Pizza Order Management

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

***Pizza Order Management***

[***Click here to download the code skeleton***](https://cognizant.tekstac.com/pluginfile.php/69140/mod_vpl/intro/PizzaOrderManagement.zip)

***Pizza Order Management System***is an automated application for manipulating the various Pizzas ordered by the customers.

They have developed an application for taking various reports based on the Pizzas ordered. 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 PizzaOrder

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| PizzaOrder | int orderId  String pizzaName  String customerName  String pizzaType  String size  int quantity | Necessary getters and setters are provided.  A constructor is also provided. |

Here the **size** can take a value either “Small” or “Medium” or “Large”.

[Note: Values are case insensitive].

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

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

You are also provided with a utility class PizzaShop and the below business requirements are implemented in it for which JUnit test cases are to be written and tested.

You are also provided with an utility class PizzaShop with business methods.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Attributes** | **Methods** |
| PizzaShop | PizzaShop | List<PizzaOrder> pizzaOrderList | Getter and setter for the pizzaOrderList is provided. |

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | Type (Class) | **Methods** | **Responsibilities** | **Exception** |
| Validating the pizza size.. | PizzaShop | public boolean validateSize( String size) | Validate the size..  If valid return true, else this method should throw a user defined exception | Throw a user defined exception “InvalidPizzaOrderException” if the size Is neither “Small” nor “Medium” nor “Large”. |
| View Pizza Order by orderId | PizzaShop | public PizzaOrder viewPizzaOrderById(int orderId) | This method should return the PizzaOrder object with the orderId passed as parameter from pizzaOrderList. If the pizzaOrderList is empty or if there is no PizzaOrder with the given order id it should throw a user defined exception | Throw a user defined exception “InvalidPizzaOrderException” if the orderList is empty or if there is no PizzaOrder in the given order id. |
| View the list of Pizza Order for a given pizza type | PizzaShop | public List<PizzaOrder> viewPizzaOrdersByType(String pizzaType) | This method takes the pizzaType as argument. It should return the list of PizzaOrder for the given pizza type. If the pizzaOrderList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidPizzaOrderException” if the pizzaOrderList is empty. |
| View pizza type wise Pizza Order details | PizzaShop | public Map<String,List<PizzaOrder>> viewPizzaOrdersTypeWise() | This method should return foodType wise Order The return type is map, where the pizzaType is key and value is the List of PizzaOrder pertaining to that pizzaType. If the pizzaOrderList is empty it should throw a user-defined exception. | Throw a user defined exception “InvalidPizzaOrderException” if the pizzaOrderList is empty. |
| View the total quantity for each size | PizzaShop | Public Map<String,Integer> countTotalQuantityForEachSize() | This method should return the total quantity of pizza for each size based on the pizzaOrderList. It returns a Map with key as size and value as the total quantity of pizza. If the pizzaOrderList is empty it should throw a user defined exception. | Throw a user-defined exception “InvalidPizzaOrderException” if the pizzaOrderList is empty. |

You need to write Junit test for the PizzaShop class.

**Testing Scenarios:**

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

**Note:**

To perform testing the pizzaOrderList should contain objects of PizzaOrder.

To do this, in PizzaShopTest class you are provided with a setup method. Use this method to initialize the  pizzaOrderList  attribute in PizzaShop class.

That is, create few objects for PizzaOrder, populate a list with these objects and set the pizzaOrderList  to this list using the setPizzaOrderList method in PizzaShop class.

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateSizeWhenSmall | This method should test the validateSize method when “Small” is passed as parameter |
| test12ValidateSizeWhenMedium | This method should test the validateSize method when “Medium” is passed as parameter. |
| test13ValidateSizeWhenLarge | This method should test the validateSize method when “Large” is passed as parameter. |
| test14ValidateSizeWhenInvalid | This method should test the validateSize method when invalid value is passed as parameter  validateSize is expected to throw InvalidPizzaOrderException when type is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test15ViewPizzaOrderByIdWhenValid | This method should test the correctness of  viewPizzaOrderById method for an existing order id.  Perform testing for the correctness of the value returned. |
| test16ViewPizzaOrderByIdWhenInvalid | This method should test the correctness of  viewPizzaOrderById method for a non existing order id.  Perform testing for the correctness of the value returned.  viewPizzaOrderById method is expected to throw InvalidPizzaOrderException when order id does not exist.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test17ViewPizzaOrdersByType | This method should test the correctness of  viewPizzaOrdersByType method.  Perform testing for the correctness of the value returned. |
| test18ViewPizzaOrdersTypeWise | This method should test the correctness of  viewPizzaOrdersTypeWise method.  Perform testing for the correctness of the value returned. |
| test19CountTotalQuantityForEachSize | This method should test the correctness of countTotalQuantityForEachSize method.  Perform testing for the correctness of the value returned. |
| test20ViewPizzaOrdersByTypeForEmptyList | This method should test the correctness of viewPizzaOrdersByType method for an empty pizzaOrderList.  viewPizzaOrdersByType method is expected to throw InvalidPizzaOrderException when pizzaOrderList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test21ViewPizzaOrdersTypeWiseForEmptyList | This method should test the correctness of  viewPizzaOrdersTypeWise method for an empty pizzaOrderList.  viewPizzaOrdersTypeWise method is expected to throw InvalidPizzaOrderException when pizzaOrderList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test22CountTotalQuantityForEachSizeForEmptyList | This method should test the correctness of countTotalQuantityForEachSize method for an empty pizzaOrderList.  countTotalQuantityForEachSize method is expected to throw InvalidPizzaOrderException when pizzaOrderList 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 PizzaShopTest 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 PizzaShopTest class.

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