Lab5:

1. From https://start.spring.io/, create:
    - A maven project for Java 8
    - Group: example.com
    - Artifact: restserverjpa
    - Dependencies: Spring Web, Spring Data JPA, H2, MariaDB Driver
2. Download file and uncompress it in a directory, (rename test directory to test1)
3. Open project in an IDE such as IntelliJ
4. Create a DB persistence object that represents the person in the database.
    - Add class representing the database table `impl/Person.java`
    - Add a fields (long id, String name, String surname)
    - use the @Entity before the class definition
    - Use @Id @GeneratedValue for the primary key
5. Add JPA repository for Person
    - Add Java interface PersonRepository (`impl/PersonRepository.java)` that extends JpaRepository<Person, Long>
6. In `src/main/resources/application.properties` add
    - spring.h2.console.enabled=true
    - Optional:
    - spring.jpa.show-sql=true
    - spring.jpa.properties.hibernate.format_sql=true
7. Start application and notice a line similar to:

    "H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:dc52175e-c4db-422c-ab15-edab766f4ba0'
8. In a browser, open location: `http://localhost:8080/h2-console`
    - As url paste `"jdbc:h2:mem:dc52175e-c4db-422c-ab15-edab766f4ba0"`
    - Login "sa"
    - Password is blank
    - Notice the table "PERSON"
9. Iimplement a webserver server to provide API to CRUD operations for the person type, that persist data using JPA:
    - Implement REST controller (PersonController)
        i. Add variable personRepository of type PersonRepository

        ii. Create a constructor that sets personRepository

- The list of persons is stored in PersonController
- Publish GET "/persons", that returns all the persons in the List
    i. Fetch all data from DB using `personRepository.findAll()`
- Publish POST "/persons", that add a person with the data in the JSON RequestBody
    i. Add data to DB using `personRepository.save(newPerson);`
- Publish DELETE "/persons", that takes id as parameter, that deletes the person
    i. Delete data from DB using `personRepository. deleteById(id);`
- Publish PUT "/persons{id}", that updates the details of the person identified by id with the data provided in the JSON RequestBody. Implementation detail (if a person exists in DB update it, else create a new one
    i. Update data in DB using `personRepository.save(employee);`

10. Implement error handling to cater for Person not found (needed by findpersonbyid and modify person)
    - Create class `PersonNotFoundException` that extends `RuntimeException`
        i. Implement a constructor that takes a Long, that calls:

        ```
        super("Could not find person " + id);
        ```
    - Create class `PersonNotFoundAdvise`
        i. Class must have annotation `@ControllerAdvice`
        ii. Implement method below method

        ```
        @ResponseBody

        @ExceptionHandler(PersonNotFoundException.class)

        @ResponseStatus(HttpStatus.NOT_FOUND)

        String personNotFoundHandler(PersonNotFoundException ex)
        {

            return ex.getMessage();

        }
        ```
11. Implement the additional rest methods
    - Publish GET "/persons/{id}", that returns the person with the given ID
        i. Fetch data from DB using `personRepository. findById(id).orElseThrow(() -> new PersonNotFoundException(id));`
12. Run project and test method using Postman