Lucky Chill AC Showroom

**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/69141/mod_vpl/intro/ACShowRoom.zip)

***Lucky Chill***is an AC showroom which displays different brand ACs for sale, they wanted to automate the AC details. Using the application, you can maintain information of the ACs, where information like the AC model number, type of AC, brand name, colour, date of booking and its cost.

Vimal 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 ACSpecifications

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| ACSpecifications | String modelNumber  String acType  String brandName  String acColour  Date dateOfBooking  double acCost | Necessary getters and setters are provided.  A constructor is also provided. |

Here the **acType** can take a value “Central Air Conditioner” or “Ductless Mini-Split” or “Window Air Conditioner” or “Portable Air Conditioner” or “Dual Fuel Air Conditioner” or “Smart Air Conditioner”.

The **brandName** can take a value “Lennox” or “Carrier” or “Samsung”.

The **acColour** can take a value “Red” or “White” or “Black”.

[Note: Values are case insensitive].

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

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

You are also provided with a utility class ACBooking 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:**ACBooking**(Utility Class)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | Type (Class) | **Methods** | **Responsibilities** | **Exception** |
| Validating the model number. | ACBooking | public boolean validateModelNumber(String modelNumber) | Validate the modelNumber.  If valid return true else this method should throw a user defined exception | Throw a user defined exception “InvalidACSpecException” if the modelNumber does not contain the String “LUCH” followed by 5 digits in it. |
| View AC by Model number | ACBooking | public ACSpecifications viewACByModelNumber(List<ACSpecifications> acList, String modelNumber) | This method should return the AC object with the modelNumber passed as parameter from acList.  If the acList is empty or if there is no AC with the given model number it should throw a user defined exception | Throw a user defined exception “InvalidACSpecException” if the acList is empty or if there is no AC in the given model number. |
| View the list of ACs for a given types | ACBooking | public List<ACSpecifications> viewACByItsType(List<ACSpecifications> acList, String actype) | This method takes the acType as argument. It should return the list of ACs for the given type. If the acList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidACSpecException” if the acList is empty. |
| Calculate the total amount based on the date of booking | ACBooking | public double calculateTotalAmountByDateOfBooking(List<ACSpecifications> acList, Date dateOfBooking) | This method should return total amount that has to be received on the particular date (date of booking). If the acList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidACSpecException” if the acList is empty. |
| View the number of ACs for each brand | ACBooking | public Map<String, List<ACSpecifications>> viewACBrandNameWise (List<ACSpecifications> acList) | This method should return the number of AC objects for each brand based on the acList. It returns a Map with key as brand and value as list of AC objects in that brand.  If the acList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidACSpecException” if the acList is empty. |
| View the number of ACs by its colour | ACBooking | public int countOfACByACColour(List<ACSpecifications> acList, String colour) | This method should return total count of ACs based on its colour. If the acList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidACSpecException” if the acList is empty. |

You need to write Junit test for the ACBooking class.

**Testing Scenarios:**

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

**Note:**

To perform testing, the acList should contain objects of ACs.

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateModelNumber | This method should test the validateModelNumber method when a valid model number is passed as parameter |
| test12ValidateModelNumberWithoutStringLUCH | This method should test the validateModelNumber method when an invalid model number without the String “LUCH” is passed as parameter.  validateModelNumber is expected to throw InvalidModelNumberException when model number is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test13ValidateModelNumberWithMoreDigits | This method should test the validateModelNumber method when an invalid model with more digits than specified is passed as parameter.  validateModelNumber is expected to throw InvalidModelNumberException when model number is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ValidateModelNumberWithLessDigits | This method should test the validateModelNumber method when an invalid model with less digits than specified is passed as parameter.  validateModelNumber is expected to throw InvalidModelNumberException when model number is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test15ViewACByValidModelNumber | This method should test the correctness of viewACByModelNumber method for an existing model number.  Perform testing for the correctness of the value returned. |
| test16ViewACByInvalidModelNumber | This method should test the correctness of viewACByModelNumber method for a non-existing model number.  Perform testing for the correctness of the value returned.  viewACByModelNumber method is expected to throw InvalidModelNumberException when model number does not exist.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test17ViewACByItsType | This method should test the correctness of viewACByItsType method.  Perform testing for the correctness of the value returned. |
| test18ViewACByItsTypeForEmptyList | This method should test the correctness of viewACByItsType method for an empty acList.  viewACByItsType method is expected to throw InvalidModelNumberException when acList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test19CalculateTotalAmountByDateOfBooking | This method should test the correctness of calculateTotalAmountByDateOfBooking method.  Perform testing for the correctness of the value returned. |
| test20CalculateTotalAmountByDateOfBookingForEmptyList | This method should test the correctness of calculateTotalAmountByDateOfBooking method for an empty acList.  calculateTotalAmountByDateOfBooking method is expected to throw InvalidModelNumberException when acList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test21ViewACBrandNameWise | This method should test the correctness of viewACBrandNameWise method.  Perform testing for the correctness of the value returned. |
| test22ViewACBrandNameWiseForEmptyList | This method should test the correctness of viewACBrandNameWise  method for an empty acList.  viewACBrandNameWise method is expected to throw InvalidModelNumberException when acList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test23CountOfACsByACColour | This method should test the correctness of countOfACsByACColour method.  Perform testing for the correctness of the value returned. |
| test24CountOfACsByACColourForEmptyList | This method should test the correctness of countOfACsByACColour method for an empty acList.  countOfACsByACColour method is expected to throw InvalidModelNumberException when acList 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 ACBookingTest 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 ACBookingTest class.

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