Web API Design with Spring Boot Week 3 Coding Assignment

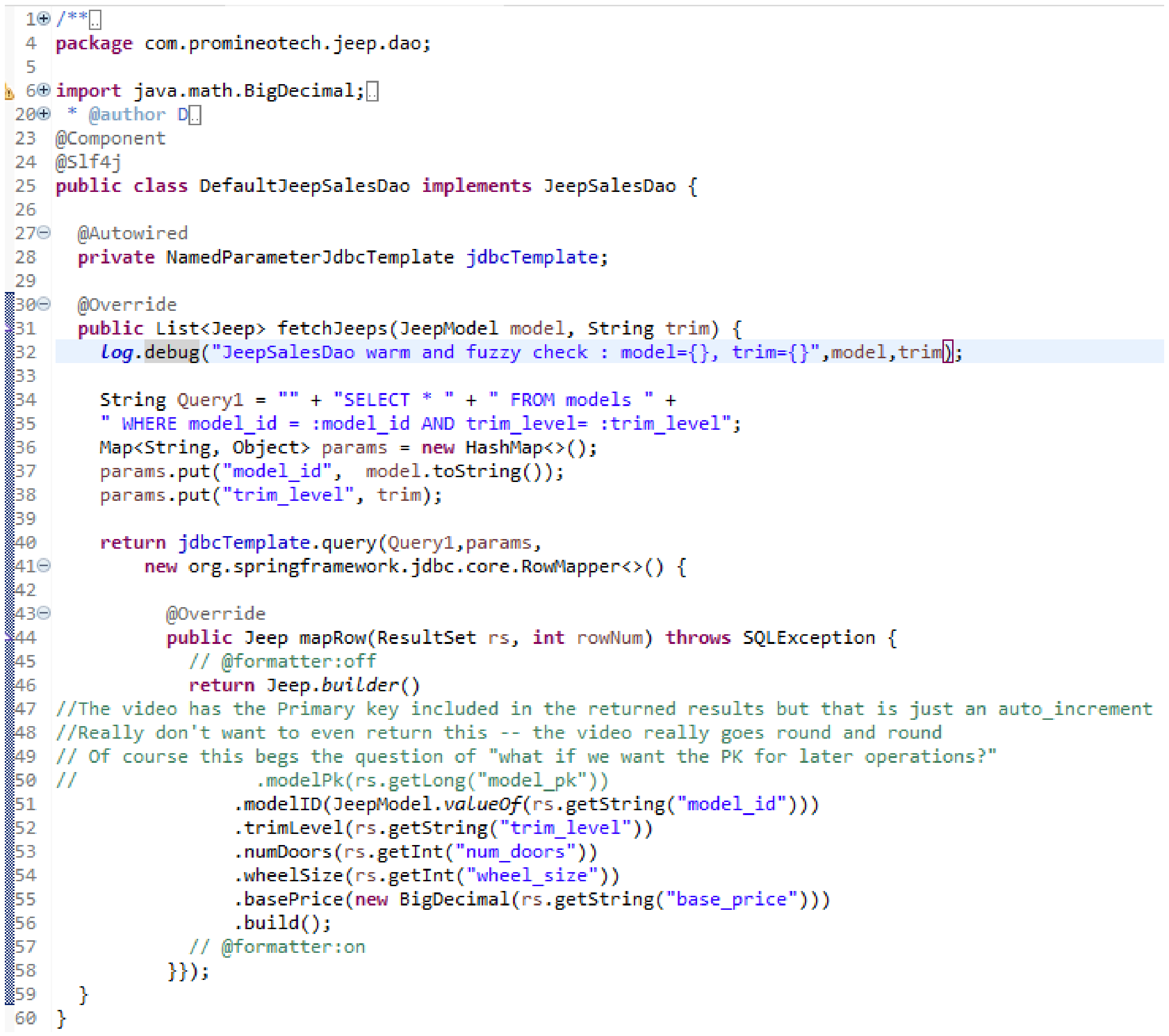
1)

2)

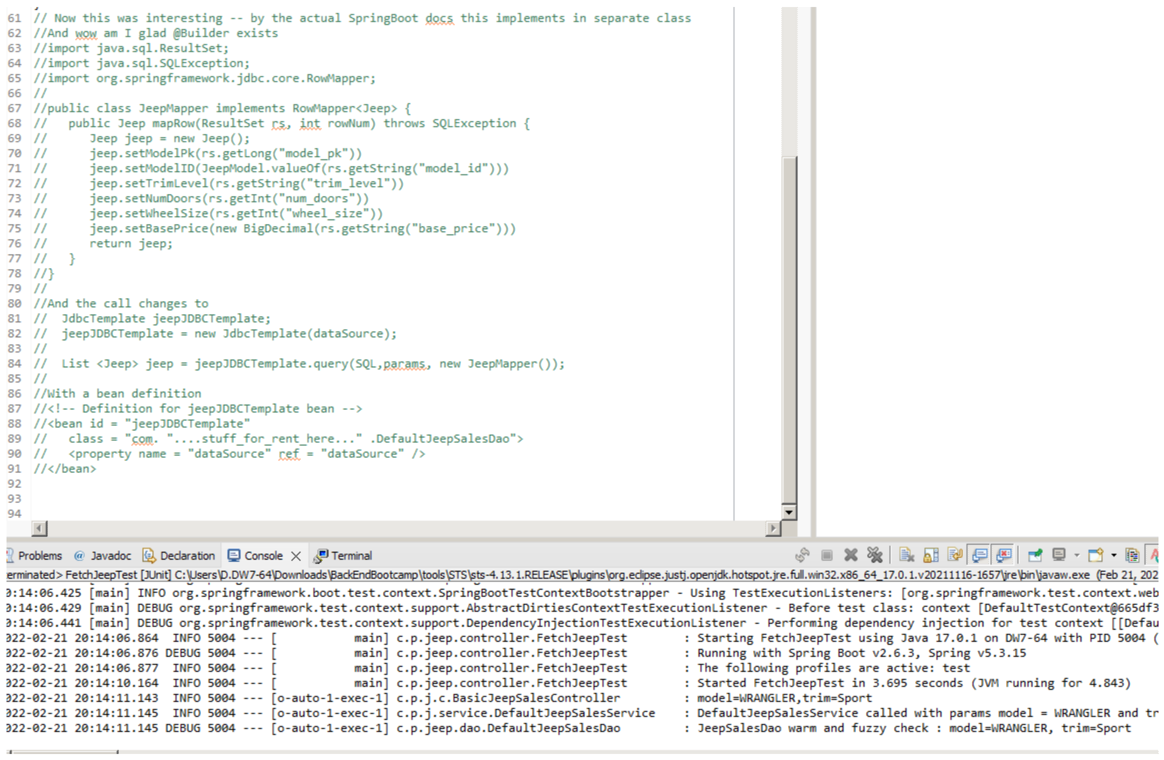
3) In the DAO implementation class (`DefaultJeepSalesDao`):

   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.



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

import java.math.BigDecimal;
 * @author D
@Component
@Slf4j
public class DefaultJeepSalesDao implements JeepSalesDao {

  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;

  @Override
  public List<Jeep> fetchJeeps(JeepModel model, String trim) {
    log.debug("JeepSalesDao warm and fuzzy check : model={}, trim={}",model,trim);

    String Query1 = "" + "SELECT * " + " FROM models " +
    " WHERE model_id = :model_id AND trim_level= :trim_level";
    Map<String, Object> params = new HashMap<>();
    params.put("model_id",  model.toString());
    params.put("trim_level", trim);

    return jdbcTemplate.query(Query1,params,
        new org.springframework.jdbc.core.RowMapper<>() {

          @Override
          public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
            // @formatter:off
            return Jeep.builder()
//The video has the Primary key included in the returned results but that is just an auto_increment
//Really don't want to even return this -- the video really goes round and round
// Of course this begs the question of "what if we want the PK for later operations?"
//                .modelPk(rs.getLong("model_pk"))
                .modelID(JeepModel.valueOf(rs.getString("model_id")))
                .trimLevel(rs.getString("trim_level"))
                .numDoors(rs.getInt("num_doors"))
                .wheelSize(rs.getInt("wheel_size"))
                .basePrice(new BigDecimal(rs.getString("base_price")))
                .build();
            // @formatter:on
          }});
  }
}
```



```
// Now this was interesting -- by the actual SpringBoot docs this implements in separate class
//And wow am I glad @Builder exists
//import java.sql.ResultSet;
//import java.sql.SQLException;
//import org.springframework.jdbc.core.RowMapper;
//
//public class JeepMapper implements RowMapper<Jeep> {
//   public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
//      Jeep jeep = new Jeep();
//      jeep.setModelPk(rs.getLong("model_pk"))
//      jeep.setModelID(JeepModel.valueOf(rs.getString("model_id")))
//      jeep.setTrimLevel(rs.getString("trim_level"))
//      jeep.setNumDoors(rs.getInt("num_doors"))
//      jeep.setWheelSize(rs.getInt("wheel_size"))
//      jeep.setBasePrice(new BigDecimal(rs.getString("base_price")))
//      return jeep;
//   }
//}
//
//And the call changes to
//  JdbcTemplate jeepJDBCTemplate;
//  jeepJDBCTemplate = new JdbcTemplate(dataSource);
//
//  List <Jeep> jeep = jeepJDBCTemplate.query(SQL,params, new JeepMapper());
//
//With a bean definition
//<!-- Definition for jeepJDBCTemplate bean -->
//<bean id = "jeepJDBCTemplate"
//   class = "com. "....stuff_for_rent_here..." .DefaultJeepSalesDao">
//   <property name = "dataSource" ref = "dataSource" />
//</bean>
```

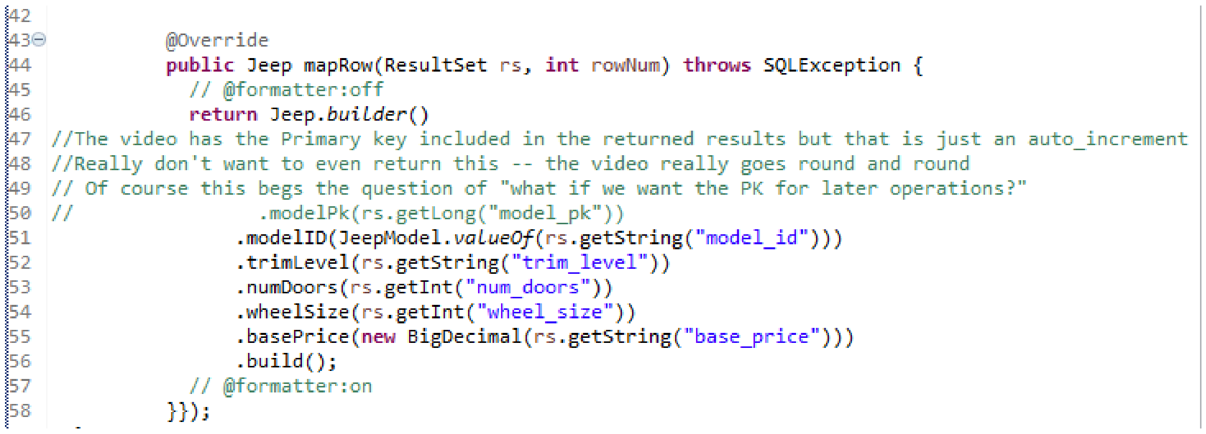
```
erminated> FetchJeepTest [JUnit] C:\Users\D.DW7-64\Downloads\BackEndBootcamp\tools\STS\sts-4.13.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe  (Feb 21, 202
0:14:06.425 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web
0:14:06.429 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@665df3
0:14:06.441 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[Defau
022-02-21 20:14:06.864  INFO 5004 --- [           main] c.p.jeep.controller.FetchJeepTest        : Starting FetchJeepTest using Java 17.0.1 on DW7-64 with PID 5004 (
022-02-21 20:14:06.876 DEBUG 5004 --- [           main] c.p.jeep.controller.FetchJeepTest        : Running with Spring Boot v2.6.3, Spring v5.3.15
022-02-21 20:14:06.877  INFO 5004 --- [           main] c.p.jeep.controller.FetchJeepTest        : The following profiles are active: test
022-02-21 20:14:10.164  INFO 5004 --- [           main] c.p.jeep.controller.FetchJeepTest        : Started FetchJeepTest in 3.695 seconds (JVM running for 4.843)
022-02-21 20:14:11.143  INFO 5004 --- [o-auto-1-exec-1] c.p.j.c.BasicJeepSalesController         : model=WRANGLER,trim=Sport
022-02-21 20:14:11.145  INFO 5004 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService    : DefaultJeepSalesService called with params model = WRANGLER and tr
022-02-21 20:14:11.145 DEBUG 5004 --- [o-auto-1-exec-1] c.p.jeep.dao.DefaultJeepSalesDao         : JeepSalesDao warm and fuzzy check : model=WRANGLER, trim=Sport
```

c)

d)

e) 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.

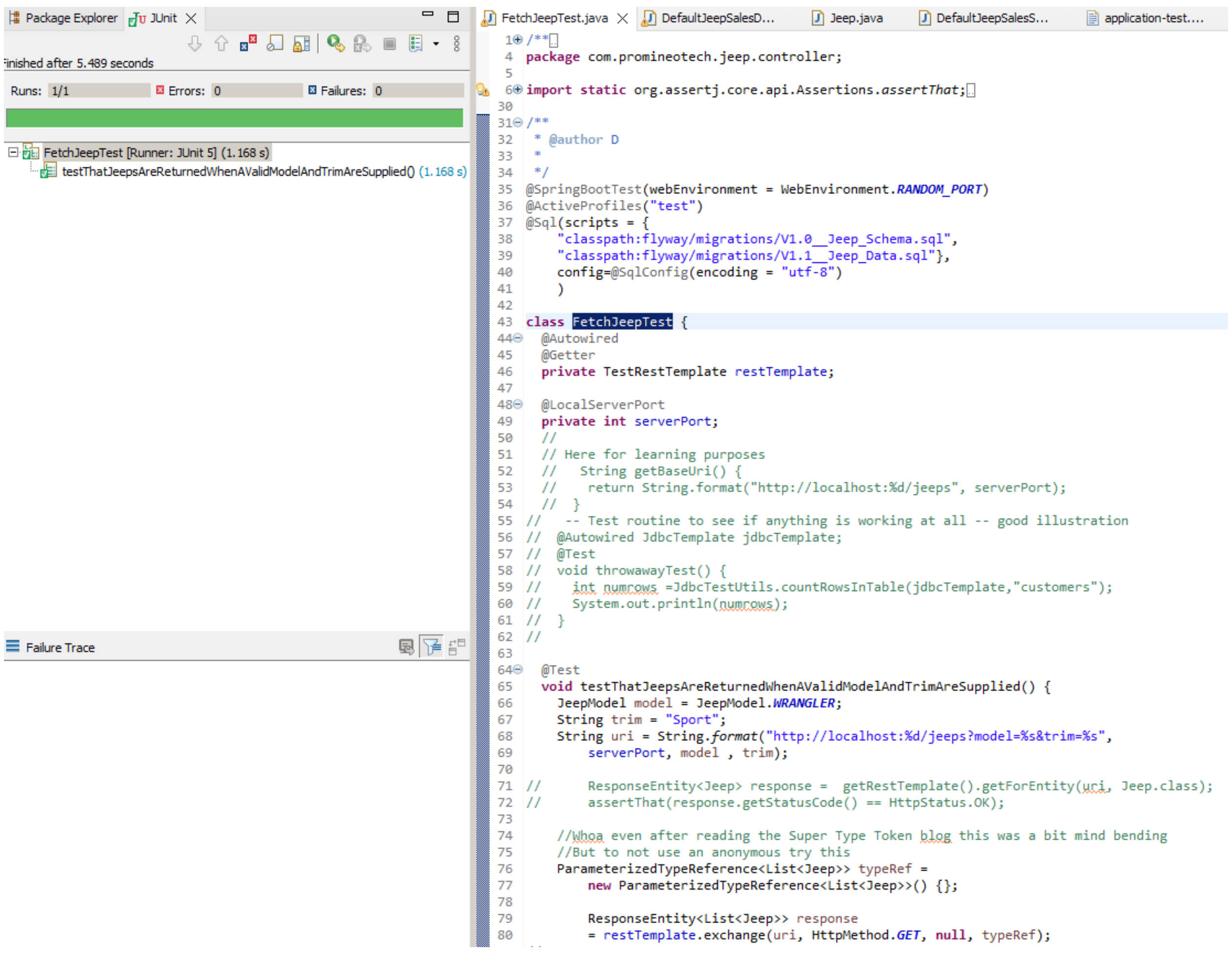
(Student Note : Complete class in screenshots above. Only method call here)



```
            @Override
            public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
              // @formatter:off
              return Jeep.builder()
//The video has the Primary key included in the returned results but that is just an auto_increment
//Really don't want to even return this -- the video really goes round and round
// Of course this begs the question of "what if we want the PK for later operations?"
//                     .modelPk(rs.getLong("model_pk"))
                     .modelID(JeepModel.valueOf(rs.getString("model_id")))
                     .trimLevel(rs.getString("trim_level"))
                     .numDoors(rs.getInt("num_doors"))
                     .wheelSize(rs.getInt("wheel_size"))
                     .basePrice(new BigDecimal(rs.getString("base_price")))
                     .build();
              // @formatter:on
            }});
```

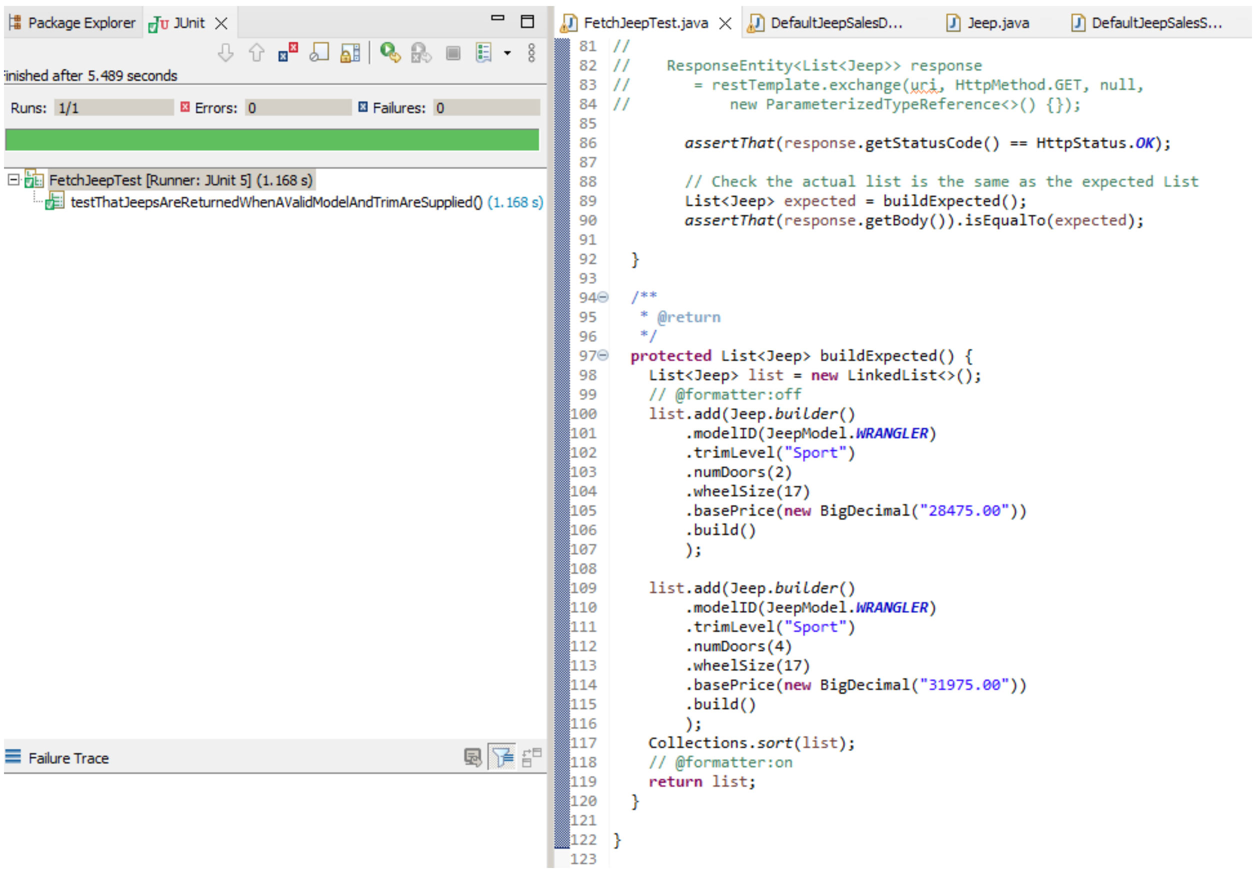
4)

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



```
Finished after 5.489 seconds
Runs: 1/1    Errors: 0    Failures: 0

FetchJeepTest [Runner: JUnit 5] (1.168 s)
  testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() (1.168 s)
```

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

import static org.assertj.core.api.Assertions.assertThat;

/**
 * @author D
 *
 */
@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 {
  @Autowired
  @Getter
  private TestRestTemplate restTemplate;

  @LocalServerPort
  private int serverPort;
  //
  // Here for learning purposes
  //   String getBaseUri() {
  //     return String.format("http://localhost:%d/jeeps", serverPort);
  //  }
//   -- Test routine to see if anything is working at all -- good illustration
//  @Autowired JdbcTemplate jdbcTemplate;
//  @Test
//  void throwawayTest() {
//    int numrows =JdbcTestUtils.countRowsInTable(jdbcTemplate,"customers");
//    System.out.println(numrows);
//  }
//

  @Test
  void testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() {
    JeepModel model = JeepModel.WRANGLER;
    String trim = "Sport";
    String uri = String.format("http://localhost:%d/jeeps?model=%s&trim=%s",
        serverPort, model , trim);

//      ResponseEntity<Jeep> response =  getRestTemplate().getForEntity(uri, Jeep.class);
//      assertThat(response.getStatusCode() == HttpStatus.OK);

    //Whoa even after reading the Super Type Token blog this was a bit mind bending
    //But to not use an anonymous try this
    ParameterizedTypeReference<List<Jeep>> typeRef =
        new ParameterizedTypeReference<List<Jeep>>() {};

        ResponseEntity<List<Jeep>> response
        = restTemplate.exchange(uri, HttpMethod.GET, null, typeRef);
```



```
inished after 5.489 seconds
Runs: 1/1    Errors: 0    Failures: 0

FetchJeepTest [Runner: JUnit 5] (1.168 s)
  testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() (1.168 s)
```

```java
//
//    ResponseEntity<List<Jeep>> response
//        = restTemplate.exchange(uri, HttpMethod.GET, null,
//            new ParameterizedTypeReference<>() {});

        assertThat(response.getStatusCode() == HttpStatus.OK);

        // Check the actual list is the same as the expected List
        List<Jeep> expected = buildExpected();
        assertThat(response.getBody()).isEqualTo(expected);

  }

  /**
   * @return
   */
  protected List<Jeep> buildExpected() {
    List<Jeep> list = new LinkedList<>();
    // @formatter:off
    list.add(Jeep.builder()
        .modelID(JeepModel.WRANGLER)
        .trimLevel("Sport")
        .numDoors(2)
        .wheelSize(17)
        .basePrice(new BigDecimal("28475.00"))
        .build()
        );

    list.add(Jeep.builder()
        .modelID(JeepModel.WRANGLER)
        .trimLevel("Sport")
        .numDoors(4)
        .wheelSize(17)
        .basePrice(new BigDecimal("31975.00"))
        .build()
        );
    Collections.sort(list);
    // @formatter:on
    return list;
  }

}
```

6) **URL to GitHub Repository:**

https://github.com/david2joh/springweek3.git