

Banking Application

Project Objective:

Create a console based Java application that would allow the customer of a bank to perform day to day bank transactions. The following are the tasks that need to be performed by the Customer.

1. View balance.
2. Transfer amount.

Overview:

View balance: If the account number is given the balance should be returned

Transfer Amount: This function is used to transfer money from one account to another account.

For the operation to be successful, the following conditions are to be met.

1. Both the account numbers should be valid
2. The account number from where the money is transferred should have enough money for performing the transfer operation

If all these conditions are met, the given amount has to be debited from the payer and credited to the beneficiary (account_tbl) and an entry has to be made in the transfer_tbl

A. Database Design:

1. **Create Table [To be done using sql commands, after logging-in as the new user that has been created in above step]**

Table Name: ACCOUNT_TBL

Values for this table will be hardcoded directly.

| Column | Datatype | Description |
|----------------|--------------|----------------------|
| Account_Number | Varchar2(10) | Primary Key. |
| Customer_Name | Varchar2(15) | Account holder name. |
| Balance | Number(10,2) | Account Balance |

Insert some records into the Account_TBL

Sample Records

| ACCOUNT_NUMBER | CUSTOMER_NAME | BALANCE |
|----------------|---------------|---------|
| 1234567890 | Reddy | 80000 |
| 1234567891 | Mahesh | 0 |
| 1234567892 | Dhanu | 100 |
| 1234567893 | Sam | 500 |

Table Name: TRANSFER_TBL

| Column | Datatype | Description |
|----------------------------|--------------|---|
| Transaction_ID | Number(4) | Primary Key |
| Account_Number | Varchar2(10) | Foreign Key, this field references Account_Number field of Account_tbl. |
| Beneficiary_account_number | Varchar2(10) | Foreign Key, this field references Account_Number field of Account_tbl. |
| Transaction_Date | Date | Date of transaction. |
| Transaction_Amount | Number(10,2) | Amount to be transferred. |

2. Create Sequence:

Sequence Name : transactionId_seq

| Sequence Name | Minimum Value | Max Values | Incremental value | Start Value |
|-------------------|---------------|------------|-------------------|-------------|
| transactionId_seq | 1000 | 9999 | 1 | 1000 |

B. System Design:

| Name of the package | Usage |
|------------------------|--|
| com.wipro.bank.service | This package will contains the class which displays the console menu and takes the user input. It contains the methods that performs validation on the given input and invokes the respective DAO operations |
| com.wipro.bank.bean | This package will contain the entity class named TransferBean. |
| com.wipro.bank.dao | This package will contain the class that will do the database related JDBC code. |
| com.wipro.bank.util | This package will contain the class to establish database connection and also the class that handles the user defined exception. |

Package: com.wipro.bank.util

| Class | Method and Variables | Description |
|----------------------------|---|---|
| DBUtil | | DB connection class |
| | public static Connection getDBConnection() | Establish a connection to the database and return the java.sql.Connection reference |
| InsufficientFundsException | | User defined exception class |
| | public String toString | Returns a String "INSUFFICIENT" |

| | | |
|--|--|---|
| | | FUNDS” .The details about when it has to be thrown is given in the appropriate methods |
|--|--|---|

Package: com.wipro.bank.bean

| Class | Method and Variables | Description |
|---------------------|----------------------------------|--|
| TransferBean | | Class |
| | private int transactionID | Transaction Id |
| | private String fromAccountNumber | AccountNumber from where money is going to be transferred *Maps to Account_Number field of Transfer_tbl |
| | private String toAccountNumber | AccountNumber to where money is going to be transferred *Maps to Beneficiary_account_number field of Transfer_tbl |
| | private Date dateOfTransaction | Date on which transaction is taking place-current Date [java.util.Date] |
| | private float amount | Amount to be transferred |
| | setters & getters | Should create the getter and setter methods for all the attributes mentioned in the class |

Package: com.wipro.bank.dao

| Class | Method and Variables | Description |
|----------------|---|--|
| BankDAO | | DAO class |
| | public int generateSequenceNumber() | <ul style="list-style-type: none"> This method generates 4 digit auto generated number using transactionId_seq sequence |
| | public boolean validateAccount(String accountNumber) | <ul style="list-style-type: none"> Check account_tbl and return true if account number is valid, else return false. |
| | public float findBalance(String accountNumber) | <ul style="list-style-type: none"> Check account_tbl and return balance if accountNumber is valid else return -1 |
| | public boolean transferMoney(TransferBean transferBean) | <ul style="list-style-type: none"> Insert the transferBean values into the transfer_tbl. The transactionID is the value got from generateSequenceNumber The transaction date is today's date |

| | | |
|--|--|--|
| | | <ul style="list-style-type: none"> On successful insertion return true else return false |
| | public boolean updateBalance(String accountNumber, float newBalance) | <ul style="list-style-type: none"> Update account_tbl with the newBalance for the given accountNumber Return true for successful updation and false if not |

Package: com.wipro.bank.service

| Class | Method and Variables | Description |
|----------|---|-------------|
| BankMain | | Main class |
| | public static void main (String[] args) The code that is needed to test your program goes here. A sample code is shown at the end of the document. | |
| | public String checkBalance(String accountNumber) <u>Steps to perform:</u> Invoke appropriate BankDAO methods and perform the following: <ol style="list-style-type: none"> Validate the accountNumber If valid, find the Balance for the given accountNumber Return message in given format For eg) If the balance returned by findBalance method is 10000 then the return value is BALANCE IS:10000.0 If AccountNumber is invalid return the following message ACCOUNT NUMBER INVALID | |
| | public String transfer(TransferBean transferBean) <u>Steps to perform:</u> Invoke appropriate BankDAO methods and perform the following: <ol style="list-style-type: none"> If transferBean is null the function should return "INVALID" Validate both the accountnumbers in the transferbean. In case if any of the accountNumbers are invalid the function should return INVALID ACCOUNT If both the numbers are valid, check if the fromAccountNumber has sufficientfunds to transfer The function will throw "InsufficientFundsException" if the payer does not have sufficient money. The exception will be caught in the same method itself. If exception is caught the function should return "INSUFFICIENT FUNDS" [Note: Do not use System.exit(0) while handling exception] If the Payer has enough money, update account_tbl for both the account numbers to perform the transfer operation (reduce the given amount from fromAccountNumber and add the given amount into | |

| | |
|--|---|
| | toAccountNumber] and invoke the transferMoney function of the BankDAO class to include the transaction detail in the transfer_tbl 6. If step5 was successful, the method would return "SUCCESS" . |
|--|---|

Main Method:

You can write code in the main method and test all the above test cases. A sample code of the main method to test the first test case is shown below for your reference.

```
public static void main(String[] args) {  
  
    // View Balance  
    System.out.println(bankMain.checkBalance("1234567890"));  
  
    // TransferMoney  
    TransferBean transferBean = new TransferBean();  
  
    transferBean.setFromAccountNumber("1234567890");  
    transferBean.setAmount(500);  
    transferBean.setToAccountNumber("1234567891");  
    transferBean.setDateOfTransaction(new java.util.Date());  
  
    System.out.println(bankMain.transfer(transferBean));  
}
```

Test Cases:

Below is the actual set of test cases that the CPC test engine will run in the background. Please ensure that the conditions mentioned in these test-cases are handled by your class design.

1. Test for SequenceNumber Creation
2. Test for Balance checking with valid account number
3. Test for Balance checking with invalid account numnber
4. Test for successful transfer of funds
5. Test for transfer with low funds
6. Test for transfer with zero balance
7. Test for transfer with invalid payer account number
8. Test for transfer with invalid beneficiary account number