Micro services with Java
Exam

Goal

This exam evaluates your ability to develop, test, and integrate advanced Java applications, focusing on working with CSV data, dependency injection, JPA, REST, and optimization. You may use System.out.println or logging libraries for output as per your preference.

```
project-root/
    output/
    core-module/
        src/main/java
            <your_code>
        src/test/java
            ○ TestMVD2
            ○ TestMVD3
            ○ TestJUN1
            ○ TestJUN2
            ○ TestSPR1
            ○ TestSPR2
            ○ TestJPA1
            ○ TestJPA2
    rest-module/
        src/main/java
            <your_code>
        src/test/java
            ○ TestRST1
            ○ TestRST2
            ○ TestRST3
```

Dataset: Titanic (CSV)

You are provided with the Titanic dataset

```
Name                                                ;PClass;Age ;Sex    ;Survived
Allen, Miss Elisabeth Walton                        ;1st   ;29  ;female;1
Allison, Miss Helen Loraine                         ;1st   ;2   ;female;0
Allison, Mr Hudson Joshua Creighton                 ;1st   ;30  ;male  ;0
Allison, Mrs Hudson JC (Bessie Waldo Daniels)       ;1st   ;25  ;female;0
```

Domain 1: Maven and Data

Implement this domain in the core-module

MVD1- Project Setup

Task:

- Set up a Maven project with dependencies for Spring Boot, OpenCSV (or Apache Commons CSV, or manual processing), and JPA.
- Perform a mvn clean install and save the output to output/mvd1.out.

MVD2- Data Modeling

Create a Passenger class with the following fields:

```
Name (String)
Pclass (String)
Age (double)
Sex (String)
Survived (boolean)
```

Create a test class (main) **TestMVD2** to initialize a Passenger instance. Print the Passenger details to the console

MVD3- CSV Data Import

Write a service class PassengerService to read the titanic.csv file and create a list of Passenger objects.

Create a test class (main) **TestMVD3** to test this functionality. Print the list of passengers to the console

Domain 2: JUnit

JUN1- BDD

Make sure junit is in the maven dependencies.

- Convert TestMVD3 into a new JUnit test class. Name it **TestJUN1**.
- Assert that the number of passengers in the list is correct

JUN2- Folder Scanning

Enhance PassengerService to scan a directory for CSV files and load all passengers into memory.

Test this functionality with multiple CSV files (train and test files) in a folder using **TestJUN2**.

Domain 3: Dependency Injection (DI)

SPR1- DI Basics

Set up Spring Dependency Injection with a configuration class.

Inject a String bean with the value "Spring Dependency Injection for Titanic!" and print it in the console. Make this call from **TestSPR1** class.

SPR2- JDBC with DI

Write a DAO class PassengerDAO to save Passenger objects into an H2 in-memory database using JDBC.

- Configure and inject a javax.sql.DataSource bean into the DAO.
- Combine the CSV loading functionality from MVD3 with the DAO to persist passengers. Test this in **TestSPR2** and print in the console to confirm the passengers are stored.

Domain 4: JPA with Hibernate

JPA1- JPA Context

Set up Hibernate with JPA, annotate the Passenger class, and configure an EntityManager.

Persist a Passenger object and verify its insertion in **TestJPA1** using correct assertions.

JPA2- Repository

Create a PassengerRepository class (equivalent of DAO, but using JPA) to manage Passenger entities. Include a savePassenger method.

Test the repository functionality in **TestJPA2**. Use correct assertions

Domain 5: REST

Do the following in a separate module “rest-module”

RST1- Spring Boot Setup

Initialize a Spring Boot application in **TestRST1** (main)

RST2- REST Controller

Create a REST controller PassengerController with GET and POST endpoints for Passenger objects.

Verify these endpoints in **TestRST2** (main) using console output. You can use a http file to trigger the calls.

RST3- Full Integration

Integrate the data access layer with PassengerController. Test POST and GET requests end-to-end in TestRST3 (junit). Print results to the console. Write a http client code using this reference:

https://www.baeldung.com/rest-template