Elegant Outcomes

**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/74070/mod_vpl/intro/EventManagement.zip)

Elegant Outcomes is a famous event management company in the city. Due to the increased events, they have approached Zee Software to automate their various requirements.

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 EventDetails

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| EventDetails | String eventId  String eventType  Date dateOfRegistration  Date dateOfEvent  long pointOfContact  double payment | Necessary getters, setters are provided  Parameterized constructor and no-argument constructor are provided as a part of the code skeleton |

·         Here, eventType can take a value either “Wedding” or “Birthday” or “Conference” or “Product Launch” or “Corporate Hospitality” [Note: Values are case insensitive]

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the event  type | EventManagement | public boolean validateEventType(String eventType) | Validate the eventType.  If valid, return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidEventException”  if the eventType Is not “Wedding” or “Birthday” or “Conference”  or “Product Launch” or “Corporate Hospitality” |
| View Event based on Event Id | EventManagement | public EventDetails viewEventDetailsByEventId(List<EventDetails> eventList, String eventId) | This method should return the Event details object with the Event Id passed as parameter from list of events, which is also passed as parameter.  If the eventList is empty or if there is no event with the given Event Id it should throw a user defined exception | Throw a user defined exception “InvalidEventException” if the  eventList is empty or if  no event exists with the given Event Id. |
| View the list of Events for a given Event type | EventManagement | public List<EventDetails> viewEventDetailsByEventType(List<EventDetails> eventList, String type) | This method takes the eventList and an event type as an argument. It should return the list of Events for the given event type. If the eventList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidEventException” if the  eventList is empty |
| Count the Events based on date of registration | EventManagement | public int countEventsByDateOfRegistration(List<EventDetails> eventList, Date date) | This method takes the eventList and the date as argument. It should return the count of events based on the date of registration.  If the eventList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidEventException” if the eventList is empty. |
| View the event details based on the date of event | EventManagement | public Map<Date, List<EventDetails>> viewEventsByDateOfEvent (List<EventDetails> eventList) | This method should return the event details based on the date of event. It takes the eventList as argument and return a Map with key as Date of event and value as event details based on the date of event. If the eventList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidEventException” if the eventList is empty. |
| Calculate amount based on the date of event | EventManagement | public double calculateAmountByDateOfEvent(List<EventDetails> eventList, Date date) | This method takes the eventList and the date as arguments. It should calculate the amount based on the date of event and return the same. If the eventList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidEventException” if the eventList is empty. |

You need to write Junit test for the EventManagement class.

**Testing Scenarios:**

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

**Note:**

To perform testing, the eventList should contain objects of EventDetails.

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateEventTypeForProductLaunch | This method should test the validateEventType method when “Product Launch” is passed as parameter |
| test12ValidateEventTypeForCorporateHospitality | This method should test the validateEventType method when “Corporate Hospitality” is passed as parameter |
| test13ValidateEventTypeForInvalidEventType | This method should test the validateEventType method when invalid value is passed as parameter  validateEventType is expected  to throw InvalidEventException when type is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ViewEventDetailsByValidEventId | This method should test the correctness of viewEventDetailsByEventId method for  an existing Event Id.  Perform testing for the correctness of the value returned. |
| test15ViewEventDetailsByInvalidEventId | This method should test the correctness of viewEventDetailsByEventId method for a non existing Event Id.  Perform testing for the correctness of the value returned.  viewEventDetailsByEventId method is expected  to throw InvalidEventException when Event Id does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test16ViewEventDetailsByEventType | This method should test the correctness of  viewEventDetailsByEventType method.  Perform testing for the correctness of the value returned. |
| test17ViewEventDetailsByEventTypeForEmptyList | This method should test the correctness of viewEventDetailsByEventType method for an empty eventList.  viewEventDetailsByEventType method is expected  to throw InvalidEventException when eventList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test18CountEventsByDateOfRegistration | This method should test the correctness of countEventsByDateOfRegistration method.  Perform testing for the correctness of the value returned. |
| test19CountEventsByDateOfRegistrationForEmptyList | This method should test the correctness of countEventsByDateOfRegistration method for an empty eventList.  countEventsByDateOfRegistration method is expected  to throw InvalidEventException when eventList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test20ViewEventsByDateOfEvent | This method should test the correctness of viewEventsByDateOfEvent method.  Perform testing for the correctness of the value returned. |
| test21ViewEventsByDateOfEventForEmptyList | This method should test the correctness of viewEventsByDateOfEvent method for an empty eventList.  viewEventsByDateOfEvent method is expected  to throw InvalidEventException when eventList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test22CalculateAmountByDateOfEvent | This method should test the correctness of calculateAmountByDateOfEvent method.  Perform testing for the correctness of the value returned. |
| test23CalculateAmountByDateOfEventForEmptyList | This method should test the correctness of calculateAmountByDateOfEvent method for an empty eventList.  calculateAmountByDateOfEvent method is expected  to throw InvalidEventException when eventList 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 EventManagementTest 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 EventManagementTest class.

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