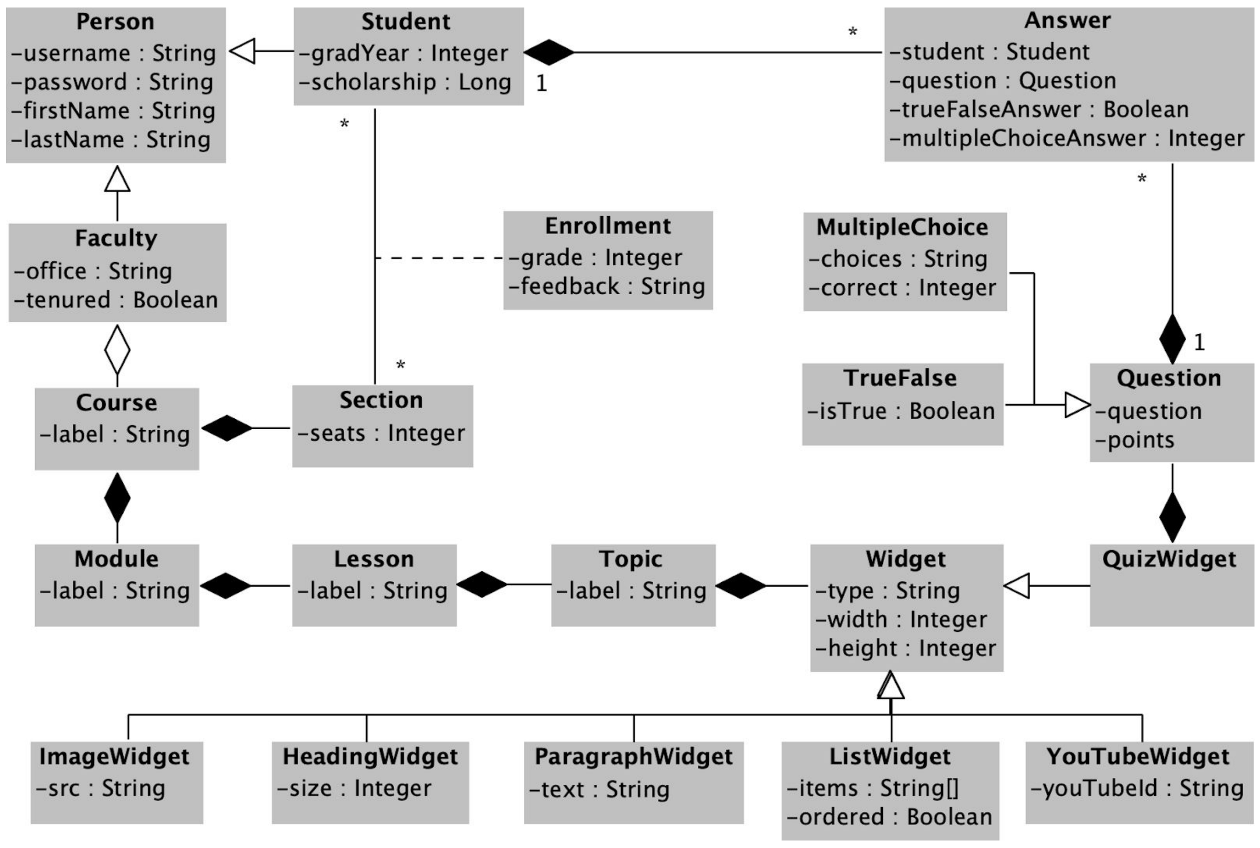
JPA Assignment

Introduction

Consider the class diagram below. This assignment will map the following classes to equivalent JPA entities: Person, Faculty, Student, Course, Section and Enrollment. Feel free to add additional fields or rename them where necessary.



Implement JPA data model (40pts)

Annotate all classes as JPA entities where appropriate. Name all primary keys as "id" and configure them to auto increment. All JPA entities must have corresponding JPA repositories.

Implement JPA inheritance

Create base class Person and derived classes Faculty and Student. Use JPA's single table strategy to implement the inheritance relationship.

Implement JPA one to many

Create classes Course and Section annotating them as JPA entities implementing their one to many relationship using JPA. Also implement the one to many relationship between Faculty and Course.

Implement JPA many to many

Create classes Student and Section, annotating them as JPA entities and implementing association class Enrollment as a JPA mapping table between Student and Section.

Create DAOs for each of the entities (40pts)

In a class called UniversityDao, create a DAO that implements the following methods. If you prefer, feel free to breakup the DAO into several DAOs. The DAOs must use the JPA repositories implemented earlier. Do not use JDBC. Feel free to modify the signature of the methods if appropriate and create additional methods if needed. Implement the following update methods:

1. void truncateDatabase() - removes all the data from the database. Note that you might need to remove records in a particular order
2. Faculty createFaculty(Faculty faculty)
3. Student createStudent(Student student)
4. Course createCourse(Course course)
5. Section createSection(Section section)
6. Course addSectionToCourse(Section section, Course course)
7. Course setAuthorForCourse(Faculty faculty, Course course)
8. Boolean enrollStudentInSection(Student student, Section section) - enrolls a student in a section updating the number of seats available and returning true. If the current available seats is zero then the enrollment is refused and method returns false

Implement the following finder methods

1. List<Person> findAllUsers()
2. List<Faculty> findAllFaculty()
3. List<Students> findAllStudents()
4. List<Course> findAllCourses()
5. List<Section> findAllSections()
6. List<Course> findCoursesForAuthor(Faculty faculty)
7. List<Section> findSectionForCourse(Course course)
8. List<Student> findStudentsInSection(Section section)

9. List<Section> findSectionsForStudent(Student student)

Create a test suite that tests your code (20pts)

Create a test suite that uses the DAO(s) creates earlier to test the JPA data model as follows. All passwords are “password” and all usernames are the lowercase of the corresponding first name

1. Empty the database - the test suite must remove all data from the database before running the rest of the tests
2. Creates faculties - the test suite must insert the following faculties before running the tests in the test suite

First Name	Last Name	Office	Tenured
Alan	Turin	123A	True
Ada	Lovelace	123B	True

10. Creates students - the test suite must insert the following students before running the tests in the test suite

First Name	Last Name	Grad Year	Scholarship
Alice	Wonderland	2020	12000
Bob	Hope	2021	23000
Charlie	Brown	2019	21000
Dan	Craig	2019	0
Edward	Scissorhands	2022	11000
Frank	Herbert	2018	0
Gregory	Peck	2023	10000

11. Create courses - the test suite must create the following courses authored by the faculty shown

Title (Label)	Author
CS1234	Alan
CS2345	Alan

CS3456	Ada
CS4567	Ada

12. Create sections - the test suite must insert the following sections for the courses shown

Title	Seats	Course
SEC4321	50	CS1234
SEC5432	50	CS1234
SEC6543	50	CS2345
SEC7654	50	CS3456

13. Enroll students in sections - the test suite must enroll the following students in the sections shown

Student	Section
Alice	SEC4321
Alice	SEC5432
Bob	SEC5432
Charlie	SEC6543

Validate data

14. Validates uses - write a test that validates the total number of users
15. Validates faculty - write a test that validates the total number of faculty
16. Validates students - write a test that validates the total number of students
17. Validates courses - write a test that validates the total number of courses
18. Validates sections - write a test that validates the total number of sections
19. Validates Course authorship - write a test that validates the total number of courses authored by each faculty
20. Validates Section per Course - write a test that validates the total number of sections per each course
21. Validates Section enrollments - write a test that validates the total number of students in each section

22. Validates student enrollments - write a test that validates the total number of sections for each student
23. Validates Section seats - write a test that validates the number of section seats