Java loppuprojekti

Yleiskuvaus:

Lopputyönä tehdään sovellus opiskelijoiden kurssien hallintaa varten. Perusrakenne mahdollistaa opiskelijoiden ja kurssien lisäämisen sekä opiskelijoiden rekisteröimisen tietylle kurssille. Työ sisältää luokkarakenteet, tarvittavat toiminnot sekä lisäksi RESTful palvelun. Tarkemmat määrittelyt seuraavilla sivuilla. Pyri käyttämään nimeämisissä englantia.

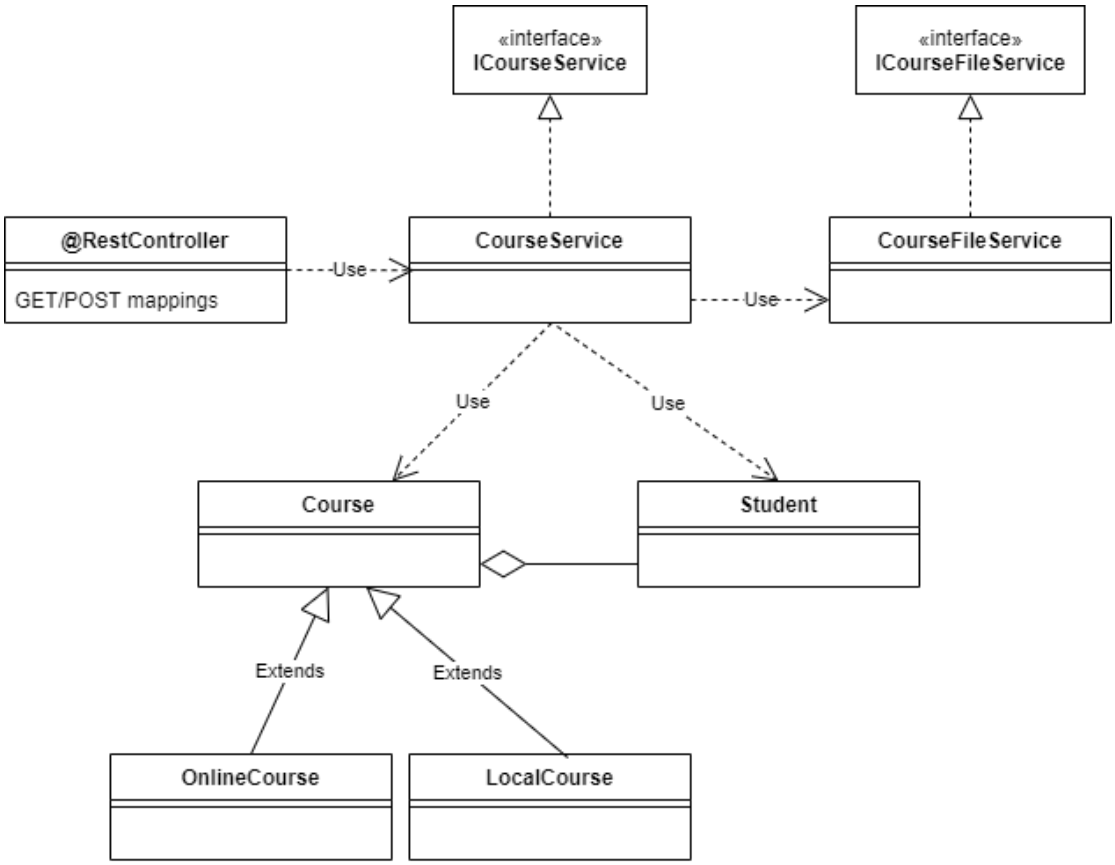
Palautus:

Projekti tulee palauttaa Githubin kautta linkillä siten, että ladatun repon/kansion nimi on **SukunimiEtunimiJava** (Muista julkinen repo!!!)

Huom! Tee vain vaaditut toiminnot. Älä lisää palautettavaan versioon ylimääräistä, koska se hidastaa töiden tarkistamista. (turha koodi saattaa jopa vähentää pisteitä).

Voit tehdä kaiken suoraan Spring Boot -sovellukseen tai tehdä ensin kohdat 1-3 normaalina Java-projektina (ja testata malleja siellä) ja lopuksi siirtää luokat erilliseen Spring Boot -projektiin.

Kuva tueksi tuleviin osioihin:



1. Luokkarakenne

- Ohjelman tietomalli koostuu kahdesta dataluokasta:
    - **Student**
        - Opiskelijalla on yksilöllinen **id**(automaattinen). Lisäksi opiskelijalla on **etunimi** ja **sukunimi**.
        - Opiskelijan tiedot voidaan lukea ja kirjoittaa uudelleen. Id voidaan vain lukea. Lisäksi ylikirjoitetaan **toString**-metodi siten, että se palauttaa opiskelijan tiedon ”Sukunimi Etunimi”.
    - **Course**
        - Course-luokasta ei voida luoda suoraan olioita.
        - Kurssilla on yksilöllinen **id** (automaattinen) sekä kurssin **nimi**, **opettajan nimi** ja **opiskelijalista**.
            - Nimi ja opettajan nimi voidaan lukea ja asettaa. Id voidaan vain lukea.
            - Kurssille voidaan lisätä ja kurssilta poistaa opiskelijoita.
                - Opiskelijan lisääminen palauttaa boolean arvon (false jos kurssille ei voida lisätä esim. kun lähiopetus on täynnä)
            - Kurssilta voidaan pyytää opiskelijalista (ei alkuperäistä listaa, vaan uusi lista, joka sisältää opiskelijat)
    - Aliluokka **OnlineCourse**
        - Lisätietona luokassa verkkokurssin osoite. (get/set)
    - Aliluokka **LocalCourse** (lähiopetus)
        - Luokkatila (get/set)
        - Max. 25 opiskelijaa mahtuu kurssille. Opiskelijan lisääminen ylikirjoitetaan ja tarkistetaan mahtuuko kurssille.
    - Kurssien tulee palauttaa toString-ylikirjoituksella siisti kuvaus kurssista
        - Esim. ”*HTML Fundamentals – Reima Riihimäki – www.moodle.oulu.fi*”

2. Service

Servicen tehtävä on tarjota palveluita kurssidatan hallintaan. Tee seuraava rajapinta ja siitä toteutus (implements) ***CourseService***. Service pitää yllä listaa opiskelijoista ja kursseista ja tarjoaa työkalut niiden käsittelyyn. Kuvan koodikommentteja ei tarvitse lisätä, ne on laitettu vain kuvaamaan metodin tehtävää.

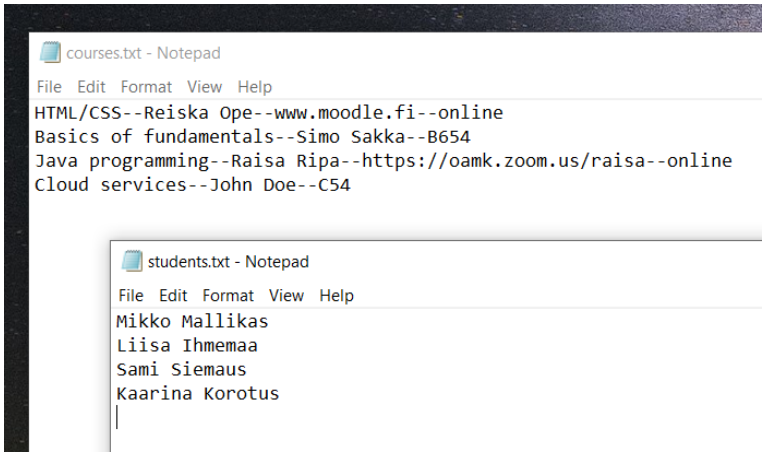
(Huom! Serviceen karsittu vain tässä sovelluksessa tarvittavat toiminnot)

```
public interface ICourseService {
    /**
     * Gets list of students
     */
    List<Student> getStudents();
    /**
     * Gets list of courses
     */
    List<Course> getCourses();
     /**
     * Gets student by ID. Returns null if student not found.
     */
    Student getStudentById(long studentId);
    /**
     * Gets course by ID.Returns null if course not found.
     */
    Course getCourseById(long courseId);
    /**
     * Gets list of courses that the student is registered in
     */
    List<Course> getCoursesOfStudent(long studentId);
    /**
     * Registers the student in the course
     * @return true if the registration was successful
     */
    boolean addStudentToCourse(long studentId, long courseId);
}
```

3. Tiedostonhallinta

Tee ohjelmaan seuraava rajapinta ja siitä toteutus. Toteutuksen kautta voidaan hakea kurssit ja opiskelijat omista tiedostoistaan. Tiedostot ovat tekstitiedostoja (.txt) ja esim. yksi opiskelija/kurssi omalla rivillään. Voit päättää tiedoston rakenteen itse, toimivuus tärkeintä. Alla esimerkit miltä tiedostot voivat näyttää.

```
public interface ICourseFileService {
    /**
     * Reads list of students from txt-file
     */
    List<Student> readStudentsFromFile(String filePath) throws FileNotFoundException;
    /**
     * Reads list of courses from txt-file
     */
    List<Course> readCoursesFromFile(String filePath) throws FileNotFoundException;
}
```



Ota tekemäsi tiedostonhallinta opiskelijarekisterin CourseService:ssä käyttöön. Opiskelija- ja kurssilistat haetaan aina oletuksena aluksi tiedostosta, kun CourseService luodaan.

Huom! Sijoita tiedostot Git-repon kansion juureen. VS Code käyttää työkansiona sitä kansiota, joka on VS Codeen avattu. Toisin sanoen, kun repokansion vetää VS Codeen, tulee tiedoston luvun toimia suoraan.

Huom! Jos et osaa toteuttaa tiedostonhallintaa, luo esimerkkioliot suoraan koodissa käynnistyksen yhteydessä.

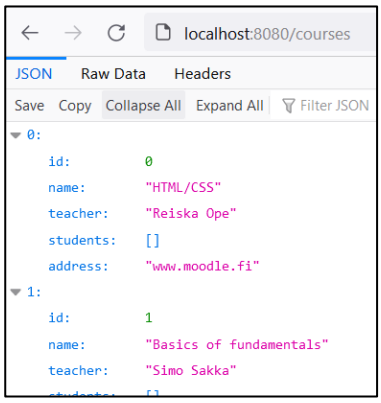
4. Spring Boot REST API

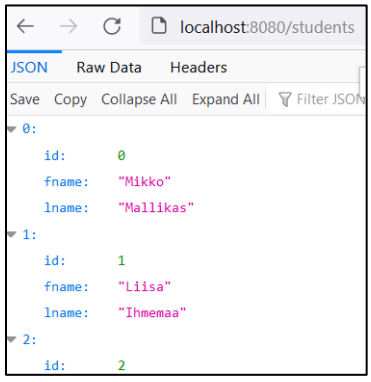
Toteuta REST API (RestController), johon voidaan tehdä opiskelijoista/kursseista kyselyjä selaimen kautta. Käytä toteutuksessa hyödyksi toteuttamiasi dataluokkia ja servicejä..

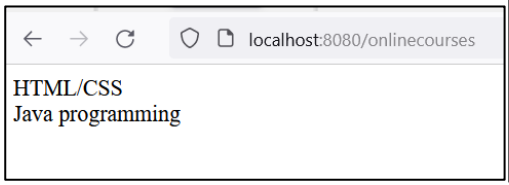
Palvelun tulee toimia seuraavilla osoitteilla:

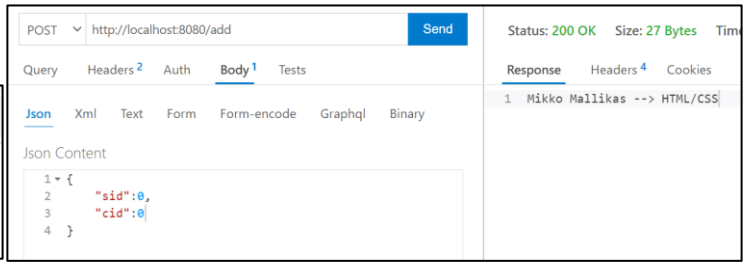
- **localhost:8080/students**
  - Palauttaa tiedot kaikista opiskelijoista. Voidaan näyttää sivulla JSON-muodossa.
- **localhost:8080/courses**
  - Palauttaa tiedot kaikista kursseista. Voidaan näyttää sivulla JSON-muodossa.
- **localhost:8080/onlinecourses**
  - Palauttaa tiedot verkkokursseista. Näytä vain kunkin kurssin nimi sivulla.
- **localhost:8080/students/{id}**
  - Palauttaa id:n perusteella opiskelijan tiedot ja myös kurssit, joissa opiskelija on. Voidaan palauttaa HTML-muodossa merkkijonona.
- **localhost:8080/courses/{id}**
  - Palauttaa kurssin nimen ja siihen rekisteröityneet opiskelijat html-tekstinä
- **localhost:8080/add**
  - Lisää POST-mappauksella opiskelija kurssille JSON-rakenteen perusteelle. JSON annetaan esim. muodossa **{"sid": "0", "cid": "1"}** . sid on opiskelijan id ja cid kurssin id. Testaa toimintaa esim. Postmanilla tai Thunder Clientilla.

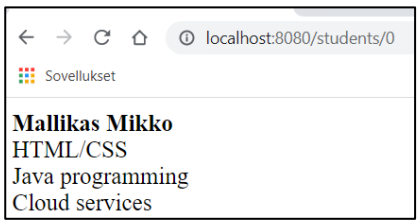
Esimerkkejä miltä sivut voivat näyttää:











5. Pisteytys

Luokkarakenteet ja tarvittavat alustimet jäsenmuuttujat ja operaatiot.	30p
Periytymisen ja rajapintojen hyödyntäminen.	10p
Kontrollerin toteutus ja koosteiden hallinta	20p
Spring Boot REST toteutus	20p
Tiedostonhallinta	15p
Virhetilanteiden järkevä hallinta, koodin luettavuus, nimeäminen jne.	5p