Lab 6

Q 1) Using Spring framework deploy the banking system.

Java Code:

Beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <!-- Initialize for data source -->
   <bean id="datasource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
      <property name="url" value="jdbc:mysql://localhost:3306/ajp"/>
      <property name="username" value="root"/>
      <property name="password" value=""/>
   </bean>

   <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
      <property name="dataSource" ref="datasource"/>
   </bean>

   <!-- Definition for JDBCTemplate bean-->
   <bean id="bankJDBCTemplate" class="DataAccess.DataAccessTemplate">
      <property name="jdbcTemplateObject" ref="jdbcTemplate"/>
   </bean>
</beans>
```

Login.java

```
String uname = un.getText();
    String pass = pa.getText();

    if (uname.equals("admin") && pass.equals("admin")) {
        this.setVisible(false);
        new AdminMaster().setVisible(true);
    } else {

        Bank bank = new Bank();
        bank.setUsername(uname);
        bank.setPassword(pass);

        ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
        DataAccessTemplate dat = (DataAccessTemplate) context.getBean("bankJDBCTemplate");

        List<Bank> lst = dat.login(bank);
        if (lst.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Invalid Credentials! OR Admin has blocked You!");
        } else {
            JOptionPane.showMessageDialog(null, "Login Successful!");
            int a = lst.indexOf(bank);
            for (Bank b : lst) {
                bank.setUid(b.getUid());
                bank.setAccount_number(b.getAccount_number());
                bank.setBalance(b.getBalance());
            }
            this.setVisible(false);
            Dashboard db = new Dashboard(bank);
```

```
            db.setVisible(true);
        }
    }
```

Dashboard.java

```
nm = bank.getUsername();
    accn = bank.getAccount_number();
    bal = bank.getBalance();

    name.setText(nm);
    acc.setText(accn);
    balance.setText("" + bal);

    DefaultListModel model = new DefaultListModel();

    bank.setPay(bank.getUid());

    List<Bank> lst = dat.fetchpayTrans(bank, "payee");
    for (Bank b : lst) {
        model.addElement("You Payed " + b.getAmount() + " to " + b.getUsername() + "  on " +
b.getTimestamp() + " ");
    }

    lst = dat.fetchpayTrans(bank, "payer");
    for (Bank b : lst) {
        model.addElement("You received " + b.getAmount() + " from " + b.getUsername() + "  on " +
b.getTimestamp() + " ");
    }

    trans.setModel(model);
```

Bank.java

```
package BankPOJO;
```

```
/**
 *
 * @author Abhishek Karan
 */
public class Bank {

    private String username, password, account_number, timestamp;
    private double balance, amount;
    int uid, pay;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getAccount_number() {
        return account_number;
    }
```

```
public void setAccount_number(String account_number) {
    this.account_number = account_number;
}

public int getPay() {
    return pay;
}

public void setPay(int pay) {
    this.pay = pay;
}

public String getTimestamp() {
    return timestamp;
}

public void setTimestamp(String timestamp) {
    this.timestamp = timestamp;
}

public double getBalance() {
    return balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}

public double getAmount() {
```

```
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public int getUid() {
        return uid;
    }

    public void setUid(int uid) {
        this.uid = uid;
    }

}//class
```

DataAccessTemplate.java

```
package DataAccess;

import BankPOJO.Bank;
import Mappers.*;
import com.mysql.jdbc.PreparedStatement;
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 *
 * @author Abhishek Karan
 */
public class DataAccessTemplate {
```

```
    private JdbcTemplate jdbcTemplateObject;
    Bank bank = null;
    String query = "";

    public void setJdbcTemplateObject(JdbcTemplate jdbcTemplateObject) {
        this.jdbcTemplateObject = jdbcTemplateObject;
    }

    public List<Bank> login(Bank banks) {

        query = "select uid,uname,account_no,balance from bank_user where uname=? and upass=? and
status=? ";
        return jdbcTemplateObject.query(query, new Object[]{banks.getUsername(), banks.getPassword(),
1}, new LoginMapper());

    }//login()

    public int signUp(Bank banks) {

        if (!checkAccNo(banks).isEmpty()) {
            return -2;
        }
        if (!checkUname(banks).isEmpty()) {
            return -1;
        }

        query = "insert into bank_user(uname,upass,account_no) values(?,?,?)";
        return jdbcTemplateObject.update(query, new Object[]{banks.getUsername(), banks.getPassword(),
banks.getAccount_number()});
```

```
    }//signup()

    public List<Bank> checkAccNo(Bank bank) {

        query = "select uid from bank_user where account_no=? ";
        return jdbcTemplateObject.query(query, new Object[]{bank.getAccount_number()}, new
CheckMapper());

    }//checkAccno()

    public List<Bank> checkUname(Bank bank) {

        query = "select uid from bank_user where uname=? ";
        return jdbcTemplateObject.query(query, new Object[]{bank.getUsername()}, new CheckMapper());

    }//checkUname()

    public int updateAdmin(Bank bank, char stat) {

        query = "update bank_user set status=? where account_no=?";
        if (stat == 'G') {
            return jdbcTemplateObject.update(query, new Object[]{1, bank.getAccount_number()});
        } else if (stat == 'F') {
            return jdbcTemplateObject.update(query, new Object[]{0, bank.getAccount_number()});
        }
        return 0;
    }//UpdateAdmin()

    public List<Bank> fetchpayTrans(Bank bank, String pay) {
        query = "select bt.amount,bu.uname,bt.timestamp from bank_trans bt,bank_user bu "
```

```
            + "where bt.payer=bu.uid and " + pay + "=? ";
        return jdbcTemplateObject.query(query, new Object[]{bank.getPay()}, new TransactionsMapper());

    }//fetchPayTrans()

    public int updateAmount(Bank bank) {
        query = "update bank_user set balance=balance+? where account_no=?";
        if (jdbcTemplateObject.update(query, new Object[]{bank.getAmount(),
bank.getAccount_number()}) == 0) {
            return 0;
        } else {
            query = "update bank_user set balance=balance-? where uid=?";
            if (jdbcTemplateObject.update(query, new Object[]{bank.getAmount(), bank.getUid()}) == 0) {
                return 0;
            } else {
                return 1;
            }
        }
    }//updateAmt()

    public int updateBankTrans(Bank bank) {
        query = "select uid from bank_user where account_no=?";
        List<Bank> lst = jdbcTemplateObject.query(query, new Object[]{bank.getAccount_number()}, new
CheckMapper());
        int payer = 0;
        for (Bank b : lst) {
            payer = b.getUid();
        }

        query = "insert into bank_trans(payee,payer,amount) values(?,?,?)";
```

```
        if (jdbcTemplateObject.update(query, new Object[]{bank.getUid(), payer, bank.getAmount()}) == 0)
{
            return 0;
        }
        return 1;
    }//updatetrans()

}//DataAccessTemplate
```

TransactionsMapper.java

```
package Mappers;

import BankPOJO.Bank;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/**
 *
 * @author Abhishek Karan
 */
public class TransactionsMapper implements RowMapper<Bank> {

    @Override
    public Bank mapRow(ResultSet rs, int i) throws SQLException {

        Bank bank = new Bank();
        bank.setAmount(rs.getDouble(1));
        bank.setUsername(rs.getString(2));
        bank.setTimestamp(rs.getString(3));
        return bank;
```

```
    }//mapRow()

}//class
```

CheckMapper.java

```
package Mappers;

import BankPOJO.Bank;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/**
 *
 * @author Abhishek Karan
 */
public class CheckMapper implements RowMapper<Bank> {

    @Override
    public Bank mapRow(ResultSet rs, int i) throws SQLException {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        Bank bank = new Bank();
        bank.setUid(rs.getInt(1));
        return bank;
    }

}//checkMapper()
```

LoginMapper.java

```
package Mappers;

import BankPOJO.Bank;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/**
 *
 * @author Abhishek Karan
 */
public class LoginMapper implements RowMapper<Bank> {

    @Override
    public Bank mapRow(ResultSet rs, int i) throws SQLException {
        //throw new UnsupportedOperationException("Not supported yet.");
        Bank bank = new Bank();
        bank.setUid(rs.getInt(1));
        bank.setUsername(rs.getString(2));
        bank.setAccount_number(rs.getString(3));
        bank.setBalance(rs.getDouble(4));
        return bank;
    }

}//bankMapper
```

Abhishek Karan

130911122