Web API Design with Spring Boot Week 4 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:

For this week's homework you need to copy source code from the supplied resources.

For this week's homework you need to copy source code from the Source folder in the supplied resources. Wait until the instructions tell you to copy the resources or you will get errors.

1) Select some options for a Jeep order:

   a) Use the `data.sql` file or the jeep database tables to select options for a Jeep order. Select any one of each of the following for the order:

        i) color
        ii) customer
        iii) engine
        iv) model
        v) tire(s)

    b) Select one or more options from the options table as well. Keep in mind that some options may work better than others – but if you want to put 37-inch tires on your Jeep Renegade, so be it!

2) Create a new integration test class to test a Jeep order named `CreateOrderTest.java`. Create this class in `src/test/java` in the `com.promineotech.jeep.controller` package.

    a) Add the Spring Boot Test annotations: `@SpringBootTest`, `@ActiveProfiles`, and `@Sql`. They should have the same parameters as the test created in weeks 1 and 2.

    b) Create a test method (annotated with `@Test`) named `testCreateOrderReturnsSuccess201`.

    c) In the test class, create a method named `createOrderBody`. This method returns a type of String. In this method, return a JSON object with the IDs that you picked in Step 1a and b. For example:

    ```
    {
      "customer":"MORISON_LINA",
      "model":"WRANGLER",
      "trim":"Sport Altitude",
      "doors":4,
      "color":"EXT_NACHO",
      "engine":"2_0_TURBO",
      "tire":"35_TOYO",
      "options":[
        "DOOR_QUAD_4",
        "EXT_AEV_LIFT",
        "EXT_WARN_WINCH",
        "EXT_WARN_BUMPER_FRONT",
        "EXT_WARN_BUMPER_REAR",
        "EXT_ARB_COMPRESSOR"
      ]
    }
    ```

    Make sure that the JSON is correct! If necessary, use a JSON formatter/validator like the one here: https://jsonformatter.curiousconcept.com/.

    Produce a screenshot of the `createOrderBody()` method.

    In the test method, assign the return value of the `createOrderBody()` method to a variable named `body`.

    d) In the test class, add an instance variable named `serverPort` to hold the port that Tomcat is listening on in the test. Annotate the variable with `@LocalServerPort`.

e) Add another instance variable for an injected `TestRestTemplate` named `restTemplate`.

f) In the test method, assign a value to a local variable named uri as follows:

```
String uri = String.format("http://localhost:%d/orders", serverPort);
```

g) In the test method, create an HttpHeaders object and set the content type to "application/json" like this:

```
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
```

Make sure to import the package `org.springframework.http.HttpHeaders`.

h) Create an HttpEntity object and set the request body and headers:

```
HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
```

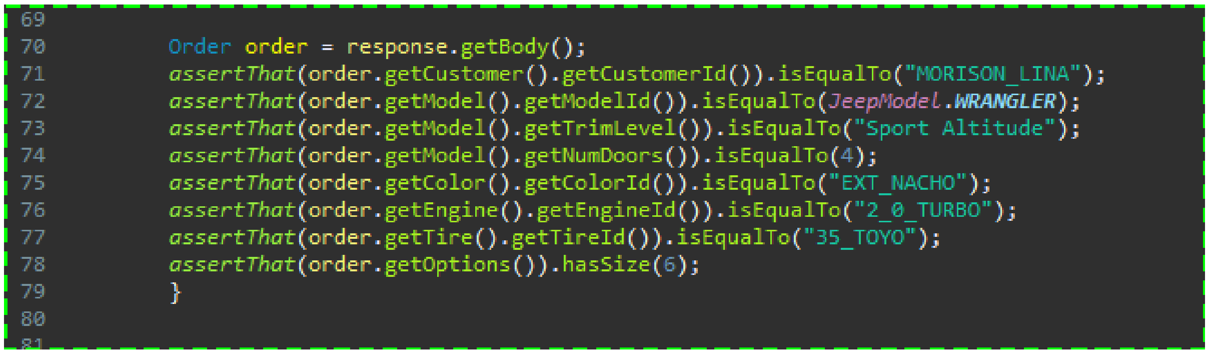
i) Send the request body and headers to the server. The Order class should have been copied earlier from the supplied resources. Ensure that you import `com.promineotech.jeep.entity.Order` and not some other Order class.

```
ResponseEntity<Order> response = restTemplate.exchange(uri,
    HttpMethod.POST, bodyEntity, Order.class);
```

j) Add the AssertJ assertions to ensure that the response is correct. Replace the expected values to match the JSON in step 2c.

```
assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
assertThat(response.getBody()).isNotNull();

Order order = response.getBody();
assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
assertThat(order.getModel().getNumDoors()).isEqualTo(4);
assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
assertThat(order.getOptions()).hasSize(6);
```

k) Produce a screenshot of the test method.



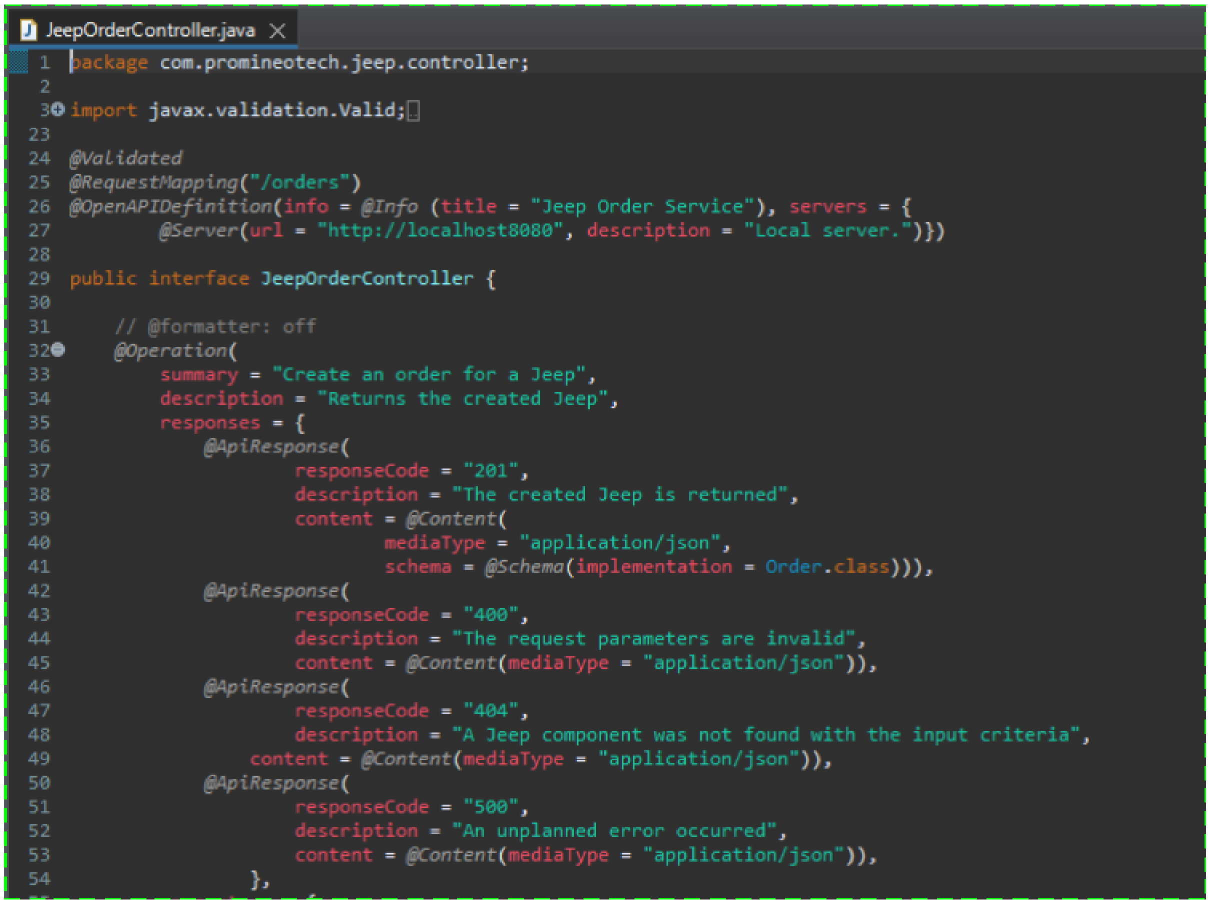
```

        Order order = response.getBody();
        assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
        assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
        assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
        assertThat(order.getModel().getNumDoors()).isEqualTo(4);
        assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
        assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
        assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
        assertThat(order.getOptions()).hasSize(6);
        }

```

3) In the `controller` sub-package in `src/main/java`, create an interface named `JeepOrderController`. Add `@RequestMapping("/orders")` as a class-level annotation.

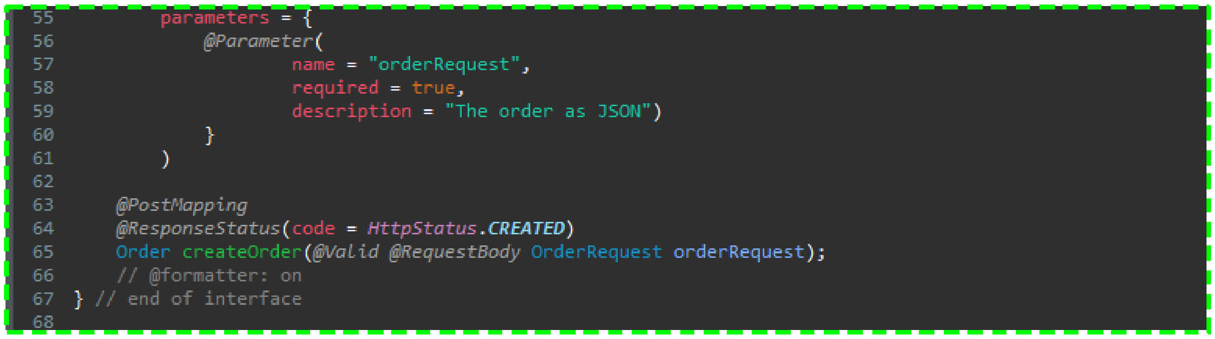
   a) Create a method in the interface to create an order (`createOrder`). It should return an object of type `Order` (see below). It should accept a single parameter of type `OrderRequest` as described in the video. Make sure it accepts an `HTTP POST` request and returns a status code of 201 (created).

   b) Add the `@RequestBody` annotation to the orderRequest parameter. Make sure to add the `RequestBody` annotation from the `org.springframework.web.bind.annotation` package.

   c) Produce a screenshot of the finished `JeepOrderController` interface showing no compile errors.


JeepOrderController.java

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

import javax.validation.Valid;

@Validated
@RequestMapping("/orders")
@OpenAPIDefinition(info = @Info (title = "Jeep Order Service"), servers = {
        @Server(url = "http://localhost8080", description = "Local server.")})

public interface JeepOrderController {

    // @formatter: off
    @Operation(
        summary = "Create an order for a Jeep",
        description = "Returns the created Jeep",
        responses = {
            @ApiResponse(
                    responseCode = "201",
                    description = "The created Jeep is returned",
                    content = @Content(
                            mediaType = "application/json",
                            schema = @Schema(implementation = Order.class))),
            @ApiResponse(
                    responseCode = "400",
                    description = "The request parameters are invalid",
                    content = @Content(mediaType = "application/json")),
            @ApiResponse(
                    responseCode = "404",
                    description = "A Jeep component was not found with the input criteria",
                content = @Content(mediaType = "application/json")),
            @ApiResponse(
                    responseCode = "500",
                    description = "An unplanned error occurred",
                    content = @Content(mediaType = "application/json")),
                },
```



```
        parameters = {
            @Parameter(
                    name = "orderRequest",
                    required = true,
                    description = "The order as JSON")
            }
        )

    @PostMapping
    @ResponseStatus(code = HttpStatus.CREATED)
    Order createOrder(@Valid @RequestBody OrderRequest orderRequest);
    // @formatter: on
} // end of interface
```

4) Create a class that implements `JeepOrderController` named `DefaultJeepOrderController`.

   a) Add `@RestController` as a class-level annotation.

   b) Add a log line to the implementing controller method showing the input request body (orderRequest)

   c) Run the test to show a red status bar. Produce a screenshot that shows the test method, the log line, and the red JUnit status bar.

      *****I coded along with the video (before looking at this assignment). Thus I cannot produce a red status bar.*****

5) Find the Maven dependency `spring-boot-starter-validation` by looking it up at https://mvnrepository.com/. Add this repository to the project POM file (pom.xml).

6) Add the class-level annotation `@Validated` to the `JeepOrderController` interface.

7) Add Bean Validation annotations to the `OrderRequest` class as shown in the video.

   a) Use these annotations for String types:

      i) `@NotNull`
      ii) `@Length(max = 30)`
      iii) `@Pattern(regexp = "[\\w\\s]*")`

   b) Use these annotations for integer types:

      i) `@Positive`
      ii) `@Min(2)`
      iii) `@Max(4)`

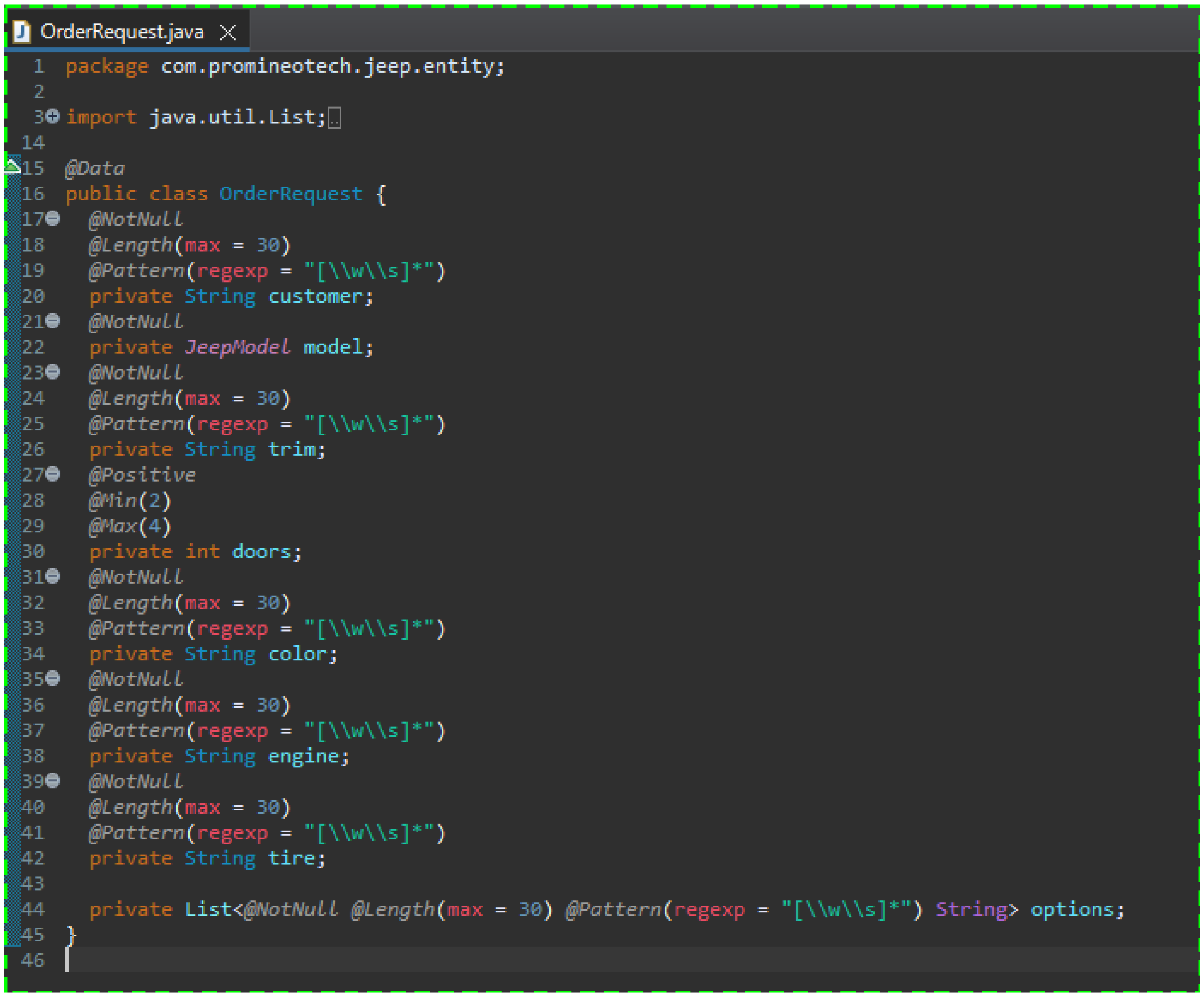
   c) Add `@NotNull` to the enum type.

   d) Add validation to the list element (type String) by adding the validation annotations *inside* the generic definition. So, to add the String validation to the options, you would do this:

```
private List<@NotNull @Length(max = 30) @Pattern(regexp = "[\\w\\s]*") String>
options;
```

Do not apply a `@NotNull` annotation to the `List` because if you have no options the `List` may be null.

e) Produce a screenshot of this class with the annotations.



OrderRequest.java

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

import java.util.List;

@Data
public class OrderRequest {
  @NotNull
  @Length(max = 30)
  @Pattern(regexp = "[\\w\\s]*")
  private String customer;
  @NotNull
  private JeepModel model;
  @NotNull
  @Length(max = 30)
  @Pattern(regexp = "[\\w\\s]*")
  private String trim;
  @Positive
  @Min(2)
  @Max(4)
  private int doors;
  @NotNull
  @Length(max = 30)
  @Pattern(regexp = "[\\w\\s]*")
  private String color;
  @NotNull
  @Length(max = 30)
  @Pattern(regexp = "[\\w\\s]*")
  private String engine;
  @NotNull
  @Length(max = 30)
  @Pattern(regexp = "[\\w\\s]*")
  private String tire;

  private List<@NotNull @Length(max = 30) @Pattern(regexp = "[\\w\\s]*") String> options;
}

```

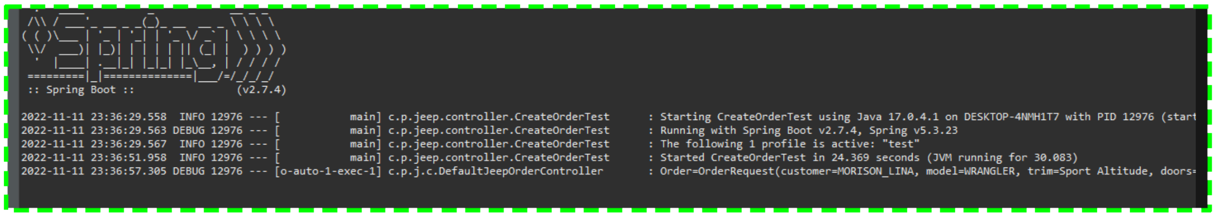
8) In the `jeep.service` sub-package, create the empty (no methods yet) order service interface (named `JeepOrderService`) and implementation (named `DefaultJeepOrderService`).

a) Inject the interface into the order controller implementation class.
b) Add the `@Service` annotation to the service implementation class.
c) Create the `createOrder` method in the interface and implementing service. The method signature should look like this:

```
Order createOrder(OrderRequest orderRequest);
```

d) Call the `createOrder` method from the controller and return the value returned by the service.
e) Add a log line in the createOrder method and log the `orderRequest` parameter.

f) Run the test CreateOrderTest again. Produce a screenshot showing that the service layer createOrder method correctly prints the log line in the console. (e.g. prints out the OrderRequest in the console from within the Service Layer).



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.4)

2022-11-11 23:36:29.558  INFO 12976 --- [           main] c.p.jeep.controller.CreateOrderTest      : Starting CreateOrderTest using Java 17.0.4.1 on DESKTOP-4NMH1T7 with PID 12976 (start
2022-11-11 23:36:29.563 DEBUG 12976 --- [           main] c.p.jeep.controller.CreateOrderTest      : Running with Spring Boot v2.7.4, Spring v5.3.23
2022-11-11 23:36:29.567  INFO 12976 --- [           main] c.p.jeep.controller.CreateOrderTest      : The following 1 profile is active: "test"
2022-11-11 23:36:51.958  INFO 12976 --- [           main] c.p.jeep.controller.CreateOrderTest      : Started CreateOrderTest in 24.369 seconds (JVM running for 30.083)
2022-11-11 23:36:57.305 DEBUG 12976 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepOrderController       : Order=OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altitude, doors=
```

9) In the `jeep.dao` sub-package, create the empty (no methods yet) DAO interface (named `JeepOrderDao`) and implementation (named `DefaultJeepOrderDao`).

   a) Inject the DAO interface into the order service implementation class.
   b) Add the `@Component` annotation to the DAO implementation class.

10) Replace the entire content of `JeepOrderDao.java` with the source found in `JeepOrderDao.source`. The source file is found in the Source folder of the supplied project resources.

11) ***** The next steps require you to copy source code from the Source directory in the supplied resources. Please follow the instructions EXACTLY. Some steps require you to replace ALL the source in a file. Some steps require you to ADD source to a file.**

12) Copy the *contents* of the file `DefaultJeepOrderDao.source` *into* `DefaultJeepOrderDao.java`. The source file is found in the Source folder of the supplied project resources.

    In Eclipse, click the "Source" menu and select "Organize Imports". Pick packages from your project where applicable. Make sure you pick the import `java.util.Optional`, `java.util.List`, and `org.springframework.jdbc.core.RowMapper`.

13) Copy the *contents* of the file `DefaultJeepOrderService.source` *into* `DefaultJeepOrderService.java`. Add the source after the `createOrder()` method, but *inside* the class body. The source file is found in the `Source` folder of the supplied project resources.

    In Eclipse, click the "Source" menu and select "Organize Imports". Pick packages from your project where applicable.

14) In `DefaultJeepOrderService.java`, work with the method `createOrder`.

    a) Add the `@Transactional` annotation to the createOrder method.

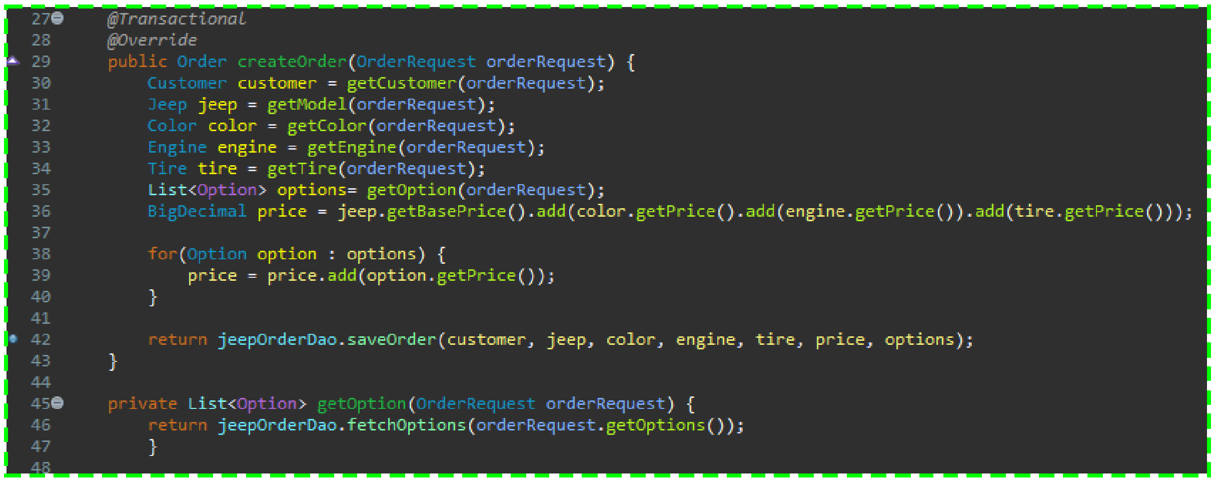
    b) In the `createOrder` method call the copied methods: `getCustomer`, `getModel`, `getColor`, `getEngine`, `getTire` and `getOption`, assigning the return values of these methods to variables of the appropriate types.

    c) Calculate the price, including all options.

15) In `JeepOrderDao.java` and `DefaultJeepOrderDao.java`, add the method:

```
Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire
        tire, BigDecimal price, List<Option> options);
```

a) Call the jeepOrder.Dao.saveOrder method from the jeepOrderSalesService.createOrder service. Produce a screenshot of the jeepOrderSalesService.createOrder method.



```
    @Transactional
    @Override
    public Order createOrder(OrderRequest orderRequest) {
        Customer customer = getCustomer(orderRequest);
        Jeep jeep = getModel(orderRequest);
        Color color = getColor(orderRequest);
        Engine engine = getEngine(orderRequest);
        Tire tire = getTire(orderRequest);
        List<Option> options= getOption(orderRequest);
        BigDecimal price = jeep.getBasePrice().add(color.getPrice().add(engine.getPrice()).add(tire.getPrice()));

        for(Option option : options) {
            price = price.add(option.getPrice());
        }

        return jeepOrderDao.saveOrder(customer, jeep, color, engine, tire, price, options);
    }

    private List<Option> getOption(OrderRequest orderRequest) {
        return jeepOrderDao.fetchOptions(orderRequest.getOptions());
        }
```

b) Write the implementation of the `saveOrder` method in the DAO.

i) Call the supplied `generateInsertSql` method, passing in the customer, jeep, color, engine, tire and price. Assign the return value of the method to a `SqlParams` object.

ii) Call the `update` method on the `NamedParameterJdbcTemplate` object, passing in a `KeyHolder` object as shown in the video. Create the `KeyHolder` like this:

```
KeyHolder keyHolder = new GeneratedKeyHolder();
```

Be sure to extract the order primary key from the `KeyHolder` object into a variable of type `Long` named `orderPK`.

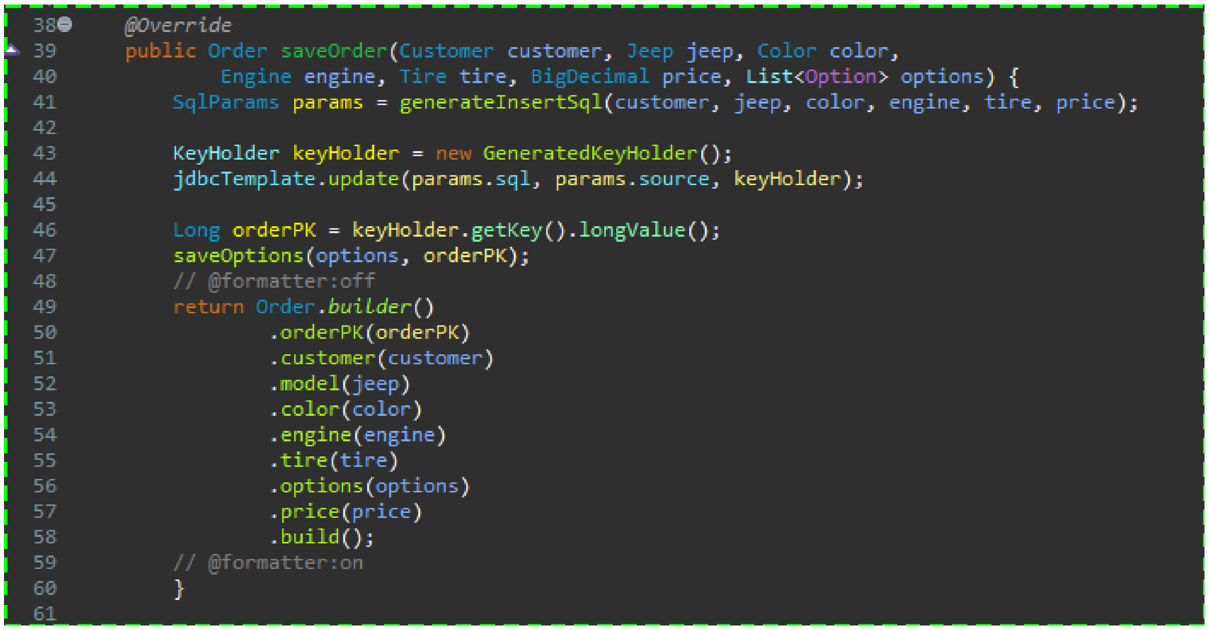
iii) Write a method named `saveOptions` as shown in the video. This method should have the following method signature:

```
private void saveOptions(List<Option> options, Long orderPK)
```

For each option in the `Options` list, call the supplied `generateInsertSql` method passing the parameters `option` and order primary key (`orderPK`). Call the `update` method on the `NamedParameterJdbcTemplate` object.

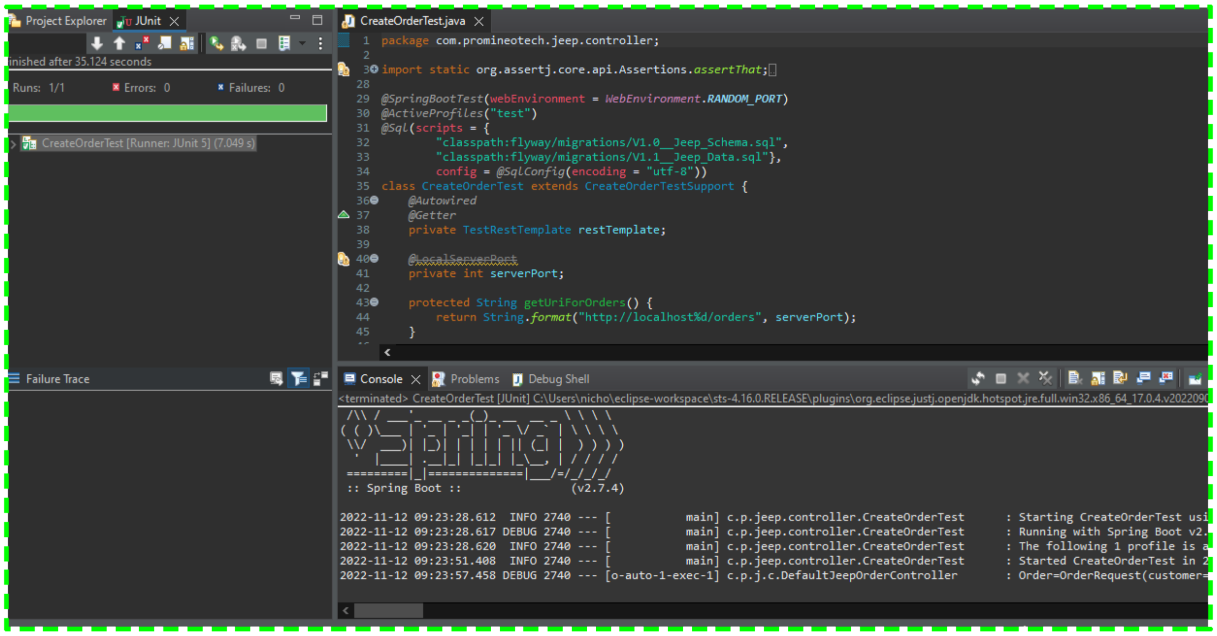
iv) In the `saveOrder` method in the DAO implementation, return an `Order` object using the `Order.builder`. The `Order` should include orderPK, customer, jeep (model), color, engine, tire, options and price.

v) Produce a screenshot of the saveOrder method.



```java
    @Override
    public Order saveOrder(Customer customer, Jeep jeep, Color color,
            Engine engine, Tire tire, BigDecimal price, List<Option> options) {
        SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);

        KeyHolder keyHolder = new GeneratedKeyHolder();
        jdbcTemplate.update(params.sql, params.source, keyHolder);

        Long orderPK = keyHolder.getKey().longValue();
        saveOptions(options, orderPK);
        // @formatter:off
        return Order.builder()
                .orderPK(orderPK)
                .customer(customer)
                .model(jeep)
                .color(color)
                .engine(engine)
                .tire(tire)
                .options(options)
                .price(price)
                .build();
        // @formatter:on
        }

```

c) Run the integration test in `CreateOrderTest`. Produce a screenshot of the test method that shows the green JUnit status bar, the console output, and the test class.



Project Explorer JUnit

inished after 35.124 seconds

Runs: 1/1 Errors: 0 Failures: 0

CreateOrderTest [Runner: JUnit 5] (7.049 s)

Failure Trace

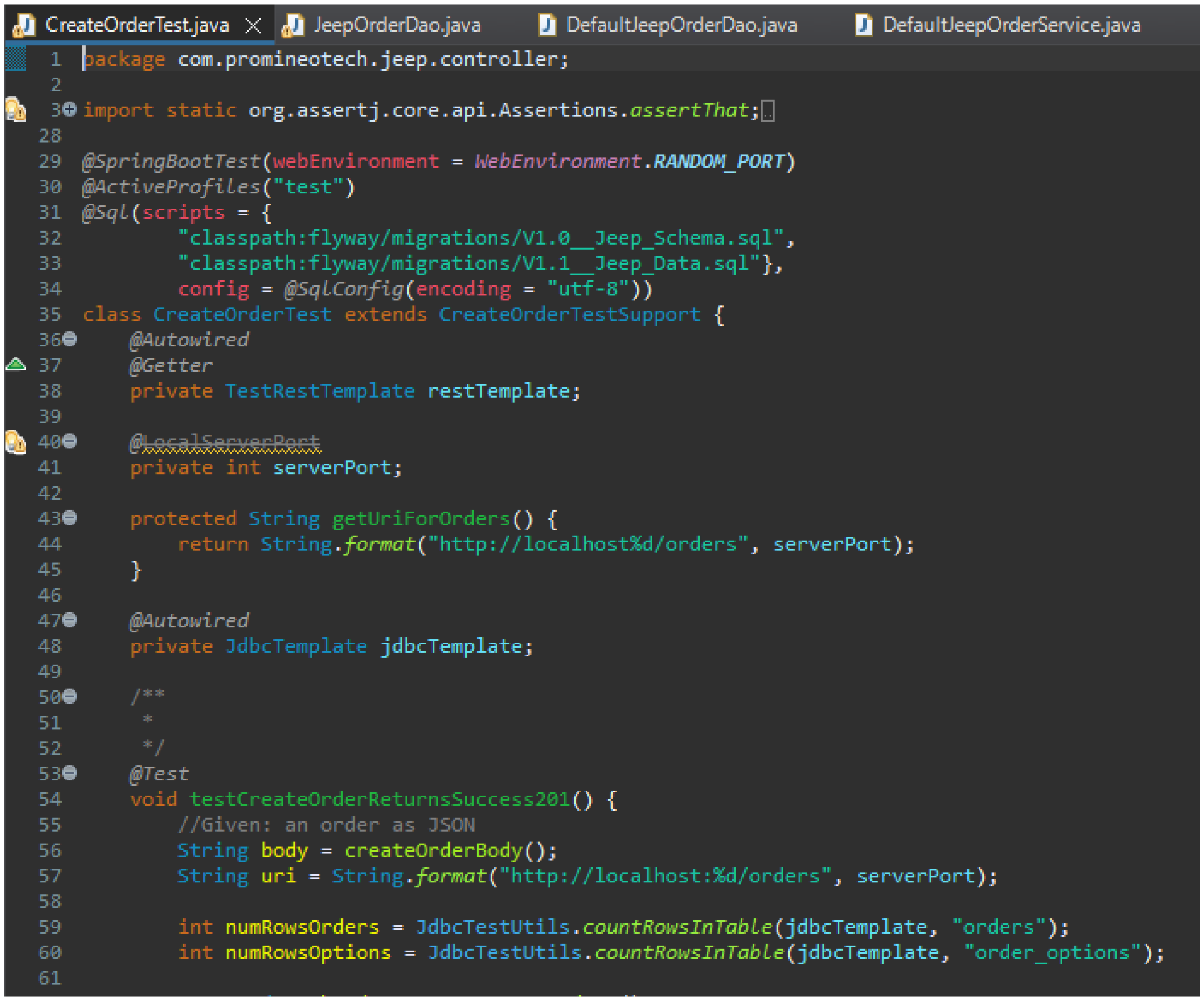
CreateOrderTest.java

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

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

@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 CreateOrderTest extends CreateOrderTestSupport {
    @Autowired
    @Getter
    private TestRestTemplate restTemplate;

    @LocalServerPort
    private int serverPort;

    protected String getUriForOrders() {
        return String.format("http://localhost%d/orders", serverPort);
    }
```

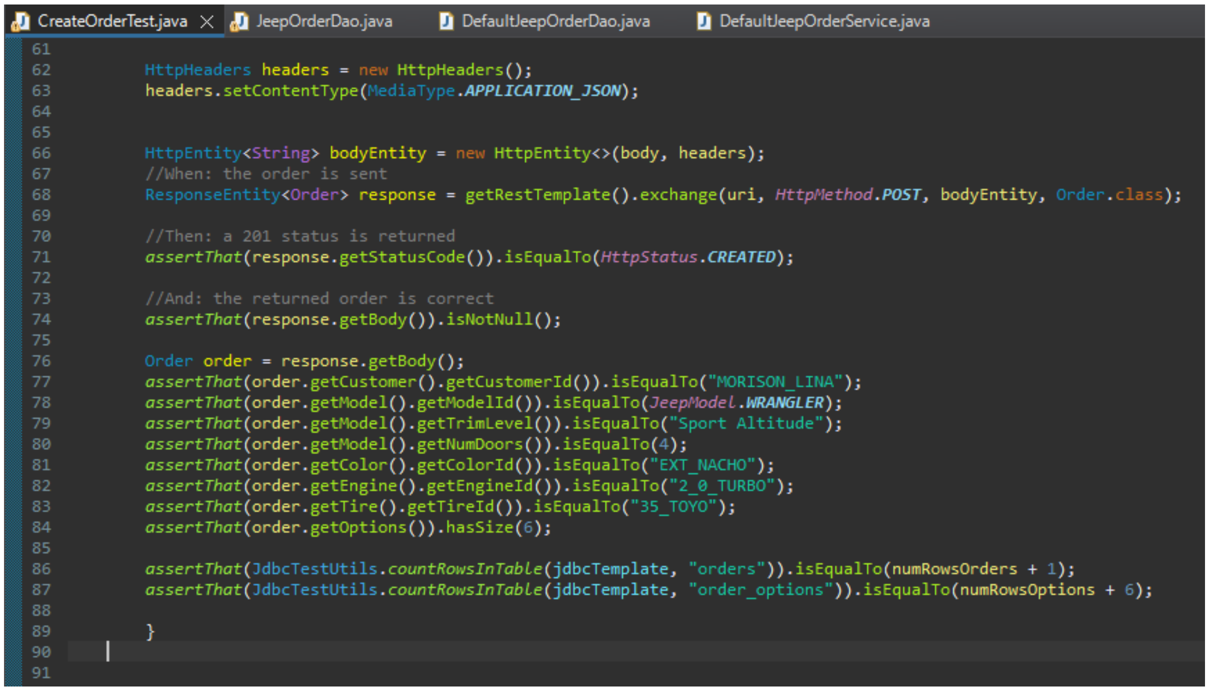
Console Problems Debug Shell

```
<terminated> CreateOrderTest [JUnit] C:\Users\nicho\eclipse-workspace\sts-4.16.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v2022090
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.4)

2022-11-12 09:23:28.612  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Starting CreateOrderTest usi
2022-11-12 09:23:28.617 DEBUG 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Running with Spring Boot v2.
2022-11-12 09:23:28.620  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : The following 1 profile is a
2022-11-12 09:23:51.408  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Started CreateOrderTest in 2
2022-11-12 09:23:57.458 DEBUG 2740 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepOrderController       : Order=OrderRequest(customer=
```

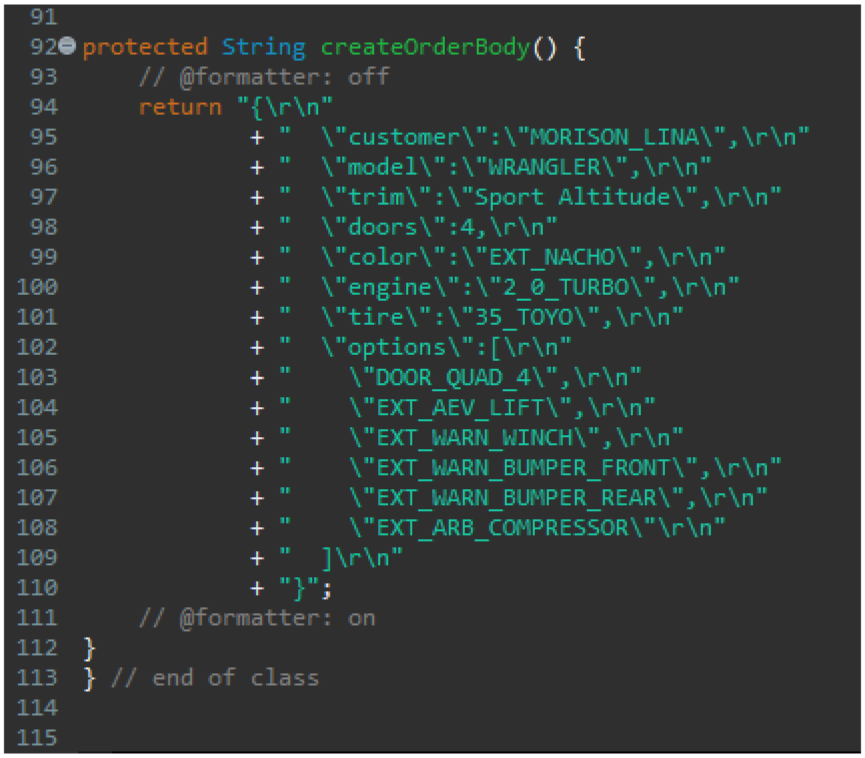
Screenshots of Code:



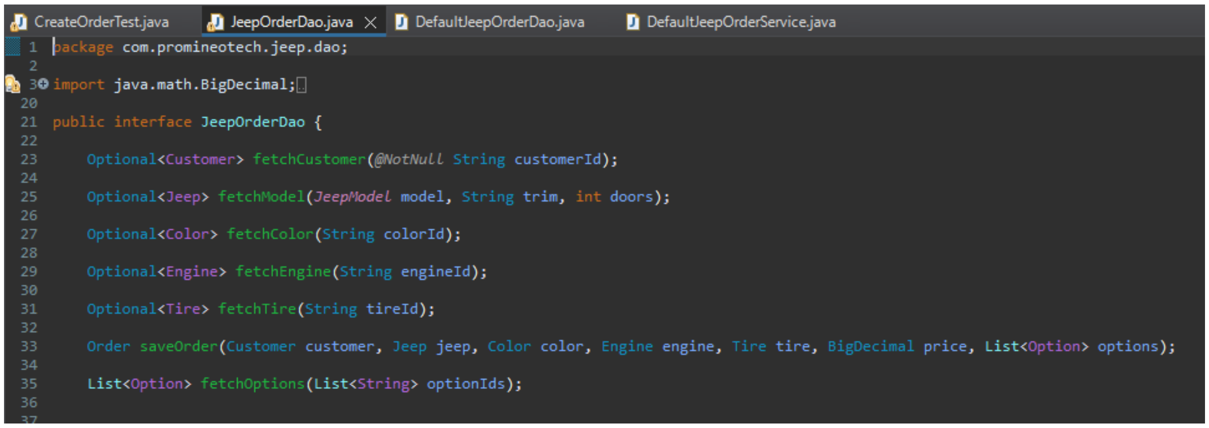
```java
package com.promineotech.jeep.controller;

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

@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 CreateOrderTest extends CreateOrderTestSupport {
    @Autowired
    @Getter
    private TestRestTemplate restTemplate;

    @LocalServerPort
    private int serverPort;

    protected String getUriForOrders() {
        return String.format("http://localhost%d/orders", serverPort);
    }

    @Autowired
    private JdbcTemplate jdbcTemplate;

    /**
     *
     */
    @Test
    void testCreateOrderReturnsSuccess201() {
        //Given: an order as JSON
        String body = createOrderBody();
        String uri = String.format("http://localhost:%d/orders", serverPort);

        int numRowsOrders = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
        int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");

```



```java

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);


        HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);
        //When: the order is sent
        ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST, bodyEntity, Order.class);

        //Then: a 201 status is returned
        assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);

        //And: the returned order is correct
        assertThat(response.getBody()).isNotNull();

        Order order = response.getBody();
        assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
        assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
        assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
        assertThat(order.getModel().getNumDoors()).isEqualTo(4);
        assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
        assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
        assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
        assertThat(order.getOptions()).hasSize(6);

        assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders")).isEqualTo(numRowsOrders + 1);
        assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options")).isEqualTo(numRowsOptions + 6);

        }

```

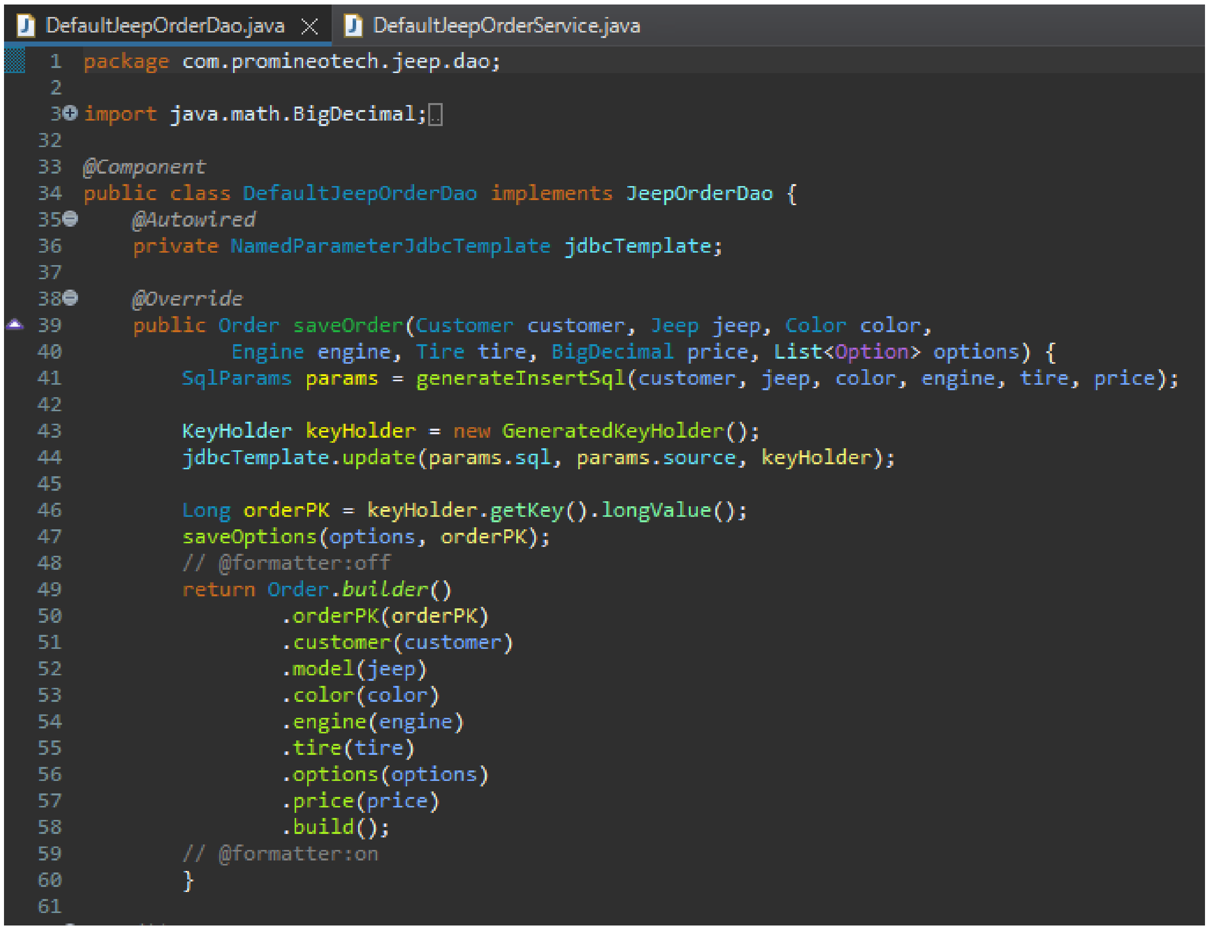


```java

protected String createOrderBody() {
    // @formatter: off
    return "{\r\n"
            + "  \"customer\":\"MORISON_LINA\",\r\n"
            + "  \"model\":\"WRANGLER\",\r\n"
            + "  \"trim\":\"Sport Altitude\",\r\n"
            + "  \"doors\":4,\r\n"
            + "  \"color\":\"EXT_NACHO\",\r\n"
            + "  \"engine\":\"2_0_TURBO\",\r\n"
            + "  \"tire\":\"35_TOYO\",\r\n"
            + "  \"options\":[\r\n"
            + "    \"DOOR_QUAD_4\",\r\n"
            + "    \"EXT_AEV_LIFT\",\r\n"
            + "    \"EXT_WARN_WINCH\",\r\n"
            + "    \"EXT_WARN_BUMPER_FRONT\",\r\n"
            + "    \"EXT_WARN_BUMPER_REAR\",\r\n"
            + "    \"EXT_ARB_COMPRESSOR\"\r\n"
            + "  ]\r\n"
            + "}";
    // @formatter: on
}
} // end of class

```



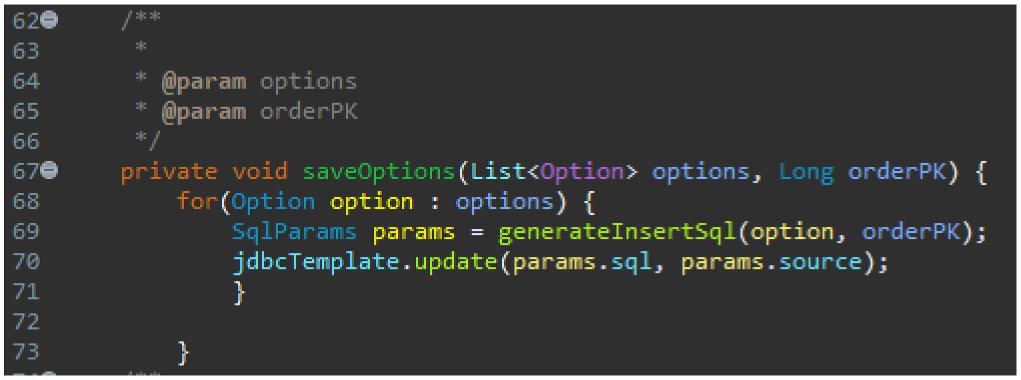
CreateOrderTest.java JeepOrderDao.java DefaultJeepOrderDao.java DefaultJeepOrderService.java

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

import java.math.BigDecimal;

public interface JeepOrderDao {

    Optional<Customer> fetchCustomer(@NotNull String customerId);

    Optional<Jeep> fetchModel(JeepModel model, String trim, int doors);

    Optional<Color> fetchColor(String colorId);

    Optional<Engine> fetchEngine(String engineId);

    Optional<Tire> fetchTire(String tireId);

    Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire, BigDecimal price, List<Option> options);

    List<Option> fetchOptions(List<String> optionIds);

```

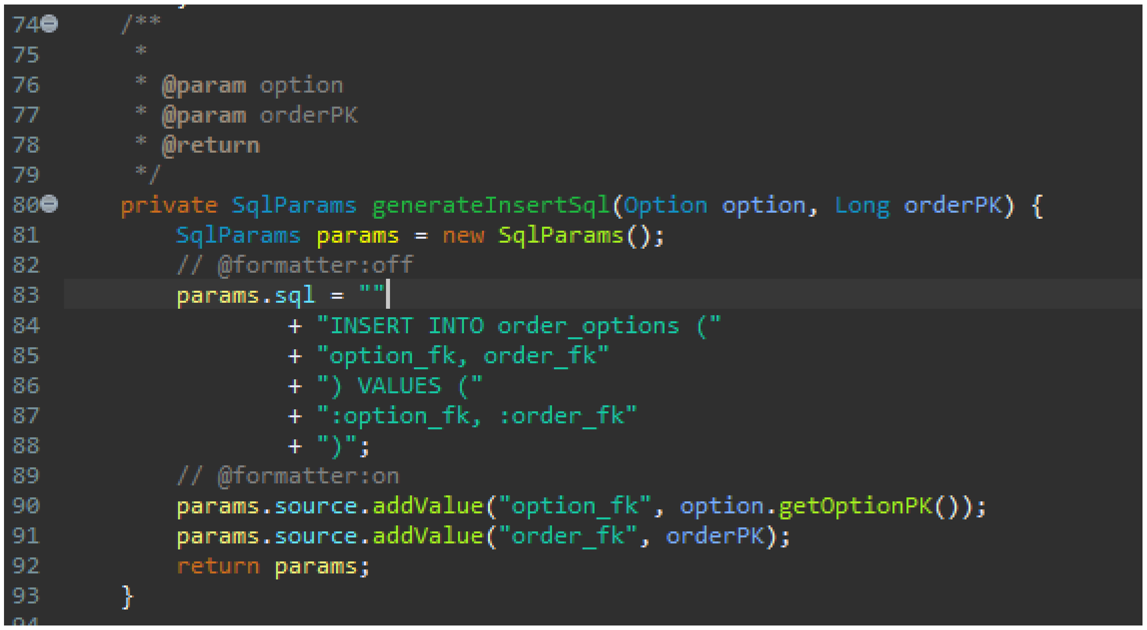


DefaultJeepOrderDao.java   DefaultJeepOrderService.java

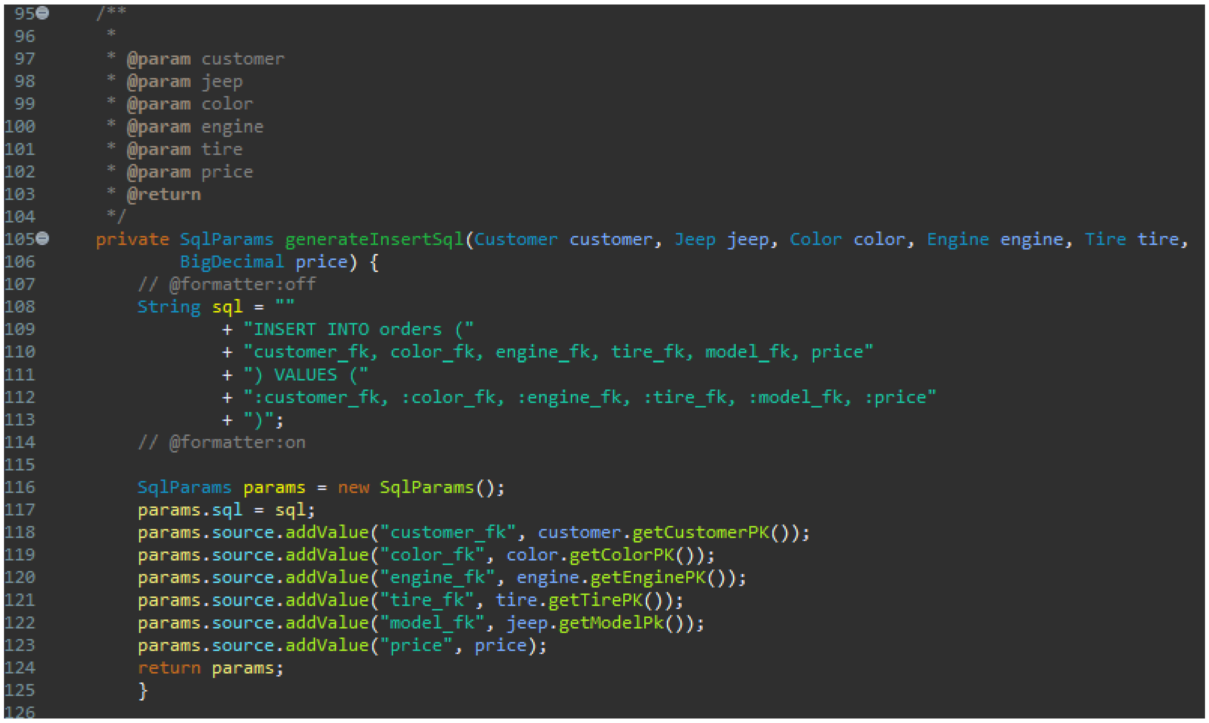
```java
package com.promineotech.jeep.dao;

import java.math.BigDecimal;

@Component
public class DefaultJeepOrderDao implements JeepOrderDao {
    @Autowired
    private NamedParameterJdbcTemplate jdbcTemplate;

    @Override
    public Order saveOrder(Customer customer, Jeep jeep, Color color,
            Engine engine, Tire tire, BigDecimal price, List<Option> options) {
        SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);

        KeyHolder keyHolder = new GeneratedKeyHolder();
        jdbcTemplate.update(params.sql, params.source, keyHolder);

        Long orderPK = keyHolder.getKey().longValue();
        saveOptions(options, orderPK);
        // @formatter:off
        return Order.builder()
                .orderPK(orderPK)
                .customer(customer)
                .model(jeep)
                .color(color)
                .engine(engine)
                .tire(tire)
                .options(options)
                .price(price)
                .build();
        // @formatter:on
        }

```



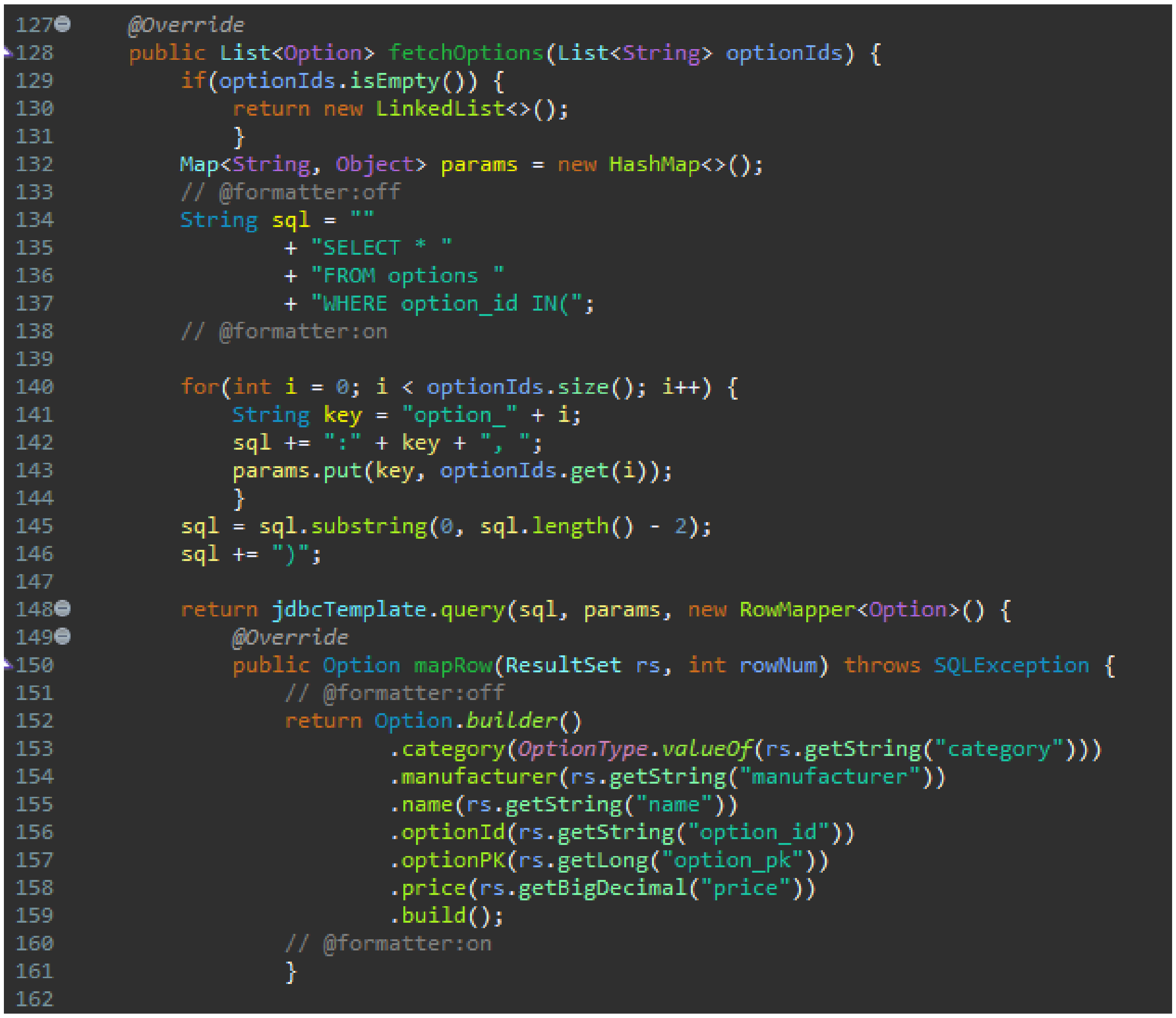
```java
    /**
     *
     * @param options
     * @param orderPK
     */
    private void saveOptions(List<Option> options, Long orderPK) {
        for(Option option : options) {
            SqlParams params = generateInsertSql(option, orderPK);
            jdbcTemplate.update(params.sql, params.source);
            }

        }
```



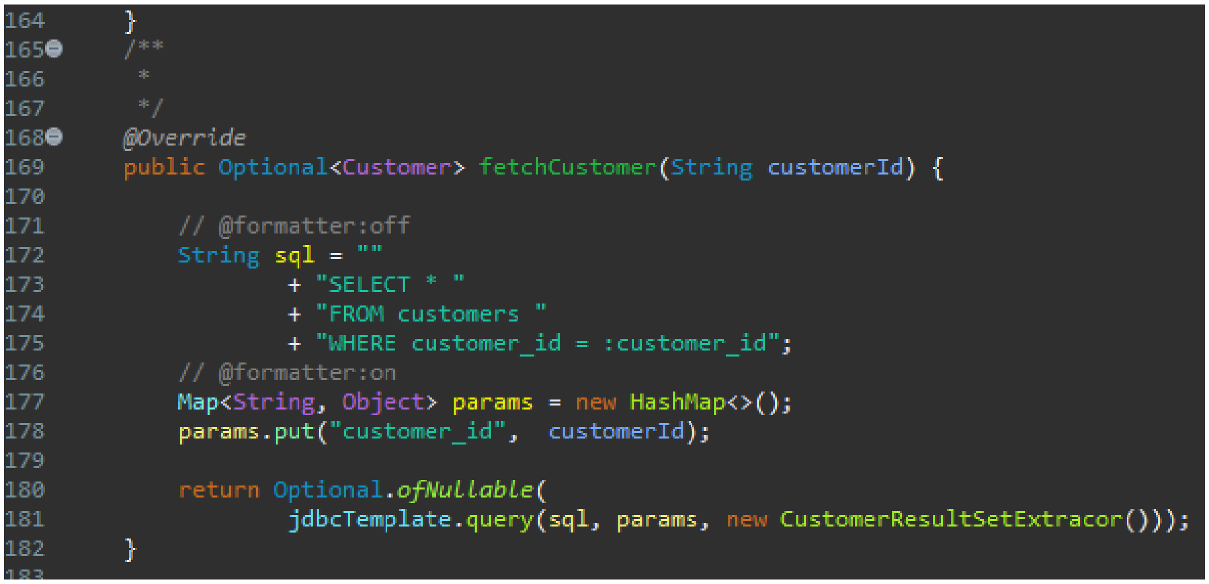
```java
    /**
     *
     * @param option
     * @param orderPK
     * @return
     */
    private SqlParams generateInsertSql(Option option, Long orderPK) {
        SqlParams params = new SqlParams();
        // @formatter:off
        params.sql = ""
                + "INSERT INTO order_options ("
                + "option_fk, order_fk"
                + ") VALUES ("
                + ":option_fk, :order_fk"
                + ")";
        // @formatter:on
        params.source.addValue("option_fk", option.getOptionPK());
        params.source.addValue("order_fk", orderPK);
        return params;
    }
```



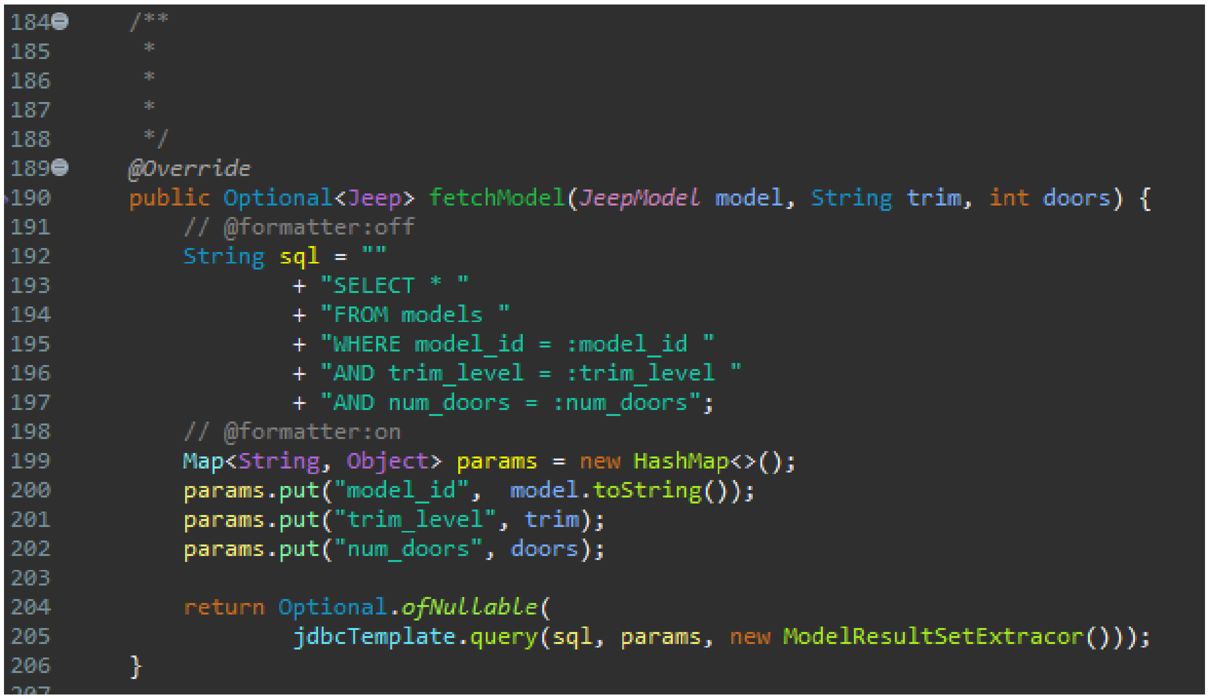
```java
    /**
     *
     * @param customer
     * @param jeep
     * @param color
     * @param engine
     * @param tire
     * @param price
     * @return
     */
    private SqlParams generateInsertSql(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire,
            BigDecimal price) {
        // @formatter:off
        String sql = ""
                + "INSERT INTO orders ("
                + "customer_fk, color_fk, engine_fk, tire_fk, model_fk, price"
                + ") VALUES ("
                + ":customer_fk, :color_fk, :engine_fk, :tire_fk, :model_fk, :price"
                + ")";
        // @formatter:on

        SqlParams params = new SqlParams();
        params.sql = sql;
        params.source.addValue("customer_fk", customer.getCustomerPK());
        params.source.addValue("color_fk", color.getColorPK());
        params.source.addValue("engine_fk", engine.getEnginePK());
        params.source.addValue("tire_fk", tire.getTirePK());
        params.source.addValue("model_fk", jeep.getModelPk());
        params.source.addValue("price", price);
        return params;
        }
```



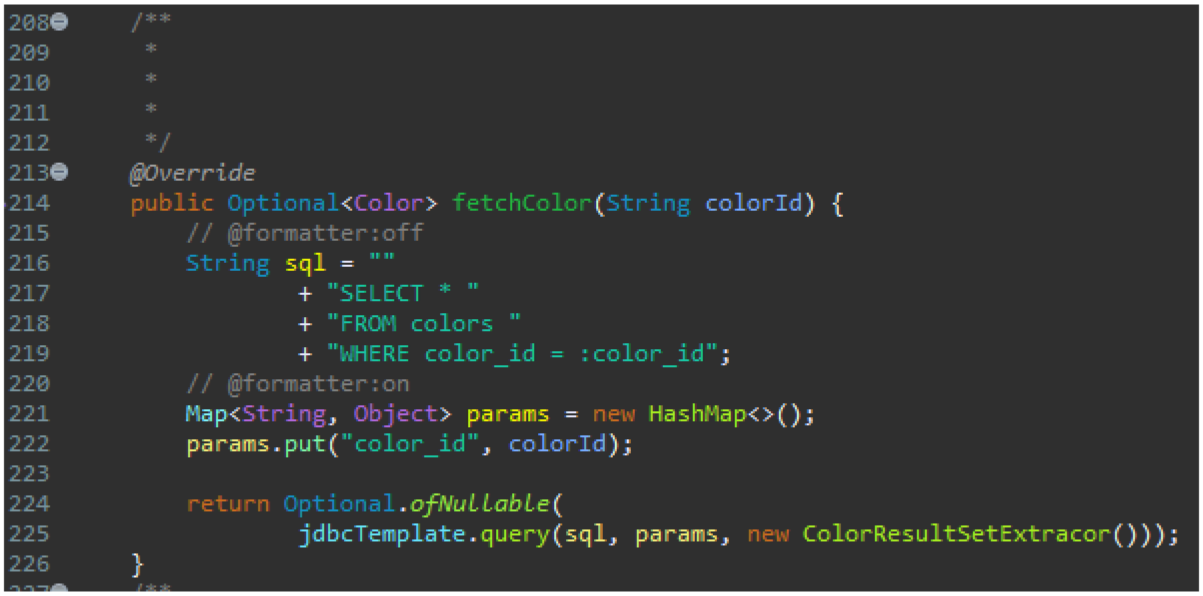
```java
@Override
public List<Option> fetchOptions(List<String> optionIds) {
    if(optionIds.isEmpty()) {
        return new LinkedList<>();
        }
    Map<String, Object> params = new HashMap<>();
    // @formatter:off
    String sql = ""
            + "SELECT * "
            + "FROM options "
            + "WHERE option_id IN(";
    // @formatter:on

    for(int i = 0; i < optionIds.size(); i++) {
        String key = "option_" + i;
        sql += ":" + key + ", ";
        params.put(key, optionIds.get(i));
        }
    sql = sql.substring(0, sql.length() - 2);
    sql += ")";

    return jdbcTemplate.query(sql, params, new RowMapper<Option>() {
        @Override
        public Option mapRow(ResultSet rs, int rowNum) throws SQLException {
            // @formatter:off
            return Option.builder()
                    .category(OptionType.valueOf(rs.getString("category")))
                    .manufacturer(rs.getString("manufacturer"))
                    .name(rs.getString("name"))
                    .optionId(rs.getString("option_id"))
                    .optionPK(rs.getLong("option_pk"))
                    .price(rs.getBigDecimal("price"))
                    .build();
            // @formatter:on
            }

```



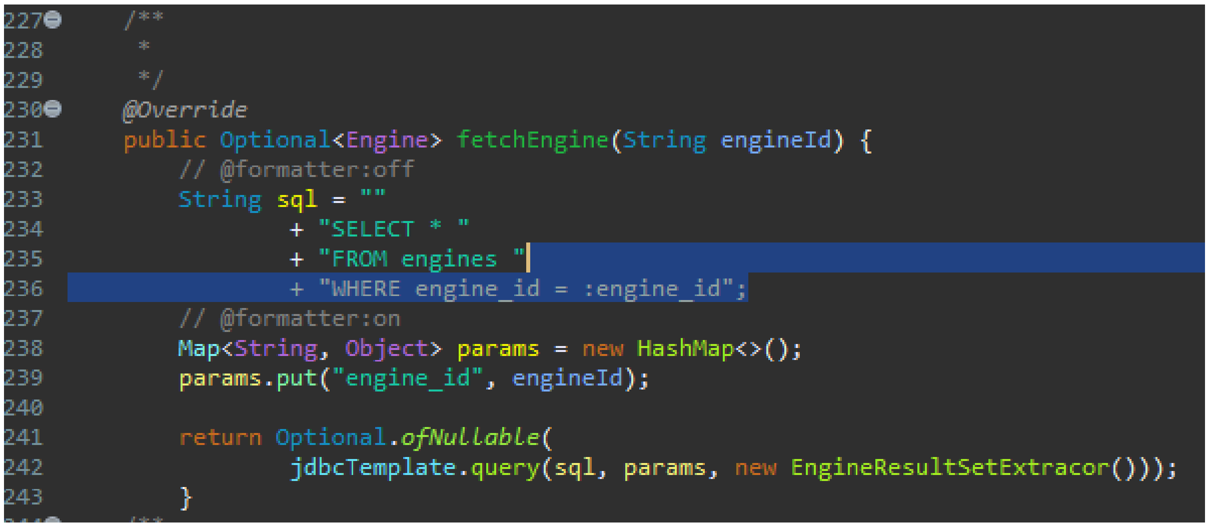
```java
}
/**
 *
 */
@Override
public Optional<Customer> fetchCustomer(String customerId) {

    // @formatter:off
    String sql = ""
            + "SELECT * "
            + "FROM customers "
            + "WHERE customer_id = :customer_id";
    // @formatter:on
    Map<String, Object> params = new HashMap<>();
    params.put("customer_id",  customerId);

    return Optional.ofNullable(
            jdbcTemplate.query(sql, params, new CustomerResultSetExtracor()));
}
```



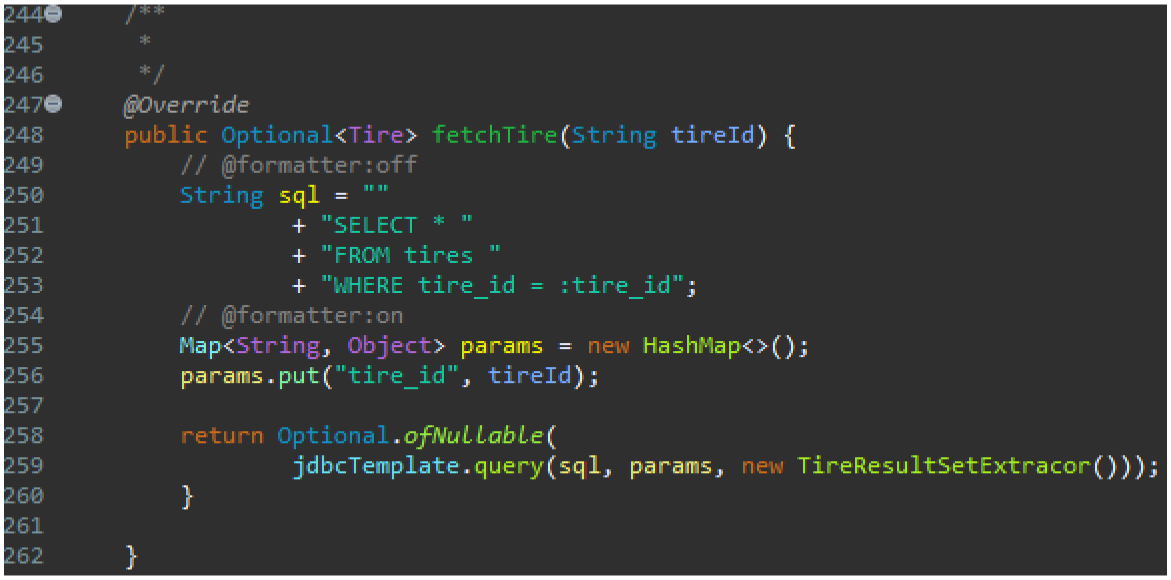
```java
    /**
     *
     *
     *
     */
    @Override
    public Optional<Jeep> fetchModel(JeepModel model, String trim, int doors) {
        // @formatter:off
        String sql = ""
                + "SELECT * "
                + "FROM models "
                + "WHERE model_id = :model_id "
                + "AND trim_level = :trim_level "
                + "AND num_doors = :num_doors";
        // @formatter:on
        Map<String, Object> params = new HashMap<>();
        params.put("model_id",  model.toString());
        params.put("trim_level", trim);
        params.put("num_doors", doors);

        return Optional.ofNullable(
                jdbcTemplate.query(sql, params, new ModelResultSetExtracor()));
    }
```



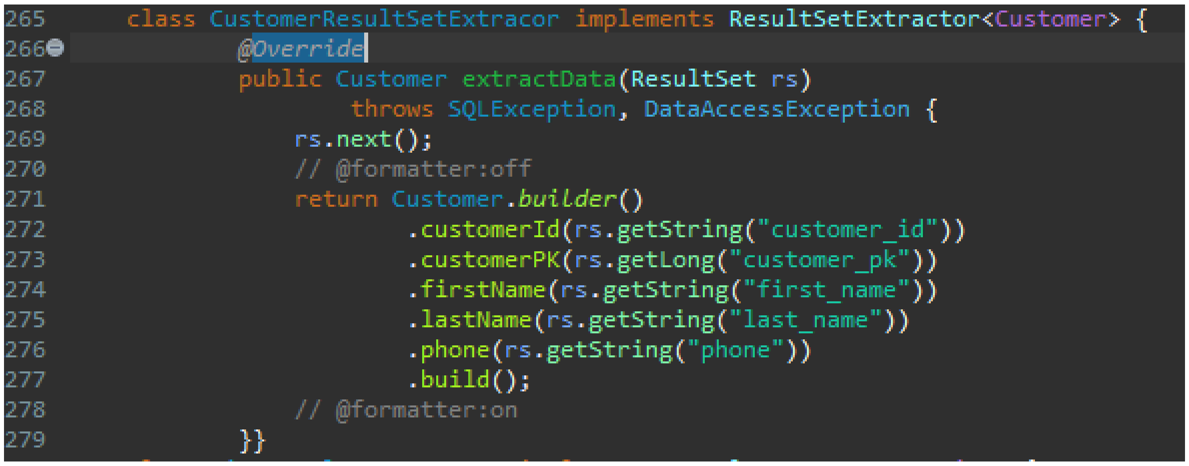
```java
    /**
     *
     *
     *
     */
    @Override
    public Optional<Color> fetchColor(String colorId) {
        // @formatter:off
        String sql = ""
                + "SELECT * "
                + "FROM colors "
                + "WHERE color_id = :color_id";
        // @formatter:on
        Map<String, Object> params = new HashMap<>();
        params.put("color_id", colorId);

        return Optional.ofNullable(
                jdbcTemplate.query(sql, params, new ColorResultSetExtracor()));
    }
```



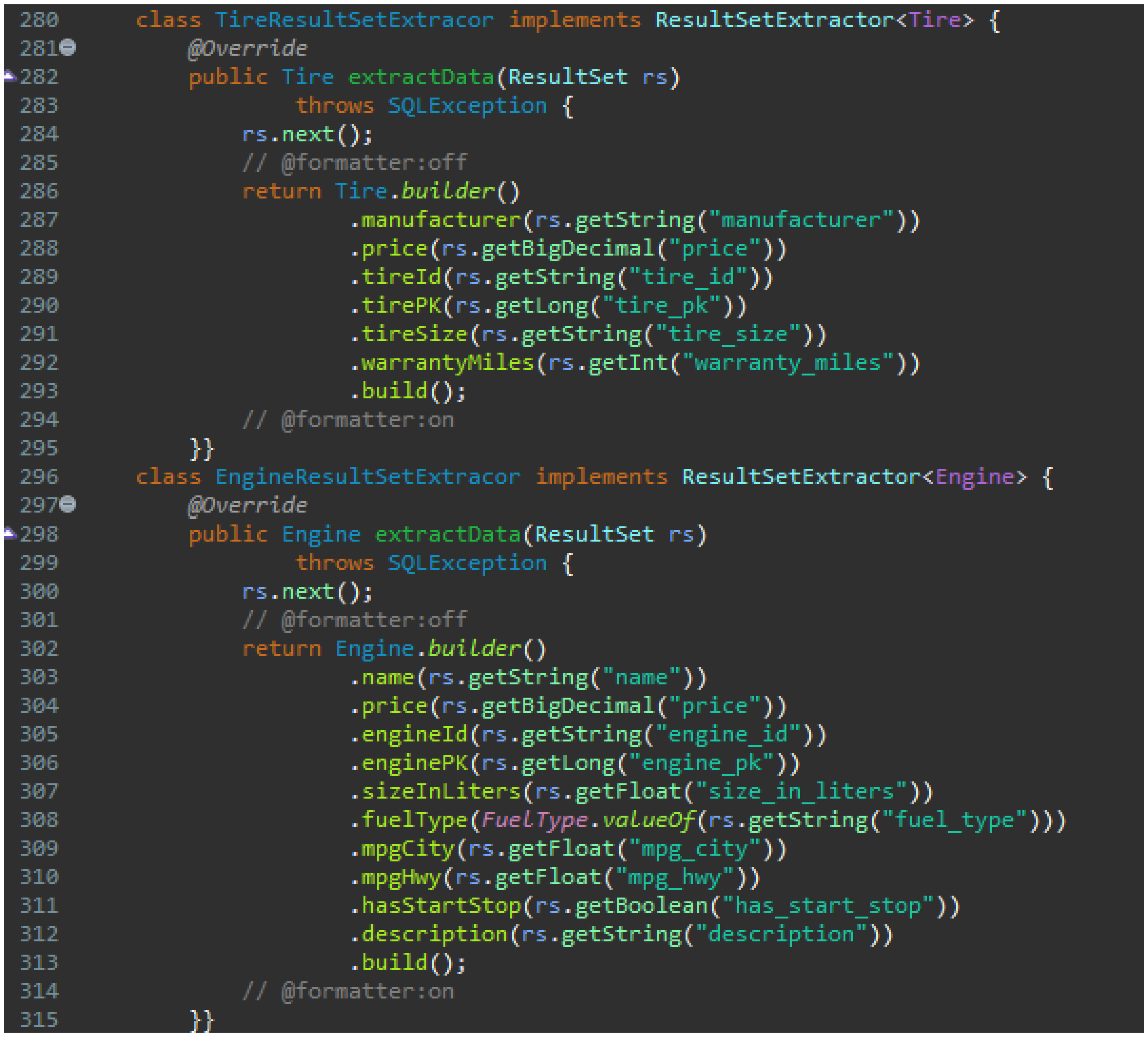
```java
    /**
     *
     */
    @Override
    public Optional<Engine> fetchEngine(String engineId) {
        // @formatter:off
        String sql = ""
                + "SELECT * "
                + "FROM engines "
                + "WHERE engine_id = :engine_id";
        // @formatter:on
        Map<String, Object> params = new HashMap<>();
        params.put("engine_id", engineId);

        return Optional.ofNullable(
                jdbcTemplate.query(sql, params, new EngineResultSetExtracor()));
        }
```



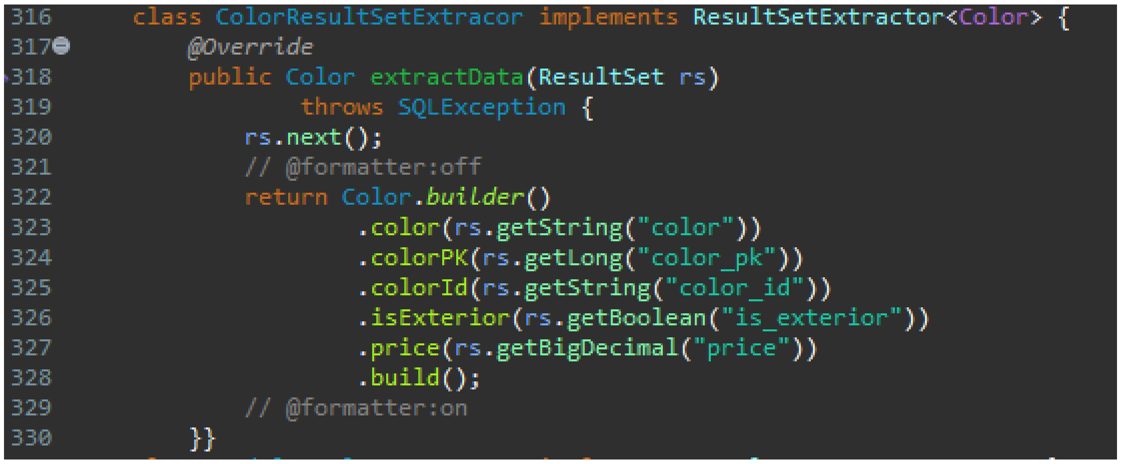
```java
    /**
     *
     */
    @Override
    public Optional<Tire> fetchTire(String tireId) {
        // @formatter:off
        String sql = ""
                + "SELECT * "
                + "FROM tires "
                + "WHERE tire_id = :tire_id";
        // @formatter:on
        Map<String, Object> params = new HashMap<>();
        params.put("tire_id", tireId);

        return Optional.ofNullable(
                jdbcTemplate.query(sql, params, new TireResultSetExtracor()));
        }

    }
```



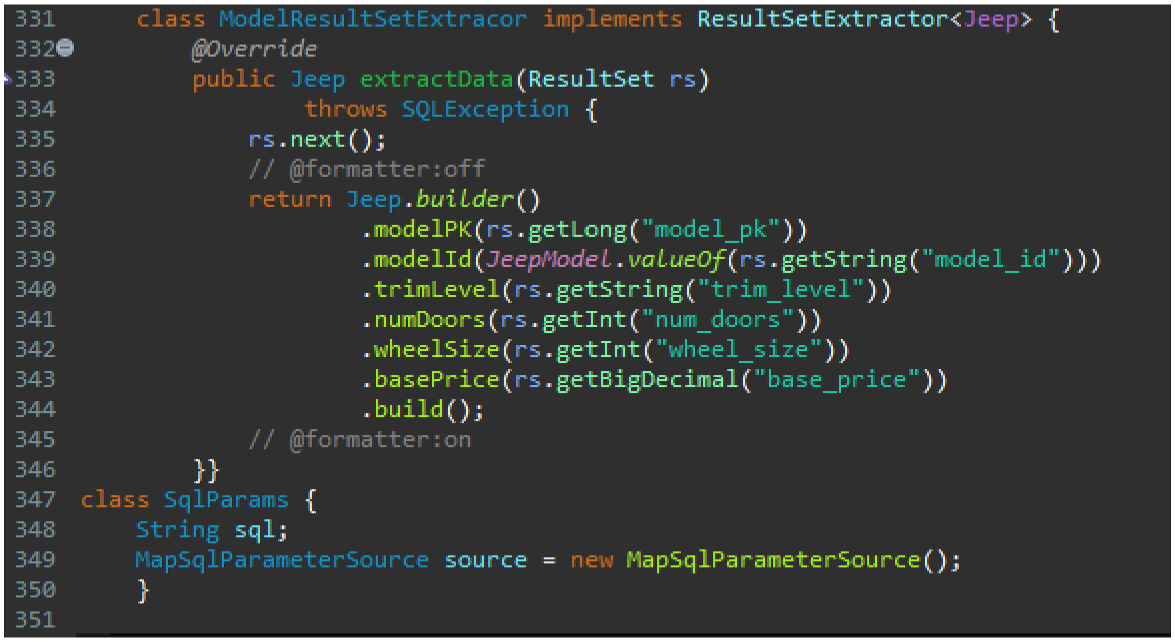
```java
    class CustomerResultSetExtracor implements ResultSetExtractor<Customer> {
            @Override
            public Customer extractData(ResultSet rs)
                    throws SQLException, DataAccessException {
                rs.next();
                // @formatter:off
                return Customer.builder()
                        .customerId(rs.getString("customer_id"))
                        .customerPK(rs.getLong("customer_pk"))
                        .firstName(rs.getString("first_name"))
                        .lastName(rs.getString("last_name"))
                        .phone(rs.getString("phone"))
                        .build();
                // @formatter:on
            }}
```



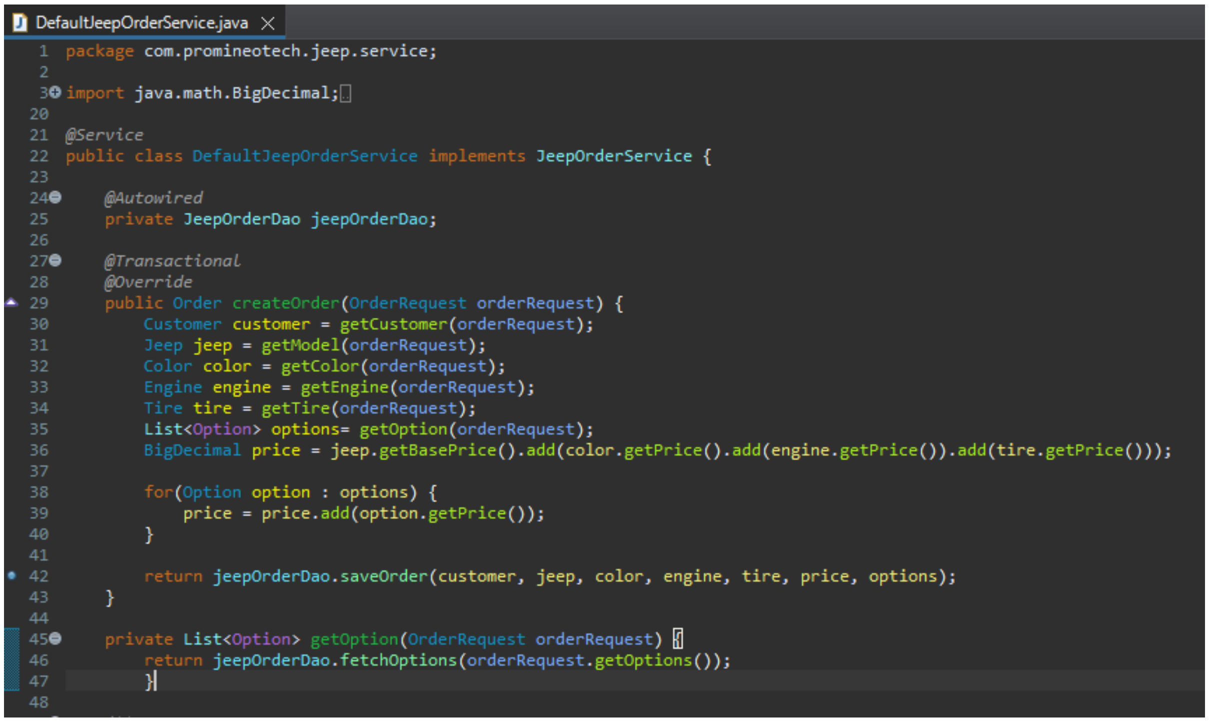
```java
    class TireResultSetExtracor implements ResultSetExtractor<Tire> {
        @Override
        public Tire extractData(ResultSet rs)
                throws SQLException {
            rs.next();
            // @formatter:off
            return Tire.builder()
                    .manufacturer(rs.getString("manufacturer"))
                    .price(rs.getBigDecimal("price"))
                    .tireId(rs.getString("tire_id"))
                    .tirePK(rs.getLong("tire_pk"))
                    .tireSize(rs.getString("tire_size"))
                    .warrantyMiles(rs.getInt("warranty_miles"))
                    .build();
            // @formatter:on
        }}
    class EngineResultSetExtracor implements ResultSetExtractor<Engine> {
        @Override
        public Engine extractData(ResultSet rs)
                throws SQLException {
            rs.next();
            // @formatter:off
            return Engine.builder()
                    .name(rs.getString("name"))
                    .price(rs.getBigDecimal("price"))
                    .engineId(rs.getString("engine_id"))
                    .enginePK(rs.getLong("engine_pk"))
                    .sizeInLiters(rs.getFloat("size_in_liters"))
                    .fuelType(FuelType.valueOf(rs.getString("fuel_type")))
                    .mpgCity(rs.getFloat("mpg_city"))
                    .mpgHwy(rs.getFloat("mpg_hwy"))
                    .hasStartStop(rs.getBoolean("has_start_stop"))
                    .description(rs.getString("description"))
                    .build();
            // @formatter:on
        }}
```



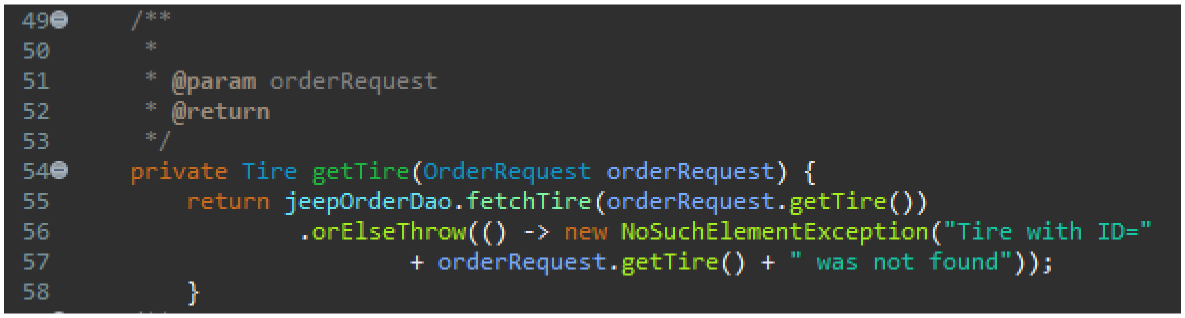
```java
    class ColorResultSetExtracor implements ResultSetExtractor<Color> {
        @Override
        public Color extractData(ResultSet rs)
                throws SQLException {
            rs.next();
            // @formatter:off
            return Color.builder()
                    .color(rs.getString("color"))
                    .colorPK(rs.getLong("color_pk"))
                    .colorId(rs.getString("color_id"))
                    .isExterior(rs.getBoolean("is_exterior"))
                    .price(rs.getBigDecimal("price"))
                    .build();
            // @formatter:on
        }}
```



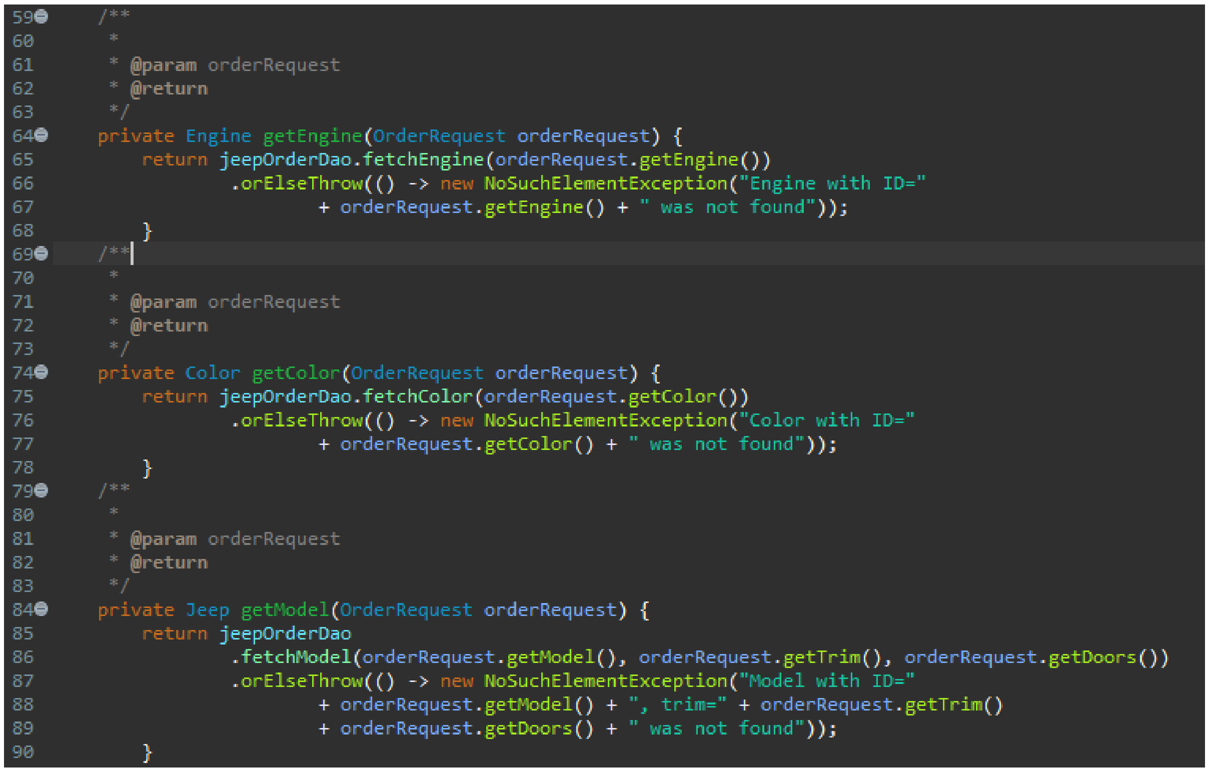
```java
    class ModelResultSetExtracor implements ResultSetExtractor<Jeep> {
        @Override
        public Jeep extractData(ResultSet rs)
                throws SQLException {
            rs.next();
            // @formatter:off
            return Jeep.builder()
                    .modelPK(rs.getLong("model_pk"))
                    .modelId(JeepModel.valueOf(rs.getString("model_id")))
                    .trimLevel(rs.getString("trim_level"))
                    .numDoors(rs.getInt("num_doors"))
                    .wheelSize(rs.getInt("wheel_size"))
                    .basePrice(rs.getBigDecimal("base_price"))
                    .build();
            // @formatter:on
        }}
class SqlParams {
    String sql;
    MapSqlParameterSource source = new MapSqlParameterSource();
    }

```



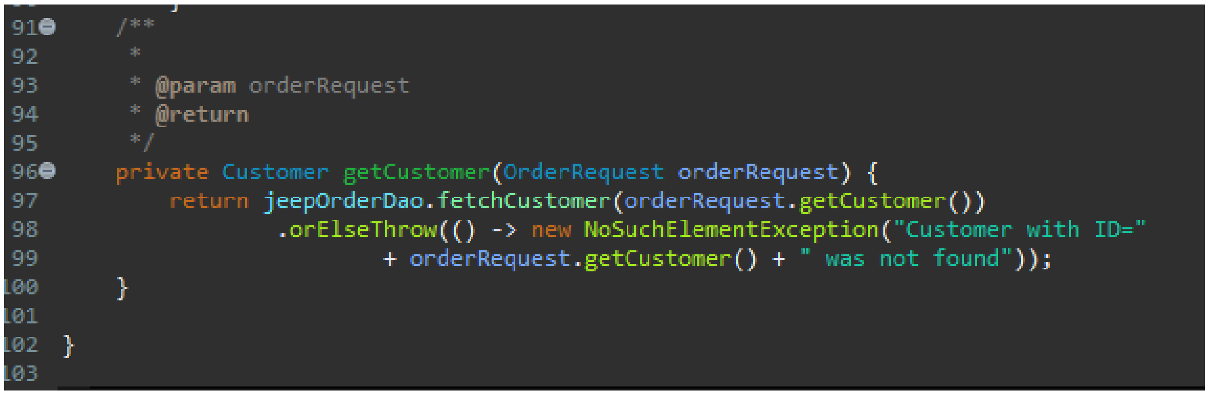
```java
package com.promineotech.jeep.service;

import java.math.BigDecimal;

@Service
public class DefaultJeepOrderService implements JeepOrderService {

    @Autowired
    private JeepOrderDao jeepOrderDao;

    @Transactional
    @Override
    public Order createOrder(OrderRequest orderRequest) {
        Customer customer = getCustomer(orderRequest);
        Jeep jeep = getModel(orderRequest);
        Color color = getColor(orderRequest);
        Engine engine = getEngine(orderRequest);
        Tire tire = getTire(orderRequest);
        List<Option> options= getOption(orderRequest);
        BigDecimal price = jeep.getBasePrice().add(color.getPrice().add(engine.getPrice()).add(tire.getPrice()));

        for(Option option : options) {
            price = price.add(option.getPrice());
        }

        return jeepOrderDao.saveOrder(customer, jeep, color, engine, tire, price, options);
    }

    private List<Option> getOption(OrderRequest orderRequest) {
        return jeepOrderDao.fetchOptions(orderRequest.getOptions());
        }

```



```java
    /**
     *
     * @param orderRequest
     * @return
     */
    private Tire getTire(OrderRequest orderRequest) {
        return jeepOrderDao.fetchTire(orderRequest.getTire())
            .orElseThrow(() -> new NoSuchElementException("Tire with ID="
                    + orderRequest.getTire() + " was not found"));
        }
```

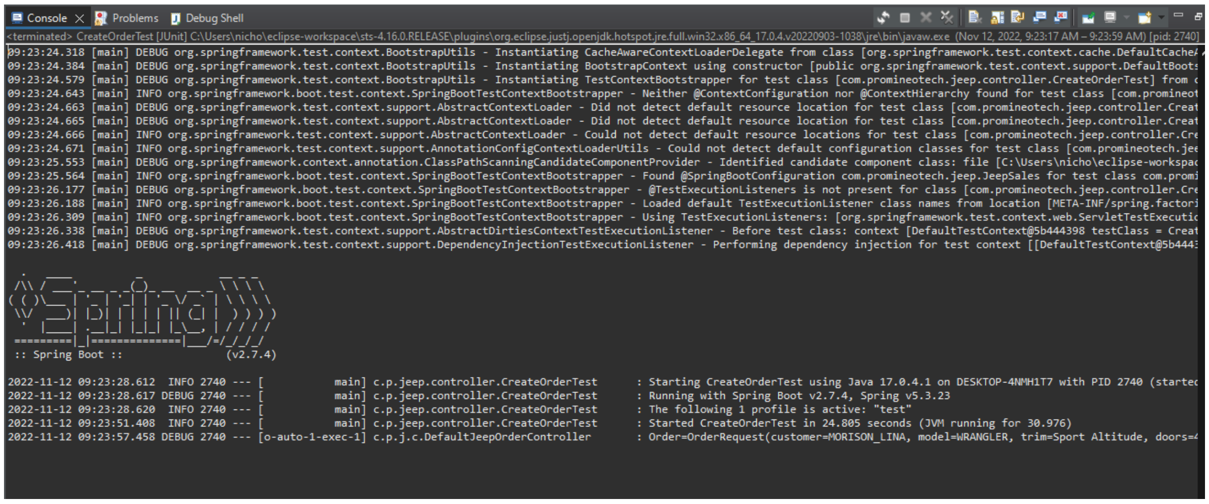


```java
    /**
     *
     * @param orderRequest
     * @return
     */
    private Engine getEngine(OrderRequest orderRequest) {
        return jeepOrderDao.fetchEngine(orderRequest.getEngine())
                .orElseThrow(() -> new NoSuchElementException("Engine with ID="
                        + orderRequest.getEngine() + " was not found"));
        }
    /**
     *
     * @param orderRequest
     * @return
     */
    private Color getColor(OrderRequest orderRequest) {
        return jeepOrderDao.fetchColor(orderRequest.getColor())
                .orElseThrow(() -> new NoSuchElementException("Color with ID="
                        + orderRequest.getColor() + " was not found"));
        }
    /**
     *
     * @param orderRequest
     * @return
     */
    private Jeep getModel(OrderRequest orderRequest) {
        return jeepOrderDao
                .fetchModel(orderRequest.getModel(), orderRequest.getTrim(), orderRequest.getDoors())
                .orElseThrow(() -> new NoSuchElementException("Model with ID="
                        + orderRequest.getModel() + ", trim=" + orderRequest.getTrim()
                        + orderRequest.getDoors() + " was not found"));
        }
```

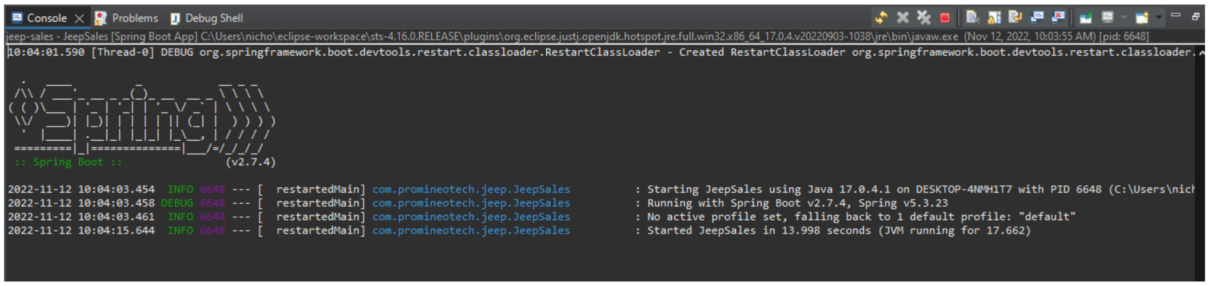


```java
    /**
     *
     * @param orderRequest
     * @return
     */
    private Customer getCustomer(OrderRequest orderRequest) {
        return jeepOrderDao.fetchCustomer(orderRequest.getCustomer())
                .orElseThrow(() -> new NoSuchElementException("Customer with ID="
                        + orderRequest.getCustomer() + " was not found"));
    }

}
```

Screenshots of Running Application:



```
<terminated> CreateOrderTest [JUnit] C:\Users\nicho\eclipse-workspace\sts-4.16.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Nov 12, 2022, 9:23:17 AM – 9:23:59 AM) [pid: 2740]
09:23:24.318 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.cache.DefaultCacheA
09:23:24.384 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.support.DefaultBoots
09:23:24.579 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [com.promineotech.jeep.controller.CreateOrderTest] from c
09:23:24.643 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test class [com.promineot
09:23:24.663 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeep.controller.Creat
09:23:24.665 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [com.promineotech.jeep.controller.Creat
09:23:24.666 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [com.promineotech.jeep.controller.Cre
09:23:24.671 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [com.promineotech.jee
09:23:25.553 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class: file [C:\Users\nicho\eclipse-workspac
09:23:25.564 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration com.promineotech.jeep.JeepSales for test class com.promi
09:23:26.177 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class [com.promineotech.jeep.controller.Cre
09:23:26.188 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from location [META-INF/spring.factori
09:23:26.309 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutic
09:23:26.338 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test class: context [DefaultTestContext@5b444398 testClass = Creat
09:23:26.418 [main] DEBUG org.springframework.test.context.support.DependencyInjectionTestExecutionListener - Performing dependency injection for test context [[DefaultTestContext@5b4443

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.4)

2022-11-12 09:23:28.612  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Starting CreateOrderTest using Java 17.0.4.1 on DESKTOP-4NMH1T7 with PID 2740 (startec
2022-11-12 09:23:28.617 DEBUG 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Running with Spring Boot v2.7.4, Spring v5.3.23
2022-11-12 09:23:28.620  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : The following 1 profile is active: "test"
2022-11-12 09:23:51.408  INFO 2740 --- [           main] c.p.jeep.controller.CreateOrderTest      : Started CreateOrderTest in 24.805 seconds (JVM running for 30.976)
2022-11-12 09:23:57.458 DEBUG 2740 --- [o-auto-1-exec-1] c.p.j.c.DefaultJeepOrderController       : Order=OrderRequest(customer=MORISON_LINA, model=WRANGLER, trim=Sport Altitude, doors=4
```



```
jeep-sales - JeepSales [Spring Boot App] C:\Users\nicho\eclipse-workspace\sts-4.16.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Nov 12, 2022, 10:03:55 AM) [pid: 6648]
10:04:01.590 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader.

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.4)

2022-11-12 10:04:03.454  INFO 6648 --- [  restartedMain] com.promineotech.jeep.JeepSales          : Starting JeepSales using Java 17.0.4.1 on DESKTOP-4NMH1T7 with PID 6648 (C:\Users\nick
2022-11-12 10:04:03.458 DEBUG 6648 --- [  restartedMain] com.promineotech.jeep.JeepSales          : Running with Spring Boot v2.7.4, Spring v5.3.23
2022-11-12 10:04:03.461  INFO 6648 --- [  restartedMain] com.promineotech.jeep.JeepSales          : No active profile set, falling back to 1 default profile: "default"
2022-11-12 10:04:15.644  INFO 6648 --- [  restartedMain] com.promineotech.jeep.JeepSales          : Started JeepSales in 13.998 seconds (JVM running for 17.662)
```

URL to GitHub Repository:

nichspragg/Spring-Boot-Project (github.com)
https://github.com/nichspragg/Spring-Boot-Project