Assessment Management

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

***AssessmentManagement***

[*Click here to download the code skeleton*](https://cognizant.tekstac.com/pluginfile.php/69137/mod_vpl/intro/Assessment%20Management.zip)

Zee School conducts online assessment for class/standard 5  to 10. To infer the Assessment Report of the students, the school have approached hi-Tech software to automate their various functional 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 AssessmentReport

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| AssessmentReport | int  rollNo  String studentName  int  totalMarks  int standard  char section  String classTeacherName | Necessary getters,setters are provided  A Constructor is also provided |

·         Here, section can take a value either ‘A’ or ‘B’ [Note: Values are case insensitive].

·         Assume: For all classes the total marks is out of 500

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the section | ZeeSchool | public boolean validateSection (char section) | Validate the section.  If valid, return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidAssessmentException”  if the section Is neither ‘A’ or ‘B’ |
| View AssessmentReport based on RollNo | ZeeSchool | public AssessmentReport viewAssessment(List<AssessmentReport> assessmentReportList,int rollNo) | This method should return the AssessmentReport object for the rollNo passed as parameter from list of  AssessmentReport,which is also passed as parameter.  If the assessmentReportList is empty or if there is no AssessmentReport with the given rollNo it should throw a user defined exception | Throw a user defined exception “InvalidAssessmentException” if the  assessmentReportList is empty or if  no AssessmentReport exists with the given rollNo. |
| View the list of AssessmentReports for a given class | ZeeSchool | public List<AssessmentReport> viewAssessmentReportForTheGivenClass (List<AssessmentReport> assessmentReportList ,int standard) | This method takes the assessmentReportList and a standard as an argument. It should return the list of AssessmentReport for the given class/standard. If the assessmentReportList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAssessmentException” if the  assessmentReportList is empty |
| View all the AssessmentReports class wise | ZeeSchool | public Map<Integer,List<AssessmentReport> > viewAssessmentReportClassWise(List<AssessmentReport> assessmentReportList) | This method takes the assessmentReportList as argument. It should return class wise, all the Assessment Reports in the list. The return type is map, where the class is key and value is the List of AssessmentReport  belonging to that class.  If the assessmentReportList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAssessmentException” if the  assessmentReportList is empty. |
| View the total number of high performers in each class | ZeeSchool | public  Map<Integer,Integer> countHighPerformersClassWise(List<AssessmentReport> assessmentReportList) | This method should return the number of high performers for each class in the assessmentReportList.(Student who have scored >=450 is considered as a high performer). It takes the assessmentReportList as argument and returns a Map with key as class and value as count of high performers.                If the assessmentReportList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAssessmentException” if the assessmentReportList is empty. |

You need to write Junit test for the ZeeSchool class.

**Testing Scenarios:**

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

**Note:**

To perform testing, the assessmentReportList should contain objects of AssessmentReport.

To do this, in the ZeeSchoolTest class you are provided with a setup method. Use this method to populate the static variable assessmentReportList in ZeeSchoolTest class.

That is, create few objects for  AssessmentReport and populate the assessmentReportList given in ZeeSchoolTest class with these objects and use that list to test the methods  in ZeeSchool class that needs a AssessmentReport list to be passed as parameter.

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateSectionWithA | This method should test the validateSection method when ‘A’ is passed as parameter |
| test12ValidateSectionWithB | This method should test the validateSection method when ‘B’ is passed as parameter. |
| test13ValidateSectionWhenInvalid | This method should test the validateSection method when invalid value is passed as parameter  validateSection is expected  to throw InvalidAssessmentException when type is invalid.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block. |
| test14ViewAssessmentForValidRollNo | This method should test the correctness of  viewAssessment method for  an existing rollNo.  Perform testing for the correctness of the value returned. |
| test15ViewAssessmentForInvalidRollNo | This method should test the correctness of  viewAssessment method for a non existing rollNo.  Perform testing for the correctness of the value returned.  viewAssessment method is expected  to throw InvalidAssessmentException when rollNo does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test16ViewAssessmentReportForTheGivenClass | This method should test the correctness of  viewAssessmentReportForTheGivenClass method.  Perform testing for the correctness of the value returned. |
| test17ViewAssessmentReportForTheGivenClassForEmptyList | This method should test the correctness of viewAssessmentReportForTheGivenClass method for an empty assessmentReportList.  viewAssessmentReportForTheGivenClass method is expected  to throw InvalidAssessmentException when assessmentReportList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test18ViewAssessmentReportClassWise | This method should test the correctness of viewAssessmentReportClassWise method.  Perform testing for the correctness of the value returned. |
| test19ViewAssessmentReportClassWiseForEmptyList | This method should test the correctness of viewAssessmentReportClassWise method for an empty assessmentReportList.  viewAssessmentReportClassWise method is expected  to throw InvalidAssessmentException when assessmentReportList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test20CountHighPerformersClassWise | This method should test the correctness of countHighPerformersClassWise method.  Perform testing for the correctness of the value returned. |
| test21 countHighPerformersClassWiseForEmptyList | This method should test the correctness of countHighPerformersClassWise method for an empty assessmentReportList.  countHighPerformersClassWise method is expected  to throw InvalidAssessmentException when assessmentReportList 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 ZeeSchoolTest 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 ZeeSchoolTest class.

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