JAVA SWING BASED – Insurance Management System - SQL CONNECTIVITY USING JDBC

A

Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

K. Sai Prasad <1602-19-737-095>

Under the guidance of Ms B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2020-2021

BONAFIDE CERTIFICATE

This is to certify that this project report titled
'Insurance Management System'
is a project work of **K Sai Prasad** bearing roll no. 1602-19-737-095 who carried out this project under my supervision in the IV semester for the academic year 2020- 2021

Signature
External Examiner

Signature
Internal Examiner

ABSTRACT

The insurance management project deals the adding new insurance schemes and managing the clients for the insurance. The project has complete access for the crud operations that are to create, read, update and delete the database entries. At first you need to add a branch and the staff members for the branch then secondly add a user to the database now you can add an insurance scheme and finally make the payments for the client to the added insurance.

REQUIREMENT ANALYSIS:

List of Tables:

- Branch
- Employee
- Client
- Insurance
- Payment

List of attributes with their domain types:

Branch:

- branch_code:varchar2(50)
- branch_name:varchar2(50)
- branch_address:varchar2(150)

Employee:

- employee_id: number(5)
- employee _name: varchar2(50)
- employee _address: varchar2(150)
- employee _contact: varchar2(15)

Client:

- client_id: number(5)
- client_name: varchar2(50)
- client_contact: varchar2(15)
- client_address: varchar2(150)

Insurance:

- insurance_num: number(5)
- insurance_type: varchar2(50)
- insurance_amount: number(5)
- pay_per_month: number(5)
- start_date: DATE
- end_date: DATE

Payment:

- payment_id
- client_id: number(5)
- insurance_num:number(5)
- amount:number(5)
- date_of_payment:DATE

AIM AND PRIORITY OF THE PROJECT

To create a Java GUI-based desktop application that connects people looking for insurance with employees looking for clients . It takes values like client name, employee name, branch name, etc through forms which are then updated in the database using JDBC connectivity.

ARCHITECTURE AND TECHNOLOGY

Software used: Java Eclipse, Oracle 11g Database, Java SE version 13, SQL*Plus.

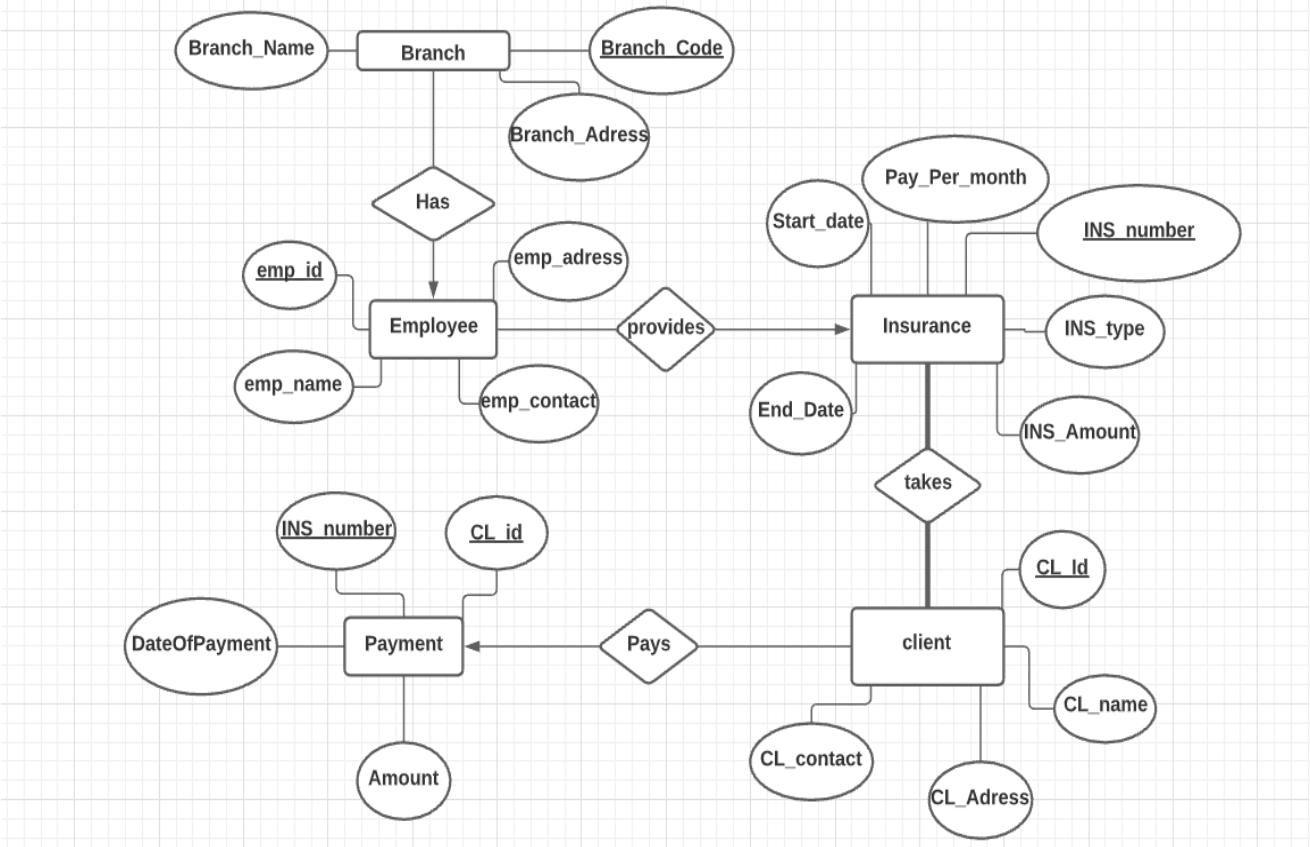
Java SWING: Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

SQL: Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

DESIGN

Entity relationship diagram



DATABASE DESIGN

MAPPING CARDINALITY AND PARTICIPATION CONSTRAINTS

branch-employee:

A branch have many employees but an employee can be in only one branch so, ONE TO MANY relationship

Employee-Insurance:

One employee can provide many insurances , so

ONE TO MANY relationship

Client-insurance :

A client can take many insurances and one insurance contains many clients so, MANY TO MANY relationship

Client-payment:

A client can have many payments and one payment is associated with one client so, ONE TO MANY

DDL COMMANDS:

1.create table branch(

branch_code varchar2(5),

branch_name varchar2(10) NOT NULL,

branch_address varchar2(5) NOT NULL,

CONSTRAINT pk_branch_code PRIMARY KEY (branch_code));

2.create table employee(

employee_id number(5),

employee_name varchar2(10) NOT NULL,

employee_contact varchar2(10) NOT NULL,

employee_address varchar2(5) NOT NULL,

branch_code varchar2(5) NOT NULL,

CONSTRAINT fk_employee_bc FOREIGN KEY (branch_code) REFERENCES branch(branch_code) ON DELETE CASCADE,

CONSTRAINT pk_emp_id PRIMARY KEY (employee_id)) ;

3. create table client(

client_id number(5),

client_name varchar2(10) NOT NULL,

client_contact varchar2(10) NOT NULL,

client_address varchar2(10) NOT NULL,

CONSTRAINT pk_cl_id PRIMARY KEY (client_id));

4. create table insurance(

insurance_num number(5),

insurance_type varchar2(10) NOT NULL,

client_id number(5) NOT NULL,

insurance_amount number(7) NOT NULL,

pay_per_month number(5) NOT NULL,

start_date DATE NOT NULL,

end_date DATE NOT NULL,

branch_code varchar2(10) NOT NULL,

employee_id number(5) NOT NULL,

CONSTRAINT fk_insurance_ci FOREIGN KEY (client_id) REFERENCES client(client_id) ON DELETE CASCADE,

CONSTRAINT fk_insurance_bc FOREIGN KEY (branch_code) REFERENCES branch(branch_code) ON DELETE CASCADE,

CONSTRAINT fk_insurance_ei FOREIGN KEY (employee_id) REFERENCES employee(employee_id) ON DELETE CASCADE,

CONSTRAINT pk_ insurance_num PRIMARY KEY (insurance_num));

5. create table payment(

Payment_id(5) NOT NULL,

client_id number(5) NOT NULL,

insurnace_num number(5) NOT NULL,

amount number(5) NOT NULL,

date_of_payment DATE NOT NULL,

CONSTRAINT fk_payment_ci FOREIGN KEY (client_id) REFERENCES client(client_id) ON DELETE CASCADE,

CONSTRAINT fk_payment_in FOREIGN KEY (insurance_num) REFERENCES insurance(insurance_num) ON DELETE CASCADE,

CONSTRAINT pk_payment_pk PRIMARY KEY (payment_id));

```
SQL> client;
SP2-0042: unknown command "client" - rest of line ignored.
SQL> desc branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_CODE                               NOT NULL VARCHAR2(5)
 BRANCH_NAME                               NOT NULL VARCHAR2(10)
 BRANCH_ADDRESS                            NOT NULL VARCHAR2(5)

SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                               NOT NULL NUMBER(5)
 EMPLOYEE_NAME                             NOT NULL VARCHAR2(10)
 EMPLOYEE_CONTACT                          NOT NULL VARCHAR2(10)
 EMPLOYEE_ADDRESS                          NOT NULL VARCHAR2(5)
 BRANCH_CODE                               NOT NULL VARCHAR2(5)

SQL> desc client;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CLIENT_ID                                 NOT NULL NUMBER(5)
 CLIENT_NAME                               NOT NULL VARCHAR2(10)
 CLIENT_CONTACT                            NOT NULL VARCHAR2(10)
 CLIENT_ADDRESS                            NOT NULL VARCHAR2(10)
```

```
SQL> desc insurance;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSURANCE_NUM                             NOT NULL NUMBER(5)
 INSURANCE_TYPE                            NOT NULL VARCHAR2(10)
 CLIENT_ID                                 NOT NULL NUMBER(5)
 INSURANCE_AMOUNT                          NOT NULL NUMBER(7)
 PAY_PER_MONTH                             NOT NULL NUMBER(5)
 START_DATE                                NOT NULL DATE
 END_DATE                                  NOT NULL DATE
 BRANCH_CODE                               NOT NULL VARCHAR2(10)
 EMPLOYEE_ID                               NOT NULL NUMBER(5)

SQL> desc payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENT_ID                                NOT NULL NUMBER(5)
 CLIENT_ID                                 NOT NULL NUMBER(5)
 INSURANCE_NUM                             NOT NULL NUMBER(5)
 AMOUNT                                    NOT NULL NUMBER(7)
 DATE_OF_PAYMENT                           NOT NULL DATE
```

DML COMMANDS

1.insert into branch values('&branch_code','&branch_name','&branch_address');

```
SQL> insert into branch values('&branch_code','&branch_name','&branch_address');
Enter value for branch_code: 101
Enter value for branch_name: delhi
Enter value for branch_address: xyz1
old   1: insert into branch values('&branch_code','&branch_name','&branch_address')
new   1: insert into branch values('101','delhi','xyz1')

1 row created.

SQL> /
Enter value for branch_code: 102
Enter value for branch_name: hyderabad
Enter value for branch_address: xyz2
old   1: insert into branch values('&branch_code','&branch_name','&branch_address')
new   1: insert into branch values('102','hyderabad','xyz2')

1 row created.

SQL> /
Enter value for branch_code: 103
Enter value for branch_name: mumbai
Enter value for branch_address: xyz3
old   1: insert into branch values('&branch_code','&branch_name','&branch_address')
new   1: insert into branch values('103','mumbai','xyz3')

1 row created.

SQL> /
Enter value for branch_code: 104
Enter value for branch_name: kolkata
Enter value for branch_address: xyz4
old   1: insert into branch values('&branch_code','&branch_name','&branch_address')
new   1: insert into branch values('104','kolkata','xyz4')

1 row created.

SQL> /
Enter value for branch_code: 105
Enter value for branch_name: chennai
Enter value for branch_address: xyz5
old   1: insert into branch values('&branch_code','&branch_name','&branch_address')
new   1: insert into branch values('105','chennai','xyz5')

1 row created.
```

```
SQL> select * from branch;

BRANC BRANCH_NAM BRANC
----- ---------- -----
101   delhi      xyz1
102   hyderabad  xyz2
103   mumbai     xyz3
104   kolkata    xyz4
105   chennai    xyz5
```

2.insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code');

```
SQL> insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code');
Enter value for emp_id: 201
Enter value for emp_name: Arpith
Enter value for emp_contact: 9999999991
Enter value for emp_address: abc1
Enter value for branch_code: 101
old   1: insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code')
new   1: insert into employee values(201,'Arpith','9999999991','abc1','101')

1 row created.

SQL> /
Enter value for emp_id: 202
Enter value for emp_name: nikhil
Enter value for emp_contact: 9999999992
Enter value for emp_address: abc2
Enter value for branch_code: 105
old   1: insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code')
new   1: insert into employee values(202,'nikhil','9999999992','abc2','105')

1 row created.

SQL> /
Enter value for emp_id: 203
Enter value for emp_name: suraj
Enter value for emp_contact: 9999999993
Enter value for emp_address: abc3
Enter value for branch_code: 101
old   1: insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code')
new   1: insert into employee values(203,'suraj','9999999993','abc3','101')

1 row created.
```

```
SQL> /
Enter value for emp_id: 204
Enter value for emp_name: krishna
Enter value for emp_contact: 9999999994
Enter value for emp_address: abc4
Enter value for branch_code: 102
old   1: insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code')
new   1: insert into employee values(204,'krishna','9999999994','abc4','102')

1 row created.

SQL> /
Enter value for emp_id: 205
Enter value for emp_name: vijay
Enter value for emp_contact: 9999999995
Enter value for emp_address: abc5
Enter value for branch_code: 105
old   1: insert into employee values(&emp_id,'&emp_name','&emp_contact','&emp_address','&branch_code')
new   1: insert into employee values(205,'vijay','9999999995','abc5','105')

1 row created.

SQL> select * from employee;

    EMP_ID EMP_NAME   EMP_CONTAC EMP_A BRANC
---------- ---------- ---------- ----- -----
       201 Arpith     9999999991 abc1  101
       202 nikhil     9999999992 abc2  105
       203 suraj      9999999993 abc3  101
       204 krishna    9999999994 abc4  102
       205 vijay      9999999995 abc5  105
```

3. insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address');

```
SQL> /
Enter value for cl_id: 305
Enter value for cl_name: tharun
Enter value for cl_contact: 8888888885
Enter value for cl_address: pqr5
old   1: insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address')
new   1: insert into client values(305,'tharun','8888888885','pqr5')

1 row created.

SQL> select
  2
SQL> select * from client;

     CL_ID CL_NAME    CL_CONTACT CL_ADDRESS
---------- ---------- ---------- ----------
       301 salman     8888888881 pqr1
       302 nanda      8888888882 pqr2
       303 kashif     8888888883 pqr3
       304 gautam     8888888884 pqr4
       305 tharun     8888888885 pqr5
```

```
SQL> insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address');
Enter value for cl_id: 301
Enter value for cl_name: salman
Enter value for cl_contact: 8888888881
Enter value for cl_address: pqr1
old   1: insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address')
new   1: insert into client values(301,'salman','8888888881','pqr1')

1 row created.

SQL> /
Enter value for cl_id: 302
Enter value for cl_name: nanda
Enter value for cl_contact: 8888888882
Enter value for cl_address: pqr2
old   1: insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address')
new   1: insert into client values(302,'nanda','8888888882','pqr2')

1 row created.

SQL> /
Enter value for cl_id: 303
Enter value for cl_name: kashif
Enter value for cl_contact: 8888888883
Enter value for cl_address: pqr3
old   1: insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address')
new   1: insert into client values(303,'kashif','8888888883','pqr3')

1 row created.

SQL> /
Enter value for cl_id: 304
Enter value for cl_name: gautam
Enter value for cl_contact: 8888888884
Enter value for cl_address: pqr4
old   1: insert into client values(&cl_id,'&cl_name','&cl_contact','&cl_address')
new   1: insert into client values(304,'gautam','8888888884','pqr4')

1 row created.
```

4. Insert into insurance values (&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id);

```
SQL> insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id);
Enter value for ins_num: 401
Enter value for ins_type: health
Enter value for cl_id: 301
Enter value for ins_amount: 1000000
Enter value for pay_per_month: 5000
Enter value for start_date: TO_DATE('8/11/2018','DD/MM/YY')
Enter value for end_date: TO_DATE('8/11/2024','DD/MM/YY')
Enter value for branch_code: 101
Enter value for emp_id: 201
old   1: insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id)
new   1: insert into insurance values(401,'health',301,1000000,5000,TO_DATE('8/11/2018','DD/MM/YY'),TO_DATE('8/11/2024','DD/MM/YY'),'101',201)

1 row created.

SQL> select * from insurance;

   INS_NUM INS_TYPE        CL_ID INS_AMOUNT PAY_PER_MONTH START_DAT END_DATE
---------- ---------- ---------- ---------- ------------ --------- ---------
BRANCH_COD     EMP_ID
---------- ----------
       401 health            301    1000000         5000 08-NOV-18 08-NOV-24
101               201


SQL> insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id);
Enter value for ins_num: 402
Enter value for ins_type: life
Enter value for cl_id: 302
Enter value for ins_amount: 2000000
Enter value for pay_per_month: 6000
Enter value for start_date: TO_DATE('10/05/2010','DD/MM/YY')
Enter value for end_date: TO_DATE('10/05/2030','DD/MM/YY')
Enter value for branch_code: 102
Enter value for emp_id: 202
old   1: insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id)
new   1: insert into insurance values(402,'life',302,2000000,6000,TO_DATE('10/05/2010','DD/MM/YY'),TO_DATE('10/05/2030','DD/MM/YY'),'102',202)

1 row created.
```

```
SQL> insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id);
Enter value for ins_num: 403
Enter value for ins_type: vehicle
Enter value for cl_id: 303
Enter value for ins_amount: 200000
Enter value for pay_per_month: 2000
Enter value for start_date: TO_DATE('30/06/2020','DD/MM/YY')
Enter value for end_date: TO_DATE('30/06/2024','DD/MM/YY')
Enter value for branch_code: 103
Enter value for emp_id: 203
old   1: insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id)
new   1: insert into insurance values(403,'vehicle',303,200000,2000,TO_DATE('30/06/2020','DD/MM/YY'),TO_DATE('30/06/2024','DD/MM/YY'),'103',203)

1 row created.

SQL> /
Enter value for ins_num: 404
Enter value for ins_type: life
Enter value for cl_id: 304
Enter value for ins_amount: 5000000
Enter value for pay_per_month: 10000
Enter value for start_date: TO_DATE('30/08/2000','DD/MM/YY')
Enter value for end_date: TO_DATE('30/08/2030','DD/MM/YY')
Enter value for branch_code: 104
Enter value for emp_id: 204
old   1: insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id)
new   1: insert into insurance values(404,'life',304,5000000,10000,TO_DATE('30/08/2000','DD/MM/YY'),TO_DATE('30/08/2030','DD/MM/YY'),'104',204)

1 row created.

SQL> /
Enter value for ins_num: 405
Enter value for ins_type: property
Enter value for cl_id: 305
Enter value for ins_amount: 1500000
Enter value for pay_per_month: 8000
Enter value for start_date: TO_DATE('13/07/2016','DD/MM/YY')
Enter value for end_date: TO_DATE('13/07/2027','DD/MM/YY')
Enter value for branch_code: 105
Enter value for emp_id: 205
old   1: insert into insurance values(&ins_num,'&ins_type',&cl_id,&ins_amount,&pay_per_month,&start_date,&end_date,'&branch_code',&emp_id)
new   1: insert into insurance values(405,'property',305,1500000,8000,TO_DATE('13/07/2016','DD/MM/YY'),TO_DATE('13/07/2027','DD/MM/YY'),'105',205)
```

```
1 row created.

SQL> select * from insurance;

   INS_NUM INS_TYPE        CL_ID INS_AMOUNT PAY_PER_MONTH START_DAT END_DATE
---------- ---------- ---------- ---------- ------------- --------- ---------
BRANCH_COD     EMP_ID
---------- ----------
       401 health            301    1000000          5000 08-NOV-18 08-NOV-24
101               201

       402 life              302    2000000          6000 10-MAY-10 10-MAY-30
102               202

       403 vehicle           303     200000          2000 30-JUN-20 30-JUN-24
103               203


   INS_NUM INS_TYPE        CL_ID INS_AMOUNT PAY_PER_MONTH START_DAT END_DATE
---------- ---------- ---------- ---------- ------------- --------- ---------
BRANCH_COD     EMP_ID
---------- ----------
       404 life              304    5000000         10000 30-AUG-00 30-AUG-30
104               204

       405 property          305    1500000          8000 13-JUL-16 13-JUL-27
105               205
```

5.insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment);

```
SQL> insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment);
Enter value for cl_id: 301
Enter value for ins_num: 401
Enter value for amount: 5000
Enter value for dateofpayment: TO_DATE('03/05/2021','DD/MM/YY')
old   1: insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment)
new   1: insert into payment values(301,401,5000,TO_DATE('03/05/2021','DD/MM/YY'))

1 row created.

SQL> /
Enter value for cl_id: 302
Enter value for ins_num: 402
Enter value for amount: 6000
Enter value for dateofpayment: TO_DATE('01/03/2021','DD/MM/YY')
old   1: insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment)
new   1: insert into payment values(302,402,6000,TO_DATE('01/03/2021','DD/MM/YY'))

1 row created.

SQL> /
Enter value for cl_id: 303
Enter value for ins_num: 403
Enter value for amount: 2000
Enter value for dateofpayment: TO_DATE('20/09/2020','DD/MM/YY')
old   1: insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment)
new   1: insert into payment values(303,403,2000,TO_DATE('20/09/2020','DD/MM/YY'))

1 row created.

SQL> /
Enter value for cl_id: 304
Enter value for ins_num: 404
Enter value for amount: 10000
Enter value for dateofpayment: TO_DATE('21/03/2021','DD/MM/YY')
old   1: insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment)
new   1: insert into payment values(304,404,10000,TO_DATE('21/03/2021','DD/MM/YY'))

1 row created.
```

```
SQL> /
Enter value for cl_id: 305
Enter value for ins_num: 405
Enter value for amount: 8000
Enter value for dateofpayment: TO_DATE('15/04/2021','DD/MM/YY')
old   1: insert into payment values(&cl_id,&ins_num,&amount,&dateofpayment)
new   1: insert into payment values(305,405,8000,TO_DATE('15/04/2021','DD/MM/YY'))

1 row created.

SQL> select * from payment;

     CL_ID    INS_NUM     AMOUNT DATEOFPAY
---------- ---------- ---------- ---------
       301        401       5000 03-MAY-21
       302        402       6000 01-MAR-21
       303        403       2000 20-SEP-20
       304        404      10000 21-MAR-21
       305        405       8000 15-APR-21
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
public void connectDatabase()
      {
            try
            {
                  Connection con=DriverManager.getConnection(

      "jdbc:oracle:thin:@localhost:1521:xe","it19737095","vasavi");

                  stmt=con.createStatement();
                  stmt.executeUpdate("commit");

            }
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
}
```

Thus, the connection from Java to Oracle database is performed and therefore, can be used for

updating tables in the database directly.

Branch table:

```
//K.SAI PRASAD 1602-19-737-095 INSURANCE MANAGEMENT SYSTEM
package IMS;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.Properties;

public class branch
{
    private JPanel pn,pn1,pn2,pn3;
    private JFrame jframe;
    private JButton
JB_insert,JB_modify,JB_view,JB_delete;
    private JLabel
JL_branch_code,JL_branch_name,JL_branch_address;
    private JTextField
JTF_branch_code,JTF_branch_name,JTF_branch_address;
    Connection con;
    ResultSet rs;
    Statement stmt;
    private JMenuItem insert1,update1,view1,delete1;
    private List branchList;

    public branch(JPanel pn,JFrame jframe,JMenuItem
insert1,JMenuItem update1,JMenuItem view1,JMenuItem
delete1)
    {
        try
        {
```

```
    Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and
load driver");
            System.exit(1);
        }
        connectDatabase();

        this.jframe=jframe;
        this.insert1=insert1;
        this.update1=update1;
        this.view1=view1;
        this.delete1=delete1;

        JL_branch_code=new JLabel("Branch_Code:");
        JTF_branch_code=new JTextField(10);
        JL_branch_name=new JLabel("Branch Name:");
        JTF_branch_name=new JTextField(10);
        JL_branch_address=new JLabel("Branch
Address:");
        JTF_branch_address=new JTextField(10);

        this.pn=pn;

    }
    public void connectDatabase()
    {
        try
        {
            Connection
con=DriverManager.getConnection(
```

```
    "jdbc:oracle:thin:@localhost:1521:xe","it19737095
","vasavi");

            stmt=con.createStatement();
            stmt.executeUpdate("commit");

        }
        catch (SQLException connectException)
        {

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());
          System.exit(1);
        }
    }
    private void displaySQLErrors(SQLException e)
    {

    JOptionPane.showMessageDialog(pn1,"\nSQLException
: " + e.getMessage() + "\n"+"SQLState:     " +
e.getSQLState() + "\n"+"VendorError:  " +
e.getErrorCode() + "\n");
    }
    public void loadbranch()
    {
        try
        {
            branchList=new List();
            branchList.removeAll();
            rs=stmt.executeQuery("select * from
branch");
```

```
            while(rs.next())
            {

    branchList.add(rs.getString("branch_code"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void buildGUI()
    {

        insert1.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_insert=new JButton("Submit");

                JTF_branch_code.setText(null);
                JTF_branch_name.setText(null);
                JTF_branch_address.setText(null);

                loadbranch();

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                pn1=new JPanel();
```

```
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_branch_code);
                pn1.add(JTF_branch_code);
                pn1.add(JL_branch_name);
                pn1.add(JTF_branch_name);
                pn1.add(JL_branch_address);
                pn1.add(JTF_branch_address);

                pn3=new JPanel(new FlowLayout());
                pn3.add(JB_insert);
                pn1.setBounds(115,80,300,250);
                pn3.setBounds(200,350,75,35);

                pn2=new JPanel(new FlowLayout());
                branchList=new List(10);
                loadbranch();
                pn2.add(branchList);
                pn2.setBounds(200,350,300,180);

                pn.add(pn1);
                pn.add(pn3);
                pn.add(pn2);

                pn.setLayout(new BorderLayout());
                jframe.add(pn);
                jframe.setSize(800,800);
                jframe.validate();

                JB_insert.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
```

```
                        try
                        {
                            String query= "INSERT
INTO branch VALUES(" + JTF_branch_code.getText() +
","
                            + "'"
+JTF_branch_name.getText() +"'," +
"'"+JTF_branch_address.getText() +"')";
                            int i =
stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nInserted
"+i+" rows successfully");
                            loadbranch();

    System.out.println("Done");
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        update1.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_modify=new JButton("Modify");

                JTF_branch_code.setText(null);
                JTF_branch_name.setText(null);
```

```
                    JTF_branch_address.setText(null);

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_branch_code);
                    pn1.add(JTF_branch_code);
                    pn1.add(JL_branch_name);
                    pn1.add(JTF_branch_name);
                    pn1.add(JL_branch_address);
                    pn1.add(JTF_branch_address);

                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_modify);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    branchList=new List(10);
                    loadbranch();
                    pn2.add(branchList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
```

```
                jframe.validate();

                branchList.addItemListener(new
ItemListener() {
                    public void
itemStateChanged(ItemEvent ievt)
                    {
                        try
                        {

    rs=stmt.executeQuery("select * from branch");
                            while (rs.next())
                            {
                                if
(rs.getString("branch_code").equals(branchList.getSel
ectedItem()))
                                break;
                            }
                            if (!rs.isAfterLast())
                            {

    JTF_branch_code.setText(rs.getString("branch_code
"));

    JTF_branch_name.setText(rs.getString("branch_name
"));

    JTF_branch_address.setText(rs.getString("branch_a
ddress"));

                            }
                        }
                        catch (SQLException
selectException)
                        {
```

```
    displaySQLErrors(selectException);
                        }
                    }
                });
                JB_modify.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        try
                        {
                            int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to update:");

    if(a==JOptionPane.YES_OPTION)
                            {
                                String
pack=JOptionPane.showInputDialog(pn,"Enter New Branch
Name:");

    JTF_branch_name.setText(pack);
                                String query="update
branch set branch_name='"+pack+"' where
branch_code="+JTF_branch_code.getText();
                                int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nUpdated "+i+"
rows succesfully");
                                loadbranch();
                            }
                        }
```

```
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        delete1.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_delete=new JButton("Delete");

                JTF_branch_code.setText(null);
                JTF_branch_name.setText(null);
                JTF_branch_address.setText(null);

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                pn1=new JPanel();
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_branch_code);
                pn1.add(JTF_branch_code);
                pn1.add(JL_branch_name);
                pn1.add(JTF_branch_name);
                pn1.add(JL_branch_address);
                pn1.add(JTF_branch_address);
```

```
                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_delete);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    branchList=new List(10);
                    loadbranch();
                    pn2.add(branchList);
                    pn2.setBounds(200,350,300,200);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    branchList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from branch");
                                while (rs.next())
                                {
                                    if
(rs.getString("branch_code").equals(branchList.getSel
ectedItem()))
                                        break;
```

```
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_branch_code.setText(rs.getString("branch_code
"));

    JTF_branch_name.setText(rs.getString("branch_name
"));

    JTF_branch_address.setText(rs.getString("branch_a
ddress"));

                                }
                            }
                            catch (SQLException
selectException)
                            {

    displaySQLErrors(selectException);
                            }
                        }
                    });
                    JB_delete.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                             try
                             {
                                int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to Delete:");
```

```
    if(a==JOptionPane.YES_OPTION)
                                {
                                    //String
query="DELETE FROM branch WHERE
branch_code="+branchList.getSelectedItem();
                                    String query="DELETE
FROM branch WHERE
branch_code="+JTF_branch_code.getText();
                                    int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nDeleted "+i+"
rows succesfully");
                                    loadbranch();
                                }
                             }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
            });
            view1.addActionListener(new ActionListener()
{
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();
```

```
                JLabel view=new JLabel("Branch
View");
                JB_view=new JButton("View");
                Font myFont = new
Font("Serif",Font.BOLD,50);
                view.setFont((myFont));

                pn1=new JPanel();
                pn2=new JPanel();
                pn1.add(view);
                pn2.add(JB_view);
                pn.add(pn1);
                pn.add(pn2);
                pn.setBounds(500,800,300,300);
                pn.setLayout(new FlowLayout());

                jframe.add(pn);
                jframe.setSize(800,800);
                jframe.validate();

                JB_view.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        JFrame jf=new JFrame("Branch
Details");
                        JTable jt;
                        DefaultTableModel model =
new DefaultTableModel();
                        jt = new JTable(model);

model.addColumn("branch_code");
```

```
                        model.addColumn("branch
name");
                        model.addColumn("branch
address");
                        try
                        {

    rs=stmt.executeQuery("select * from branch");
                            while(rs.next())
                            {
                                model.addRow(new
Object[]{rs.getInt(1),

rs.getString(2),rs.getString(3)});
                            }
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                        jt.setEnabled(false);
                        jt.setBounds(30, 40, 300,
300);
                        JScrollPane jsp = new
JScrollPane(jt);
                        jf.add(jsp);
                        jf.setSize(800, 400);
                        jf.setVisible(true);
                    }
                });
            }
        });
    }
}
```

Employee table:

```
//K.SAI PRASAD 1602-19-737-095 INSURANCE MANAGEMENT
SYSTEM
package IMS;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.Properties;

public class employee
{
    private JPanel pn,pn1,pn2,pn3;
    private JFrame jframe;
    private JButton
JB_insert,JB_modify,JB_view,JB_delete;
    private JLabel
JL_employee_id,JL_employee_name,JL_employee_contact,J
L_employee_address,JL_branch_code;
    private JTextField
JTF_employee_id,JTF_employee_name,JTF_employee_contac
t,JTF_employee_address,JTF_branch_code;
    Connection con;
    ResultSet rs;
    Statement stmt;
    private JMenuItem insert2,update2,view2,delete2;
    private List employeeList;
    private Choice branchCode;

    public employee(JPanel pn,JFrame jframe,JMenuItem
insert2,JMenuItem update2,JMenuItem view2,JMenuItem
delete2)
    {
```

```
        try
        {

    Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and
load driver");
            System.exit(1);
        }
        connectDatabase();

        this.jframe=jframe;
        this.insert2=insert2;
        this.update2=update2;
        this.view2=view2;
        this.delete2=delete2;

        JL_employee_id=new JLabel("employee_id:");
        JTF_employee_id=new JTextField(10);
        JL_employee_name=new JLabel("employee
Name:");
        JTF_employee_name=new JTextField(10);
        JL_employee_contact=new JLabel("employee
Contact:");
        JTF_employee_contact=new JTextField(10);
        JL_employee_address=new JLabel("employee
Address:");
        JTF_employee_address=new JTextField(10);
        JL_branch_code=new JLabel("Branch Code:");
        branchCode=new Choice();
        JTF_branch_code=new JTextField(10);

        this.pn=pn;
```

```
    }
    public void connectDatabase()
    {
        try
        {
            Connection
con=DriverManager.getConnection(

    "jdbc:oracle:thin:@localhost:1521:xe","it19737095
","vasavi");

            stmt=con.createStatement();
            stmt.executeUpdate("commit");

        }
        catch (SQLException connectException)
        {

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());
          System.exit(1);
        }
    }
    private void displaySQLErrors(SQLException e)
    {

    JOptionPane.showMessageDialog(pn1,"\nSQLException
: " + e.getMessage() + "\n"+"SQLState:     " +
e.getSQLState() + "\n"+"VendorError:  " +
e.getErrorCode() + "\n");
    }
```

```
    public void loadbranch()
    {
        try
        {
            //branchCode=new Choice();
            branchCode.removeAll();
            rs=stmt.executeQuery("select * from
branch");
            while(rs.next())
            {

    branchCode.add(rs.getString("branch_code"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void loademployee()
    {
        try
        {
            employeeList=new List();
            employeeList.removeAll();
            rs=stmt.executeQuery("select * from
employee");
            while(rs.next())
            {

    employeeList.add(rs.getString("employee_id"));
            }
        }
        catch(SQLException e)
        {
```

```
                displaySQLErrors(e);
            }
        }
        public void buildGUI()
        {

            insert2.addActionListener(new
ActionListener() {
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    JB_insert=new JButton("Submit");
                    loadbranch();

                    JTF_employee_id.setText(null);
                    JTF_employee_name.setText(null);
                    JTF_employee_contact.setText(null);
                    JTF_employee_address.setText(null);
                    //JTF_branch_code.setText(null);

                    loademployee();

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_employee_id);
                    pn1.add(JTF_employee_id);
                    pn1.add(JL_employee_name);
                    pn1.add(JTF_employee_name);
```

```
                    pn1.add(JL_employee_contact);
                    pn1.add(JTF_employee_contact);
                    pn1.add(JL_employee_address);
                    pn1.add(JTF_employee_address);
                    pn1.add(JL_branch_code);
                    pn1.add(branchCode);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_insert);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    employeeList=new List(10);
                    loademployee();
                    pn2.add(employeeList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    JB_insert.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                            try
```

```
                            {
                                String query= "INSERT
INTO employee VALUES(" + JTF_employee_id.getText() +
","
                                + "'"
+JTF_employee_name.getText() +"',"
+"'"+JTF_employee_contact.getText() +"'," +
"'"+JTF_employee_address.getText()
+"','"+"'"+branchCode.getSelectedItem()+"')";
                                int i =
stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nInserted
"+i+" rows successfully");
                                loademployee();

    System.out.println("Done");
                            }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
            });
            update2.addActionListener(new
ActionListener() {
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    JB_modify=new JButton("Modify");

                    JTF_employee_id.setText(null);
```

```
                    JTF_employee_name.setText(null);
                    JTF_employee_contact.setText(null);
                    JTF_employee_address.setText(null);
                    JTF_branch_code.setText(null);

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_employee_id);
                    pn1.add(JTF_employee_id);
                    pn1.add(JL_employee_name);
                    pn1.add(JTF_employee_name);
                    pn1.add(JL_employee_contact);
                    pn1.add(JTF_employee_contact);
                    pn1.add(JL_employee_address);
                    pn1.add(JTF_employee_address);
                    pn1.add(JL_branch_code);
                    pn1.add(JTF_branch_code);

                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_modify);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    employeeList=new List(10);
                    loademployee();
                    pn2.add(employeeList);
                    pn2.setBounds(200,350,300,180);
```

```
                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    employeeList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from employee");
                                while (rs.next())
                                {
                                    if
(rs.getString("employee_id").equals(employeeList.getS
electedItem()))
                                    break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_employee_id.setText(rs.getString("employee_id
"));

    JTF_employee_name.setText(rs.getString("employee_
name"));
```

```
    JTF_employee_contact.setText(rs.getString("employ
ee_contact"));

    JTF_employee_address.setText(rs.getString("employ
ee_address"));

    JTF_branch_code.setText(rs.getString("branch_code
"));

                                }
                            }
                            catch (SQLException
selectException)
                            {

    displaySQLErrors(selectException);
                            }
                        }
                    });
                    JB_modify.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                             try
                             {
                                int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to update:");

    if(a==JOptionPane.YES_OPTION)
                                {
```

```
                                    String
pack=JOptionPane.showInputDialog(pn,"Enter New
employee Name:");

    JTF_employee_name.setText(pack);
                                    String query="update
employee set employee_name='"+pack+"' where
employee_id="+JTF_employee_id.getText();
                                    int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nUpdated "+i+"
rows succesfully");
                                    loademployee();
                                }
                             }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
            });
            delete2.addActionListener(new
ActionListener() {
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    JB_delete=new JButton("Delete");

                    JTF_employee_id.setText(null);
                    JTF_employee_name.setText(null);
                    JTF_employee_contact.setText(null);
```

```
                    JTF_employee_address.setText(null);
                    JTF_branch_code.setText(null);

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_employee_id);
                    pn1.add(JTF_employee_id);
                    pn1.add(JL_employee_name);
                    pn1.add(JTF_employee_name);
                    pn1.add(JL_employee_contact);
                    pn1.add(JTF_employee_contact);
                    pn1.add(JL_employee_address);
                    pn1.add(JTF_employee_address);
                    pn1.add(JL_branch_code);
                    pn1.add(JTF_branch_code);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_delete);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    employeeList=new List(10);
                    loademployee();
                    pn2.add(employeeList);
                    pn2.setBounds(200,350,300,200);

                    pn.add(pn1);
```

```
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    employeeList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from employee");
                                while (rs.next())
                                {
                                    if
(rs.getString("employee_id").equals(employeeList.getS
electedItem()))
                                     break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_employee_id.setText(rs.getString("employee_id
"));

    JTF_employee_name.setText(rs.getString("employee_
name"));

    JTF_employee_contact.setText(rs.getString("employ
ee_contact"));
```

```
    JTF_employee_address.setText(rs.getString("employ
ee_address"));

    JTF_branch_code.setText(rs.getString("branch_code
"));

                                }
                            }
                            catch (SQLException
selectException)
                            {

    displaySQLErrors(selectException);
                                }
                        }
                    });
                    JB_delete.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                             try
                             {
                                int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to Delete:");

    if(a==JOptionPane.YES_OPTION)
                                {
                                    //String
query="DELETE FROM employee WHERE
employee_id="+employeeList.getSelectedItem();
```

```
                                        String query="DELETE
FROM employee WHERE
employee_id="+JTF_employee_id.getText();
                                        int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nDeleted "+i+"
rows succesfully");
                                        loademployee();
                                    }
                                 }
                                catch(SQLException e)
                                {
                                    displaySQLErrors(e);
                                }
                            }
                        });
                    }
            });
            view2.addActionListener(new ActionListener()
{
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    JLabel view=new JLabel("employee
View");
                    JB_view=new JButton("View");
                    Font myFont = new
Font("Serif",Font.BOLD,50);
```

```
                    view.setFont((myFont));

                    pn1=new JPanel();
                    pn2=new JPanel();
                    pn1.add(view);
                    pn2.add(JB_view);
                    pn.add(pn1);
                    pn.add(pn2);
                    pn.setBounds(500,800,300,300);
                    pn.setLayout(new FlowLayout());

                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    JB_view.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                            JFrame jf=new
JFrame("employee Details");
                            JTable jt;
                            DefaultTableModel model =
new DefaultTableModel();
                            jt = new JTable(model);

model.addColumn("employee_id");
                            model.addColumn("employee
name");
                            model.addColumn("employee
contact");
                            model.addColumn("employee
address");
```

```
                        model.addColumn("branch
Code");
                        try
                        {

    rs=stmt.executeQuery("select * from employee");
                            while(rs.next())
                            {
                                model.addRow(new
Object[]{rs.getString("employee_id"),

    rs.getString("employee_name"),rs.getString("emplo
yee_contact"),rs.getString("employee_address")

    ,rs.getString("branch_code")});
                            }
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                        jt.setEnabled(false);
                        jt.setBounds(30, 40, 300,
300);
                        JScrollPane jsp = new
JScrollPane(jt);
                        jf.add(jsp);
                        jf.setSize(800, 400);
                        jf.setVisible(true);
                    }
                });
            }
        });
    }
}
```

Client table:

```
//K.SAI PRASAD 1602-19-737-095 INSURANCE MANAGEMENT
SYSTEM
package IMS;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.Properties;

public class client
{
    private JPanel pn,pn1,pn2,pn3;
    private JFrame jframe;
    private JButton
JB_insert,JB_modify,JB_view,JB_delete;
    private JLabel
JL_client_id,JL_client_name,JL_client_contact,JL_clie
nt_address;
    private JTextField
JTF_client_id,JTF_client_name,JTF_client_contact,JTF_
client_address;
    Connection con;
    ResultSet rs;
    Statement stmt;
    private JMenuItem insert3,update3,view3,delete3;
    private List clientList;

    public client(JPanel pn,JFrame jframe,JMenuItem
insert3,JMenuItem update3,JMenuItem view3,JMenuItem
delete3)
    {
        try
```

```
        {

    Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and
load driver");
            System.exit(1);
        }
        connectDatabase();

        this.jframe=jframe;
        this.insert3=insert3;
        this.update3=update3;
        this.view3=view3;
        this.delete3=delete3;

        JL_client_id=new JLabel("client_id:");
        JTF_client_id=new JTextField(10);
        JL_client_name=new JLabel("client Name:");
        JTF_client_name=new JTextField(10);
        JL_client_contact=new JLabel("Client
Contact:");
        JTF_client_contact=new JTextField(10);
        JL_client_address=new JLabel("client
Address:");
        JTF_client_address=new JTextField(10);

        this.pn=pn;

    }
    public void connectDatabase()
    {
        try
```

```
            {
                Connection
con=DriverManager.getConnection(

    "jdbc:oracle:thin:@localhost:1521:xe","it19737095
","vasavi");

                stmt=con.createStatement();
                stmt.executeUpdate("commit");

            }
            catch (SQLException connectException)
            {

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
        }
        private void displaySQLErrors(SQLException e)
        {

        JOptionPane.showMessageDialog(pn1,"\nSQLException
: " + e.getMessage() + "\n"+"SQLState:      " +
e.getSQLState() + "\n"+"VendorError:  " +
e.getErrorCode() + "\n");
        }
        public void loadclient()
        {
            try
            {
                clientList=new List();
```

```
            clientList.removeAll();
            rs=stmt.executeQuery("select * from
client");
            while(rs.next())
            {

    clientList.add(rs.getString("client_id"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void buildGUI()
    {

        insert3.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_insert=new JButton("Submit");

                JTF_client_id.setText(null);
                JTF_client_name.setText(null);
                JTF_client_contact.setText(null);
                JTF_client_address.setText(null);

                loadclient();

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
```

```
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_client_id);
                    pn1.add(JTF_client_id);
                    pn1.add(JL_client_name);
                    pn1.add(JTF_client_name);
                    pn1.add(JL_client_contact);
                    pn1.add(JTF_client_contact);
                    pn1.add(JL_client_address);
                    pn1.add(JTF_client_address);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_insert);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    clientList=new List(10);
                    loadclient();
                    pn2.add(clientList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();
```

```
                JB_insert.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        try
                        {
                            String query= "INSERT
INTO client VALUES(" + JTF_client_id.getText() + ","
                            + "'"
+JTF_client_name.getText() +"',"
+"'"+JTF_client_contact.getText() +"'," +
"'"+JTF_client_address.getText() +"')";
                            int i =
stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nInserted
"+i+" rows successfully");
                            loadclient();

    System.out.println("Done");
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        update3.addActionListener(new
ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_modify=new JButton("Modify");

                JTF_client_id.setText(null);
                JTF_client_name.setText(null);
                JTF_client_contact.setText(null);
                JTF_client_address.setText(null);

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                pn1=new JPanel();
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_client_id);
                pn1.add(JTF_client_id);
                pn1.add(JL_client_name);
                pn1.add(JTF_client_name);
                pn1.add(JL_client_contact);
                pn1.add(JTF_client_contact);
                pn1.add(JL_client_address);
                pn1.add(JTF_client_address);

                pn3=new JPanel(new FlowLayout());
                pn3.add(JB_modify);
                pn1.setBounds(115,80,300,250);
                pn3.setBounds(200,350,75,35);

                pn2=new JPanel(new FlowLayout());
                clientList=new List(10);
                loadclient();
```

```
                    pn2.add(clientList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    clientList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from client");
                                while (rs.next())
                                {
                                    if
(rs.getString("client_id").equals(clientList.getSelec
tedItem()))
                                    break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_client_id.setText(rs.getString("client_id"));

    JTF_client_name.setText(rs.getString("client_name
"));
```

```
    JTF_client_contact.setText(rs.getString("client_c
ontact"));

    JTF_client_address.setText(rs.getString("client_a
ddress"));

                                }
                            }
                            catch (SQLException
selectException)
                            {

    displaySQLErrors(selectException);
                                }
                        }
                    });
                    JB_modify.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                             try
                             {
                                int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to update:");

    if(a==JOptionPane.YES_OPTION)
                                {
                                    String
pack=JOptionPane.showInputDialog(pn,"Enter New client
Name:");
```

```
    JTF_client_name.setText(pack);
                                    String query="update
client set client_name='"+pack+"' where
client_id="+JTF_client_id.getText();
                                    int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nUpdated "+i+"
rows succesfully");
                                        loadclient();
                                    }
                                 }
                                catch(SQLException e)
                                {
                                    displaySQLErrors(e);
                                }
                            }
                        });
                }
        });
        delete3.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_delete=new JButton("Delete");

                JTF_client_id.setText(null);
                JTF_client_name.setText(null);
                JTF_client_contact.setText(null);
                JTF_client_address.setText(null);

                pn.removeAll();
```

```
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_client_id);
                    pn1.add(JTF_client_id);
                    pn1.add(JL_client_name);
                    pn1.add(JTF_client_name);
                    pn1.add(JL_client_contact);
                    pn1.add(JTF_client_contact);
                    pn1.add(JL_client_address);
                    pn1.add(JTF_client_address);

                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_delete);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    clientList=new List(10);
                    loadclient();
                    pn2.add(clientList);
                    pn2.setBounds(200,350,300,200);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();
```

```
                clientList.addItemListener(new
ItemListener() {
                    public void
itemStateChanged(ItemEvent ievt)
                    {
                        try
                        {

    rs=stmt.executeQuery("select * from client");
                            while (rs.next())
                            {
                                if
(rs.getString("client_id").equals(clientList.getSelec
tedItem()))
                                 break;
                            }
                            if (!rs.isAfterLast())
                            {

    JTF_client_id.setText(rs.getString("client_id"));

    JTF_client_name.setText(rs.getString("client_name
"));

    JTF_client_contact.setText(rs.getString("client_c
ontact"));

    JTF_client_address.setText(rs.getString("client_a
ddress"));

                            }
                        }
                        catch (SQLException
selectException)
```

```
                        {

    displaySQLErrors(selectException);
                        }
                    }
                });
                JB_delete.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                         try
                         {
                            int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to Delete:");

    if(a==JOptionPane.YES_OPTION)
                            {
                                //String
query="DELETE FROM client WHERE
client_id="+clientList.getSelectedItem();
                                String query="DELETE
FROM client WHERE
client_id="+JTF_client_id.getText();
                                int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nDeleted "+i+"
rows succesfully");
                                loadclient();
                            }
                         }
                        catch(SQLException e)
```

```
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        view3.addActionListener(new ActionListener()
{
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                JLabel view=new JLabel("client
View");
                JB_view=new JButton("View");
                Font myFont = new
Font("Serif",Font.BOLD,50);
                view.setFont((myFont));

                pn1=new JPanel();
                pn2=new JPanel();
                pn1.add(view);
                pn2.add(JB_view);
                pn.add(pn1);
                pn.add(pn2);
                pn.setBounds(500,800,300,300);
                pn.setLayout(new FlowLayout());

                jframe.add(pn);
```

```
                    jframe.setSize(800,800);
                    jframe.validate();

                    JB_view.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                            JFrame jf=new JFrame("client
Details");
                            JTable jt;
                            DefaultTableModel model =
new DefaultTableModel();
                            jt = new JTable(model);

model.addColumn("client_id");
                            model.addColumn("client
name");
                            model.addColumn("client
contact");
                            model.addColumn("client
address");
                            try
                            {

    rs=stmt.executeQuery("select * from client");
                                while(rs.next())
                                {
                                    model.addRow(new
Object[]{rs.getInt(1),

rs.getString(2),rs.getString(3),rs.getString(4)});
                                }
                            }
```

```
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                        jt.setEnabled(false);
                        jt.setBounds(30, 40, 300,
300);
                        JScrollPane jsp = new
JScrollPane(jt);
                        jf.add(jsp);
                        jf.setSize(800, 400);
                        jf.setVisible(true);
                    }
                });
            }
        });
    }
}
```

Insurance table:

```
//K.SAI PRASAD 1602-19-737-095 INSURANCE MANAGEMENT
SYSTEM
package IMS;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.Properties;

public class insurance
{
    private JPanel pn,pn1,pn2,pn3;
    private JFrame jframe;
    private JButton
JB_insert,JB_modify,JB_view,JB_delete;
    private JLabel
JL_insurance_num,JL_insurance_type,JL_client_id,JL_in
surance_amount,JL_pay_per_month,JL_start_date,JL_end_
date,JL_branch_code,JL_employee_id;
    private JTextField
JTF_insurance_num,JTF_insurance_type,JTF_client_id,JT
F_insurance_amount,JTF_pay_per_month,JTF_start_date,J
TF_end_date,JTF_branch_code,JTF_employee_id;
    Connection con;
    ResultSet rs;
    Statement stmt;
    private JMenuItem insert4,update4,view4,delete4;
    private List insuranceList;
    private Choice clientId,branchCode,employeeId;
```

```
    public insurance(JPanel pn,JFrame
jframe,JMenuItem insert4,JMenuItem update4,JMenuItem
view4,JMenuItem delete4)
    {
        try
        {

    Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and
load driver");
            System.exit(1);
        }
        connectDatabase();

        this.jframe=jframe;
        this.insert4=insert4;
        this.update4=update4;
        this.view4=view4;
        this.delete4=delete4;

        JL_insurance_num=new
JLabel("insurance_num:");
        JTF_insurance_num=new JTextField(10);
        JL_insurance_type=new JLabel("insurance
Name:");
        JTF_insurance_type=new JTextField(10);
        JL_client_id=new JLabel("Client Id:");
        clientId=new Choice();
        JTF_client_id=new JTextField(10);
        JL_insurance_amount=new JLabel("insurance
Amount:");
        JTF_insurance_amount=new JTextField(10);
```

```
        JL_pay_per_month=new JLabel("Pay Per
Month:");
        JTF_pay_per_month=new JTextField(10);
        JL_start_date=new JLabel("Start Date:");
        JTF_start_date=new JTextField(10);
        JL_end_date=new JLabel("End Date:");
        JTF_end_date=new JTextField(10);
        JL_branch_code=new JLabel("Branch Code:");
        branchCode=new Choice();
        JTF_branch_code=new JTextField(10);
        JL_employee_id=new JLabel("Employee Id:");
        employeeId=new Choice();
        JTF_employee_id=new JTextField(10);

        this.pn=pn;

    }
    public void connectDatabase()
    {
        try
        {
            Connection
con=DriverManager.getConnection(

    "jdbc:oracle:thin:@localhost:1521:xe","it19737095
","vasavi");

            stmt=con.createStatement();
            stmt.executeUpdate("commit");

        }
        catch (SQLException connectException)
        {

System.out.println(connectException.getMessage());
```

```
System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());
          System.exit(1);
        }
    }
    private void displaySQLErrors(SQLException e)
    {

    JOptionPane.showMessageDialog(pn1,"\nSQLException
: " + e.getMessage() + "\n"+"SQLState:     " +
e.getSQLState() + "\n"+"VendorError:  " +
e.getErrorCode() + "\n");
    }
    public void loadclient()
    {
        try
        {
            //clientId=new Choice();
            clientId.removeAll();
            rs=stmt.executeQuery("select * from
client");
            while(rs.next())
            {

    clientId.add(rs.getString("client_id"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void loadbranch()
```

```
    {
        try
        {
            //branchCode=new Choice();
            branchCode.removeAll();
            rs=stmt.executeQuery("select * from
branch");
            while(rs.next())
            {

    branchCode.add(rs.getString("branch_code"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void loademployee()
    {
        try
        {
            //employeeId=new Choice();
            employeeId.removeAll();
            rs=stmt.executeQuery("select * from
employee");
            while(rs.next())
            {

    employeeId.add(rs.getString("employee_id"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
```

```
        }
    }
    public void loadinsurance()
    {
        try
        {
            insuranceList=new List();
            insuranceList.removeAll();
            rs=stmt.executeQuery("select * from
insurance");
            while(rs.next())
            {

    insuranceList.add(rs.getString("insurance_num"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void buildGUI()
    {

        insert4.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_insert=new JButton("Submit");
                loadclient();
                loadbranch();
                loademployee();
                loadinsurance();
```

```
                    JTF_insurance_num.setText(null);
                    JTF_insurance_type.setText(null);
                    //JTF_client_id.setText(null);
                    JTF_insurance_amount.setText(null);
                    JTF_pay_per_month.setText(null);
                    JTF_start_date.setText(null);
                    JTF_end_date.setText(null);
                    //JTF_branch_code.setText(null);
                    //JTF_employee_id.setText(null);

                    loadinsurance();

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_insurance_num);
                    pn1.add(JTF_insurance_num);
                    pn1.add(JL_insurance_type);
                    pn1.add(JTF_insurance_type);
                    pn1.add(JL_client_id);
                    pn1.add(clientId);
                    pn1.add(JL_insurance_amount);
                    pn1.add(JTF_insurance_amount);
                    pn1.add(JL_pay_per_month);
                    pn1.add(JTF_pay_per_month);
                    pn1.add(JL_start_date);
                    pn1.add(JTF_start_date);
                    pn1.add(JL_end_date);
                    pn1.add(JTF_end_date);
```

```
                    pn1.add(JL_branch_code);
                    pn1.add(branchCode);
                    pn1.add(JL_employee_id);
                    pn1.add(employeeId);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_insert);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    insuranceList=new List(10);
                    loadinsurance();
                    pn2.add(insuranceList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    JB_insert.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                            try
                            {
```

```
                                String query= "INSERT
INTO insurance VALUES(" + JTF_insurance_num.getText()
+ ","
                                + "'"
+JTF_insurance_type.getText()
+"','"+"'"+clientId.getSelectedItem()+"','"+JTF_insuran
ce_amount.getText()+","+JTF_pay_per_month.getText()+"
,"+JTF_start_date.getText()+","+JTF_end_date.getText(
)+","+"'"+branchCode.getSelectedItem()+"','"+"'"+emplo
yeeId.getSelectedItem()+"')";
                                int i =
stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nInserted
"+i+" rows successfully");
                                loadinsurance();

    System.out.println("Done");
                            }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
            });
            update4.addActionListener(new
ActionListener() {
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    JB_modify=new JButton("Modify");
```

```
                    JTF_insurance_num.setText(null);
                    JTF_insurance_type.setText(null);
                    JTF_client_id.setText(null);
                    JTF_insurance_amount.setText(null);
                    JTF_pay_per_month.setText(null);
                    JTF_start_date.setText(null);
                    JTF_end_date.setText(null);
                    JTF_branch_code.setText(null);
                    JTF_employee_id.setText(null);

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_insurance_num);
                    pn1.add(JTF_insurance_num);
                    pn1.add(JL_insurance_type);
                    pn1.add(JTF_insurance_type);
                    pn1.add(JL_client_id);
                    pn1.add(JTF_client_id);
                    pn1.add(JL_insurance_amount);
                    pn1.add(JTF_insurance_amount);
                    pn1.add(JL_pay_per_month);
                    pn1.add(JTF_pay_per_month);
                    pn1.add(JL_start_date);
                    pn1.add(JTF_start_date);
                    pn1.add(JL_end_date);
                    pn1.add(JTF_end_date);
                    pn1.add(JL_branch_code);
                    pn1.add(JTF_branch_code);
                    pn1.add(JL_employee_id);
```

```
                    pn1.add(JTF_employee_id);

                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_modify);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    insuranceList=new List(10);
                    loadinsurance();
                    pn2.add(insuranceList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    insuranceList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from insurance");
                                while (rs.next())
                                {
```

```
                                    if
(rs.getString("insurance_num").equals(insuranceList.g
etSelectedItem()))
                                    break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_insurance_num.setText(rs.getString("insurance
_num"));

    JTF_insurance_type.setText(rs.getString("insuranc
e_type"));

    JTF_client_id.setText(rs.getString("client_id"));

    JTF_insurance_amount.setText(rs.getString("insura
nce_amount"));

    JTF_pay_per_month.setText(rs.getString("pay_per_m
onth"));

    JTF_start_date.setText(rs.getString("start_date")
);

    JTF_end_date.setText(rs.getString("end_date"));

    JTF_branch_code.setText(rs.getString("branch_code
"));

    JTF_employee_id.setText(rs.getString("employee_id
"));

                                }
                            }
```

```
                        catch (SQLException
selectException)
                        {

    displaySQLErrors(selectException);
                        }
                    }
                });
                JB_modify.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                         try
                         {
                            int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to update:");

    if(a==JOptionPane.YES_OPTION)
                            {
                                String
pack=JOptionPane.showInputDialog(pn,"Enter New
insurance Type:");

    JTF_insurance_type.setText(pack);
                                String query="update
insurance set insurance_type='"+pack+"' where
insurance_num="+JTF_insurance_num.getText();
                                int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nUpdated "+i+"
rows succesfully");
```

```
                                    loadinsurance();
                                }
                             }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
            });
            delete4.addActionListener(new
ActionListener() {
                @Override
                public void actionPerformed(ActionEvent
aevt)
                {
                    JB_delete=new JButton("Delete");

                    JTF_insurance_num.setText(null);
                    JTF_insurance_type.setText(null);
                    JTF_client_id.setText(null);
                    JTF_insurance_amount.setText(null);
                    JTF_pay_per_month.setText(null);
                    JTF_start_date.setText(null);
                    JTF_end_date.setText(null);
                    JTF_branch_code.setText(null);
                    JTF_employee_id.setText(null);

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
```

```
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_insurance_num);
                pn1.add(JTF_insurance_num);
                pn1.add(JL_insurance_type);
                pn1.add(JTF_insurance_type);
                pn1.add(JL_client_id);
                pn1.add(JTF_client_id);
                pn1.add(JL_insurance_amount);
                pn1.add(JTF_insurance_amount);
                pn1.add(JL_pay_per_month);
                pn1.add(JTF_pay_per_month);
                pn1.add(JL_start_date);
                pn1.add(JTF_start_date);
                pn1.add(JL_end_date);
                pn1.add(JTF_end_date);
                pn1.add(JL_branch_code);
                pn1.add(JTF_branch_code);
                pn1.add(JL_employee_id);
                pn1.add(JTF_employee_id);


                pn3=new JPanel(new FlowLayout());
                pn3.add(JB_delete);
                pn1.setBounds(115,80,300,250);
                pn3.setBounds(200,350,75,35);

                pn2=new JPanel(new FlowLayout());
                insuranceList=new List(10);
                loadinsurance();
                pn2.add(insuranceList);
                pn2.setBounds(200,350,300,200);

                pn.add(pn1);
                pn.add(pn3);
```

```
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    insuranceList.addItemListener(new
ItemListener() {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from insurance");
                                while (rs.next())
                                {
                                    if
(rs.getString("insurance_num").equals(insuranceList.g
etSelectedItem()))
                                     break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_insurance_num.setText(rs.getString("insurance
_num"));

    JTF_insurance_type.setText(rs.getString("insuranc
e_type"));

    JTF_client_id.setText(rs.getString("client_id"));
```

```
    JTF_insurance_amount.setText(rs.getString("insura
nce_amount"));

    JTF_pay_per_month.setText(rs.getString("pay_per_m
onth"));

    JTF_start_date.setText(rs.getString("start_date")
);

    JTF_end_date.setText(rs.getString("end_date"));

    JTF_branch_code.setText(rs.getString("branch_code
"));

    JTF_employee_id.setText(rs.getString("employee_id
"));


                        }
                    }
                    catch (SQLException
selectException)
                    {

    displaySQLErrors(selectException);
                    }
                }
            });
            JB_delete.addActionListener(new
ActionListener() {
                @Override
                public void
actionPerformed(ActionEvent aevt)
                {
```

```
                            try
                            {
                                int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to Delete:");

    if(a==JOptionPane.YES_OPTION)
                                {
                                    //String
query="DELETE FROM insurance WHERE
insurance_num="+insuranceList.getSelectedItem();
                                    String query="DELETE
FROM insurance WHERE
insurance_num="+JTF_insurance_num.getText();
                                    int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nDeleted "+i+"
rows succesfully");
                                    loadinsurance();
                                }
                             }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                        }
                    });
                }
        });
        view4.addActionListener(new ActionListener()
{
            @Override
            public void actionPerformed(ActionEvent
aevt)
```

```
            {
                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                JLabel view=new JLabel("insurance
View");
                JB_view=new JButton("View");
                Font myFont = new
Font("Serif",Font.BOLD,50);
                view.setFont((myFont));

                pn1=new JPanel();
                pn2=new JPanel();
                pn1.add(view);
                pn2.add(JB_view);
                pn.add(pn1);
                pn.add(pn2);
                pn.setBounds(500,800,300,300);
                pn.setLayout(new FlowLayout());

                jframe.add(pn);
                jframe.setSize(800,800);
                jframe.validate();

                JB_view.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        JFrame jf=new
JFrame("insurance Details");
                        JTable jt;
```

```
                        DefaultTableModel model =
new DefaultTableModel();
                        jt = new JTable(model);

model.addColumn("insurance_num");
                        model.addColumn("insurance
type");
                        model.addColumn("Client
Id");
                        model.addColumn("insurance
amount");
                        model.addColumn("Pay per
month");
                        model.addColumn("Start
day");
                        model.addColumn("End day");
                        model.addColumn("branch
Code");
                        model.addColumn("Employee
Id");
                        try
                        {

    rs=stmt.executeQuery("select * from insurance");
                            while(rs.next())
                            {
                                model.addRow(new
Object[]{rs.getString("insurance_num"),

    rs.getString("insurance_type"),rs.getString("clie
nt_id"),rs.getString("insurance_amount"),rs.getString
("pay_per_month")

    ,rs.getString("start_date"),rs.getString("end_dat
```

```
e"),rs.getString("branch_code"),rs.getString("employe
e_id")});
                                }
                            }
                            catch(SQLException e)
                            {
                                displaySQLErrors(e);
                            }
                            jt.setEnabled(false);
                            jt.setBounds(30, 40, 300,
300);
                            JScrollPane jsp = new
JScrollPane(jt);
                            jf.add(jsp);
                            jf.setSize(800, 400);
                            jf.setVisible(true);
                        }
                    });
                }
            });
        }
}
```

Payment table:

```
//K.SAI PRASAD 1602-19-737-095 INSURANCE MANAGEMENT
SYSTEM
package IMS;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;
import java.util.Properties;

public class payment
{
    private JPanel pn,pn1,pn2,pn3;
    private JFrame jframe;
    private JButton
JB_insert,JB_modify,JB_view,JB_delete;
    private JLabel
JL_payment_id,JL_client_id,JL_insurance_num,JL_amount
,JL_date_of_payment;
    private JTextField
JTF_payment_id,JTF_client_id,JTF_insurance_num,JTF_am
ount,JTF_date_of_payment;
    Connection con;
    ResultSet rs;
    Statement stmt;
    private JMenuItem insert5,update5,view5,delete5;
    private List paymentList;
    private Choice clientId,insuranceNum;
    public payment(JPanel pn,JFrame jframe,JMenuItem
insert5,JMenuItem update5,JMenuItem view5,JMenuItem
delete5)
```

```
    {
        try
        {

    Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and
load driver");
            System.exit(1);
        }
        connectDatabase();

        this.jframe=jframe;
        this.insert5=insert5;
        this.update5=update5;
        this.view5=view5;
        this.delete5=delete5;

        JL_payment_id=new JLabel("Payment_Id:");
        JTF_payment_id=new JTextField(10);
        JL_client_id=new JLabel("Client Id:");
        JTF_client_id=new JTextField(10);
        JL_insurance_num=new JLabel("Insurance
Num:");
        JTF_insurance_num=new JTextField(10);
        JL_amount=new JLabel("amount:");
        JTF_amount=new JTextField(10);
        JL_date_of_payment=new
JLabel("date_of_payment:");
        JTF_date_of_payment=new JTextField(10);

        this.pn=pn;
    }
```

```
    public void connectDatabase()
    {
        try
        {
            Connection
con=DriverManager.getConnection(

    "jdbc:oracle:thin:@localhost:1521:xe","it19737095
","vasavi");

            stmt=con.createStatement();
            stmt.executeUpdate("commit");

        }
        catch (SQLException connectException)
        {

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());
          System.exit(1);
        }
    }
    private void displaySQLErrors(SQLException e)
    {

    JOptionPane.showMessageDialog(pn1,"\nSQLException
: " + e.getMessage() + "\n"+"SQLState:     " +
e.getSQLState() + "\n"+"VendorError:  " +
e.getErrorCode() + "\n");
    }
    public void loadclient()
    {
```

```
        try
        {
            clientId=new Choice();
            clientId.removeAll();
            rs=stmt.executeQuery("select * from
client");
            while(rs.next())
            {

    clientId.add(rs.getString("client_id"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void loadinsurance()
    {
        try
        {
            insuranceNum=new Choice();
            insuranceNum.removeAll();
            rs=stmt.executeQuery("select * from
insurance");
            while(rs.next())
            {

    insuranceNum.add(rs.getString("insurance_num"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
```

```
    }
    public void loadpayment()
    {
        try
        {
            paymentList=new List();
            paymentList.removeAll();
            rs=stmt.executeQuery("select * from
payment");
            while(rs.next())
            {

    paymentList.add(rs.getString("payment_id"));
            }
        }
        catch(SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void buildGUI()
    {

        insert5.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_insert=new JButton("Submit");
                loadclient();
                loadinsurance();
                loadpayment();

                JTF_payment_id.setText(null);
```

```
                    //JTF_client_id.setText(null);
                    //JTF_insurance_num.setText(null);
                    JTF_amount.setText(null);
                    JTF_date_of_payment.setText(null);

                    loadpayment();

                    pn.removeAll();
                    jframe.invalidate();
                    jframe.validate();
                    jframe.repaint();

                    pn1=new JPanel();
                    pn1.setLayout(new
GridLayout(10,10));
                    pn1.add(JL_payment_id);
                    pn1.add(JTF_payment_id);
                    pn1.add(JL_client_id);
                    pn1.add(clientId);
                    pn1.add(JL_insurance_num);
                    pn1.add(insuranceNum);
                    pn1.add(JL_amount);
                    pn1.add(JTF_amount);
                    pn1.add(JL_date_of_payment);
                    pn1.add(JTF_date_of_payment);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_insert);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    paymentList=new List(10);
                    loadpayment();
```

```
                    pn2.add(paymentList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    JB_insert.addActionListener(new
ActionListener() {
                        @Override
                        public void
actionPerformed(ActionEvent aevt)
                        {
                            try
                            {
                                String query= "INSERT
INTO payment VALUES(" + JTF_payment_id.getText() +
","

+"'"+clientId.getSelectedItem()+"',"+"'"+insuranceNum
.getSelectedItem()+"',"+JTF_amount.getText() +","
+JTF_date_of_payment.getText() +")";
                                int i =
stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nInserted
"+i+" rows successfully");
                                loadpayment();

    System.out.println("Done");
```

```
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        update5.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_modify=new JButton("Modify");

                JTF_payment_id.setText(null);
                JTF_client_id.setText(null);
                JTF_insurance_num.setText(null);
                JTF_amount.setText(null);
                JTF_date_of_payment.setText(null);

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                pn1=new JPanel();
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_payment_id);
                pn1.add(JTF_payment_id);
                pn1.add(JL_client_id);
                pn1.add(JTF_client_id);
```

```
                    pn1.add(JL_insurance_num);
                    pn1.add(JTF_insurance_num);
                    pn1.add(JL_amount);
                    pn1.add(JTF_amount);
                    pn1.add(JL_date_of_payment);
                    pn1.add(JTF_date_of_payment);

                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_modify);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    paymentList=new List(10);
                    loadpayment();
                    pn2.add(paymentList);
                    pn2.setBounds(200,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();


                    paymentList.addItemListener(new
ItemListener()
                    {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
```

```
                        {

    rs=stmt.executeQuery("select * from payment");

                            while (rs.next())
                            {
                                if
(rs.getString("payment_Id").equals(paymentList.getSel
ectedItem()))
                                 break;
                            }
                            if (!rs.isAfterLast())
                            {

    JTF_payment_id.setText(rs.getString("payment_id")
);

    JTF_client_id.setText(rs.getString("client_id"));

    JTF_insurance_num.setText(rs.getString("insurance
_num"));

    JTF_amount.setText(rs.getString("amount"));

    JTF_date_of_payment.setText(rs.getString("date_of
_payment"));
                            }
                        }
                        catch (SQLException
selectException)
                        {

    displaySQLErrors(selectException);
                        }
                    }
```

```
                });
                JB_modify.addActionListener(new
ActionListener() {
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        try
                        {

                            int
a=JOptionPane.showConfirmDialog(pn,"Are you sure do
you want to update:");

    if(a==JOptionPane.YES_OPTION)
                            {
                                String
pack=JOptionPane.showInputDialog(pn,"Enter New
amount:");

    JTF_amount.setText(pack);
                                String query="update
payment set amount='"+pack+"' where
payment_id="+JTF_payment_id.getText();
                                int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nUpdated "+i+"
rows succesfully");
                                loadpayment();
                            }
                        }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
```

```
                        }
                    }
                });
            }
        });
        delete5.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent
aevt)
            {
                JB_delete=new JButton("Delete");

                JTF_payment_id.setText(null);
                JTF_client_id.setText(null);
                JTF_insurance_num.setText(null);
                JTF_amount.setText(null);
                JTF_date_of_payment.setText(null);

                pn.removeAll();
                jframe.invalidate();
                jframe.validate();
                jframe.repaint();

                pn1=new JPanel();
                pn1.setLayout(new
GridLayout(10,10));
                pn1.add(JL_payment_id);
                pn1.add(JTF_payment_id);
                pn1.add(JL_client_id);
                pn1.add(JTF_client_id);
                pn1.add(JL_insurance_num);
                pn1.add(JTF_insurance_num);
                pn1.add(JL_amount);
                pn1.add(JTF_amount);
```

```
                    pn1.add(JL_date_of_payment);
                    pn1.add(JTF_date_of_payment);


                    pn3=new JPanel(new FlowLayout());
                    pn3.add(JB_delete);
                    pn1.setBounds(115,80,300,250);
                    pn3.setBounds(200,350,75,35);

                    pn2=new JPanel(new FlowLayout());
                    paymentList=new List(10);
                    loadpayment();
                    pn2.add(paymentList);
                    pn2.setBounds(250,350,300,180);

                    pn.add(pn1);
                    pn.add(pn3);
                    pn.add(pn2);

                    pn.setLayout(new BorderLayout());
                    jframe.add(pn);
                    jframe.setSize(800,800);
                    jframe.validate();

                    paymentList.addItemListener(new
ItemListener()
                    {
                        public void
itemStateChanged(ItemEvent ievt)
                        {
                            try
                            {

    rs=stmt.executeQuery("select * from payment");
                                while(rs.next())
```

```
                                {
                                    if
(rs.getString("payment_id").equals(paymentList.getSel
ectedItem()))
                                     break;
                                }
                                if (!rs.isAfterLast())
                                {

    JTF_payment_id.setText(rs.getString("payment_id")
);

    JTF_client_id.setText(rs.getString("client_id"));

    JTF_insurance_num.setText(rs.getString("insurance
_num"));

    JTF_amount.setText(rs.getString("amount"));

    JTF_date_of_payment.setText(rs.getString("date_of
_payment"));

                                }
                            }
                            catch (SQLException
selectException)
                            {

    displaySQLErrors(selectException);
                            }
                        }
                    });
                    JB_delete.addActionListener(new
ActionListener() {
                        @Override
```

```
                    public void
actionPerformed(ActionEvent aevt)
                    {
                         try
                         {
                            int
a=JOptionPane.showConfirmDialog(pn,"Are you sure want
to Delete:");

    if(a==JOptionPane.YES_OPTION)
                              {
                                  //String
query="DELETE FROM employee WHERE
employee_id="+employeeList.getSelectedItem();
                                  String query="DELETE
FROM payment WHERE
payment_id="+JTF_payment_id.getText();
                                  int
i=stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(pn,"\nDeleted "+i+"
rows succesfully");
                                   loadpayment();
                              }
                          }
                        catch(SQLException e)
                        {
                            displaySQLErrors(e);
                        }
                    }
                });
            }
        });
        view5.addActionListener(new ActionListener()
{
```

```
                    @Override
                    public void
actionPerformed(ActionEvent aevt)
                    {
                        pn.removeAll();
                        jframe.invalidate();
                        jframe.validate();
                        jframe.repaint();


                        JLabel view=new
JLabel("payment View");
                        JB_view=new JButton("View");
                        Font myFont = new
Font("Serif",Font.BOLD,50);
                        view.setFont((myFont));

                        pn1=new JPanel();
                        pn2=new JPanel();
                        pn1.add(view);
                        pn2.add(JB_view);
                        pn.add(pn1);
                        pn.add(pn2);
                        pn.setLayout(new
FlowLayout());

                        jframe.add(pn);
                        jframe.setSize(800,800);
                        jframe.validate();

                    JB_view.addActionListener(new
ActionListener() {
                            @Override
                           public void
actionPerformed(ActionEvent e)
```

```
                                    {
                                        JFrame jf=new
JFrame("payment Details");
                                        JTable jt;
                                        DefaultTableModel
model = new DefaultTableModel();
                                        jt= new
JTable(model);

model.addColumn("Payment Id");

model.addColumn("Client Id");

model.addColumn("insurance num");

model.addColumn("amount");

model.addColumn("date_of_payment");
                                        try
                                        {

    rs=stmt.executeQuery("select * from payment");
                                            while(rs.next())
                                            {

model.addRow(new
Object[]{rs.getString("payment_id"),rs.getString("cli
ent_id"),rs.getString("insurance_num"),

rs.getString("amount"),rs.getString("date_of_payment"
)});
                                            }
                                        }
                                        catch(SQLException
exp)
```

```
                                        {

    displaySQLErrors(exp);
                                        }


    jt.setEnabled(false);
                                        jt.setBounds(30, 40,
180, 150);
                                        JScrollPane sp = new
JScrollPane(jt);
                                        jf.add(sp);
                                        jf.setSize(800,
400);
                                        jf.setVisible(true);
                            }
                        });
                    }
                });
            }
}
```

Main

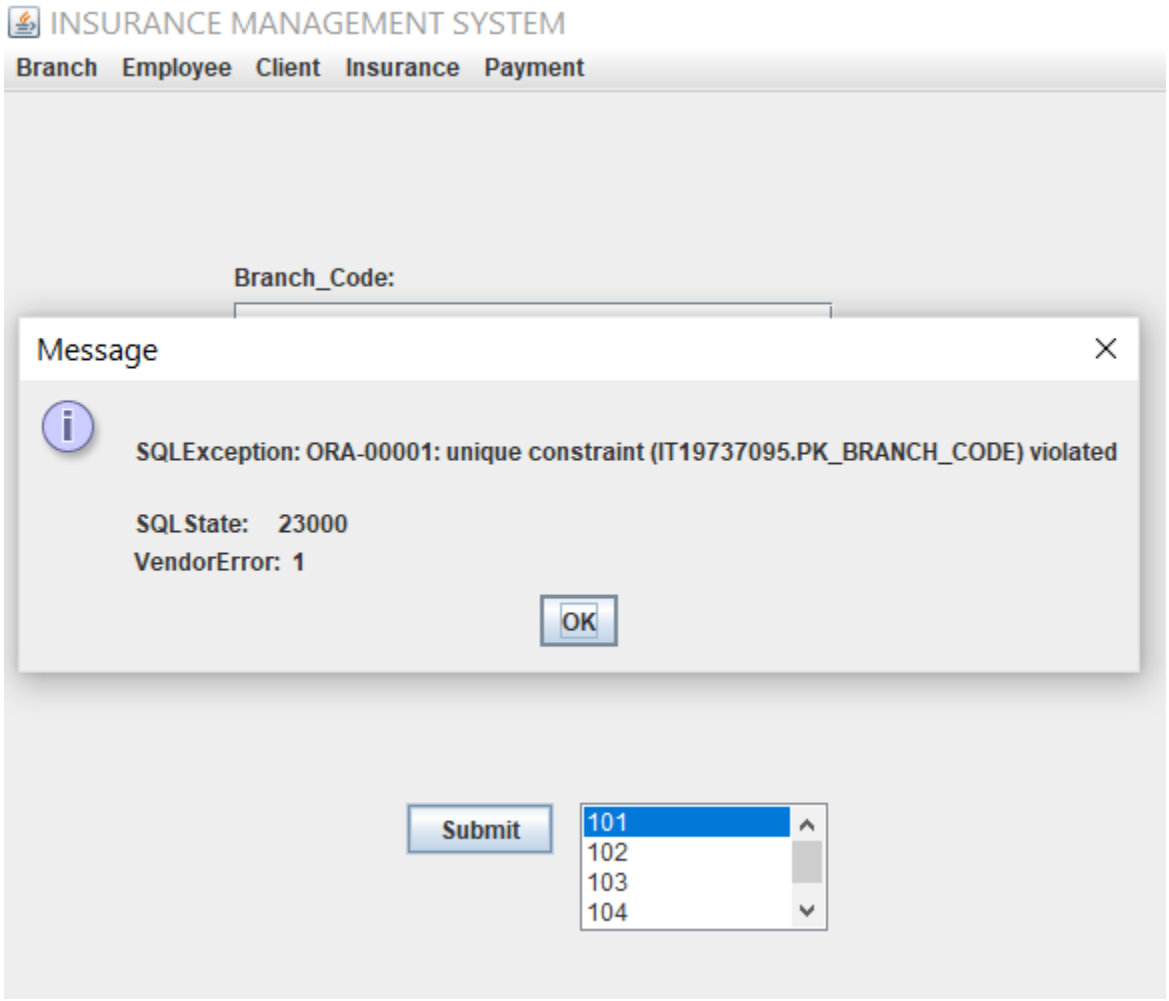
```
package IMS;
import java.awt.*;
import javax.swing.*;
public class IMSUI extends JFrame
{
    private JPanel Jpn0,Jpn1;
    private JMenuBar mbar;
    private JMenu
branch,employee,client,insurance,payment;
    private JMenuItem insert1,update1,view1,delete1;
    private JMenuItem insert2,update2,view2,delete2;
    private JMenuItem insert3,update3,view3,delete3;
    private JMenuItem insert4,update4,view4,delete4;
    private JMenuItem insert5,update5,view5,delete5;
    /*private JMenuItem
insert6,update6,view6,delete6;
    private JMenuItem s1,s2,s3,s4,s5;*/

    private JLabel labelName;

    public void defineComponents()
    {
    Jpn0=new JPanel();
    Jpn1=new JPanel();

    mbar=new JMenuBar();

    //statesAvailable=new JMenu("List Of States
Available");//d_1
    branch=new JMenu("Branch");
    employee=new JMenu("Employee");
    client=new JMenu("Client");
```

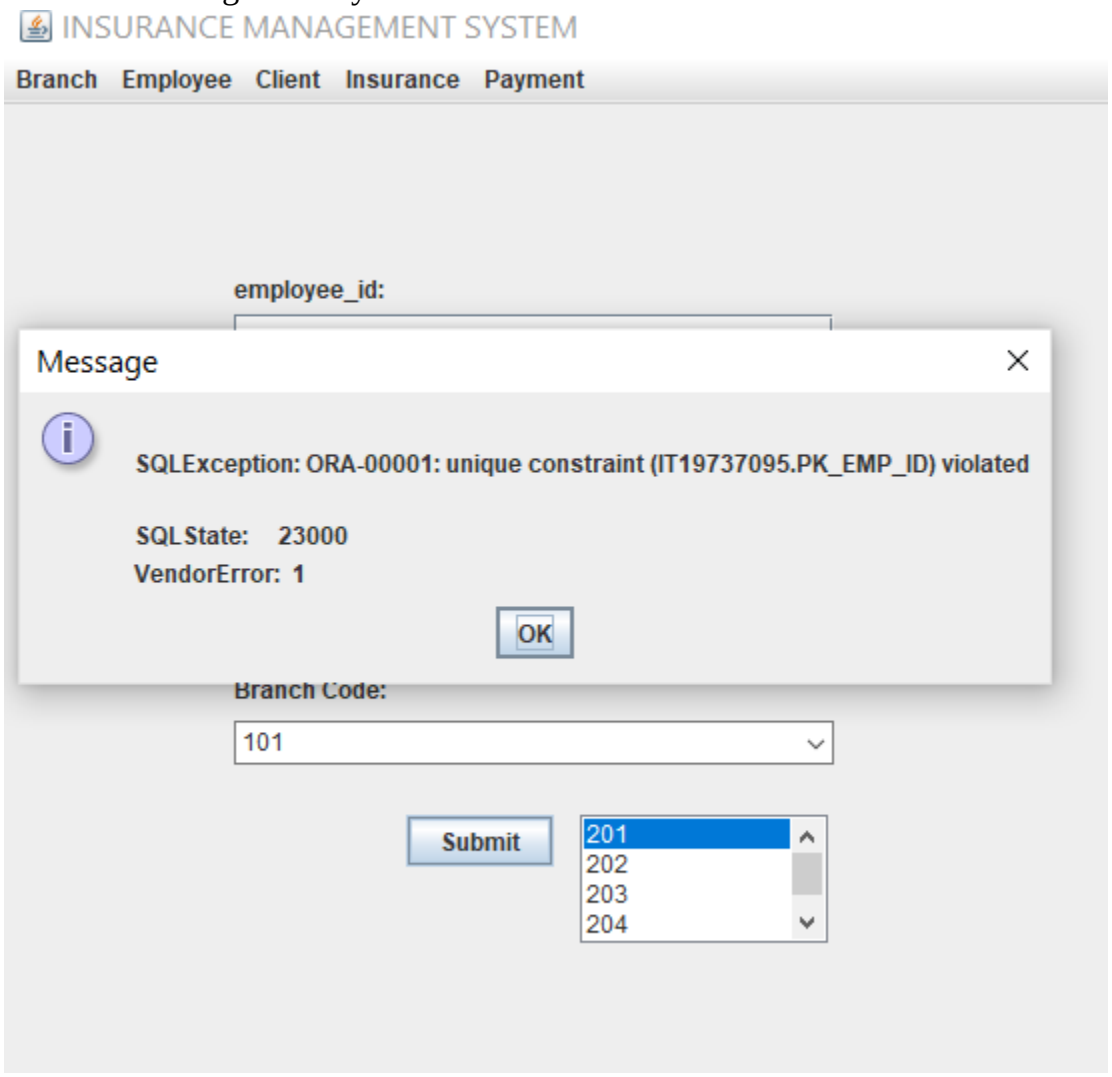
```
    insurance=new JMenu("Insurance");
    payment=new JMenu("Payment");

    labelName=new JLabel("Insurance Management
System");
    }
    public void addComponents()
    {
    add(Jpn0);
    Jpn1.add(labelName);
    Jpn1.setAlignmentY(CENTER_ALIGNMENT);
    Jpn1.setBounds(500,500,800,100);
    Jpn0.add(Jpn1);

    setJMenuBar(mbar);

    mbar.add(branch);
    mbar.add(employee);
    mbar.add(client);
    mbar.add(insurance);
    mbar.add(payment);

    /*statesAvailable.add(s1=new JMenuItem("Goa"));
    statesAvailable.add(s2=new JMenuItem("Kerala"));
    statesAvailable.add(s3=new
JMenuItem("Telangana"));
    statesAvailable.add(s4=new
JMenuItem("Maharashtra"));
    statesAvailable.add(s5=new JMenuItem("Uttar
Pradesh"));*/

    branch.add(insert1=new JMenuItem("Insert"));
    branch.add(update1=new JMenuItem("Update"));
    branch.add(view1=new JMenuItem("View"));
    branch.add(delete1=new JMenuItem("Delete"));
```

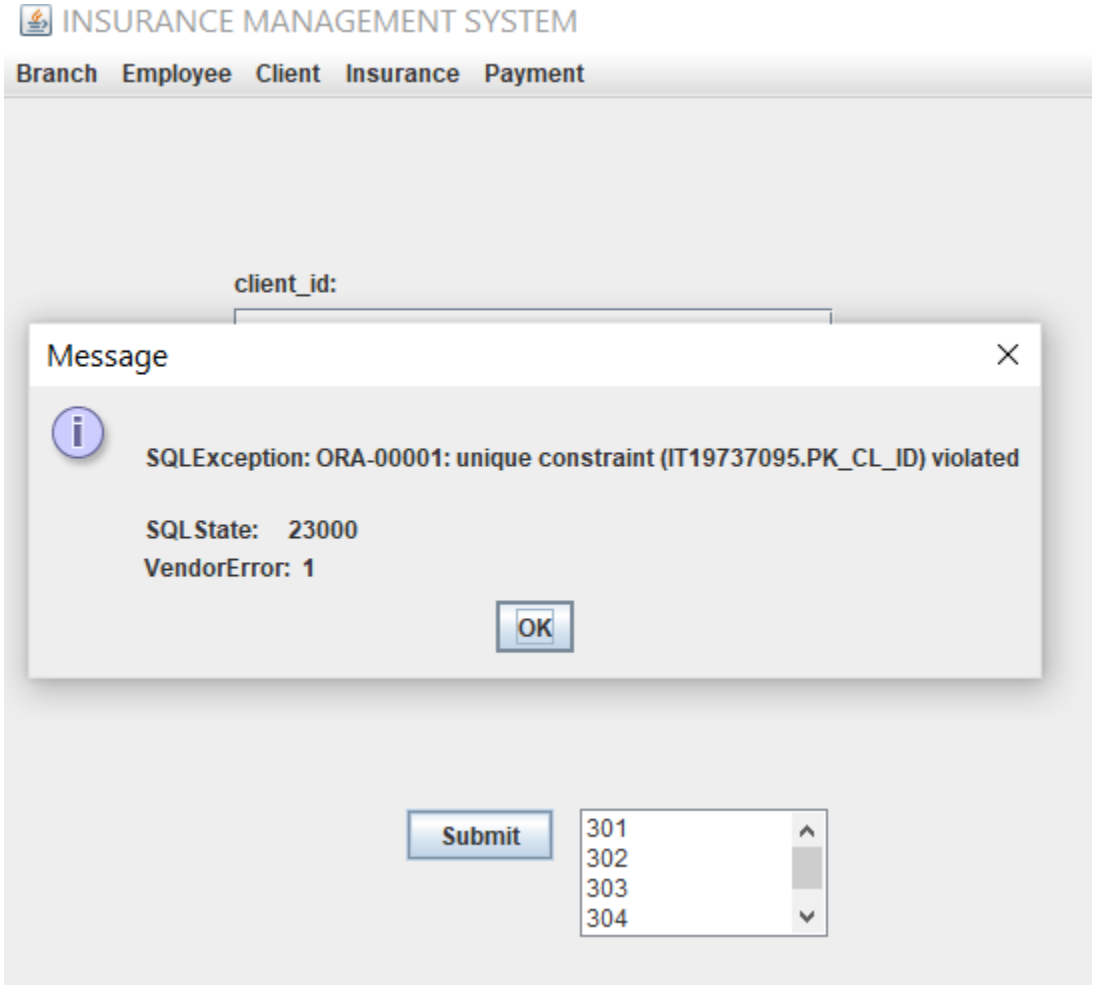
```
    employee.add(insert2=new JMenuItem("Insert"));
    employee.add(update2=new JMenuItem("Update"));
    employee.add(view2=new JMenuItem("View"));
    employee.add(delete2=new JMenuItem("Delete"));

    client.add(insert3=new JMenuItem("Insert"));
    client.add(update3=new JMenuItem("Update"));
    client.add(view3=new JMenuItem("View"));
    client.add(delete3=new JMenuItem("Delete"));

    insurance.add(insert4=new JMenuItem("Insert"));
    insurance.add(update4=new JMenuItem("Update"));
    insurance.add(view4=new JMenuItem("View"));
    insurance.add(delete4=new JMenuItem("Delete"));

    payment.add(insert5=new JMenuItem("Insert"));
    payment.add(update5=new JMenuItem("Update"));
    payment.add(view5=new JMenuItem("View"));
    payment.add(delete5=new JMenuItem("Delete"));

    }
    public void registerComponents()
    {
    branch obj_branch=new
branch(Jpn0,IMSUI.this,insert1,update1,view1,delete1)
;
    obj_branch.buildGUI();
    employee obj_employee=new
employee(Jpn0,IMSUI.this,insert2,update2,view2,delete
2);
    obj_employee.buildGUI();
    client obj_client=new
client(Jpn0,IMSUI.this,insert3,update3,view3,delete3)
;
```

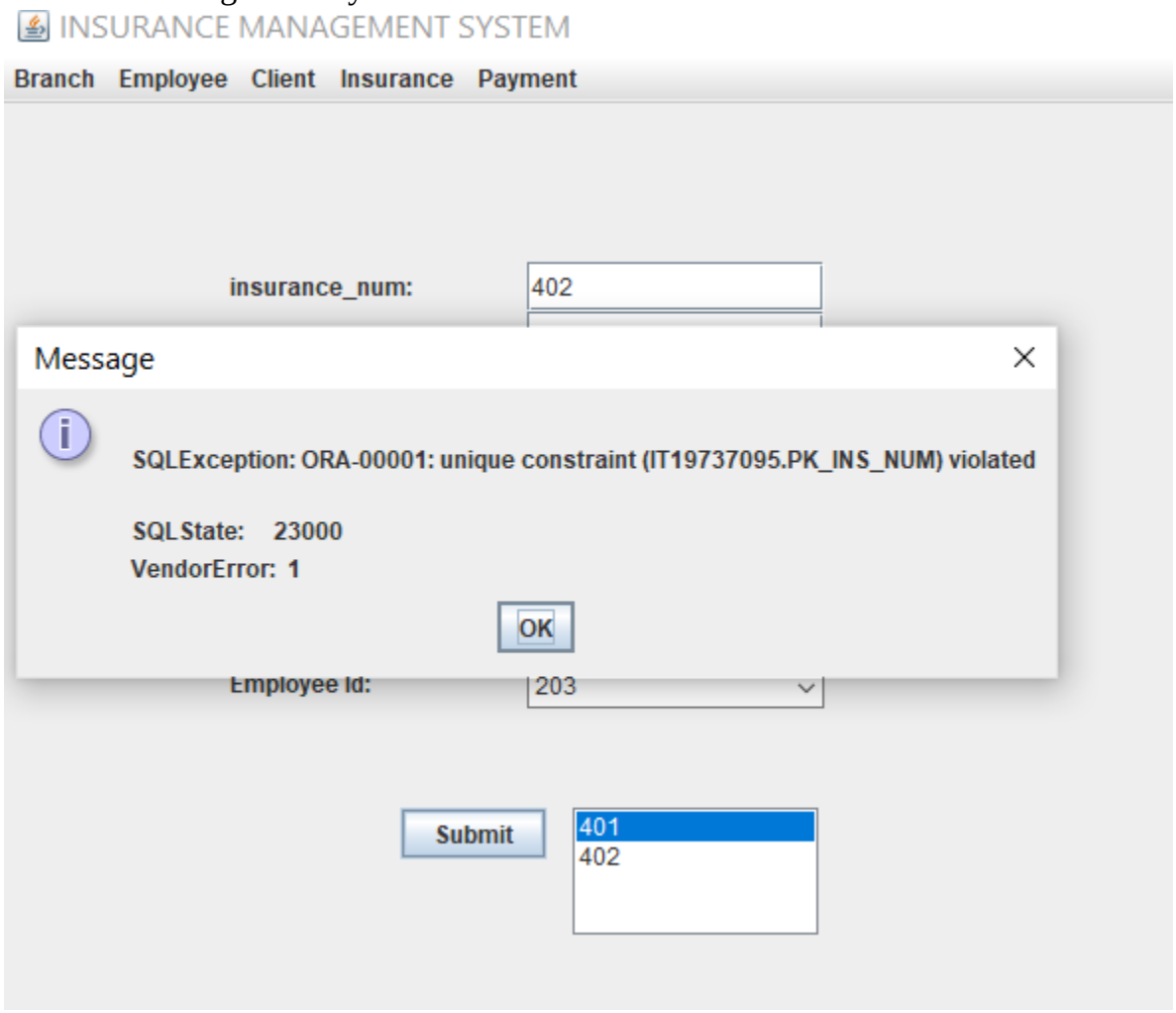
```
    obj_client.buildGUI();
    insurance obj_insurance=new
insurance(Jpn0,IMSUI.this,insert4,update4,view4,delet
e4);
    obj_insurance.buildGUI();
    payment obj_payment=new
payment(Jpn0,IMSUI.this,insert5,update5,view5,delete5
);
    obj_payment.buildGUI();
    }
    public IMSUI()
    {
        defineComponents();
        addComponents();
        registerComponents();
        setSize(400,500);
        setBackground(Color.GRAY);
        setVisible(true);
        setTitle("INSURANCE MANAGEMENT SYSTEM");

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
package IMS;
public class Main
{
    public static void main(String []args)
    {
        new IMSUI();
    }
}
```

TESTING


INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Branch_Code:
Message
SQLException: ORA-00001: unique constraint (IT19737095.PK_BRANCH_CODE) violated
SQLState: 23000
VendorError: 1
OK
Submit
101
102
103
104

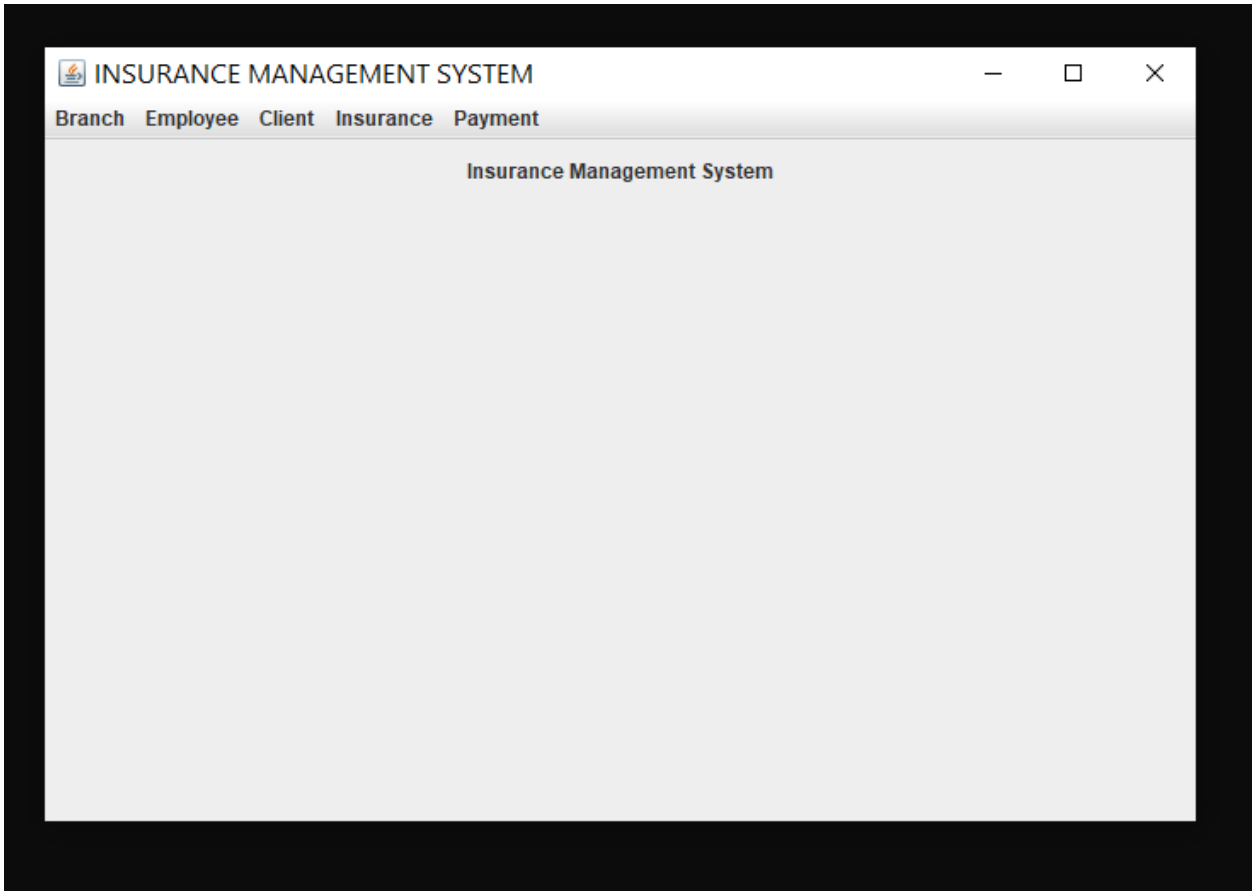

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
employee_id:
Message
SQLException: ORA-00001: unique constraint (IT19737095.PK_EMP_ID) violated
SQL State: 23000
VendorError: 1
OK
Branch Code:
101
Submit
201
202
203
204


INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
client_id:
Message
SQLException: ORA-00001: unique constraint (IT19737095.PK_CL_ID) violated
SQL State: 23000
VendorError: 1
OK
Submit
301
302
303
304


INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
insurance_num:
402
Message
SQLException: ORA-00001: unique constraint (IT19737095.PK_INS_NUM) violated
SQLState: 23000
VendorError: 1
OK
Employee Id:
203
Submit
401
402

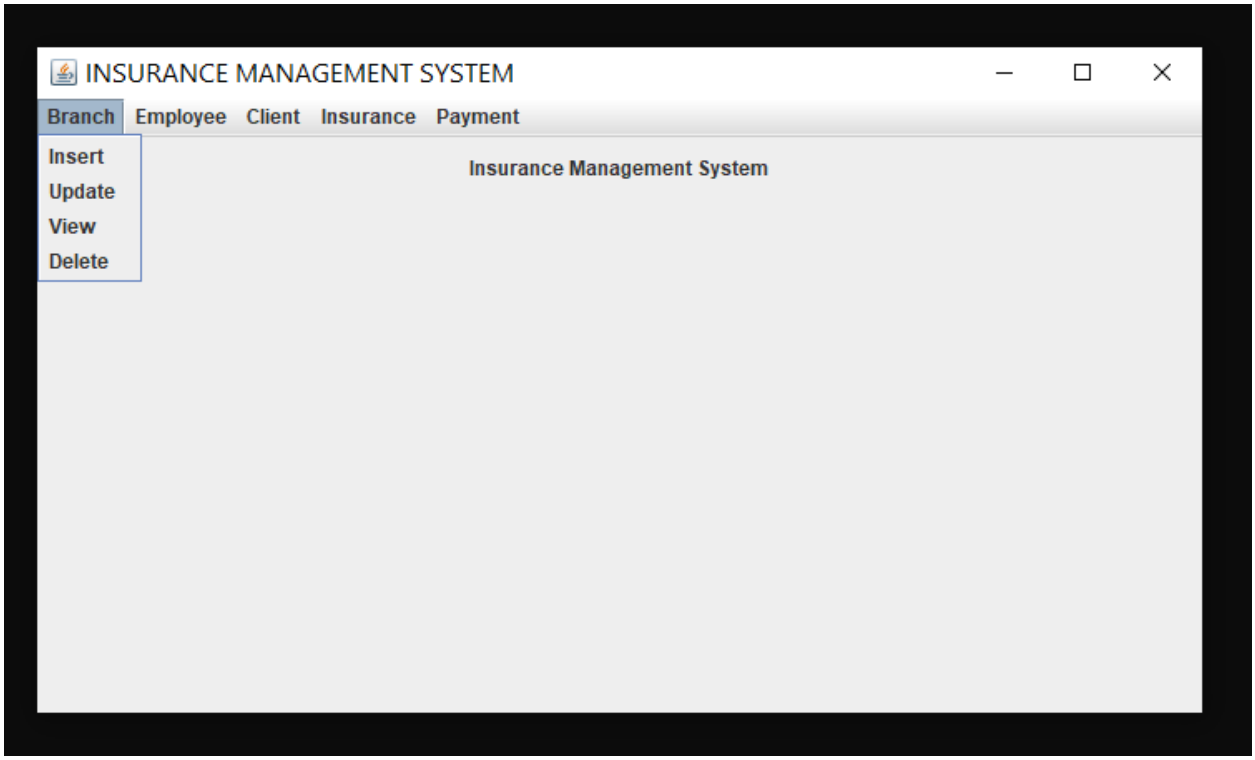
RESULTS

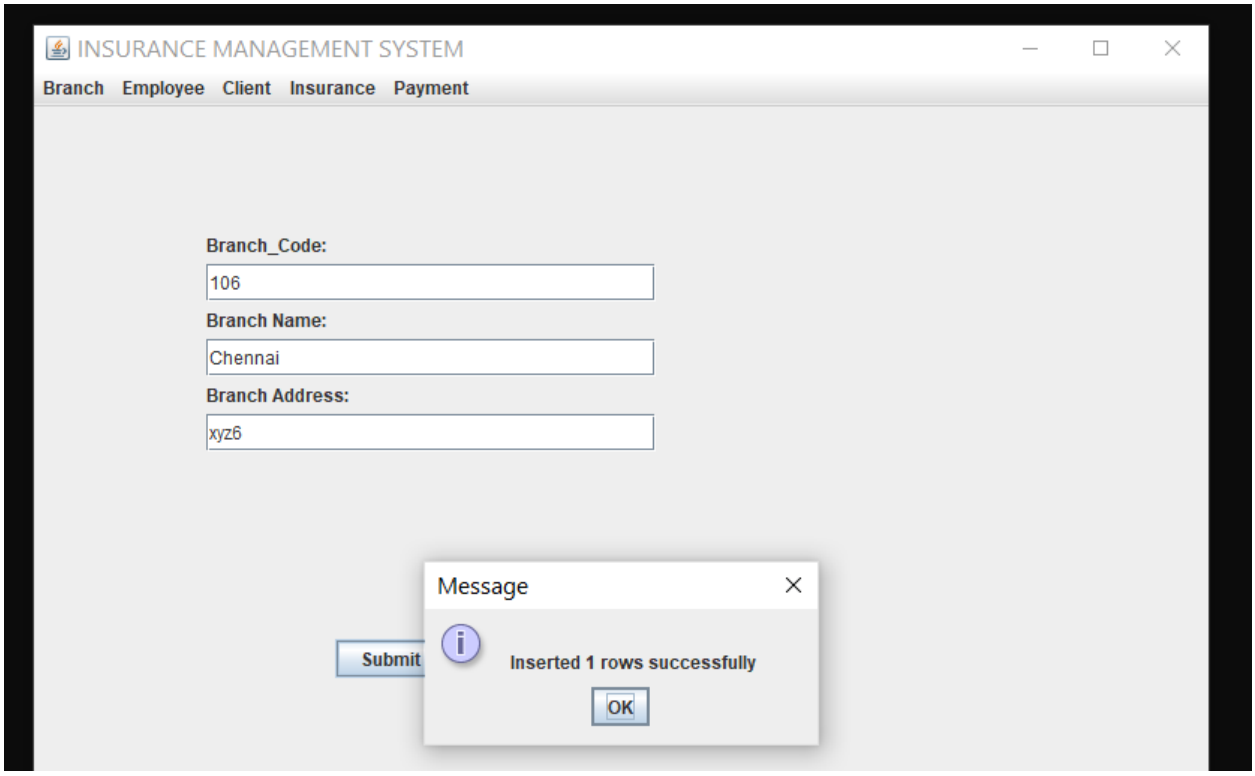
MAIN:

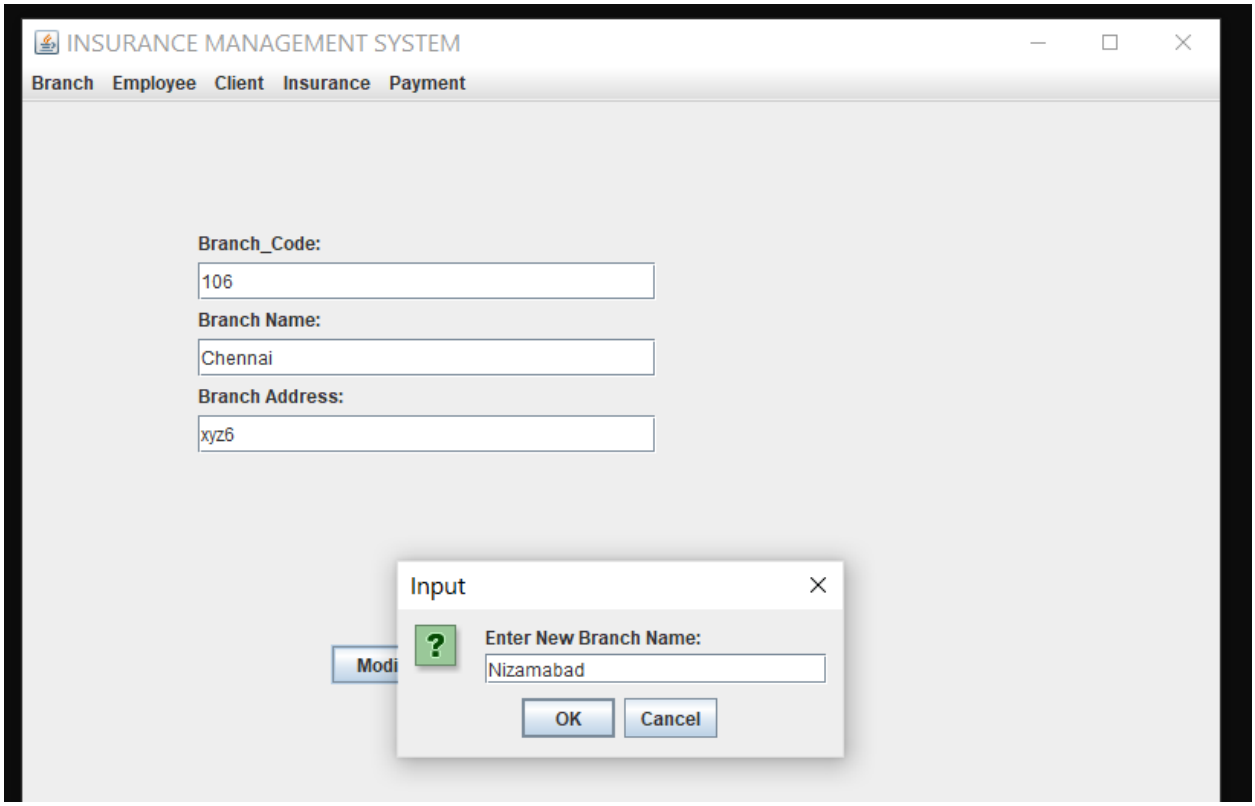


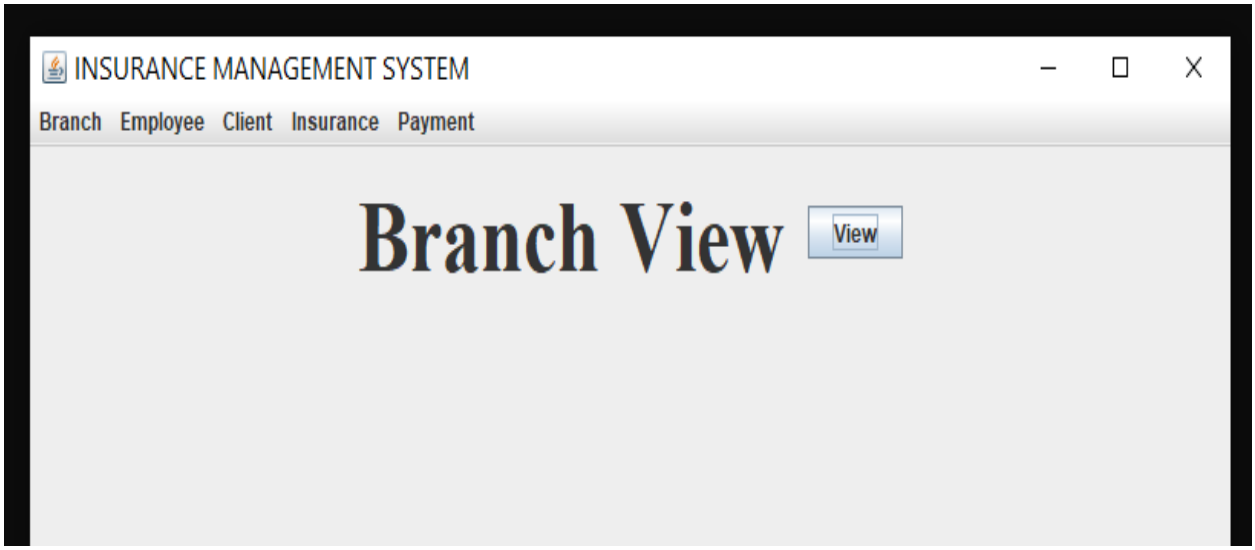
BRANCH TABLE:

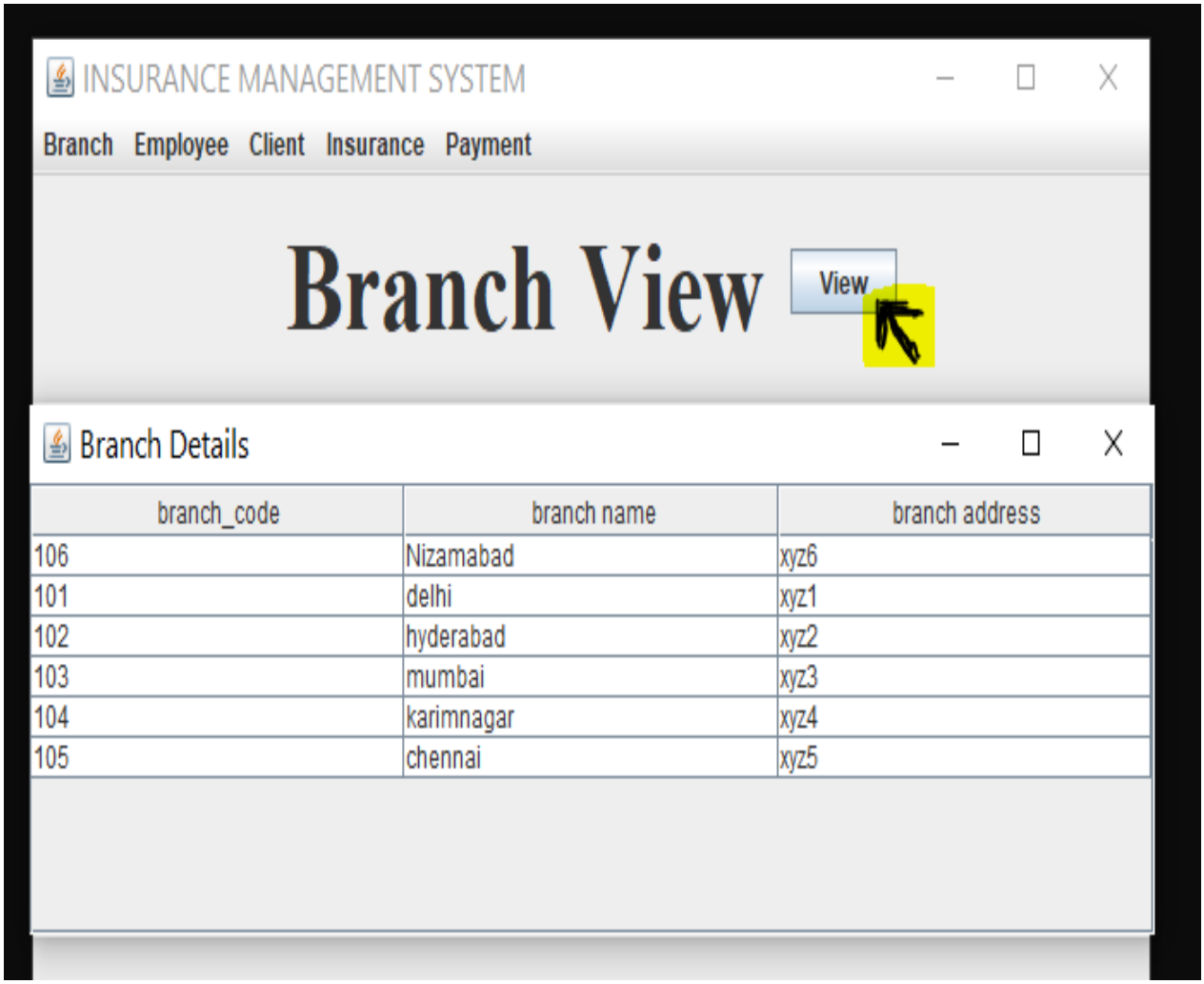
The branch_code is an unique value and it can only have a numeric value and not anything else, similarly the branch_name can only be a string. In case of any unexpected or incorrect input it throws an exception.

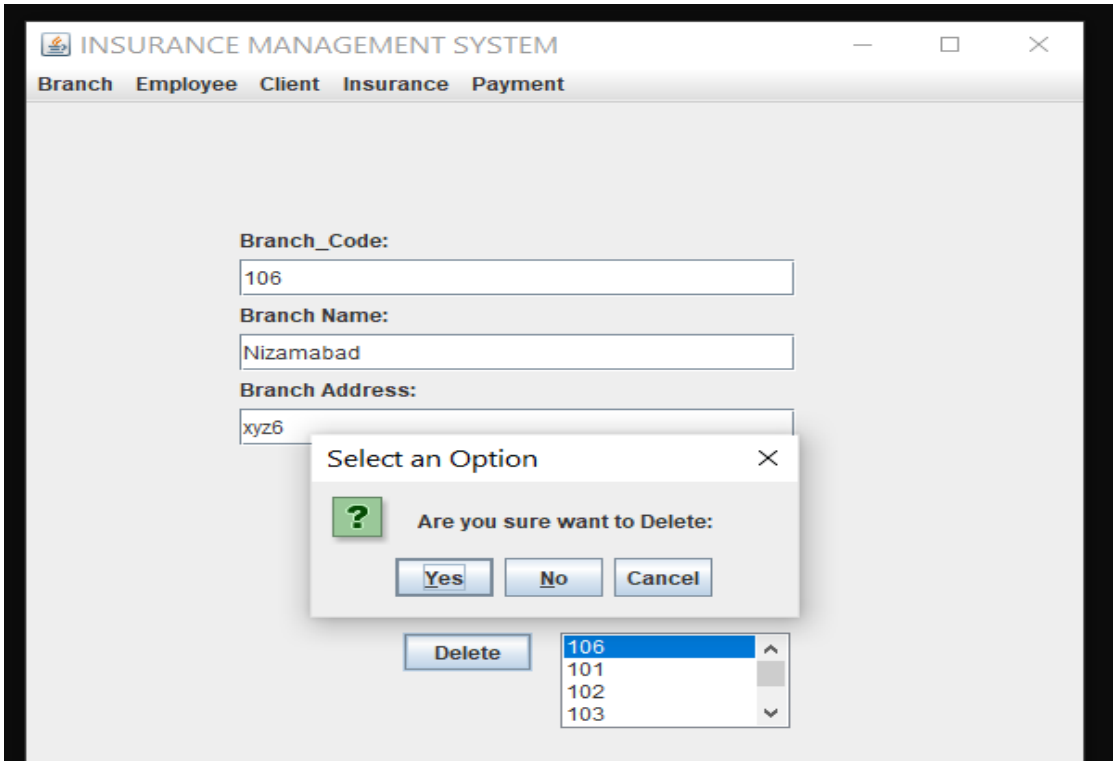

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Insert
Update
View
Delete
Insurance Management System

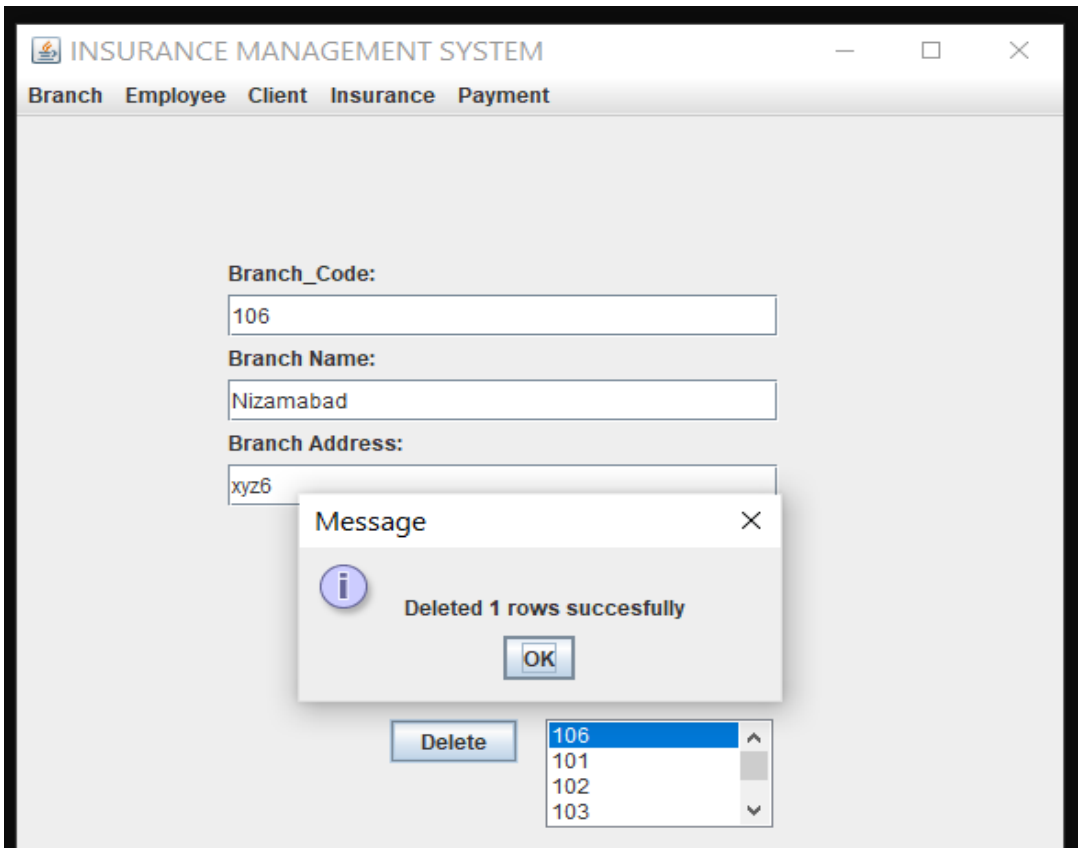

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Branch_Code:
106
Branch Name:
Chennai
Branch Address:
xyz6
Submit
Message
Inserted 1 rows successfully
OK


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
Branch_Code:
106
Branch Name:
Chennai
Branch Address:
xyz6
Modi
Input
Enter New Branch Name:
Nizamabad
OK
Cancel


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
Branch View
View

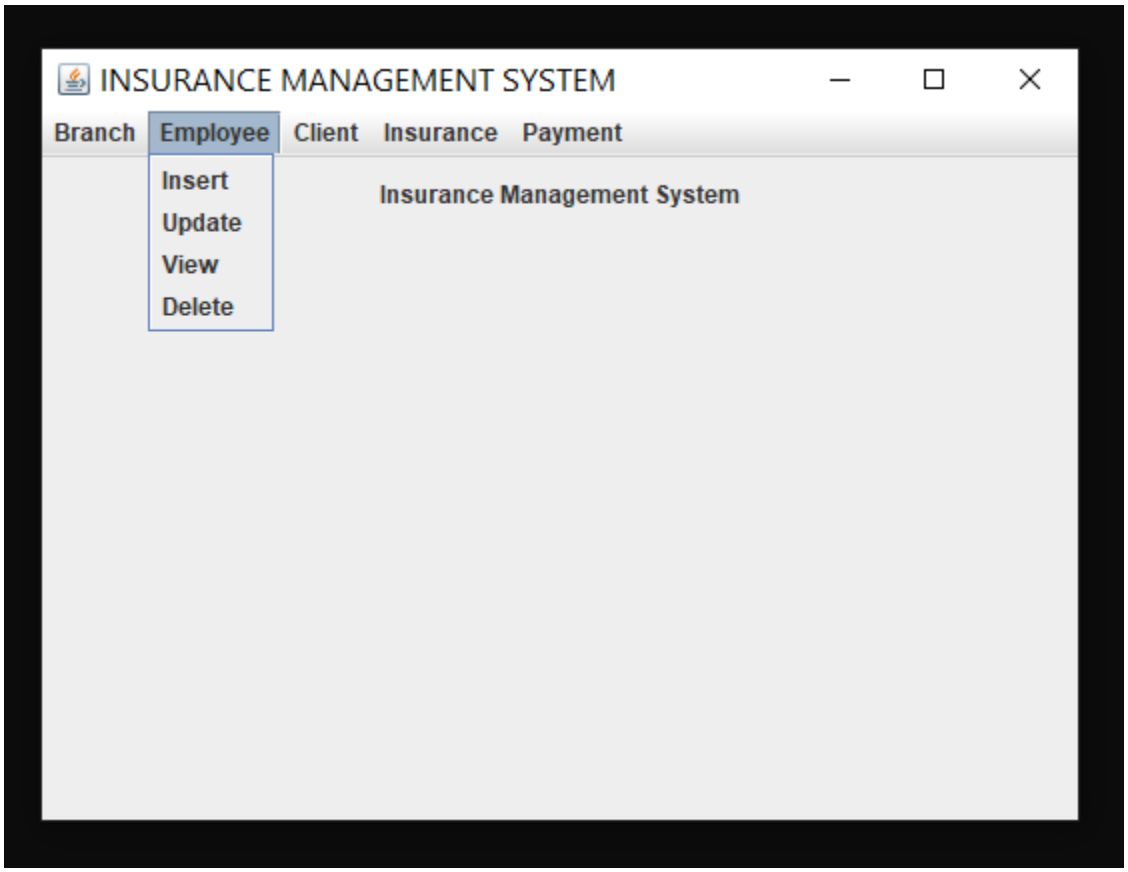

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Branch View
View
Branch Details
branch_code
branch name
branch address
106
Nizamabad
xyz6
101
delhi
xyz1
102
hyderabad
xyz2
103
mumbai
xyz3
104
karimnagar
xyz4
105
chennai
xyz5

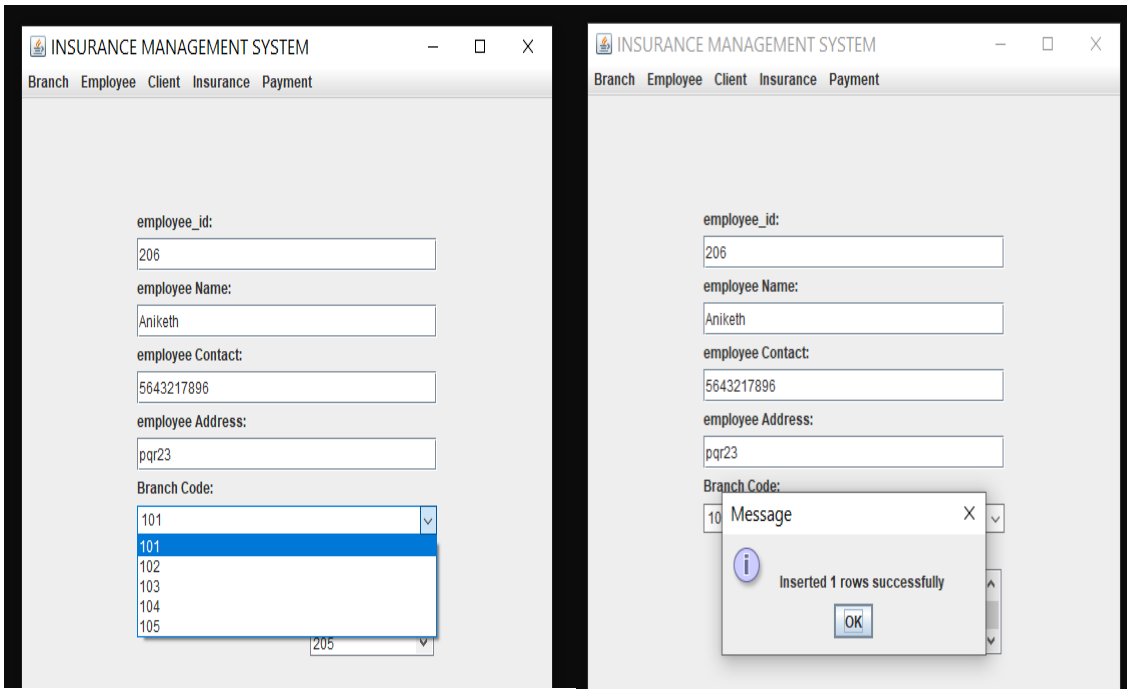

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Branch_Code:
106
Branch Name:
Nizamabad
Branch Address:
xyz6
Select an Option
Are you sure want to Delete:
Yes
No
Cancel
Delete
106
101
102
103

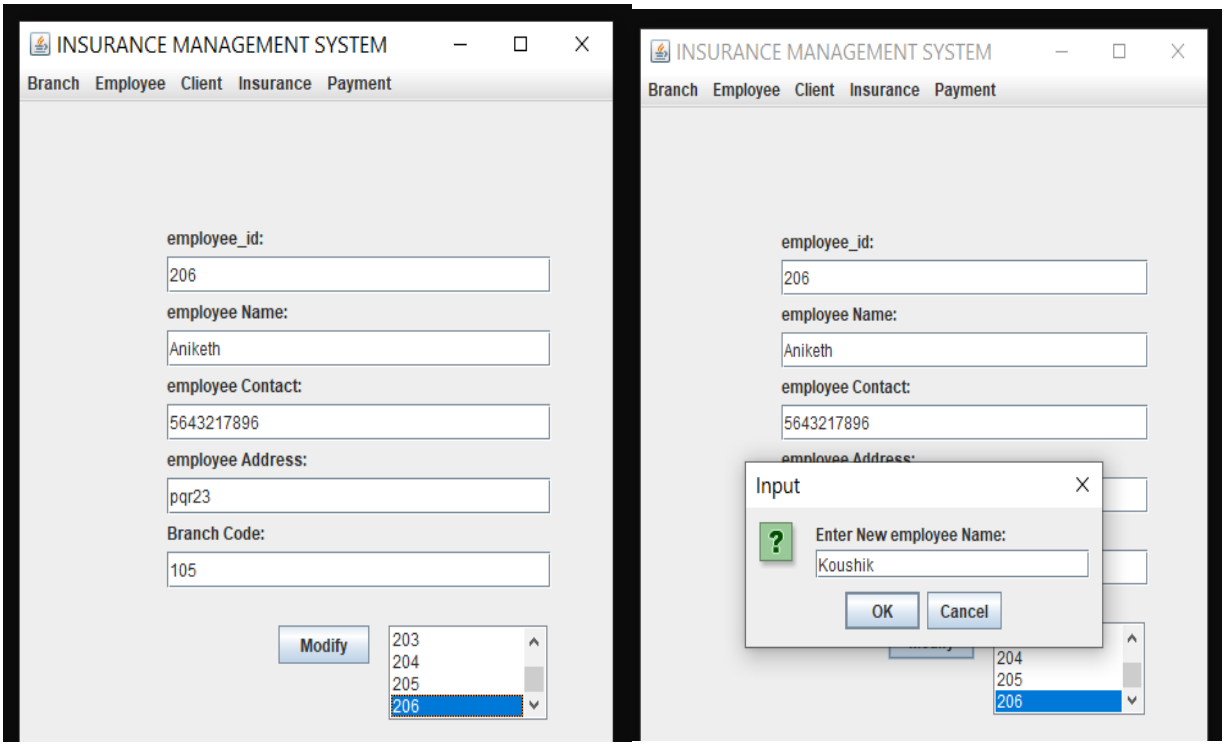

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Branch_Code:
106
Branch Name:
Nizamabad
Branch Address:
xyz6
Message
Deleted 1 rows succesfully
OK
Delete
106
101
102
103

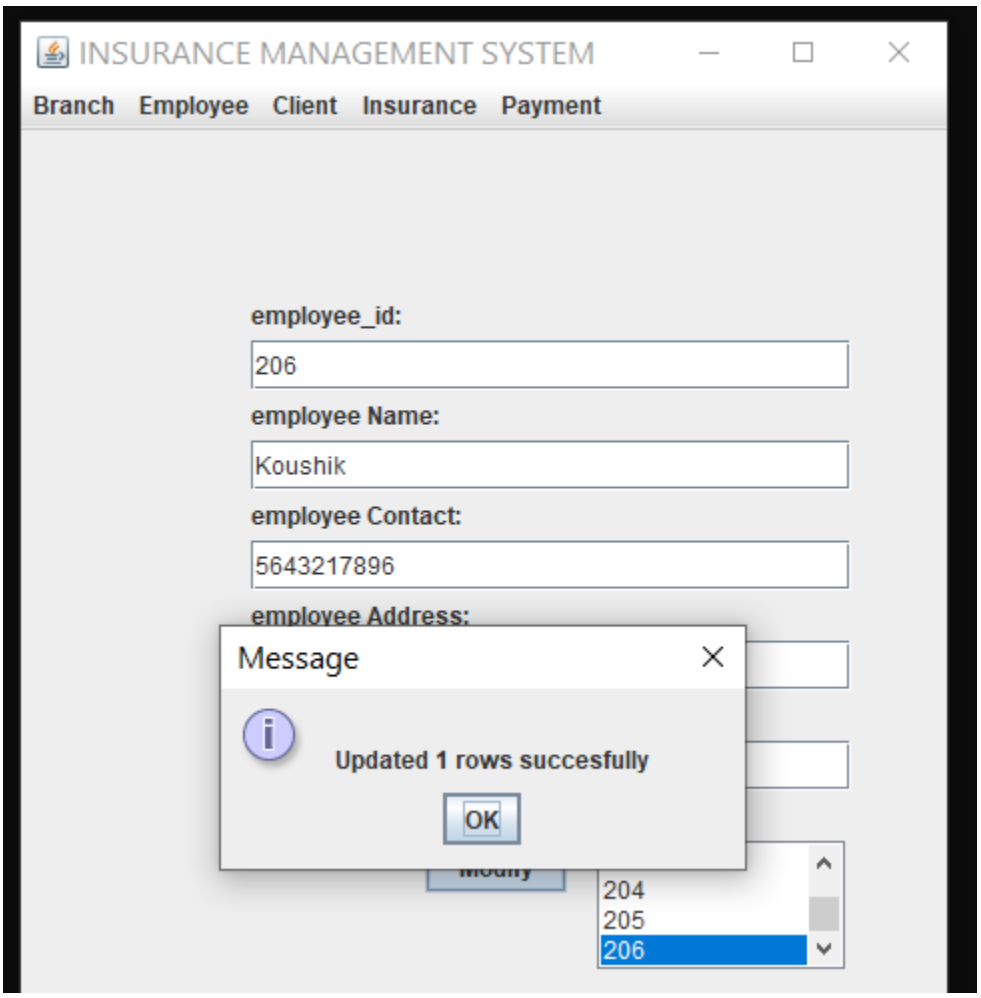
EMPLOYEE TABLE:

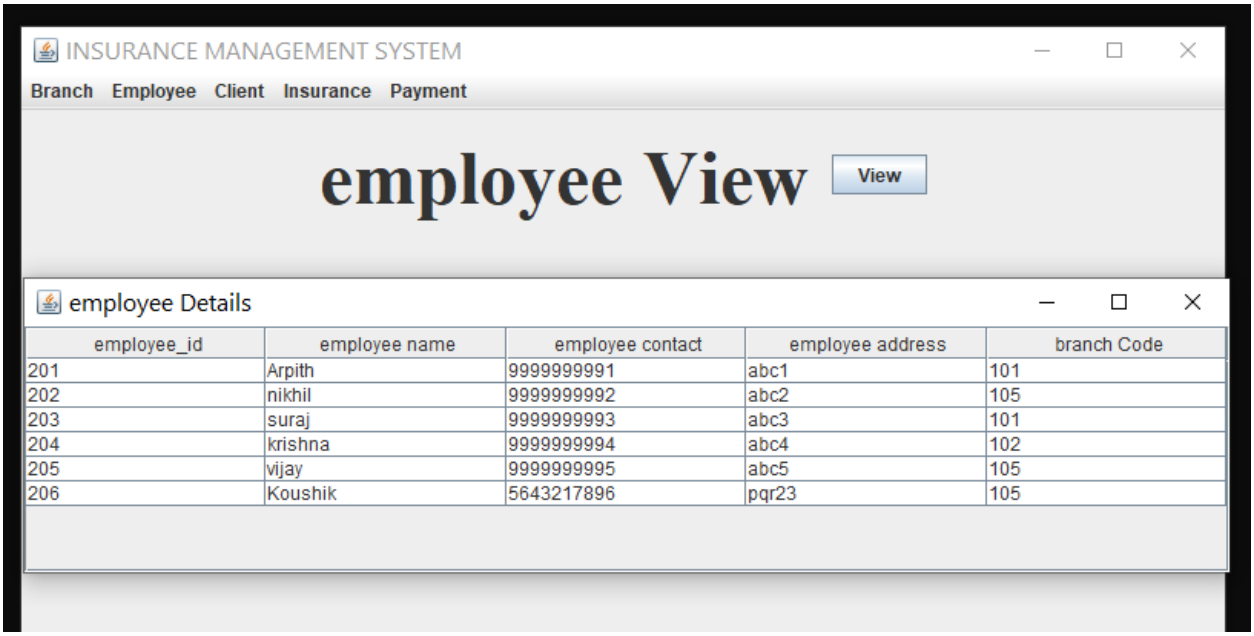
Employee_Id is the primary and unique key. branch_code is the foreign key.

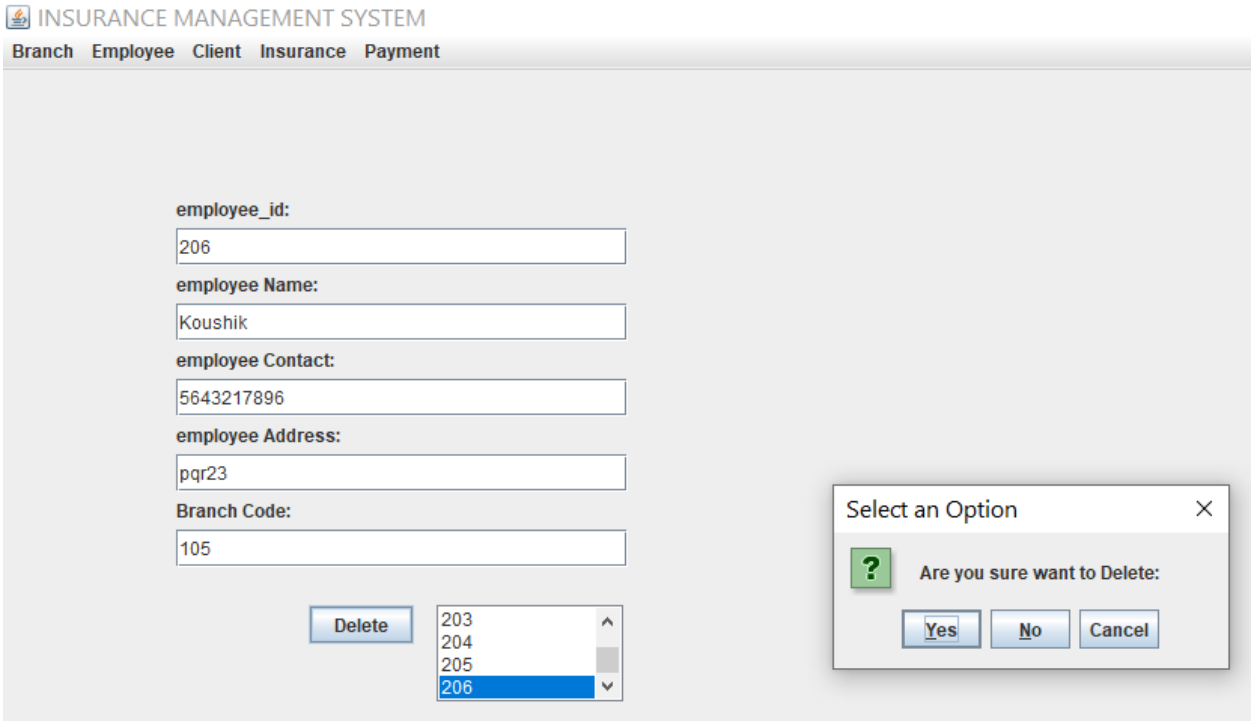





INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
employee_id:
206
employee Name:
Aniketh
employee Contact:
5643217896
employee Address:
pqr23
Branch Code:
105
Modify
203
204
205
206
INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
employee_id:
206
employee Name:
Aniketh
employee Contact:
5643217896
Input
Enter New employee Name:
Koushik
OK
Cancel
204
205
206

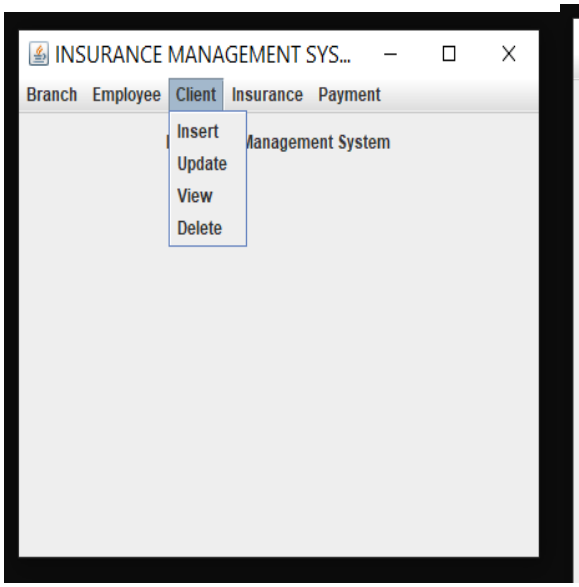

INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
employee_id:
206
employee Name:
Koushik
employee Contact:
5643217896
employee Address:
Message
Updated 1 rows succesfully
OK
204
205
206

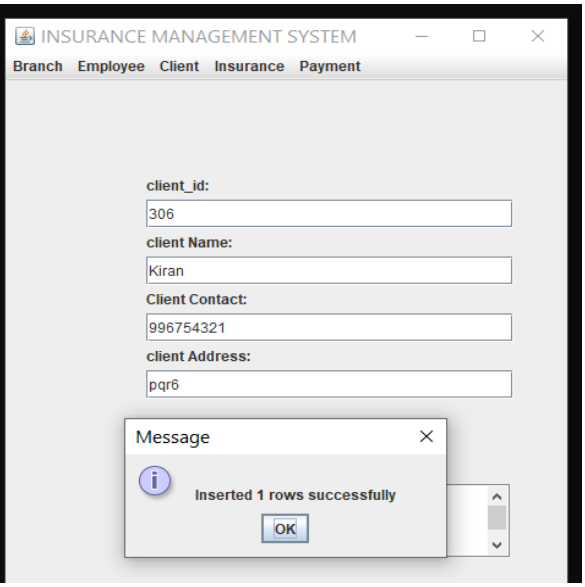


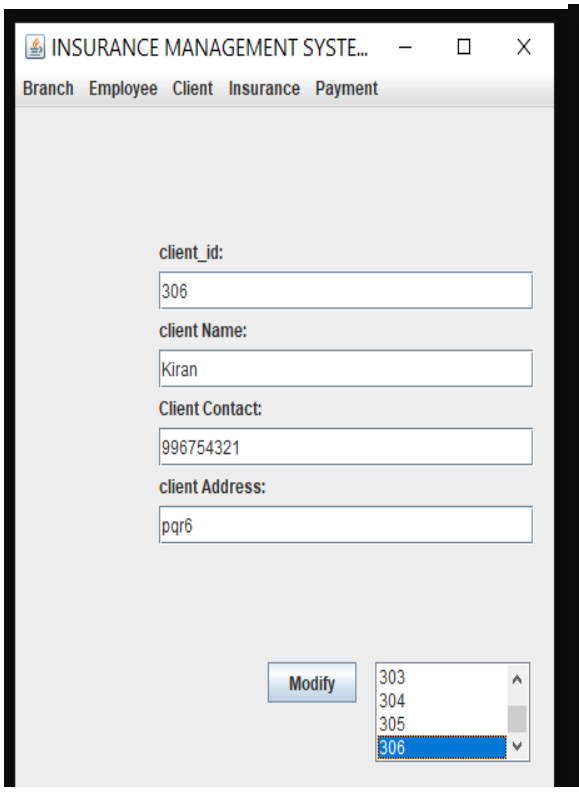


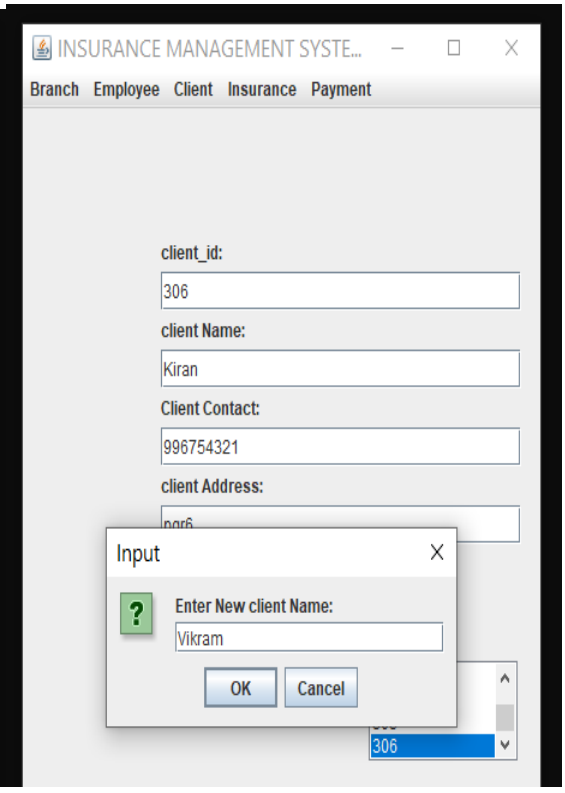
CLIENT TABLE:

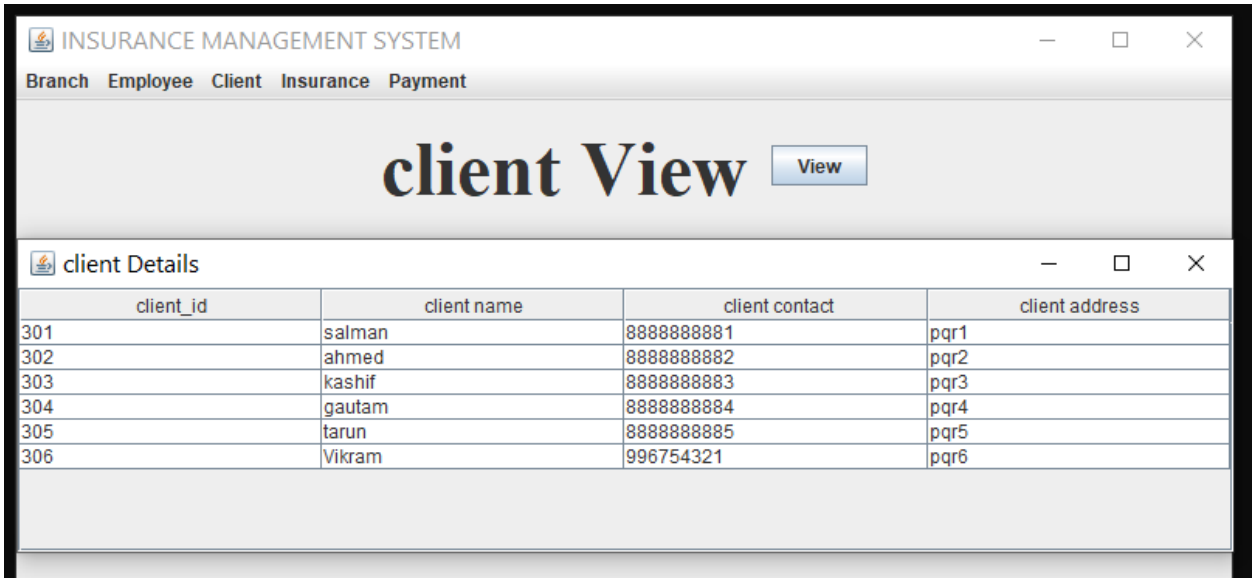
The client_id is an unique value and it can only have a numeric value and not anything else, similarly the client_name can only be a string. In case of any unexpected or incorrect input it throws an exception.

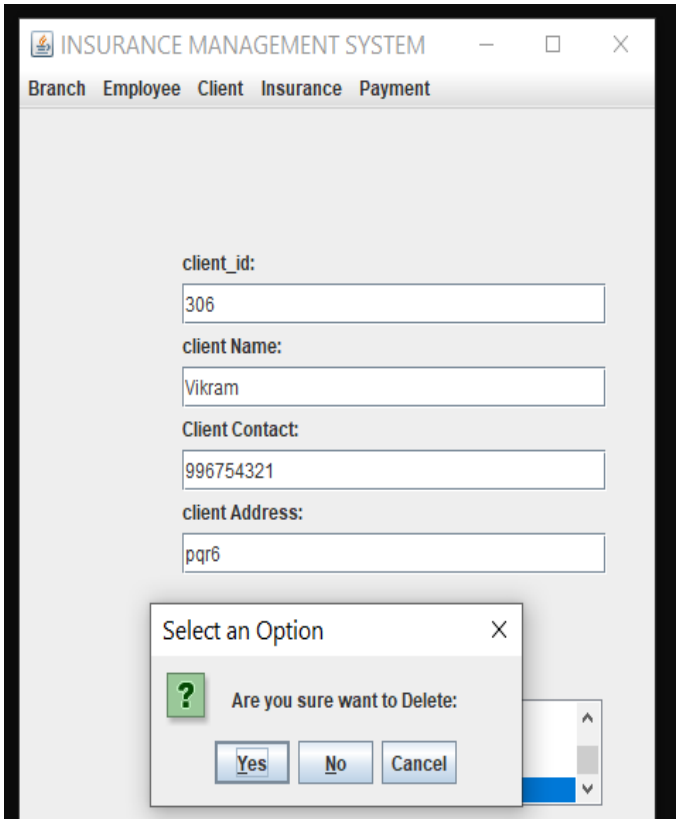

INSURANCE MANAGEMENT SYS...
Branch
Employee
Client
Insurance
Payment
Insert
Update
View
Delete
anagement System

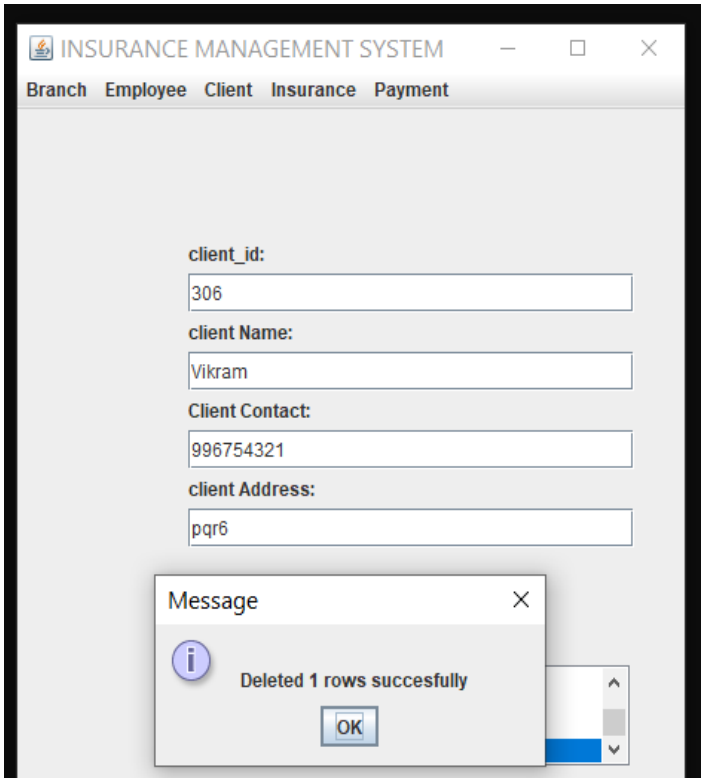

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
client_id:
306
client Name:
Kiran
Client Contact:
996754321
client Address:
pqr6
Message
Inserted 1 rows successfully
OK


INSURANCE MANAGEMENT SYSTE...
Branch
Employee
Client
Insurance
Payment
client_id:
306
client Name:
Kiran
Client Contact:
996754321
client Address:
pqr6
Modify
303
304
305
306


INSURANCE MANAGEMENT SYSTE...
Branch
Employee
Client
Insurance
Payment
client_id:
306
client Name:
Kiran
Client Contact:
996754321
client Address:
Input
Enter New client Name:
Vikram
OK
Cancel
306

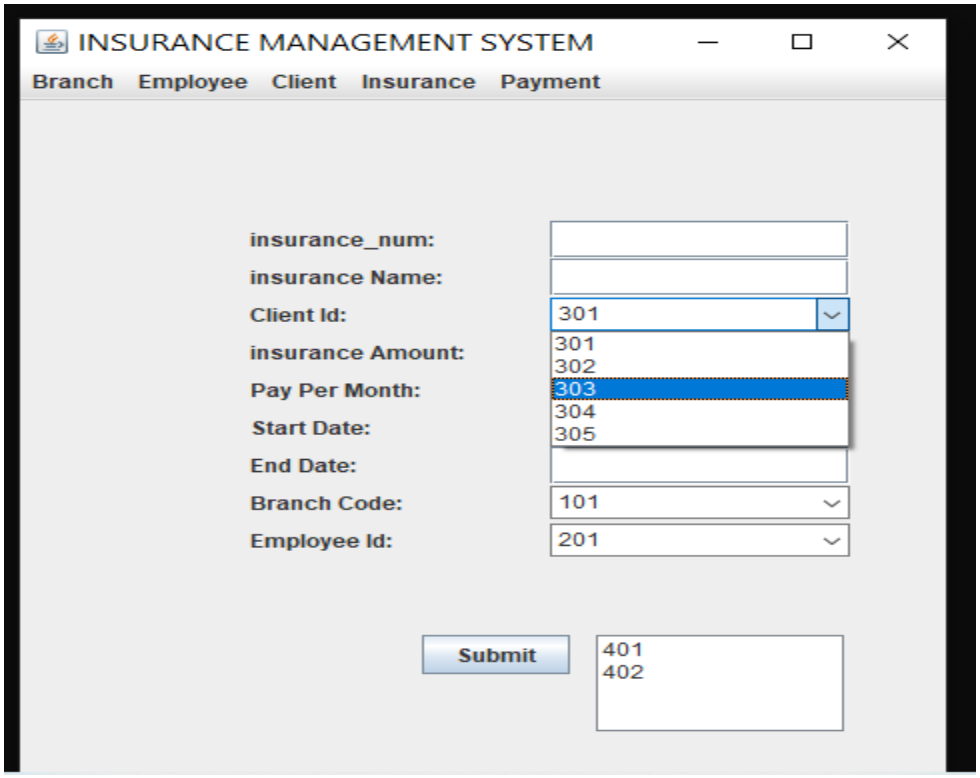

INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
client View
View
client Details
client_id client name client contact client address
301 salman 8888888881 pqr1
302 ahmed 8888888882 pqr2
303 kashif 8888888883 pqr3
304 gautam 8888888884 pqr4
305 tarun 8888888885 pqr5
306 Vikram 996754321 pqr6

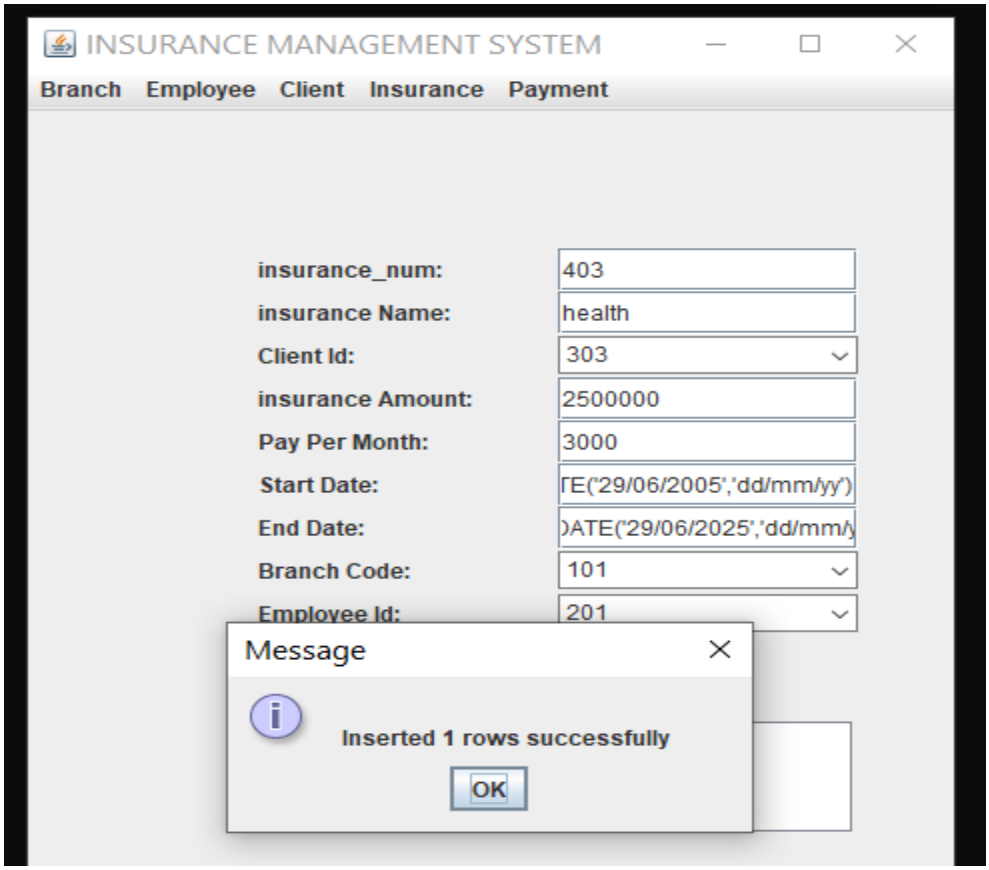

INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
client_id:
306
client Name:
Vikram
Client Contact:
996754321
client Address:
pqr6
Select an Option
Are you sure want to Delete:
Yes
No
Cancel

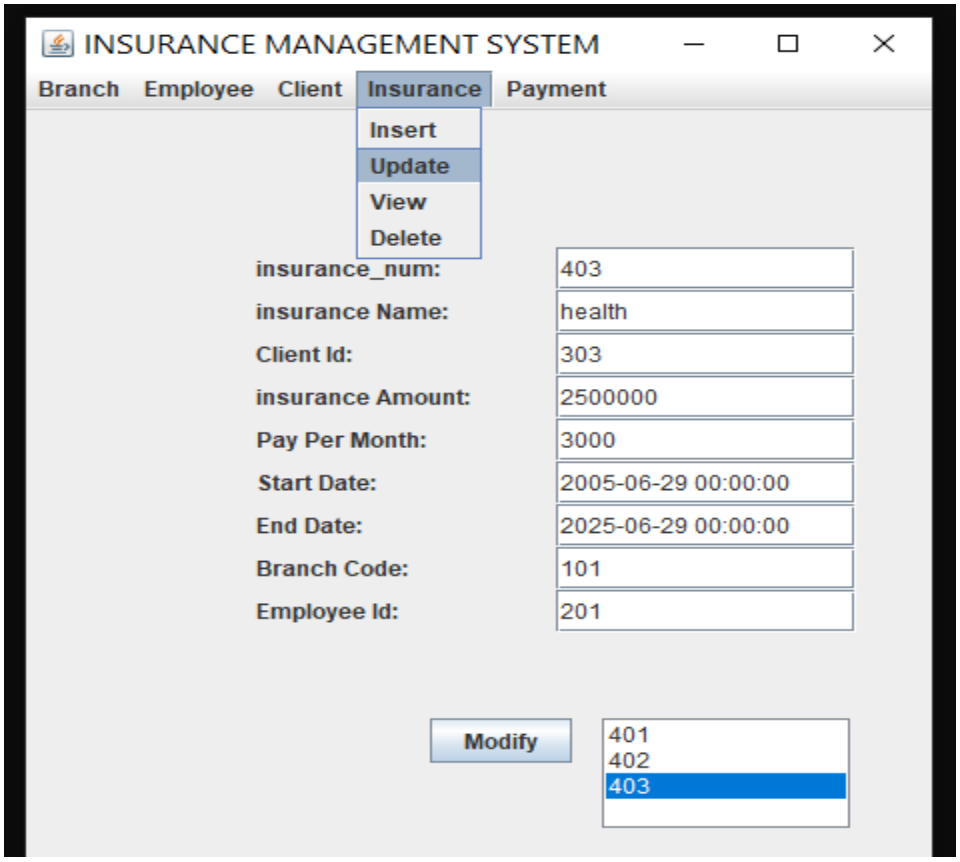


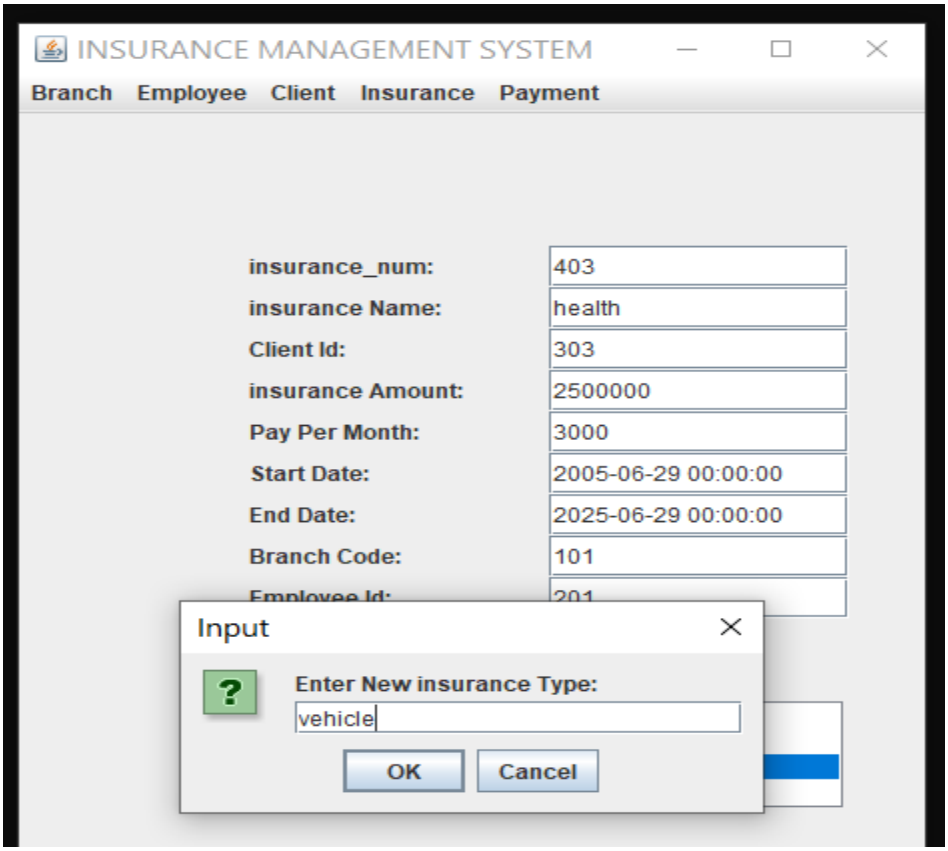
INSURANCE TABLE:

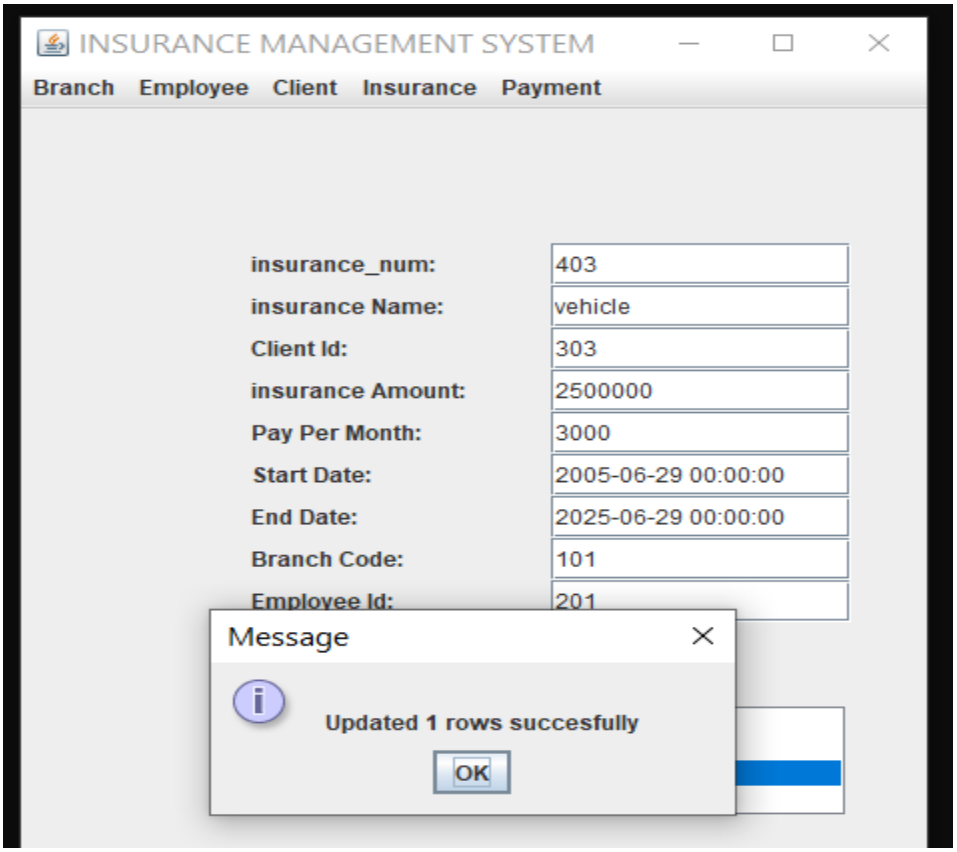
In insurance table insurance_num is primary key and we have drop down for all foreign keys.

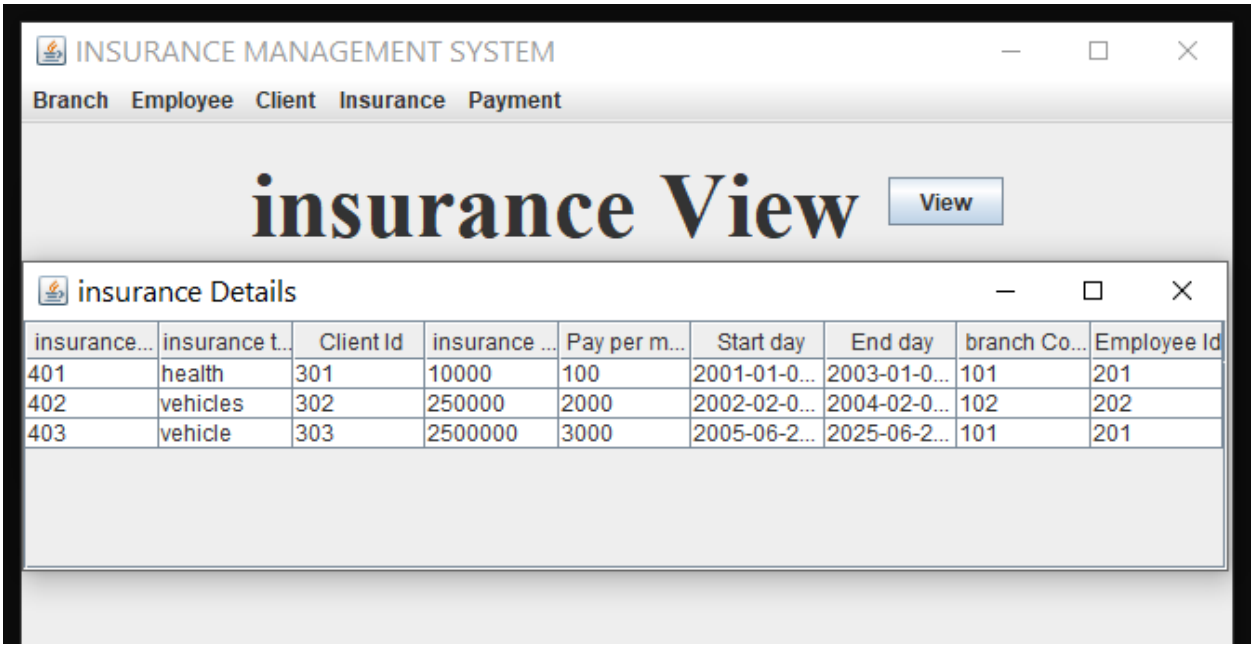


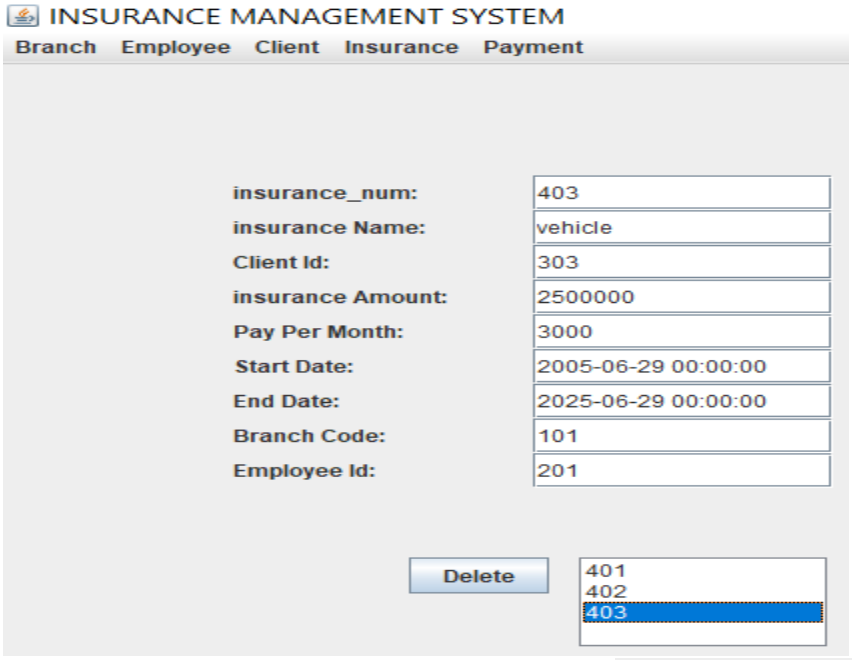

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
insurance_num:
403
insurance Name:
health
Client Id:
303
insurance Amount:
2500000
Pay Per Month:
3000
Start Date:
TE('29/06/2005','dd/mm/yy')
End Date:
)ATE('29/06/2025','dd/mm/y
Branch Code:
101
Employee Id:
201
Message
Inserted 1 rows successfully
OK

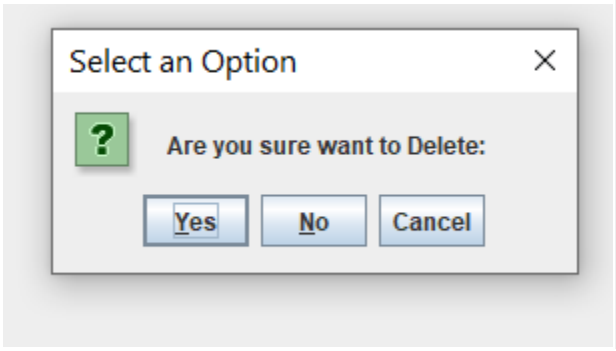

INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Insert
Update
View
Delete
insurance_num:
403
insurance Name:
health
Client Id:
303
insurance Amount:
2500000
Pay Per Month:
3000
Start Date:
2005-06-29 00:00:00
End Date:
2025-06-29 00:00:00
Branch Code:
101
Employee Id:
201
Modify
401
402
403

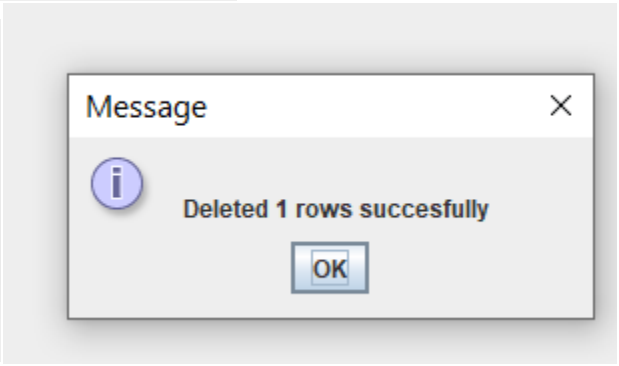

INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
insurance_num: 403
insurance Name: health
Client Id: 303
insurance Amount: 2500000
Pay Per Month: 3000
Start Date: 2005-06-29 00:00:00
End Date: 2025-06-29 00:00:00
Branch Code: 101
Input
Enter New insurance Type:
vehicle
OK
Cancel


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
insurance_num: 403
insurance Name: vehicle
Client Id: 303
insurance Amount: 2500000
Pay Per Month: 3000
Start Date: 2005-06-29 00:00:00
End Date: 2025-06-29 00:00:00
Branch Code: 101
Employee Id: 201
Message
Updated 1 rows succesfully
OK


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
insurance View
View
insurance Details
insurance... insurance t... Client Id insurance ... Pay per m... Start day End day branch Co... Employee Id
401 health 301 10000 100 2001-01-0... 2003-01-0... 101 201
402 vehicles 302 250000 2000 2002-02-0... 2004-02-0... 102 202
403 vehicle 303 2500000 3000 2005-06-2... 2025-06-2... 101 201

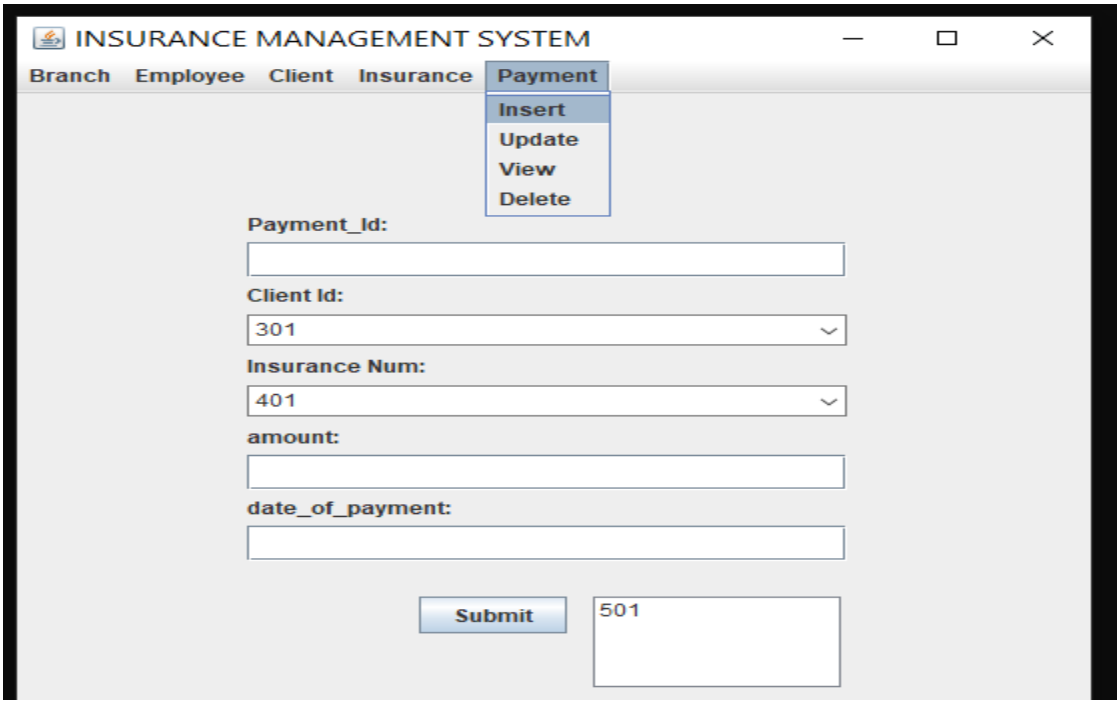

INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
insurance_num: 403
insurance Name: vehicle
Client Id: 303
insurance Amount: 2500000
Pay Per Month: 3000
Start Date: 2005-06-29 00:00:00
End Date: 2025-06-29 00:00:00
Branch Code: 101
Employee Id: 201
Delete
401
402
403

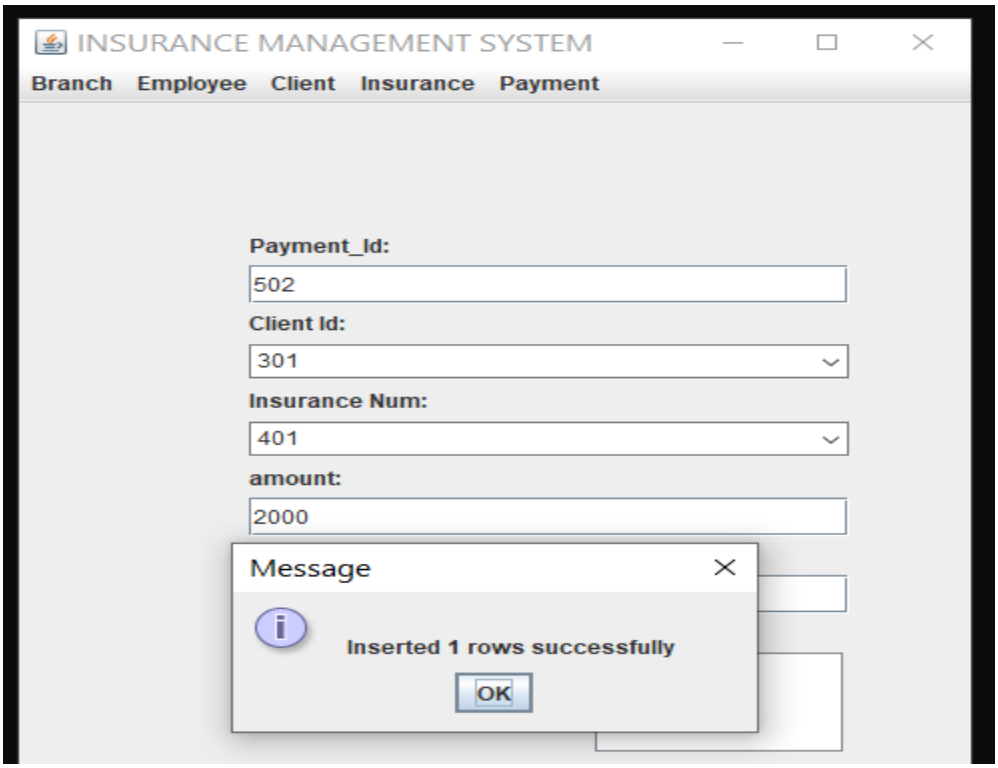

Select an Option
Are you sure want to Delete:
Yes
No
Cancel

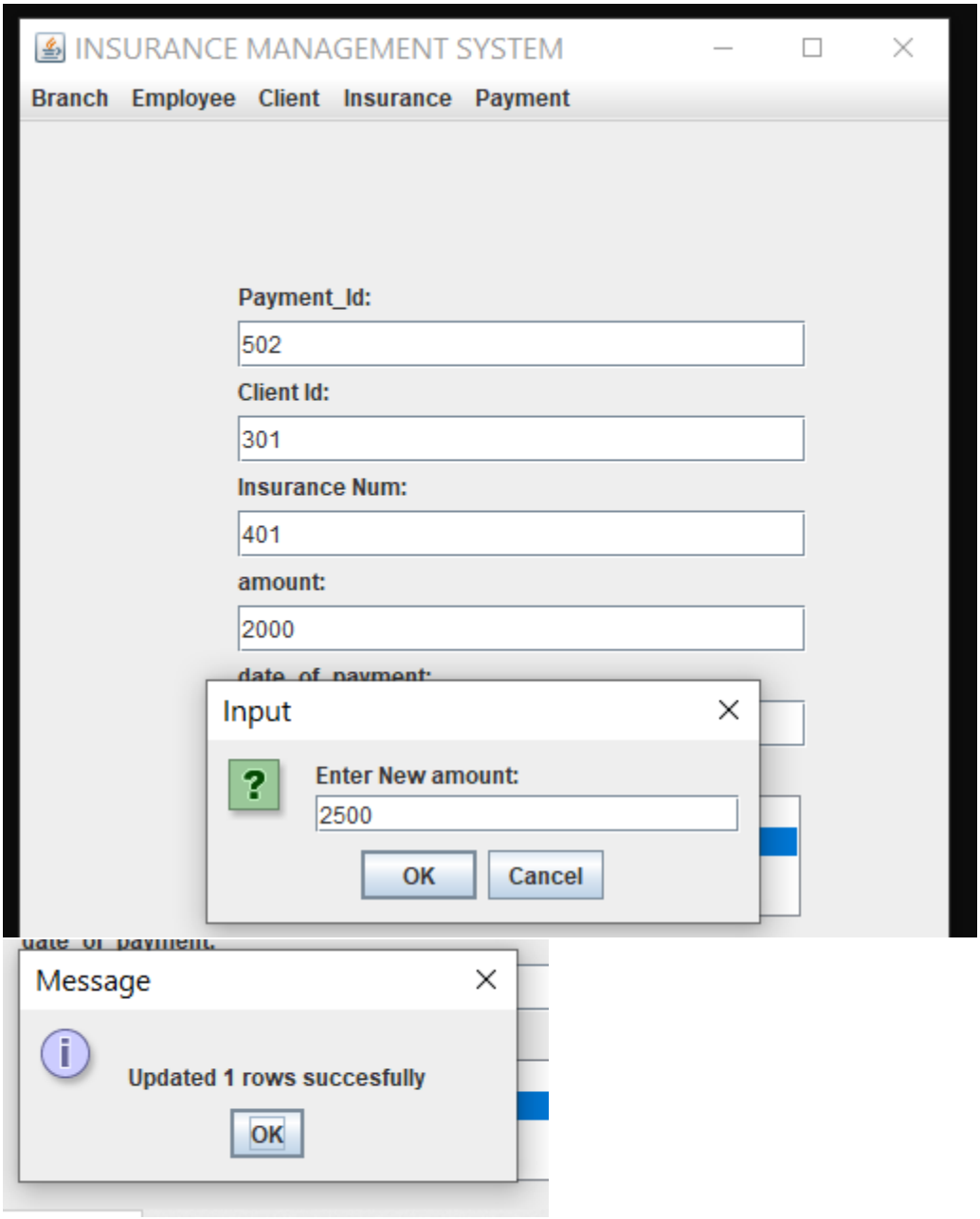

Message
Deleted 1 rows succesfully
OK

PAYMENT TABLE:

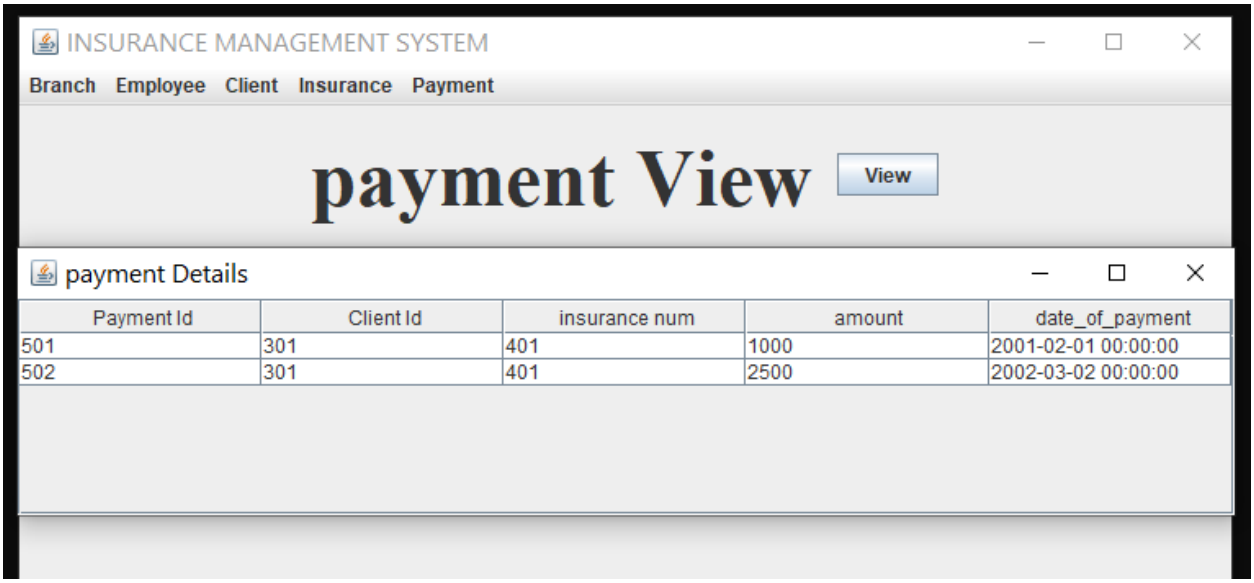
Here in payment table payment id is primary key and foreign keys are client_id , insurance_num and they have drop downs froms which we can select.

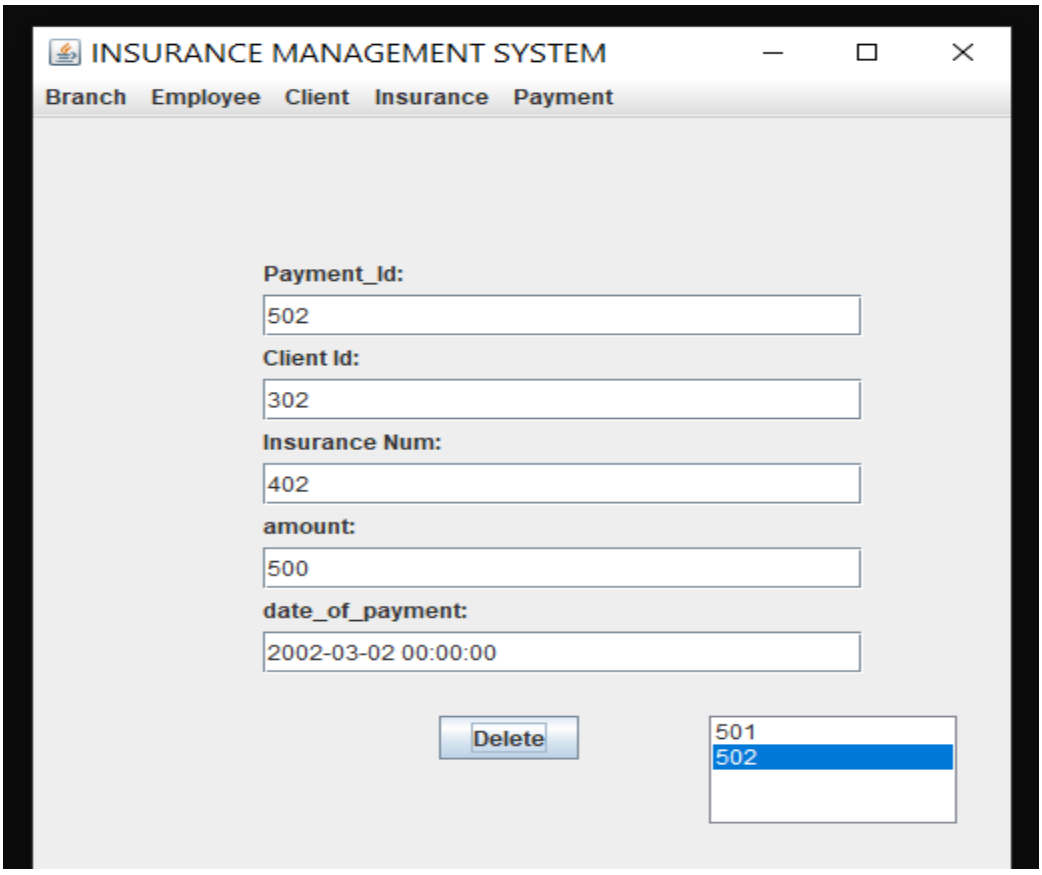


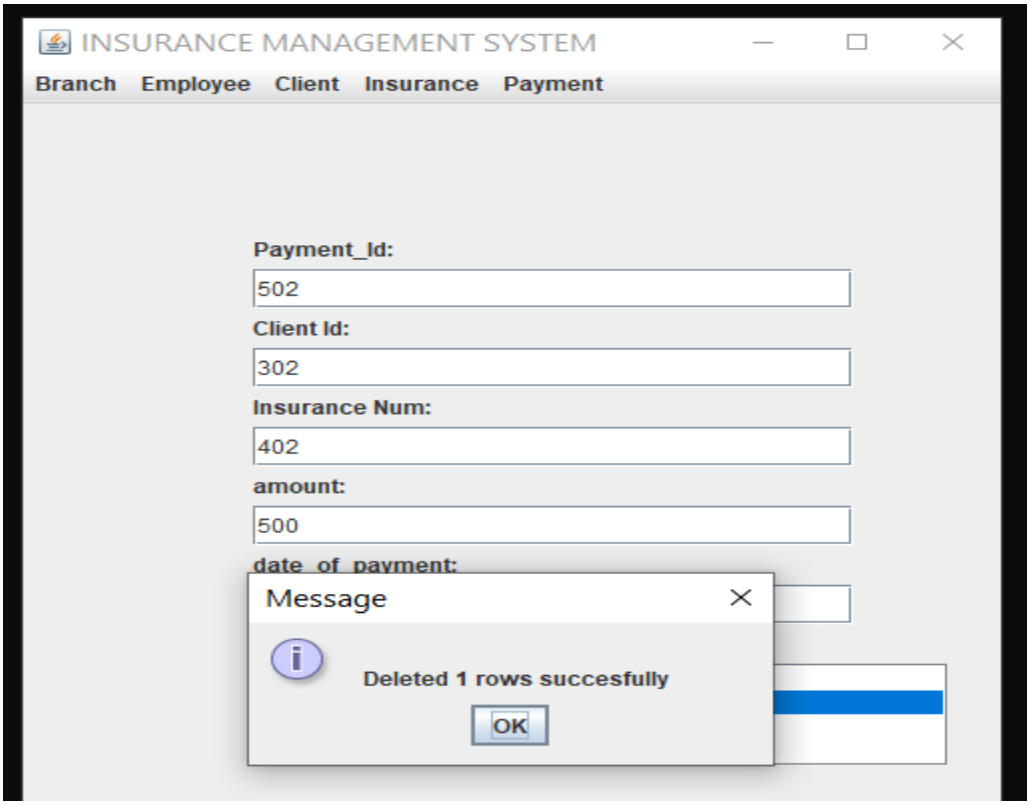



INSURANCE MANAGEMENT SYSTEM
Branch
Employee
Client
Insurance
Payment
Payment_Id:
502
Client Id:
301
Insurance Num:
401
amount:
2000
Input
Enter New amount:
2500
OK
Cancel

Message
Updated 1 rows succesfully
OK


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
payment View
View
payment Details
Payment Id
Client Id
insurance num
amount
date_of_payment
501
301
401
1000
2001-02-01 00:00:00
502
301
401
2500
2002-03-02 00:00:00


INSURANCE MANAGEMENT SYSTEM
Branch Employee Client Insurance Payment
Payment_Id:
502
Client Id:
302
Insurance Num:
402
amount:
500
date_of_payment:
2002-03-02 00:00:00
Delete
501
502



GITHUB LINK

https://github.com/saiprasadkoyalkar/INSURANCE-MANAGEMENT-SYSTEM/

DISCUSSION AND FUTURE WORK

This project contains the basic interaction of applying for insurance by clients and getting insurance with the help of employees. It has a very basic user interface. Future scope would be to make the UI more appealing by using graphics. One more feature would be to allow client-users to upload their financial status , business ,official documents required so that the employees can take a look while they give insurance to them. We can also think of including a feedback system to allow the users to leave their valuable feedback after using this app. Making this feedback to be publicly viewable, would attract many more users to use this app.

REFERENCES

➔ Overview (Java Platform SE 7) (oracle.com)

➔ Java Swing Tutorial - javatpoint

➔ Stack Overflow - Where Developers Learn, Share, & Build Careers