23CSR306 JAVA PROGRAMMING

Assignment

Banking Application

Submitted by

Roll Number : 717823F203
Name : Aburose M

KARPAGAM
COLLEGE OF ENGINEERING
Rediscover | Refine | Redefine
Approved by AICTE and Affiliated to Anna University
Autonomous | Accredited by NAAC with 'A+' Grade, 2nd Cycle
Accredited by NBA (CSE, CIVIL, ECE, EEE, IT, MECH)
(An ISO 9001:2015 and ISO 14001:2015 Certified Institution)

Department of Information Technology

KARPAGAM COLLEGE OF ENGINEERING

(Autonomous)

Myleripalayam Village, Othakkal Mandapam Post,

Coimbatore - 641032, Tamilnadu, India

NOVEMBER – 2024

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

To provide reliable and modern technology resources to the faculty and students to develop the competence in Information Technology and to endure with the rapidly changing world to serve the mankind.

MISSION

- Imparting technical knowledge through innovative teaching and research for budding professionals.
- To equip the students with strong fundamentals, programming and problem solving skills with an exposure to emerging technologies and inculcate leadership qualities with a passion to serve society.

Programme Educational Objectives (PEOs)

- **PEO1:** Graduates will be able to comprehend mathematics, science, engineering fundamentals, laboratory and work-based experiences to formulate and solve problems in the domain of Information Technology and acquire proficiency in Computer-based engineering and the use of computational tools..

- **PEO2:** Graduates will be prepared to communicate and work effectively on multidisciplinary engineering projects and practicing the ethics of their profession.

- **PEO3:** Graduates will realize the importance of self-learning and engage in lifelong learning to become experts either as entrepreneurs or employees in the field to widen the professional knowledge.

Program Outcomes

- **PO1:** Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

- **PO2:** Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

- **PO3:** Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- **PO4:** Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

- **PO5:** Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- **PO6:** The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- **PO7:** Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **PO8:** Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- **PO9:** Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **PO10:** Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- **PO11:** Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **PO12:** Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSO)

After successful completion of the program, graduates of B.E (CSE) will:

- **PSO-1** Ability to organize an IT infrastructure, secure the data and analyze the data analytic techniques in the field of data mining, big data as to facilitate in solving problems.

- **PSO-2** Ability to analyze and design the system in the domain of Cloud and Internet of Things.

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

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.

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()	• This method generates 4 digit auto generated number
	public boolean validateAccount(String accountNumber)	• Check account_tbl and return true if account number is valid, else return false.
	public float findBalance(String accountNumber)	• Check account_tbl and return balance if accountNumber is valid else return -1
	public boolean transferMoney(TransferBean transferBean)	• Insert the transferBean values into thetransfer_tbl. • The transactionID is the value got from generateSequnceNumber • The transaction date is today's date • On successful insertion return true else return false
	public boolean updateBalance(String accountNumber, float newBalance)	• Update account_tbl with the newBalance for the given accountNumber • Return true for successful updation andfalse 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:** 1. Validate the accountNumber 2. If valid, find the Balance for the given accountNumber 3. 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** 4. 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:** • 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	

Implementation:

Database Design:

Code :-

```
use project;
create table ACCOUNT_TBL(
 Account_Number varchar(10) PRIMARY KEY,
 Customer_Name varchar(15),
 Balance decimal(10,2)
 );
insert into ACCOUNT_TBL
values('1234567890','Rajini',80000.00),('7894561230','Ram',150000.00),('4578963210','Suriya',4
500000.00),('3698521470','Kumar',1000.00);

create table TRANSFER_TBL(
 Transaction_ID int(4) PRIMARY KEY,
 Account_Number varchar(10),
 Beneficiary_account_number varchar(10),
 Transaction_Date date,
 Transaction_Amount decimal(10,2),
 foreign key (Account_Number) references ACCOUNT_TBL(Account_Number),
 foreign key (Beneficiary_account_number) references ACCOUNT_TBL(Account_Number)
 );
 insert into transfer_tbl values
(1245, '1234567890', '3698521470', '2024-10-27', 20000.00),
(6598, '3698521470', '4578963210', '2024-10-27', 410000.00),
(2036, '4578963210', '7894561230', '2024-10-27', 982000.00),
(2010, '7894561230', '1234567890', '2024-10-27', 9000.00);
```

Package : com.wipro.bank.util

Class : DBUtil.java

Code :-

```
package com.wipro.bank.util;
import java.sql.Connection;x
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBUtil {
        static Connection con = null;
        public static Connection getDBConnection() {
                try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/project",
"root", "sqlroot@27");
                } catch (ClassNotFoundException e) {
                        System.out.println(e);
                } catch (SQLException e) {
                        System.out.println(e);
                }
                if (con == null)      return null;
                else    return con;
        }
}
```

Class : InsufficientFundsException.java

Code :-

```
package com.wipro.bank.util;
public class InsufficientFundsException {
    public String toString() {
            return "INSUFFICIENT FUNDS";
    }
}
```

Package : com.wipro.bank.bean

Class : TransferBean.java

Code :-

```
package com.wipro.bank.bean;
import java.sql.Date;
public class TransferBean {
    private int transactionID;
    private String fromAccountNumber;
    private String toAccountNumber;
    private Date dateOfTransaction;
    private float amount;
    public int getTransactionID() {
            return transactionID;
    }
    public void setTransactionID(int transactionID) {
            this.transactionID = transactionID;
    }
    public String getFromAccountNumber() {
            return fromAccountNumber;
```

```
	}
	public void setFromAccountNumber(String fromAccountNumber) {
		this.fromAccountNumber = fromAccountNumber;
	}
	public String getToAccountNumber() {
		return toAccountNumber;
	}
	public void setToAccountNumber(String toAccountNumber) {
		this.toAccountNumber = toAccountNumber;
	}
	public Date getDateOfTransaction() {
		return dateOfTransaction;
	}
	public void setDateOfTransaction(Date date) {
		this.dateOfTransaction = (Date) date;
	}
	public float getAmount() {
		return amount;
	}
	public void setAmount(float amount) {
		this.amount = amount;
	}
}
```

Package : com.wipro.bank.dao

Class :BankDAO.java

Code :-

```
package com.wipro.bank.dao;
import com.wipro.bank.bean.TransferBean;
import com.wipro.bank.util.DBUtil;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class BankDAO {
	public static int generateSequenceNumber() {
		return (int) (Math.random() * 9000) + 1000;
	}
	public boolean validateAccount(String accountNumber) {
		try (Connection con = DBUtil.getDBConnection()) {
			String query = "SELECT COUNT(*) FROM ACCOUNT_TBL WHERE
			Account_Number = ?";
```

```
                PreparedStatement pt = con.prepareStatement(query);
                pt.setString(1, accountNumber);
                ResultSet rs = pt.executeQuery();
                if (rs.next()) {
                        int count = rs.getInt(1);
                        return count > 0;
                }
        } catch (SQLException e) {
                System.out.println(e);
        }
        return false;
}
public float findBalance(String accountNumber) {
        try (Connection con = DBUtil.getDBConnection()) {
                PreparedStatement pt = con.prepareStatement("Select Balance from
        ACCOUNT_TBL WHERE Account_Number = ? ");
                pt.setString(1, accountNumber);
                ResultSet rs = pt.executeQuery();
                if (rs.next())
                        return rs.getFloat("Balance");

        } catch (SQLException e) {
                System.out.println(e);
        }return -1;    }
public boolean transferMoney(TransferBean transferBean) {
        try (Connection con = DBUtil.getDBConnection()) {
                PreparedStatement pt = con.prepareStatement("insert into TRANSFER_TBL
Values(?,?,?,?,?)");
                pt.setInt(1, generateSequenceNumber());
                pt.setString(2, transferBean.getFromAccountNumber());
                pt.setString(3, transferBean.getToAccountNumber());
                pt.setDate(4, transferBean.getDateOfTransaction());
                pt.setFloat(5, transferBean.getAmount());

                return pt.executeUpdate() > 0;
        } catch (SQLException e) {
                System.out.println(e);
        }
        return false;
}
public boolean updateBalance(String accountNumber, float newBalance) {
        try (Connection con = DBUtil.getDBConnection()) {
                PreparedStatement pt = con.prepareStatement("Update account_tbl set
        Balance = ? where Account_Number = ?");
```

```
			pt.setFloat(1, newBalance);
			pt.setString(2, accountNumber);
			return pt.executeUpdate() > 0;
		} catch (SQLException e) {
			System.out.println(e);
		}
		return false;
	}
}
```

Package : com.wipro.bank.service

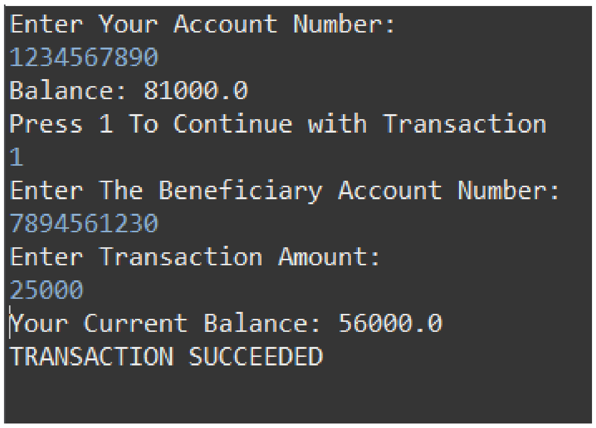
Class : BankMain.java

Code :-

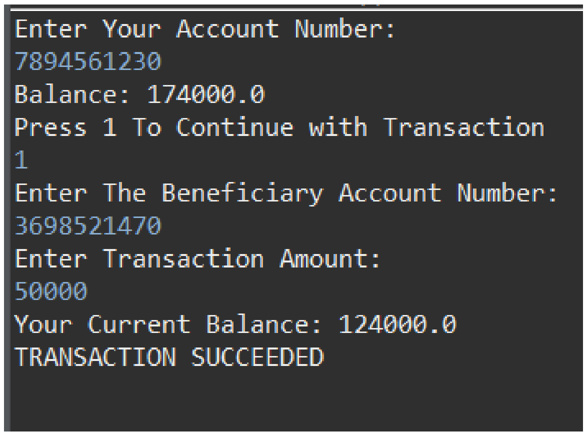
```
package com.wipro.bank.service;
import com.wipro.bank.bean.TransferBean;
import com.wipro.bank.dao.BankDAO;
import com.wipro.bank.util.InsufficientFundsException;
import java.sql.Date;
import java.util.Scanner;

public class BankMain {
	static BankDAO dao = new BankDAO();
	public static void main(String[] args) {
		BankMain bm = new BankMain();
		Scanner input = new Scanner(System.in);
		System.out.println("Enter Your Account Number: ");
		String ac_no = input.next();
		if (dao.validateAccount(ac_no)) {
			System.out.println(bm.checkBalance(ac_no));
			System.out.println("Press 1 To Continue with Transaction");
			int num = input.nextInt();
			if (num == 1) {
				System.out.println("Enter The Beneficiary Account Number: ");
				String t_ac_no = input.next();
				System.out.println("Enter Transaction Amount: ");
				int amount = input.nextInt();
				TransferBean t = new TransferBean();
				t.setFromAccountNumber(ac_no);
				t.setAmount(amount);
				t.setToAccountNumber(t_ac_no);
				t.setDateOfTransaction(new Date(System.currentTimeMillis()));
				System.out.println(bm.transfer(t));
```

```
                    } else {
                            System.out.println("Transaction Terminated");
                    }
            } else {
                    System.out.println("INVALID ACCOUNT NUMBER");
            }
    }
    public static String checkBalance(String accountNumber) {
            if (dao.validateAccount(accountNumber)) {
                    float balance = dao.findBalance(accountNumber);
                    return (balance != -1) ? "Balance: " + balance : "BALANCE NOT FOUND";
            } else {
                    return "INVALID ACCOUNT NUMBER";
            }
    }
    public String transfer(TransferBean transferBean) {
            if (transferBean == null)
                    return "INVALID INPUT";
            String fromAccountNumber = transferBean.getFromAccountNumber();
            String toAccountNumber = transferBean.getToAccountNumber();
            if (!dao.validateAccount(fromAccountNumber) ||
!dao.validateAccount(toAccountNumber)) {
                    return "INVALID ACCOUNT NUMBER";
            }
            float balance = dao.findBalance(fromAccountNumber);
            if (balance < transferBean.getAmount()) {
                    return new InsufficientFundsException().toString();
            }
            boolean updateFromAccount = dao.updateBalance(fromAccountNumber, balance -
transferBean.getAmount());
            if (updateFromAccount) {
                    float newBalance = dao.findBalance(toAccountNumber);
                    boolean updateToAccount = dao.updateBalance(toAccountNumber,
newBalance + transferBean.getAmount());
                    if (updateToAccount) {
                            dao.transferMoney(transferBean);
                            System.out.println("Your Current " +
BankMain.checkBalance(fromAccountNumber));
                            return "TRANSACTION SUCCEEDED";
                    } else {
                            return "TRANSFER FAILED";
                    }
            } else {
                    return "TRANSACTION FAILED";}}}
```

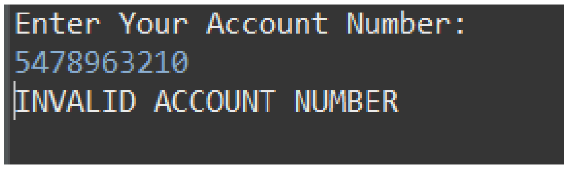
Output :



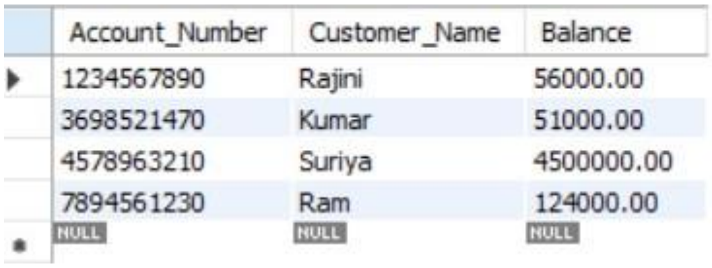
```
Enter Your Account Number:
1234567890
Balance: 81000.0
Press 1 To Continue with Transaction
1
Enter The Beneficiary Account Number:
7894561230
Enter Transaction Amount:
25000
Your Current Balance: 56000.0
TRANSACTION SUCCEEDED
```



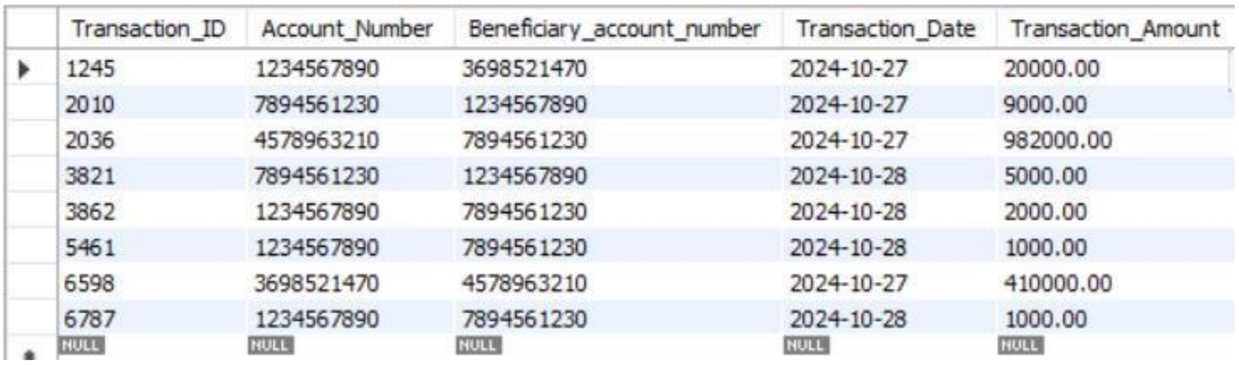
```
Enter Your Account Number:
7894561230
Balance: 174000.0
Press 1 To Continue with Transaction
1
Enter The Beneficiary Account Number:
3698521470
Enter Transaction Amount:
50000
Your Current Balance: 124000.0
TRANSACTION SUCCEEDED
```



```
Enter Your Account Number:
5478963210
INVALID ACCOUNT NUMBER
```



Account_Number	Customer_Name	Balance
1234567890	Rajini	56000.00
3698521470	Kumar	51000.00
4578963210	Suriya	4500000.00
7894561230	Ram	124000.00
NULL	NULL	NULL



Transaction_ID	Account_Number	Beneficiary_account_number	Transaction_Date	Transaction_Amount
1245	1234567890	3698521470	2024-10-27	20000.00
2010	7894561230	1234567890	2024-10-27	9000.00
2036	4578963210	7894561230	2024-10-27	982000.00
3821	7894561230	1234567890	2024-10-28	5000.00
3862	1234567890	7894561230	2024-10-28	2000.00
5461	1234567890	7894561230	2024-10-28	1000.00
6598	3698521470	4578963210	2024-10-27	410000.00
6787	1234567890	7894561230	2024-10-28	1000.00
NULL	NULL	NULL	NULL	NULL