

**JAVA AWT BASED-  
STRICT POLICIES FOR RECOVERY OF DELAY PAYMENT  
- SQL CONNECTIVITY USING JDBC**

A

*Report*

*Submitted in partial fulfilment of the  
Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

**VINUTHNA TATIKONDA <1602-18-737-118>**

Under the guidance of

B.Leelavathy



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31**

**2019-2020**

## **BONAFIDE CERTIFICATE**

This is certify that this project report titled ‘**STRICT POLICIES FOR RECOVERY OF DELAY PAYMENT**’ is a bonafide mini project work of **Vinuthna Tatikonda** bearing H.T.No.**1602-18-737-118** under the guidance of **B.Leelavathy** during the 4<sup>th</sup> Sem B.E for the Academic Year 2019-2020.

**External Examiner**

**Internal Examiner**

**B.Leelavathy**

**Associate Professor**

**Department of Information Technology**

# **ABSTRACT**

Taking and giving money is happening in every sector. In manufacturing, if the buyer fails to make payment of the amount to the supplier, he shall be liable to pay compound interest with monthly rests to the supplier on the amount from the appointed day or, on the date agreed on, at three times of the Bank Rate notified by Reserve Bank. This may not be applicable in all the sectors. Following the policy of legally seizing the property, both the lender and borrower will get advantage. Borrower can buy his lost property again and the lender can get his money recovered. The lender sets the deadline for the borrower to return the money along with interest. If the borrower does not make payment on a timely basis, he will receive notice in the form of letter or mail or a text message. If the borrower does not return the money the lender owe, even after receiving reminders, the lender has the right to legally take his/her property.

## **Aim and Priority of the project :**

To create a **Java GUI based “Strict policies for recovery of Delay Payment”** which has the entities like: Lender, Borrower, Notices and Property whose values are taken from the user. These values are to be inserted, updated and deleted in the database using **JDBC connectivity**.

# INTRODUCTION

## Requirements:

Tables required - 7

- Lender
- Borrower
- Lender\_Borrower
- Borrower\_notices
- Notices
- Property
- Lender\_Property

| ENTITY           | ATTRIBUTES   | DOMAIN   |
|------------------|--|--|
| Lender           | i.Name<br>ii.Lender _id<br>iii.Contact_no<br>iv.Money _lent<br>v.Deadline                        | Varchar2(10)<br>Number(10)<br>Number(20)<br>Number(30)<br>Date                           |
| Borrower         | i.Name<br>ii.Payment _id<br>iii.Contact_no<br>iv. Address<br>v. Payment_status<br>vi. Profession | Varchar2(10)<br>Number(10)<br>Number(10)<br>Varchar2(20)<br>Varchar2(10)<br>Varchar2(15) |
| Lender_Borrower  | i. Lender _id<br>ii.Payment_id<br>iii.Day  | Number(10)<br>Number(10)<br>Date   |
| Borrower_notices | i. Payment_id<br>ii.Mail_id<br>iii.Conformation<br>iv.Date_received                              | Number(10)<br>Varchar2(30)<br>Varchar2(20)<br>Date                                       |
| Property         | i.Ownership<br>ii.Value<br>iii.Acres<br>iv.Location  | Varchar2(10)<br>Number(15)<br>Number(10)<br>Varchar2(20)                                 |
| Notices          | i.Mail_id<br>ii. Contact_info<br>iii.Days_delayed  | Varchar2(30)<br>Number(15)<br>Number(10)   |
| Lender_property  | i.Lender_id<br>ii.Ownwership<br>iii.date_siezed  | Number(10)<br>Varchar2(20)<br>Date   |

## **Architecture and Technology :**

### Software used :

- i. Java Eclipse
- ii. Oracle 11g Database Enterprise Edition Release 11.2.0.1.0-64 bit  
SQL\*Plus Release 9.0.1.3.0.
- iii.

### Java AWT:

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.

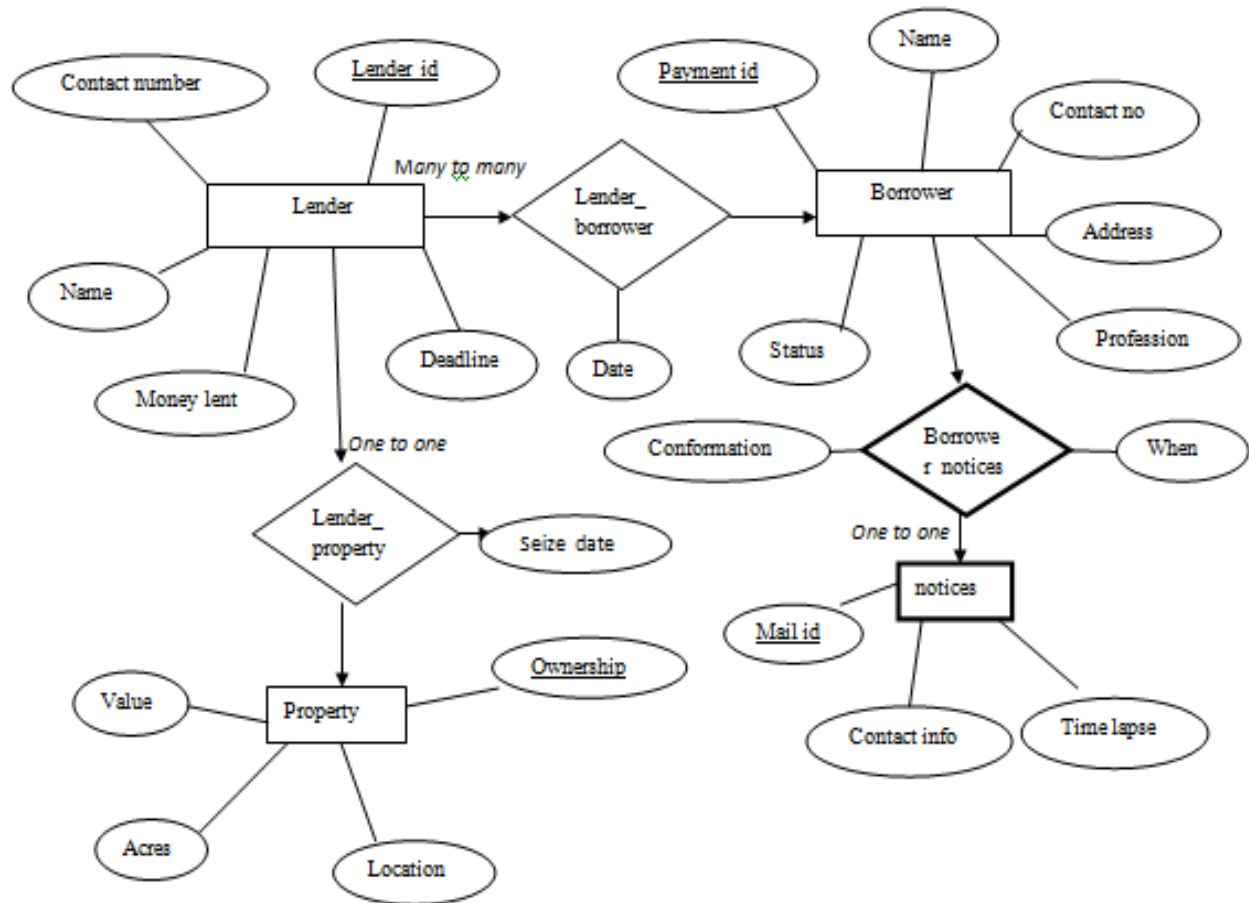
Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS. The java.awt package provides classes for AWT API such as TextField, Label, TextArea, Button, Choice, List ,etc.

### 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 MODEL



Tables:

Here are creation of tables lender, borrower and relationship lender\_borrower between them :

Run SQL Command Line

```
SQL> create table lender(  
2  name varchar2(10),  
3  lender_id number(10) primary key,  
4  contact_no number(20),  
5  money_lent number(30),  
6  deadline date);
```

Table created.

```
SQL> create table borrower(  
2  name varchar2(10),  
3  payment_id number(10),  
4  contact_no number(10),  
5  address varchar2(20),  
6  payment_sttaus varchar2(10),  
7  profession varchar2(15));
```

Table created.

```
SQL> alter table borrower modify(primary key(payment_id));
```

Table altered.

```
SQL> create table lender_borrower(  
2  lender_id number(10),  
3  payment_id number(10),  
4  day date,  
5  foreign key(lender_id) references lender,  
6  foreign key(payment_id) references borrower
```

Table created.

Description of the tables :

```
SQL> desc lender;
```

| Name       | Null?    | Type         |
|------------|----------|--------------|
| NAME       |          | VARCHAR2(10) |
| LENDER_ID  | NOT NULL | NUMBER(10)   |
| CONTACT_NO |          | NUMBER(20)   |
| MONEY_LENT |          | NUMBER(30)   |
| DEADLINE   |          | DATE         |

```
SQL> desc borrower;
```

| Name           | Null?    | Type         |
|----------------|----------|--------------|
| NAME           |          | VARCHAR2(10) |
| PAYMENT_ID     | NOT NULL | NUMBER(10)   |
| CONTACT_NO     |          | NUMBER(10)   |
| ADDRESS        |          | VARCHAR2(20) |
| PAYMENT_STTAUS |          | VARCHAR2(10) |
| PROFESSION     |          | VARCHAR2(15) |

```
SQL> desc lender_borrower;
```

| Name       | Null? | Type       |
|------------|-------|------------|
| LENDER_ID  |       | NUMBER(10) |
| PAYMENT_ID |       | NUMBER(10) |
| DAY        |       | DATE       |

Records of the table :

```
SQL> select * from lender;
```

| NAME     | LENDER_ID | CONTACT_NO | MONEY_LENT | DEADLINE  |
|----------|-----------|------------|------------|-----------|
| sridhar  | 6021      | 109        | 10000      | 05-MAR-20 |
| Srinivas | 6024      | 104        | 18000      | 04-MAR-20 |
| Gopal    | 6025      | 105        | 15500      | 03-MAR-20 |
| Suresh   | 6026      | 106        | 16000      | 02-MAR-20 |
| Kavitha  | 6027      | 107        | 17600      | 03-MAR-20 |
| Rekha    | 6022      | 102        | 15000      | 03-MAR-20 |

6 rows selected.

```
SQL> select * from borrower;
```

| NAME   | PAYMENT_ID | CONTACT_NO | ADDRESS       | PAYMENT_ST | PROFESSION   |
|--------|------------|------------|---------------|------------|--------------|
| kim    | 1033       | 203        | uppal         | paid       | clerk        |
| Priya  | 1031       | 201        | banjara hills | paid       | professor    |
| Suresh | 1032       | 202        | lb nagar      | not paid   | s/w engineer |
| jones  | 1034       | 204        | secundrabad   | paid       | bank manager |
| nick   | 1035       | 205        | ibrahimbagh   | not paid   | teacher      |
| edward | 1036       | 206        | suncity       | not paid   | doctor       |
| ellis  | 1038       | 208        | hitec city    | paid       | lawyer       |

7 rows selected.

```
SQL> select * from lender_borrower;
```

| LENDER_ID | PAYMENT_ID | DAY       |
|-----------|------------|-----------|
| 6025      | 1038       | 04-MAR-20 |
| 6021      | 1034       | 06-MAR-20 |
| 6021      | 1038       | 02-MAR-20 |
| 6027      | 1033       | 03-MAR-20 |
| 6026      | 1031       | 05-MAR-20 |
| 6025      | 1035       | 01-MAR-20 |

6 rows selected.



# Implementation :

## i. Front end programs and connectivity

//CONNECTIVITY PROGRAM :

```
package xyz;
/*import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException; */

import java.sql.*;

class OracleCon
{
public static void main(String args[]){
try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","vinuthna","9989");

Statement stmt=con.createStatement();

con.close();

}catch(Exception e)
{
    System.out.println(e);
}

}
}
```

Thus, the connection from Java to Oracle database is performed and therefore, can be used for inserting, updating and deleting tables in the database directly.

//PROGRAM FOR INSERTING LENDER TABLE :

```
package abc;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class InsertLender extends Frame
{

```

```

        Button insertLenderButton;

        TextField nameText, lender_idText, contact_noText, money_lentText, deadlineText;

        TextArea errorText;

        Connection connection;

        Statement statement;

        public InsertLender()
        {
            try
            {
                Class.forName("oracle.jdbc.driver.OracleDriver");
            }

            catch (Exception e) {

                System.err.println("Unable to find and load driver");

                System.exit(1);
            }

            connectToDB();
        }

        public void connectToDB()
        {
            try
            {
                connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","vinuthna","9989");

                statement = connection.createStatement();

            }

            catch (SQLException connectException) {

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

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

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

                System.exit(1);
            }
        }

        public void buildGUI()
        {

            insertLenderButton = new Button("Insert Lender");

            insertLenderButton.addActionListener(new ActionListener()
            {

                public void actionPerformed(ActionEvent e)

                {
                    try
                    {
                        String query= "INSERT INTO lender VALUES(" + nameText.getText() + "," + lender_idText.getText() + ","
+ contact_noText.getText() + "," + money_lentText.getText() + ","+deadlineText.getText()+ ")";

                        int i = statement.executeUpdate(query);

```

```

        errorText.append("\nInserted " + i + " rows successfully"); }

        catch (SQLException insertException) {

            displaySQLErrors(insertException); }

    }

});

nameText = new TextField(15);          lender_idText = new TextField(15);

contact_noText = new TextField(15);    money_lentText = new TextField(15);

deadlineText = new TextField(15);      errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Name:"));          first.add(nameText);

first.add(new Label("Lender_id:"));      first.add(lender_idText);

first.add(new Label("Contact_no:"));      first.add(contact_noText);

first.add(new Label("Money Lent:"));      first.add(money_lentText);

first.add(new Label("Deadline:"));        first.add(deadlineText);

first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertLenderButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

setLayout(null);

add(first); add(second); add(third);

setTitle("New Lender Creation");

setSize(500, 600);

setVisible(true);

}

private void displaySQLErrors(SQLException e)

{

    errorText.append("\nSQLException: " + e.getMessage() + "\n");

    errorText.append("SQLState: " + e.getSQLState() + "\n");

    errorText.append("VendorError: " + e.getErrorCode() + "\n");

}

public static void main(String[] args)

```

```

{

    InsertLender l = new InsertLender();

    l.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)

        {

            System.exit(0); }

    });

    l.buildGUI();

} }

```

## **//PROGRAM FOR UPDATING LENDER TABLE :**

```

package abc;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class UpdateLender extends Frame

{

    Button updateLenderButton;

    List lenderIDList;

    TextField nameText, lender_idText, contact_noText, money_lentText,deadlineText;

    TextArea errorText;

    Connection connection;

    Statement statement;

    ResultSet rs;

    public UpdateLender()

    {

        try

        {

            Class.forName("oracle.jdbc.driver.OracleDriver");

        }

        catch (Exception e) {

            System.err.println("Unable to find and load driver");

            System.exit(1); }

        connectToDB();

    }

    public void connectToDB()

    {

        try {

            Connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","vinuthna","9989");

            statement = connection.createStatement();


```

```

    }

    catch (SQLException connectException) {

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

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

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

        System.exit(1);

    }

}

private void loadLenders()

{
    try
    {
        rs = statement.executeQuery("SELECT LENDER_ID FROM lender");

        while (rs.next())

        {

            lenderIDList.add(rs.getString("LENDER_ID"));

        }

    }

    catch (SQLException e) {

        displaySQLErrors(e);
    }

}

public void buildGUI()

{

    lenderIDList = new List(10);

    loadLenders();

    add(lenderIDList);

    lenderIDList.addItemListener(new ItemListener()

    {

        public void itemStateChanged(ItemEvent e)

        {

            try

            {

                rs = statement.executeQuery("SELECT * FROM lender where LENDER_ID

=" + lenderIDList.getSelectedItem());

                rs.next();

                nameText.setText(rs.getString("NAME"));

                lender_idText.setText(rs.getString("LENDER_ID"));

                contact_noText.setText(rs.getString("CONTACT_NO"));

                money_lentText.setText(rs.getString("MONEY_LENT"));

                deadlineText.setText(rs.getString("DEADLINE"));

```

```

        }

        catch (SQLException selectException) {

            displaySQLErrors(selectException); }

    }

});

updateLenderButton = new Button("Update Lender");

updateLenderButton.addActionListener(new ActionListener()

{

    public void actionPerformed(ActionEvent e)

    {

        try {

            Statement statement = connection.createStatement();

            int i = statement.executeUpdate("UPDATE lender "

            + "SET name=" + nameText.getText() +

            "",contact_no="+contact_noText.getText()+

            ",money_lent="+money_lentText.getText()+",deadline="+deadlineText.getText()+"    WHERE

lender_id ="

            + lenderIDList.getSelectedItemAt());

            errorText.append("\nUpdated " + i + " rows successfully");

            lenderIDList.removeAll();

            loadLenders();

        }

        catch (SQLException insertException)

        {

            displaySQLErrors(insertException);        }

    }

});

lender_idText = new TextField(15);

lender_idText.setEditable(false);

nameText = new TextField(15);    money_lentText = new TextField(15);

contact_noText = new TextField(15); deadlineText = new TextField(15);

errorText = new TextArea(10, 40);    errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Lender ID:"));    first.add(lender_idText);

first.add(new Label("Name:"));    first.add(nameText);

first.add(new Label("Contact no:"));    first.add(contact_noText);

first.add(new Label("Money Lent:")); first.add(money_lentText);

```

```

        first.add(new Label("Deadline:"));    first.add(deadlineText);

        Panel second = new Panel(new GridLayout(4, 1));

        second.add(updateLenderButton);

        Panel third = new Panel();

        third.add(errorText);

        add(first); add(second); add(third);

        setTitle("Update Lender");

        setSize(500, 600);

        setLayout(new FlowLayout());

        setVisible(true);
    }

    private void displaySQLExceptions(SQLException e)
    {

        errorText.append("\nSQLException: " + e.getMessage() + "\n");

        errorText.append("SQLState:  " + e.getSQLState() + "\n");

        errorText.append("VendorError: " + e.getErrorCode() + "\n");

    }

    public static void main(String[] args)

    {

        UpdateLender l = new UpdateLender();

        l.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e)

            {

                System.exit(0);

            }

        });

        l.buildGUI();

    }

}

```

## //PROGRAM FOR DELETING LENDER TABLE :

```

package abc;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class DeleteLender extends Frame

{

    Button deleteLenderButton;

```

```

List lenderIDList;

TextField nameText, lender_idText, contact_noText, money_lentText, deadlineText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public DeleteLender()

{
    try {

        Class.forName("oracle.jdbc.driver.OracleDriver");

    }

    catch (Exception e) {

        System.err.println("Unable to find and load driver");

        System.exit(1);    }

    connectToDB();

}

public void connectToDB()

{
    try {

        connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","vinuthna","9989");

        statement = connection.createStatement();

    }

    catch (SQLException connectException) {

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

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

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

        System.exit(1); }

}

private void loadLenders()

{
    try {

        rs = statement.executeQuery("SELECT * FROM lender");

        while (rs.next())

        {

            lenderIDList.add(rs.getString("LENDER_ID"));

        }

    }

    catch (SQLException e)

    {
        displaySQLErrors(e);    }
}

```



```

    }

    public void buildGUI()
    {

        lenderIDList = new List(10);

        loadLenders();

        add(lenderIDList);

        lenderIDList.addItemListener(new ItemListener()
        {

            public void itemStateChanged(ItemEvent e)

            {
                try {

                    rs = statement.executeQuery("SELECT * FROM lender");

                    while (rs.next())

                    {

                        if(rs.getString("LENDER_ID").equals(lenderIDList.getSelectedItem()))

                        break;

                    }

                    if (!rs.isAfterLast())

                    {

                        lender_idText.setText(rs.getString("LENDER_ID"));

                        nameText.setText(rs.getString("NAME"));

                        contact_noText.setText(rs.getString("CONTACT_NO"));

                        money_lentText.setText(rs.getString("MONEY_LENT"));

                        deadlineText.setText(rs.getString("DEADLINE"));

                    }

                }

                catch (SQLException selectException)

                {
                    displaySQLErrors(selectException);
                }

            }

        });

        deleteLenderButton = new Button("Delete Lender");

        deleteLenderButton.addActionListener(new ActionListener()
        {

            public void actionPerformed(ActionEvent e)

            {
                try {

                    Statement statement = connection.createStatement();

                    int i = statement.executeUpdate("DELETE FROM lender WHERE LENDER_ID =" +

lenderIDList.getSelectedItem());

```

```

        errorText.append("\nDeleted " + i + " rows successfully");

        lender_idText.setText(null);        nameText.setText(null);

        contact_noText.setText(null);        money_lentText.setText(null);

        deadlineText.setText(null);        lenderIDList.removeAll();

        loadLenders();

    }

    catch (SQLException insertException) {

        displaySQLErrors(insertException);}

    }

});

lender_idText = new TextField(15); lender_idText.setEditable(false);

nameText = new TextField(15);    money_lentText = new TextField(15);

contact_noText = new TextField(15); deadlineText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();        first.setLayout(new GridLayout(5, 2));

first.add(new Label("Lender ID:")); first.add(lender_idText);

first.add(new Label("Name:"));    first.add(nameText);

first.add(new Label("Contact no:")); first.add(contact_noText);

first.add(new Label("Money Lent:")); first.add(money_lentText);

first.add(new Label("Deadline:")); first.add(deadlineText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(deleteLenderButton);

Panel third = new Panel();        third.add(errorText);

add(first); add(second); add(third);

setTitle("Remove Lender");

setSize(450, 600);    setLayout(new FlowLayout());

setVisible(true);

}

private void displaySQLErrors(SQLException e)

{

    errorText.append("\nSQLException: " + e.getMessage() + "\n");

    errorText.append("SQLState:  " + e.getSQLState() + "\n");

    errorText.append("VendorError: " + e.getErrorCode() + "\n");

}

public static void main(String[] args)

```

```

{

    DeleteLender d = new DeleteLender();

    d.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)

        {
            System.exit(0);
        }

    });

    d.buildGUI();
}

```

## //MAIN PROGRAM:

```

package abc;

import java.awt.*;

import java.awt.event.*;

public class PoliciesForDelayPayment extends Frame implements ActionListener{

    String msg = "";
    Label ll;

    InsertLender inl;;
    UpdateLender upl;
    DeleteLender dell;

    InsertBorrower inb;
    UpdateBorrower upb;
    DeleteBorrower delb;

    InsertProperty inp;
    UpdateProperty upp;
    DeleteProperty delp;

    InsertNotices inn;
    UpdateNotices upn;
    DeleteNotices deln;

    InsertLender_Borrower inlb;
    UpdateLender_Borrower uplb;
    DeleteLender_Borrower dellb;

    InsertBorrower_Notices inbn;
    UpdateBorrower_Notices upbn;
    DeleteBorrower_Notices delbn;

    InsertLender_Property inlp;
    UpdateLender_Property;
    DeleteLender_Property delp;

    PoliciesForDelayPayment()

    {

        ll = new Label();
        ll.setAlignment(Label.CENTER);

        ll.setBounds(15,150,350,150);
        ll.setText("Welcome to Life Pay Policies!!!");

        add(ll);
        MenuBar mbar = new MenuBar();

        setMenuBar(mbar);

        MenuItem item1, item2, item3,item4, item5,item6, item7, item8,item9,item10, item11,item12,item13,item14,item15,
        item16,item17,item18,item19,item20,item21;

        Menu lender = new Menu("Lenders");

        lender.add(item1 = new MenuItem("Insert Lender"));

        lender.add(item2 = new MenuItem("View Lender"));

        lender.add(item3 = new MenuItem("Delete Lender"));

        Menu borrower = new Menu("Borrowers");

        borrower.add(item4 = new MenuItem("Insert Borrower"));
    }
}

```

```

borrower.add(item5 = new MenuItem("View Borrower"));

borrower.add(item6 = new MenuItem("Delete Borrower"));

Menu property = new Menu("Properties");

property.add(item7 = new MenuItem("Insert Property"));

property.add(item8 = new MenuItem("View Property"));

property.add(item9 = new MenuItem("Delete Property"));

Menu notices = new Menu("Notices");

notices.add(item10 = new MenuItem("Insert Notices"));

notices.add(item11 = new MenuItem("View Notices"));

notices.add(item12 = new MenuItem("Delete Notices"));

Menu l_b = new Menu("Lender_Borrower");

l_b.add(item13 = new MenuItem("Insert Lender_Borrower"));

l_b.add(item14 = new MenuItem("View Lender_Borrower"));

l_b.add(item15 = new MenuItem("Delete Lender_Borrower"));

Menu b_n = new Menu("Borrower_Notice");

b_n.add(item16 = new MenuItem("Insert Borrower_Notice"));

b_n.add(item17 = new MenuItem("View Borrower_Notice"));

b_n.add(item18 = new MenuItem("Delete Borrower_Notice"));

Menu l_p = new Menu("Lender_Property");

l_p.add(item19 = new MenuItem("Insert Lender_Property"));

l_p.add(item20 = new MenuItem("View Lender_Property"));

l_p.add(item21 = new MenuItem("Delete Lender_Property"));

mbar.add(lender);    mbar.add(borrower); mbar.add(property);

mbar.add(notices);   mbar.add(l_b);      mbar.add(b_n);      mbar.add(l_p);

item1.addActionListener(this);    item2.addActionListener(this);

item3.addActionListener(this);    item4.addActionListener(this);

item5.addActionListener(this);    item6.addActionListener(this);

item7.addActionListener(this);    item8.addActionListener(this);

item9.addActionListener(this);    item10.addActionListener(this);

item11.addActionListener(this);    item12.addActionListener(this);

item13.addActionListener(this);    item14.addActionListener(this);

item15.addActionListener(this);    item16.addActionListener(this);

item17.addActionListener(this);    item18.addActionListener(this);

item19.addActionListener(this);    item20.addActionListener(this);

item21.addActionListener(this);

addWindowListener(new WindowAdapter(){

```

```

        public void windowClosing(WindowEvent we) {

            System.exit(0);        }

    });

    setTitle("Strict Policies For Recovery of Delay Payment");

    setFont(new Font("Dialog", Font.ITALIC, 21));

    setLayout(null);        setSize(500, 450);        setVisible(true);

}

public void actionPerformed(ActionEvent e)

{

    String arg = e.getActionCommand();

    if(arg.equals("Insert Lender"))    {

        inl = new InsertLender();

        inl.addWindowListener(new WindowAdapter(){

            public void windowClosing(WindowEvent e)    {

                inl.dispose(); }

        });

        inl.buildGUI();        }

    else if(arg.equals("View Lender")) {

        upl = new UpdateLender();

        upl.addWindowListener(new WindowAdapter(){

            public void windowClosing(WindowEvent e)    {

                upl.dispose(); }

        });

        upl.buildGUI();        }

    else if(arg.equals("Delete Lender")) {

        dell = new DeleteLender();

        dell.addWindowListener(new WindowAdapter(){

            public void windowClosing(WindowEvent e)    {

                dell.dispose();        }

        });

        dell.buildGUI();        }

    else if(arg.equals("Insert Borrower")) {

        inb = new InsertBorrower();

        inb.addWindowListener(new WindowAdapter(){

            public void windowClosing(WindowEvent e)    {

                inb.dispose(); }

```

```

    });

    inb.buildGUI(); }

else if(arg.equals("View Borrower")) {

    upb = new UpdateBorrower();

    upb.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            upb.dispose(); }

    });

    upb.buildGUI(); }

else if(arg.equals("Delete Borrower")) {

    delb = new DeleteBorrower();

    delb.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            delb.dispose();        }

    });

    delb.buildGUI(); }

else if(arg.equals("Insert Property")) {

    inp = new InsertProperty();

    inp.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            inp.dispose(); }

    });

    inp.buildGUI(); }

else if(arg.equals("View Property")) {

    upp = new UpdateProperty();

    upp.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            upp.dispose();        }

    });

    upp.buildGUI(); }

else if(arg.equals("Delete Property")) {

    delp = new DeleteProperty();

    delp.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            delp.dispose();        }

    });

```

```

        delp.buildGUI();    }

else if(arg.equals("Insert Notices")) {

    inn = new InsertNotices();

    inn.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            inn.dispose();    }

    });

    inn.buildGUI();    }

else if(arg.equals("View Notices")) {

    upn = new UpdateNotices();

    upn.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            upn.dispose();    }

    });

    upn.buildGUI();    }

else if(arg.equals("Delete Notices")) {

    deln = new DeleteNotices();

    deln.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)

        {

            deln.dispose(); }

    });

    deln.buildGUI();    }

else if(arg.equals("Insert Lender_Borrower"))

{

    inlb = new InsertLender_Borrower();

    inlb.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            inlb.dispose();    }

    });

    inlb.buildGUI();

}

else if(arg.equals("View Lender_Borrower"))

{

    uplb = new UpdateLender_Borrower();

    uplb.addWindowListener(new WindowAdapter(){

```

```

        public void windowClosing(WindowEvent e)    {

            uplb.dispose();        }

    });

    uplb.buildGUI();
}

else if(arg.equals("Delete Lender_Borrower"))
{

    dellb = new DeleteLender_Borrower();

    dellb.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            dellb.dispose();

        }

    });

    dellb.buildGUI();
}

else if(arg.equals("Insert Borrower_Notice"))
{

    inbn = new InsertBorrower_Notices();

    inbn.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            inbn.dispose();        }

    });

    inbn.buildGUI();
}

else if(arg.equals("View Borrower_notice"))
{

    upbn = new UpdateBorrower_Notices();

    upbn.addWindowListener(new WindowAdapter(){

        public void windowClosing(WindowEvent e)    {

            upbn.dispose();        }

    });

    upbn.buildGUI();
}

else if(arg.equals("Delete Borrower_notice"))
{

    delbn = new DeleteBorrower_Notices();

```



```

        delbn.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)    {
                delbn.dispose();
            }
        });

        delbn.buildGUI();
    }

    else if(arg.equals("Insert Lender_Borrower"))
    {

        inlb = new InsertLender_Borrower();

        inlb.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)    {
                inlb.dispose();            }
        });

        inlb.buildGUI();
    }

    else if(arg.equals("View Lender_Property"))
    {

        uplp = new UpdateLender_Property();

        uplp.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)    {
                uplp.dispose();
            }
        });

        uplp.buildGUI();
    }

    else if(arg.equals("Delete Lender_Property"))
    {

        dellp = new DeleteLender_Property();

        dellp.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)    {
                dellp.dispose();            }
        });

        dellp.buildGUI();
    }
}

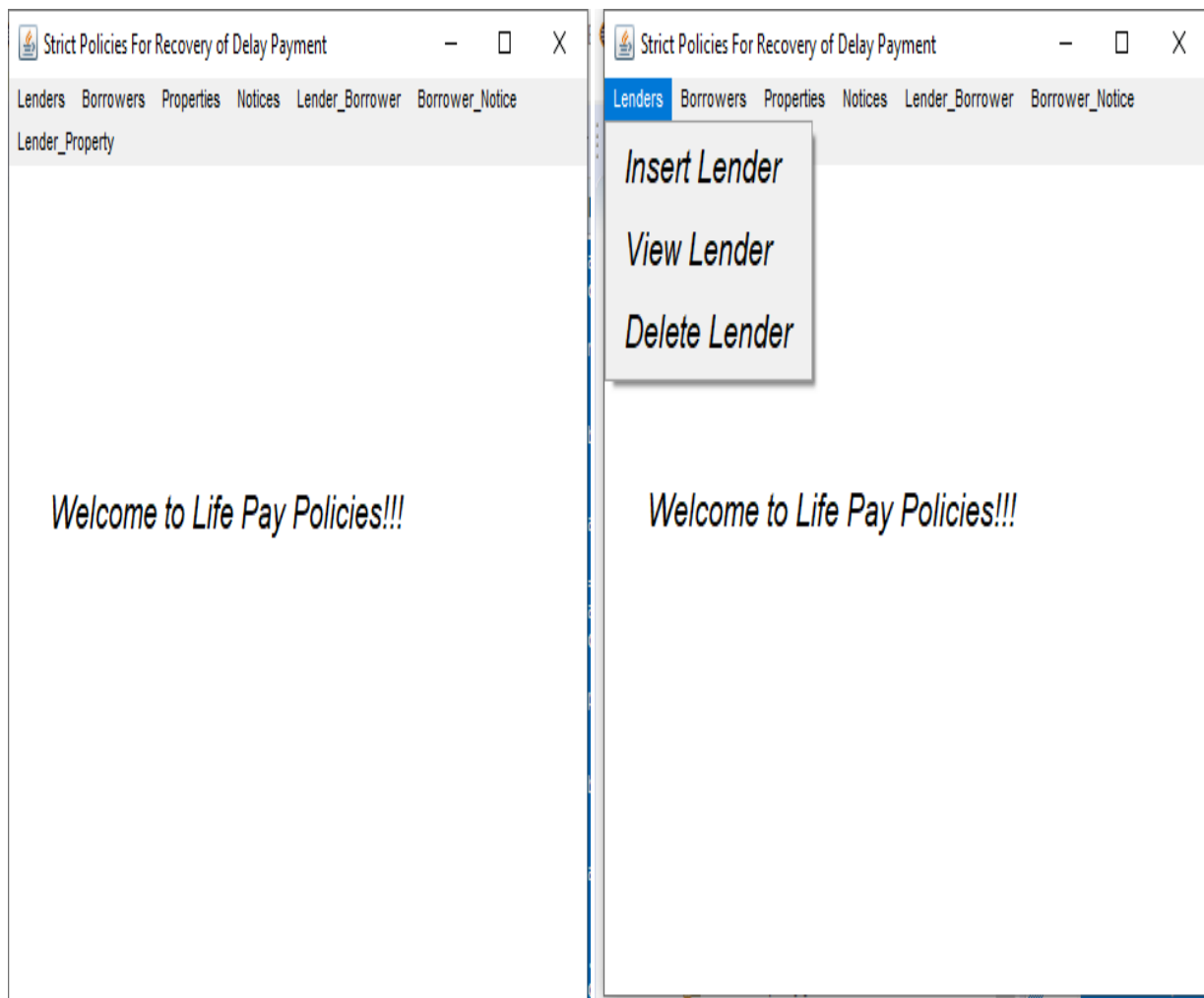
```

```

public static void main(String ... args)
{
    new PoliciesForDelayPayment();
}
}

```

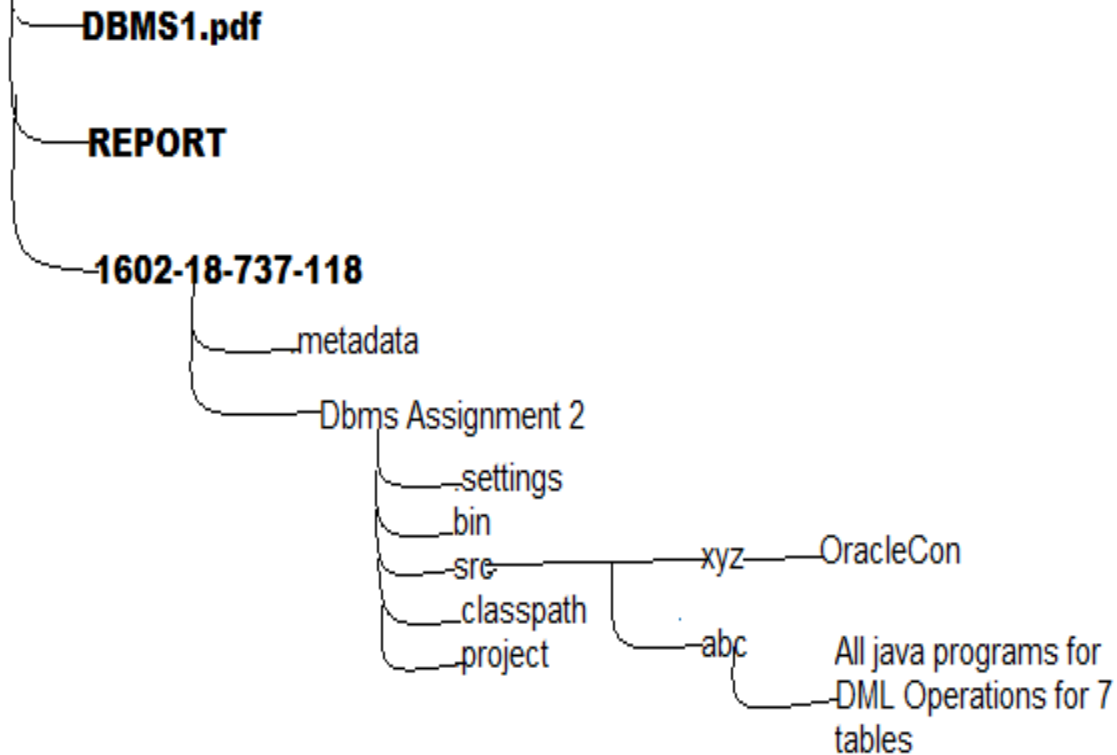
After executing main program, the following frames are displayed :





## Github Link & Folder Structure :

Link- <https://github.com/Vinuthna123/DBMS-PROJECT.git>

### GitHub Link


























| 1602-18-737-118   |                    |             |      |  |
|---|--------------------|-------------|------|--|
| <input type="checkbox"/> Name   | Date modified      | Type        | Size |  |
|  .metadata         | 10/11/2019 9:46 AM | File folder |      |  |
|  Dbms assignment 2 | 08/03/2020 6:13 PM | File folder |      |  |


> 1602-18-737-118 > Dbms assignment 2 > src

| <input type="checkbox"/> Name | Date modified        | Type        | Size |
|-------------------------------|----------------------|-------------|------|
| abc                           | 15/04/2020 7:52 PM   | File folder |      |
| xyz                           | 09/03/2020 10:36 ... | File folder |      |

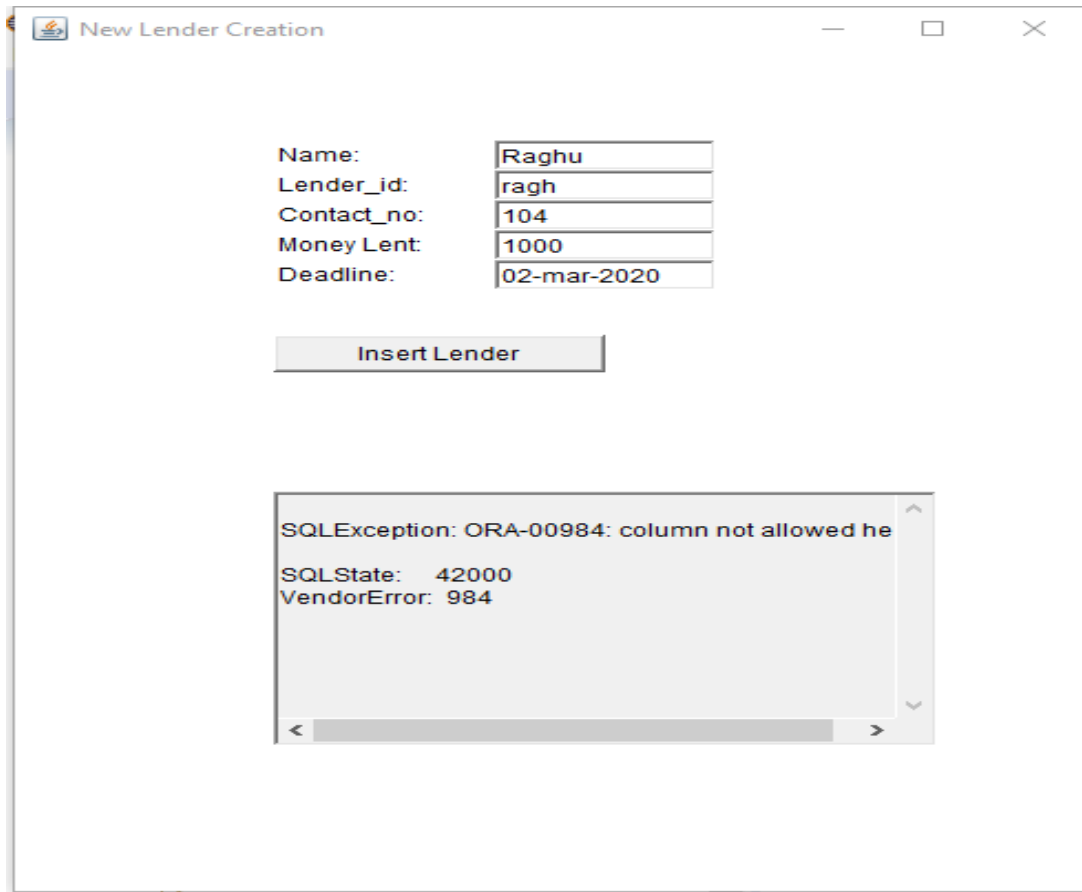
1602-18-737-118 > Dbms assignment 2 > src > abc

| <input type="checkbox"/> Name  | Date modified        | Type      | Size  |
|--|----------------------|-----------|-------|
|  DeleteBorrower           | 09/04/2020 2:32 PM   | JAVA File | 5 KB  |
|  DeleteBorrower_Notices   | 09/04/2020 2:30 PM   | JAVA File | 5 KB  |
|  DeleteLender             | 14/04/2020 6:16 PM   | JAVA File | 5 KB  |
|  DeleteLender_Borrower    | 09/04/2020 2:34 PM   | JAVA File | 5 KB  |
|  DeleteLender_Property    | 09/04/2020 2:36 PM   | JAVA File | 5 KB  |
|  DeleteNotices            | 15/03/2020 3:28 PM   | JAVA File | 5 KB  |
|  DeleteProperty           | 15/03/2020 3:27 PM   | JAVA File | 5 KB  |
|  InsertBorrower           | 15/04/2020 8:31 PM   | JAVA File | 4 KB  |
|  InsertBorrower_Notices   | 09/04/2020 3:15 PM   | JAVA File | 4 KB  |
|  InsertLender             | 15/04/2020 7:57 PM   | JAVA File | 4 KB  |
|  InsertLender_Borrower    | 09/04/2020 8:03 AM   | JAVA File | 4 KB  |
|  InsertLender_Property    | 09/04/2020 12:25 ... | JAVA File | 4 KB  |
|  InsertNotices            | 15/03/2020 3:32 PM   | JAVA File | 3 KB  |
|  InsertProperty           | 15/03/2020 3:34 PM   | JAVA File | 4 KB  |
|  PoliciesForDelayPayment  | 15/04/2020 10:06 ... | JAVA File | 10 KB |
|  UpdateBorrower          | 15/03/2020 3:36 PM   | JAVA File | 5 KB  |
|  UpdateBorrower_Notices | 09/04/2020 12:18 ... | JAVA File | 5 KB  |
|  UpdateLender           | 15/03/2020 3:40 PM   | JAVA File | 5 KB  |
|  UpdateLender_Borrower  | 09/04/2020 10:34 ... | JAVA File | 4 KB  |
|  UpdateLender_Property  | 09/04/2020 12:41 ... | JAVA File | 4 KB  |
|  UpdateNotices          | 15/03/2020 3:43 PM   | JAVA File | 5 KB  |
|  UpdateProperty         | 15/03/2020 3:46 PM   | JAVA File | 5 KB  |
|  WrongInputException    | 15/04/2020 7:53 PM   | JAVA File | 1 KB  |

1602-18-737-118 > Dbms assignment 2 > src > xyz

| <input type="checkbox"/> Name   | Date modified      | Type      | Size |
|---|--------------------|-----------|------|
|  OracleCon | 14/04/2020 1:22 PM | JAVA File | 1 KB |

## Testing :



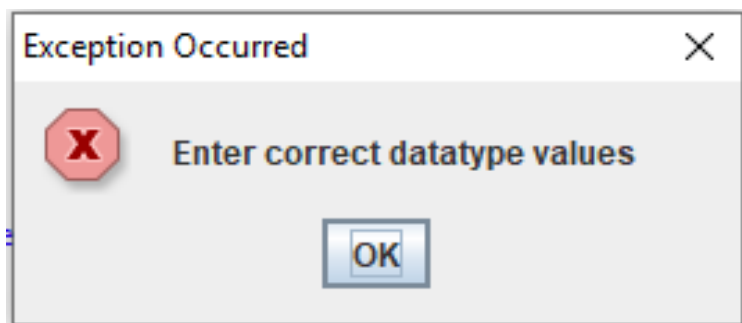
The screenshot shows a window titled "New Lender Creation" with the following fields and values:

|             |             |
|-------------|-------------|
| Name:       | Raghu       |
| Lender_id:  | ragh        |
| Contact_no: | 104         |
| Money Lent: | 1000        |
| Deadline:   | 02-mar-2020 |

Below the fields is an "Insert Lender" button. Below the button is a text area displaying an SQL exception error:

```
SQLException: ORA-00984: column not allowed here  
SQLState: 42000  
VendorError: 984
```

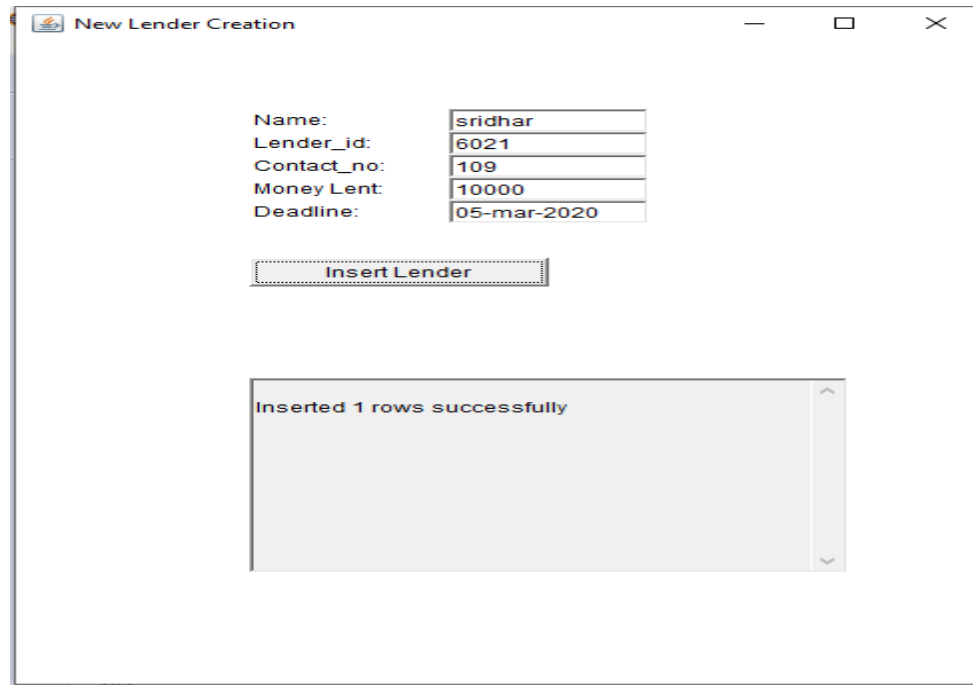
If incorrect values are entered which mismatch datatypes, the following pop up box is shown:



# RESULTS :

Here are DML operations for two tables Lender and Borrower

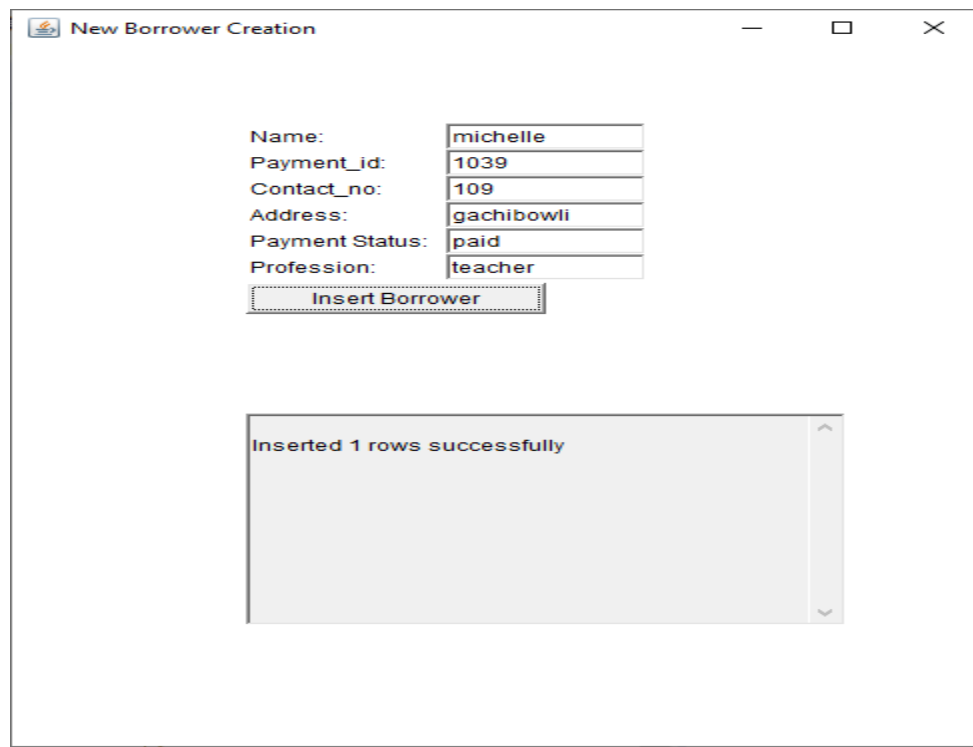
## i.Insert :



The screenshot shows a window titled "New Lender Creation". It contains a form with the following fields and values:

|             |             |
|-------------|-------------|
| Name:       | sridhar     |
| Lender_id:  | 6021        |
| Contact_no: | 109         |
| Money Lent: | 10000       |
| Deadline:   | 05-mar-2020 |

Below the form is a button labeled "Insert Lender". At the bottom of the window, a message box states "Inserted 1 rows successfully".



The screenshot shows a window titled "New Borrower Creation". It contains a form with the following fields and values:

|                 |            |
|-----------------|------------|
| Name:           | michelle   |
| Payment_id:     | 1039       |
| Contact_no:     | 109        |
| Address:        | gachibowli |
| Payment Status: | paid       |
| Profession:     | teacher    |

Below the form is a button labeled "Insert Borrower". At the bottom of the window, a message box states "Inserted 1 rows successfully".

## 2.Update :

Update Lender

6022  
6024  
6025  
6026  
6027  
6028

Lender ID:

Name:

Contact no:

Money Lent:

Deadline:

6026

Ramesh

106

16000

02-mar-2020

Update Lender

6022  
6024  
6025  
6026  
6027  
6028

Lender ID:

Name:

Contact no:

Money Lent:

Deadline:

6026

Suresh

106

16000

02-mar-2020

Update Lender

Updated 1 rows successfully

Update Borrower

1031  
1032  
1033  
1034  
1035  
1036  
1038  
1039

Name:

Payment\_id:

Contact\_no:

Address:

Payment Status:

Profession:

michelle

1039

109

gachibowli

paid

teacher

Update Borrower

1031  
1032  
1033  
1034  
1035  
1036  
1038  
1039

Name:

Payment\_id:

Contact\_no:

Address:

Payment Status:

Profession:

michelle

1039

110

gachibowli

paid

teacher

Update Borrower

Updated 1 rows successfully



### 3.Delete :

The image displays four screenshots of a database application interface, arranged in a 2x2 grid, demonstrating the deletion of records.

**Top Left: Remove Lender (Initial State)**

- Lender List:** 6021, 6024, 6025, 6026, 6027, **6028** (selected), 6022
- Form Fields:**
  - Lender ID: 6028
  - Name: Ramesh
  - Contact no: 108
  - Money Lent: 15000
  - Deadline: 2020-03-03 00:00:00
- Action:** Delete Lender button

**Top Right: Remove Lender (Post-Deletion State)**

- Lender List:** 6021, 6024, 6025, 6026, 6027, 6022
- Form Fields:** All fields are empty.
- Message:** Deleted 1 rows successfully
- Action:** Delete Lender button

**Bottom Left: Remove Borrower (Initial State)**

- Borrower List:** 1033, **1039** (selected), 1031, 1032, 1034, 1035, 1036, 1038
- Form Fields:**
  - Lender ID: 1039
  - Name: michelle
  - Contact no: 110
  - Address: gachibowli
  - Payment\_status: paid
  - Profession: teacher
- Action:** Delete Borrower button

**Bottom Right: Remove Borrower (Post-Deletion State)**

- Borrower List:** 1033, 1031, 1032, 1034, 1035, 1036, 1038
- Form Fields:** All fields are empty.
- Message:** Deleted 1 rows successfully
- Action:** Delete Borrower button

## **DISCUSSION & FUTURE WORK :**

The application done till now is basically how to get back money from the borrower the lender gave and here it showed that it can be recovered from by seizing the property. There are numerous other ways in which we can recover the amount lent. Furthermore, other programming languages can also be used along with database by connecting SQL with it. Other actors can be included like guarantor and policy manager and other attributes can also be added to extend the application.

## **REFERENCES :**

<https://www.oracle.com/technetwork/java/javase/documentation/index.html>

<https://nptel.ac.in/courses/106105175/>

<https://google.github.io/styleguide/javaguide.html>

<https://nptel.ac.in/courses/106105191/>

