**Phone Book Using**

**Swing GUI, ArrayList and RMI**

**Software Design Assignment-1**

**BE in Software Engineering**

**(4th year) Level-8**

**Submitted By:- Deepansh Dubey**

**Student Id- A00268737**

**Athlone Institute Of Technology**

**Ireland**

November 13, 2019

**Final Report On Phone Book Application**

**Title of the Project:-** Phone Book Application

**Objective:-**

The main objective of this system is to store the contact details of the desired persons in a file format. It is a friendly easy to use interface developed in Java with to store the details. This application stores all the details like Name, Surname, Phone Number. This system was developed to reduce the errors that creep up in manual systems. It was very difficult to store all the details manually there such a system was developed. This software also allows to edit, update and search various contact details. It is secure, easy to use and reliable software system. It also provides a good level of security as there is an admin who can only edit and update details.

The aim is to record the details/contact of user. It will simplify the task and reduce the paper work. The system is very user friendly and it is anticipated that applicants will easily access the functions of the system. This will help user to easily search and manage contacts using this system.

**Existing system:-**

Earlier there was a manual system which requires a lot of paperwork and required a lot of time to maintain the contact records . The records stored were also not secured as anyone can view it. So there was a risk of storing records. Besides this, if any changes were to be made then all things need to be updated which was very difficult to do. There was also a risk for loss of information in this system. This also lead to errors so there was a need for the new system to be developed.

**Proposed system:-**

The telephone directory system allows storing contact details in a file which is handled by the user. So this provides a secure system which could easily insert, search, edit or update any records. It also allows to easily update the contact details without modifying much. The records can be modified only by the user. In this, you can easily view the contact details. This reduces dependence on the manual system which earlier required a lot of paperwork. This also saves time and the cost of paper. All the records are safely stored in a file.

**Classes:-**

* **Contact.Java:-**

This class contain the Java code for Phone bookand havingdifferent functionality for the buttonslike

* **Add –** To add the contact in a contact list.
* **Delete –** To delete the particular contact from contactlist.
* **Save -** To save the contact in a contact list as a file format.
* **Search Bar -** To search the contact from the contact list.
* **Load Button –** To load the file to open.
* **Server.java:-**

Server.java file will be having the GUI and action event to call the classes.

* **Client.Java:-**

To use the system as RMI there has to be a client.java file to use system to on their machine.

* **Implement.java:-**

Implement.java file will be having all the calculation in order to find the answer of each functionality.

* **Interface.java:-**

It is very similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

**Modules:-**

* **Contact module**: This module allows the user to enter all the details like name, phone number, address, website, facebook account details.

* **Insert module:** In this module, all the details regarding contacts are inserted.

* **Delete module:** In this module, the records with a specific id is deleted.

* **Save module:** This module allows the user to view a specific contact detail.

* **Search module:** This module allows to search for the contact details by just entering its name.

**Functionality Used:-**

* Java
* Array List
* RMI
* Swing GUI

**Key Parts of Code :-**

**Array list code -**

Object[] columns = {"FirstName", "LastName", "Phone Number"}; // Arraylist of headers names

DefaultTableModel model = **new** DefaultTableModel();

model.setColumnIdentifiers(columns);// headers

table.setModel(model);

TableRowSorter<TableModel> tr = **new** TableRowSorter<TableModel>(model);

table.setRowSorter(tr);

**Getting the text from textfields-**

**public** **void** actionPerformed(ActionEvent actionEvent) {

//getting the text from text fields and add them into the table

row[0] = textName.getText();

row[1] = textLastName.getText();

row[2] = textPhNumber.getText();

model.addRow(row);

}

});

**Upadate the list-**

**public** **void** mouseClicked(MouseEvent e) {

//double click selected row and update it

**int** i = table.getSelectedRow();

textName.setText(model.getValueAt(i,0).toString());

textLastName.setText(model.getValueAt(i,1).toString());

textPhNumber.setText(model.getValueAt(i,2).toString());

}

});

**Function to save to the file-**

// function to save file

**public** **void** actionPerformed(ActionEvent actionEvent) {

**try** {

JFileChooser jF = **new** JFileChooser(**new** File("D://"));

jF.setDialogTitle("Save a file");

**int** result = jF.showSaveDialog(**null**);

**if** (result == JFileChooser.***APPROVE\_OPTION***)

System.***out***.println("File: "+ jF.getSelectedFile());

FileOutputStream out;

PrintStream pS;

out = **new** FileOutputStream(jF.getSelectedFile());

pS = **new** PrintStream(out);

// Print to file

**for** (**int** i = 0;i < table.getRowCount();i ++) {

**for** (**int** j = 0; j < table.getColumnCount(); j++) {

pS.print(table.getValueAt(i, j).toString()+" ");

}

pS.println(" ");

}

pS.close();

JOptionPane.*showMessageDialog*(**null**, " File Saved!");

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

}

});

**Load the file-**

//load the file

**public** **void** actionPerformed(ActionEvent actionEvent) {

JFileChooser fc = **new** JFileChooser(**new** File( "C://"));

**int** returnVal = fc.showOpenDialog(**null**);

**if**(returnVal == JFileChooser.***APPROVE\_OPTION***)

System.***out***.println("File: " + fc.getSelectedFile());

**try**{

FileReader fr = **new** FileReader(fc.getSelectedFile());

BufferedReader in = **new** BufferedReader(fr);

Object[] inLine = in.lines().toArray();

**for** (**int** i = 0; i < inLine.length; i++){

String[] row1 = inLine[i].toString().split(" ");

model.addRow(row1);

}

in.close();

}**catch**(Exception e){

System.***out***.println(e);

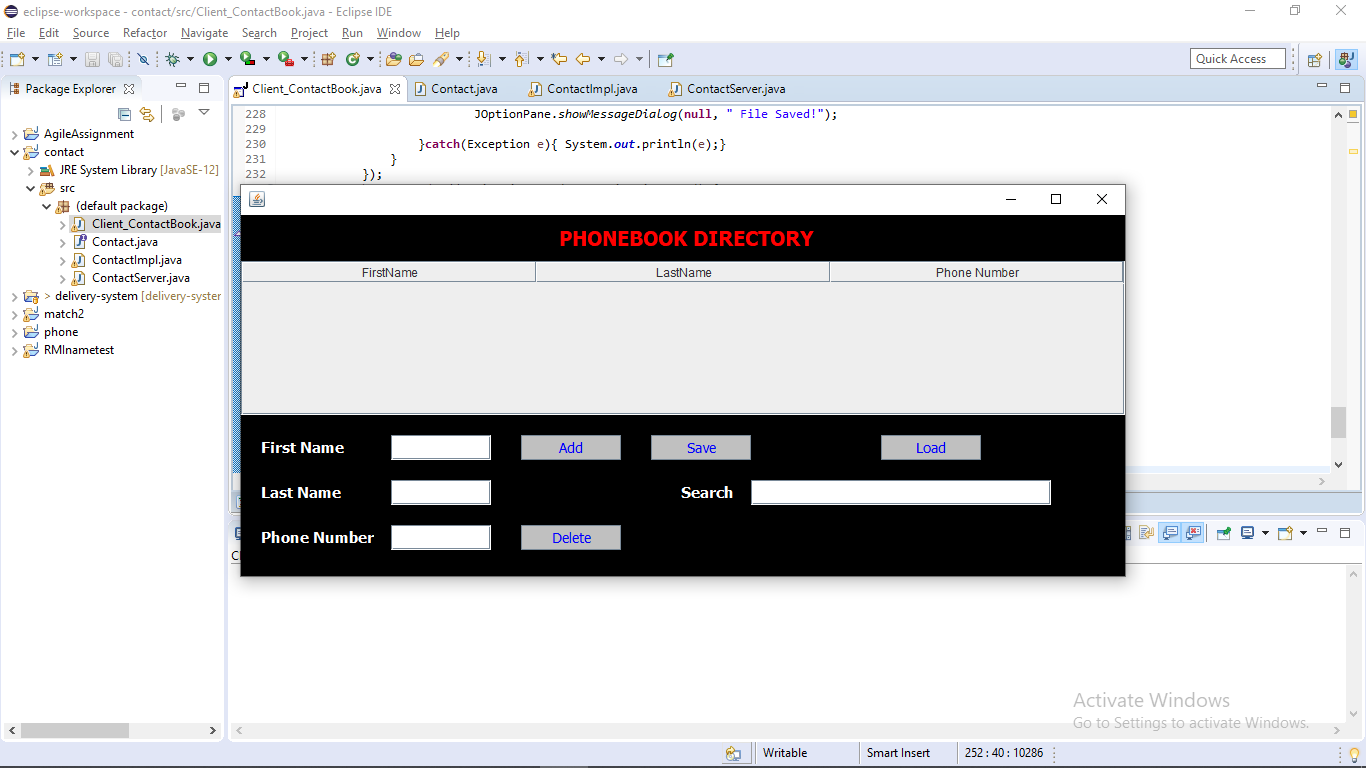
}

}

});

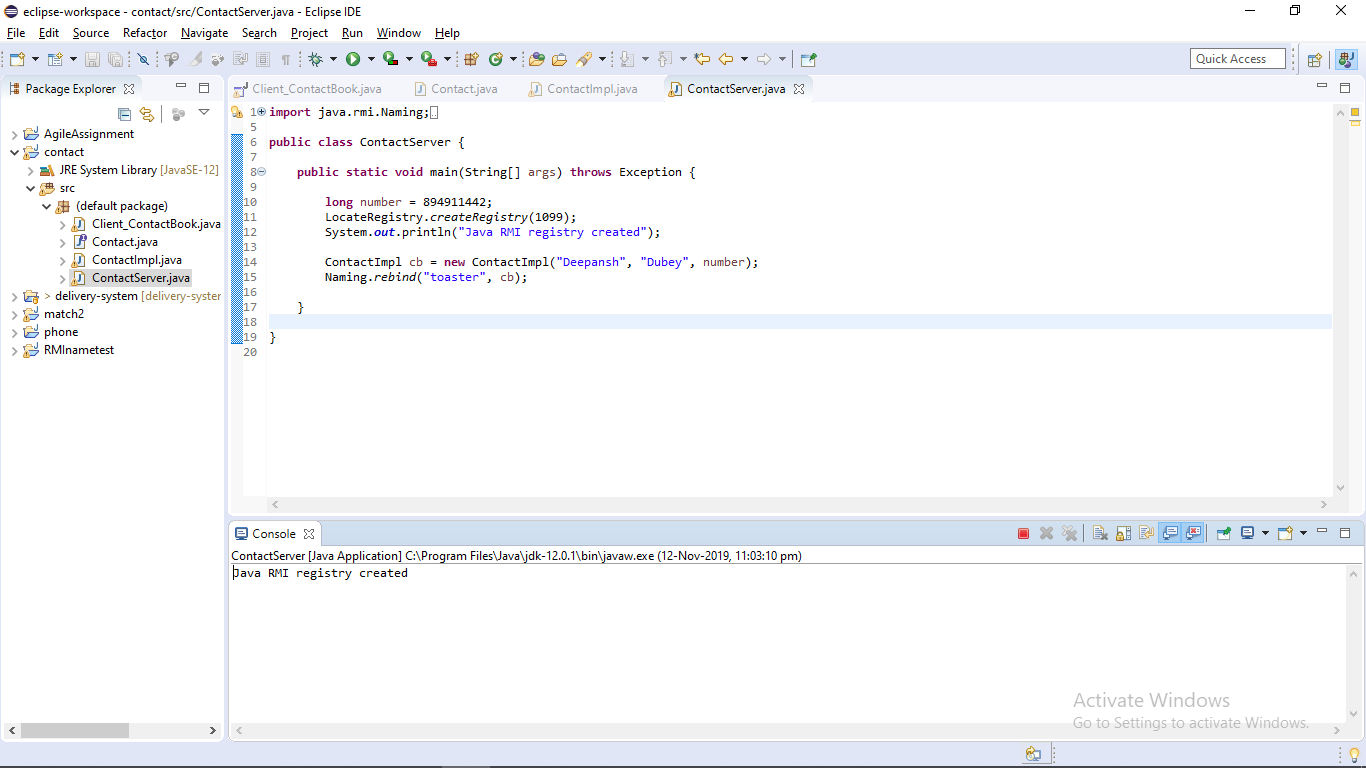
**Screenshots:-**

1. Phonebook Directory:-

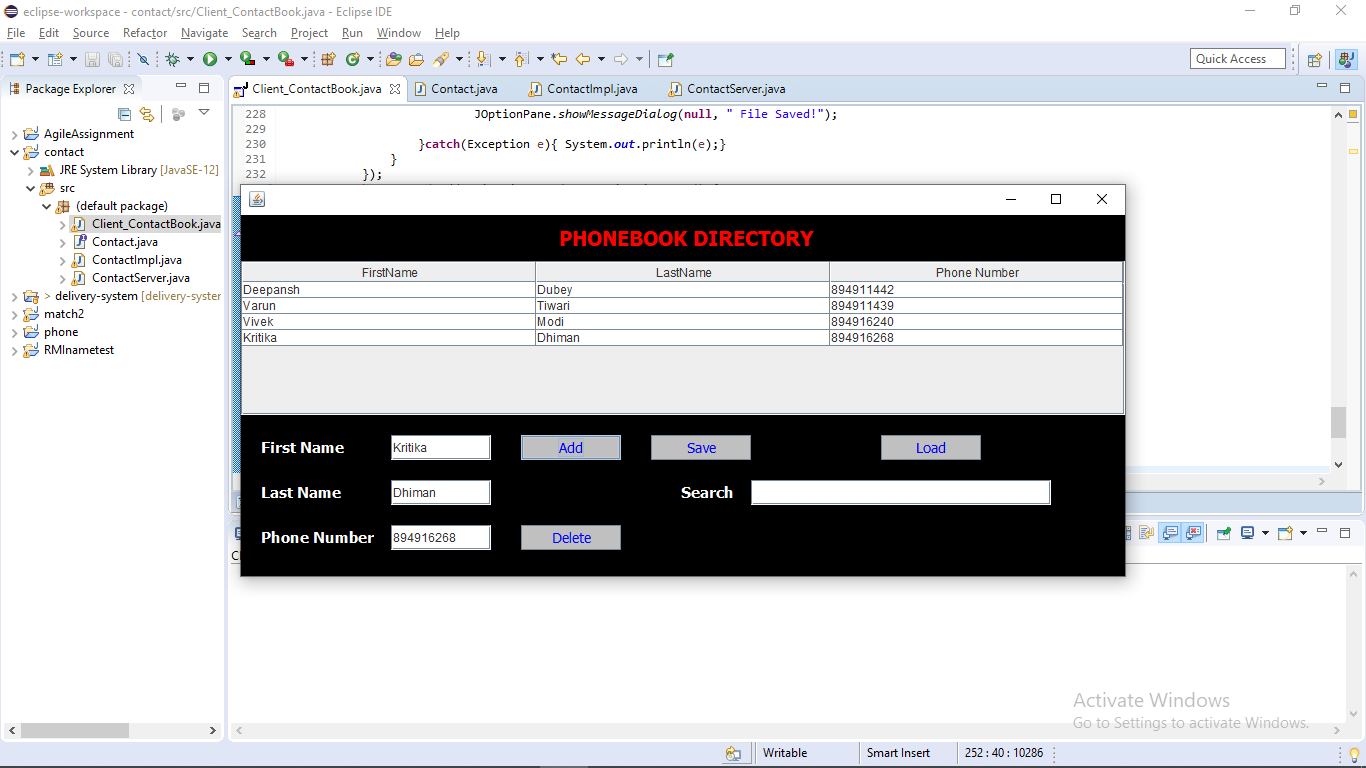


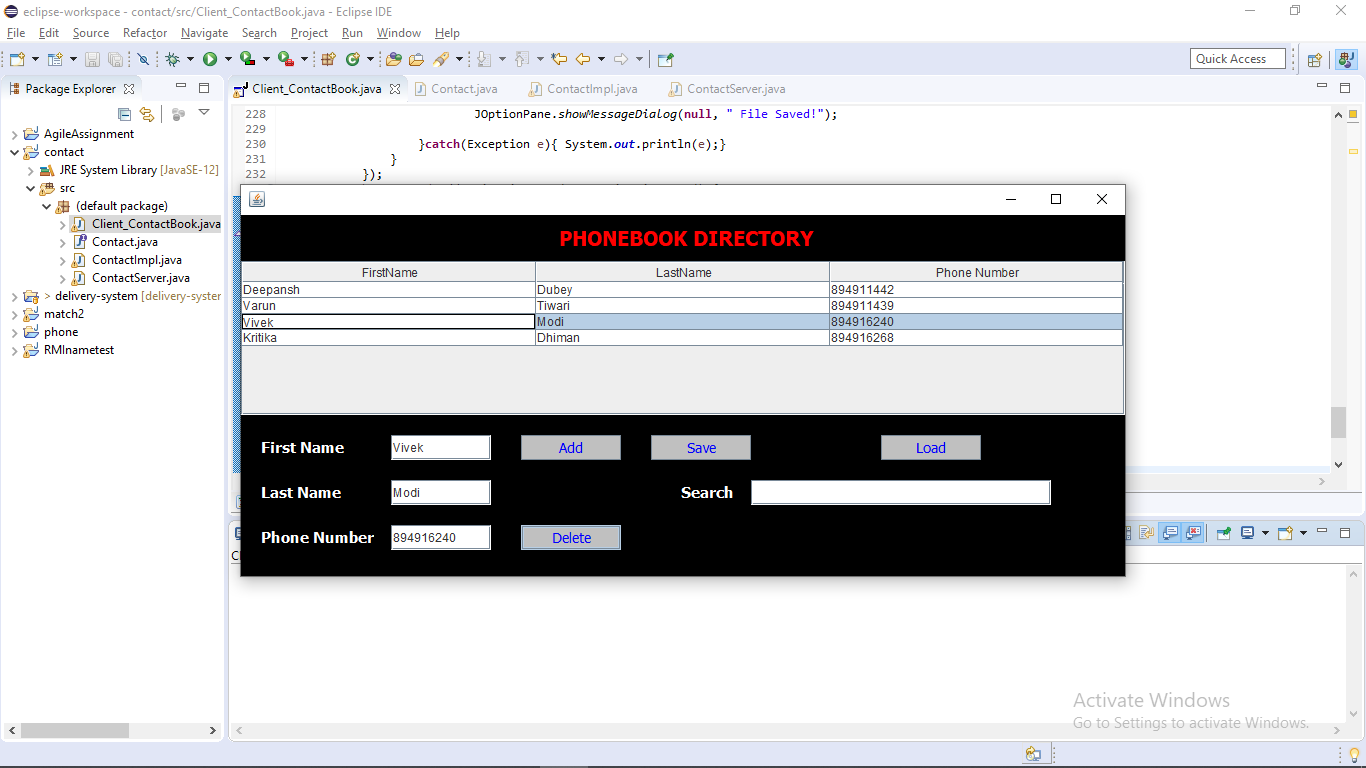
­­­

2. RMI :-

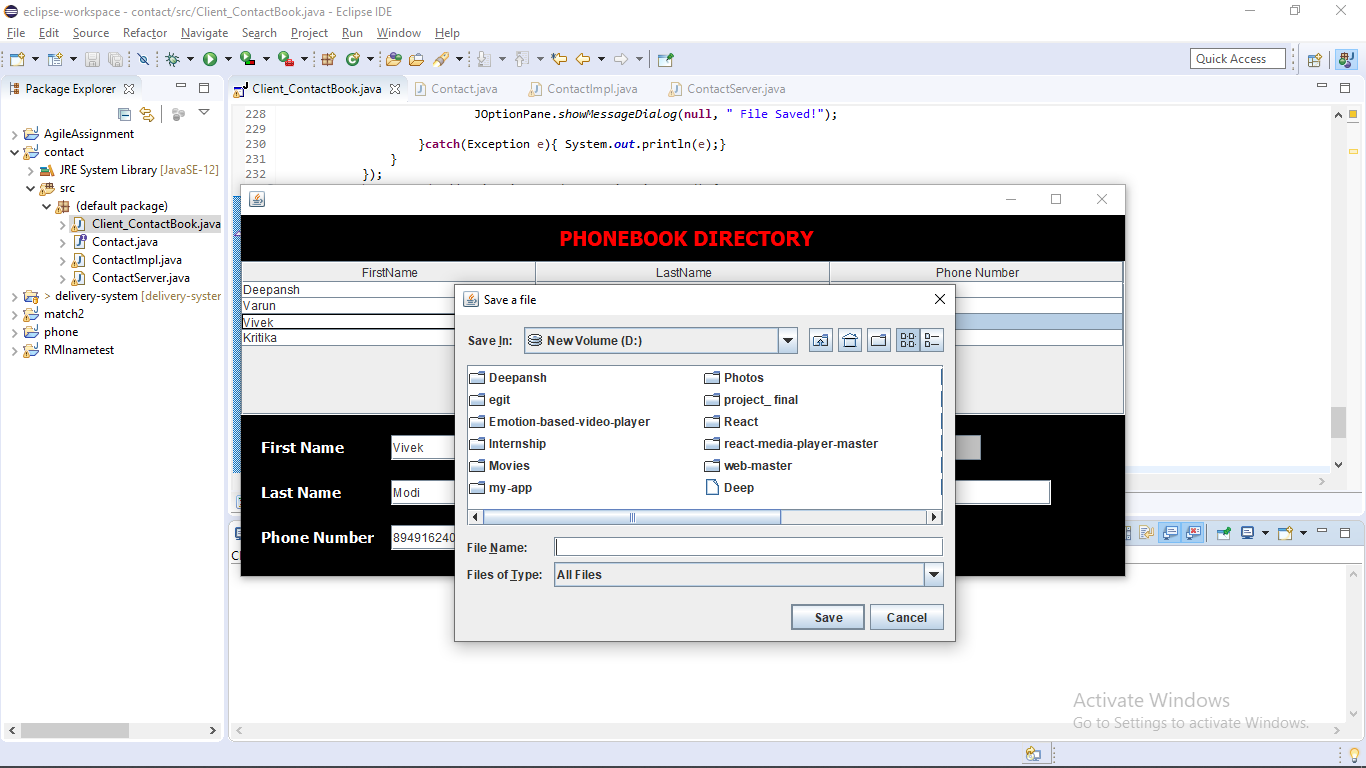


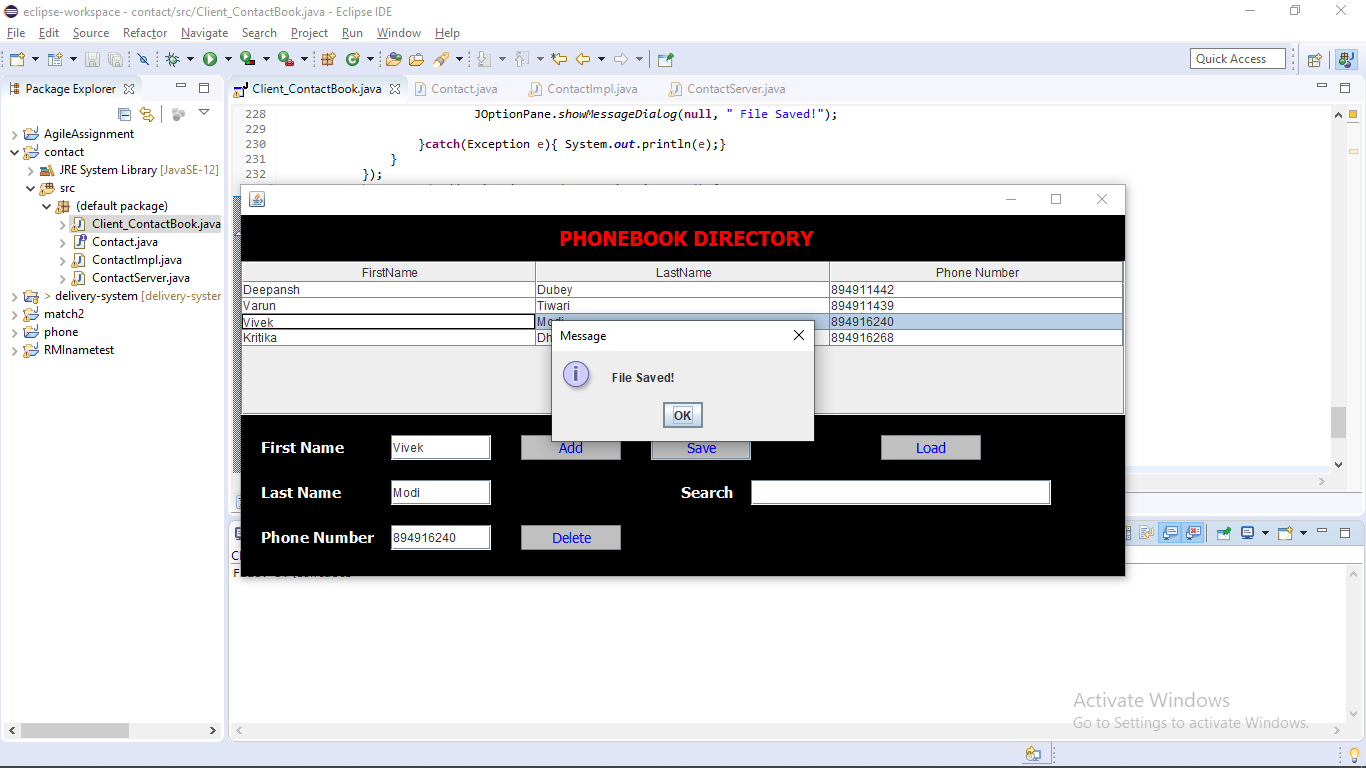
3. Adding the information into phonebook directory:-



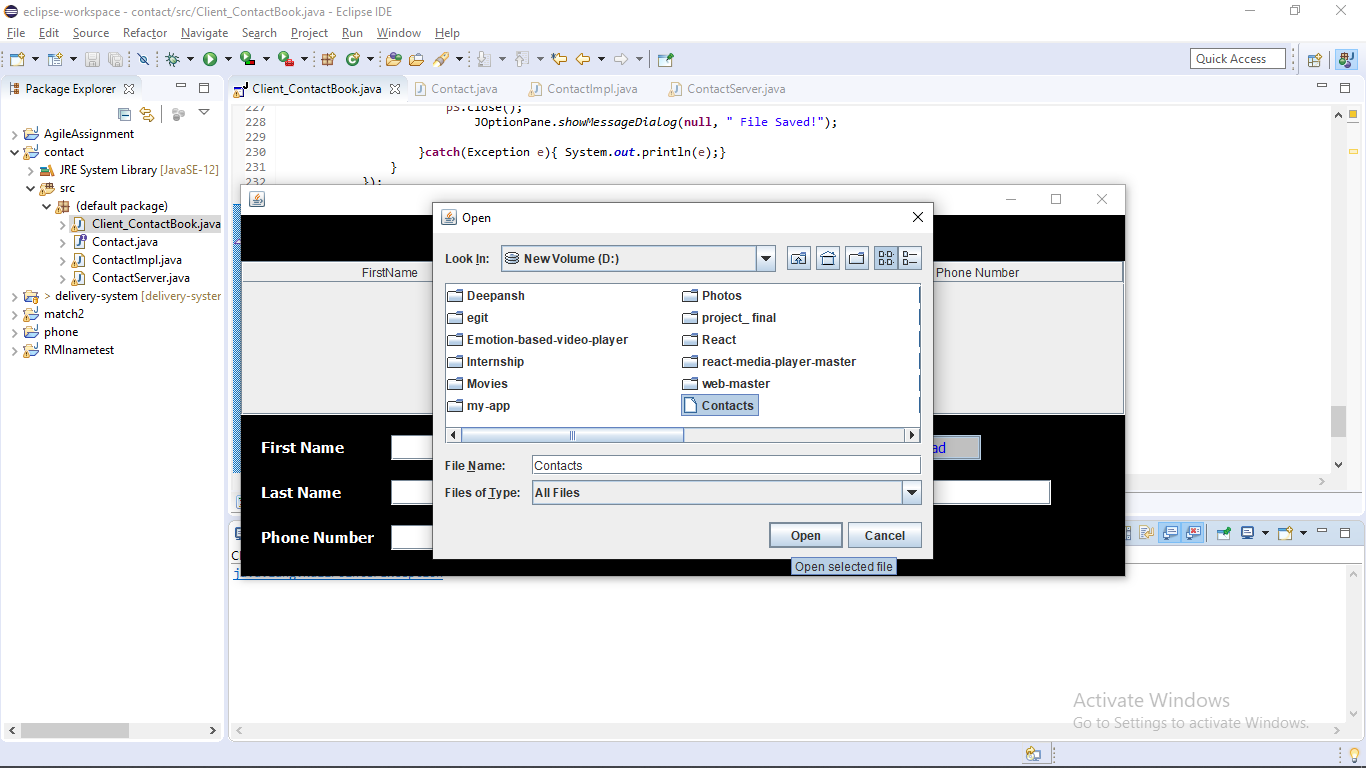
4. Delete the information from Phonebook directory:-

5 .Save the information into file :-

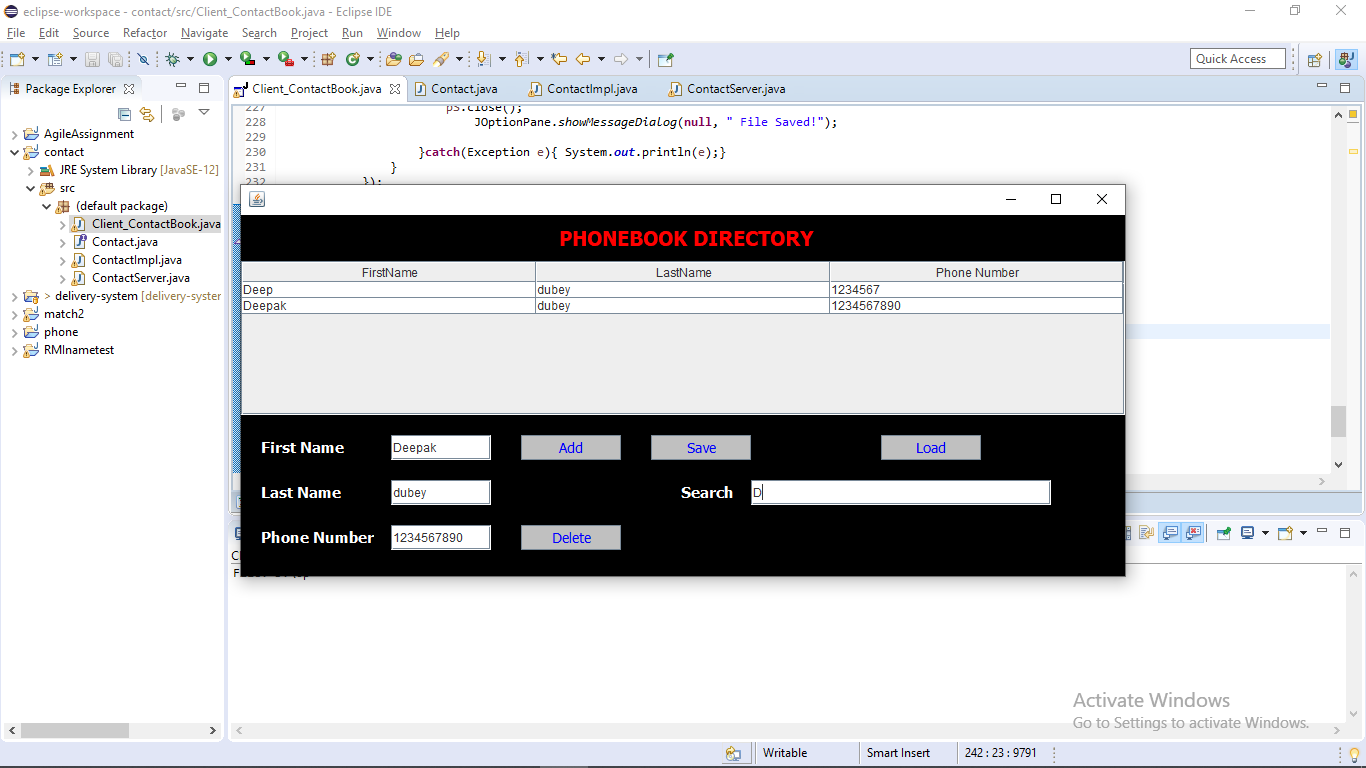




6. Load the file :-



7. Search the file:-



**Problems encountered During Development :-**

1. When I run the RMI server it shows me the error like this :-

Port already in use: 1099; nested exception is

Then I fix the error by solving and looking some notes on Google that how to fix it. Then there is one command which is like a script which will resolve the RMI server .

**Command -** taskkill /f /im chrom\*

**Note :-** Use this command in the command prompt before running the RMI server.

1. Finding difficulties in taking the data from the textfield into the Array List.
2. Problems while doing the code for save button to save the contact as a file format .

**Limitations and Future Enhancement:-**

Limitations

* System works in all formats and its compatible environment.
* No online

Future Enhancements

* As the technology emerges , it is possible to upgrade the system and can be online
* Based on the future security issues , security can be improved using emerging technologies .

**Conclusion:-**

This application has been computed successfully. It is user friendly and has required various options which can be utilized by the user to perform the desired operations.

The goals that are achieved by the software are :-

* Optimum utilization of resources
* Efficient management of records
* User Friendly
* Less processing time and getting required information
* Portable and flexible for further enhancement