Z2RF1WMJ3JAVA BASIC



OCTOBER 18, 2018

JAVA BASIC PROJECT - CLIENT, SERVER & DATABASE PROGRAMMING

Table of Contents

Program details	
Description of the program:	
Instructions on which OS to use:	
Type of & version of programs used:	
What do you need to run the program?	
How to run the program?	
Error handling & user input	
Example 1	2
Example 2	2
Creating the database	3
Creating tables.	4
Server	5
Server UI Class	5
Session Class	10
Protocol Class	13
Client	24
Client UI Class	24
User Interface	37
Project Structure	39

Program details.

Description of the program:

The program consists of a client, server & database. There is a clientUI & serverUI. The client sends the server requests to perform certain actions. The server receives the client requests decodes/process the client request & then gets or writes input/output from a database. Then returns the output to client.

Instructions on which OS to use:

Any platform can be used to run the program which contains the java runtime.

Type of & version of programs used:

NetBeans IDE 8.2 was used on the Windows 10 operating system.

What do you need to run the program?

The following things are needed to run the program.

- Microsoft SQL Server Management Studio
- Microsoft JDBC Driver 6.2 for SQL server
- Java SDK
- NetBeans IDE

How to run the program?

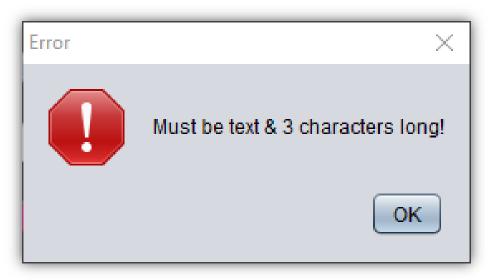
To open the program the following are the instructions:

- 1) Install the Java SDK
- 2) Install Microsoft SQL Server Management Studio
- 3) Install NetBeans IDE
- 4) Download the Microsoft JDBC Driver 6.2 for SQL driver (included in project dir.)
- 5) Run NetBeans IDE
- 6) Create a database connection in NetBeans
- 7) Import Microsoft JDBC SQL Driver jar file (included already)
- 8) Open the "Dashboard" project file & click run

Error handling & user input.

For error handling, **try...catch** blocks are used & exceptions are caught. Regular expressions are also used to get the input from the user in the correct format.

Example 1



Handling an error when user tries to search for a name with less than 3 characters.

Example 2



Handling an error when user tries to connect to server when server is offline.

Creating the database.

This SQL query creates a database.

```
Filename:
                             goData
3
         Author:
                             Aysham Ali Hameed
4
                             13th October 2018
         Created:
5
         Operating System: Windows 10
                            MSSMS v17.8.1
6
         Version:
7
         Description:
                            Creating the database and tables.
8
9
    L*/
.0
11
         12
         USE MASTER
13
         GO
14
15
         PRINT 'Using master database.'
16
17
18
         IF EXISTS (SELECT name FROM master.dbo.sysdatabases where name='pDatabase')
19
         BEGIN
             DROP DATABASE pDatabase
21
             PRINT 'Existing database dropped.'
22
         END
23
         GO
24
25
         CREATE DATABASE pDatabase
26
                 ON PRIMARY (
                     NAME='pDatabase_data',
27
28
                     FILENAME='C:\DB\pDatabase data.mdf',
29
                     SIZE=5MB,
                     FILEGROWTH=10%
31
32
                 LOG ON (
33
                     NAME='pDatabase log',
34
                     FILENAME='C:\DB\pDatabase log.ldf',
35
                     SIZE=5MB,
36
                     FILEGROWTH=10%
37
38
         GO
39
40
```

Creating tables.

Creates tables inside the database.

```
13
          -- CREATING TABLES ======
45
          USE pDatabase
46
47
48
          DROP TABLE IF EXISTS dbo.Users
49
50
          GO
51
          DROP TABLE IF EXISTS dbo.Animals
52
53
          DROP TABLE IF EXISTS dbo. Species
54
55
56
57
          CREATE TABLE Species (
58
              speciesID VARCHAR(20) NOT NULL PRIMARY KEY,
59
              speciesName VARCHAR(30) NOT NULL
60
          )
61
          GO
62
63
          CREATE TABLE Animals (
64
              animalID VARCHAR(20) NOT NULL PRIMARY KEY,
65
              animalName VARCHAR(30) NOT NULL,
66
              animalDesc VARCHAR(100) NOT NULL,
              speciesID VARCHAR(20) NOT NULL REFERENCES Species(speciesID) ON DELETE CASCADE
68
          )
69
          GO
70
71
          CREATE TABLE Users (
72
              userID VARCHAR(20) NOT NULL PRIMARY KEY,
73
              userName VARCHAR(30) NOT NULL UNIQUE,
74
              userPassword VARCHAR(30) NOT NULL
75
          )
76
          GO
77
78
          --INSERTING DATA INTO Users TABLE
79
80
          INSERT INTO Users
81
          VALUES ('1', 'BruceW7', 'bruceisbatman'),
82
                   ('2', 'PeterP8', 'peterisspiderman#123')
83
          GO
```

Server.

Server UI Class

```
Filename
                              Server
 2
              Author
                              Aysham Hameed
 3
              Created
                             15th October 2018
 4
 5
              05
                              Windows 10
                            Netbeans IDE 8.2
 6
              Version
 7
              Desription : The Server UI, starts and shutdowns server.
 8
 9
10
      package UI.Server; //contained in UI.Server package.
11
12
   p import java.awt.Color; //imported for changing button colors.
13
      import javax.swing.*; //imported for showing dialogs.
14
      import java.net.*; //imported to work with sockets.
15
      import java.io.*; //imported to work with streams.
16
17
18
      public class Server extends javax.swing.JFrame { //main class Server extends
19
20
                                                      //JFrame to display frame
21
          boolean loggedIn = false; //flag variable to verify login.
22
23
          ServerSocket serverSocket; //creating ServerSocket object.
          boolean listening; //flag variable to check server status
24
25
```

```
public static void main(String args[]) {
318
                /* Set the Nimbus look and feel */
319
320
                 Look and feel setting code (optional)
341
                /* Create and display the form */
342
                java.awt.EventQueue.invokeLater(new Runnable() {
                    public void run() {
                        new Server().setVisible(true);
345
346
347
                });
348
349
```

```
38
           //INSTANTIATES SERVER SOCKET OBJECT & STARTS LISTENING
39
           public void turnOnServer(){
    if(listening==false){
40
41
                   try{
                       serverSocket = new ServerSocket(7777); //listening on port 7777.
42
                       listening =true:
                                            //set listening true.
43
                       System.out.println("Server closed :" + serverSocket.isClosed());
44
                   }catch (IOException e){ //catch IO exception.
45
                       System.out.println(e.toString()); //print exception message.
46
47
                       System.exit(1); }//exit process.
                    jLabel1.setText("Server status : online"); //updating server status.
48
49
               } else {
                   ///UPDATING SERVER STATUS USING ANONYMOUS THREAD OBJECT
50
                   Thread timer = new Thread(new Runnable() { //create new thread
<u>Q.</u>
                       public void run() { //using overriden run method
53
                            //chaning text
54
                          jLabel1.setText("Server status : already running!");
                          try{
55
                          Thread.sleep(1000); //pause for 1 second
56
                           }catch (InterruptedException e){} //catching exception
57
                          jLabel1.setText("Server status : online"); //update text
58
59
60
                   });timer.start(); //starts thread object.
                 //LISTENING INSIDE THREAD
61
                   Thread t = new Thread(new Runnable() {
<u>Q.</u>
₩.
                       public void run() {
64
                            EXPLANATION: so while the server is listening then create
65
                                         a new session. (Meaning allow clients to connect
66
                                         to the server)
67
68
69
70
                             while(listening){
                               try{
71
                                     new Session(serverSocket.accept());
Q.
73
                                     System.out.println("Listening!");
74
                                }catch (IOException e){
                                     System.out.println(e.toString());
75
76
77
                             }
78
79
                   });
                   t.start();
80
81
```

```
//CLOSES SERVER SOCKET
            public void shutdownServer(){
 84
                //RUNNING INSDE ANONYMOUSE THREAD
 85
                Thread s = new Thread(new Runnable() {
87
                    @Override
                    public void run() {;
₩‡
 89
                        try{
 90
                            EXPLANATION: Create a socket. Create outputstream object.
 91
 92
                                          Tell self (Server) to shutdow.
 93
                                         And close serverSocket
                                          And set listening to false so can check if
 94
                                          server is online or offline (SEE OTHER
 95
                                          METHODS)
 96
 97
                             Socket sSocket = new Socket("localhost",7777);
 98
                             ObjectOutputStream send = new ObjectOutputStream(sSocket.getOutputStream());
99
                             send.writeObject("shutdown");
100
101
                             serverSocket.close();
102
103
                        listening = false;
104
                        System.out.println("Server closed : " + serverSocket.isClosed());
105
                        jLabel1.setText("Server status : offline"); //updating server status.
106
107
                        }catch (IOException e){ //catch IOException
108
109
110
                });
111
                s.start(); //start thread
112
113
```

```
//SERVER CONSTRUCTOR
public Server() {
    initComponents();
    listening = false; //set listening to false by default.
}
```

```
//LOGIN REQUEST
116
            public void login() {
117
               JTextField username = new JTextField("admin"); //get username
118
119
               JPasswordField password = new JPasswordField("password");//get password
               //create input fields
120
121
               Object message[] = {"Username : ", username, "Password : ", password};
               //display input fields
122
               int option = JOptionPane.showConfirmDialog(null, message, "Login", JOptionPane.OK_CANCEL_OPTION);
123
124
125
                //based on option slected
               if (option == JOptionPane.OK_OPTION) {
126
                    //if ok option is selected the check is username = "admin
127
128
                    if (username.getText().equals("admin")) {
                        //if username is correct then check if password is correct
129
130
                        if (password.getText().equals("password")) {
131
                            //if password is correct then set
                            //logged in = true
132
133
                            loggedIn = true;
134
                            //display welcome message
                            JOptionPane.showMessageDialog(null, "Welcome admin!");
135
136
                            //if password is wrong show correct message
137
138
                            if (!password.getText().equals("password")) {
139
                            JOptionPane.showMessageDialog(null, "Wrong password!", "Error", JOptionPane.ERROR_MESSAGE);
140
                   } else
141
142
                       //if username is wrong show correct message
143
                        if (!username.getText().equals("admin")) {
144
                        JOptionPane.showMessageDialog(null, "Wrong username!", "Error", JOptionPane.ERROR_MESSAGE);
145
146
147
               } else {
                   //if cancel button is pressed then exit system
148
149
                   System.exit(0);
150
151
```

```
//WHEN WINDOW IS OPENED (THE FIRST THING TO BE DONE)

private void formWindowOpened(java.awt.event.WindowEvent evt) {

while (loggedIn == false) {//while not logged int

login(); //keep displayin login request.
}

}
```

```
//START BUTTON HOVER IN
279
           private void startButtonMouseEntered(java.awt.event.MouseEvent evt) {
280
    281
                startButton.setContentAreaFilled(true); //make button colorful
                startButton.setBackground(new Color(0, 202, 106)); //change to color
282
                startButton.setForeground(Color.black); //change font to black
283
284
285
286
           //START BUTTON HOVER OUT
           private void startButtonMouseExited(java.awt.event.MouseEvent evt) {
287
                startButton.setContentAreaFilled(false); //make button transparent
288
289
                startButton.setForeground(Color.white); //change font color to white
290
291
292
           //SHUTDOWN BUTTON HOVER IN
           private void shutdownButtonMouseEntered(java.awt.event.MouseEvent evt) {
293
                shutdownButton.setContentAreaFilled(true);//make button colorful
294
295
                shutdownButton.setBackground(new Color(204, 0, 102));//change to color
296
                shutdownButton.setForeground(Color.black);//change font to black
297
298
           //SHUTDOWN BUTTON HOVER OUT
299
           private void shutdownButtonMouseExited(java.awt.event.MouseEvent evt) {
300
    301
                shutdownButton.setContentAreaFilled(false); //make button transparent
302
                shutdownButton.setForeground(Color.white);//change font color to white
303
304
```

```
//STARTS SERVER
305
            private void startButtonActionPerformed(java.awt.event.ActionEvent evt) {
306
307
                turnOnServer(); //turn on server
308
309
            //SHUTDOWNS SERVER
310
            private void shutdownButtonActionPerformed(java.awt.event.ActionEvent evt) {
311
    巨
                shutdownServer(); //turn off server
312
313
```

Session Class

```
Filename
                              Session
 2
                         :
              Author
                          : Aysham Hameed
 3
              Created
                         : 15th October 2018
 4
                          : Windows 10
 5
 6
              Version
                         : Netbeans IDE 8.2
 7
              Description : Allows multiple clients & creates a session for each.
 8
 9
10
      package UI.Server; //contained in UI.Server package
11
12
13

□ import java.net.*; //imported to work with sockets.

     import java.io.*;
                         //imported to work with streams.
14
15
      public class Session implements Runnable { //Implementing the Runnable class.
16
17
          //VARIABLES FOR SOCKET, OBJECT STREAMS & THREAD
18
19
          public Socket clientSocket;
20
          public ObjectOutputStream send2CLIENT = null;
          public ObjectInputStream getfromCLIENT = null;
21
          private Thread runner;
22
23
24
```

```
//SESSION CONSTRUCTOR RECIEVES A SOCKET AS PARAMETER
27
28
          public Session(Socket s){
   阜
29
              clientSocket = s; //recives socket parameter
30
31
              //CREATING STREAM OBJECTS
32
              try{
                  //instantiates new ObjectOutputStream
33
                  send2CLIENT = new ObjectOutputStream(clientSocket.getOutputStream());
34
35
                  //instantiates new ObjectInputStream
                  getfromCLIENT = new ObjectInputStream(clientSocket.getInputStream());
36
              } catch (IOException e){ //catches IO exception
37
38
                  System.out.println(e.toString()); //prints exceptio message
39
40
              //STARTING THREAD
41
              if(runner==null){  //if runner thread is empty
42
                  runner = new Thread(this); //redefine thread object
43
                  runner.start(); //start runner thread.
45
46
47
```

```
public void run(){
               //WHILE IT IS BUSY WITH CURRENT THREAD
51
               while(runner==Thread.currentThread()){//while1 start----------
52
                   //PAUSING FOR 10 MILISECONDS
53
                   try{
                       Thread.sleep(10);
56
                   }catch (InterruptedException e){System.out.println(e.toString());}
57
                   //KEEP ON LISTENING
58
                   while(true){
59
                       try{
60
                           //prints out list of active connections.
                           System.out.println("Who is connected?: "+clientSocket.
62
63
                                                                    getInetAddress());
                           //gets client request
                           String clientRequest = (String) getfromCLIENT.readObject();
65
                           System.out.println("Client : "+clientRequest);
66
                           //new Protocol class object
67
                           Protocol decoder = new Protocol();
68
69
                           //clientRequest being processed
70
                           String output = decoder.processInput(clientRequest);
                           //sends client output
                           send2CLIENT.writeObject(output);
72
73
                           String tableName = decoder.SEARCH_TABLE;
74
                           //if output when user tries searching is
75
                           if(output.equals("nameExists")){ //"nameExists"
76
                                //sending the client name of table
77
78
                                send2CLIENT.writeObject(tableName);
                               if(tableName.equals("Animals")){ //IF ANIMAL TABLE
79
                                   //GET ANIMAL DATA
                                                      //SELECETED
80
                                   String animalID[] = (String[]) decoder.getAnimalID();
81
82
                                   String animalNames[]=(String[]) decoder.getAnimalNames();
83
                                   String description[]=(String[]) decoder.getDescription();
                                   String speciesIDFK[]=(String[]) decoder.getSpeciesIDFK();
84
                                   send2CLIENT.writeObject(animalID);
85
                                   send2CLIENT.writeObject(animalNames);
86
87
                                   send2CLIENT.writeObject(description);
                                   send2CLIENT.writeObject(speciesIDFK);
```

```
90
                                 //IF SPECIES TABLE IS SELECTED
  91
                                 if(tableName.equals("Species")){
                                     //GET SPECIES DATA
  92
                                     String speciesID[] = (String[]) decoder.getSpeciesID();
  93
  94
                                     String speciesNames[] =(String[]) decoder.getSpeciesNames();
  95
                                     send2CLIENT.writeObject(speciesID);
  96
                                     send2CLIENT.writeObject(speciesNames);
  97
  98
  99
                             } else
 100
                              if(output.equals("shutdown")){
101
102
                                  System.out.println("output = "+output);
                                  break;
103
104
                              }
 105
106
107
108
                         }catch (IOException e){ //catching IOException
109
 110
                             System.out.println(e.toString()); //exception message
111
                             System.exit(1); //exit process
                         catch (ClassNotFoundException e){ //catch ClassNotFound excep
112
                             System.out.println(e.toString()); //prints excep message
 113
114
115
116
                     }//while2 end---
117
 118
                     try{
                     clientSocket.close();
119
                     getfromCLIENT.close();
120
 121
                     send2CLIENT.close();
                     } catch (IOException e){}
122
                 }//while1 end-----
123
 124
125
126
```

Protocol Class

```
2
               Filename
                               Protocol
              Author
                               Aysham Hameed
 3
              Created
                               15th October 2018
 4
                               Windows 10
 5
 6
              Version
                               Netbeans IDE 8.2
 7
              Desription :
                               Decodes each request from client.
 8
 9
10
11
      package UI.Server; //contained in UI.Server package
12
   □ import java.sql.*; //importing sql package
13
14
15
      public class Protocol{ //Class name ="Protocol"
16
17
18
          //VARIBLES
          //database driver name
19
          String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
20
          //database source url
21
          String sourceURL = "jdbc:sqlserver://localhost:1433;databaseName=pDatabase";
22
23
          String dusername = "sa";
                                           //database username
          String dpassword = "123456";
24
                                           //database password
          Connection connection = null;
                                         //connection object
25
26
27
          String [] animalName; //used to store animal names
          String [] animalID; //used to store animal id
28
29
          String [] description; //used to store animal description
          String [] speciesIDFK; //used to store species ID FK
30
31
          String [] speciesName; //used to store species names
32
          String [] speciesID;//used to store species id
33
34
          public String SEARCH_TABLE = ""; //variable to get name of table
35
36
```

```
Protocol(){
37
            //Setting up database driver and connection
38
39
            try{
             Class.forName(driverName);
40
             connection= DriverManager.getConnection(sourceURL,dusername,dpassword);
41
            }catch (ClassNotFoundException e){System.out.println(e.toString());}
            catch (SQLException e){System.out.println(e.toString());}
            System.out.println("Database connected!");
44
45
            //-----
46
```

```
public int getAnimalTableCount(String name){
48
49
               //Gets the number of records in Animal table
50
               int count=0:
51
52
               try{
                   Statement statement = connection.createStatement();
53
                   //displays all
54
                   String query = "select distinct animalName from Animals where "
55
                           + "(Animals.animalName like '%"+name+"%')";
56
                   ResultSet rec = statement.executeQuery(query);
                   //counts by incrementing count variable (NUMBER OF VARIABLES)
58
                   while(rec.next()){
59
                       count++;
60
61
62
                   rec.close();
63
                   //catches SQL exception
64
               }catch (SQLException e){
65
                   System.out.println(e.toString());
66
67
               return count;
68
69
```

```
public int getSpeciesTableCount(String name){
71
72
               //Gets the number of records in Species table
73
74
               int count=0;
               try{
75
                   Statement statement = connection.createStatement();
76
                   //displays all
77
                   String query = "select distinct speciesName from Species where "
78
                           + "(Species.speciesName like '%"+name+"%')";
79
                   ResultSet rec = statement.executeQuery(query);
81
                   //counts by incrementing count variable (NUMBER OF VARIABLES)
82
                   while(rec.next()){
83
84
                       count++;
                   }
85
                   rec.close();
86
                   //catches SQL exception
87
               }catch (SQLException e){
88
                   System.out.println(e.toString());
89
90
91
               return count;
92
```

```
public String checkSearchName(String name, String table){
  94
  95
                boolean NAMEfound=false; //set name found to false
                try{
  96
                        Statement statement = connection.createStatement();
  97
                        String query="";//blank query
  99
                        ResultSet rec=null; //rec object set to null
                        //-----
100
                        //if table = "Animals" then
 101
                        if(table.equals("Animals")){
102
                         SEARCH TABLE = "Animals";
 103
104
                        //make query find all different animals from Animal table where
                        //name is found part of
 105
                        query = "select distinct * from Animals where"
106
                                + " (Animals.animalName like '%"+name+"%')";
 107
108
                        rec = statement.executeQuery(query);//execute query
 109
                        //redefines array size to number of records in animal table
110
                        animalName = new String[getAnimalTableCount(name)];
 111
                        animalID = new String[getAnimalTableCount(name)];
112
 113
                        description = new String[getAnimalTableCount(name)];
                        speciesIDFK = new String[getAnimalTableCount(name)];
114
115
                        int counter=0; //used as an index for inserting into array
116
117
                        while(rec.next()){
                            //if record starts with name or contains part of name then
118
                            if(rec.getString("animalName").contains(name) ||
 119
                                    rec.getString("animalName").equalsIgnoreCase(name)){
120
 121
                                NAMEfound = true; //make name found true
                                String n = rec.getString("animalName"); //get name
122
123
                                String i = rec.getString("animalID"); //get id
                                String d = rec.getString("animalDesc"); //get description
124
125
                                String f = rec.getString("speciesID"); // get fk
                                counter++; //increase counter
126
                                animalName[counter-1] = n; //& add name to array
 127
                                animalID[counter-1] = i; //add animalID to arrayID
128
129
                                description[counter-1] = d; //adding description
                                speciesIDFK[counter-1] = f; //adding fk
1.30
 131
                                System.out.println(rec.getString(1));
132
                                //if name is not found
 133
                            } else if(!rec.getString("animalName").contains(name)){
134
                                NAMEfound = false; //set namefound to false
 135
136
137
138
 139
```

```
//else if table = "Species" then
143
144
                        else if(table.equals("Species")){
                           SEARCH TABLE = "Species";
145
146
                        //make query find all different species from Species table where
147
                        //name is found part of
148
                            query = "select distinct * from Species "
                                    + "where (Species.speciesName like '%"+name+"%')";
 149
150
151
                            rec = statement.executeQuery(query);//execute query
152
                            //redefines array size to number of records in species table
153
                            speciesName = new String[getSpeciesTableCount(name)];
154
                            speciesID = new String[getSpeciesTableCount(name)];
155
156
                            int counter=0; //used as an index for inserting into array
157
                            while(rec.next()){
158
                            //if record starts with name or contains part of name then
                            if(rec.getString("speciesName").contains(name) ||
159
                                    rec.getString("speciesName").equalsIgnoreCase(name)){
•
                                NAMEfound = true; //make name found true
161
162
                                String i = rec.getString("speciesID");//get id
163
                                String n = rec.getString("speciesName"); //get name
164
165
                                counter++: //increase counter
                                speciesID[counter-1] = i; //add animalID to arrayID
166
167
                                speciesName[counter-1] = n; //& add name to array
168
169
                                System.out.println(rec.getString(1));
170
                                //if name is not found
171
                             } else if(!rec.getString("speciesName").contains(name)){
172
                                NAMEfound = false; //set namefound to false
173
174
175
176
177
178
```

```
₽
                rec.close(); //closes resultSet
187
188
          }catch (SQLException e){
189
             System.out.println(e.toString());
190
191
          if(NAMEfound==true){ //-----
192
             return "nameExists";
                       //RETURNS VALUES AS 'Yes' or 'No'
193
          } else
194
195
             return "!nameExists";
196
197
```

```
public String[] getAnimalNames(){
199
             return animalName; //returns animal names
200
201
202
           public String[] getAnimalID(){
203
               return animalID; //returns animal id
204
205
206
           public String[] getSpeciesNames(){
207
               return speciesName; //returns species names
208
209
210
           public String[] getSpeciesID(){
211
               return speciesID; //returns species id
212
213
214
215
           public String[] getDescription(){
               return description; //returns animal description
216
217
218
           public String[] getSpeciesIDFK(){
219
               return speciesIDFK; //returns animal speciesID FK
220
221
222
```

```
public String doesUserExist(String username, String password) {
223
                    boolean userNameExists = false; //flag variable 1
224
                    boolean passwordMatched = false; //flag variable 2
225
226
                try{
227
                    //statement object
                    Statement statement = connection.createStatement();
228
229
                    String query = "Select * from Users";
230
                    //result object
231
                    ResultSet rec = statement.executeQuery(query);
232
                    //go through records
233
234
                    while(rec.next()){
                        //if username is found
235
                        if(username.equals(rec.getString("userName"))){
236
237
                            userNameExists = true;//set userNameExist = true
238
                        //if username exists & password is found then
239
                        if(userNameExists==true && password.equals(rec.getString("userP"
240
                                + "assword"))){
241
                            passwordMatched = true; //set passWord matched = true
242
243
244
245
                }catch (SQLException e){ //catch SQL exception
                    System.out.println(e.toString()); //display exception message
246
247
               String output = ""; //output variable
248
                //if username exists & password is matched then
249
                if(userNameExists==true && passwordMatched==true){
250
                    output = "Yes";
                                       //update output
251
                } else if(userNameExists==false){ //if username does not exist
252
253
                    output = "User does not exist!";//update output
                    //if username exists but password is wrong then
254
                } else if(userNameExists==true && passwordMatched==false){
255
                    output ="Wrong password!"; //update input
256
257
258
                return output;//returns output back
259
260
```

```
public boolean doesSpeciesExist(String speciesID){
262
263
                //Recivies parameters speciesID, speciesName
264
                 int rowsadded; //rowsadded integer
                 boolean doesExist = false; //flag varible
265
                 String result = ""; //blank string
266
267
                 try{
268
                     //creates statement object
                    Statement statement = connection.createStatement();
269
270
                    //display query (DISPLAYS speciesID only from Species table)
271
                    String displaySpeciesID = "SELECT speciesID from Species";
272
273
                    //executes display query
274
                    ResultSet rec = statement.executeQuery(displaySpeciesID);
275
276
                    while(rec.next()){//if species if found
277
                            if(speciesID.equals(rec.getString("speciesID"))){
                                doesExist = true;//set does exist to true
278
279
280
281
                     //catches SQL exception
282
                    }catch (SQLException e){System.out.println(e.toString());}
283
284
                 return doesExist;
285
286
```

```
287
            public String insertIntoAnimals(String animalID, String animalName, String
288
                    description, String speciesID){
                //Recivies parameters animalID, animalName, description, speciesID
289
290
                int rowsadded; //rowsadded integer
291
            if(doesSpeciesExist(speciesID)){
292
293
294
                try{
295
                    //creates statement object
296
                    Statement statement = connection.createStatement();
                    //insert query (SELF EXPLAINATORY)
297
                    String query = "INSERT INTO Animals VALUES "
298
                            + "('"+animalID+"','"+animalName+"','"+description+"','"+
299
300
                            speciesID+"')";
                    //executes query
301
                    rowsadded = statement.executeUpdate(query);
302
303
                    //catches SQL exception
                }catch (SQLException e){System.out.println(e.toString());}
304
305
                //returns message
306
                return "Successully added to Animals table!";
307
                return "Species does not exist!";
308
309
310
```

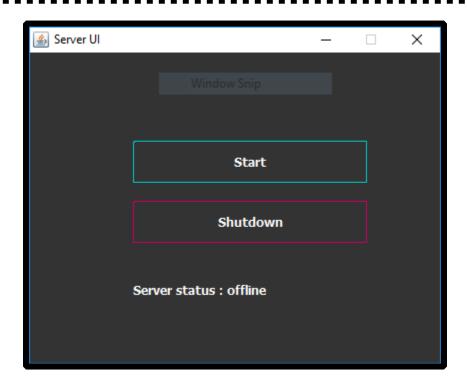
```
313
           public String insertIntoSpecies(String speciesID, String speciesName){
                //Recivies parameters speciesID, speciesName
314
315
                 int rowsadded; //rowsadded integer
                 String result = ""; //blank string
316
317
                 try{
                    //creates statement object
318
                    Statement statement = connection.createStatement();
319
320
                    //insert query (SELF EXPLAINATORY)
                    String insert = "INSERT INTO Species VALUES "
321
                            + "('"+speciesID+"', '"+speciesName+"')";
322
323
                    //display query (DISPLAYS speciesID only from Species table)
324
                    String displaySpeciesID = "SELECT speciesID from Species";
325
                    //executes display query
326
                    ResultSet rec = statement.executeQuery(displaySpeciesID);
327
328
329
                    boolean doesSpeciesExist=false; //flag varible
                    while(rec.next()){//if species if found
330
                        if(speciesID.equals(rec.getString("speciesID"))){
331
                            doesSpeciesExist = true;//set does exist to true
332
333
334
                    //species does not exist then
335
                    if(doesSpeciesExist==false){
336
                    //execute insert query
337
                    rowsadded = statement.executeUpdate(insert);
338
                    //update result message
339
                    result = "Successully added to Species table!";
340
341
                    }else
342
                    //else if it does exist then
343
                    if(doesSpeciesExist==true){
344
345
                        //update result message
                        result = "Species already exists!";
346
347
348
349
                    //catches SQL exception
350
                }catch (SQLException e){System.out.println(e.toString());}
351
                //returns result
                return result;
352
353
```

```
355
           public String deleteRecord(String animalID){
356
                int rowsdeleted; //rowsadded integer
357
                String result="";//blank string
358
                try{
359
                    //creates statement object
360
                    Statement statement = connection.createStatement();
                    //display query
361
                    String displayAnimals = "Select * from Animals";
362
                    //delete query including speciesID (input)
363
                    String deleteQuery = "DELETE FROM Animals WHERE animalID='"+animalID
364
365
366
                    //executes displat query
                    ResultSet rec = statement.executeOuery(displayAnimals);
367
368
                    boolean doesExist=false; //flag variable
369
370
                    while(rec.next()){//go through each record
                        //if animalID is found
371
372
                        if(animalID.equals(rec.getString("animalID"))){
                            doesExist = true; // set doesExist to true
373
374
375
376
                    //if animal id exists
377
378
                    if(doesExist==true){
379
                        //the execute delete query
                        rowsdeleted = statement.executeUpdate(deleteQuery);
380
381
                        //return correct message
382
                        result = "Successfully deleted from Animals table!";
383
                    } else
384
                    //if it does not exists
385
                    if(doesExist==false){
386
387
                        //return correct message
388
                        result = "Animal does not exist!";
389
390
391
                    //catches SQL exception & prints exception message
392
                }catch (SQLException e){System.out.println(e.toString());}
393
394
                //returns result
                return result;
395
396
```

```
public String processInput(String s){
398
399
              String output=""; //blank string
400
              //SEARCHING ----->
401
402
              if(s.startsWith("s~")){
403
                   //s~bob=Species
404
                   String name = s.replace("s~","");
name = name.substring(0,s.index0f("=")-2);
405
                   String table = s.substring(s.indexOf("=")+1, s.length());
406
407
408
                  output= checkSearchName(name, table);
409
               } else
410
411
              //LOGIN -----
412
              if(s.startsWith("login~")){
413
414
                  //login~username;password
415
                  s = s.replace("login~", "");
416
                  //username;password
417
                  String lusername = s.substring(0,s.indexOf(";"));
418
                  String lpassword = s.substring(s.indexOf(";")+1, s.length());
419
420
                  output = doesUserExist(lusername, lpassword);
421
422
              } else
```

```
194 //TMCEDTING
 424
                //TNSERTING -
425
                if(s.startsWith("i~")){
426
                    //i~Animals;animalID,animalName,description,speciesID
427
                    //i~Species:speciesID.speciesName
428
                    s = s.replace("i~", "");
429
430
                    //Animals;animalID,animalName,description,speciesID
431
                    if(s.startsWith("Animals")){
432
                        s = s.replace("Animals;", "");//animalID,animalName,description,speciesID
433
                        String animalID = s.substring(0,s.indexOf(","));
435
                       s = s.replace(animalID+",", "");//animalName,description,speciesID
436
438
                        String animalName = s.substring(0,s.indexOf(","));
                        s = s.replace(animalName+",",""); //description,speciesID
439
                       String description = s.substring(0,s.indexOf(","));
441
                        s = s.replace(description+",", ""); //speciesID
                       String speciesIDFK = s;
443
444
                      output = insertIntoAnimals(animalID, animalName, description, speciesIDFK);
445
446
                    }else
447
448
                    //~Species;speciesID,speciesName
449
                    if(s.startsWith("Species")){
450
                       s = s.replace("Species;", ""); //speciesID,speciesName
451
                        String speciesID = s.substring(0,s.indexOf(","));
453
                        s = s.replace(speciesID+",", "");
454
                       String speciesName = s; //speciesName
456
457
                       output = insertIntoSpecies(speciesID, speciesName);
458
459
460
               } else
```

```
//DELETING -----
464
              if(s.startsWith("d~")){
 465
466
                 //d~animalID
                  s = s.replace("d~", "");
467
468
                  //animalID
                  String animalID = s:
 470
                  output= deleteRecord(animalID);
471
              }else
472
              //SHUTDOWN---->
473
474
              if(s.equals("shutdown")){ //if request = shutdown
                  output = "shutdown"; //then return shutdown
475
476
              else //even though cannot have invalid statement but
477
                  // just returning
 478
              //INVALID STATEMENT ----->
479
480
                  output = "Invalid statement!";
481
482
483
484
              return output;
486
 487
488
```



Client.

Client UI Class

```
Filename
                               Client
2
3
               Author
                           :
                               Aysham Hameed
               Created
                               15th October 2018
 4
 5
               05
                               Windows 10
              Version
                               Netbeans IDE 8.2
 6
                               Client, contains UI & connects to Server.
7
              Desription
8
9
      package UI.Client; //contained in UI.Client package.
10
11
   p import java.awt.Color; //used to change button colors.
12
      import javax.swing.*; //used for swing components
13
      import java.net.*; //used for socket object.
14
      import java.io.*; //used of streams object.
15
      import java.util.regex.*; //used to check input
16
      import javax.swing.table.*;
17
18
      public class Client extends javax.swing.JFrame { //main class Client extends
19
20
                                                        //JFrame to display frame
21
           int xMouse; //for x mouse position
22
           int yMouse; //for y mouse position
23
24
25
          //SOCKET & STREAM OBJECTS
          Socket clientSocket;
26
          ObjectOutputStream send2SERVER = null;
27
          ObjectInputStream getfromSERVER = null;
28
29
30
          DefaultTableModel model;
                                       //GETS TABLE MODEL
31
          boolean SERVER_ONLINE = false; //FLAG VARIABLE
32
```

```
public Client() {
34
              initComponents();
               //disableSearchPanel();
35
              disableLoginPanel();
38
              //CREATING CLIENT SOCKET & STREAM OBJECTS
39
40
              try{
41
                  //creates Socket object
                 clientSocket = new Socket("localhost",7777);
42
43
                 //creates ObjectOutputStream object
44
                  send2SERVER = new ObjectOutputStream(clientSocket.getOutputStream());
45
                  //creates ObjectInputStream object
46
                  getfromSERVER = new ObjectInputStream(clientSocket.getInputStream());
47
                 jLabel17.setText("Server status : online"); //shows server status
48
                  SERVER_ONLINE = true;//sets server flag variable to true
49
              }catch (IOException e){ //catches IO Exception
50
                   System.out.println(e.toString()); //displays exception message
                   jLabel17.setText("Server status : offline"); //shows server status
51
52
                  SERVER_ONLINE = false; //sets server flag variable to false
53
54
55
```

```
public void enableAdminPanel(){
646
               //ENABLES THE ADMIN PANEL & ALL ITS COMPONENTS
                      adminPanel.show();
647
                                                         /**/
                      adminPanel.enable();
                                                          /**/
            /**/
648
649
            /**/
                                                          /**/
            /**/
                                                          /**/
650
                      jComboBox2.setEnabled(true);
            /**/
                                                          /**/
651
                      jComboBox2.setVisible(true);
                      jComboBox3.setEnabled(true);
652
            /**/
                      jComboBox3.setVisible(true);
                                                          /**/
653
            /**/
                                                          /**/
654
                      jLabel15.setEnabled(true);
                                                          /**/
655
            /**/
                      jLabel15.setVisible(true);
            /**/
                                                          /**/
                      jLabel6.setEnabled(true);
656
657
                      jLabel16.setVisible(true);
658
                      actionButton.setEnabled(true);
                      actionButton.setVisible(true);
                                                          /**/
659
660
                      logoutButton.setEnabled(true);
                      logoutButton.setVisible(true);
661
662
663
664
665
            public void disableAdminPanel(){
666
667
               //DISABLES THE ADMIN PANEL & ALL ITS COMPONENTS
                      adminPanel.hide();
                                                          /**/
668
            /**/
669
                      adminPanel.disable();
                                                          /**/
            /**/
                                                          /**/
670
                      jComboBox2.setEnabled(false);
671
            /**/
                                                          /**/
                      jComboBox2.setVisible(false);
672
            /**/
673
                      jComboBox3.setEnabled(false);
                                                          /**/
                                                          /**/
            /**/
674
                      jComboBox3.setVisible(false);
                                                          /**/
            /**/
675
                      jLabel15.setEnabled(false);
676
                      jLabel15.setVisible(false);
677
                      jLabel6.setEnabled(false);
678
                      jLabel16.setVisible(false);
679
                      actionButton.setEnabled(false);
                      actionButton.setVisible(false);
680
                      logoutButton.setEnabled(false);
681
            /**/
682
                      logoutButton.setVisible(false);
                                                      *******/
684
```

```
public void enableSearchPanel(){
686
                //ENABLES SEARCH PANEL & ALL ITS COMPONENTS
687
            /**/
688
                    searchPanel.enable(true); //enable search panel
            /**/
                    searchPanel.<del>show</del>();
                                                 //show search panel
689
            /**/
                    jLabel7.setBackground(new Color(0,202,106)); //change color of bar
690
                                                          /**/
            /**/
691
            /**/
                    jLabel9.setEnabled(true);
                                                          /**/
692
693
            /**/
                    iLabel9.setVisible(true);
                                                          /**/
            /**/
                    jComboBox1.setEnabled(true);
                                                          /**/
694
            /**/
                    jComboBox1.setVisible(true);
                                                          /**/
695
            /**/
                    jLabel10.setEnabled(true);
                                                          /**/
696
            /**/
                    iLabel10.setVisible(true);
697
                                                          /**/
698
            /**/
                    jTextField1.setVisible(true);
                                                          /**/
            /**/
                    jTextField1.setEnabled(true);
699
                                                          /**/
            /**/
                    searchButton.setEnabled(true);
700
                                                          /**/
                    searchButton.setVisible(true);
701
            /**/
                                                          /**/
                    jScrollPane1.setEnabled(true);
                                                          /**/
702
            /**/
703
            /**/
                    jScrollPane1.setVisible(true);
                                                          /**/
            704
705
706
            public void disableSearchPanel(){
707
                //DISABLES SEARCH PANEL & ALL ITS COMPONENTS
708
            /**/
                    searchPanel.enable(false); //disable search panel
709
            /**/
                    searchPanel.hide();
710
                                                 //hide search panel
            /**/
                    jLabel7.setBackground(Color.WHITE); //change color of bar
711
            /**/
                                                          /**/
712
                    jLabel9.setEnabled(false);
                                                          /**/
            /**/
713
714
            /**/
                    jLabel9.setVisible(false);
                                                          /**/
            /**/
                    jComboBox1.setEnabled(false);
                                                          /**/
715
716
            /**/
                    jComboBox1.setVisible(false);
                                                          /**/
            /**/
                    jLabel10.setEnabled(false);
                                                          /**/
717
            /**/
                    jLabel10.setVisible(false);
                                                          /**/
718
            /**/
                    iTextField1.setVisible(false);
                                                          /**/
719
            /**/
                    jTextField1.setEnabled(false);
                                                          /**/
720
            /**/
                    searchButton.setEnabled(false);
                                                          /**/
721
722
            /**/
                    searchButton.setVisible(false);
                                                          /**/
            /**/
                    jScrollPane1.setEnabled(false);
723
                                                          /**/
                    jScrollPane1.setVisible(false);
724
                                                          /**/
725
726
            3
```

```
public void enableLoginPanel(){
728
729
               //ENABLES LOGIN PANEL
           /**/
                   loginPanel.enable(true); //enable login panel
730
           /**/
                   loginPanel.show();
                                            //show login panel
731
                   jLabel8.setBackground(new Color(0,202,106));//change color of bar
732
           /**/
           /**/
                                                        /**/
733
                                                        /**/
734
           /**/
                   loginButton.setEnabled(true);
                                                        /**/
           /**/
                   loginButton.setVisible(true);
735
736
           /**/
                   jLabel1.setEnabled(true);
                                                        /**/
                   jLabel1.setVisible(true);
           /**/
                                                        /**/
737
738
           /**/
                   jTextField2.setEnabled(true);
                                                        /**/
                   jTextField2.setVisible(true);
           /**/
                                                        /**/
739
           /**/
                   jPasswordField1.setEnabled(true);
                                                        /**/
740
           /**/
                   jPasswordField1.setVisible(true);
                                                        /**/
741
           742
743
           }
744
           public void disableLoginPanel(){
745
               //DISABLING LOGIN PANE
746
           /**/
                   loginPanel.enable(false); //disble login panel
747
           /**/
                   loginPanel.hide();
                                              //hide login panel
748
                   jLabel8.setBackground(Color.WHITE);//change color of bar
           /**/
749
                                                        /**/
           /**/
750
                                                        /**/
           /**/
                   loginButton.setEnabled(false);
751
                   loginButton.setVisible(false);
           /**/
                                                        /**/
752
           /**/
                   jLabel1.setEnabled(false);
                                                        /**/
753
           /**/
                   jLabel1.setVisible(false);
                                                        /**/
754
                   jTextField2.setEnabled(false);
                                                        /**/
755
           /**/
                   jTextField2.setVisible(false);
           /**/
                                                        /**/
756
                   jPasswordField1.setEnabled(false);
757
           /**/
                                                        /**/
           /**/
                   jPasswordField1.setVisible(false);
                                                        /**/
758
759
           }
760
```

```
private void buttonExitMouseEntered(java.awt.event.MouseEvent evt) {
762
763
               buttonExit.setBackground(new Color(232,17,35)); //hover in effect
764
765
           private void buttonExitMouseExited(java.awt.event.MouseEvent evt) {
766
              buttonExit.setBackground(new Color(255,255,255)); //hover out effect
767
768
769
770
           private void buttonExitMouseClicked(java.awt.event.MouseEvent evt) {
                //An array of options to be provided for user.
771
               String options [] = {"Exit", "Minimise", "Cancel"};
772
                //Asks user which option to pick, and store as an int.
773
                int option = JOptionPane.showOptionDialog(null, "Are you sure you want"
774
                       + " to quit?", "Exit",0 ,0, null, options, options[0]);
775
776
                if(option==0){
                                        //if user pickts the first choice (EXIT)
777
778
                   System.exit(0);
                                        //then exit the system
779
                }else
                                        //else
780
781
               if(option==1){
                                        //is user picks the second choice (MINIMISE)
                    setState(ICONIFIED);//then minimise.
782
783
784
```

```
private void jMenuBar1MousePressed(java.awt.event.MouseEvent evt) {
794
795
               xMouse = evt.getX(); //gets the x postion of mouse on header
               yMouse = evt.getY(); //gets the y position of mouse on header
796
797
798
           private void jMenuBar1MouseDragged(java.awt.event.MouseEvent evt) {
799
800
               int x = evt.getXOnScreen(); //gets x position of mouse on screen
801
               int y = evt.getYOnScreen(); //gets y position of mouse on screen
802
               this.setLocation(x-xMouse, y-yMouse); //postions of mouse on header -
803
                                                    //postions of mouse on screen
804
805
           private void searchButtonMouseEntered(java.awt.event.MouseEvent evt) {
806
                //WHEN MOUSE IS ENTERED (HOVERING EFFECT)
807
808
                searchButton.setBackground(new Color(204, 0, 102));
                searchButton.setForeground(Color.black);
809
810
811
812
           private void searchButtonMouseExited(java.awt.event.MouseEvent evt) {
813
                //WHE MOUSE LEVEAS BUTTON (HOVERING EFFECT)
814
                searchButton.setBackground(new Color(238,238,238));
815
                searchButton.setForeground(Color.black);
816
817
818
           private void searchMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
819
               disableLoginPanel(); //disable login panel
               enableSearchPanel();//enable search panel
820
821
822
823
           private void loginMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
824
825
                enableLoginPanel(); //enable login panel
826
                disableSearchPanel();//disable search panel
827
```

```
829
             private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
  830
                 model = (DefaultTableModel)jTable1.getModel(); //type of jtable model
 831
  832
                 //which table the user chose
  833
                 String dataLocation = (String) jComboBox1.getSelectedItem();
  834
  835
                 //what they want to search
  836
                 String name = jTextField1.getText();
  837
  838
                 //the regex must only have letters and must be >=3
                 boolean correcInput = Pattern.matches("[a-zA-Z]{3,}", name);
  839
  840
                 //if input is wrong
  841
  842
                 if(correcInput==false){
  843
  844
                     //display a message
                     JOptionPane.showMessageDialog(null, "Must be text & 3 characters "
 845
                             + "long!", "Error", JOptionPane. ERROR_MESSAGE);
 846
 847
                 }else{
 848
                     //else if its right
 849
 850
                     //s~bob=Species (COMPILE STRING TO BE SEND TO SERVER)
                      String input = "s~"+name+"="+dataLocation;
 851
 852
853
                      //flag variable
                      boolean userExists=true:
                      if(SERVER_ONLINE){ //if server is online only then
855
                      //while(true){//-----
 856
857
                      /* | */
                               try{
                      /* | */
                                    //asking server does user exist?
858
 859
                      /* | */
                                    send2SERVER.writeObject(input);
 860
                      /* | * /
                      /* | */
861
                                    //getting answer from server
                                    String messageFromServer = (String) getfromSERVER.readObject();
 862
                       /* | * /
 863
                       /* | * /
```

```
864
  865
                     /* | */
  866
                                 /*|*/ //if name exists then get array data/*|*/
                                 867
                     /* | */
                                  if(messageFromServer.equals("nameExists")){
  868
  869
                                         //gets the name of table from server
                     /* | */
                                 /* | */
                                          String nameOfTable = (String) getfromSERVER.readObject();
                     /* | */
                                 /*|*/
  871
  872
                                 /*|*/
                                          //if Animal table is seleected
                     /* | */
                                 /* | */
                                         /*|*/ if(nameOfTable.equals("Animals")){
                     /*|*/
                                 /*|*/
                                         /*|*/
  874
                                                //GETTING DATA FROM SERVER
                     /* | * /
                                 /*|*/
                                         /*|*/
  875
                                                  String animalID[]=(String []) getfromSERVER.readObject();
                     /* | */
                                /* | */
                                         /* | */
                                                 String animalNames[]=(String[]) getfromSERVER.readObject();
                                                  String description[]=(String[]) getfromSERVER.readObject();
  877
                     /* | */
                                 /* | */
                                         /* | * /
  878
                                 /* | * /
                                          /* | * /
                                                  String speciesIDFK[]=(String[]) getfromSERVER.readObject();
                     /* | */
                                  /* | */
                                          /*|*/
                                          /*|*/
                     /* | * /
                                 /* | */
  880
                     /*|*/
                                 /* | */
  881
                                          /* | * /
                                                  //make proper Animal columns
                     /* | */
                                 /* | */
                                          /* | */
                                                  makeAnimalColumns();
                     /* | */
                                 /* | */
                                          /* | */
                                                  model = (DefaultTableModel) jTable1.getModel();
  883
                                 /*|*/
                                          /*|*/
  884
                                                  for (int x=0; x<animalID.length; x++){</pre>
                     /*|*/
                                 /* | */
                                          /* | */
                                                  model.addRow(new Object[]{animalID[x], animalNames[x],description[x],speciesIDFK[x]});
                                 /* | */
                                          /* | */
  886
                                          /*|*/
  887
                     /* | * /
                     /* | * /
                                 /* | */
  889
                                 /* | */
                                 /* | * /
                                                  //GETTING DATA FROM SERVER
  890
                     /* | * /
                                 /*||*/
                                          /*|*/if(nameOfTable.equals("Species")){
                                         /* | */
  892
                                                   String speciesID[]=(String[]) getfromSERVER.readObject();
                                 /*|*/
                                          /*|*/
  893
                                                   String speciesNames[]=(String[]) getfromSERVER.readObject();
                     /*|*/
                                 /* | */
                                          /* | */
                                                   makeSpeciesColumns();
                                 /* | */
                                          /* | */
                                                  model = (DefaultTableModel) jTable1.getModel();
  895
                                 /*|*/
                                          /*|*/
                                                  for (int x=0; x<speciesID.length; x++){</pre>
  896
                                         /*|*/
  897
                     /* | * /
                                 /* |*/
                                                      model.addRow(new Object[]{speciesID[x], speciesNames[x]});
                                         /* |*/
  898
                                         /* | * / 3
  899
                     /* | */
                                  900
  901
  902
                                 } else
                              ------
```

```
904
905
                 /* | */
                           /*|*/ //if name does not exists then display message/*|*/
906
                 /* | */
                           /***********************************/
907
                 /* | */
                           /*|*/ if(messageFromServer.equals("!nameExists")){
                 /* | */
                           /* | */
                                 userExists = false; //set user exists to false
 909
                 /* | * /
                           /*|*/
                                   if(userExists==false){//and display message
                 /*|*/
                           /* | */
910
                                      JOptionPane.showMessageDialog(null, "Name does"
911
                 /* | * /
                           /* | * /
                                           + " not exist!","Unknown name!",
                           /*|*/
912
                 /* | */
                                             JOptionPane.ERROR_MESSAGE);
913
                 /* | */
                           /* | */
914
                           /*|*/ }
                 /* | */
                            915
                 /* | */
916
917
                 /* | * /
                         } catch (IOException e){
918
                 /* | */
                            System.out.println(e.toString());
919
                 /* | */
                            System.exit(1):
                 /*|*/
920
                         }catch (ClassNotFoundException e){
                 /* | */
921
                            System.out.println(e.toString());
922
                 /*|*/
923
                 /*|*///}
924
925
                 /*|*/} else {
                 /*|*/JOptionPane.showMessageDialog(null, "Server is offline!", "Unreachable host!", JOptionPane.ERROR_MESSAGE);
926
927
                  928
929
930
```

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    // model.setRowCount(0);
    //BASE ON SELECTION IN COMBOBOX MAKE CORRECT COLUMNS
    if(jComboBox1.getSelectedIndex()==0){makeAnimalColumns();} else
    if(jComboBox1.getSelectedIndex()==1){makeSpeciesColumns();}
}
```

```
.....................................
  946
             private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
  947
                  //FLAG VARIABLES-----
  948
                 boolean usernameFieldValid = true;
  949
                 boolean passwordFieldValid = true;
  950
  951
                 //IF TEXTFIELS ARE EMPTY THEN
                 if(jTextField2.getText().equals("") || jTextField2.getText().length()<3){
  952
  953
                     usernameFieldValid = false; //SET FALG TO FALSE
  954
                     //SHOW MESSAGE PLEASE ENTER A VALID USERNAME
  955
                     JOptionPane.showMessageDialog(null, "Please enter a valid username!"
  956
  957
                             + " username must at least be 3 characters long",
  958
                             "Invalid username", JOptionPane.ERROR_MESSAGE);
  959
                 } else //ELSE
  960
  961
                 //IF PASSWORD FIELD IS EMPTY THEN
  962
                 if(jPasswordField1.getText().equals("") || jPasswordField1.getText().length()<5){
  963
  964
                     passwordFieldValid = false;//SET FLAH TO FALSE
 965
                     //SHOW MESSAGE PLEASE ENTER A VALID PASSWORD
  966
                     JOptionPane.showMessageDialog(null, "Please enter a valid password!
  967
  968
                             + "password must atleast be 5 characters long",
                             "Invalid password", JOptionPane. ERROR_MESSAGE);
  969
  970
                 String username=""; //EMPTY VARIABLE USERNAME
                 String password=""; //EMPTY VARIBALE PASSWORD
973
                 //if textfields are not empty then
  974
  975
                 if(usernameFieldValid==true && passwordFieldValid==true){
  976
                     username = jTextField2.getText(); //GET TEXT FROM USERNAME FIELD
                     password= jPasswordField1.getText();//GET PASSWORD TOO
  977
  978
  979
                     //check if server is online
                     if(SERVER_ONLINE){
  980
  981
                         try{
  982
                             ///login~username;password(SENDING TO SERVER)
                             send2SERVER.writeObject("login~"+username+";"+password);
 983
  984
  985
                             //getting response from server
                             String response = (String)getfromSERVER.readObject();
  986
  987
  988
                             //if response is yes (USER EXISTS) then
                             if(response.equals("Yes")){ //show welcome message &
  989
                                 JOptionPane.showMessageDialog(null, "Welcome! "+username);
  990
```

```
991
                                 jLabel1.setText("You are logged in as, "+username);
                                 enableAdminPanel();
992
993
                                 disableLoginPanel();
994
                                 disableSearchPanel();
995
996
                             } else{
  997
                                 //showing response to user
  998
                                 JOptionPane.showMessageDialog(null, response, "Unknown", JOptionPane.ERROR_MESSAGE);
  999
                             //CATCHES BOTH EXCEPTION
 1000
                         }catch (IOException e){
 1002
                             System.out.println(e.toString());
1003
                         } catch (ClassNotFoundException e){
1004
                             System.out.println(e.toString());
1005
1006
1007
                     }else { //ELSE IF SERVER IS OFFLINE THEN PRINT CORRECT MESSAGE
                         JOptionPane.showMessageDialog(null, "Server is offline!", "Unreachable host!", JOptionPane.ERROR_MESSAGE);
1008
1009
1010
1011
1012
                 //USERNAME FIELD IS FOCUSED
1013
                 jTextField2.requestFocusInWindow();
1014
```

```
private void logoutButtonActionPerformed(java.awt.event.ActionEvent evt) {
    jLabel1.setText("You are not logged in."); //CHANGE LABEL
    disableAdminPanel(); //DISBALES ADMIN AND
    disableLoginPanel();// LOGIN PANEL
    enableSearchPanel();// & SHOWS SEARCH PANEL
}
```

```
1024
             private void logoutButtonMouseEntered(java.awt.event.MouseEvent evt) {
                 //HOVERING EFFECT FOR BUTTON WHEN ENTERS
1025
                 logoutButton.setContentAreaFilled(true);
1026
                 logoutButton.setBackground(new Color(204, 0, 102));
1027
1028
                 logoutButton.setForeground(Color.black);
1029
1030
1031
             private void logoutButtonMouseExited(java.awt.event.MouseEvent evt) {
1032
                //HOVERING EFFECT FOR BUTTON WHEN EXITS
1033
                logoutButton.setContentAreaFilled(false);
1034
1035
                logoutButton.setForeground(Color.white);
1036
1037
            private void actionButtonMouseEntered(java.awt.event.MouseEvent evt) {
1038
                     //HOVERING EFFECT FOR BUTTON WHEN ENTERS
1039
                 actionButton.setBackground(new Color(0, 202, 106));
1040
                 actionButton.setForeground(Color.black);
1041
1042
1043
1044
            private void actionButtonMouseExited(java.awt.event.MouseEvent evt) {
1045
                 //HOVERING EFFECT FOR BUTTON WHEN EXITS
                 actionButton.setBackground(new Color(238,238,238));
1046
                 actionButton.setForeground(Color.black);
1047
1048
1049
             private void loginButtonMouseEntered(java.awt.event.MouseEvent evt) {
1050
                     //HOVERING EFFECT FOR BUTTON WHEN ENTERS
1051
                 loginButton.setBackground(new Color(0, 202, 106));
1052
1053
                 loginButton.setForeground(Color.black);
1054
1055
             private void loginButtonMouseExited(java.awt.event.MouseEvent evt) {
1056
                 //HOVERING EFFECT FOR BUTTON WHEN EXITS
1057
                 loginButton.setBackground(new Color(238,238,238));
1058
                 loginButton.setForeground(Color.black);
1059
1060
```

```
private void actionButtonActionPerformed(java.awt.event.ActionEvent evt) {
 1062
 1063
                 //GETS ACTION TABLE FROM COMBOBOX
                 String action = (String) jComboBox2.getSelectedItem();
1064
                 //GETS SELECT TABLE FROM COMBOBOX
 1065
1066
                 String table = (String) jComboBox3.getSelectedItem();
 1067
1068
                 String requestServer = "";
 1069
                 //CHECK STATEMENT 1 (INSERT)-----
1070
                 if(action.equals("Insert")){
1071
                     if(table.equals("Animals")){    //IF ANIMAL TABLE IS SELECTED
 1072
                      //i~Animals;animalID,animalName,description,speciesID
 1073
1074
                      //INPUT FIELDS FOR ANIMAL
                      String animalID = JOptionPane.showInputDialog(null, "Animal ID:");
1075
                      String animalName = JOptionPane.showInputDialog(null, "Animal name:");
 1076
                      String description = JOptionPane.showInputDialog(null, "Description: ");
 1077
                      String speciesIDFK = JOptionPane.showInputDialog(null, "Species ID FK");
 1078
                      requestServer="i~Animals;"+animalID+","+animalName+","+description+","
1079
                              +speciesIDFK;
 1080
                     }else
1081
 1082
                     if(table.equals("Species")){ //IF SPECIES TABLE IS SELECTED
1083
                     //i~Species;speciesID,speciesName
1084
                      //INPUT FIELDS FOR SPECIES
 1085
                      String speciesID = JOptionPane.showInputDialog(null, "Species ID:");
1086
                      String speciesName = JOptionPane.showInputDialog(null, "Species name:");
 1087
1088
                      requestServer="i~Species;"+speciesID+","+speciesName;
 1089
                     3
1090
                 } else
1091
1092
                 //CHECK STATEMENT 2 (DELETE)-----
                 //if Delete action is selected
1093
1094
                 if(action.equals("Delete")){
                     //then of Animal table is selected
1095
1096
                     if(table.equals("Animals")){
                         String animalID = JOptionPane.showInputDialog(null, "Animal ID");
1097
 1098
                         ////d~animalID
                         requestServer="d~"+animalID;
1099
 1100
```

```
1102
                     //if species table is selected (CAN NOT DELETE)
                    if(table.equals("Species")){
1103
1104
                        JOptionPane.showMessageDialog(null, "Can not delete from Species.");
1105
1106
                if(!requestServer.equals("")) //IF IT IS NOT BLANKK
1108
                     try{ //SEND REQUEST TO SERVER
1109
1110
                         send2SERVER.writeObject(requestServer);
1111
                         String response = (String) getfromSERVER.readObject();
1112
                         JOptionPane.showMessageDialog(null, response);
 1113
                         //CATCHING BOTH EXCEPTIONS
1114
                     }catch (IOException e){
 1116
                         System.out.println(e.toString());
1117
                     }catch (ClassNotFoundException e){
1118
                         System.out.println(e.toString());
1119
1120
1121
```

```
private void formWindowActivated(java.awt.event.WindowEvent evt) {
1123
                         //when the form/window is activated and focused
1124
1125
                 try{
1126
                         Socket testSocket = new Socket(); //create new socket object
                         //try connection to server
1127
1128
                         testSocket.connect(new InetSocketAddress("localhost", 7777),10);
1129
                         SERVER_ONLINE = true; //and make SERVER_ONLINE = true
1130
                         System.out.println("Server is active!");
                         } catch (IOException e){ //catch exception
1131
                             System.out.println(e.toString()); //print exception
1132
1133
                             SERVER_ONLINE = false;//and make SERVER_ONLINE = false
1134
                         }
1135
                 if(SERVER ONLINE==true){ //if SERVER ONLINE THEN
1136
                     jLabel17.setText("Server status : online"); //DISPLAY CORRECT
1137
                 } else jLabel17.setText("Server status : offline");//STRING
1138
1139
1140
1141
1142
```

```
private void aboutItemActionPerformed(java.awt.event.ActionEvent evt) {

//displays about dialog
String display = "Developer : alin\nVersion : 1.0";

JOptionPane.showMessageDialog(null, display, "About",

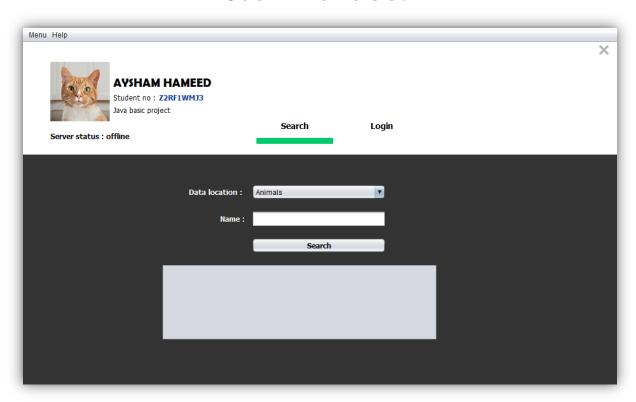
JOptionPane.PLAIN_MESSAGE);

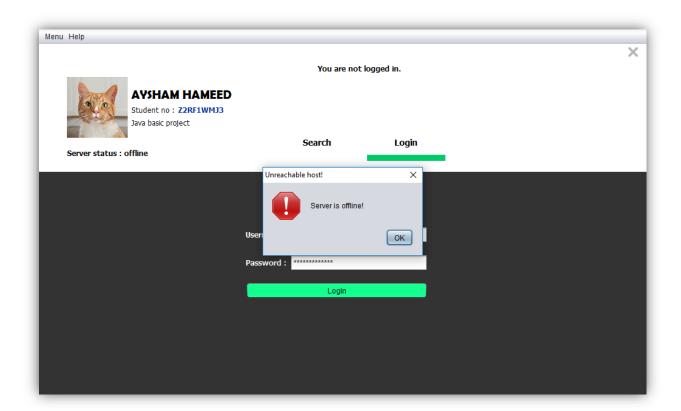
}
```

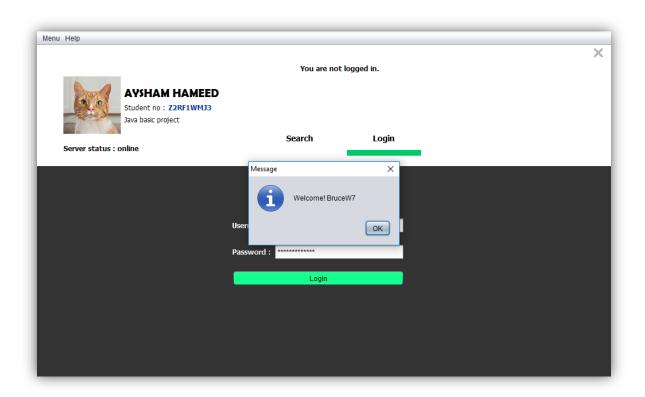
```
1152
             public void makeAnimalColumns(){
1153
1154
                  model = (DefaultTableModel) jTable1.getModel(); //GET TABLE MODEL
1155
                                             //CLEAR ALL COLUMNSS
                  model.setColumnCount(0);
1156
                  model.setRowCount(0);
                                              //CLEAR ALL ROWS
1157
                     //MAKE THESE COLUMNS
1158
                     String columns[] = {"Animal ID", "Animal name", "Description", "Species ID"};
                     for (int x=0; x<columns.length; x++){</pre>
1160
                     model.addColumn(columns[x]); //DISPLAY THE COLUMNS
1161
1162
1163
1164
             public void makeSpeciesColumns(){
1165
                  model = (DefaultTableModel) jTable1.getModel();//GET TABLE MODEL
1166
                 model.setColumnCount(0); //CLEAR ALL COLUMNSS
1167
                 model.setRowCount(0); //CLEAR ALL ROWS
1168
                     //MAKE THESE COLUMNS
                     String columns[] = {"Species ID", "Species name"};
1169
                     for (int x=0; x<columns.length; x++){</pre>
1171
                     model.addColumn(columns[x]);//DISPLAY THE COLUMNS
1172
1173
```

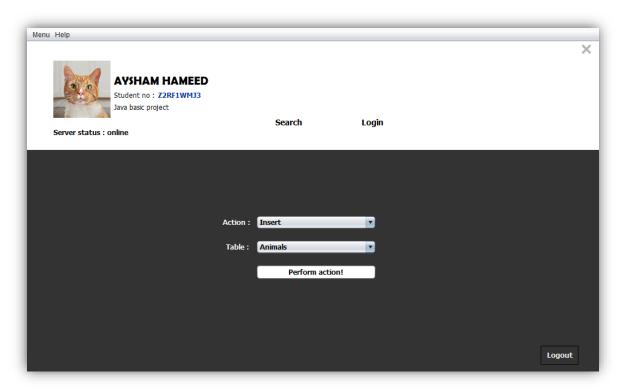
```
1175
             public static void main(String args[]) {
1176
                 /* Set the Nimbus look and feel */
1177
                 Look and feel setting code (optional)
     ф
1198
                 /* Create and display the form */
1199
                 java.awt.EventQueue.invokeLater(new Runnable() {
                     public void run() {
 ₩.
                         new Client().setVisible(true);
1202
1203
1204
                 });
1205
1206
```

User Interface.









Project Structure.

