



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

Coding Steps:

- ✓1) In the application you've been building add a DAO layer:
 - ✓a) Add the package, `com.promineotech.jeepp.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.
- ✓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. 

I finished the coding for all of the videos, I am too far past this point for a screen shot

 - ✓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. 

```

1 package com.promineotech.jeepp.dao;
2
3 import java.math.BigDecimal;
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @Service
18 @Slf4j
19 public class DefaultJeepSalesDao implements JeepSalesDao {
20
21     @Autowired
22     private NamedParameterJdbcTemplate jdbcTemplate;
23
24     @Override
25     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
26         log.info("DAO: model={}, trim={}", model, trim);
27
28         // @formatter:off
29         String sql = ""
30             + "SELECT * "
31             + "FROM models "
32             + "WHERE model_id = :model_id AND trim_level = :trim_level";
33         // @formatter:on
34
35         Map<String, Object> params = new HashMap<>();
36         params.put("model_id", model.toString());
37         params.put("trim_level", trim);
38
39         return jdbcTemplate.query(sql, params, new RowMapper<Jeep>() {
40             @Override
41             public Jeep mapRow(ResultSet rs, int rowNum) throws SQLException {
42                 // @formatter:off
43                 return Jeep.builder()
44                     .basePrice(new BigDecimal(rs.getString("base_price")))
45                     .modelId(JeepModel.valueOf(rs.getString("model_id")))
46                     .modelPK(rs.getLong("model_pk"))
47                     .numDoors(rs.getInt("num_doors"))
48                     .trimLevel(rs.getString("trim_level"))
49                     .wheelSize(rs.getInt("wheel_size"))
50                     .build();
51                 // @formatter:on
52             }
53         });
54     }
55
56     // end LIST fetchJeeps
57 }

```

- ✓4) Add a getter in the Jeep class for mode1PK. Add the @JsonIgnore annotation to the getter to exclude the mode1PK value from the returned object.
- ✓5) Run the test to produce a green status bar. Produce a screenshot showing the test & status bar.

```

63 //When: A connection is made to the URI
64 ResponseEntity<List<Jeep>> response = getRestTemplate().exchange(uri, HttpMethod.GET, null,
65     new ParameterizedTypeReference<>() {});
66
67 //Then: A valid status code is returned
68 assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);
69
70 List<Jeep> expected = buildExpected();
71 assertThat(response.getBody()).isEqualTo(expected);
72
73
74 // end TEST
75
76 protected List<Jeep> buildExpected() {
77     List<Jeep> list = new LinkedList<>();
78
79     // @formatter:off
80

```

```

2022-10-18 20:04:22.010 INFO 17824 --- [main] c.p.jeepp.controller.FetchJeepTest : Starting
FetchJeepTest using Java 17.0.4.1 on YOUNGBLOOD-SURFACE with PID 17824 (started by jderu in C:\Users\jderu\OneDrive
\Desktop\STS Workspace\jeepp-sales)
2022-10-18 20:04:22.013 DEBUG 17824 --- [main] c.p.jeepp.controller.FetchJeepTest : Running with
Spring Boot v2.7.3, Spring v5.3.22
2022-10-18 20:04:22.015 INFO 17824 --- [main] c.p.jeepp.controller.FetchJeepTest : The following 1
profile is active: "test"
2022-10-18 20:04:20.752 INFO 17824 --- [main] c.p.jeepp.controller.FetchJeepTest : Started
FetchJeepTest in 7.68 seconds (JVM running for 10.302)
2022-10-18 20:04:30.032 INFO 17824 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER,
trim=Sport
2022-10-18 20:04:30.034 INFO 17824 --- [o-auto-1-exec-1] c.p.j.service.DefaultJeepSalesService : The fetchJeeps
method was called with model=WRANGLER and trim=Sport
2022-10-18 20:04:30.034 INFO 17824 --- [o-auto-1-exec-1] c.p.jeepp.dao.DefaultJeepSalesDao : DAO:
model=WRANGLER, trim=Sport

```

URL to GitHub Repository:

<https://github.com/JaxYoungblood/Week15-SpringBootCodingAssignment.git>