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:

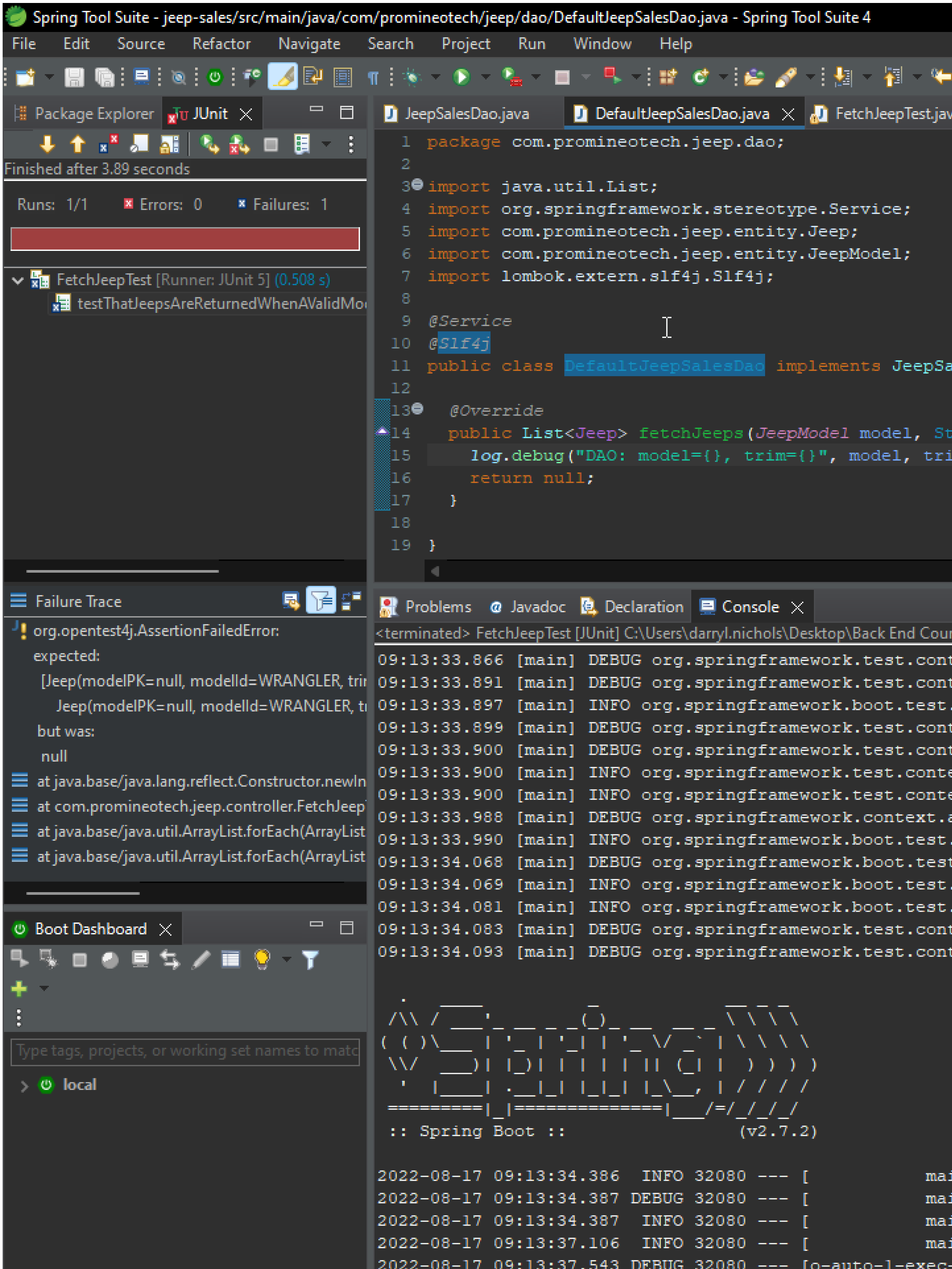
```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```

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.


Spring Tool Suite - jeep-sales/src/main/java/com/promineotech/jeep/dao/DefaultJeepSalesDao.java - Spring Tool Suite 4

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

import java.util.List;
import org.springframework.stereotype.Service;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import lombok.extern.slf4j.Slf4j;

@Service
@Slf4j
public class DefaultJeepSalesDao implements JeepSa

  @Override
  public List<Jeep> fetchJeeps(JeepModel model, St
    log.debug("DAO: model={}, trim={}", model, tri
    return null;
  }

}
```

Finished after 3.89 seconds

Runs: 1/1   Errors: 0   Failures: 1

FetchJeepTest [Runner: JUnit 5] (0.508 s)
testThatJeepsAreReturnedWhenAValidMo

```
org.opentest4j.AssertionFailedError:
expected:
 [Jeep(modelPK=null, modelId=WRANGLER, tri
   Jeep(modelPK=null, modelId=WRANGLER, t
 but was:
  null
at java.base/java.lang.reflect.Constructor.newIn
at com.promineotech.jeep.controller.FetchJeep
at java.base/java.util.ArrayList.forEach(ArrayList
at java.base/java.util.ArrayList.forEach(ArrayList
```

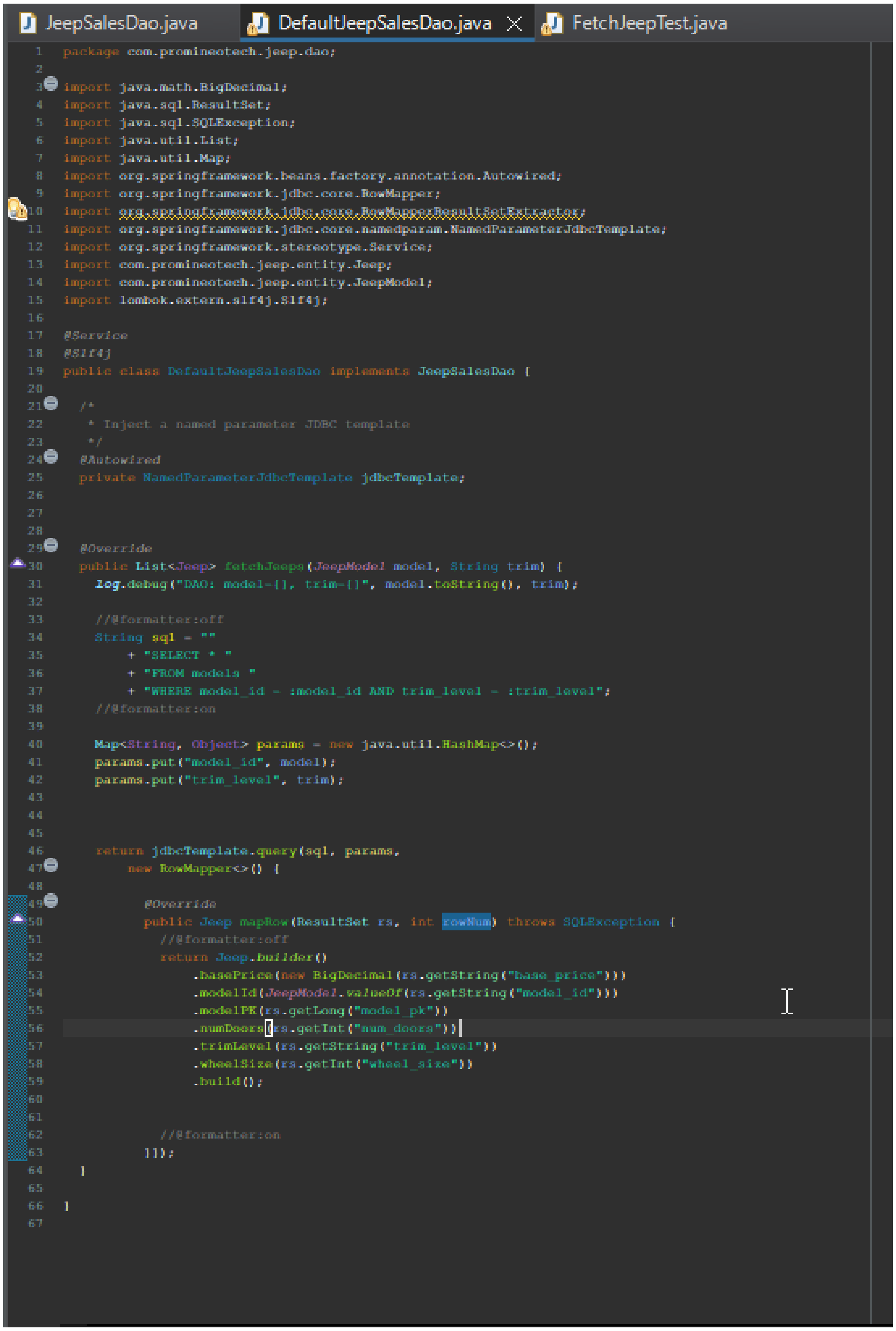
```
<terminated> FetchJeepTest [JUnit] C:\Users\darryl.nichols\Desktop\Back End Cour
09:13:33.866 [main] DEBUG org.springframework.test.cont
09:13:33.891 [main] DEBUG org.springframework.test.cont
09:13:33.897 [main] INFO org.springframework.boot.test.
09:13:33.899 [main] DEBUG org.springframework.test.cont
09:13:33.900 [main] DEBUG org.springframework.test.cont
09:13:33.900 [main] INFO org.springframework.test.conte
09:13:33.900 [main] INFO org.springframework.test.conte
09:13:33.988 [main] DEBUG org.springframework.context.a
09:13:33.990 [main] INFO org.springframework.boot.test.
09:13:34.068 [main] DEBUG org.springframework.boot.test
09:13:34.069 [main] INFO org.springframework.boot.test.
09:13:34.081 [main] INFO org.springframework.boot.test.
09:13:34.083 [main] DEBUG org.springframework.test.cont
09:13:34.093 [main] DEBUG org.springframework.test.cont


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

2022-08-17 09:13:34.386  INFO 32080 --- [           mai
2022-08-17 09:13:34.387 DEBUG 32080 --- [           mai
2022-08-17 09:13:34.387  INFO 32080 --- [           mai
2022-08-17 09:13:37.106  INFO 32080 --- [           mai
2022-08-17 09:13:37.543 DEBUG 32080 --- [o-auto-1-exec
```

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());`)

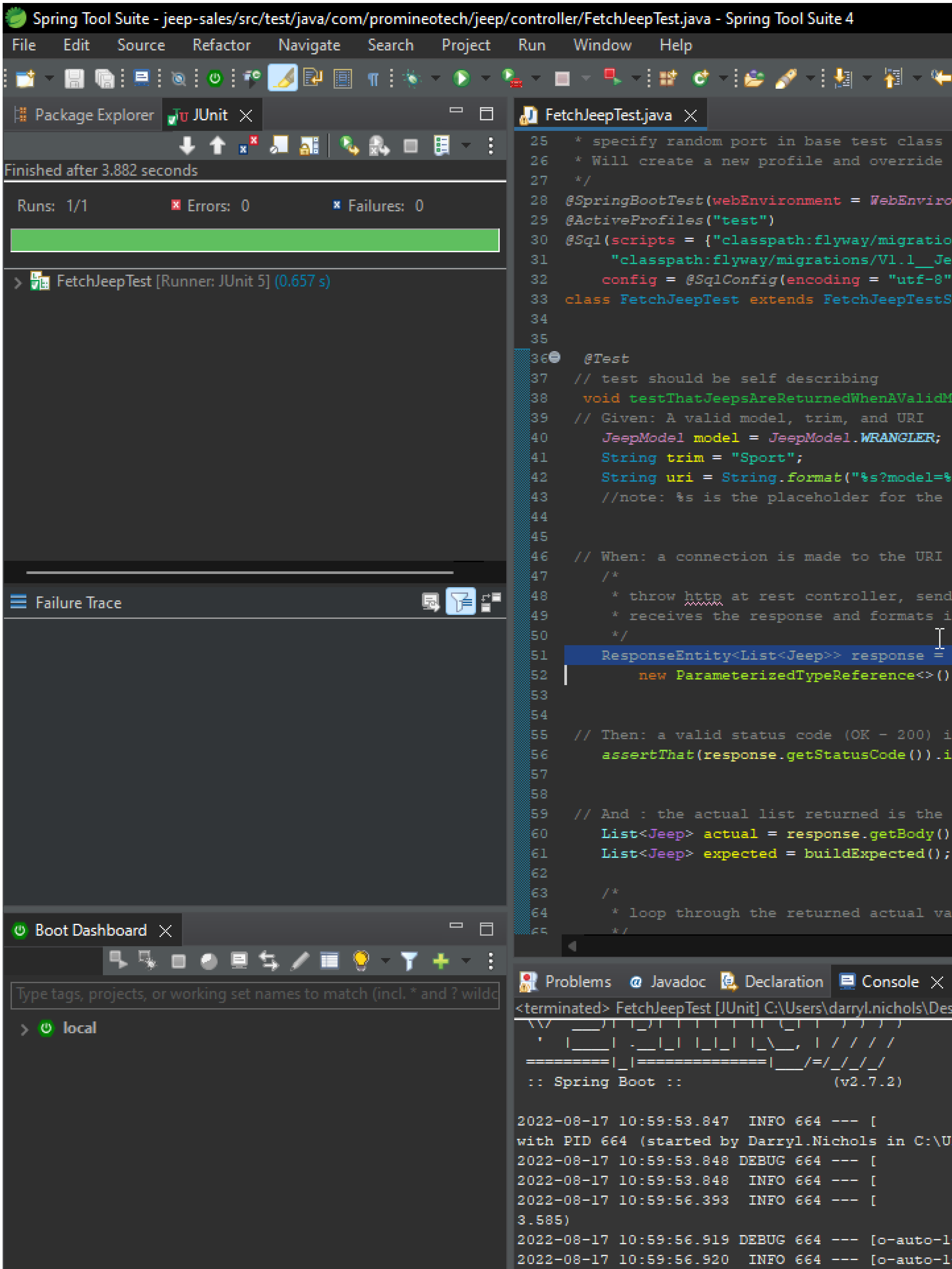
f) Call the `query` method on the `NamedParameterJdbcTemplate` instance variable to return a list of Jeep model objects. Use a RowMapper to map each row of the result set. Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a screenshot to show the complete method in the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



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

import java.math.BigDecimal;
import java.sql.ResultSet;
import java.sql.SQLException;
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.RowMapperResultSetExtractor;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Service;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import lombok.extern.slf4j.Slf4j;

@Service
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {

  /*
   * Inject a named parameter JDBC template
   */
  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;



  @Override
  public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    log.debug("DAO: model={}, trim={}", model.toString(), 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 java.util.HashMap<>();
    params.put("model_id", model);
    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
          }});
  }

}
```

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.


Spring Tool Suite - jeep-sales/src/test/java/com/promineotech/jeep/controller/FetchJeepTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
FetchJeepTest.java
Failure Trace
Boot Dashboard
Type tags, projects, or working set names to match (incl. * and ? wildc
local
Problems
Javadoc
Declaration
Console

Finished after 3.882 seconds

Runs: 1/1   Errors: 0   Failures: 0

FetchJeepTest [Runner: JUnit 5] (0.657 s)

```java
 * specify random port in base test class
 * Will create a new profile and override
 */
@SpringBootTest(webEnvironment = WebEnviro
@ActiveProfiles("test")
@Sql(scripts = {"classpath:flyway/migratio
    "classpath:flyway/migrations/V1.1__Je
    config = @SqlConfig(encoding = "utf-8"
class FetchJeepTest extends FetchJeepTestS


  @Test
 // test should be self describing
  void testThatJeepsAreReturnedWhenAValidM
 // Given: A valid model, trim, and URI
    JeepModel model = JeepModel.WRANGLER;
    String trim = "Sport";
    String uri = String.format("%s?model=%
    //note: %s is the placeholder for the


 // When: a connection is made to the URI
    /*
     * throw http at rest controller, send
     * receives the response and formats i
     */
    ResponseEntity<List<Jeep>> response =
        new ParameterizedTypeReference<>()


 // Then: a valid status code (OK - 200) i
    assertThat(response.getStatusCode()).i


 // And : the actual list returned is the
    List<Jeep> actual = response.getBody()
    List<Jeep> expected = buildExpected();

    /*
     * loop through the returned actual va
     */
```

```
<terminated> FetchJeepTest [JUnit] C:\Users\darryl.nichols\Des
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

2022-08-17 10:59:53.847  INFO 664 --- [
with PID 664 (started by Darryl.Nichols in C:\U
2022-08-17 10:59:53.848 DEBUG 664 --- [
2022-08-17 10:59:53.848  INFO 664 --- [
2022-08-17 10:59:56.393  INFO 664 --- [
3.585)
2022-08-17 10:59:56.919 DEBUG 664 --- [o-auto-1
2022-08-17 10:59:56.920  INFO 664 --- [o-auto-1
```

Screenshots of Code:

Found where 🖥 was noted

Screenshots of Running Application:

Found where 🖥 was noted

URL to GitHub Repository:

https://github.com/dnich02f/SpringBoot-Week15-CodingAssignment