Morrison Builders

**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/74071/mod_vpl/intro/MorrisonBuilders.zip)

Morrison Builders is a famous construction company in the city.    Due to the increased projects, they have approached Ozone technologies 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 ProjectInfo

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| ProjectInfo | String projectId  String constructionType  int totalSquareFeet  Date dateOfRegistration  Date dateOfCompletion  double amountQuoted | Necessary getters, setters are provided  A Constructor is also provided |

·         Here, constructionType can take a value either “Residential” or “Commercial” [Note: Values are case insensitive]

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the construction  type | BuildingContract | public boolean vaildateConstructionType(String constructionType) | Validate the constructionType.  If valid, return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidProjectInfoException”  if the constructionType Is neither “Residential” nor “Commercial” |
| View Project based on Project Id | BuildingContract | public ProjectInfo viewProjectByProjectId(List<ProjectInfo> projectList, String projectId) | This method should return the ProjectInfo object with the Project Id passed as argument from list of projects, which is also passed as parameter.  If the projectList is empty or if there is no project with the given Project Id it should throw a user defined exception | Throw a user defined exception “InvalidProjectInfoException” if the  projectList is empty or if  no project exists with the given Project Id. |
| View the list of Projects for a given construction type | BuildingContract | public List<ProjectInfo> viewProjectsByConstructionType(List<ProjectInfo> projectList, String constructionType) | This method takes the projectList and a construction type as an argument. It should return the list of Projects for the given construction type. If the projectList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidProjectInfoException” if the  projectList is empty |
| Count the Projects based on area range | BuildingContract | public int countProjectsByAreaRange(List<ProjectInfo> projectList, int fromArea,int toArea) | This method takes the projectList and the area range as arguments. It should return the count of projects based on the area of each project.  If the projectList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidProjectInfoException” if the projectList is empty. |
| View the Project Info based on the date of registration | BuildingContract | public Map<Date, List<ProjectInfo>> viewProjectsByDateOfRegistration(List<ProjectInfo> projectList) | This method should return the project info based on the date of registration. It takes the projectList as argument and return a Map with key as Date of registration and value as project details based on the date of registration. If the projectList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidProjectInfoException” if the projectList is empty. |
| Calculate amount based on the date of completion | BuildingContract | public Map<Date,Double> calculateAmountDateOfCompletionWise(List<ProjectInfo> projectList) | This method should return the project info based on the date of completion. It takes the projectList as argument and return a Map with key as Date of completion and value as amount quoted based on the date of completion. If the projectList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidProjectInfoException” if the projectList is empty. |

You need to write Junit test for the BuildingContract class.

**Testing Scenarios:**

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

**Note:**

To perform testing, the projectList should contain objects of ProjectInfo.

To do this, in BuildingContractTest class you are provided with a setup method.  Use this method to populate the static variable projectList in BuildingContractTest class.  That is, create few objects for ProjectInfo and populate the projectList given in BuildingContractTest class with these objects and use that list to test the methods in BuildingContract class that needs an ProjectInfo list to be passed as parameter.

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateConstructionTypeForResidential | This method should test the validateConstructionType method when “Residential” is passed as parameter |
| test12ValidateConstructionTypeForCommercial | This method should test the validateConstructionType method when “Commercial” is passed as parameter |
| test13ValidateConstructionTypeForInvalidConstructionType | This method should test the validateConstructionType method when invalid value is passed as parameter  validateConstructionType is expected  to throw InvalidProjectInfoException when type is invalid.  Write JUnit to test for the exception thrown either by using appropriate annotation or by using try catch block. |
| test14ViewProjectByValidProjectId | This method should test the correctness of viewProjectByProjectId method for an existing Project Id.  Perform testing for the correctness of the value returned. |
| test15ViewProjectByInvalidProjectId | This method should test the correctness of viewProjectByProjectId method for a non existing Project Id.  Perform testing for the correctness of the value returned.  viewProjectByProjectId method is expected  to throw InvalidProjectInfoException when Project Id does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test16ViewProjectsByConstructionType | This method should test the correctness of viewProjectsByConstructionType method.  Perform testing for the correctness of the value returned. |
| test17ViewProjectsByConstructionTypeForEmptyList | This method should test the correctness of viewProjectsByConstructionType method for an empty projectList.  viewProjectsByConstructionType method is expected  to throw InvalidProjectInfoException when projectList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test18CountProjectsByAreaRange | This method should test the correctness of countProjectsByAreaRange method.  Perform testing for the correctness of the value returned. |
| test19CountProjectsByAreaRangeForEmptyList | This method should test the correctness of countProjectsByAreaRange method for an empty projectList.  countProjectsByAreaRange method is expected  to throw InvalidProjectInfoException when projectList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test20ViewProjectsByDateOfRegistration | This method should test the correctness of viewProjectsByDateOfRegistration method.  Perform testing for the correctness of the value returned. |
| test21ViewProjectsByDateOfRegistrationForEmptyList | This method should test the correctness of viewProjectsByDateOfRegistration method for an empty projectList.  viewProjectsByDateOfRegistration method is expected  to throw InvalidProjectInfoException when projectList is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test22CalculateAmountDateOfCompletionWise | This method should test the correctness of calculateAmountDateOfCompletionWise method.  Perform testing for the correctness of the value returned. |
| test23CalculateAmountDateOfCompletionWiseForEmptyList | This method should test the correctness of calculateAmountDateOfCompletionWise method for an empty projectList.  calculateAmountDateOfCompletionWise method is expected  to throw InvalidProjectInfoException when projectList 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 BuildingContractTest 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 BuildingContractTest class.

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