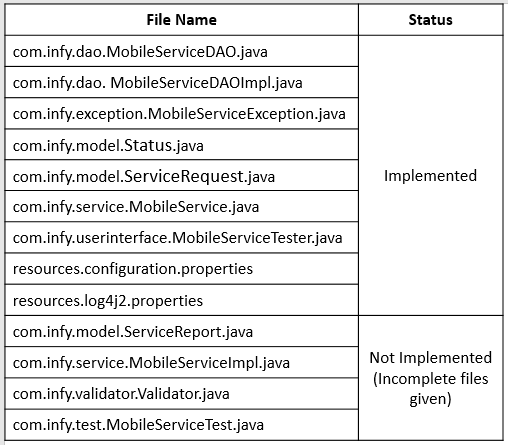
**MobileService**is an online application which a customer can use to send a service request for his mobile phone if there are any issues and get it fixed. Please implement it.

* Submit a request to the service center
* See all the previous requests based on their status

**Artifacts:**



**MobileServiceTester**is an interface between the end-user and the application. Please run the application from here.

There are two modules to this application:

**Module 1:** Submit a request to the service center, In this module the user will have to submit the details for their mobile phones along with the issue they have to the application. Based on the issue, the request maybe accepted or rejected.

**Inputs**: **ServiceRequest**object.

For a valid **ServiceRequest = { brand : "Abc", issues: ["Battery"], iMEINumber: 1234567890123456L, contactNumber: 9876543210L,customerName: "Jim" },**output should be displayed as below,



For invalid inputs,

* If **brandName**is **twoplusthree,**the output should be displayed as,



* If the **issues**is an empty list, the output should be displayed as,



* If the **contactNumber**is **98765**, the output should be displayed as



* If the **iMEINumber**is **1234567890**, the output should be displayed as,



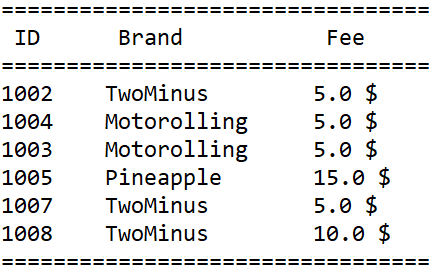
* If the **customerName**is **jim**, the output should be displayed as,



**Module 2:**View all the previous requests based on their current status.

**Input:** status

**For a valid input:**COMPLETED, the output should be displayed like below.

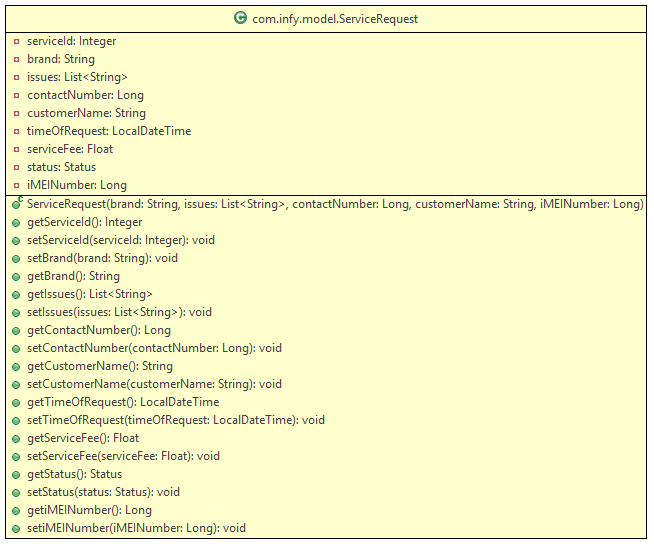


**For invalid input:**REJECTED, the output should be displayed as,

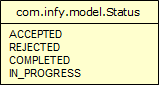


**Model class:**

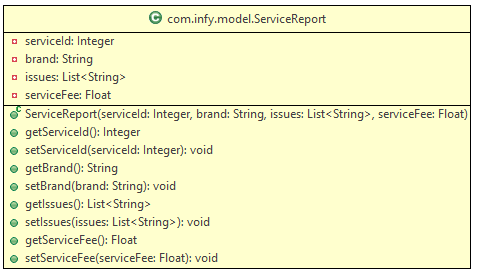
**ServiceRequest (Implemented) :**This is the class that the customer will use to submit a request to the service center.



**Status (Implemented) :** This is an ENUM which will be used to set the value of status attribute in the **StatusRequest**class. It contains **ACCEPTED, REJECTED, COMPLETED, IN\_PROGRESS**as values.



**ServiceReport (To be implemented) :**Please implement this class according the class diagram provided below.



**Exceptions**

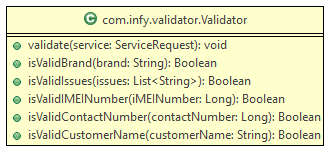
**MobileServiceException** - (Implemented) : This is a user-defined Exception used to throw an exception in this application



**MobileServiceException**(String message):

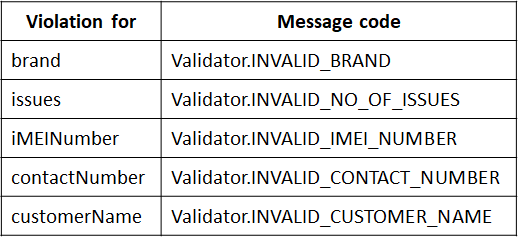
This is a constructor for MobileServiceException class with a message as a parameter.

**Validator:**This class will be used to validate the **ServiceRequest**object received.



**validate(ServiceRequest service):**

* This method is used to validate the **ServiceRequest**object received.
* Invoke the methods and throw **MobileServiceException** with the error messages given below.



* Log all **MobileServiceException** and re-throw all the **MobileServiceException**.

**isValidBrand(String brand):**

* This method is used to validate the **brand** received.
* The **brand**should start with an uppercase alphabet and followed by one or more alphabets.
* The minimum length is 2.
* If the above condition is satisfied, it should return **true**else it should return **false.**
* Valid values:
  + Abc, AB, AbcDEfgh
* Invalid values:
  + A, abc, a

**isValidIssues(List<String> issues):**

* This method is used to validate the **issues** required.
* **issues** received should not be equal to **null** or just an **empty list**.
* If the above conditions are satisfied, it should return **true**, else it should return **false**.
* Valid values:
  + ["Battery"]
* Invalid values:
  + [ ]
  + null

**isValidIMEINumber(Long iMEINumber):**

* This method is used to validate the **iMEINumber**received.
* The **iMEINumber**length should be equal to 16.
* If the above condition is satisfied, it should return **true**else it should return **false**.
* Valid values:
  + 1234567890123456L
* Invalid values:
  + 123455L
  + 908070605040L

**isValidContactNumber(Long contactNumber):**

* This method is used to validate the **contactNumber** received.
* The contactNumber length should be equal to 10 and all the ten digits should not be the same.
* If the above conditions are satisfied, it should return **true**else it should return **false**.
* Valid values:
  + 9876543210L
  + 9080706050L
* Invalid values:
  + 1111111111L
  + 98765L

**isValidCustomerName(String customerName):**

* This method validates the **customerName** received.
* The **customerName**should contain at least one word. It can contain more than one word, each word separated by a single space.
* Each word in the **customerName**should start with a uppercase alphabet followed by lowercase alphabets.
* If the above conditions are satisfied, it should return **true,**else it should return **false**.
* Valid values:
  + Abc
  + Abc Def
  + Abc Def Ghi
* Invalid values:
  + abc
  + abc   def
  + abcdef
  + abc Def

**Service Layer:**This is the layer where the logic for accepting requests, calculating the service fee, and generating a report for the previous request is written.

**MobileServiceImpl**



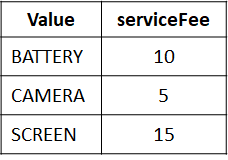
Initialize a new instance of **MobileServiceDAOImpl**and **Validator**class for the private attribute.

**registerRequest(ServiceRequest service):**

* This method is used to validate the **ServiceRequest**object received and add the request to the database based on the request status.
* Invoke the **validate()** of the **Validator**class by passing the **ServiceRequest** object as the parameter.
* If the **ServiceRequest** object is valid, invoke the **calculateEstimateCost()**method bypassing the **issues**attribute from the **ServiceRequest**object as the argument which will return a **Float**value.
* If the **Float** value returned in the previous step is less than or equal to **0,** throw a new **MobileServiceException** with the message "**Service.INVALID\_ISSUES**"
* Else, set the value of **serviceFee** in the **ServiceRequest**with the **Float** value.
* Set the values of **status** to **ACCEPTED** and **timeOfRequest**to the current date and time value. (Use LocalDateTime).
* Invoke the **registerRequest()**of the **MobileServiceDAOImpl**class by passing the **ServiceRequest** object as the argument which will return another **ServiceRequest**object.
* Return the **ServiceRequest**object retrieved in the previous step.
* Catch and log all **MobileServiceException** from the Service class and re-throw all the **MobileServiceException.**

**calculateEstimateCost(List<String> issues):**

* This method is used to calculate the **serviceFee**from the issues.
* Iterate through the list of **issues** and calculate the total serviceFee based on the issue. (Do case insensitive comparison)



* Return the sum of all values.
* If none of the issues match, return 0.

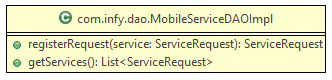
**getServices(Status status):**

* This method is used to fetch all the **ServiceReport**objects from the database which have the required status.
* Initialize a new **List<ServiceReport>** object.
* Invoke the **getServices()**of the **MobileServiceDAOImpl**class which will return a **List<ServiceRequest>.**
* Filter the **List<ServiceRequst>**based on the status and populate the **List<ServiceReport>.**
* If the **List<ServiceReport>**is empty, throw a new **MobileServiceException**with the message "**Service.NO\_RECORDS\_FOUND**".
* Else, return the **List<ServiceReport>**object.
* Catch and log all **MobileServiceException** from the Service class and re-throw all the **MobileServiceException**

**Note:**Use Streams and lambda functions to implement the **getServices()** method.

**DAO Layer:**This layer will be be used to access the database but we are statically storing the data.

**MobileServiceDAOImpl:**This class contains the method required to connect to the database and

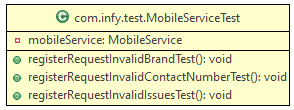


**registerRequest(ServiceRequest service):** This method receives the **ServiceRequest**object, sets the **serviceId**attribute, and returns the same **ServiceRequest**object.

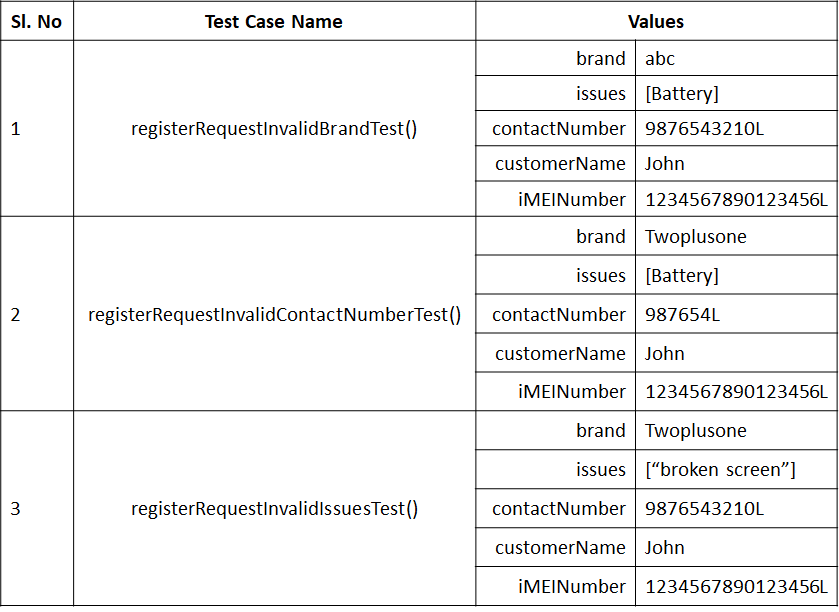
**getServices():**This method is used to return a list of all the **ServiceRequest**objects stored.

**JUNIT:**

**MobileServiceTest:**This class is used to test the methods given in **MobileServiceImpl**class.



Please use the table provided below for input for each test method.



**Note:**For the third test method, check for the MobileServiceException thrown in the service class and not the validator class.

**Note**: Check the project using SonarLint to maintain the coding standards.