IRCTC Railway Ticketing Service

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

[***Click here to download the code template***](https://cognizant.tekstac.com/pluginfile.php/69143/mod_vpl/intro/IRCTC.zip)

***IRCTC***is a railway ticket booking agency which books ticket for their customers. They had automated the ticket booking system, using that application you can maintain information of the tickets booked by their customers.

Aishwarya 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 RailwayTicket

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

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| RailwayTicket | String ticketId  String coachId  String trainName  Date dateOfDeparture  String seatType  double amount | Necessary getters,setters are provided  A Constructor is also provided |

·         Here, seatType can take a value “Air Conditioned” or “Air-Conditioned Chair” or “Sleeper” or “Unreserved” [Note: Values are case insensitive]

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

|  |  |
| --- | --- |
| **Type (Class)** | **Methods** |
| InvalidTicketException | Provided with a single argument constructor – InvalidTicketException(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:**BookingTicket**(Utility Class)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the ticket Id | BookingTicket | public boolean validateTicketId(String ticketId) | Validate the ticketId.  If valid, return true, else this method should throw a user defined exception | Throw a user defined exception “InvalidTicketException” if the ticket Id does not contain 4 letters followed by 4 digits. |
| View Ticket based on ticket Id | BookingTicket | public RailwayTicket viewTicketDetailsByTicketId(List<RailwayTicket> ticketList, String ticketId) | This method should return the Ticket object with the ticket Id passed as parameter from list of tickets, which is also passed as parameter.  If the ticketList is empty or if there is no ticket with the given ticket Id it should throw a user defined exception | Throw a user defined exception “InvalidTicketException” if the ticketList is empty or if no ticket exists with the given ticket Id. |
| View the list of tickets for a given Coach Id | BookingTicket | public List<RailwayTicket> viewTicketByCoachId(List<RailwayTicket> ticketList, String coachId) | This method takes the ticketList and coach Id as arguments. It should return the list of tickets for the given coach Id. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTicketException” if the ticketList is empty |
| Count the number of tickets based on train name | BookingTicket | public int countTicketsByTrainName(List<RailwayTicket> ticketList, String trainName) | This method takes the ticketList as argument along with the Train name. It should return the count of tickets based on the train name.  If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTicketException” if the ticketList is empty |
| View the tickets for each seat type | BookingTicket | public Map<String, List<RailwayTicket>> viewPassengersBySeatType(List<RailwayTicket> ticketList) | This method should return the list of tickets for each seat type in the ticketList. It takes the ticketList as argument and returns a Map with key as seat type and value as list of tickets according to the seat types. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTicketException” if the ticketList is empty |
| Calculate the total amount based on the departure date | BookingTicket | public double calculateAmountByDateOfDeparture(List<RailwayTicket> ticketList, Date dateOfDeparture) | This method takes the ticketList as argument along with the departure date. It should return the amount calculated based on the departure date.  If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTicketException” if the ticketList is empty |

You need to write Junit test for the BookingTicketclass.

**Testing Scenarios:**

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

**Note:**

To perform testing, the ticketList should contain objects of RailwayTicket.

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateTicketIdForValidTicketId | This method should test the validateTicketId method when a valid ticket Id is passed as parameter |
| test12ValidateTicketIdForInvalidTicketIdWithMoreLetters | This method should test the validateTicketId method when an invalid ticket Id is passed as parameter.  validateTicketId method is expected to throw InvalidTicketExceptionwhen ticket Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test13ValidateTicketIdForInvalidTicketIdWithLessLetters | This method should test the validateTicketId method when invalid ticket Id with less letters than specified is passed as parameter  validateTicketId method is expected to throw InvalidTicketExceptionwhen ticket Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ValidateTicketIdForInvalidTicketIdWithMoreDigits | This method should test the validateTicketId method when invalid ticket Id with more digits than specified is passed as parameter  validateTicketId method is expected to throw InvalidTicketExceptionwhen ticket Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test15ValidateTicketIdForInvalidTicketIdWithLessDigits | This method should test the validateTicketId method when invalid ticket Id with less digits than specified is passed as parameter  validateTicketId method is expected to throw InvalidTicketExceptionwhen ticket Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test16ViewTicketDetailsByValidTicketId | This method should test the correctness of viewTicketDetailsByTicketId method for an existing ticket Id.  Perform testing for the correctness of the value returned. |
| test17ViewTicketDetailsByInvalidTicketId | This method should test the correctness of viewTicketDetailsByTicketId method for a non-existing ticket Id.  Perform testing for the correctness of the value returned.  viewTicketDetailsByTicketId method is expected to throw InvalidTicketExceptionwhen ticket Id does not exist.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test18ViewTicketByCoachId | This method should test the correctness of viewTicketByCoachId method.  Perform testing for the correctness of the value returned. |
| test19ViewTicketByCoachIdForEmptyList | This method should test the correctness of viewTicketByCoachId method for an empty ticketList.  viewTicketByCoachId method is expected to throw InvalidTicketExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test20CountTicketsByTrainName | This method should test the correctness of countTicketsByTrainName method.  Perform testing for the correctness of the value returned. |
| test21CountTicketsByTrainNameForEmptyList | This method should test the correctness of countTicketsByTrainName method for an empty ticketList.  countTicketsByTrainName method is expected to throw InvalidTicketExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test22ViewPassengersBySeatType | This method should test the correctness of viewPassengersBySeatType method.  Perform testing for the correctness of the value returned. |
| test23ViewPassengersBySeatTypeForEmptyList | This method should test the correctness of viewPassengersBySeatType method for an empty ticketList.  viewPassengersBySeatType method is expected to throw InvalidTicketExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test24CalculateAmountByDateOfDeparture | This method should test the correctness of calculateAmountByDateOfDeparture method.  Perform testing for the correctness of the value returned. |
| test25CalculateAmountByDateOfDepartureForEmptyList | This method should test the correctness of calculateAmountByDateOfDeparture method for an empty ticketList.  calculateAmountByDateOfDeparture method is expected to throw InvalidTicketExceptionwhen ticketList 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 BookingTicketTest 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 BookingTicketTest class.

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