Malwares and its counterfeit

database

***DONE BY:-***

***A.RUCHITA***

***IT-B***

***1602-18-737-064***

**DONE BY:-**

**c.sravya**

**IT-B**

**1602-18-737-106**

**JAVA AWT BASED- MALWARES AND ITS COUNTERFEIT DATABASE- 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

**c.sravya<1602-18-737-106>**

Under the guidance of B.Leelavathy

****

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2019**

**BONAFIDE CERTIFICATE**

This to Certify that the project report titled “**Malwares and its counterfeit database**” project work of Ms. C.SRAVYA bearing Roll no: 1602-18-737-106 who carried out this project under my supervision in the IV Semester the academic year 2019-2020.

*Signature signature External Examine Internal examine*

**ABSTRACT**:-

This project is on “Malwares and its counterfeit database”.Malware(malicioussoftware)is any software intentionally designed to cause damage to a computer,server, client,or computer network. A wide variety of types of malware exist, including computer,viruses, worms, Trojan horses, ransomware, spyware, adware,and scareware.The malware is one of the major concerns in computer and cyber security. The availability of various malware toolkits and internet popularity that has led to the increase in number of malware attacks day to day. Comparing with existing framework of antivirus scanners they currently used signature based a malware detection technique which is widely. In this we get to know how to detect the malware and how to protect it using tools .Also different types of malwares and there detailed view is seen here

**AIM:**

To create a **Java GUI based Malwares and its counterfeit database** which takes the values like: malware licence key,malware created date,maware updated date,targated users,targated user id,attacker id,preventive measures,protection id etc from the user. These values are to be updated in the database using **JDBC connectivity.**

**INTRODUCTION**

**Requirements:**

**List of tables:**

* Malwares
* Attacks
* Targeted users
* Prevention measures
* Action

**List of attributes with their domain types:-**

*Malwares:*

Malware licence key : mlk varchar2(30)

Malware types :mtypes varchar2(20)

Malware created :mcreated varchar2(20)

Malware updated :mupdated varchar2(20)

*Targated user*:

Targated user id:tid number(20)

Targated user name :tname varchar2(20)

Targated by: targatedby varchar2(20)

*Attack:*

Malware license key :mlk varchar2(30)

Targated user name: tid number(20)

*preventive\_measures:*

Protected by: protected varchar2(30

Detected by:detected varchar2(30)

Action:

Malware license key: mlk varchar2(30)

**ARCHITECTURE AND TECHNOLOGY :**

**Softwareused:**

Java Eclipse, Oracle 11g Database, Java SE version 7,SQL\*Plus.

**Java AWT:**

**Java AWT** (Abstract Window Toolkit) is an APIto developGUI 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.

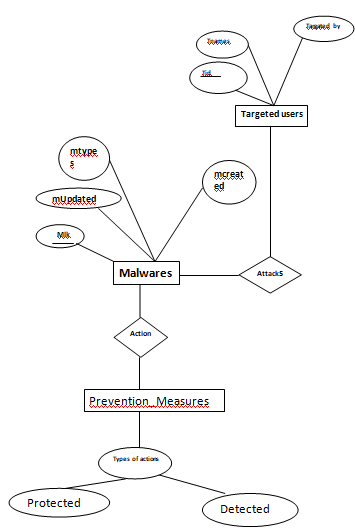
The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

**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**

**ER-Diagram**

****

**DDL COMMANDS**

SQL>create table **malwares**(

mlkvarchar2(30),primarykey,mtypes varchar2(20),mcreatedvarchar2(20),mupdated varchar2(20);

Table created.

SQL>create table **targated\_user**(

tid number(20) primary key, tname varchar2(20), targatedby varchar2(20));

Table created.

SQL>create table **attack**(

mlk varchar2(30), tid number(20),foreign key(mlk) references malwares,foreign key(tid) references targated\_user,primary key(mlk,tid));

Table created.

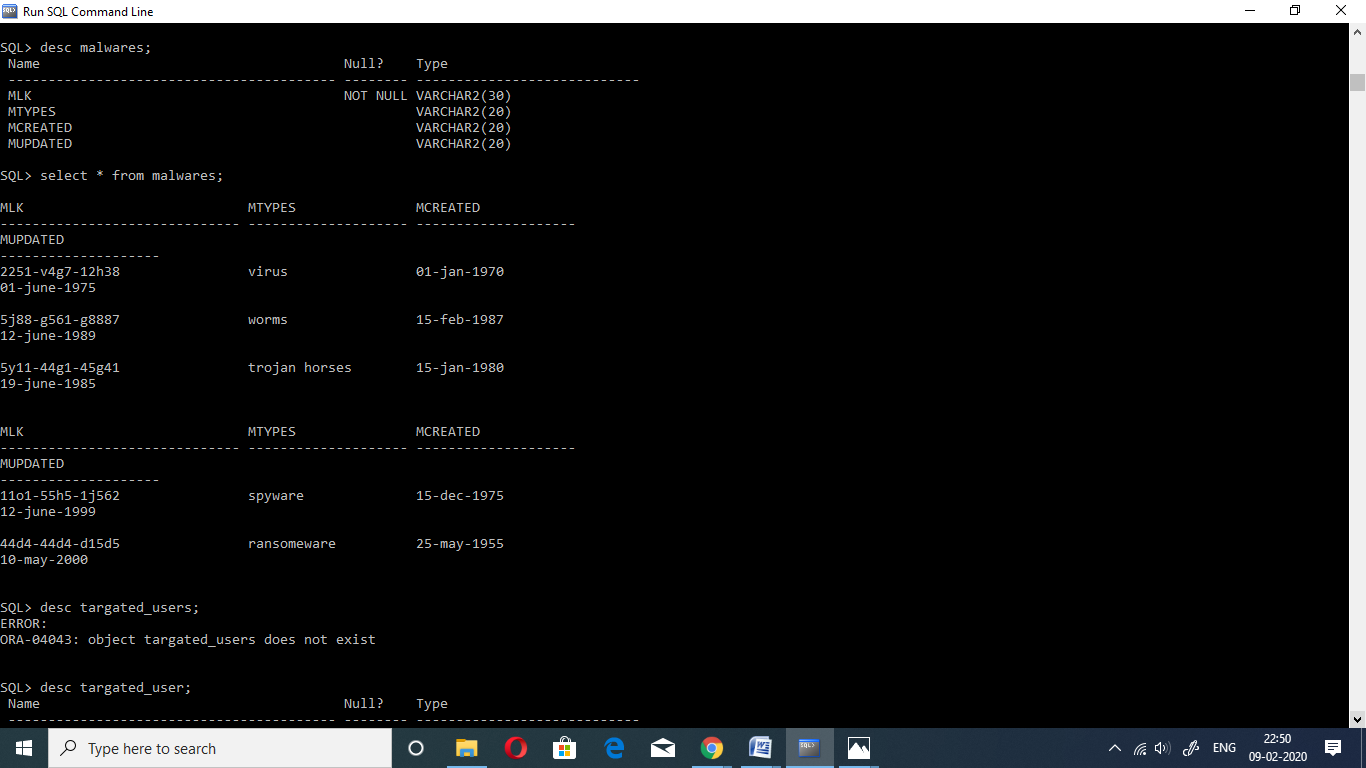
SQL>create table **preventive\_measures**(protected varchar2(30),detected varchar2(30));

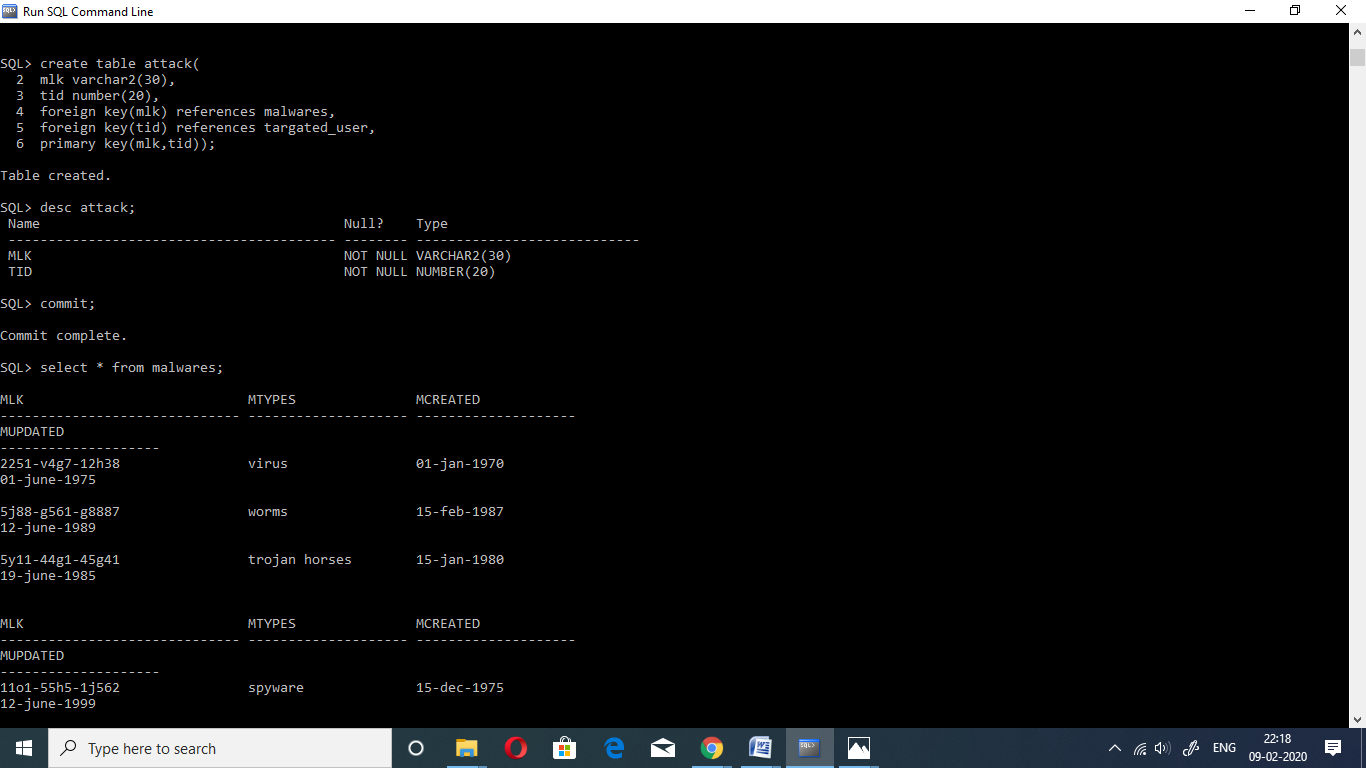
Table created.

SQL>create table **action**(

mlk varchar2(30),foreign key(mlk) references malwares, primary key(mlk));

Table created.





**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:

**privatevoid** connToDb(){

**try** {

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

connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:xe","sravya","vasavi");

statement = connection.createStatement();

} **catch** (SQLException connectException) {

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

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

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

System.*exit*(1);

}

**catch** (Exception e)

{

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

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.

**Table Created in SQL for above mentioned purpose is as:**

create table **malwares**(

mlk varchar2(30) primarykey,

mtypes varchar2(20),

mcreatedvarchar2(20),

mupdated varchar2(20)

);

**Program to insert malwares:**

package Malwares;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Insertmalwares extends Frame

{

private static final long serialVersionUID = 1L;

Button InsertmalwaresButton;

List malwaresLKList;

TextField MLKText, MTYPESText, MCREATEDText, MUPDATEDText;

TextArea errorText;

Connection connection;

Statement statement;

public Insertmalwares()

{

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","sravya","vasavi");

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()

{

InsertmalwaresButton = new Button("Insert malwares");

InsertmalwaresButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO malwares VALUES("+"'" + MLKText.getText() + "','" + MTYPESText.getText() + "','" + MCREATEDText.getText() + "','" + MUPDATEDText.getText() +"'"+ ")";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

MLKText = new TextField(15);

MTYPESText = new TextField(15);

MCREATEDText = new TextField(15);

MUPDATEDText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("MLK:"));

first.add(MLKText);

first.add(new Label("MTYPES:"));

first.add(MTYPESText);

first.add(new Label("MCREATED:"));

first.add(MCREATEDText);

first.add(new Label("MUPDATED:"));

first.add(MUPDATEDText);

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

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

second.add(InsertmalwaresButton);

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 malwares 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)

{

Insertmalwares sail = new Insertmalwares();

sail.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

sail.buildGUI();

}

}

**Program to update malwares:**

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Updatemalwares extends Frame

{

private static final long serialVersionUID = 1L;

Button updatemalwaresButton;

List malwaresLKList;

TextField MLKText, MTYPESText, MCREATEDText, MUPDATEDText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public Updatemalwares()

{

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","sravya","vasavi");

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 loadmalwares()

{

try

{

rs = statement.executeQuery("SELECT MLK FROM malwares");

while (rs.next())

{

malwaresLKList.add(rs.getString("MLK"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

malwaresLKList = new List(10);

loadmalwares();

add(malwaresLKList);

malwaresLKList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \*FROM malwares WHERE MLK ='"+malwaresLKList.getSelectedItem()+"'");

rs.next();

MLKText.setText(rs.getString("MLK"));

MTYPESText.setText(rs.getString("MTYPES"));

MCREATEDText.setText(rs.getString("MCREATED"));

MUPDATEDText.setText(rs.getString("MUPDATED"));

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

updatemalwaresButton = new Button("Update malwares");

updatemalwaresButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE malwares"

+ " SET MTYPES='" + MTYPESText.getText() + "', "

+ "MCREATED='" + MCREATEDText.getText() + "', "

+ "MUPDATED ='"+ MUPDATEDText.getText() + "' WHERE MLK = '"

+ malwaresLKList.getSelectedItem()+"'");

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

malwaresLKList.removeAll();

loadmalwares();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

MLKText = new TextField(15);

MLKText.setEditable(false);

MTYPESText = new TextField(15);

MCREATEDText = new TextField(15);

MUPDATEDText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("MLK:"));

first.add(MLKText);

first.add(new Label("MTYPES:"));

first.add(MTYPESText);

first.add(new Label("MCREATED:"));

first.add(MCREATEDText);

first.add(new Label("MUPDATED:"));

first.add(MUPDATEDText);

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

second.add(updatemalwaresButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setTitle("Update malwares");

setSize(500, 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)

{

Updatemalwares ups = new Updatemalwares();

ups.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

ups.buildGUI();

}

}

**Program to deletemalwares:**

package Malwares;

package last;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Deletemalwares extends Frame

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button deletemalwaresButton;

List malwaresLKList;

TextField MLKText, MTYPESText, MCREATEDText, MUPDATEDText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public Deletemalwares()

{

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","sravya","vasavi");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

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

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

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

System.exit(1);

}

}

// @SuppressWarnings("unused")

private void loadmalwares()

{

try

{

rs = statement.executeQuery("select \*from malwares ");

while (rs.next())

{

malwaresLKList.add(rs.getString("MLK"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

malwaresLKList = new List(10);

loadmalwares();

add(malwaresLKList);

malwaresLKList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("select \*from malwares");

while (rs.next())

{

if (rs.getString("MLK").equals(malwaresLKList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

MLKText.setText(rs.getString("MLK"));

MTYPESText.setText(rs.getString("MTYPES"));

MCREATEDText.setText(rs.getString("MCREATED"));

MUPDATEDText.setText(rs.getString("MUPDATED"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deletemalwaresButton = new Button("Delete malwares");

deletemalwaresButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM malwares WHERE MLK = '" + malwaresLKList.getSelectedItem()+"'and MTYPES= '"+ MTYPESText.getText()+"'and MUPDATED= '" + MUPDATEDText.getText()+"'and MCREATED= '"+ MCREATEDText.getText()+"'");

@SuppressWarnings("unused")

Statement statement1 = connection.createStatement();

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

MLKText.setText(null);

MTYPESText.setText(null);

MCREATEDText.setText(null);

MUPDATEDText.setText(null);

malwaresLKList.removeAll();

loadmalwares();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

MLKText = new TextField(15);

MTYPESText = new TextField(15);

MCREATEDText = new TextField(15);

MUPDATEDText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("MLK:"));

first.add(MLKText);

first.add(new Label("MTYPES:"));

first.add(MTYPESText);

first.add(new Label("MCREATED:"));

first.add(MCREATEDText);

first.add(new Label("MUPDATED:"));

first.add(MUPDATEDText);

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

second.add(deletemalwaresButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setTitle("Remove malwares");

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)

{

Deletemalwares dels = new Deletemalwares();

dels.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

dels.buildGUI();

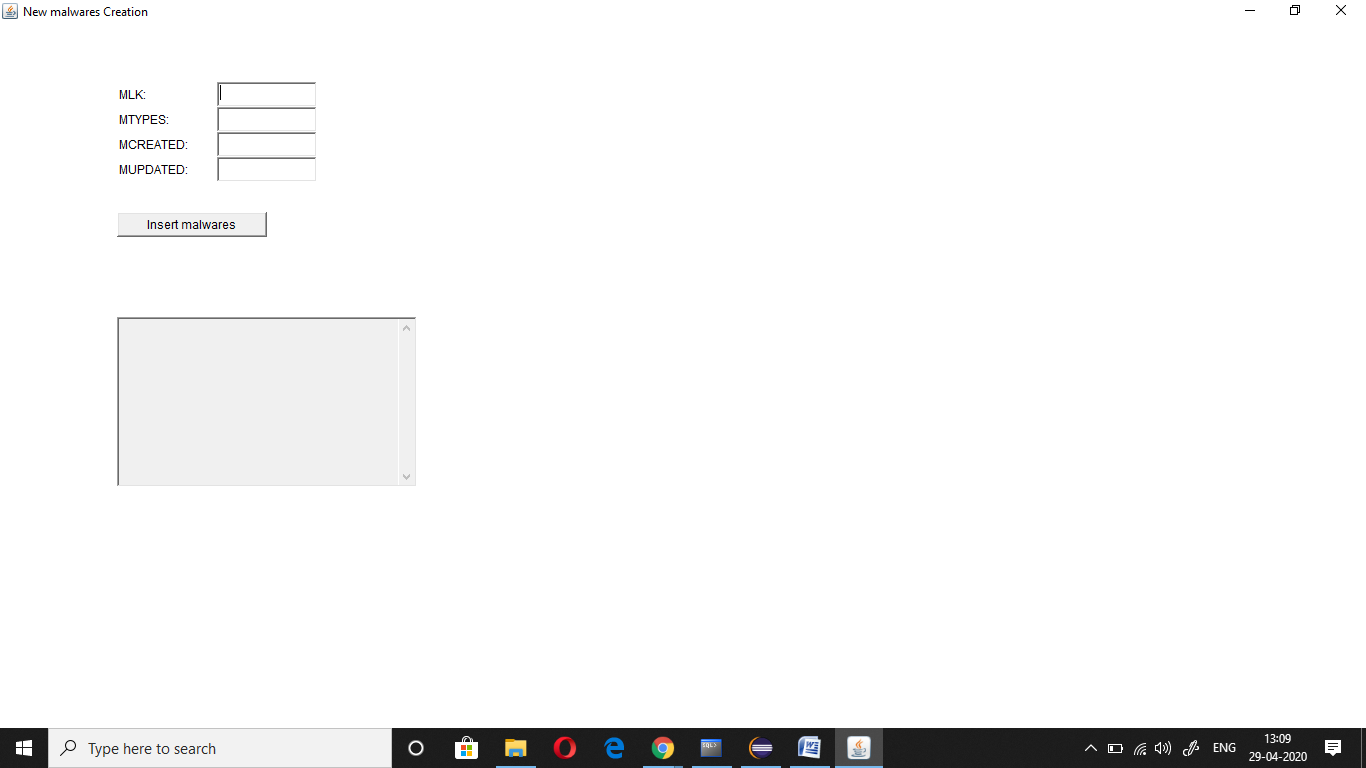
}

}

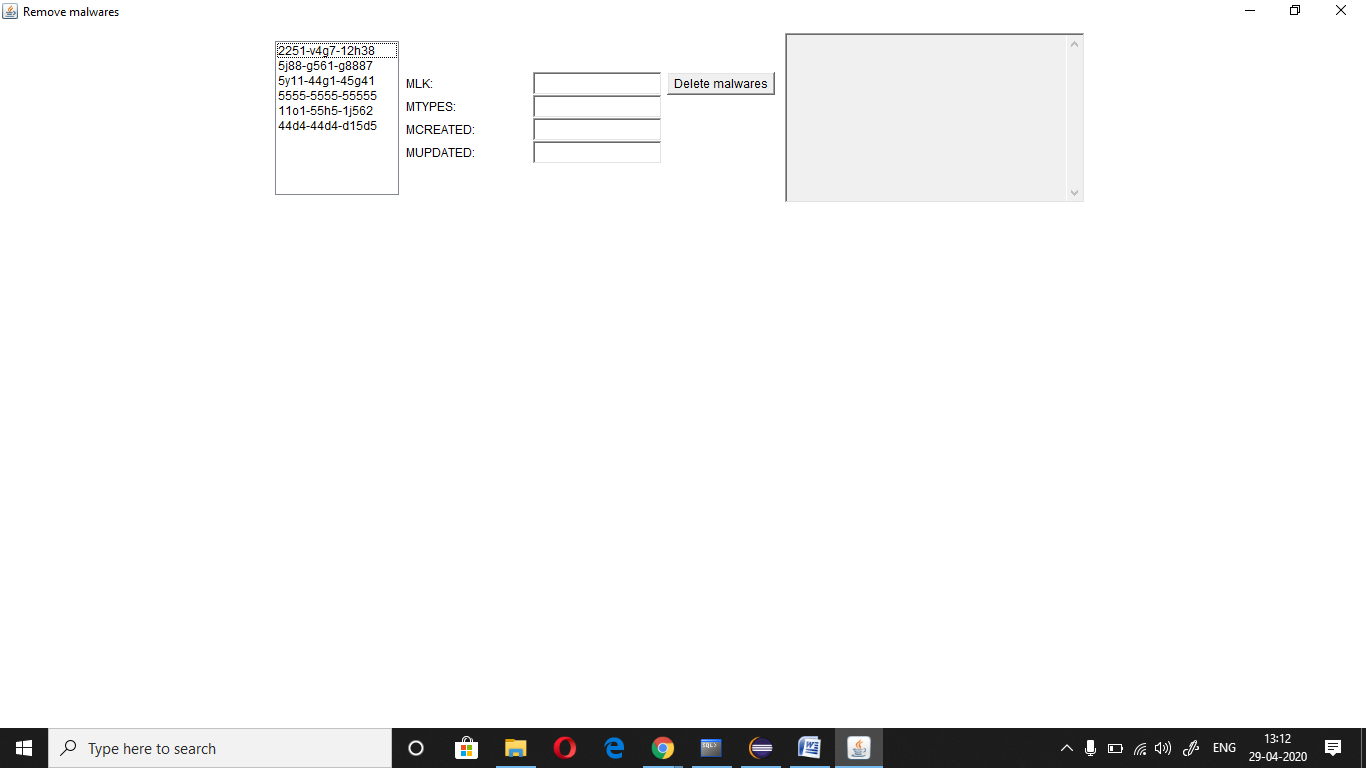
**OUTPUT SCREENSHOTS:**

**Java GUI Screenshot:**

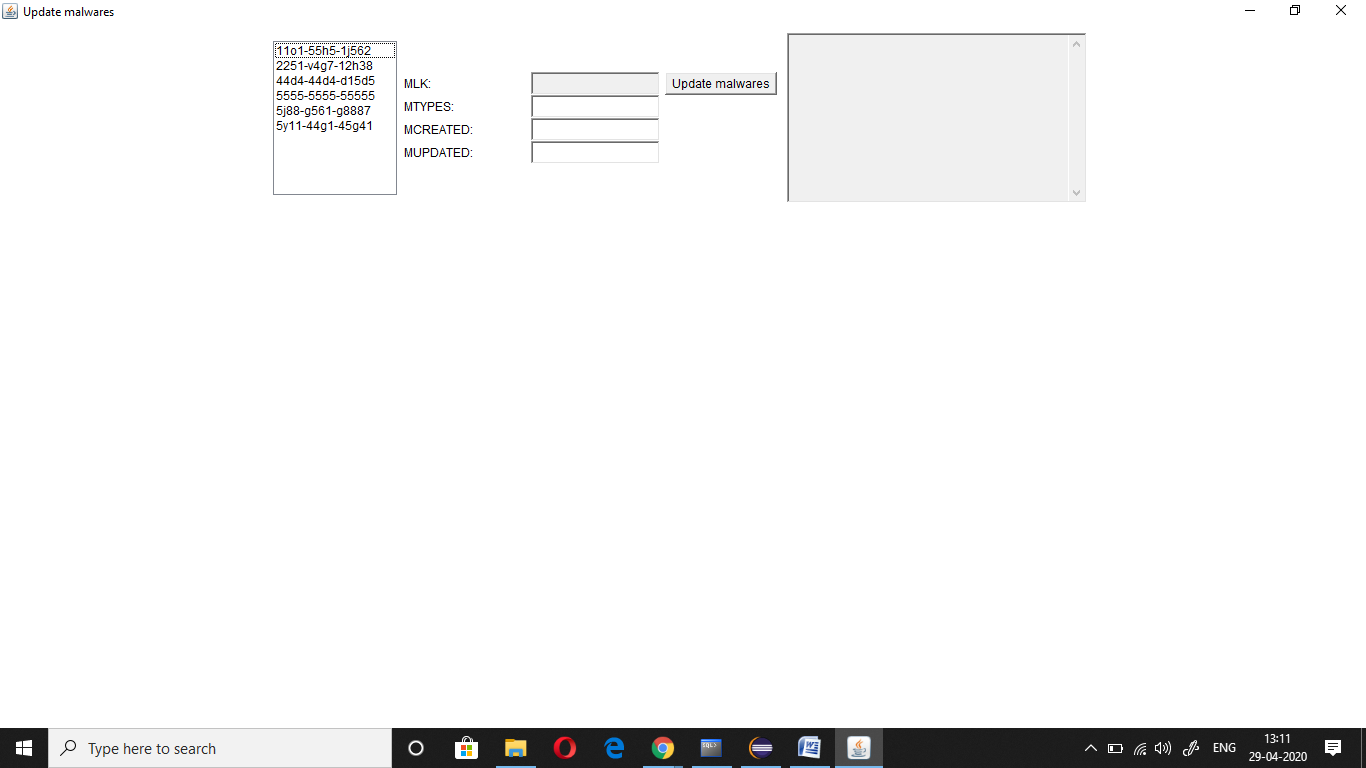
1) For inserting malwares



2) for deleting malwares



3) for updating malwares



**GITHUB Link:**

**https://github.com/sravya1206/Malwares-and-its-counterfeit-database/commits?author=sravya1206**

Github link

# Malwares and its counterfeit.pdf

DbmsReport Assignment-2

Assignment-2

Metadata

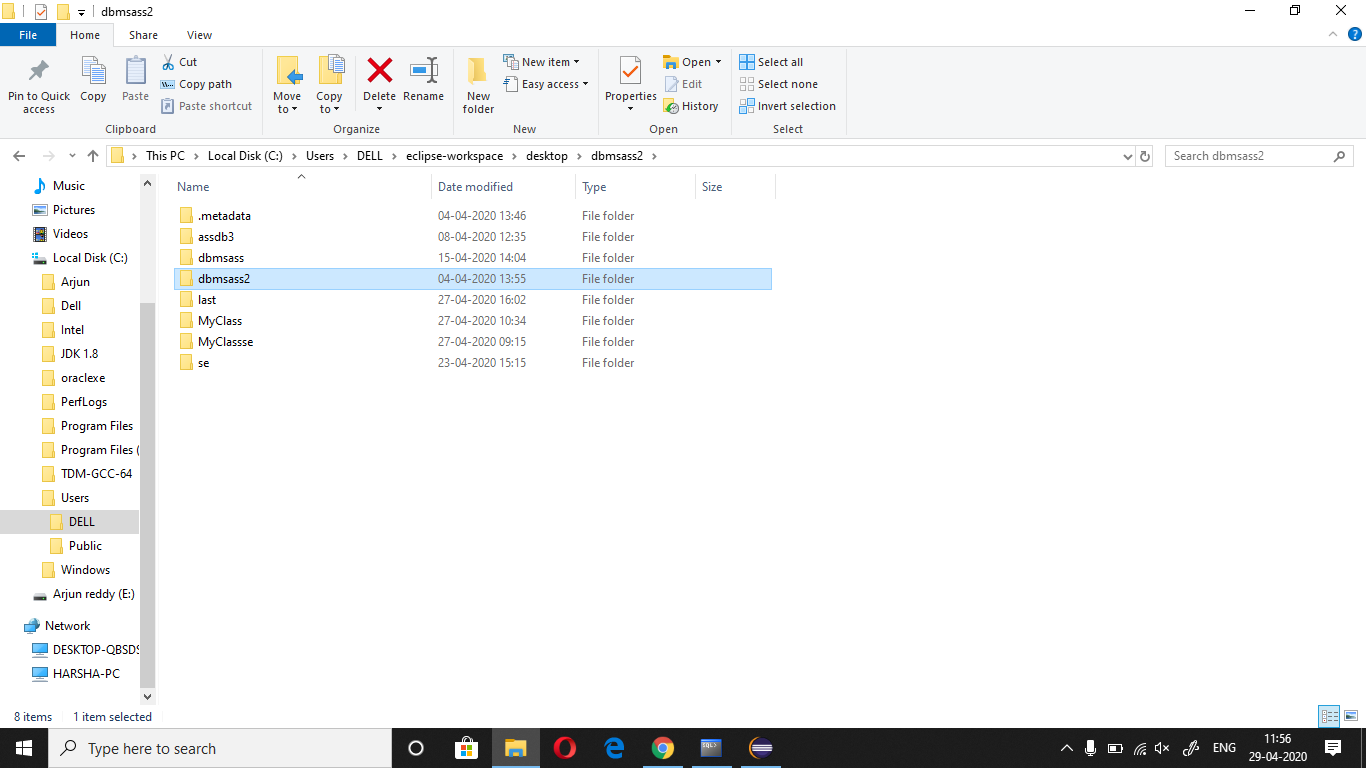
ASS-2 settings

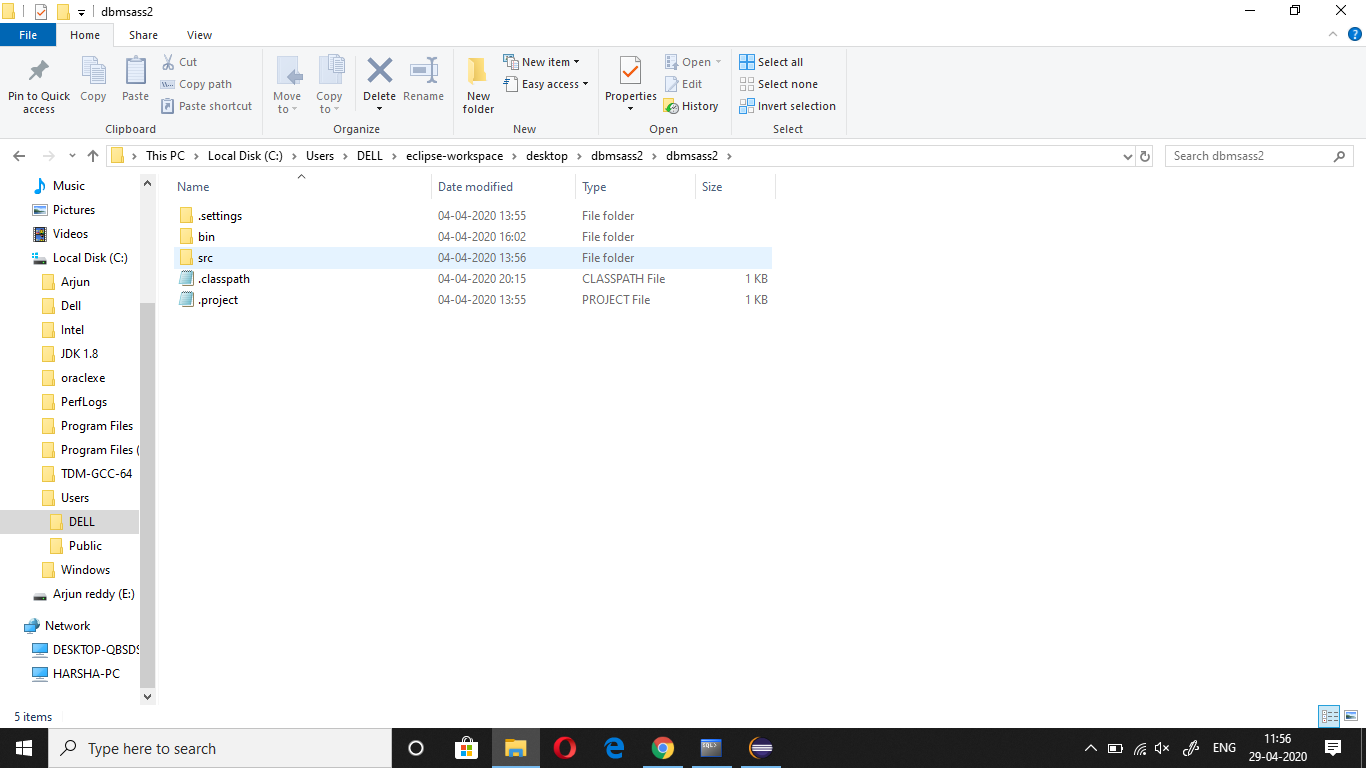
bin

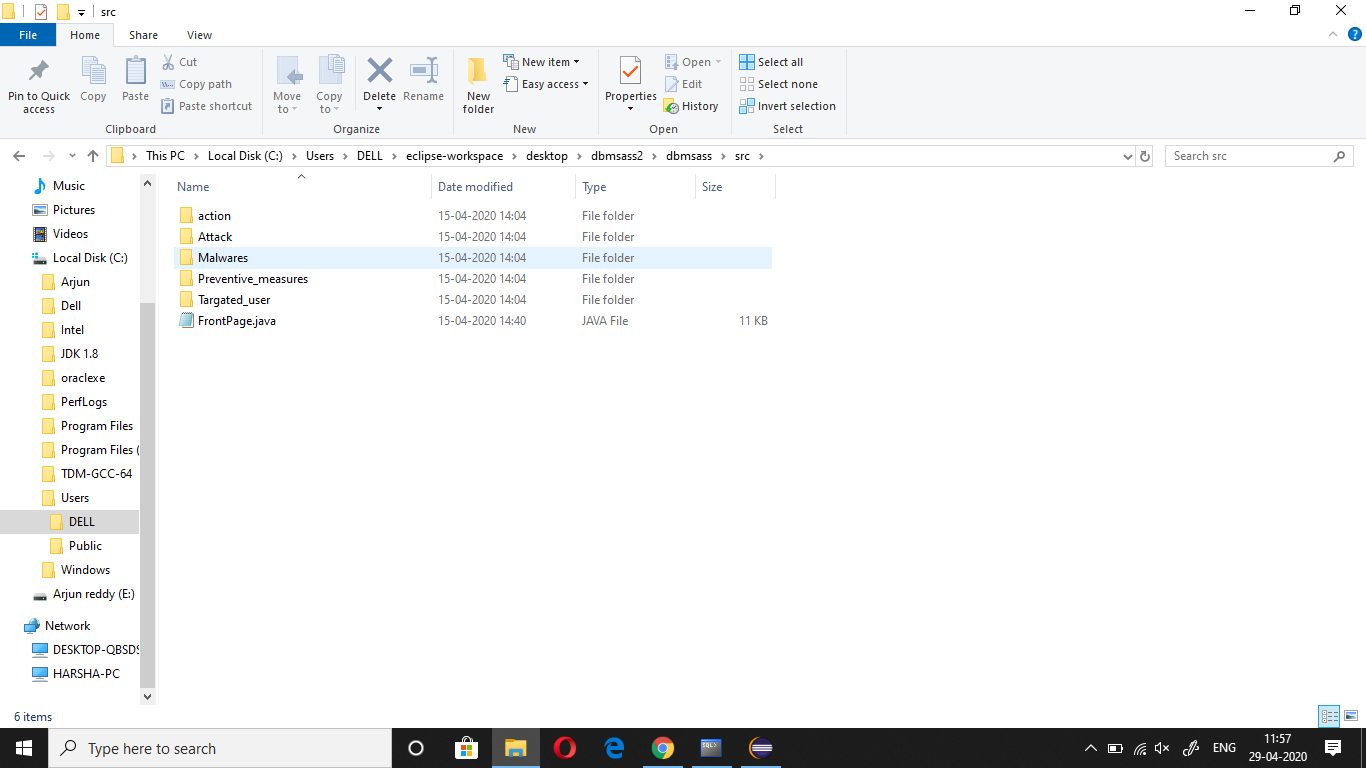
src programs of malwares and its counterfeit database

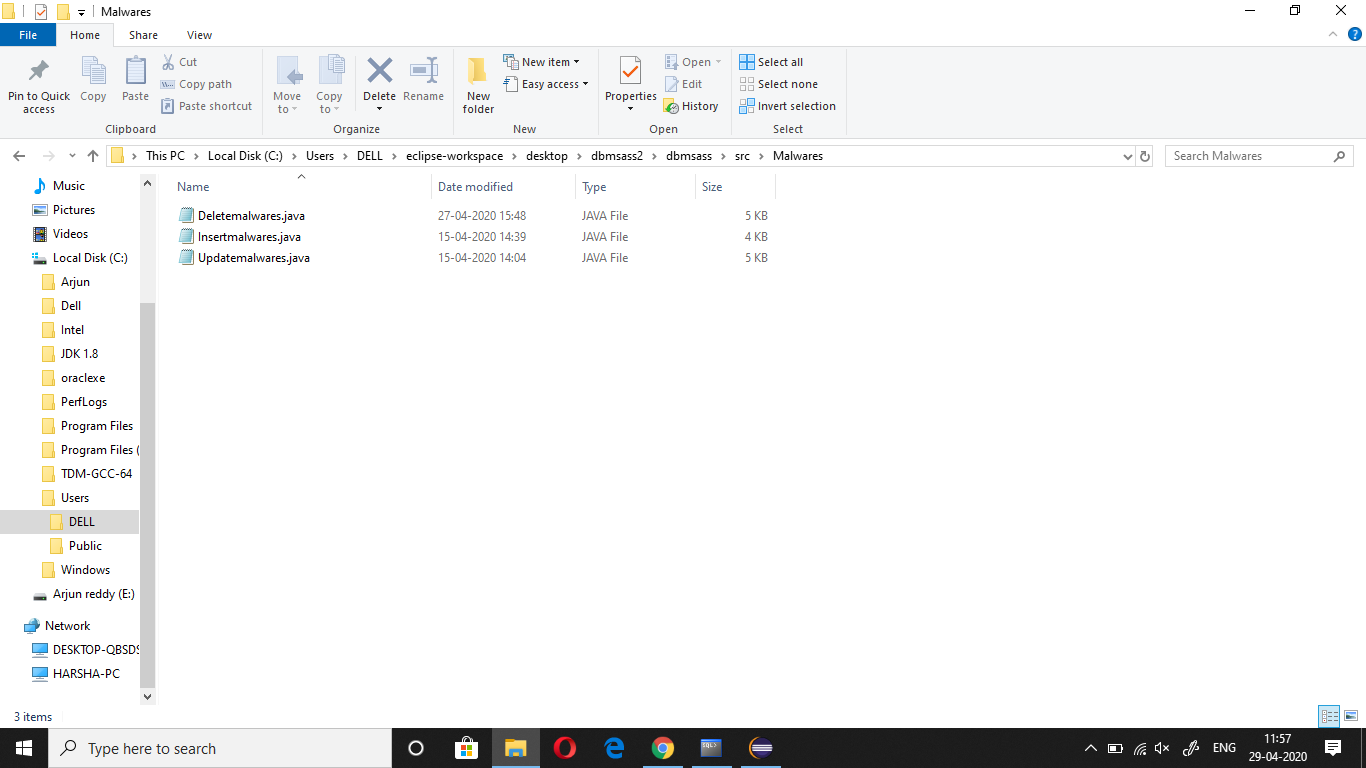
classpath

project

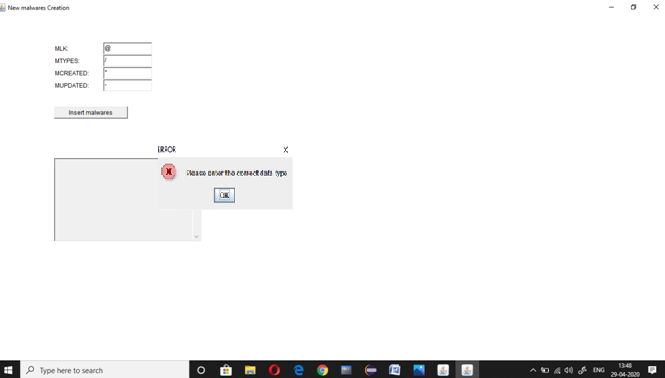








**TESTING**



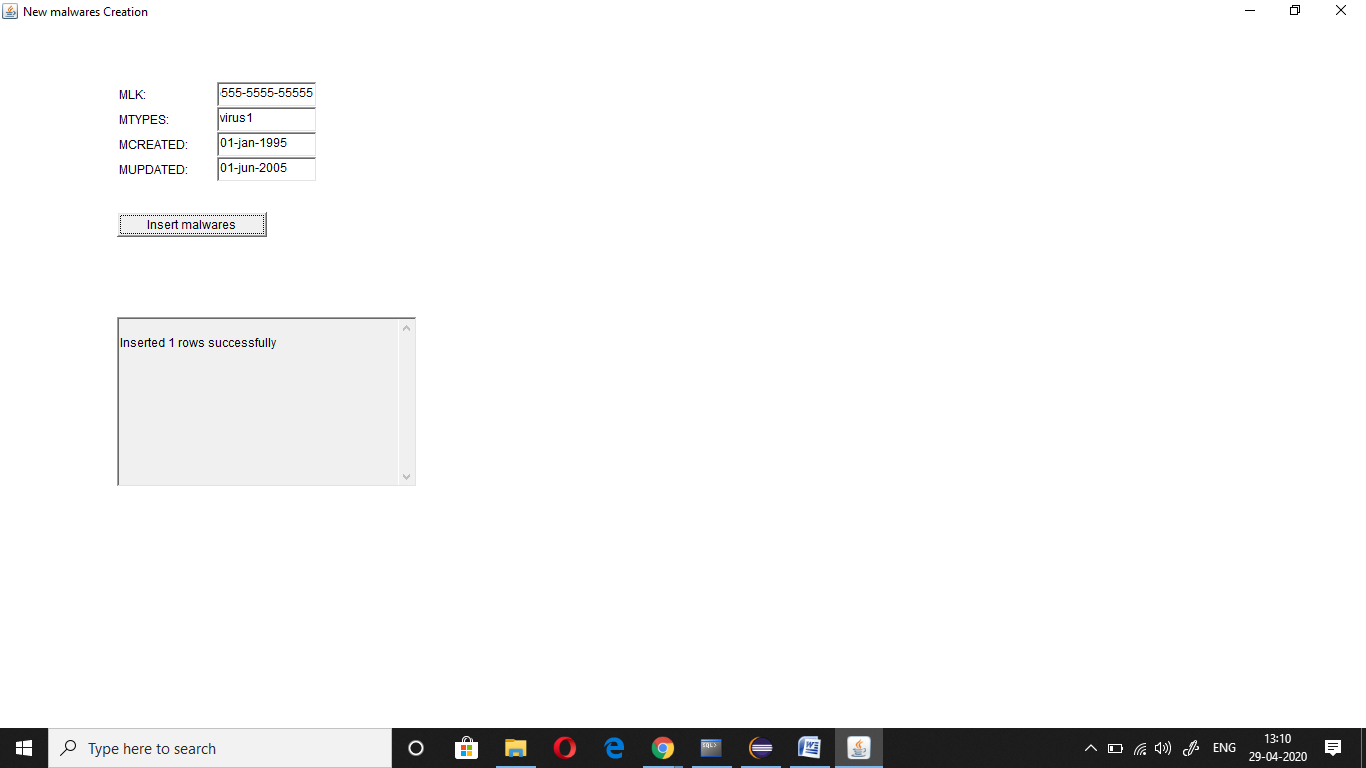
**DML COMMANDS**

1. Insert into malwares values(‘&mlk’,’&mtypes’,’&mcreated’,‘&mupdated’);
2. Insert into Targated\_user values ( ‘&tid ’ , ’ & tname ’ , ’ &targatedby ’);
3. Insert into Action values(‘&mlk’);
4. Insert into Attack values(‘&mlk’,’&tid’);
5. Insert into Prevenventive\_measures values(‘&Protectedby’, ‘&Detectedby’);

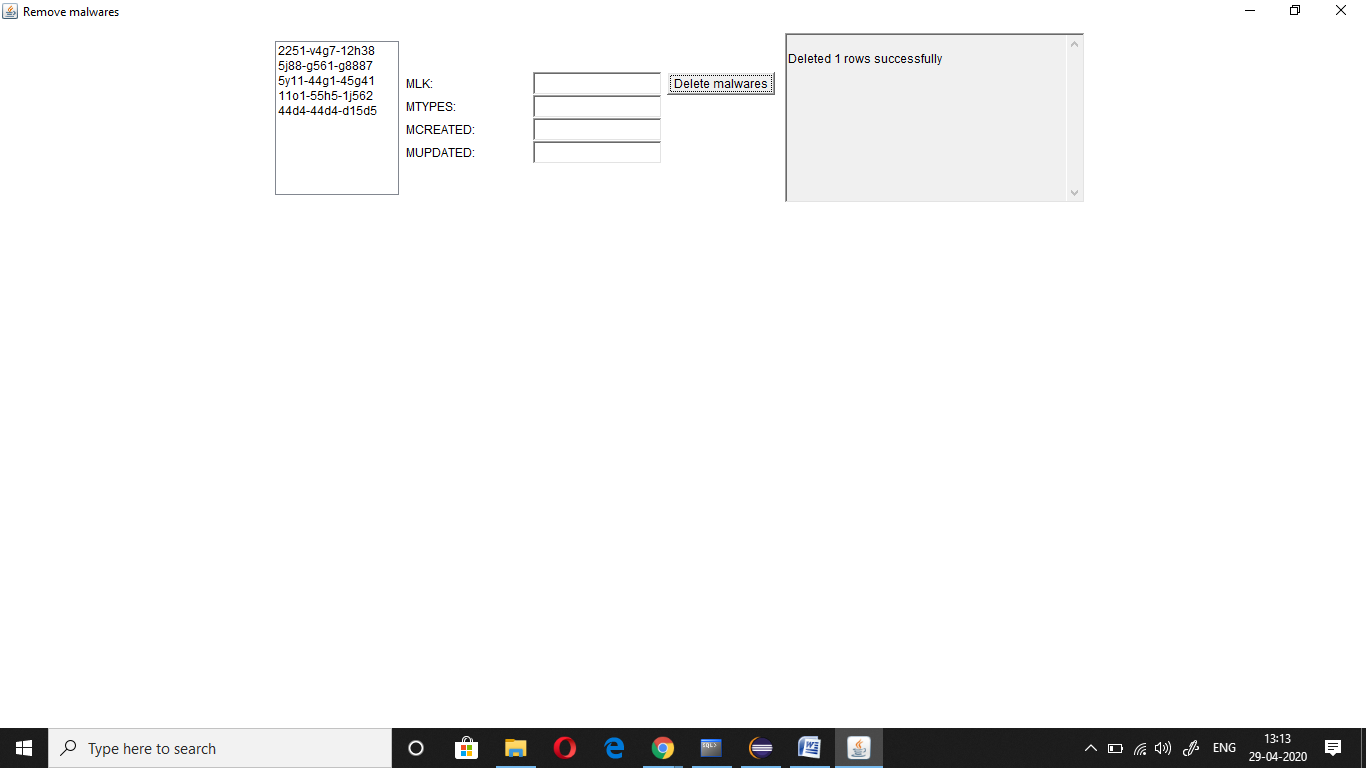
**OUTPUT SCREENSHOTS:**

**Java GUI Screenshot:**

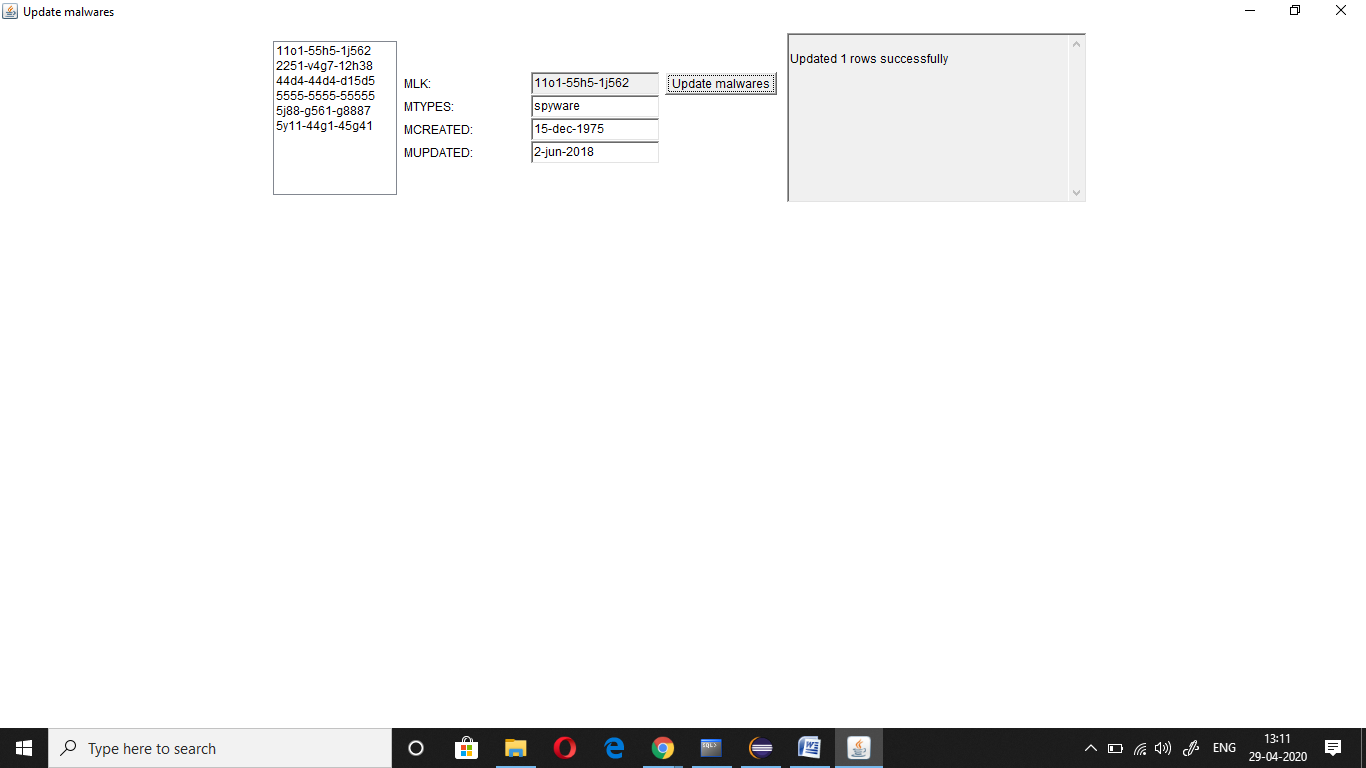
1) For inserting malwares



2) for deleting malwares

****

3) for updating malwares

****

**DISCUSSION & FUTURE WORK :**

The application done till now is to take all the information related to the malwares and its counterfeit . Furthermore, other programming languages can also be used along with database by connecting SQL with it. This application can be extended further more to know better about malwares and its counterfeit and get rid of malwares.

**CONCLUSION:**

Thus, a Java AWT based malwares and its counterfeit is created which is connected to the Oracle 11g database. Therefore, all the entries in the form are directly updated on the network table created in the database.

**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/>