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:

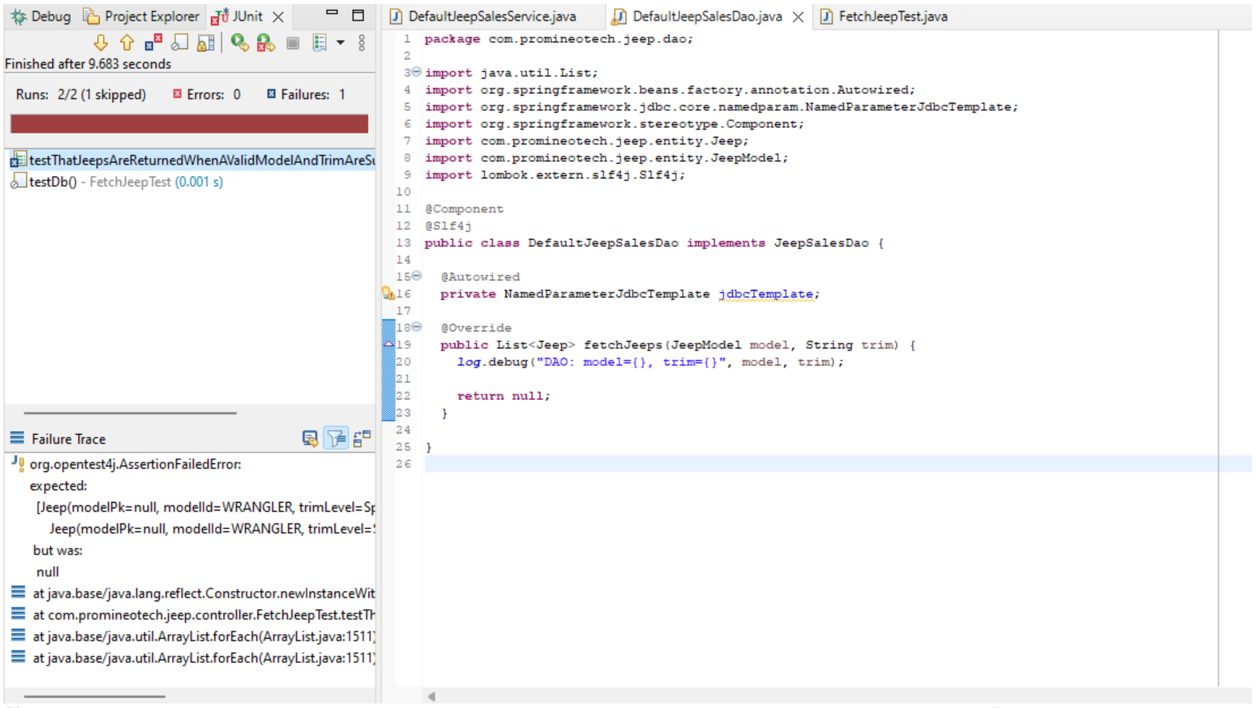
```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```

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`.

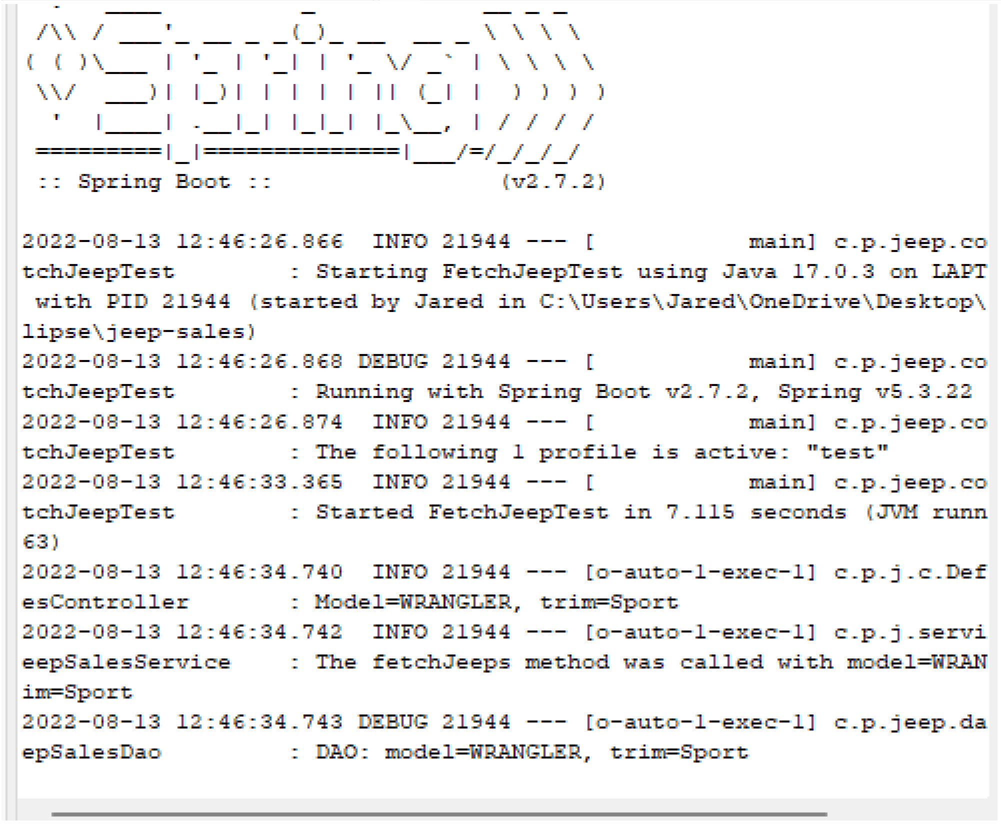
   b) Add a log statement in `DefaultJeepSalesDao.fetchJeeps()` that logs the model and trim level. Run the integration test. Produce a screenshot showing the DAO implementation class and the log line in the IDE's console.



Debug | Project Explorer | JUnit | DefaultJeepSalesService.java | DefaultJeepSalesDao.java | FetchJeepTest.java

```
Finished after 9.683 seconds

Runs:  2/2 (1 skipped)     Errors:  0     Failures:  1

testThatJeepsAreReturnedWhenAValidModelAndTrimAreSu
testDb() - FetchJeepTest (0.001 s)

Failure Trace
org.opentest4j.AssertionFailedError:
expected:
 [Jeep(modelPk=null, modelId=WRANGLER, trimLevel=Sp
   Jeep(modelPk=null, modelId=WRANGLER, trimLevel=
 but was:
 null
at java.base/java.lang.reflect.Constructor.newInstanceWit
at com.promineotech.jeep.controller.FetchJeepTest.testTh
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

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

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
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.debug("DAO: model={}, trim={}", model, trim);

    return null;
  }

}
```



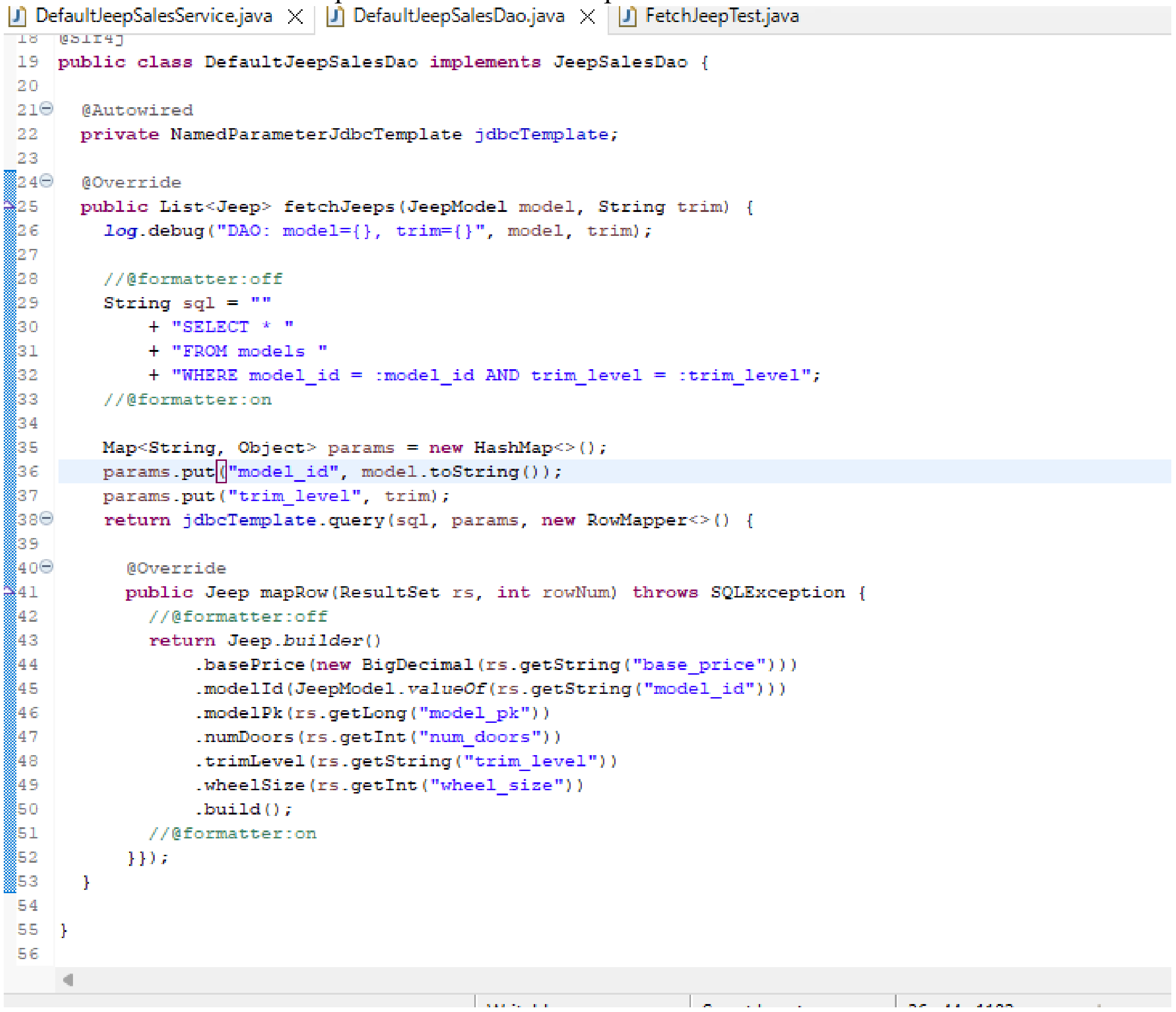
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

2022-08-13 12:46:26.866  INFO 21944 --- [           main] c.p.jeep.co
tchJeepTest           : Starting FetchJeepTest using Java 17.0.3 on LAPT
 with PID 21944 (started by Jared in C:\Users\Jared\OneDrive\Desktop\
lipse\jeep-sales)
2022-08-13 12:46:26.868 DEBUG 21944 --- [           main] c.p.jeep.co
tchJeepTest           : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-13 12:46:26.874  INFO 21944 --- [           main] c.p.jeep.co
tchJeepTest           : The following 1 profile is active: "test"
2022-08-13 12:46:33.365  INFO 21944 --- [           main] c.p.jeep.co
tchJeepTest           : Started FetchJeepTest in 7.115 seconds (JVM runn
63)
2022-08-13 12:46:34.740  INFO 21944 --- [o-auto-1-exec-1] c.p.j.c.Def
esController          : Model=WRANGLER, trim=Sport
2022-08-13 12:46:34.742  INFO 21944 --- [o-auto-1-exec-1] c.p.j.servi
eepSalesService       : The fetchJeeps method was called with model=WRAN
im=Sport
2022-08-13 12:46:34.743 DEBUG 21944 --- [o-auto-1-exec-1] c.p.jeep.da
epSalesDao            : DAO: model=WRANGLER, trim=Sport
```

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());`)

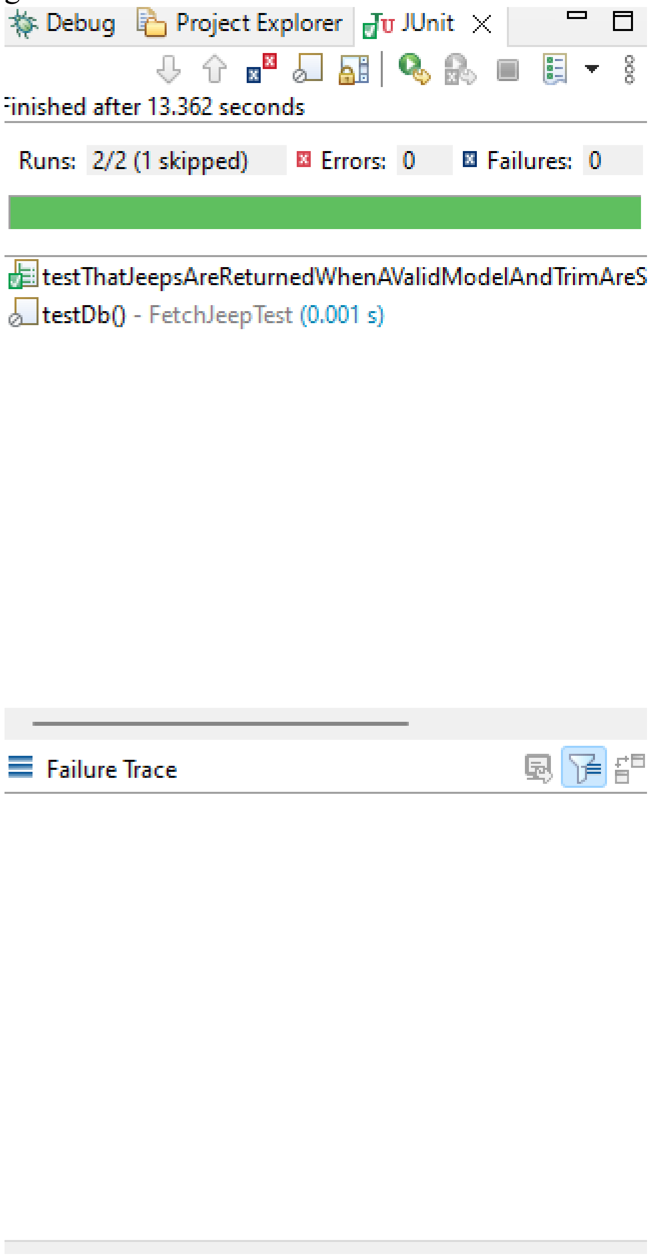
f) Call the `query` method on the `NamedParameterJdbcTemplate` instance variable to return a list of Jeep model objects. Use a RowMapper to map each row of the result set. Remember to convert `modelId` to a `JeepModel`. See the video for details. Produce a screenshot to show the complete method in the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



```
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {

  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;

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

}
```

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.

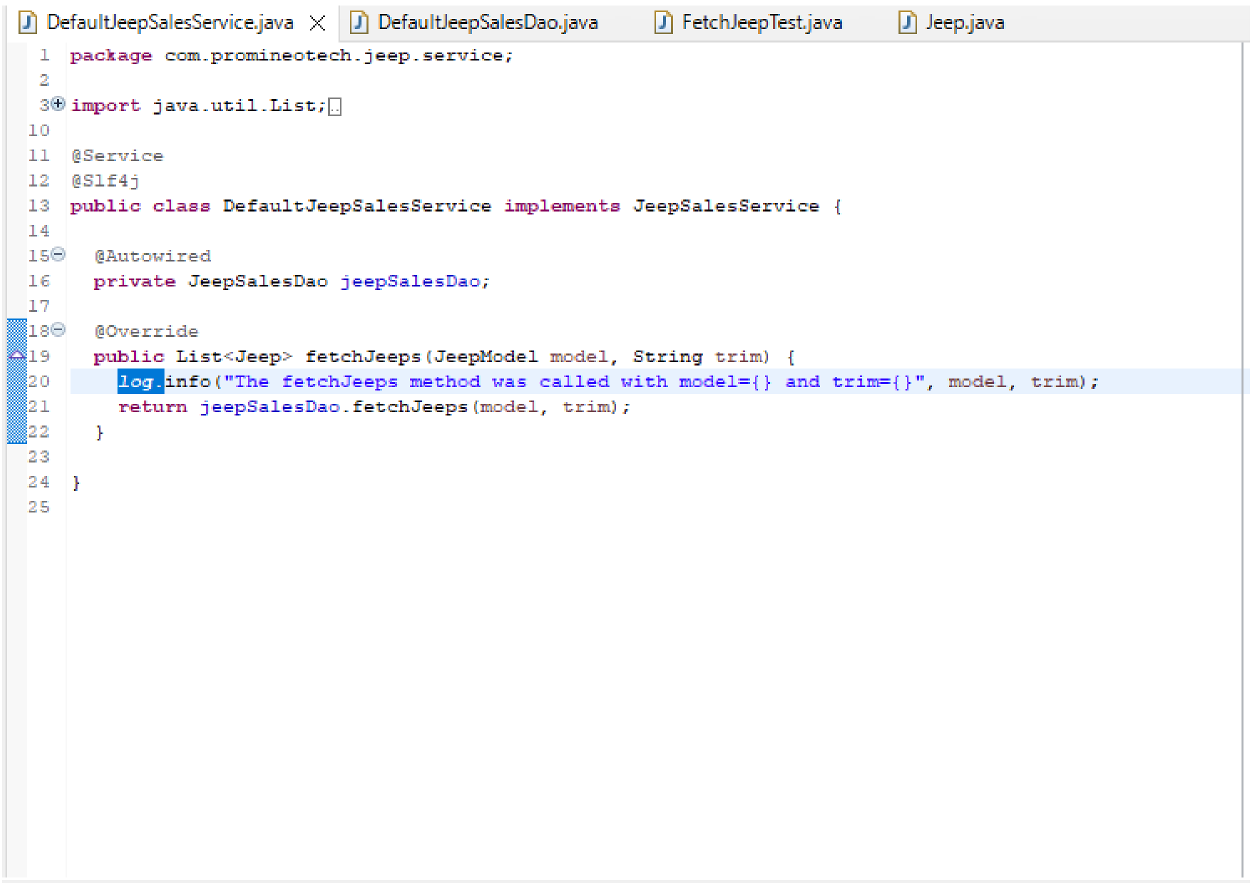


Finished after 13.362 seconds

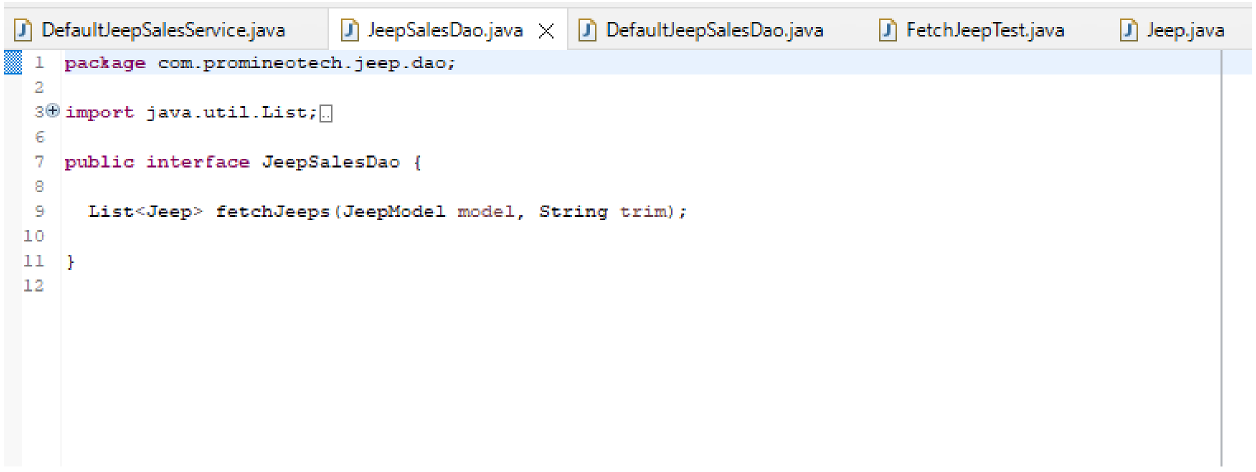
Runs: 2/2 (1 skipped) Errors: 0 Failures: 0

testThatJeepsAreReturnedWhenAValidModelAndTrimAreS
testDb() - FetchJeepTest (0.001 s)

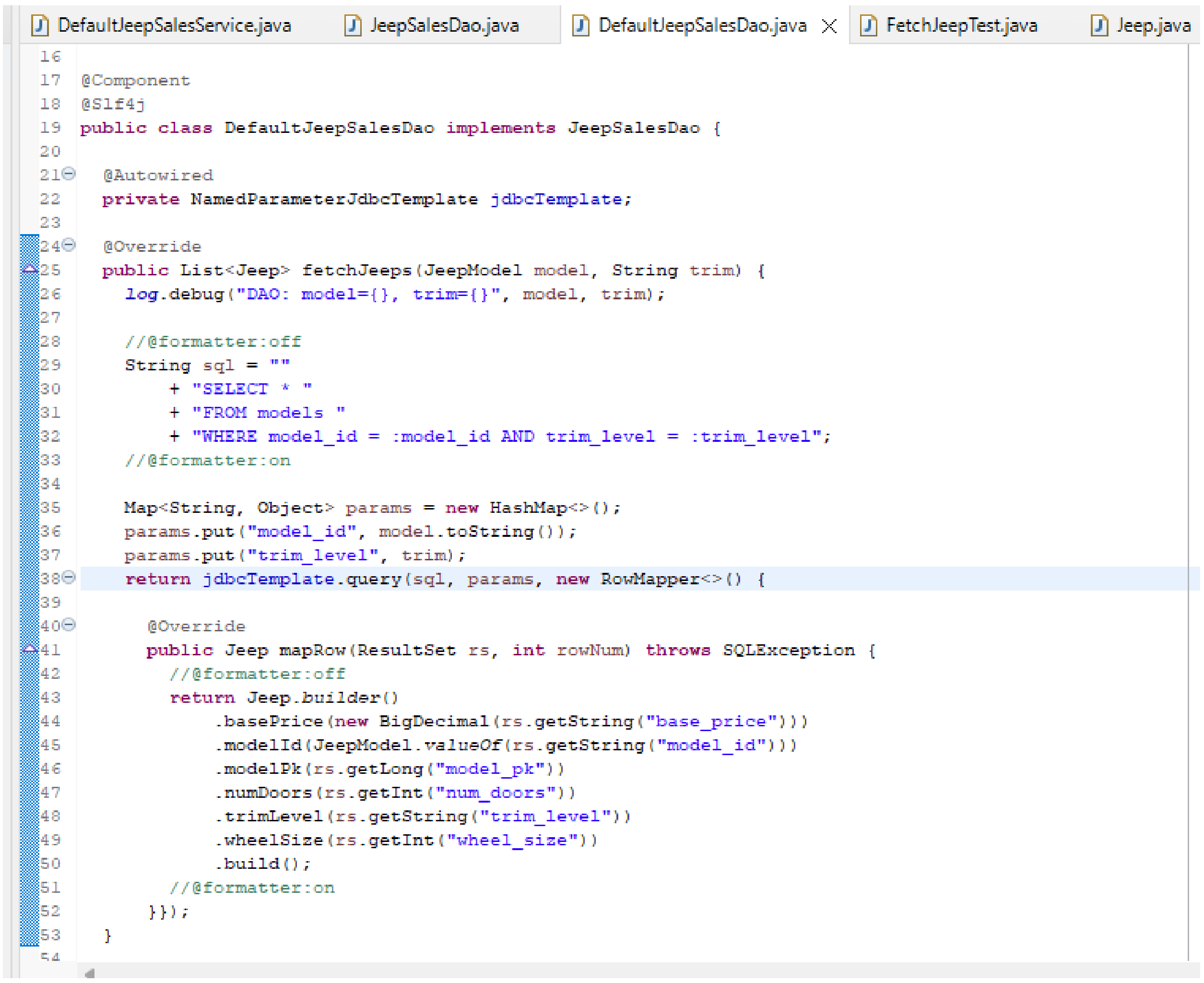
Screenshots of Code:


DefaultJeepSalesService.java | DefaultJeepSalesDao.java | FetchJeepTest.java | Jeep.java

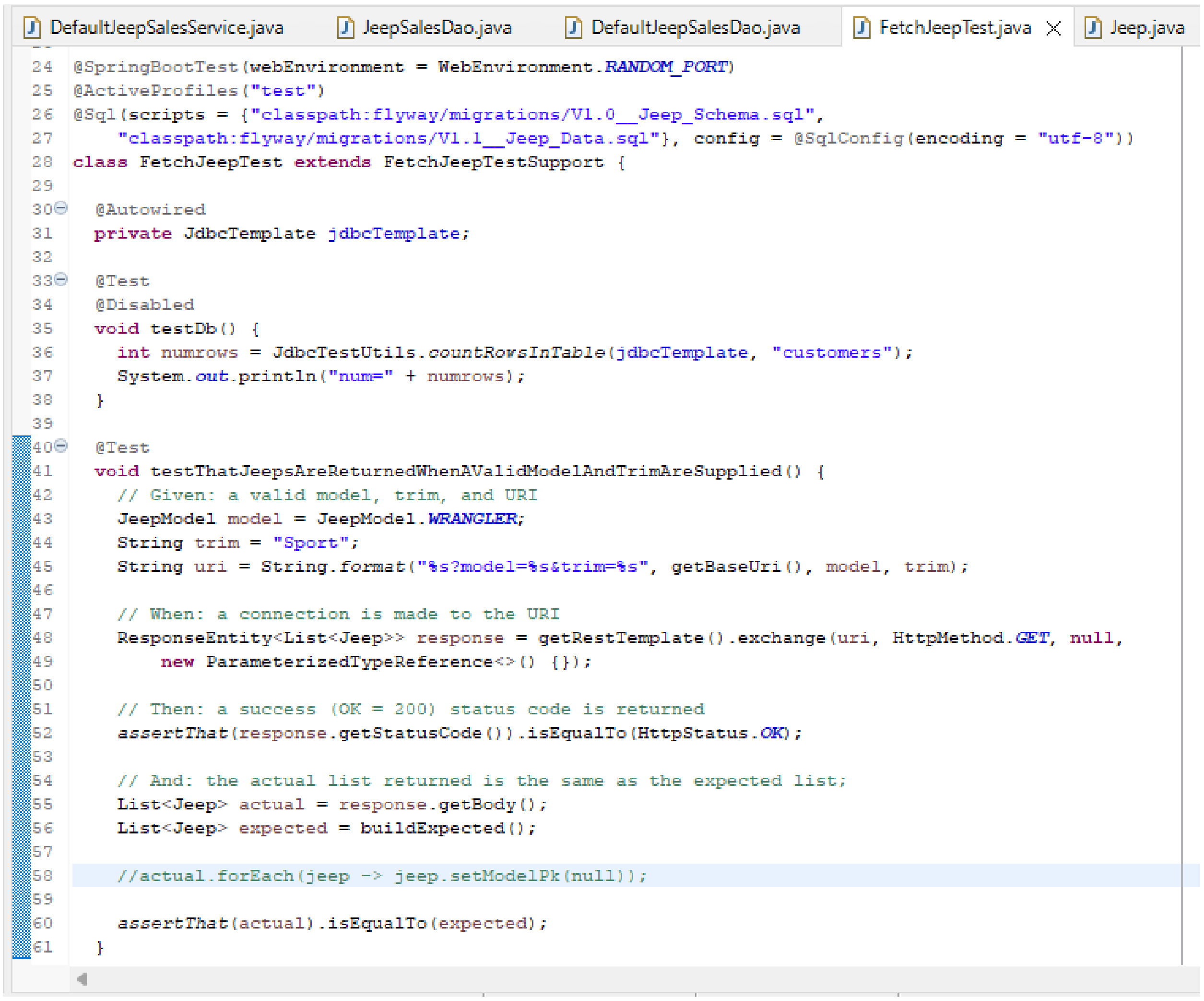
```java
package com.promineotech.jeep.service;

import java.util.List;

@Service
@Slf4j
public class DefaultJeepSalesService implements JeepSalesService {

  @Autowired
  private JeepSalesDao jeepSalesDao;

  @Override
  public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    log.info("The fetchJeeps method was called with model={} and trim={}", model, trim);
    return jeepSalesDao.fetchJeeps(model, trim);
  }

}
```


DefaultJeepSalesService.java | JeepSalesDao.java | DefaultJeepSalesDao.java | FetchJeepTest.java | Jeep.java

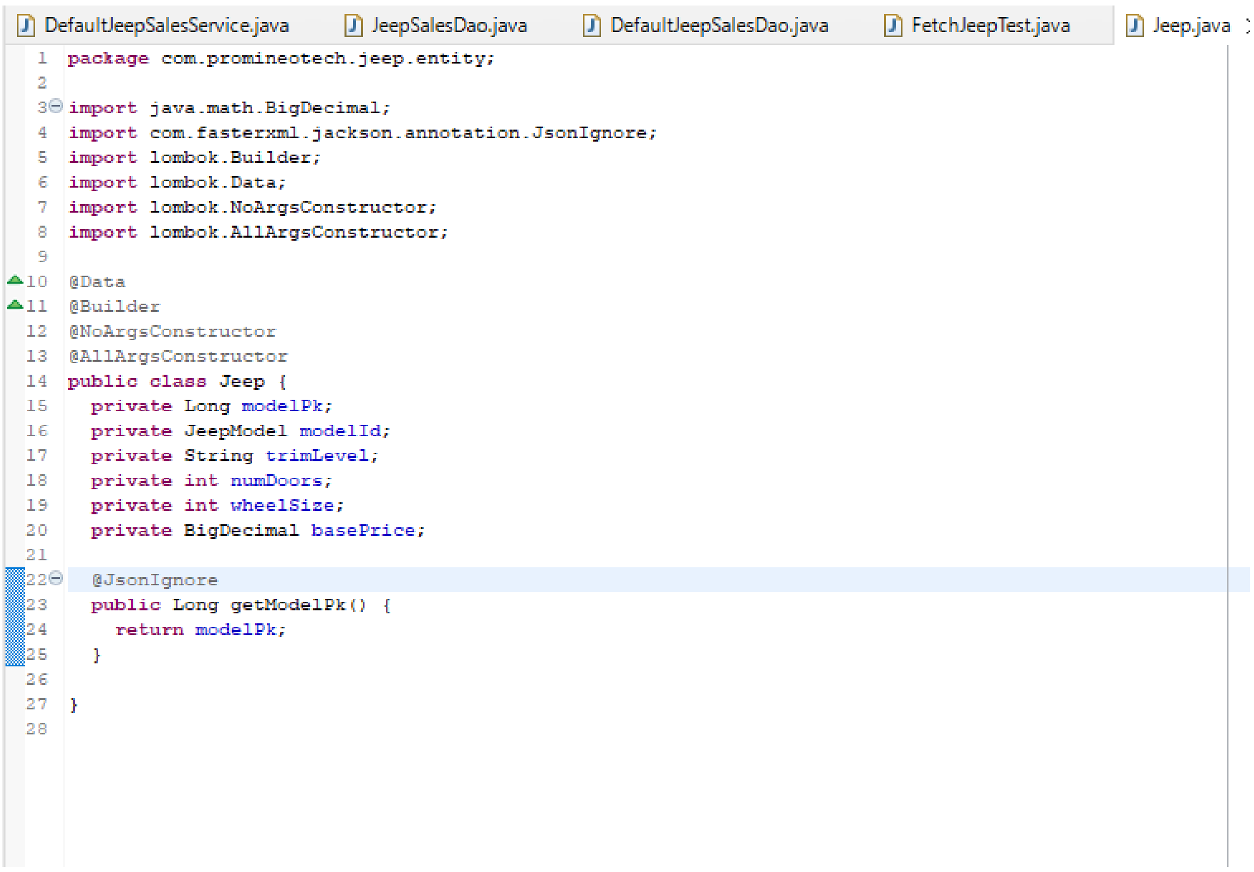
```java
package com.promineotech.jeep.dao;

import java.util.List;

public interface JeepSalesDao {

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

}
```



```java

@Component
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {

  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;

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

```



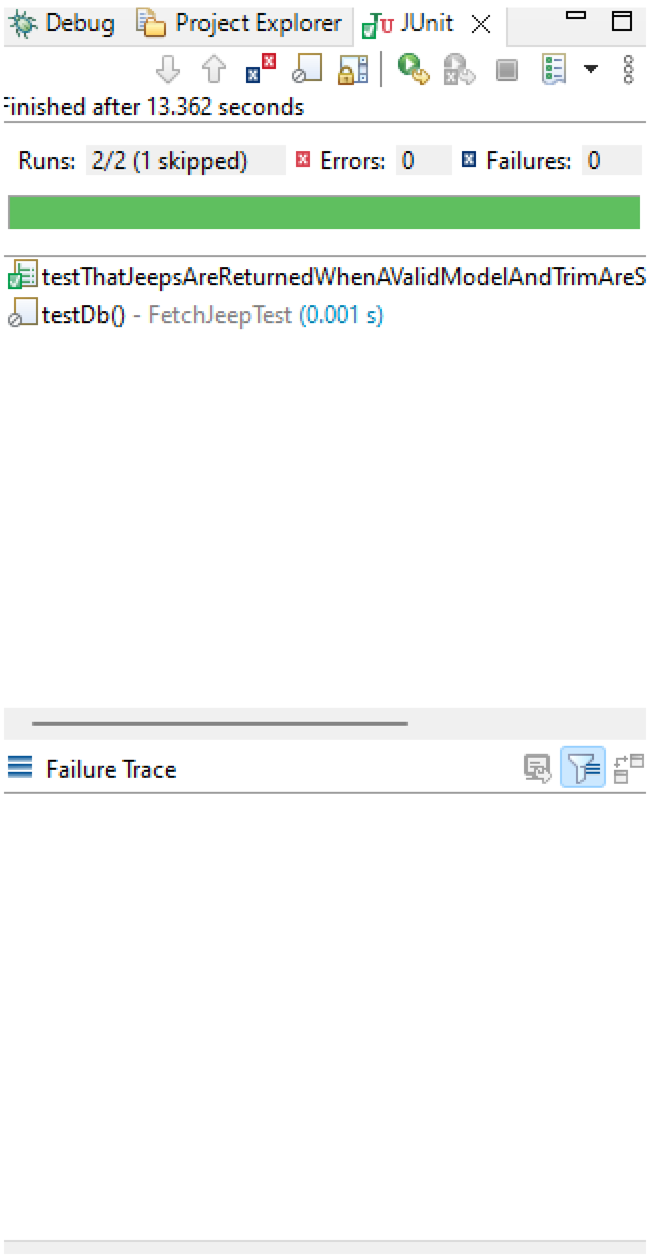
```java
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@Sql(scripts = {"classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
    "classpath:flyway/migrations/V1.1__Jeep_Data.sql"}, config = @SqlConfig(encoding = "utf-8"))
class FetchJeepTest extends FetchJeepTestSupport {

  @Autowired
  private JdbcTemplate jdbcTemplate;

  @Test
  @Disabled
  void testDb() {
    int numrows = JdbcTestUtils.countRowsInTable(jdbcTemplate, "customers");
    System.out.println("num=" + numrows);
  }

  @Test
  void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
    // Given: a valid model, trim, and URI
    JeepModel model = JeepModel.WRANGLER;
    String trim = "Sport";
    String uri = String.format("%s?model=%s&trim=%s", getBaseUri(), model, trim);

    // When: a connection is made to the URI
    ResponseEntity<List<Jeep>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
        new ParameterizedTypeReference<>() {});

    // Then: a success (OK = 200) status code is returned
    assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);

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

    //actual.forEach(jeep -> jeep.setModelPk(null));

    assertThat(actual).isEqualTo(expected);
  }
```



DefaultJeepSalesService.java | JeepSalesDao.java | DefaultJeepSalesDao.java | FetchJeepTest.java | Jeep.java

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

import java.math.BigDecimal;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Jeep {
  private Long modelPk;
  private JeepModel modelId;
  private String trimLevel;
  private int numDoors;
  private int wheelSize;
  private BigDecimal basePrice;

  @JsonIgnore
  public Long getModelPk() {
    return modelPk;
  }

}
```

Screenshots of Running Application:



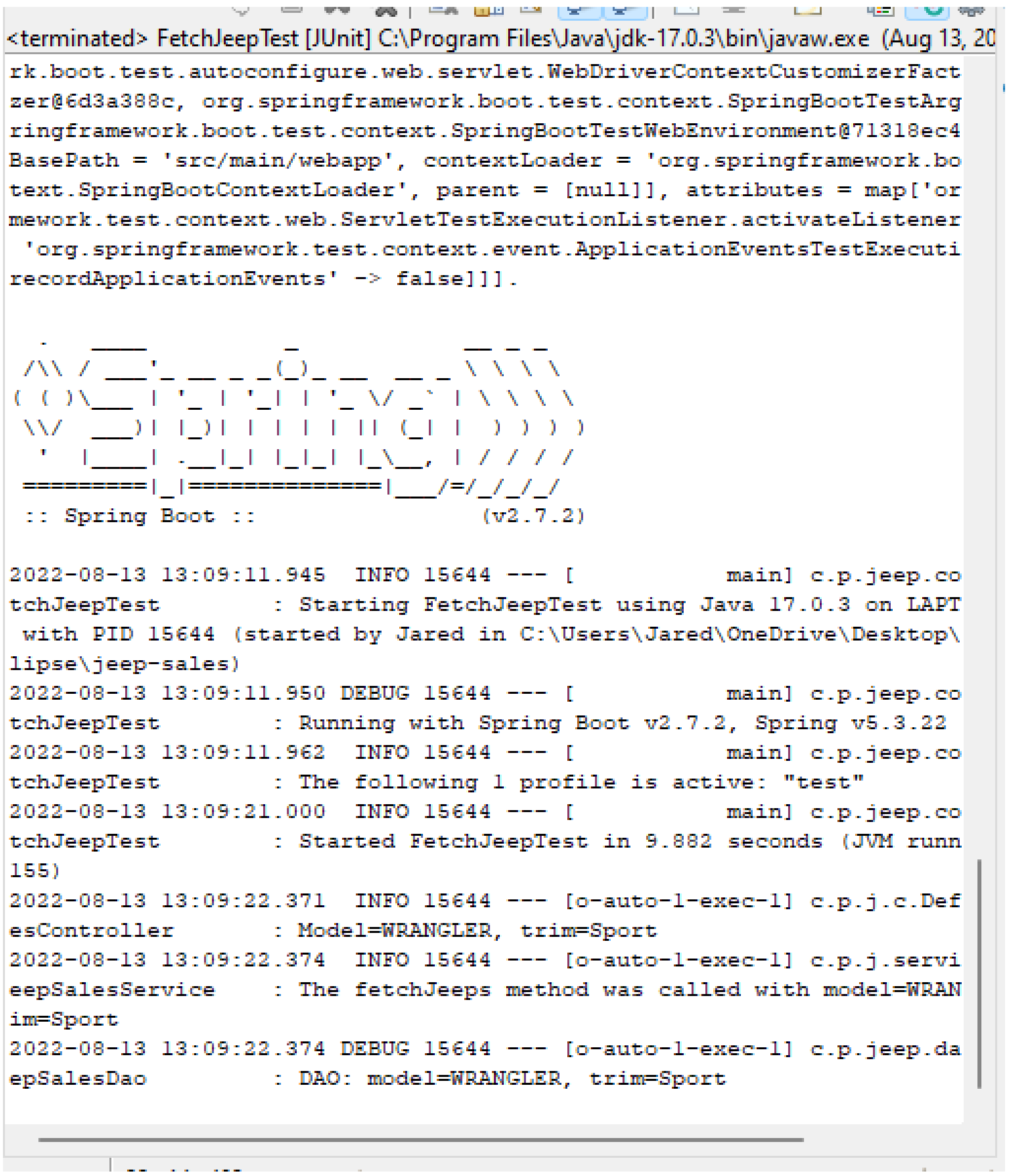
Debug
Project Explorer
JUnit

Finished after 13.362 seconds

Runs: 2/2 (1 skipped) Errors: 0 Failures: 0

testThatJeepsAreReturnedWhenAValidModelAndTrimAreS
testDb() - FetchJeepTest (0.001 s)

Failure Trace


```
<terminated> FetchJeepTest [JUnit] C:\Program Files\Java\jdk-17.0.3\bin\javaw.exe (Aug 13, 20
rk.boot.test.autoconfigure.web.servlet.WebDriverContextCustomizerFact
zer@6d3a388c, org.springframework.boot.test.context.SpringBootTestArg
ringframework.boot.test.context.SpringBootTestWebEnvironment@71318ec4
BasePath = 'src/main/webapp', contextLoader = 'org.springframework.bo
text.SpringBootContextLoader', parent = [null]], attributes = map['or
mework.test.context.web.ServletTestExecutionListener.activateListener
 'org.springframework.test.context.event.ApplicationEventsTestExecuti
recordApplicationEvents' -> false]]].

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

2022-08-13 13:09:11.945  INFO 15644 --- [           main] c.p.jeep.co
tchJeepTest          : Starting FetchJeepTest using Java 17.0.3 on LAPT
 with PID 15644 (started by Jared in C:\Users\Jared\OneDrive\Desktop\
lipse\jeep-sales)
2022-08-13 13:09:11.950 DEBUG 15644 --- [           main] c.p.jeep.co
tchJeepTest          : Running with Spring Boot v2.7.2, Spring v5.3.22
2022-08-13 13:09:11.962  INFO 15644 --- [           main] c.p.jeep.co
tchJeepTest          : The following 1 profile is active: "test"
2022-08-13 13:09:21.000  INFO 15644 --- [           main] c.p.jeep.co
tchJeepTest          : Started FetchJeepTest in 9.882 seconds (JVM runn
155)
2022-08-13 13:09:22.371  INFO 15644 --- [o-auto-1-exec-1] c.p.j.c.Def
esController         : Model=WRANGLER, trim=Sport
2022-08-13 13:09:22.374  INFO 15644 --- [o-auto-1-exec-1] c.p.j.servi
eepSalesService      : The fetchJeeps method was called with model=WRAN
im=Sport
2022-08-13 13:09:22.374 DEBUG 15644 --- [o-auto-1-exec-1] c.p.jeep.da
epSalesDao           : DAO: model=WRANGLER, trim=Sport
```

URL to GitHub Repository:

https://github.com/JaredBears/JeepSales