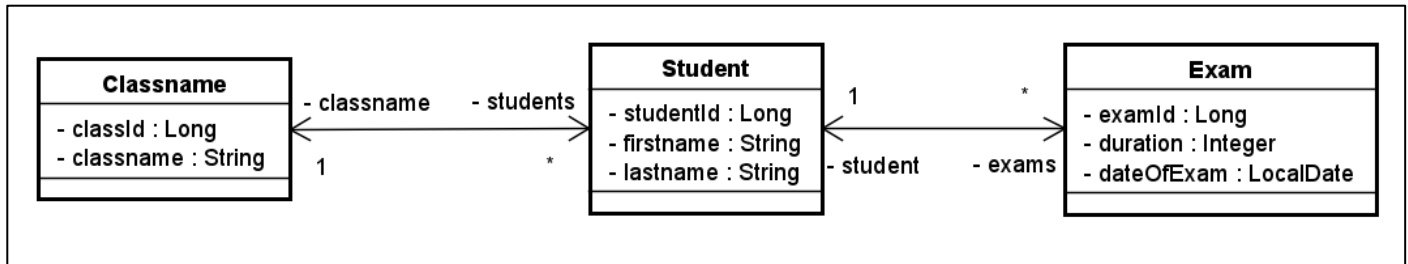


Exa_102_Rest_ExamDBService

1. Requirements

Implement a Rest-Service for an exam-database. The data for the mock-database are provided in a JSON-file that should be read before deployment of the application. Filtering and sorting of data should be implemented with Java-streaming expressions.

Read the Json-data into the following Java-class-structure:



Implement a JSON-based REST API with a Mock-Database and JSON-file support at server side.

2. Programm-description

Structure of the application:

- `web.StudentResource.java`: Backend-Rest-controller-class
- `web.ExamResource.java`: Backend- Rest-controller-class
- `web.ClassnameResource.java`: Backend- Rest-controller-class
- `json.JSON_Access.java`: read/write JSON-Data
- `database.ExamMockDB.java`: mock-database for exams and related classes

Implement the following HTTP-Methods in the **XxxResource** classes:




- **GET** `localhost:8080/classes/all` returns a list of all classnames
- **GET** `localhost:8080/students/all` returns a list of all students
- **GET** `localhost:8080/students?class=1DHIF` returns a list of all students from the specified class
- **GET** `localhost:8080/students?class=1DHIF&orderBy=firstname` returns a list of all students from the specified class, sorted in ascending order by *firstname*. Support ordering by *firstname* and *lastname*.
- **PATCH** `localhost:8080/students/{studentId}`: changes data of the specified student
- **GET** `localhost:8080/exams?studentId=1` returns a list of all exams of a student
- **DELETE** `localhost:8080/exam/{studentId}/{examId}`: removes an exam from a student
- **POST** `localhost:8080/exam/{studentId}`: adds an exam for a student

ExamMockDB.java



Implement all methods required for the Rest-services

JSON_Access.java

-  Read data from JSON-File
-  Assign a unique studentId
-  Make sure that all associations are setup correctly!

Content:

- Rest-Services
- JSON-Mapping
- Java-Streams