Bank Transaction

**Grade settings**: Maximum grade: 100  
**Disable external file upload, paste and drop external content**: Yes  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

[***Click here to download the code template***](https://cognizant.tekstac.com/pluginfile.php/54382/mod_vpl/intro/BankTransaction.zip?time=1614944685846)

***Bank System***is an automated bank transaction management system. Using the application, you can maintain information of Transactions where the transaction type is checked and information can be accessed for given transaction id or based on the account number the transactions performed can be retrieved.

Prithvi 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 Transaction

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

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** |
| Transaction | int transactionId  Date transactionDate  String accountNumber  String transactionType  double amount | Necessary getters and setters are provided.  A constructor is also provided. |

Here the **transactionType**can take a value either “Credit” or “Debit”.

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

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

You are also provided with a utility class Bank with business methods.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Attributes** | **Methods** |
| Perform various transactions in a Bank | Bank |  | Has various methods to manipulate the transactions. |

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type (Class)** | **Methods** | **Responsibilities** | **Exception** |
| Validate the transaction type | Bank | public boolean validateTransactionType(String transactionType) | Validate the transactionType..  If valid return true , else this method should throw a user defined exception | Throw a user defined exception “InvalidTransactionException”  if the transactionType Is neither “Debit” nor “Credit” |
| View the Transaction based on transactionId | Bank | public Transaction viewTransaction(List<Transaction> transactionList,int transactionId) | This method should return the Transaction object with the transactionId passed as parameter from list of  transaction which is also passed as parameter.  If the transactionList is empty or if there is no transaction in the given transaction id it should throw a user defined exception. | Throw a user defined exception “InvalidTransactionException” if the  transactionList is empty or if there is  no transaction in the given transaction id. |
| View the list of transaction for a given Account number | Bank | public List<Transaction> viewTransactionForAccount(List<Transaction> transactionList ,String accountNumber) | This method takes the transactionList and an accountNumber as argument. It should return the list of transaction for a given account number. If the transactionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTransactionException” if the  transactionList is empty |
| View accout wise transaction performed for all Accounts | Bank | public Map<String,List<Transaction>>  viewTransactionAccountwise(List<Transaction> transactionList) | This method takes the transactionList as argument. It should return accoutwise transactions made for all accounts in the list. The return type is map, where the accountNumber is key and value is the List of Transaction performed on that account.  If the transactionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTransactionException” if the  transactionList is empty. |
| View the number of transactions for each Account within a given range of date | Bank | public  Map<String,Integer> countTransactionsAccountwise(List<Transaction> transactionList , Date fromDate,Date toDate) | This method should return the number of transactions performed on each accountNumber in the transactionList for a given dateRange(From fromDate to toDate). It takes the transactionList, fromDate and toDate as arguments and returns a Map with key as accountNumber and value as count of transactions within the given date range.                          If the transactionList is empty it should throw a user defined exception. | Throw a user defined exception “InvalidTransactionException” if the  transactionList is empty. |

You need to write Junit test for the Bank class.

**Testing Scenarios:**

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

**Note:**

To perform testing the transactionList should contain objects of Transaction.

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

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

|  |  |
| --- | --- |
| **Test Method** | **Scenarios / Responsibilities** |
| test11ValidateTransactionTypeWhenCredit | This method should test the validateTransactionType method when “Credit” is passed as parameter |
| test12ValidateTransactionTypeWhenDebit | This method should test the validateTransactionType method when “Debit” is passed as parameter. |
| test13ValidateTransactionTypeWhenInvalid | This method should test the validateTransactionType method when invalid value is passed as parameter  validateTransactionType is expected  to throw InvalidTransactionException when type is invalid.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block. |
| test14ViewTransactionForValidId | This method should test the correctness of  viewTransaction method for  an existing transaction id .  Perform testing for the correctness of the value returned. |
| test15ViewTransactionForInvalidId | This method should test the correctness of  viewTransaction method for a non existing transaction id.  Perform testing for the correctness of the value returned.  viewTransaction method is expected  to throw InvalidTransactionException when transaction id does not exist.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test16ViewTransactionForAccount | This method should test the correctness of  viewTransactionForAccount method.  Perform testing for the correctness of the value returned. |
| test17ViewTransactionForAccountForEmptyList | This method should test the correctness of viewTransactionForAccount method for an empty transactionList.  viewTransactionForAccount method is expected  to throw InvalidTransactionException when transaction list is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test18ViewTransactionAccountwise | This method should test the correctness of viewTransactionAccountwise method.  Perform testing for the correctness of the value returned. |
| test19ViewTransactionAccountwiseForEmptyList | This method should test the correctness of viewTransactionAccountwise method for an empty transactionList.  viewTransactionAccountwise method is expected  to throw InvalidTransactionException when transaction list is empty.  Write JUnit to test for the exception thrown  either by using appropriate annotation or by using try catch block |
| test20CountTransactionsAccountwise | This method should test the correctness of countTransactionsAccountwise method.  Perform testing for the correctness of the value returned. |
| test21CountTransactionsAccountwiseForEmptyList | This method should test the correctness of countTransactionsAccountwise method for an empty transactionList.  countTransactionsAccountwise  method is expected  to throw InvalidTransactionException when transaction list 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 BankTest 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 BankTest class.

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