Continuous Build and Delivery

Testing Assessment Sample -15%

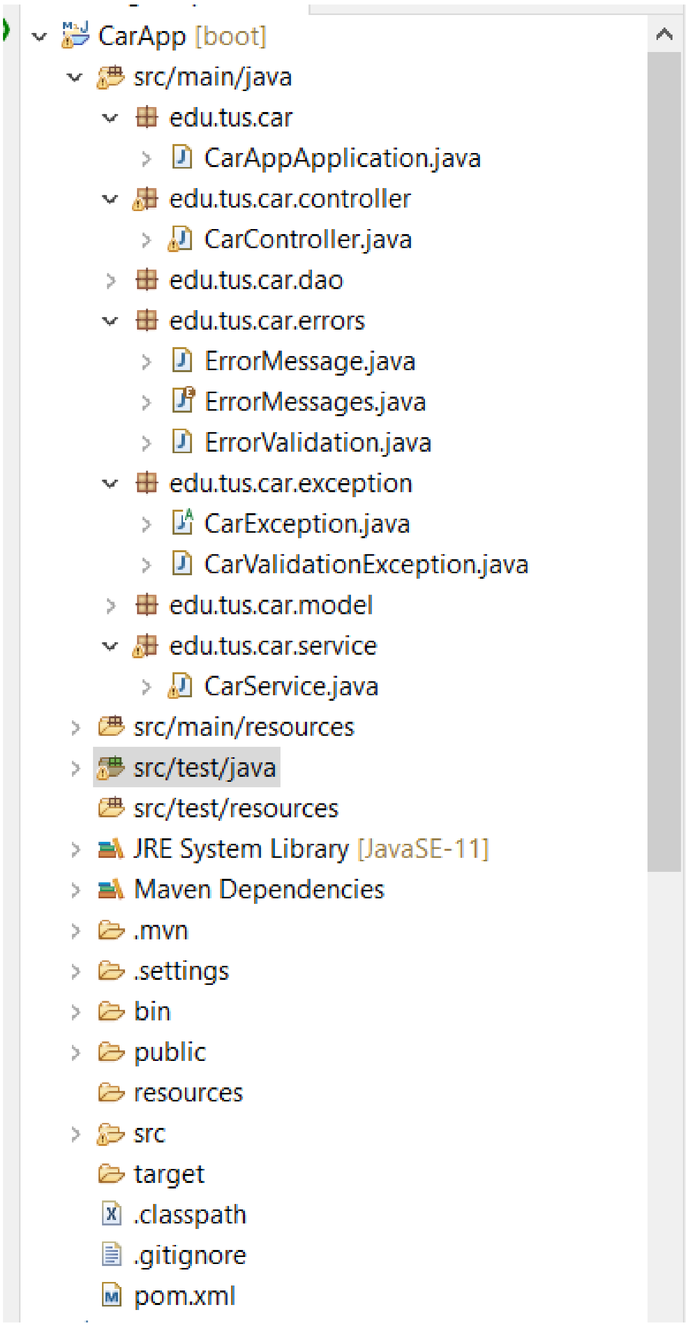
Duration – 2 hours

Application

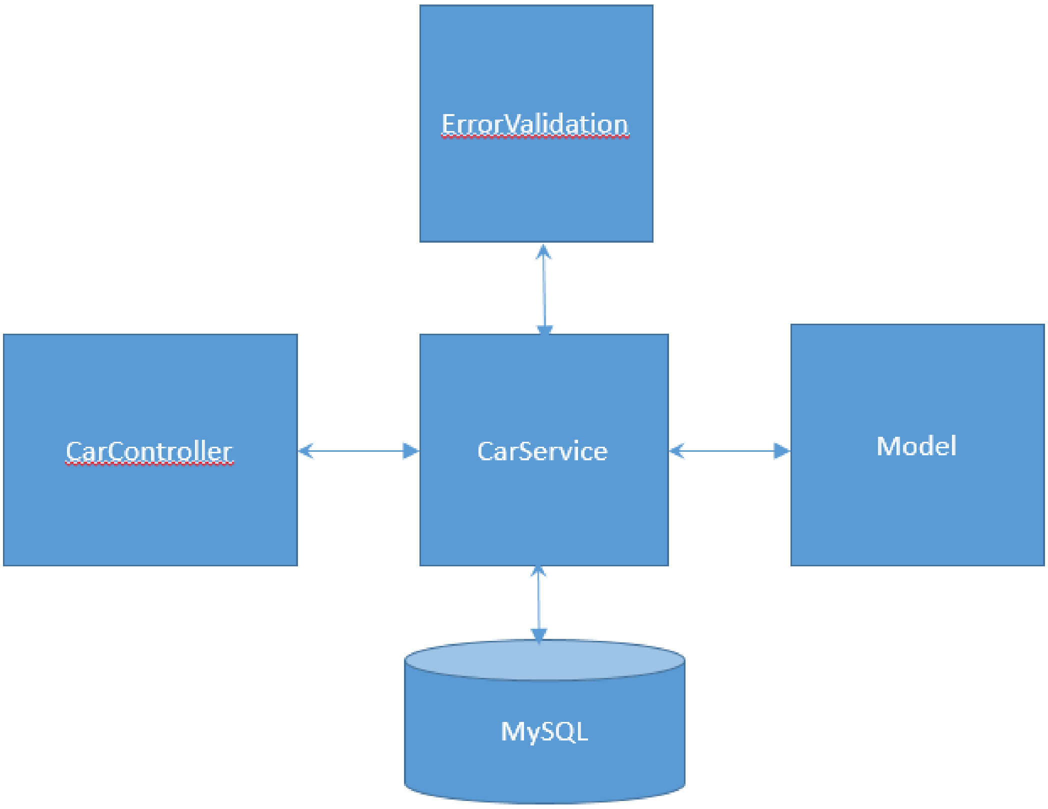
You are given the CarApp code. Import the maven project into your workspace. Don't make any changes to the code you have been given.

This exam is open book. All communication/social media apps are prohibited.

Upload your test code to Moodle placeholder (CarAppIT.java, CarrControllerTest and ErrorValidationTest)



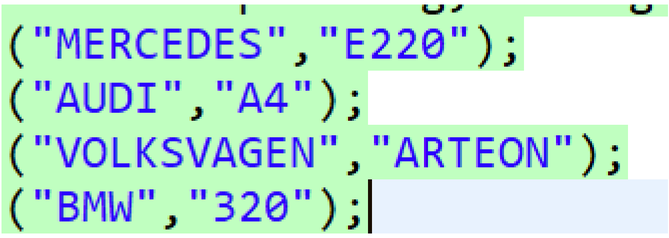
```
CarApp [boot]
  src/main/java
    edu.tus.car
      CarAppApplication.java
    edu.tus.car.controller
      CarController.java
    edu.tus.car.dao
    edu.tus.car.errors
      ErrorMessage.java
      ErrorMessages.java
      ErrorValidation.java
    edu.tus.car.exception
      CarException.java
      CarValidationException.java
    edu.tus.car.model
    edu.tus.car.service
      CarService.java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-11]
  Maven Dependencies
  .mvn
  .settings
  bin
  public
  resources
  src
  target
  .classpath
  .gitignore
  pom.xml
```



The following User Story is implemented:-

As an administrator I want to add a car to the car management application so that I can offer it for sale.

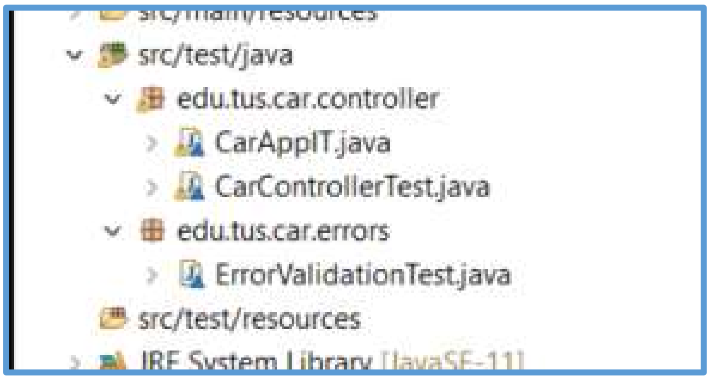
1. Success – car added with fields make, model, year and color. Status code 200 returned with car details.
2. Fail – only certain makes and models are allowed Status code 400 returned with error message for all others. The following 4 combinations of make/model are allowed.



```
("MERCEDES","E220");
("AUDI","A4");
("VOLKSVAGEN","ARTEON");
("BMW","320");
```

3. Cars older than 2020 are not accepted. Status code 400 returned with error message

Complete the following tests:



```
  src/main/resources
  src/test/java
    edu.tus.car.controller
      CarAppIT.java
      CarControllerTest.java
    edu.tus.car.errors
      ErrorValidationTest.java
  src/test/resources
  JRE System Library [JavaSE-11]
```

Step 1: Complete the tests in ErrorValidationTest that tests the methods in the ErrorValidation class. These tests can run outside of the Springboot context.

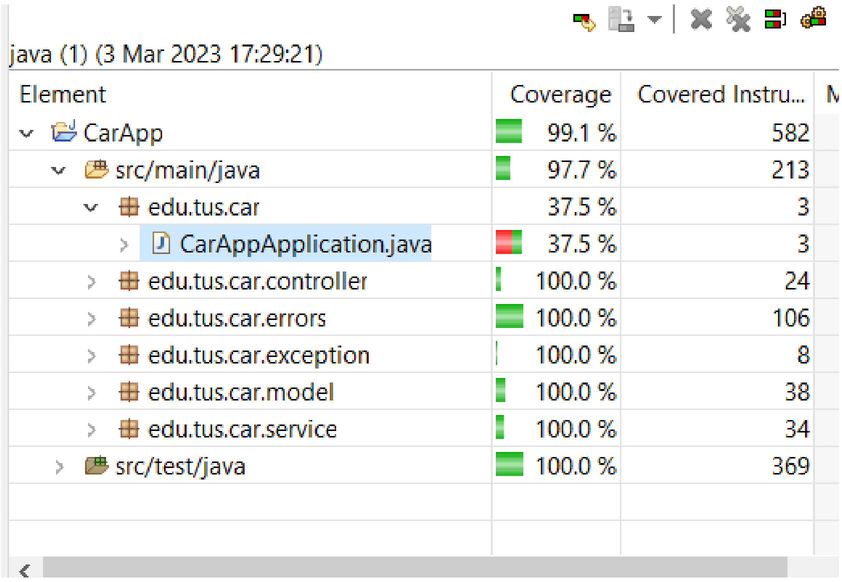
The “buildCar” method is given to create car objects if you wish to use it.

[20 marks]

Step 2: Complete the tests in CarControllerTest that the controller layer and mocks the Service layer. [40 marks]

Step 3: Complete the tests in CarAppIT. These are integration tests using TestRestTemplate. [40 marks]

When you run all the tests you should get code coverage as shown.



java (1) (3 Mar 2023 17:29:21)

Element	Coverage	Covered Instru...	M
CarApp	99.1 %	582	
src/main/java	97.7 %	213	
edu.tus.car	37.5 %	3	
CarAppApplication.java	37.5 %	3	
edu.tus.car.controller	100.0 %	24	
edu.tus.car.errors	100.0 %	106	
edu.tus.car.exception	100.0 %	8	
edu.tus.car.model	100.0 %	38	
edu.tus.car.service	100.0 %	34	
src/test/java	100.0 %	369	