Задача

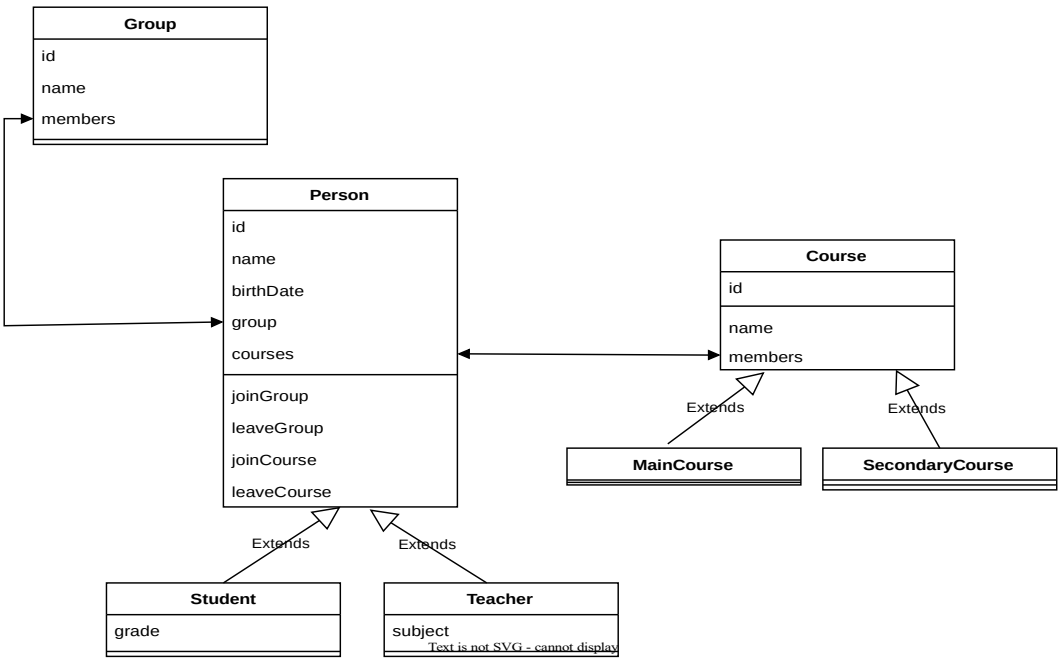
We need an application for storing Students and Teachers information like name, age, group and courses. We have two type of courses -Main and Secondary.
The application should be able to add remove or modify students and teachers. With this application we should be able to create different reports :

- how many students we have
- how many teachers we have
- how many courses by type we have
- which students participate in specific course
- which students participate in specific group
- find all teachers and students for specific group and course
- find all students older than specific age and participate in specific course

The application should provide public API.

Модел

Очевидните класове са Student, Teacher, Group и Course. Student и Teacher имат общи полета и функционалност, които съм извадил в базов клас Person. В него е събрана цялата информация: име, група, списък с курсове и дата на раждане (избрах дата на раждане, вместо възраст, както е по условие, защото е "по-динамично" :)). Тъй като според задачата в класовете Student и Teacher остават празни, добавих по едно поле в тях извън условието на задачата. Всеки обект Person може да участва в една група (Group) и в произволен брой курсове. Връзката Person – Group е ManyToOne, а Person – Course е ManyToMany. И двете зависимости ги направих двупосочни. Двата вида курсове, main и secondary, могат да се маркират с едно boolean поле, но това не е гъвкаво решение, затова направих общ базов клас Course, на които те са наследници. Общата схема изглежда така:



Двупосочните връзки се управляват с методите на класа Person joinGroup / leaveGroup и joinCourse / leaveCourse.

Представянето в базата данни на йерархията Person е с join table, а на йерархията Course – single table.

Всички класове са entity с Integer primary key.

Архитектура

Архитектурата е стандартна layered с три слоя: repositories, services и controllers. В repository интерфейсите са дефинирани и съответните @Query заявки, необходими за решаване на подусловията на задачата.
За комуникация между services и repositories се използват обекти от самите entity класове, а за комуникация между controllers и services – съответни data access (DAO) класове.

API

Controller класовете са стандартни @RestControllers. Има отделни контролери за управление (CRUD операции) съответно на Student, Teacher, Course и Group. Има специални контролер, ReportsController (mapping "/api/reports"), който обслужва заявките, описани в условието на задачата.
Приложението работи на стандартния порт 8080.

База данни

За база данни се използва in memory H2 database (user: sa, password: sa). Самото приложение инициализира данни в базата веднага след стартирането чрез CommandLineRunner. За генериране на достоверни данни съм използвал библиотеката Faker (https://github.com/DiUS/java-faker).