TestMockDB

Click here to download the code template

Important Instructions:

- Please read the document thoroughly before you code.
- Import the given skeleton code into your Eclipse.
- Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
- You can create any number of private methods inside the given class.
- Use **Mockito** annotations to mock & injects instances.
- You are provided with a **Main class** with the main method to check the correctness of the test methods written.
- Having completed writing the test methods, execute the main method and identify the result.

Assessment Coverage:

- Mocking DataSource, Connection, PreparedStatement, and mock injection
- JUnit annotations & exception rule

Given you an Bank Service application, which accepts amount to deposit in an account. If the account number is **null / empty**, it has to throw **AccountTransactionException** with message **"Invalid account number"**. If the amount is **negative/zero**, it has to throw **AccountTransactionException** with message **"Invalid amount"**. On the successful execution of deposit method, balance column in account table updated as deposit + amount.

You are required to write Junit test case using **Mockito** and check the correctness of the application developed.

Skeleton:

Functional Requirements:

The application has the below classes and methods implemented.

Component Specification : AccountDAO (DAO Class)

Class	Attribute(s)	Template Method(s)
AccountDAO	DataSource ds	

Business Rule:

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

Class	Method(s)	Responsibilities	Exception

	accNo, double amount)	if accNo & amount valid, it uses data source and add amount in ACCOUNT table balance column for the account_number accNo using prepared statement.	AccountTransactionException with message "Invalid account number". If amount parameter is less than or equal to 0, it has to throw AccountTransactionException with message "Invalid amount"

Testing Scenarios:

You are provided with a class "**AccountDAOTest**" to do this testing. You are required to **mock data source & dao** class, and verify method call times & mock exception.

Note :

- To perform testing, the AccountDAO should contain objects of DataSource.

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

Test Cases / Methods	**Scenarios / Responsibilities**
testInvalidAccount	This method should expect & handle the **AccountTransactionException &** message **"Invalid account number"** when **deposit()** method called with **empty** account number and some amount.
testNullAccount	This method should expect & handle the **AccountTransactionException &** message **"Invalid account number"** when **deposit()** method called with **null** as account number and some amount.
testInvalidAmount	This method should expect & handle the **AccountTransactionException &** message **"Invalid amount"** when **deposit()** method called with some account number with 0/negative number as amount.
testMethodCall	This method should verify whether the following methods called exactly 1 time mocked instance — method to verify DataSource — getConnection() Connection — prepareStatement() PreparedStatement — executeUpdate() if **deposit()** method invoked with valid account number and amount and **assert** it returns true.

Implement the test methods and **add the needed annotations** to all the methods in AccountDAOTest class.