Miracle Riders

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

[***Click here to download the code skeleton***](https://cognizant.tekstac.com/pluginfile.php/69145/mod_vpl/intro/Miracle%20Riders.zip)

***Miracle Riders***is a two wheeler showroom which displays different brand bikes. They had automated the Vehicle registration system, using that application you can maintain information of the vehicles registered by their customers.

Ethen has developed an application for the above purpose. The details of the various functions supported by the system are provided in this case study.

You are required to write Junit test case and check the correctness of the application developed.

**Functional Requirements:**

The application has the below classes and methods implemented.

You are provided with a model class VehicleSpecifications

**Component Specification:**VehicleSpecifications**(Model Class)**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| VehicleSpecifications | String vehicleCode  String modelNumber  String modelColour  Date registrationDate  Date dateOfDelievery  double insuranceAmount  Date insuranceDate  double vehiclePrice | Necessary getters,setters are provided  A Constructor is also provided |

·         Here, modelColour can take a value “Elephant Grey” or “White” or “Ivory” or “Metallic Grey” or “Red” [Note: Values are case insensitive]

**Component Specification:**InvalidVehicleSpecException**(This class inherits the Exception Class)**

|  |  |
| --- | --- |
| **Type (Class)** | **Methods** |
| InvalidVehicleSpecException | Provided with a single argument constructor – InvalidVehicleSpecException (String message) |

The below are the requirements implemented in the Utility class for which JUnit test cases are to be written and tested.

**Component Specification:**VehicleRegistration**(Utility Class)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the vehicle code | VehicleRegistration | public boolean validateVehicleCode(String vehicleCode) | Validate the vehicleCode.  If valid, return true, else this method should throw a user defined exception | Throw a user defined exception “InvalidVehicleSpecException” if the vehicle code does not contain 4 letters followed by 4 digits followed by the letter M twice. |
| View Vehicle based on model Number | VehicleRegistration | public VehicleSpecifications viewVehicleByModelNumber(List<VehicleSpecifications> vehicleList, String modelNumber) | This method should return the VehicleSpecifications object with the modelNumber passed as parameter from list of vehicles, which is also passed as parameter.  If the vehicleList is empty or if there is no vehicle with the given modelNumber it should throw a user defined exception | Throw a user defined exception “InvalidVehicleSpecException” if the vehicleList is empty or if no vehicle exists with the given modelNumber. |
| View Vehicle based on the Registration date | VehicleRegistration | public List<VehicleSpecifications> viewVehicleByRegistrationDate(List<VehicleSpecifications> vehicleList, Date date) | This method should return the list VehicleSpecifications objects with the registration date passed as parameter from list of vehicles, which is also passed as parameter.  If the vehicleList is empty or if there is no vehicle with the given registration date it should throw a user defined exception | Throw a user defined exception “InvalidVehicleSpecException” if the vehicleList is empty |
| Calculate the insurance for the given Insurance date | VehicleRegistration | public double calculateInsuranceAmountByInsuranceDate (List<VehicleSpecifications> vehicleList, Date date) | This method takes the vehicleList and a date as arguments. It should calculate and return insurance amount for that particular date. If the vehicleList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidVehicleSpecException” if the vehicleList is empty |
| View all the vehicles based on delivery date | VehicleRegistration | public Map<Date, List<VehicleSpecifications>> viewVehiclesByDeliveryDate(List<VehicleSpecifications> vehicleList) | This method takes the vehicleList as argument. It should return the list of vehicle objects being delivered on the particular date. The return type is map, where the Delivery date is key and value is the List of vehicle objects to be delivered on that date.  If the vehicleList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidVehicleSpecException” if the vehicleList is empty. |
| View the number of vehicles for the given model number | VehicleRegistration | public int countOfVehiclesByModelNumber(List<VehicleSpecifications> vehicleList, String modelNumber) | This method should return the number of vehicles based on the given model number. It takes the vehicleList, model number as arguments and returns the count of vehicles based on the model number. If the vehicleList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidVehicleSpecException” if the  vehicleList is empty. |

You need to write Junit test for the VehicleRegistration class.

**Testing Scenarios:**

You are provided with a class “VehicleRegistrationTest” to do this testing.

**Note:**

To perform testing, the vehicleList should contain objects of VehicleSpecifications.

To do this, in VehicleRegistrationTest class you are provided with a setup method.  Use this method to populate the static variable vehicleList in VehicleRegistrationTest class.  That is, create few objects for VehicleSpecifications and populate the vehicleList given in VehicleRegistrationTest class with these objects and use that list to test the methods in VehicleRegistration class that needs a VehicleSpecifications list to be passed as attribute.

The below are the test methods to be implemented in VehicleRegistrationTest class.

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateVehicleCodeForValidVehicleCode | This method should test the validateVehicleCode method when a valid vehicle code is passed as parameter |
| test12ValidateVehicleCodeForInvalidVehicleCode | This method should test the validateVehicleCode method when an invalid vehicle code is passed as parameter.  validateVehicleCode is expected to throw InvalidVehicleSpecException when vehicle code is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test13ValidateVehicleCodeForInvalidVehicleCodeWithoutMM | This method should test the validateVehicleCode method when invalid vehicle code without the letter M is passed as parameter  validateVehicleCode is expected to throw InvalidVehicleSpecException  when vehicle code is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ValidateVehicleCodeForInvalidVehicleCodeWithLengthGreaterThan10 | This method should test the validateVehicleCode method when invalid vehicle code with String length greater than expected is passed as parameter  validateVehicleCode is expected to throw InvalidVehicleSpecException when vehicle code is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test15ViewVehicleByValidModelNumber | This method should test the correctness of viewVehicleByModelNumber method for an existing modelNumber.  Perform testing for the correctness of the value returned. |
| test16ViewVehicleByInvalidModelNumber | This method should test the correctness of viewVehicleByValidModelNumber method for a non-existing modelNumber.  Perform testing for the correctness of the value returned.  ViewVehicleByModelNumber method is expected to throw InvalidVehicleSpecException when modelNumber does not exist.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test17ViewVehicleByRegistrationDate | This method should test the correctness of viewVehicleByRegistrationDate method.  Perform testing for the correctness of the value returned. |
| test18ViewVehicleByRegistrationDateForEmptyList | This method should test the correctness of viewVehicleByRegistrationDate method for an empty vehicleList.  viewVehicleByRegistrationDate method is expected to throw InvalidVehicleSpecException when vehicleList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test19CalculateInsuranceAmountByInsuranceDate | This method should test the correctness of calculateInsuranceAmountByInsuranceDate method.  Perform testing for the correctness of the value returned. |
| test20CalculateInsuranceAmountByInsuranceDateForEmptyList | This method should test the correctness of calculateInsuranceAmountByInsuranceDate method for an empty vehicleList.  calculateInsuranceAmountByInsuranceDate method is expected to throw InvalidVehicleSpecException when vehicleList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test21ViewVehiclesByDeliveryDate | This method should test the correctness of viewVehiclesByDeliveryDate method.  Perform testing for the correctness of the value returned. |
| test22ViewVehiclesByDeliveryDateForEmptyList | This method should test the correctness of viewVehiclesByDeliveryDate method for an empty vehicleList.  viewVehiclesByDeliveryDate method is expected to throw InvalidVehicleSpecException when vehicleList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test23CountOfVehiclesByModelNumber | This method should test the correctness of countOfVehiclesByModelNumber method.  Perform testing for the correctness of the value returned. |
| test24CountOfVehiclesByModelNumberForEmptyList | This method should test the correctness of countOfVehiclesByModelNumber method for an empty vehicleList.  countOfVehiclesByModelNumber method is expected to throw InvalidVehicleSpecException when vehicleList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |

Implement the test methods and provide the needed annotation to all the methods in VehicleRegistrationTest class.

Also, this class is provided with the annotation, so that the test methods are executed in ascending order of the test method names.

You are provided with a Main class with the main method to check the correctness of the test methods written in VehicleRegistrationTest class.

Having completed writing the test methods, uncomment the code in Main class and execute the main method.