Web API Design with Spring Boot Week 3 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Here's a friendly tip: as you watch the videos, code along with the videos. This will help you with the homework. When a screenshot is required, look for the icon: 🖥️ You will keep adding to this project throughout this part of the course. When it comes time for the final project, use this project as a starter.

Project Resources: **https://github.com/promineotech/Spring-Boot-Course-Student-Resources**

Coding Steps:

1) In the application you've been building add a DAO layer:
    a) Add the package, com.promineotech.jeep.dao.
    b) In the new package, create an interface named `JeepSalesDao`.
    c) In the same package, create a class named `DefaultJeepSalesDao` that implements `JeepSalesDao`.
    d) Add a method in the DAO interface and implementation that returns a list of Jeep models (class `Jeep`) and takes the model and trim parameters. Here is the method signature:

```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```

2) In the Jeep sales service implementation class, inject the DAO interface as an instance variable. The instance variable should be `private` and should be named `jeepSalesDao`. Call the DAO method from the service method and store the returned value in a local variable named `jeeps`. Return the value in the `jeeps` variable (we will add to this later).

3) In the DAO implementation class (`DefaultJeepSalesDao`):

   a) Add the class-level annotation: `@Service`.

   b) Add a log statement in `DefaultJeepSalesDao.fetchJeeps()` that logs the model and trim level. Run the integration test. Produce a screenshot showing the DAO implementation class and the log line in the IDE's console. 💻

   c) In `DefaultJeepSalesDao`, inject an instance variable of type `NamedParameterJdbcTemplate`.

   d) Write SQL to return a list of Jeep models based on the parameters: model and trim. Be sure to utilize the SQL Injection prevention mechanism of the NamedParameterJdbcTemplate using `:model_id` and `:trim_level` in the query.

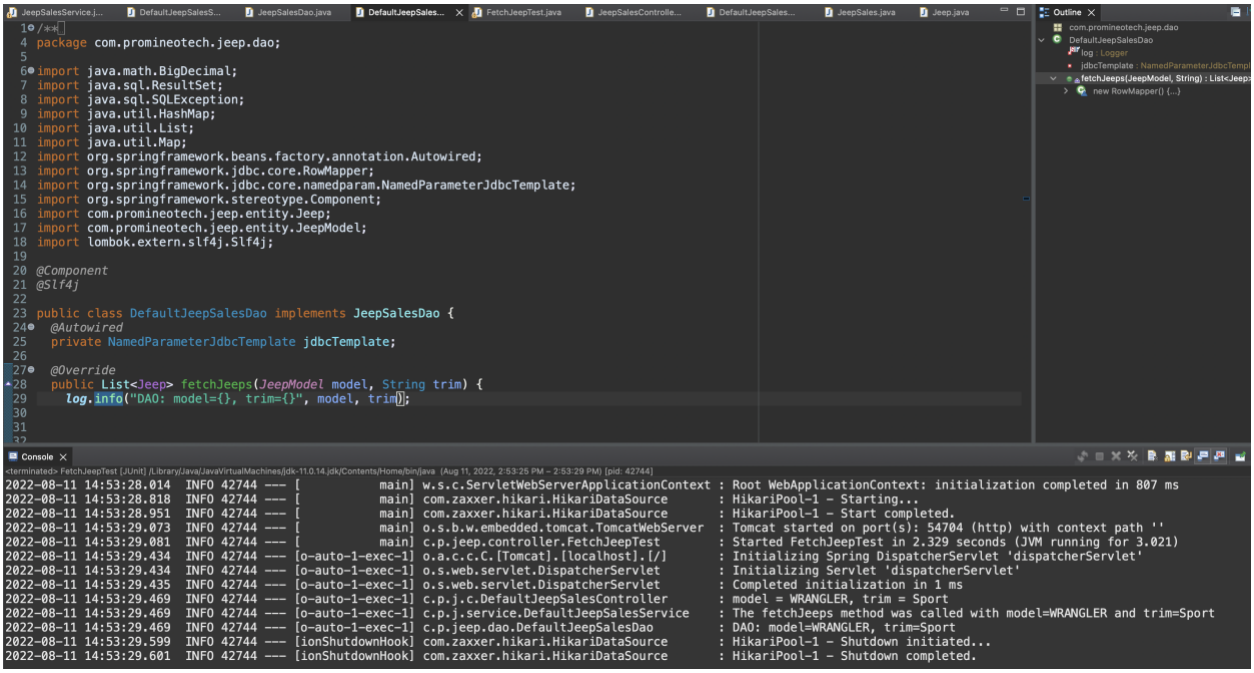
   e) Add the parameters to a parameter map as shown in the video. Don't forget to convert the JeepModel enum value to a String (i.e., `params.put("model_id", model.toString());`)

   f) Call the `query` method on the `NamedParameterJdbcTemplate` instance variable to return a list of Jeep model objects. Use a RowMapper to map each row of the result set. Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a screenshot to show the complete method in the implementation class. 💻

4) Add a getter in the `Jeep` class for `modelPK`. Add the `@JsonIgnore` annotation to the getter to exclude the `modelPK` value from the returned object.

5) Run the test to produce a green status bar. Produce a screenshot showing the test and the green status bar. 💻

Screenshots of Code:



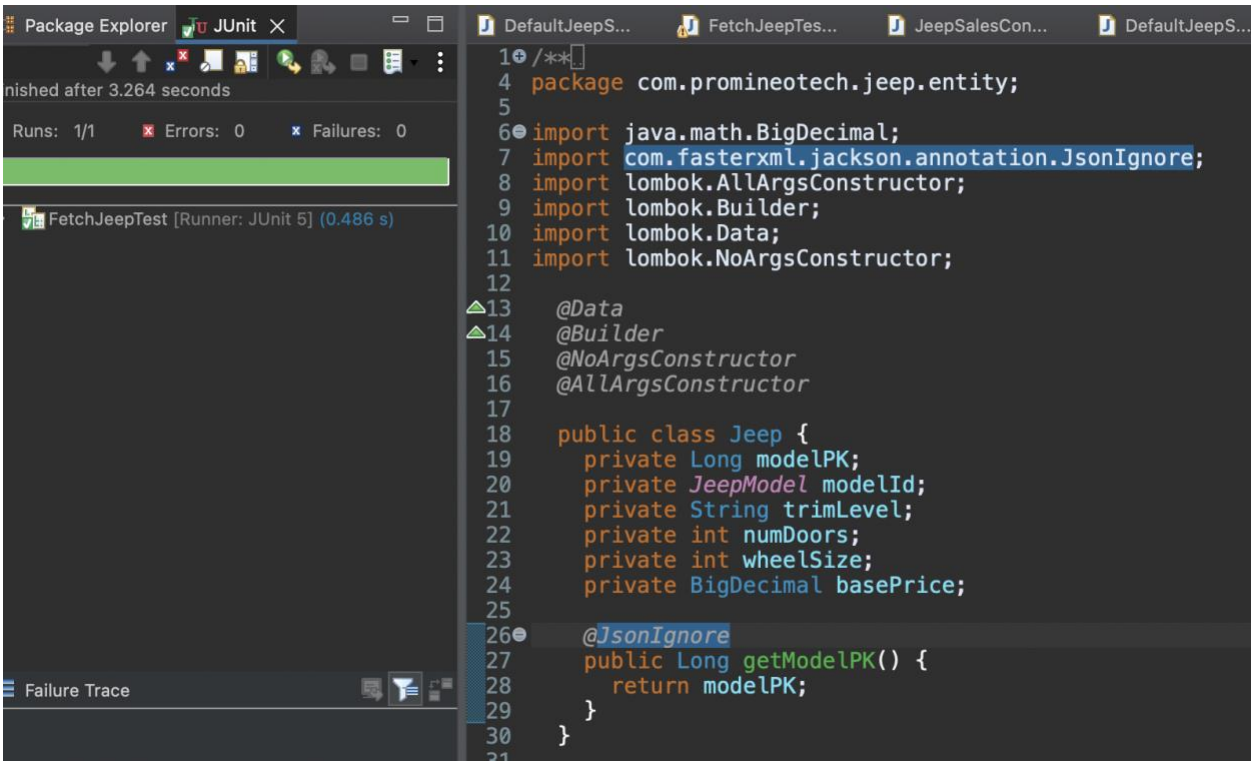
```
package com.promineotech.jeep.dao;

import java.math.BigDecimal;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Component;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j

public class DefaultJeepSalesDao implements JeepSalesDao {
  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;

  @Override
  public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    log.info("DAO: model={}, trim={}", model, trim);
```

```
2022-08-11 14:53:28.014  INFO 42744 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 807 ms
2022-08-11 14:53:28.818  INFO 42744 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2022-08-11 14:53:28.951  INFO 42744 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2022-08-11 14:53:29.073  INFO 42744 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 54704 (http) with context path ''
2022-08-11 14:53:29.081  INFO 42744 --- [           main] c.p.jeep.controller.FetchJeepTest        : Started FetchJeepTest in 2.329 seconds (JVM running for 3.021)
2022-08-11 14:53:29.434  INFO 42744 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-08-11 14:53:29.434  INFO 42744 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-08-11 14:53:29.435  INFO 42744 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2022-08-11 14:53:29.469  INFO 42744 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController       : model = WRANGLER, trim = Sport
2022-08-11 14:53:29.469  INFO 42744 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService    : The fetchJeeps method was called with model=WRANGLER and trim=Sport
2022-08-11 14:53:29.469  INFO 42744 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao         : DAO: model=WRANGLER, trim=Sport
2022-08-11 14:53:29.599  INFO 42744 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2022-08-11 14:53:29.601  INFO 42744 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Note: I used log.info because log.debug would not display the message.

```java
/**
 */
package com.promineotech.jeep.dao;

import java.math.BigDecimal;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Component;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j

public class DefaultJeepSalesDao implements JeepSalesDao {
  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;

  @Override
  public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    log.info("DAO: model={}, trim={}", model, trim);

  //@formatter:off
    String sql = ""
        + "SELECT * "
        + "FROM  models "
        + "WHERE model_id = :model_id AND trim_level = :trim_level";
  //@formatter:on

    Map<String, Object> params = new HashMap<>();
    params.put("model_id", model.toString());
    params.put("trim_level", trim);

    return jdbcTemplate.query(sql, params, new RowMapper<>() {

      @Override
      public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
        //@formatter:off
            return Jeep.builder()
            .basePrice(new BigDecimal(rs.getString("base_price")))
            .modelId(JeepModel.valueOf(rs.getString("model_id")))
            .modelPK(rs.getLong("model_pk"))
            .numDoors(rs.getInt("num_doors"))
            .trimLevel(rs.getString("trim_level"))
            .wheelSize(rs.getInt("wheel_size"))
            .build();
            //@formatter:on
      }
    });
  }
}
```

Screenshots of Running Application:



URL to GitHub Repository:

https://github.com/lindsey406/week13homework.git