Aegon Vistara Airline Company

**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/69139/mod_vpl/intro/Aegon%20Airline%20Company.zip)

***Aegon Vistara Airline Company***wanted to automate the ticket details. Using the application, you can maintain information of the ticket, where information like their date of journey and the airline code can be accessed for given ticket Id.

Amy 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 AirlineTicket

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| AirlineTicket | String ticketId  String airlineCode  Date dateOfJourney  String seatType  String source  String destination  double ticketFair | Necessary getters,setters are provided  An overloaded constructor in the class is provided.  Also toString methods are overridden. |

·         Here, seatType can take a value either “Economy” or “Premium” [Note: Values are case insensitive]

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the seat  type | ProcessingTicket | public boolean validateSeatType(String seatType) | Validate the seatType.  If valid, return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidSeatTypeException”  if the seatType Is neither “Economy” or “Premium” |
| View Ticket based on Ticket Id | ProcessingTicket | public AirlineTicket viewTicketDetailsByTicketId(List<AirlineTicket> 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 “InvalidSeatTypeException” if the ticketList is empty or if no ticket exists with the given Ticket Id. |
| View the list of tickets based on the date of journey | ProcessingTicket | public List<AirlineTicket> viewFlightTicketByDateOfJourney(List<AirlineTicket> ticketList, Date journeyDate) | This method takes the ticketList and date of journey as an argument. It should return the list of Tickets for the given date. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidSeatTypeException” if the  ticketList is empty |
| Count of Tickets based on the Airline code | ProcessingTicket | public int countTicketsByAirlineCode(List<AirlineTicket> ticketList, String airlineCode) | This method takes the ticketList and the Airline code as an argument. It should return the count of Tickets for the given Airline code. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidSeatTypeException” if the ticketList is empty. |
| View the Tickets for each seat type | ProcessingTicket | public Map<String, List<AirlineTicket>> countOfPassengersBySeatType(List<AirlineTicket> ticketList) | This method should return the 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 Tickets. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidSeatTypeException” if the ticketList is empty. |
| Calculate amount based on the date of journey | ProcessingTicket | public double calculateAmountByDateOfJourney(List<AirlineTicket> ticketList, Date journeyDate) | This method takes the ticketList and date of journey as arguments. It should calculate the total amount based on the date of journey and return the same. If the ticketList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidSeatTypeException” if the ticketList is empty. |

You need to write Junit test for the ProcessingTicket class.

**Testing Scenarios:**

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

**Note:**

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

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateSeatTypeWhenEconomy | This method should test the validateSeatType method when a seat type is  “Economy” is passed as a parameter |
| test12ValidateSeatTypeWhenPremium | This method should test the validateSeatType method when a seat type is  “Premium” is passed as a parameter |
| test13ValidateTicketIdWhenInvalid | This method should test the validateSeatType method when invalid value is passed as parameter  validateSeatType is expected  to throw InvalidSeatTypeExceptionwhen type is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ViewTicketDetailsByValidTicketId | This method should test the correctness of viewTicketDetailsByTicketId method for an existing Ticket Id.  Perform testing for the correctness of the value returned. |
| test15ViewTicketDetailsByInvalidTicketId | 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 InvalidSeatTypeExceptionwhen Ticket Id does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test16ViewFlightTicketByDateOfJourney | This method should test the correctness of viewFlightTicketByDateOfJourney method.  Perform testing for the correctness of the value returned. |
| test17ViewFlightTicketByDateOfJourneyForEmptyList | This method should test the correctness of viewFlightTicketByDateOfJourney method for an empty ticketList.  viewFlightTicketByDateOfJourney method is expected to throw InvalidSeatTypeExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test18CountTicketsByAirlineCode | This method should test the correctness of countTicketsByAirlineCode method.  Perform testing for the correctness of the value returned. |
| test19CountTicketsByAirlineCodeForEmptyList | This method should test the correctness of countTicketsByAirlineCode method for an empty ticketList.  countTicketsByAirlineCode method is expected  to throw InvalidSeatTypeExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test20CountOfPassengersBySeatType | This method should test the correctness of countOfPassengersBySeatType method.  Perform testing for the correctness of the value returned. |
| test21CountOfPassengersBySeatTypeForEmptyList | This method should test the correctness of countOfPassengersBySeatType method for an empty ticketList.  countOfPassengersBySeatType method is expected  to throw InvalidSeatTypeExceptionwhen ticketList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test22CalculateAmountByDateOfJourney | This method should test the correctness of calculateAmountByDateOfJourney method.  Perform testing for the correctness of the value returned. |
| test23CalculateAmountByDateOfJourneyForEmptyList | This method should test the correctness of calculateAmountByDateOfJourney method for an empty ticketList.  calculateAmountByDateOfJourney method is expected  to throw InvalidSeatTypeExceptionwhen 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 ProcessingTicketTest 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 ProcessingTicketTest class.

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