Mobile Connection

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

***Mobile Connection Management***

[*Click here to download the code skeleton*](https://cognizant.tekstac.com/pluginfile.php/69138/mod_vpl/intro/Mobile%20Connection.zip)

Raghu does a statistical study of the Mobile connections taken by the public by taking a random data. The study is based on the Service provider and the related plan that is opted by the public.To do this he has developed  an automated application.

He has developed an application for taking various reports based on few mobile connections. 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 Mobile

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| Mobile | String mobileNumber  String ownerName  String serviceProvider;  String connectionType  String plan | Necessary getters and setters are provided.  A constructor is also provided. |

Here the **connectionType** can take a value either “Prepaid “ or “Postpaid” or “DTH”

[Note: Values are case insensitive].

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

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

You are also provided with a utility class MobileService and the few business requirements  implemented in it.

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | Type (Class) | **Methods** | **Responsibilities** | **Exception** |
| Validating the connection type.. | MobileService | public boolean validateConnectionType( String connectionType) | Validate the connectionType..  If valid return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidMobileServiceException”  if the connectionType Is neither “Prepaid” nor “Postpaid” nor “DTH” |
| View Mobile by mobile number | MobileService | public Mobile viewMobileByMobileNumber(List<Mobile> mobileList,String mobileNumber) | This method should return the Mobile object with the mobileNumber passed as parameter from mobileList.  If the mobileList is empty or if there is no Mobile with  the given mobile number it should throw a user defined exception | Throw a user defined exception “InvalidMobileServiceException” if the  mobileList  is empty or if there is  no Mobile  in the given mobile number. |
| View the list of Mobiles for a given connection type | MobileService | public List<Mobile> viewMobilesByConnectionType(List<Mobile> mobileList,String connectionType) | This method takes the connectionType as argument. It should return the list of Mobiles for the  given connection type. If the mobileList  is empty it should throw a user defined exception. | Throw a user defined exception “InvalidMobileServiceException” if the mobileList  is empty. |
| View connection type wise EBConnection  details | MobileService | public Map<String,List<Mobile>> viewMobilesServiceProviderWise(List<Mobile> mobileList) | This method should return serviceProvider wise list of Mobiles The return type is map, where the serviceProvider  is the key and value is the List of Mobile pertaining to that serviceProvider. If the mobileList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidMobileServiceException” if the mobileList  is empty. |
| View the number of EBconnection for each phase | MobileService | public  Map<String,Integer> countTotalConnectionForEachPlan(List<Mobile> mobileList) | This method should return the number of Mobile object for each plan based on the mobilrList. It returns a Map with key as plan  and value as count of mobile objects in that plan..  If the mobileList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidMobileServiceException” if the mobileList  is empty. |

You need to write Junit test for the MobileService  class.

**Testing Scenarios:**

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

**Note:**

To perform testing the mobileList should contain objects of Mobile.

To do this, in MobileServiceTest  class you are provided with a setup method. Use this method to populate the static variable mobileList in MobileServiceTest  class.

That is, create few objects for  Mobile and populate the mobileList given in MobileServiceTest class with these objects and use that list to test the methods  in MobileService class that needs a Mobile list to be passed as parameter.

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateConnectionTypeWhenPrepaid | This method should test the validateConnectionType method when “Prepaid” is passed as parameter |
| test12ValidateConnectionTypeWhenPostpaid | This method should test the validateConnectionType  method when “Postpaid” is passed as parameter. |
| test13ValidateConnectionTypeWhenDTH | This method should test the validateConnectionType method when “DTH” is passed as parameter. |
| test14ValidateConnectionTypeWhenInvalid | This method should test the validateConnectionType method when invalid value is passed as parameter  validateConnectionType is expected  to throw InvalidMobileServiceException when type is invalid.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block. |
| test15ViewMobileByMobileNumberWhenValid | This method should test the correctness of  viewMobileByMobileNumber method for  an existing mobile number .  Perform testing for the correctness of the value returned. |
| test16ViewMobileByMobileNumberWhenInvalid | This method should test the correctness of  viewMobileByMobileNumber method for a non existing mobile number.  Perform testing for the correctness of the value returned.  viewMobileByMobileNumber method is expected  to throw InvalidMobileServiceException when mobile number does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test17ViewMobilesByConnectionType | This method should test the correctness of  viewMobilesByConnectionType method.  Perform testing for the correctness of the value returned. |
| test18ViewMobilesByConnectionTypeForEmptyList | This method should test the correctness of viewMobilesByConnectionType method for an empty mobileList.  viewMobilesByConnectionType method is expected  to throw InvalidMobileServiceException when mobileList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test19ViewMobilesServiceProviderWise | This method should test the correctness of  viewMobilesServiceProviderWise method.  Perform testing for the correctness of the value returned. |
| test20ViewMobilesServiceProviderWiseForEmptyList | This method should test the correctness of  viewMobilesServiceProviderWise method for an empty mobileList.  viewMobilesServiceProviderWise method is expected  to throw InvalidMobileServiceException when mobileList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test21CountTotalConnectionForEachPlan | This method should test the correctness of countTotalConnectionForEachPlan method.  Perform testing for the correctness of the value returned. |
| test22CountTotalConnectionForEachPhaseForEmptyList | This method should test the correctness of countTotalConnectionForEachPlan method for an empty mobileList.  countTotalConnectionForEachPlan method is expected  to throw InvalidMobileServiceException when mobileList 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 MobileServiceTest  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 MobileServiceTest  class.

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