Peking College

**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/69144/mod_vpl/intro/Peking%20%20College.zip)

***Peking College***is a famous college in the city. They had automated their admission procedures, using that application you can maintain information of the admissions for their college after counseling.

Vishruth 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 AdmissionDetails

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

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| AdmissionDetails | String admissionId  String studentId  Date dateOfCounseling  Date dateOfAdmission  String courseName  double totalFee | Necessary getters,setters are provided  A Constructor is also provided |

·         Here, courseName can take a value either “ECE” or “EEE” or “MECH”.

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the Admission Id | CollegeAdmission | public boolean validateAdmissionId(String admissionId) | Validate the Admission Id.  If valid, return true, else this method should throw a user defined exception | Throw a user defined exception “InvalidAdmissionDetailsException”  if the admissionId does not contain 4 letters followed by 4 digits followed by the letters V and C. |
| View Admission details based on student Id | CollegeAdmission | public AdmissionDetails viewAdmissionDetailsByStudentId(List<AdmissionDetails> admissionList, String studentId) | This method should return the AdmissionDetails object with the student Id passed as parameter from list of admissions, which is also passed as parameter.  If the admissionList is empty or if there are no details for the given student Id it should throw a user defined exception | Throw a user defined exception “InvalidAdmissionDetailsException” if the admissionList is empty or if no details exist with the given student Id. |
| View the list of Admission details for a given Counseling date | CollegeAdmission | public List<AdmissionDetails> viewDetailsByDateOfCounseling(List<AdmissionDetails> admissionList, Date date) | This method takes the admissionList and a date as an argument. It should return the list of Admission details for the given counseling date. If the admissionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAdmissionDetailsException” if the admissionList is empty |
| Count the number of admission details based on the course name | CollegeAdmission | public int countOfDetailsByCourseName(List<AdmissionDetails> admissionList, String courseName) | This method takes the admissionList as argument along with the course name. It should return the count of admission details based on the course name.  If the admissionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAdmissionDetailsException” if the admissionList is empty |
| View the Admission details -  Admission datewise | CollegeAdmission | public Map<Date, List<AdmissionDetails>> viewDetailsByDateOfAdmission(List<AdmissionDetails> admissionList) | This method should return the list of Admission details for each Admission date in the admissionList. It takes the admissionList as argument and returns a Map with key as Admission date and value as list of admission details on that date. If the admissionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAdmissionDetailsException” if the admissionList is empty |
| Calculate the amount based on the date of Admission | CollegeAdmission | public double calculateAmountByDateOfAdmission(List<AdmissionDetails> admissionList, Date date) | This method takes the admissionList as argument along with the date of Admission. It should calculate and return the amount received on that admission date.  If the admissionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidAdmissionDetailsException” if the admissionList is empty |

You need to write Junit test for the CollegeAdmissionclass.

**Testing Scenarios:**

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

**Note:**

To perform testing, the admissionList should contain objects of AdmissionDetails.

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateAdmissionIdForValidAdmissionId | This method should test the validateAdmissionId method when a valid Admission Id is passed as parameter |
| test12ValidateAdmissionIdForInvalidAdmissionId | This method should test the validateAdmissionId method when an invalid Admission Id is passed as parameter  validateAdmissionId is expected to throw InvalidAdmissionDetailsException when Admission Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test13ValidateAdmissionIdForInvalidAdmissionIdWithoutVC | This method should test the validateAdmissionId method when an invalid Admission Id without the letters V and C is passed as parameter  validateAdmissionId is expected to throw InvalidAdmissionDetailsException when Admission Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ValidateAdmissionIdForInvalidAdmissionIdWithLengthGreaterThan10 | This method should test the validateAdmissionId method when an invalid Admission Id whose String length is greater than expected is passed as parameter  validateAdmissionId is expected to throw InvalidAdmissionDetailsException when Admission Id is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test15ViewAdmissionDetailsByValidStudentId | This method should test the correctness of viewAdmissionDetailsByStudentId method for an existing Student Id.  Perform testing for the correctness of the value returned. |
| test16ViewAdmissionDetailsByInvalidStudentId | This method should test the correctness of viewAdmissionDetailsByStudentId method for a non-existing Student Id.  Perform testing for the correctness of the value returned.  viewAdmissionDetailsByStudentId method is expected to throw InvalidAdmissionDetailsException when Student Id does not exist.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test17ViewDetailsByDateOfCounseling | This method should test the correctness of viewDetailsByDateOfCounseling method.  Perform testing for the correctness of the value returned. |
| test18ViewDetailsByDateOfCounselingForEmptyList | This method should test the correctness of viewDetailsByDateOfCounseling method for an empty admissionList.  viewDetailsByDateOfCounseling method is expected to throw InvalidAdmissionDetailsException when admissionList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test19CountOfDetailsByCourseName | This method should test the correctness of countOfDetailsByCourseName method.  Perform testing for the correctness of the value returned. |
| test20CountOfDetailsByCourseNameForEmptyList | This method should test the correctness of countOfDetailsByCourseName method for an empty admissionList.  countOfDetailsByCourseName method is expected to throw InvalidAdmissionDetailsException when admissionList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test21ViewDetailsByDateOfAdmission | This method should test the correctness of viewDetailsByDateOfAdmission method.  Perform testing for the correctness of the value returned. |
| test22ViewDetailsByDateOfAdmissionForEmptyList | This method should test the correctness of viewDetailsByDateOfAdmission method for an empty admissionList.  viewDetailsByDateOfAdmission method is expected to throw InvalidAdmissionDetailsException when admissionList is empty.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block |
| test23CalculateAmountByDateOfAdmission | This method should test the correctness of calculateAmountByDateOfAdmission method.  Perform testing for the correctness of the value returned. |
| test24CalculateAmountByDateOfAdmissionForEmptyList | This method should test the correctness of calculateAmountByDateOfAdmission method for an empty admissionList.  calculateAmountByDateOfAdmission method is expected to throw InvalidAdmissionDetailsException when admissionList 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 CollegeAdmissionTest 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 CollegeAdmissionTest class.

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