

PROGRAMMING TOOL

PROJECT REPORT SUBMISSION 2

GROUP MEMBERS

Kumari Renuka – U101115FCS111
Gulshan Singh – U101115FCS094
Kashish Chaurasia – U101115FCS107
Hitesh Yadav - U101115FCS098

INTRODUCTION

Doubt box is a desktop application. It is a platform developed by us to facilitate communication between the student and teacher to clarify any doubt related to a particular subject or topic. Various tools are used for debugging, creating document, and analysing the complexity of the programme, Eclipse IDE and Java Programming Language, have been used to develop this application.

Contributions:

Gulshan Singh	Database Connectivity, Event Handling and Backend Development
Kashish Chaurasia	Backend and Frontend of server side program and documentation
Kumari Renuka	Backend and Frontend of client side program and testing
Kashish Chaurasia	Frontend development and debugging

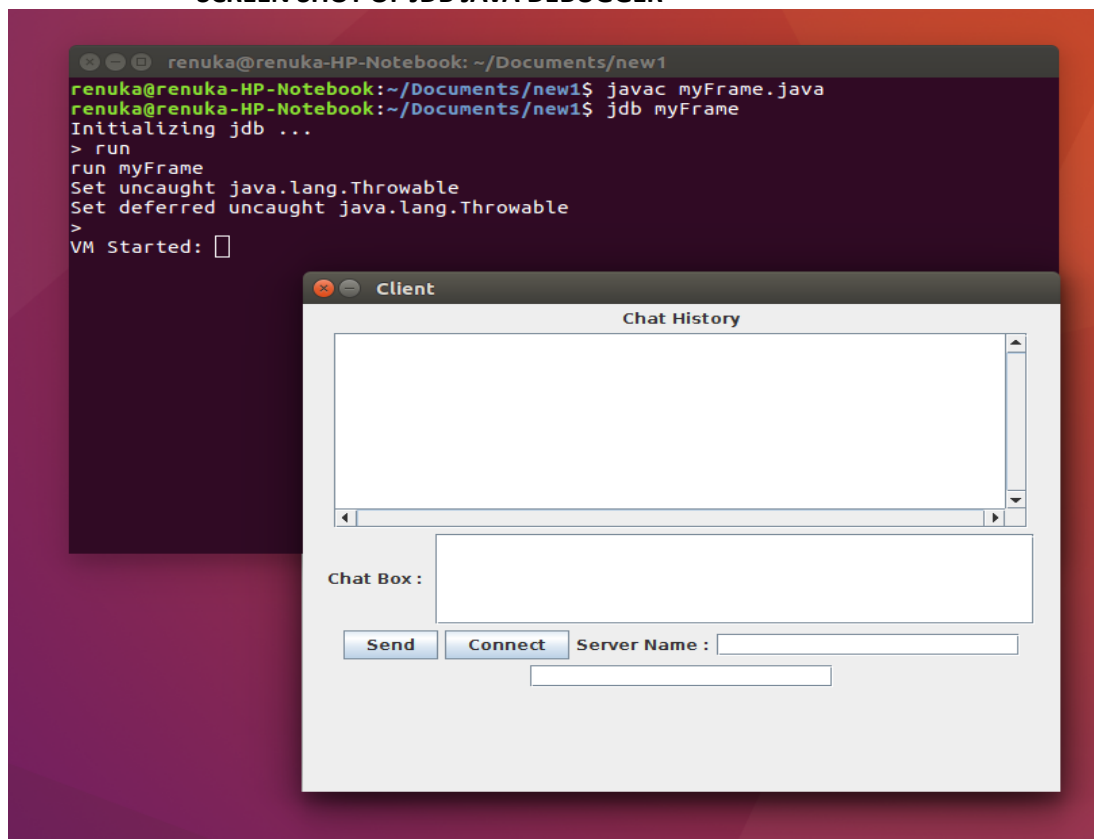
FRONT-END DEVELOPMENT AND JDB

(Hitesh Yadav)

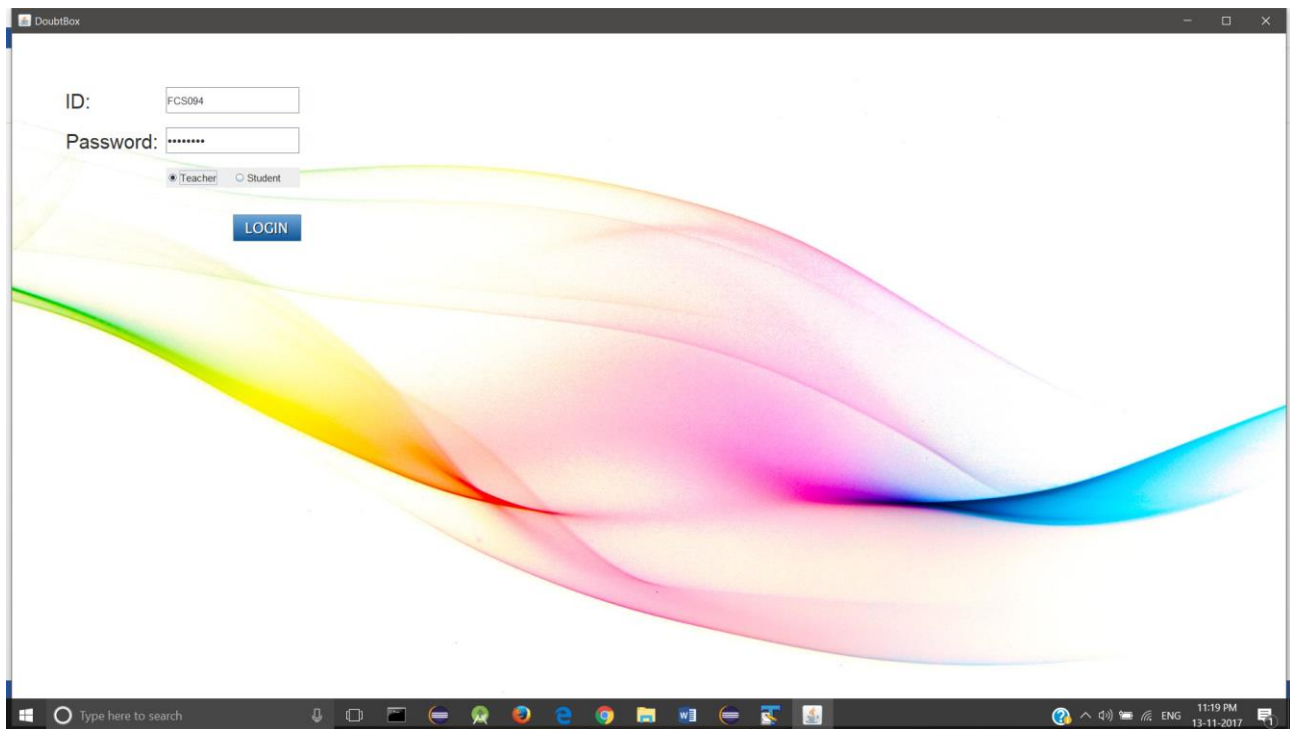
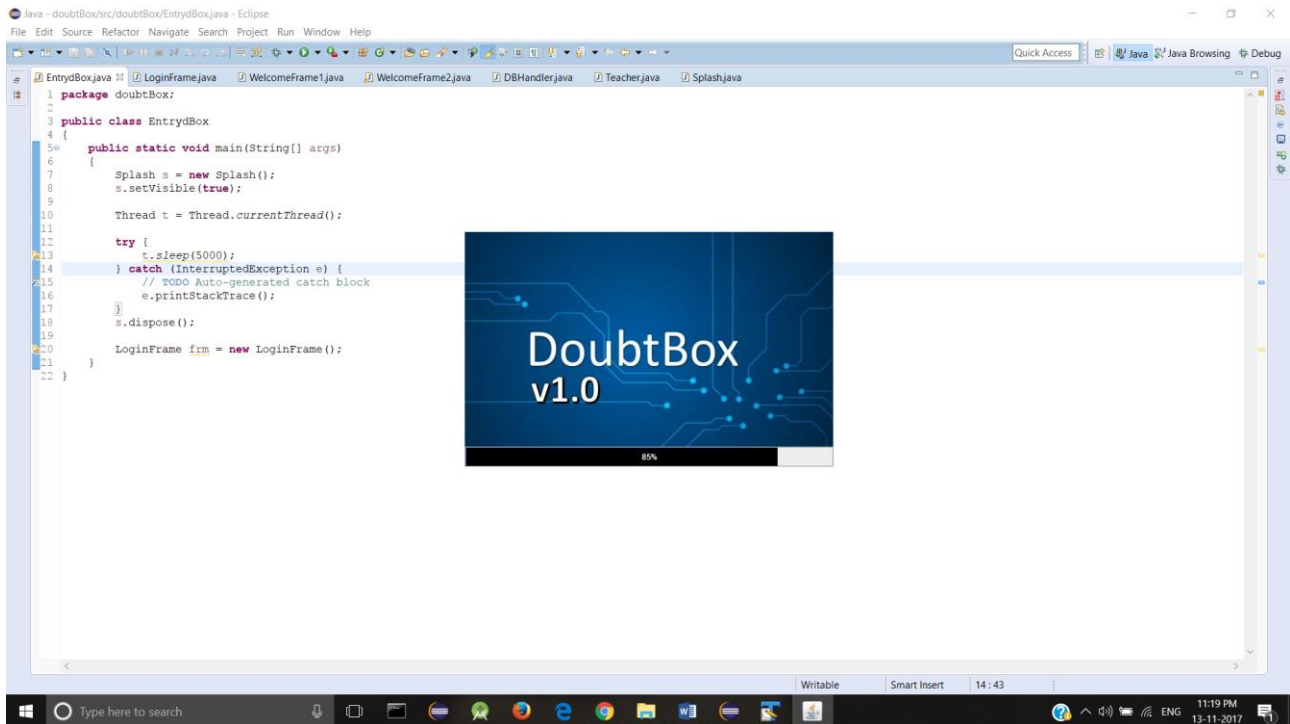
The user interface consists of two frames, one for student and another for faculties. Student frame consist of a login session in which each student will be given a UserId and Password through which they need to login. The Student Login Page consist of four fields- Id, Password, Login button and radio buttons for choosing their designation. After student successful login, it enters into a new frame in which the student can see the list of the teacher, on clicking to the particular teacher name, the complete information regarding the availability or start time of the session and the IP address of the teacher is displayed. By this the student can know the corresponding IP address of the Teacher. This frame consists of a button through which the student can enter into the doubt room. If the login is done as a teacher it will go the frame in which it can enter its whole credentials regarding the start time of session and the IP address. This frame consists of a button through which it can enter into the Doubt room in order to answer the various questions raised by a student. Whole front-end development is done using Eclipse IDE. The front end is developed using Java programming language. Swing and AWT (abstract window toolkit) is used to create the GUI (graphical user interface).

Java command-line debugging tool called JDB is used in order to debug the Java code. JDB is one of the several debuggers available for debugging Java programs. It comes as part of the Sun's JDK. JDB is used by a lot of people for debugging purposes, for the main reason that it is very simple to use, lightweight and being a command-line tool, is very fast. Those who are familiar with debugging C programs with gdb, will be more inclined to use JDB for debugging Java programs. Debugging of the code is done using JDB and the results and displayed below.

SCREEN SHOT OF JDB JAVA DEBUGGER



SCREEN SHOT OF Frontend



DoubtBox

Sign out

Name:

IP:

Time:

Type here to search

11:20 PM 13-11-2017

Gulshan Singh

Sign out

Teacher name:

IP:

Time slot:

Type here to search

11:20 PM 13-11-2017

DATABASE AND BACK-END DEVELOPMENT

(Gulshan Singh)

Database named “Doubtbox” is created in order to store the various information. I’ve used Oracle SQL developer as a database for storing this information and used jdbc6 as oracle driver to connect the database to our java program. The DBHandler is the class the handles database and consists of 9 methods (1 constructor, 1 connection establishing method and 7 methods that interact with database):

```
1. public DBHandler()  
2. public java.sql.Connection getDBConWithOracle()  
3. public Teacher getTdetails(java.lang.String strTname)  
4. public java.util.Vector<java.lang.String> getTnameTbl-  
    teacher()  
5. public void insertIntoTblteacher(java.lang.String strTname,  
    java.lang.String strIP,  
    java.lang.String strTime)  
6. public void deleteFromTblteacher(java.lang.String strIP)  
7. public void updateIntoTblteacher(java.lang.String strTname,  
    java.lang.String strIP,  
    java.lang.String strTime)  
8. public boolean isValidTeacher(java.lang.String strUid,  
    java.lang.String strPwd)  
9. public boolean isValidStudent(java.lang.String strUid,  
    java.lang.String strPwd)
```

The database consists of two tables (“USERTEACHER” and “USERSTUDENT”) which stores the student’s and teacher’s login credential (userid varchar(20) primary key, upwd varchar(20)) in order to check if the corresponding Login is valid or not by using the “isValidStudent” and “isValidTeacher” methods.

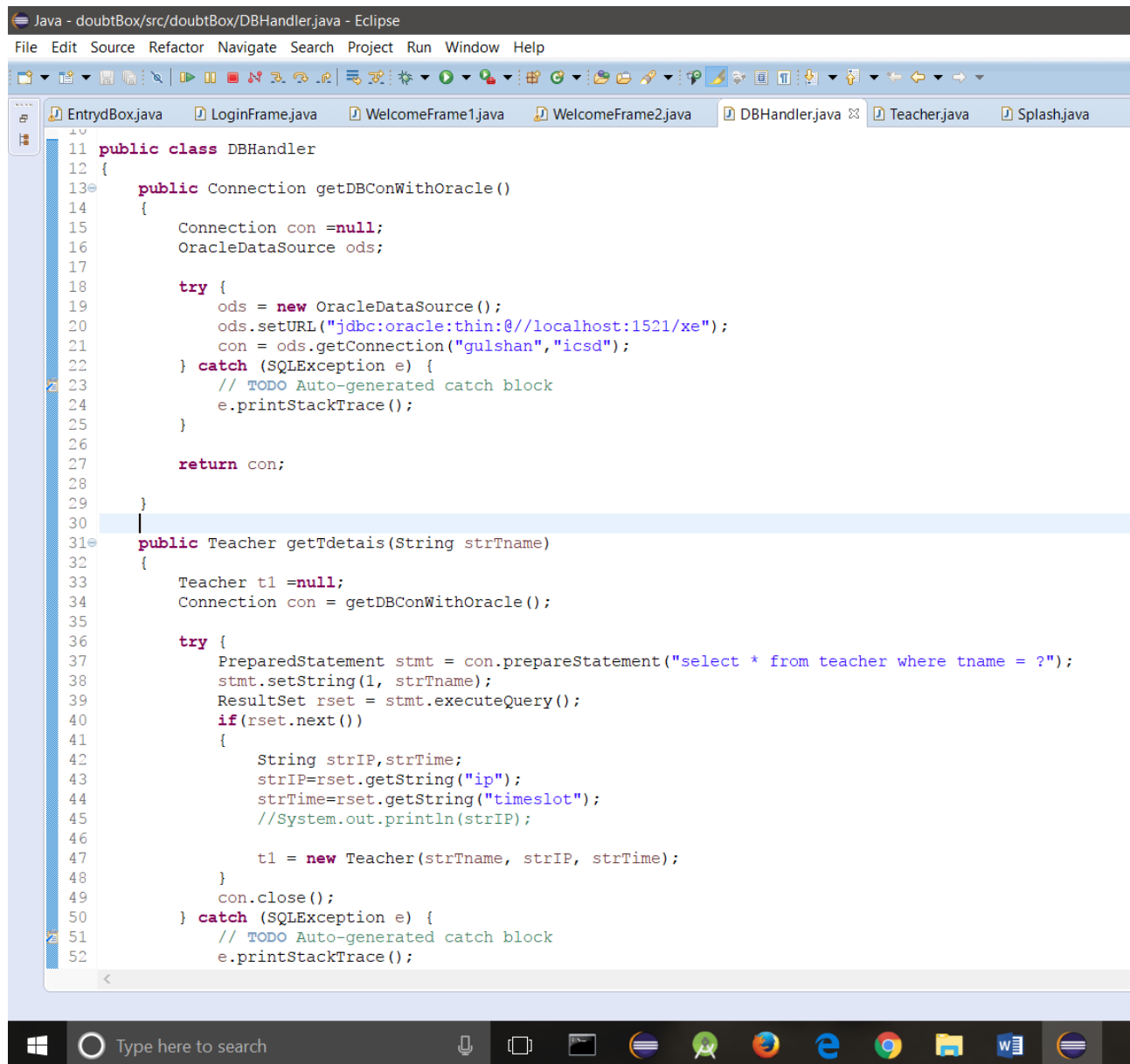
The database also consists of a table name “teacher” which stores all the information entered by a teacher such as the Teacher’s Name, IP address and the time by which a teacher can start the session in order to answer the various queries or doubts of the student. When a teacher logs in, he/she gets a frame where he/she has to enter his/her name, IP address and session time. The teacher can insert one or multiple session, delete any sessions or update any session with the help of corresponding methods declared in DBHandler class.

This information is retrieved from the database when a student logs into the application. When the student clicks on the dropbox the list of all the teachers who have provided information about sessions appears and when a student clicks on any of the name the information about teacher’s (server) IP address and session time gets filled up in their respective textfields ,so that a student can enter a session.

The Database is created using Oracle SQL Developer. Oracle SQL Developer is the Oracle Database IDE. A free graphical user interface, Oracle SQL Developer allows database users and administrators to do their database tasks in fewer clicks and keystrokes. A productivity tool, SQL Developer's main

objective is to help the end user save time and maximize the return on investment in the Oracle Database technology stack. SQL Developer provides powerful editors for working with SQL, PL/SQL, Stored Java Procedures, and XML. Run queries, generate execution plans, export data to the desired format (XML, Excel, HTML, PDF, etc.), execute, debug, test, and document your database programs, and much more with SQL Developer.

SCREENSHOT OF Database related coding



```
Java - doubtBox/src/doubtBox/DBHandler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

EntryBox.java LoginFrame.java WelcomeFrame1.java WelcomeFrame2.java DBHandler.java Teacher.java Splash.java

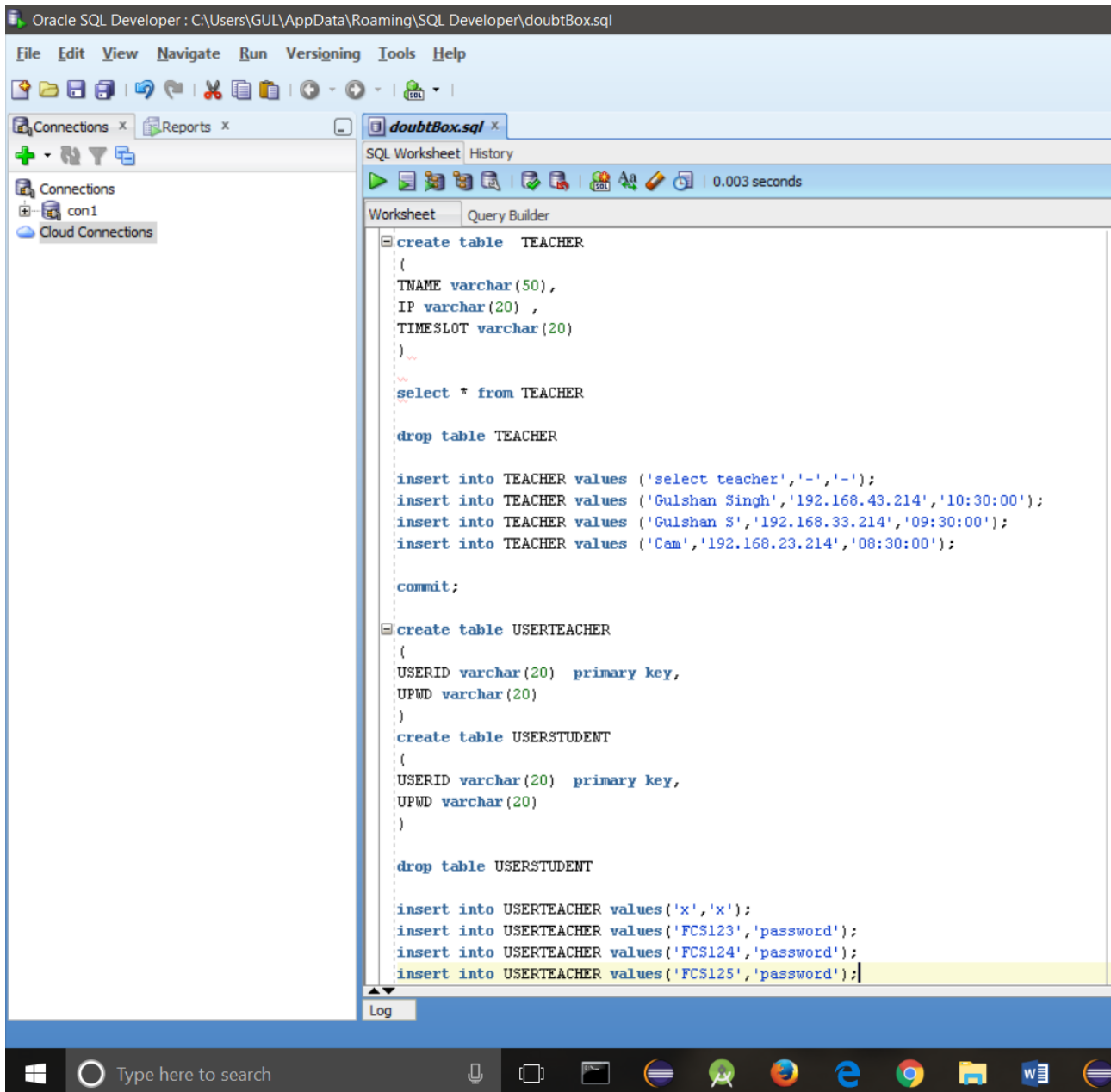
11 public class DBHandler
12 {
13     public Connection getDBConWithOracle()
14     {
15         Connection con =null;
16         OracleDataSource ods;
17
18         try {
19             ods = new OracleDataSource();
20             ods.setURL("jdbc:oracle:thin:@//localhost:1521/x");
21             con = ods.getConnection("gulshan","icsd");
22         } catch (SQLException e) {
23             // TODO Auto-generated catch block
24             e.printStackTrace();
25         }
26
27         return con;
28     }
29
30
31     public Teacher getTdetails(String strTname)
32     {
33         Teacher t1 =null;
34         Connection con = getDBConWithOracle();
35
36         try {
37             PreparedStatement stmt = con.prepareStatement("select * from teacher where tname = ?");
38             stmt.setString(1, strTname);
39             ResultSet rset = stmt.executeQuery();
40             if(rset.next())
41             {
42                 String strIP, strTime;
43                 strIP=rset.getString("ip");
44                 strTime=rset.getString("timeslot");
45                 //System.out.println(strIP);
46
47                 t1 = new Teacher(strTname, strIP, strTime);
48             }
49             con.close();
50         } catch (SQLException e) {
51             // TODO Auto-generated catch block
52             e.printStackTrace();
53         }
54     }
55 }
```

```
Java - doubtBox/src/doubtBox/DBHandler.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

EntryBox.java LoginFrame.java WelcomeFrame1.java WelcomeFrame2.java DBHandler.java Teacher.java Splash.java

128
129
130
131 public void updateIntoTblteacher(String strTname, String strIP, String strTime)
132 {
133     //1- estalish the connection
134     Connection con=getDBConWithOracle();
135     //2- specify your objective
136     try {
137         PreparedStatement stmt=con.prepareStatement("update teacher set tname=?,timeslot=? where ip=?");
138         //3- pass the paramter if any
139         stmt.setString(1, strTname);
140         stmt.setString(2, strTime);
141         stmt.setString(3, strIP);
142         //4- execute your query
143         stmt.executeUpdate();
144         //5- close the connection
145         con.close();
146         System.out.println("data updated ");
147     } catch (SQLException e) {
148         // TODO Auto-generated catch block
149         e.printStackTrace();
150     }
151
152 }
153
154
155 public boolean isValidTeacher(String strUid, String strPwd) {
156     boolean res = false;
157     Connection con = getDBConWithOracle();
158     try {
159         PreparedStatement stmt = con.prepareStatement("Select * from USERTeacher where userid=? and upwd=?");
160         stmt.setString(1, strUid);
161         stmt.setString(2, strPwd);
162         ResultSet rset = stmt.executeQuery();
163         if(rset.next())
164         {
165             res = true;
166         }
167         else
168         {
169             res = false;
170         }
171     }
172 }
```

SCREENSHOT OF Oracle SQL Developer database



Script Output x Query... x

SQL | All Rows Fetched: 7 in 0.002 seconds

	TNAME	IP	TIMESLOT
1	select teacher	-	-
2	Gulshan Singh	192.168.43.214	10:30:00
3	Gulshan S	192.168.33.214	09:30:00
4	Cam	192.168.23.214	08:30:00
5	kashish	1004	00:00
6	GS	192.168.23.43	10:40 pm
7	CAM	192.168.43.24	10:00:00

Script Output x Query... x

SQL | All Rows Fetched: 5 in 0.001 seconds

	USERID	UPWD
1	a	a
2	x	x
3	FCS123	password
4	FCS124	password
5	FCS125	password

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.001 seconds

	USERID	UPWD
1	b	b
2	FCS123	password
3	FCS124	password
4	FCS125	password

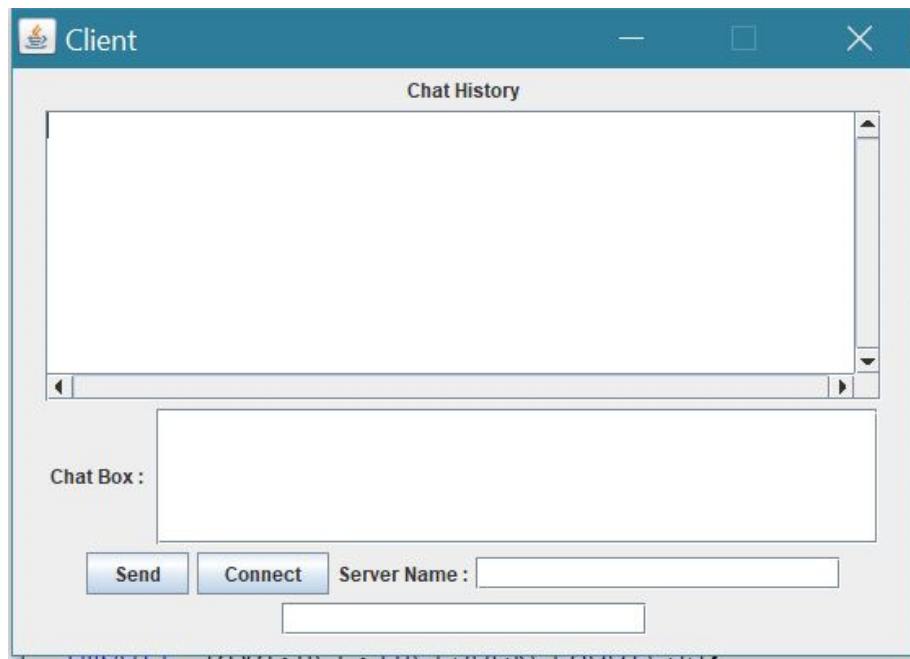
STUDENT'S DOUBT ROOM AND JHAWK METRICS TOOL

(Kumari Renuka)

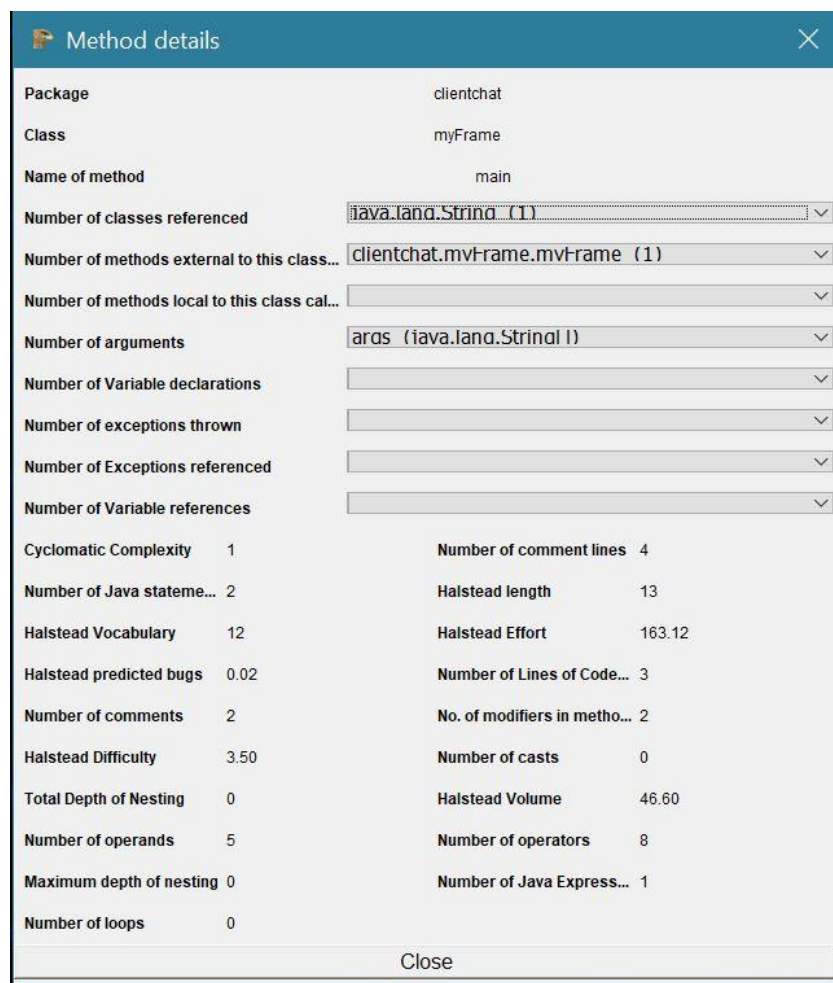
The Student's Doubt room is a window where the actual communication starts. It is a window in which the student can ask question to the faculty or teacher by entering their IP address. This window consist of five field- the "Server Name" in this field the student have to enter the IP address of the Faculty to which it want to ask queries, the "Question Box" where the student have to enter the queries or the question which they want to ask, this query entered by the student which will send to the corresponding teacher who is on the present on the session and to which the student is connected to, the "doubt history" where the student can actually view the answers of the question responded by the corresponding faculty. It gives the complete history of the conversation done between them. This window consists of two buttons - "Connect" which establish the connection between the student and the teacher whose IP Address is mentioned, "Send" is used in order to send the messages to the teacher typed in the Question Box. The coding is done in Java Programming Language in Eclipse IDE. Debugging is done using Eclipse Java Debugger in order to removing bugs, errors or abnormalities from programs. Debugging is considering several breakpoints such as conditional and exceptional breakpoints, watchpoints and stepping commands. The User Interface of the Student's Doubt Room is created using Swings and AWT (abstract window toolkit). Socket Programming in Java is used to enable the communication between the Student and the Teacher. Sockets provide the communication mechanism between two computers using TCP. A Student (client) program creates a socket on its end of the communication and attempts to connect that socket to a Teacher (server). When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket. This is how the connection between the Student and the Teacher is made to facilitate the communication between them.

JHawk Metrics Tool is used in order to calculate the complexity of the code so to make the better improvements. JHawk is a a static code analysis tool - i.e. it takes the source code of your project and calculates metrics based on numerous aspects of the code - for example volume, complexity, relationships between class and packages and relationships within classes and packages. JHawk collects metrics at four different levels. The lowest level is the method level, then up to the class level, then the package level then, finally, the system level. The method, class and package levels correlate with the Java artefacts with these names. The 'system' level represents a group of these artefacts that you wish to examine collectively e.g. all the code in an application. Full documentation relating to the calculation of these metrics is supplied with the product. This tool provides the complete information about the relation between the classes, the lines of code and measures several other parameters. Hence, code analysis is done using JHawk Metric tool and the results have been shown below.


SCREENSHOT OF STUDENT'S DOUBT ROOM



SCREEN SHOT OF JHAWK METRICS TOOL 1



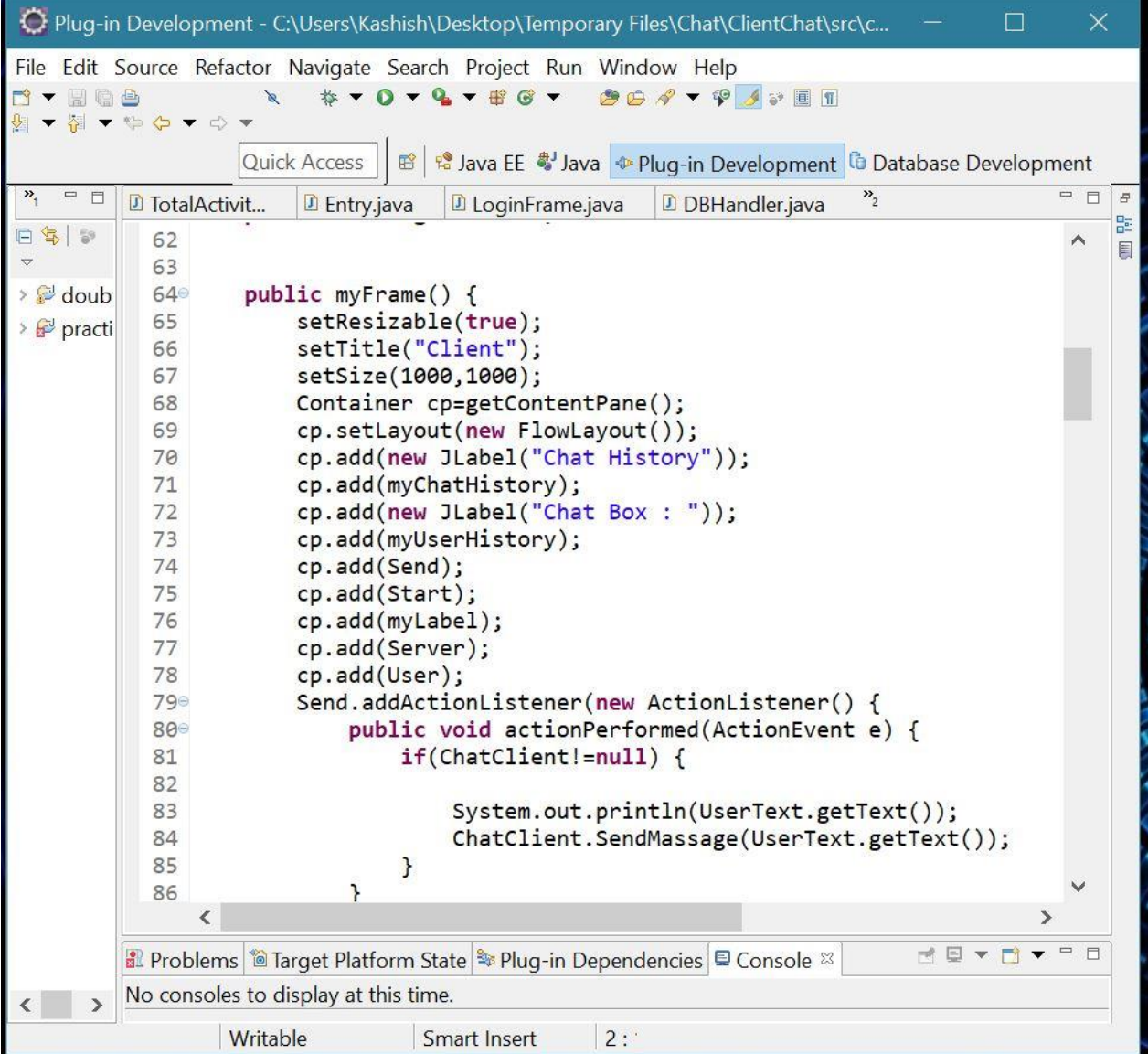
SCREEN SHOT OF JHAWK METRICS TOOL 2


Class Information
✕

Package Name	clientchat		
Name of class	myFrame		
No. of instance variables declared	ChatBox (javax.swing.JTextArea) ▾		
No. of packages imported	com.sun.corba.se.sbi.activation.Reposito... ▾		
No. of local methods called	▾		
No. of Methods called that are in the...	setTitle (1) ▾		
No. of External Methods called	javax.swing.JButton.JButton (2) ▾		
No. of interfaces implemented	▾		
Total number of methods	2	Lack of Cohesion of me...	0.38
Average Cyclomatic Co...	1.50	Total number of Java st...	36
Cumulative Halstead bu...	0.56	Cumulative Halstead eff...	11393.99
Unweighted Class size	15	Total Response For Cla...	8
CBO (Coupling Between...	1	Maintainability Index	94.74
Total Number of Comm...	9	Total Lines of Code in t...	39
Review Factor	3.00	Fan In (Afferent Coupling)	1
Depth of Inheritance Tree	2	Maintainability Index(No...	94.51
Specialization Ratio	0.00	ReUse Ratio	0.50
Cohesion	0.27	LCOM2	2.00
Maximum Cyclomatic C...	2	Cumulative Halstead vo...	1668.11
Total number of query ...	0	Fan Out (Efferent Coupli...	0
Name of superclass	javax.swing.JFrame	SIX	0.00
Total number of superc...	1	Total Cyclomatic Compl...	3
Total number of sub cla...	0	Message passing coupl...	6
Total number of comm...	2	Total Number of Comm...	3
Cumulative Halstead le...	299	No. of modifiers for this...	1

Close

SCREEN SHOT OF THE CODE OF STUDENT'S DOUBT ROOM



The screenshot shows an IDE window titled "Plug-in Development - C:\Users\Kashish\Desktop\Temporary Files\Chat\ClientChat\src\c...". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, running, and debugging. The "Quick Access" bar shows "Java EE", "Java", "Plug-in Development", and "Database Development". The "Plug-in Development" tab is active, showing a list of files: TotalActivit..., Entry.java, LoginFrame.java, and DBHandler.java. The "Entry.java" file is open, displaying the following Java code:

```
62
63
64 public myFrame() {
65     setResizable(true);
66     setTitle("Client");
67     setSize(1000,1000);
68     Container cp=getContentPane();
69     cp.setLayout(new FlowLayout());
70     cp.add(new JLabel("Chat History"));
71     cp.add(myChatHistory);
72     cp.add(new JLabel("Chat Box : "));
73     cp.add(myUserHistory);
74     cp.add(Send);
75     cp.add(Start);
76     cp.add(myLabel);
77     cp.add(Server);
78     cp.add(User);
79     Send.addActionListener(new ActionListener() {
80         public void actionPerformed(ActionEvent e) {
81             if(ChatClient!=null) {
82
83                 System.out.println(UserText.getText());
84                 ChatClient.SendMessage(UserText.getText());
85             }
86         }
87     });
88 }
```

The bottom of the IDE shows the "Problems" tab, "Target Platform State", "Plug-in Dependencies", and "Console". The "Console" tab is active, displaying the message "No consoles to display at this time." The status bar at the bottom indicates "Writable", "Smart Insert", and "2 :".

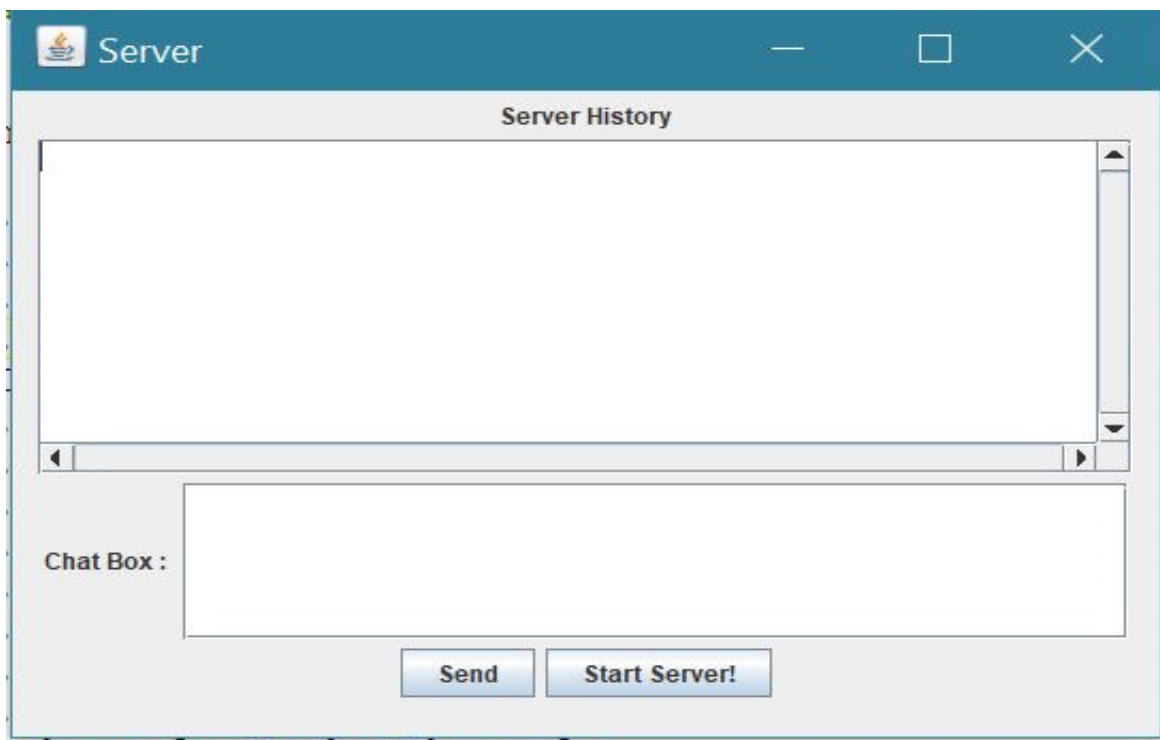
TEACHER'S DOUBT ROOM AND ECLIPSE JAVADOC TOOL

(Kashish Chaurasia)

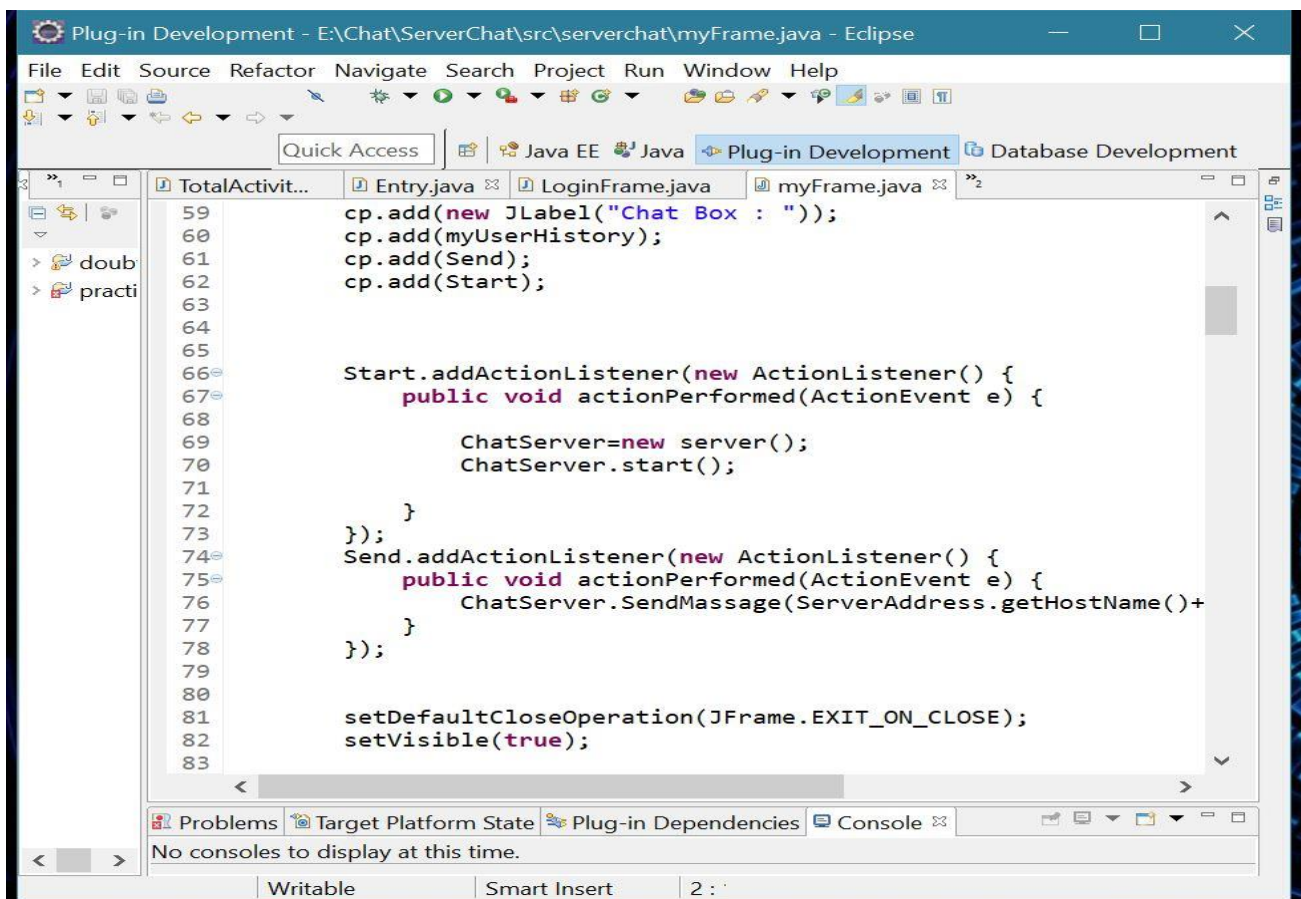
The Teacher's Doubt room is a window which get displayed after a successful login by the teacher. In this window the Teacher can start a session with the student and can answer the various queries raised by them. This window consists of five field- the "Server Name" this field will consist the information about the student's IP, the "Answer Box" where the teacher can answer to the various queries asked to them, this answer can be viewed by the student which to which they are connect to, the "doubt history" where the teacher can actually see the whole conversation or the questions raised by the Student. This window consists of two buttons - "Connect" which establish the connection between the teacher and the student whose IP Address is mentioned, "Send" is used in order to send the answers of the query to the student typed in the Answer Box. The whole coding is done in Java Eclipse IDE. Debugging is done using Eclipse Java Debugger in order to removing bugs, errors or abnormalities from programs. Debugging is considering several breakpoints such as conditional and exceptional breakpoints, watchpoints and stepping commands. The User Interface of the Teacher's Doubt Room is created using Swing and AWT (abstract window toolkit). Socket Programming in Java is used to enable the communication between the Student and the Teacher. By using Socket programming the connection is established between them so that they can communicate with each other. To start a session the teacher starts the server using a server type variable (server is a class that extends Thread class). The connection is opened up by using `ServerSocketChannel.open()` method and the the application gets ready to accept new Connections requests from client side. To register a new Client, class Client is defined in which `addClient(SocketChannel newClient)` adds the new client.

Eclipse JavaDoc Tool is used in order to create the documentation of the project. The specifications for the Eclipse platform APIs are captured in the form of Javadoc comments on API packages, interfaces and classes, methods and constructors, and fields. The Java Doc tool (running the standard doclet) extracts these specifications and formats them into browsable form (HTML web pages) which become the reference section of the documentation set describing the Eclipse platform to ISVs. As a consequence, the bar is significantly higher for API Javadoc than for non-API. Oracle requirements for writing Java API Specification deals with required semantic content of documentation comments for API specifications for the Java platform. All Eclipse project APIs should follow these conventions. Hence the Java Doc have been created for the future references.

SCREENSHOT FOR TEACHER'S DOUBT ROOM



SCREENSHOT OF CODE OF THE TEACHER'S DOUBT ROOM



SCREENSHOT OF Javadoc documentation

PACKAGECLASSUSETREEDEPRECATEDINDEXHELP

PREVNEXTFRAMESNO FRAMESALL CLASSES

Hierarchy For All Packages

Package Hierarchies:
doubtBox

Class Hierarchy

- java.lang.Object
 - java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
 - java.awt.Container
 - java.awt.Window (implements javax.accessibility.Accessible)
 - java.awt.Frame (implements java.awt.MenuContainer)
 - javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
 - doubtBox.LoginFrame
 - doubtBox.Splash
 - doubtBox.WelcomeFrame1
 - doubtBox.WelcomeFrame2
- doubtBox.DBHandler
- doubtBox.EntrydBox
- doubtBox.Teacher

PACKAGECLASSUSETREEDEPRECATEDINDEXHELP

PREV CLASSNEXT CLASSFRAMESNO FRAMESALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHODDETAIL: FIELD | CONSTR | METHOD

doubtBox

Class DBHandler

java.lang.Object
doubtBox.DBHandler

public class **DBHandler**
extends java.lang.Object

Constructor Summary

Constructors

Constructor and Description
DBHandler()

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
void	deleteFromTblteacher(java.lang.String strIP)	
java.sql.Connection	getDBConWithOracle()	
Teacher	getTdtails(java.lang.String strName)	
java.util.Vector<java.lang.String>	getNameTblteacher()	
void	insertIntoTblteacher(java.lang.String strName, java.lang.String strIP, java.lang.String strTime)	

Type here to search

Field Summary

Fields inherited from class javax.swing.JFrame

EXIT_ON_CLOSE

Fields inherited from class java.awt.Frame

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Constructors

Constructor and Description

LoginFrame()

