# Z2RF1WMJ3

## JAVA BASIC

**OCTOBER 18, 2018**

***JAVA BASIC PROJECT*** *– CLIENT, SERVER & DATABASE PROGRAMMING*

# Table of Contents

# 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?

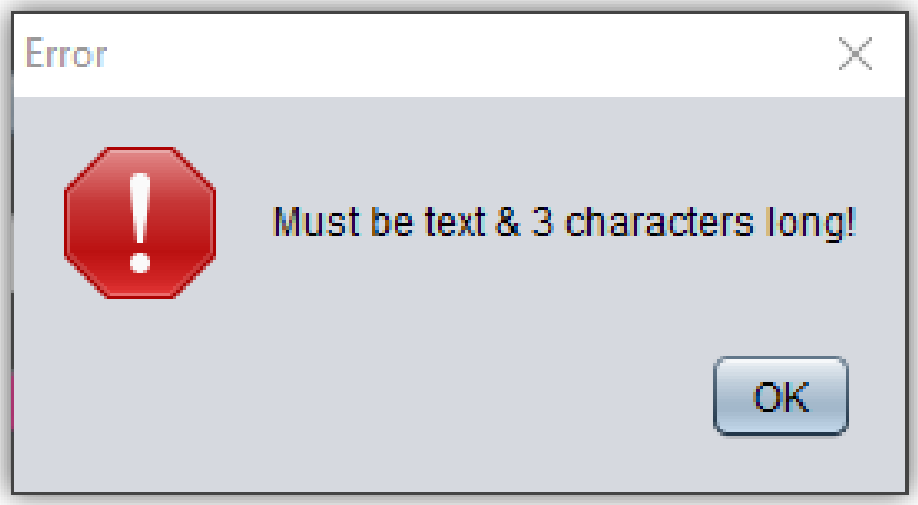
To open the program the following are the instructions:

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.

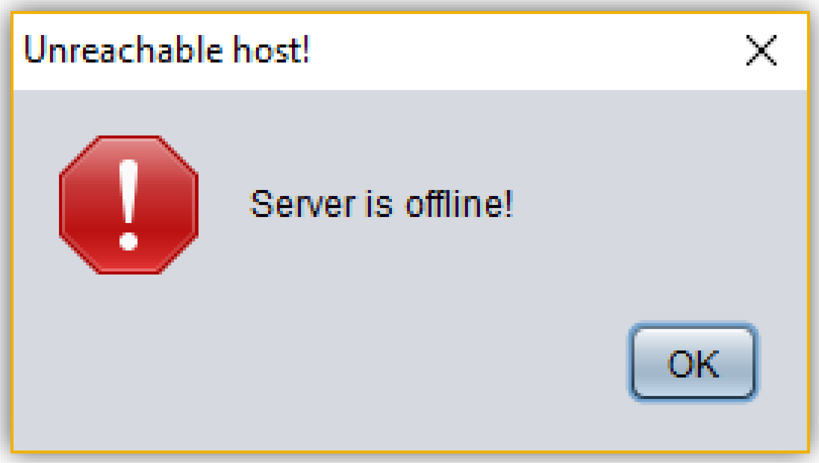
Example 1


Error

Must be text & 3 characters long!

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

Example 2

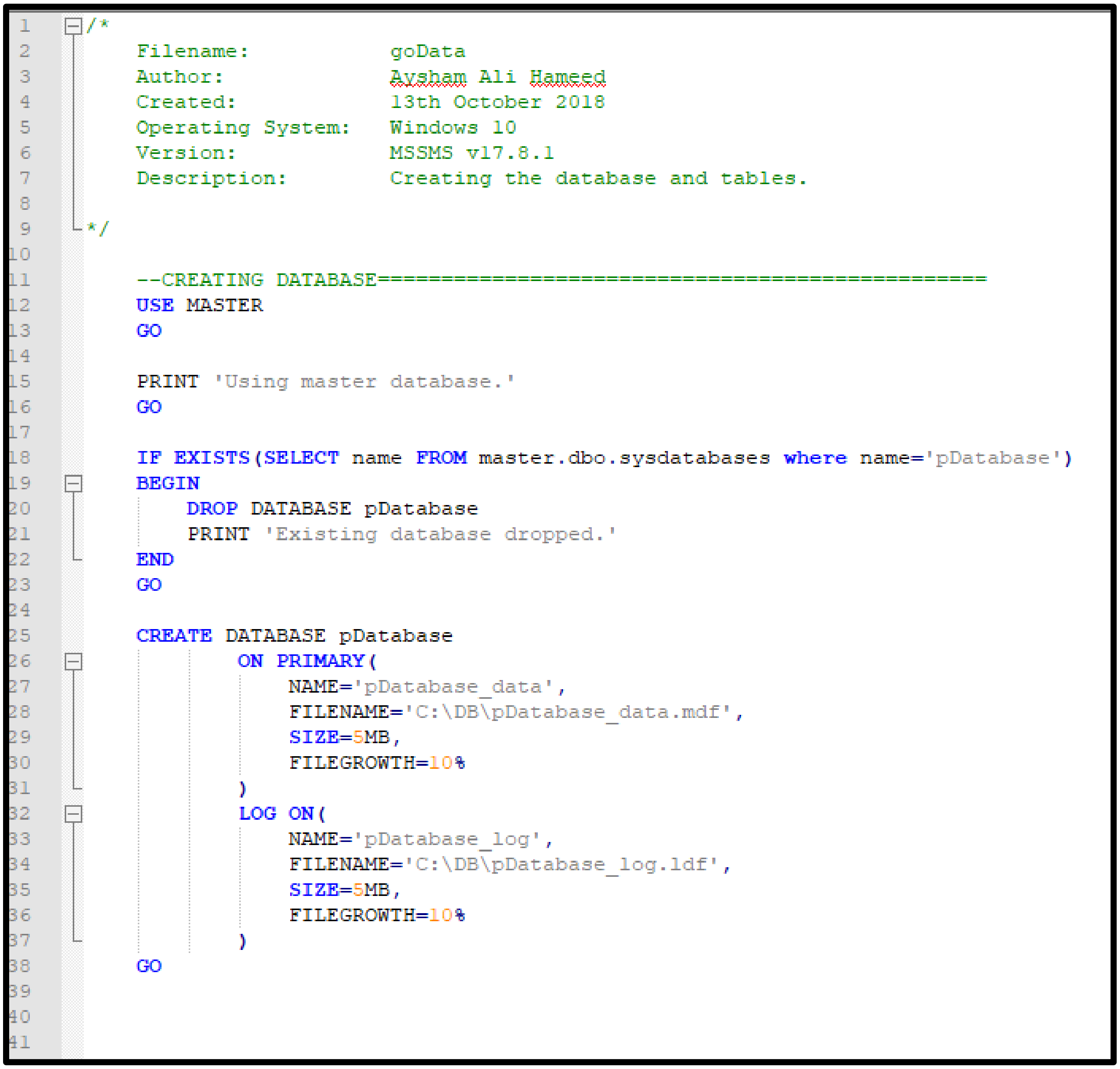

Unreachable host!

Server is offline!

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

# Creating the database.

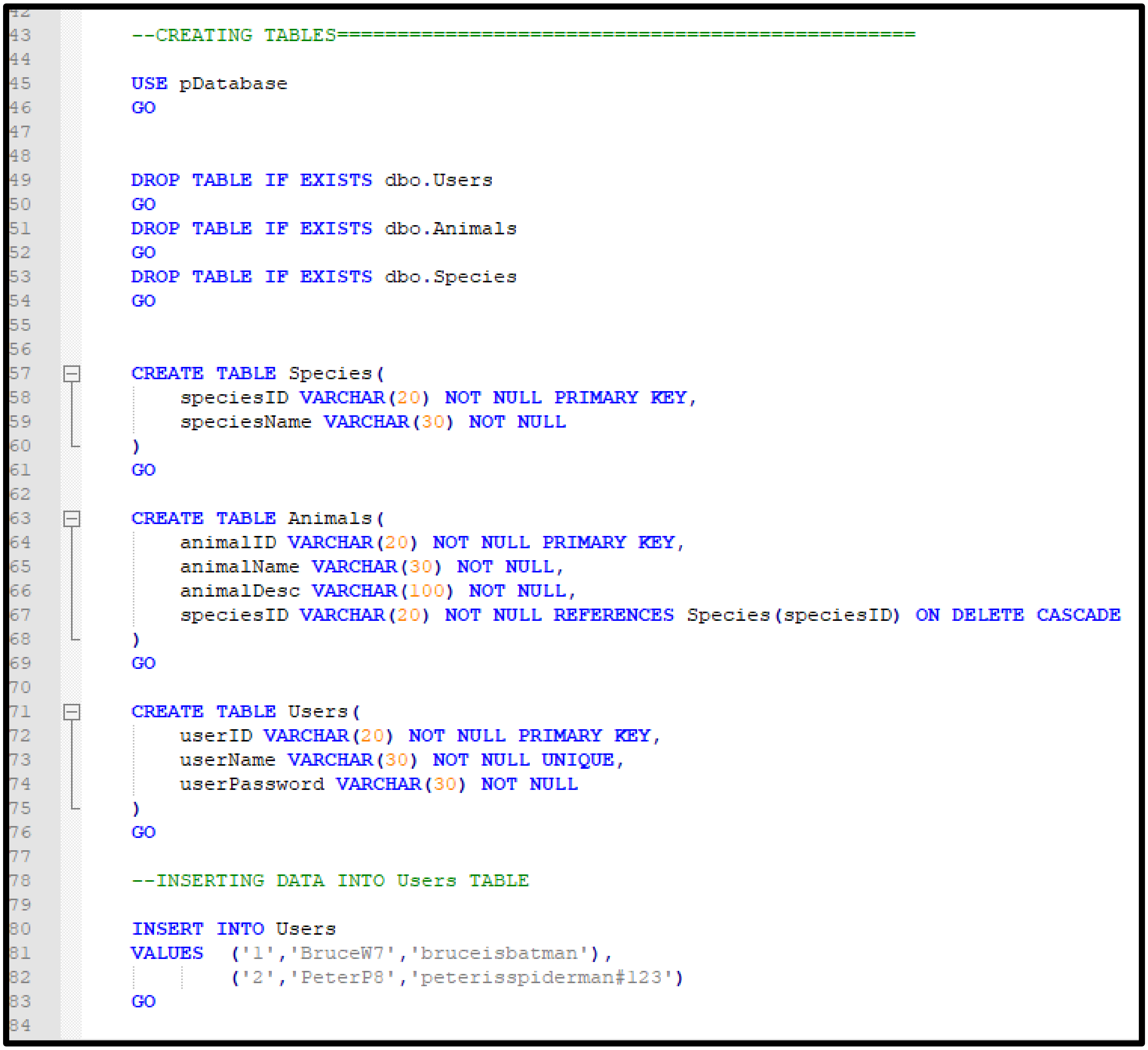
**This SQL query creates a database.**



```sql
/*
	Filename:		goData
	Author:			Aysham Ali Hameed
	Created:		13th October 2018
	Operating System:	Windows 10
	Version:		MSSMS v17.8.1
	Description:		Creating the database and tables.

*/


	--CREATING DATABASE================================================
	USE MASTER
	GO

	PRINT 'Using master database.'
	GO

	IF EXISTS(SELECT name FROM master.dbo.sysdatabases where name='pDatabase')
	BEGIN
		DROP DATABASE pDatabase
		PRINT 'Existing database dropped.'
	END
	GO

	CREATE DATABASE pDatabase
			ON PRIMARY(
				NAME='pDatabase_data',
				FILENAME='C:\DB\pDatabase_data.mdf',
				SIZE=5MB,
				FILEGROWTH=10%
			)
			LOG ON(
				NAME='pDatabase_log',
				FILENAME='C:\DB\pDatabase_log.ldf',
				SIZE=5MB,
				FILEGROWTH=10%
			)
	GO
```

# Creating tables.

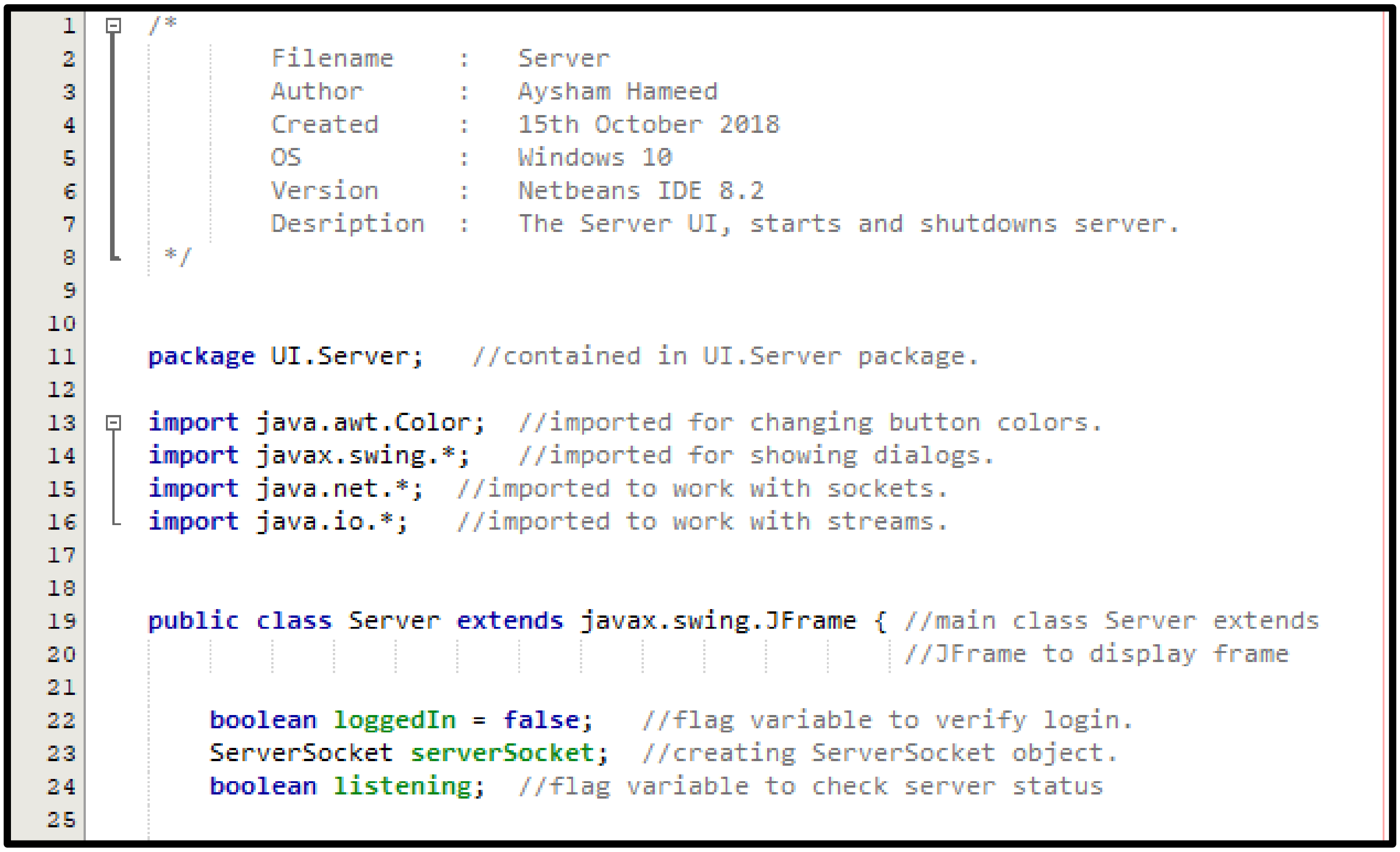
**Creates tables inside the database.**



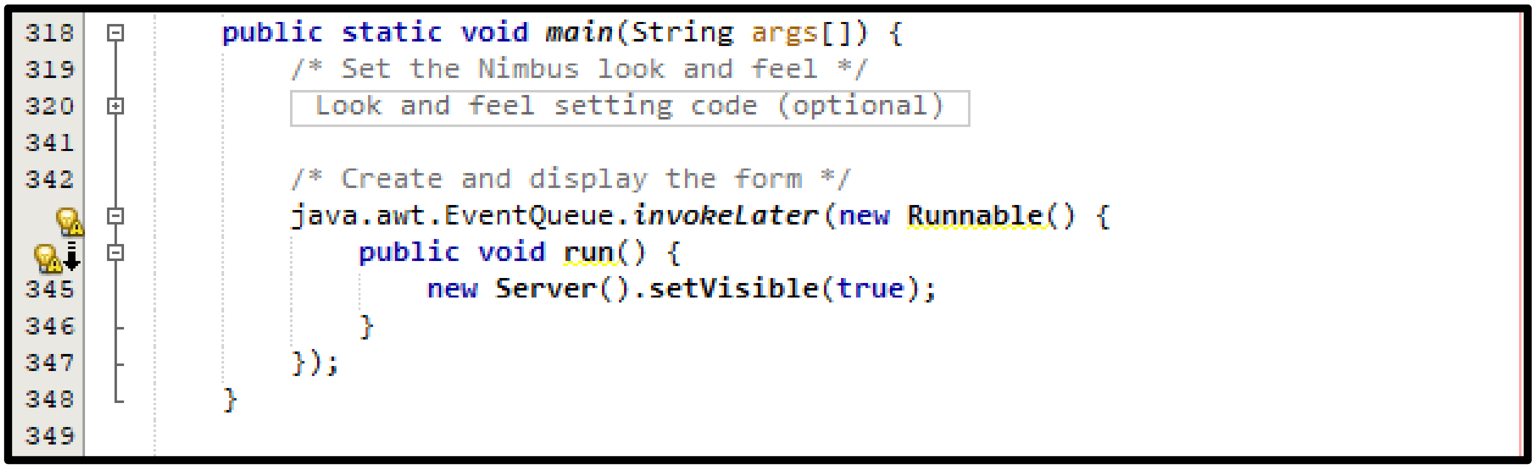
```
--CREATING TABLES==================================================

USE pDatabase
GO


DROP TABLE IF EXISTS dbo.Users
GO
DROP TABLE IF EXISTS dbo.Animals
GO
DROP TABLE IF EXISTS dbo.Species
GO


CREATE TABLE Species(
    speciesID VARCHAR(20) NOT NULL PRIMARY KEY,
    speciesName VARCHAR(30) NOT NULL
)
GO

CREATE TABLE Animals(
    animalID VARCHAR(20) NOT NULL PRIMARY KEY,
    animalName VARCHAR(30) NOT NULL,
    animalDesc VARCHAR(100) NOT NULL,
    speciesID VARCHAR(20) NOT NULL REFERENCES Species(speciesID) ON DELETE CASCADE
)
GO

CREATE TABLE Users(
    userID VARCHAR(20) NOT NULL PRIMARY KEY,
    userName VARCHAR(30) NOT NULL UNIQUE,
    userPassword VARCHAR(30) NOT NULL
)
GO

--INSERTING DATA INTO Users TABLE

INSERT INTO Users
VALUES  ('1','BruceW7','bruceisbatman'),
        ('2','PeterP8','peterisspiderman#123')
GO
```

# Server.

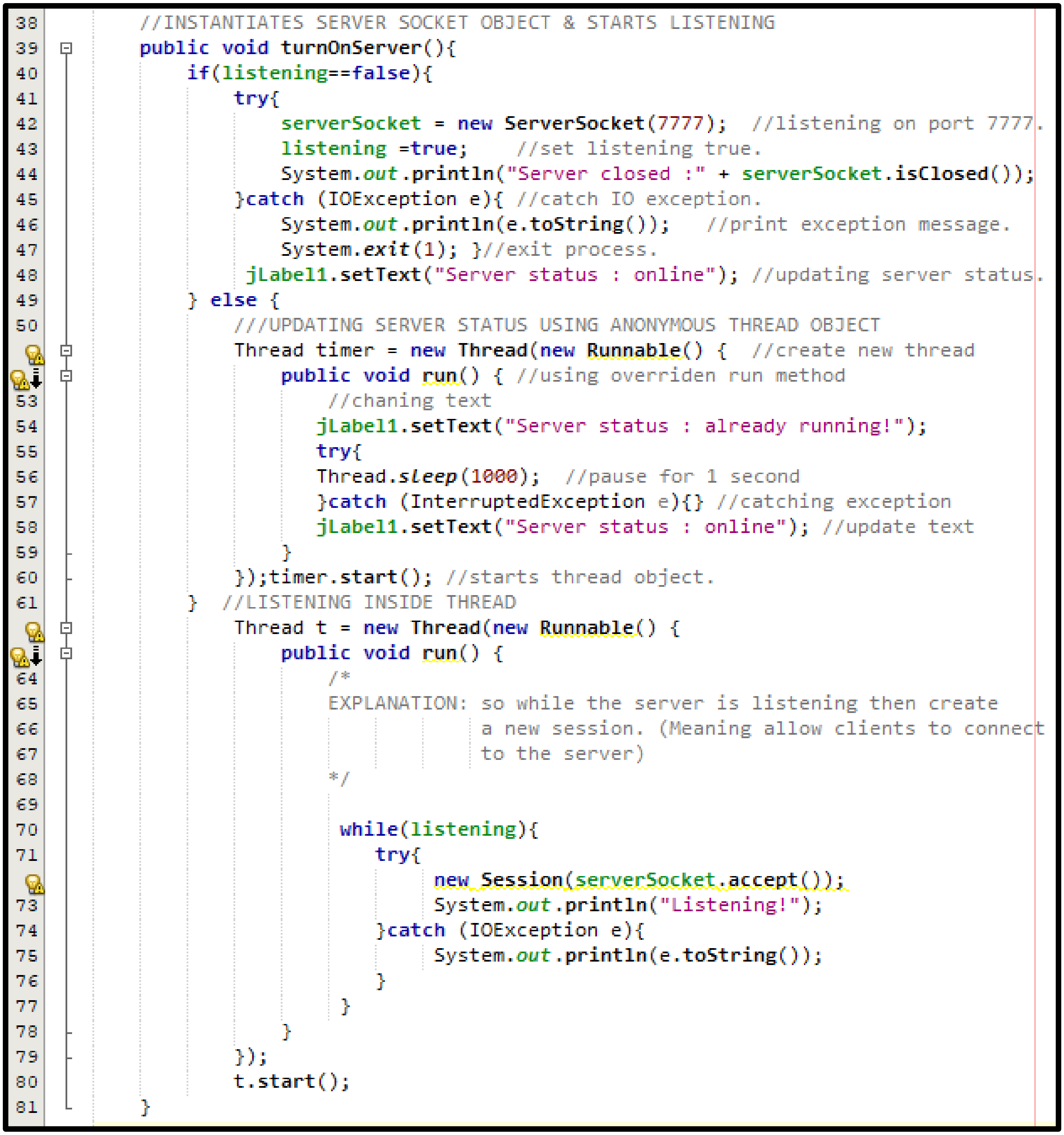
## Server UI Class

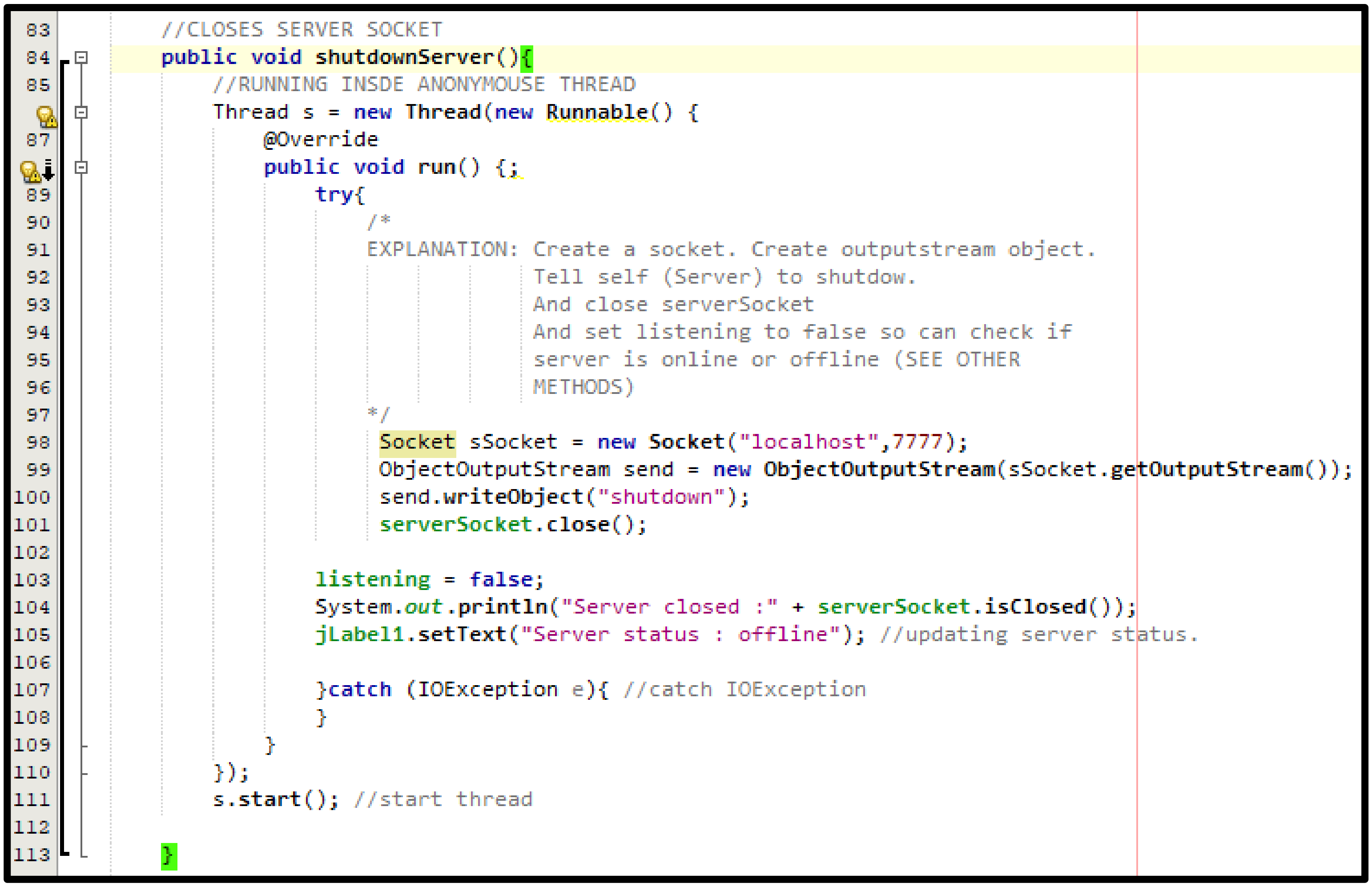


```
/*
        Filename    :   Server
        Author      :   Aysham Hameed
        Created     :   15th October 2018
        OS          :   Windows 10
        Version     :   Netbeans IDE 8.2
        Desription  :   The Server UI, starts and shutdowns server.
 */


package UI.Server;   //contained in UI.Server package.

import java.awt.Color;  //imported for changing button colors.
import javax.swing.*;   //imported for showing dialogs.
import java.net.*;  //imported to work with sockets.
import java.io.*;   //imported to work with streams.


public class Server extends javax.swing.JFrame { //main class Server extends
                                                 //JFrame to display frame

    boolean loggedIn = false;   //flag variable to verify login.
    ServerSocket serverSocket;  //creating ServerSocket object.
    boolean listening;  //flag variable to check server status

```

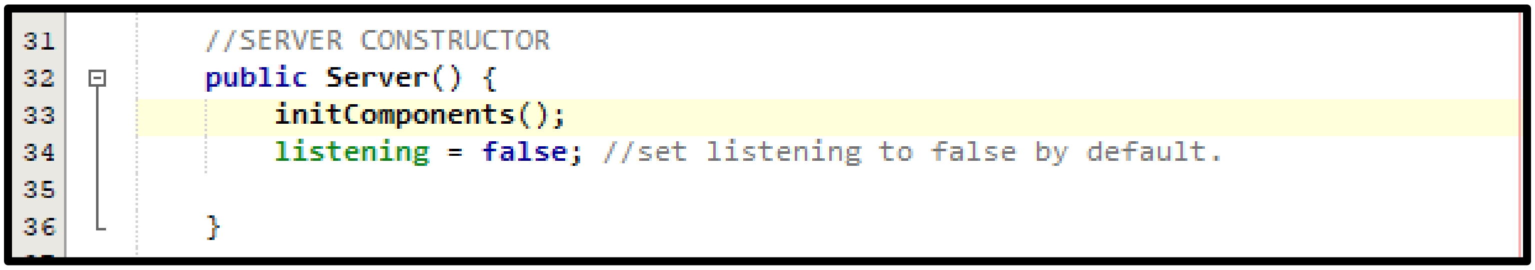


```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Server().setVisible(true);
            }
        });
    }

```

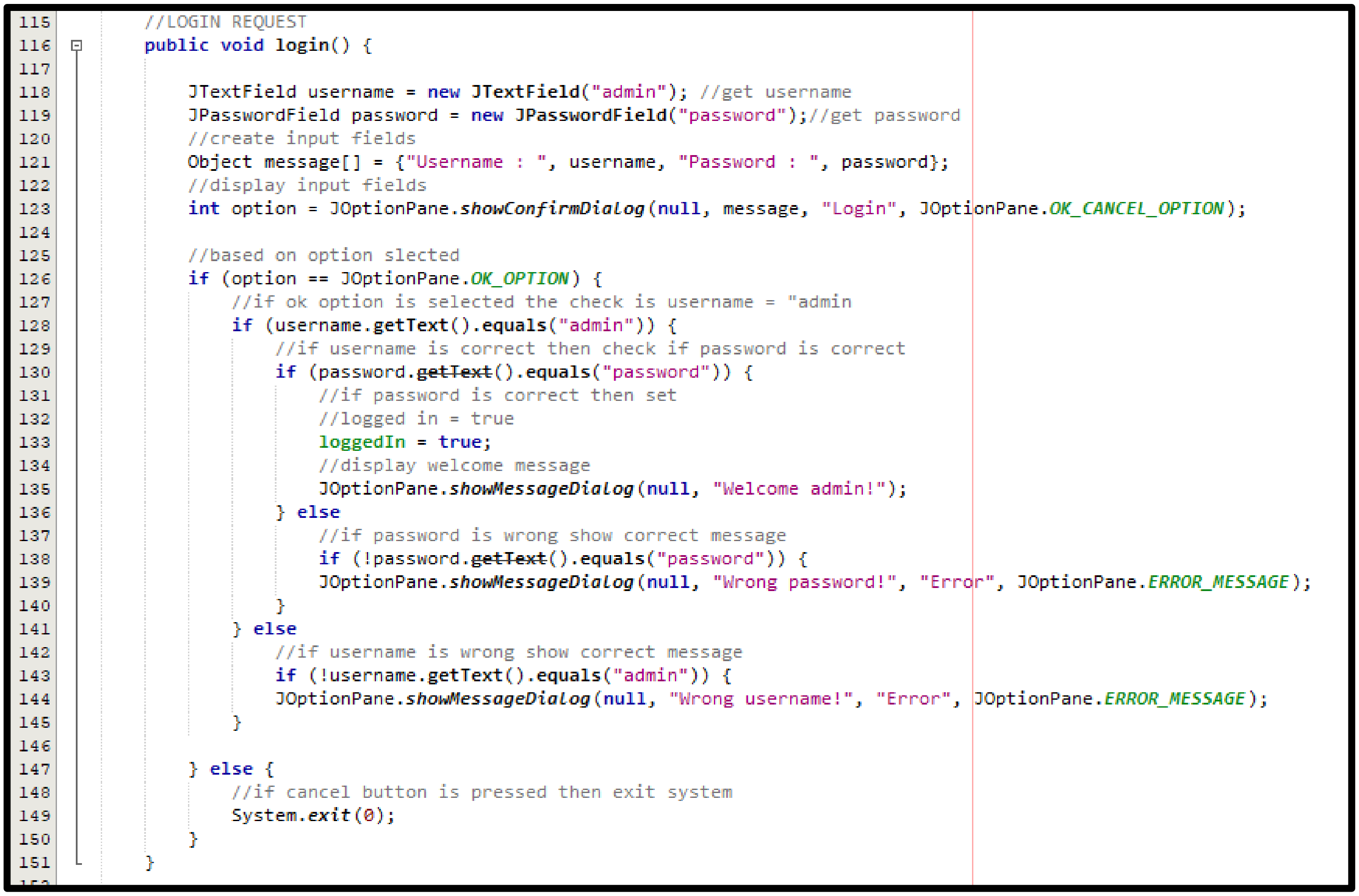

```java
    //INSTANTIATES SERVER SOCKET OBJECT & STARTS LISTENING
    public void turnOnServer(){
        if(listening==false){
            try{
                serverSocket = new ServerSocket(7777);  //listening on port 7777.
                listening =true;    //set listening true.
                System.out.println("Server closed :" + serverSocket.isClosed());
            }catch (IOException e){ //catch IO exception.
                System.out.println(e.toString());   //print exception message.
                System.exit(1); }//exit process.
             jLabel1.setText("Server status : online"); //updating server status.
        } else {
            ///UPDATING SERVER STATUS USING ANONYMOUS THREAD OBJECT
            Thread timer = new Thread(new Runnable() {  //create new thread
                public void run() { //using overriden run method
                     //chaning text
                    jLabel1.setText("Server status : already running!");
                    try{
                    Thread.sleep(1000);  //pause for 1 second
                    }catch (InterruptedException e){} //catching exception
                    jLabel1.setText("Server status : online"); //update text
                }
            });timer.start(); //starts thread object.
        }  //LISTENING INSIDE THREAD
            Thread t = new Thread(new Runnable() {
                public void run() {
                     /*
                     EXPLANATION: so while the server is listening then create
                                  a new session. (Meaning allow clients to connect
                                  to the server)
                     */

                      while(listening){
                         try{
                                new Session(serverSocket.accept());
                                System.out.println("Listening!");
                         }catch (IOException e){
                                System.out.println(e.toString());
                         }
                      }
                }
            });
            t.start();
    }
```



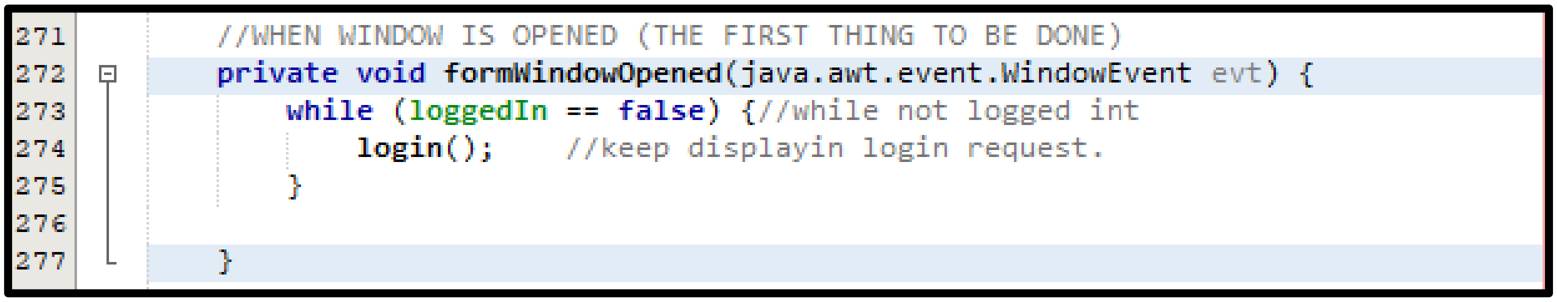
```
    //CLOSES SERVER SOCKET
    public void shutdownServer(){
        //RUNNING INSDE ANONYMOUSE THREAD
        Thread s = new Thread(new Runnable() {
            @Override
            public void run() {;
                try{
                    /*
                    EXPLANATION: Create a socket. Create outputstream object.
                                 Tell self (Server) to shutdow.
                                 And close serverSocket
                                 And set listening to false so can check if
                                 server is online or offline (SEE OTHER
                                 METHODS)
                    */
                     Socket sSocket = new Socket("localhost",7777);
                     ObjectOutputStream send = new ObjectOutputStream(sSocket.getOutputStream());
                     send.writeObject("shutdown");
                     serverSocket.close();

                listening = false;
                System.out.println("Server closed :" + serverSocket.isClosed());
                jLabel1.setText("Server status : offline"); //updating server status.

                }catch (IOException e){ //catch IOException
                }
            }
        });
        s.start(); //start thread

    }
```



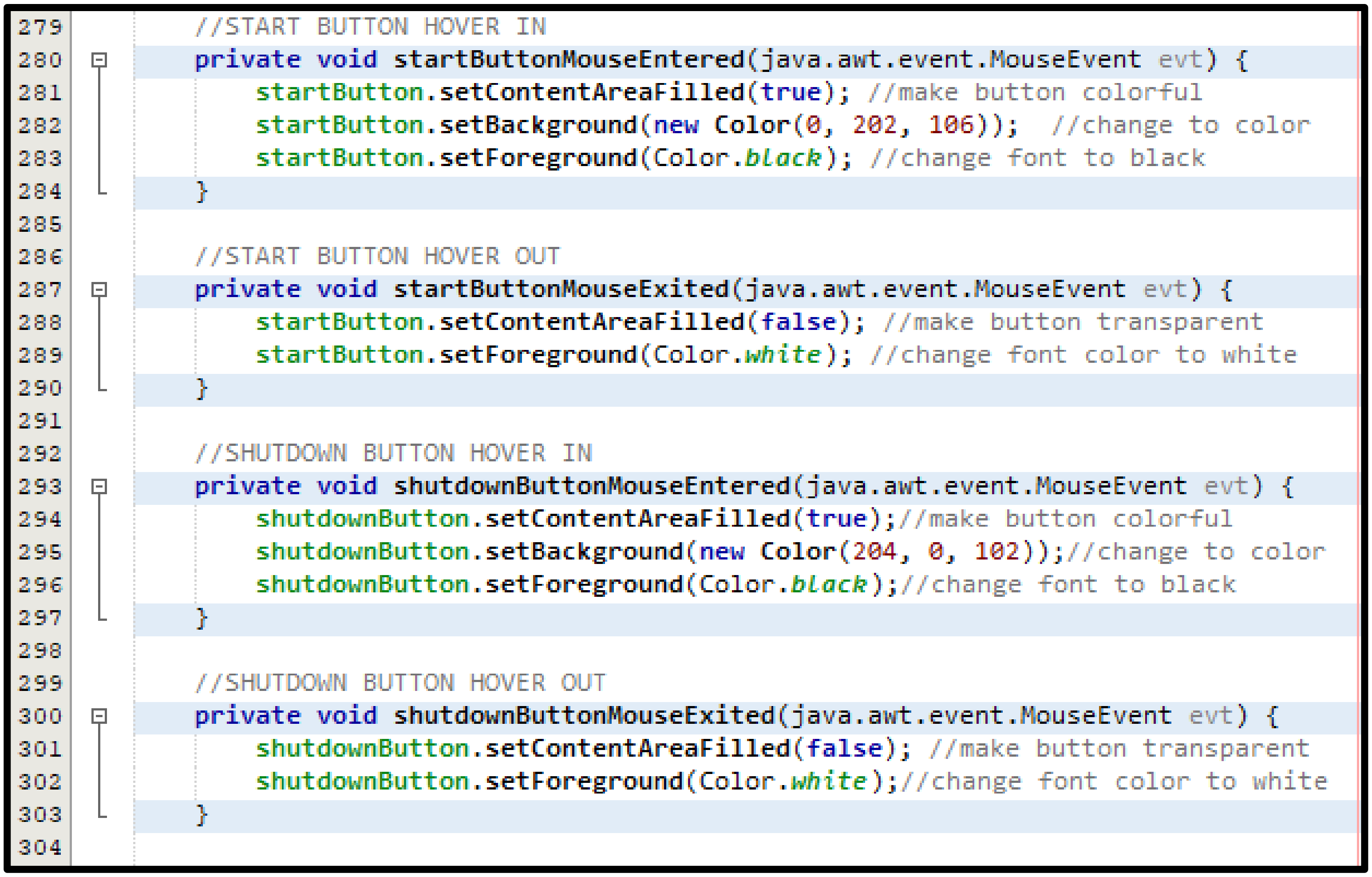
```
    //SERVER CONSTRUCTOR
    public Server() {
        initComponents();
        listening = false; //set listening to false by default.

    }
```



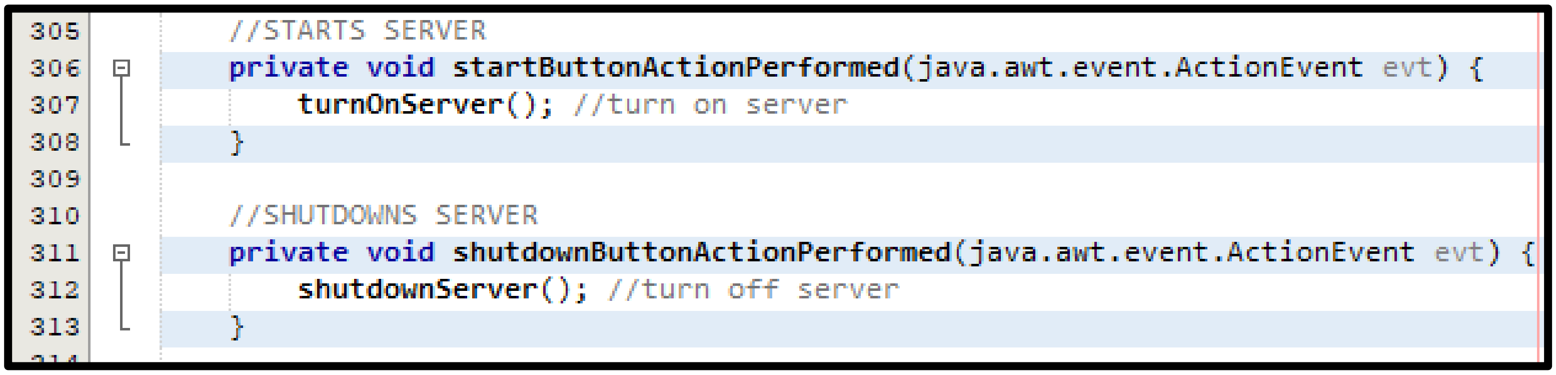
```
    //LOGIN REQUEST
    public void login() {

        JTextField username = new JTextField("admin"); //get username
        JPasswordField password = new JPasswordField("password");//get password
        //create input fields
        Object message[] = {"Username : ", username, "Password : ", password};
        //display input fields
        int option = JOptionPane.showConfirmDialog(null, message, "Login", JOptionPane.OK_CANCEL_OPTION);

        //based on option slected
        if (option == JOptionPane.OK_OPTION) {
            //if ok option is selected the check is username = "admin
            if (username.getText().equals("admin")) {
                //if username is correct then check if password is correct
                if (password.~~getText~~().equals("password")) {
                    //if password is correct then set
                    //logged in = true
                    loggedIn = true;
                    //display welcome message
                    JOptionPane.showMessageDialog(null, "Welcome admin!");
                } else
                    //if password is wrong show correct message
                    if (!password.~~getText~~().equals("password")) {
                    JOptionPane.showMessageDialog(null, "Wrong password!", "Error", JOptionPane.ERROR_MESSAGE);
                }
            } else
                //if username is wrong show correct message
                if (!username.getText().equals("admin")) {
                JOptionPane.showMessageDialog(null, "Wrong username!", "Error", JOptionPane.ERROR_MESSAGE);
            }

        } else {
            //if cancel button is pressed then exit system
            System.exit(0);
        }
    }
```



```
    //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.
        }

    }
```

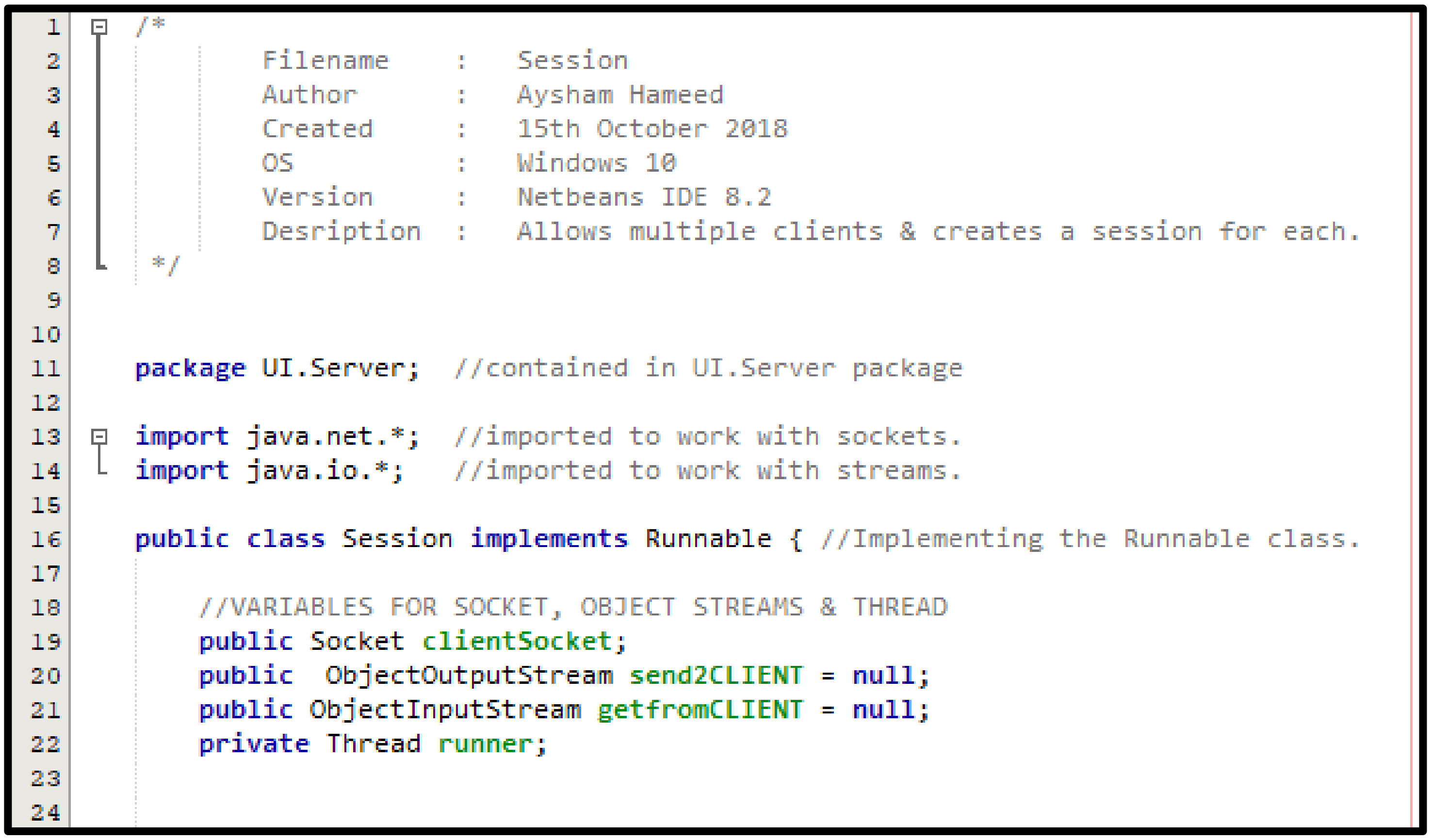


```java
    //START BUTTON HOVER IN
    private void startButtonMouseEntered(java.awt.event.MouseEvent evt) {
        startButton.setContentAreaFilled(true); //make button colorful
        startButton.setBackground(new Color(0, 202, 106));  //change to color
        startButton.setForeground(Color.black); //change font to black
    }

    //START BUTTON HOVER OUT
    private void startButtonMouseExited(java.awt.event.MouseEvent evt) {
        startButton.setContentAreaFilled(false); //make button transparent
        startButton.setForeground(Color.white); //change font color to white
    }

    //SHUTDOWN BUTTON HOVER IN
    private void shutdownButtonMouseEntered(java.awt.event.MouseEvent evt) {
        shutdownButton.setContentAreaFilled(true);//make button colorful
        shutdownButton.setBackground(new Color(204, 0, 102));//change to color
        shutdownButton.setForeground(Color.black);//change font to black
    }

    //SHUTDOWN BUTTON HOVER OUT
    private void shutdownButtonMouseExited(java.awt.event.MouseEvent evt) {
        shutdownButton.setContentAreaFilled(false); //make button transparent
        shutdownButton.setForeground(Color.white);//change font color to white
    }

```

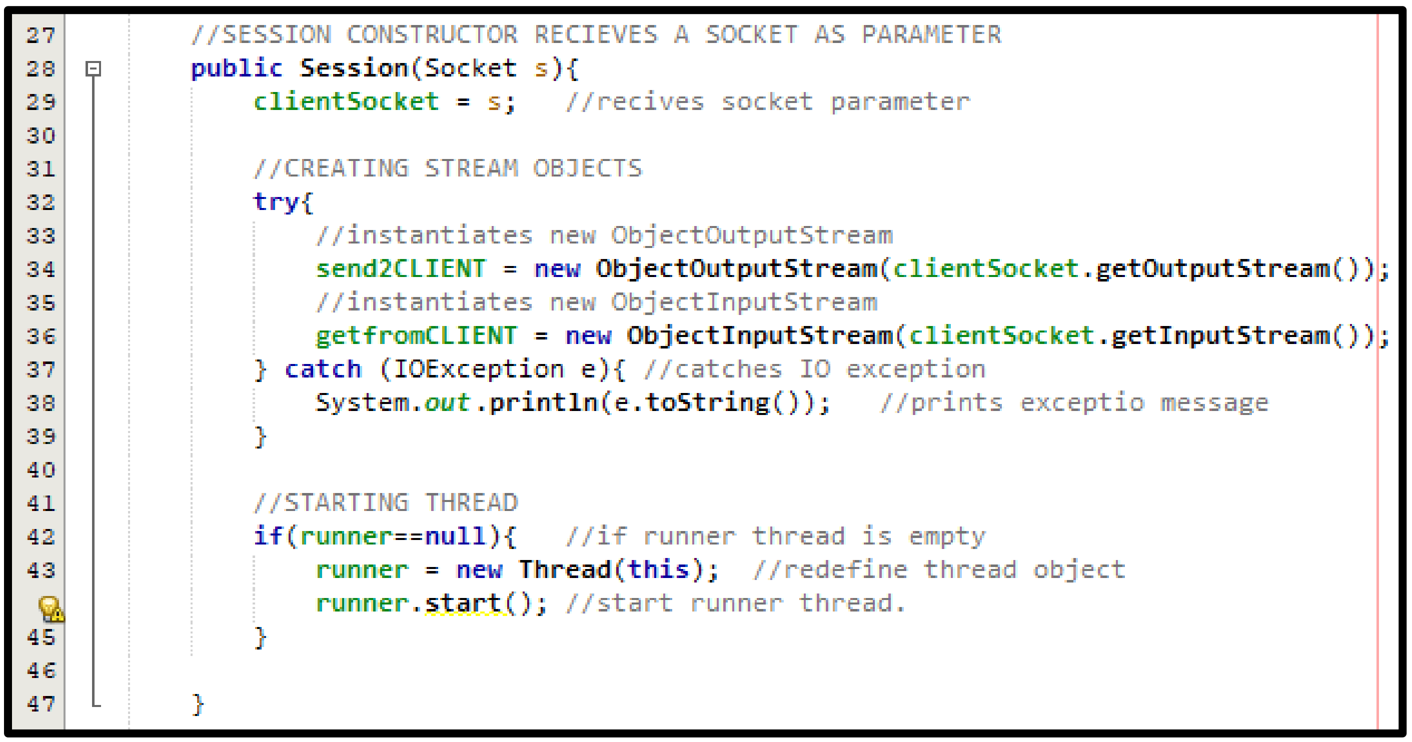


```java
    //STARTS SERVER
    private void startButtonActionPerformed(java.awt.event.ActionEvent evt) {
        turnOnServer(); //turn on server
    }

    //SHUTDOWNS SERVER
    private void shutdownButtonActionPerformed(java.awt.event.ActionEvent evt) {
        shutdownServer(); //turn off server
    }
```

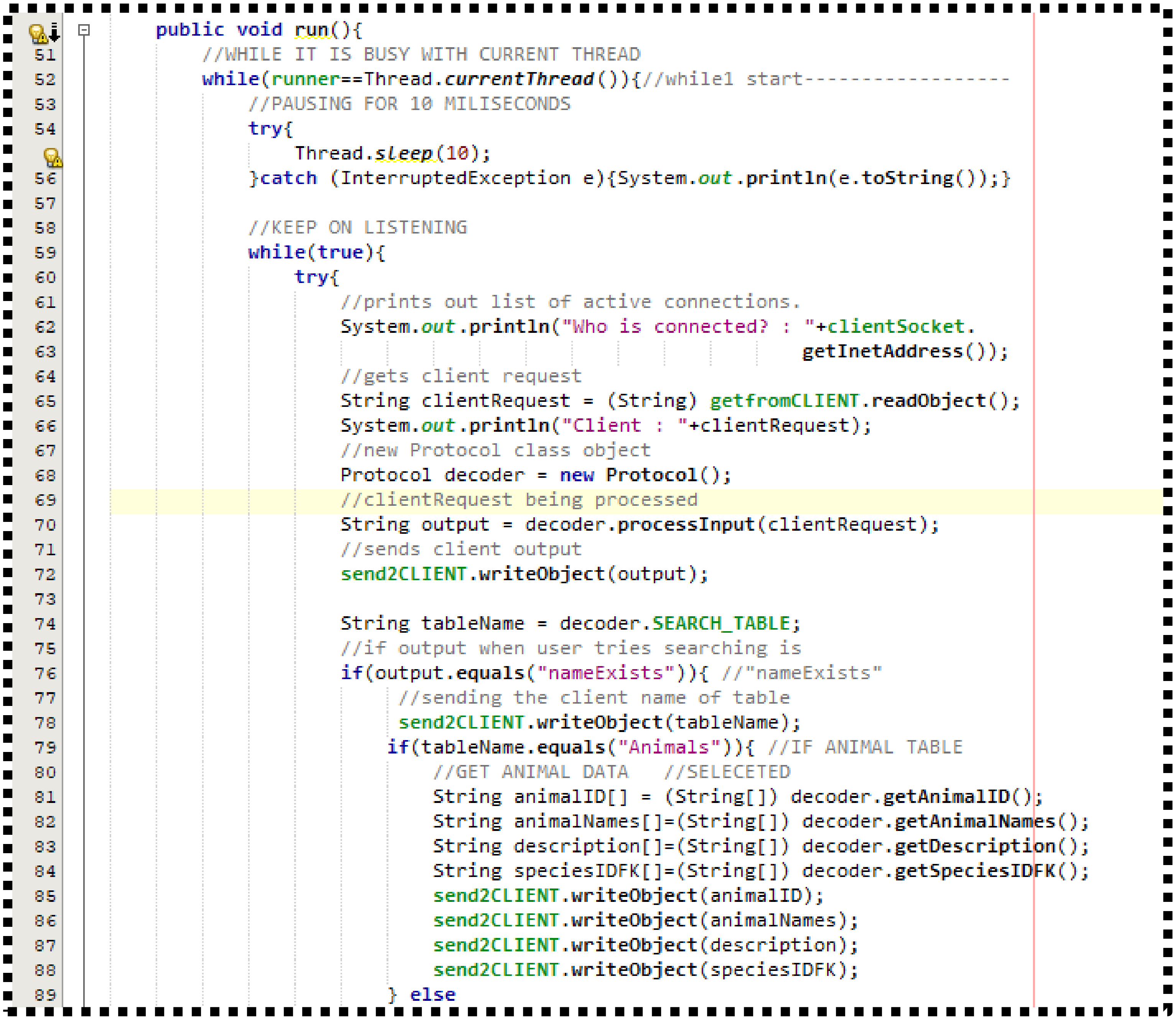
## Session Class



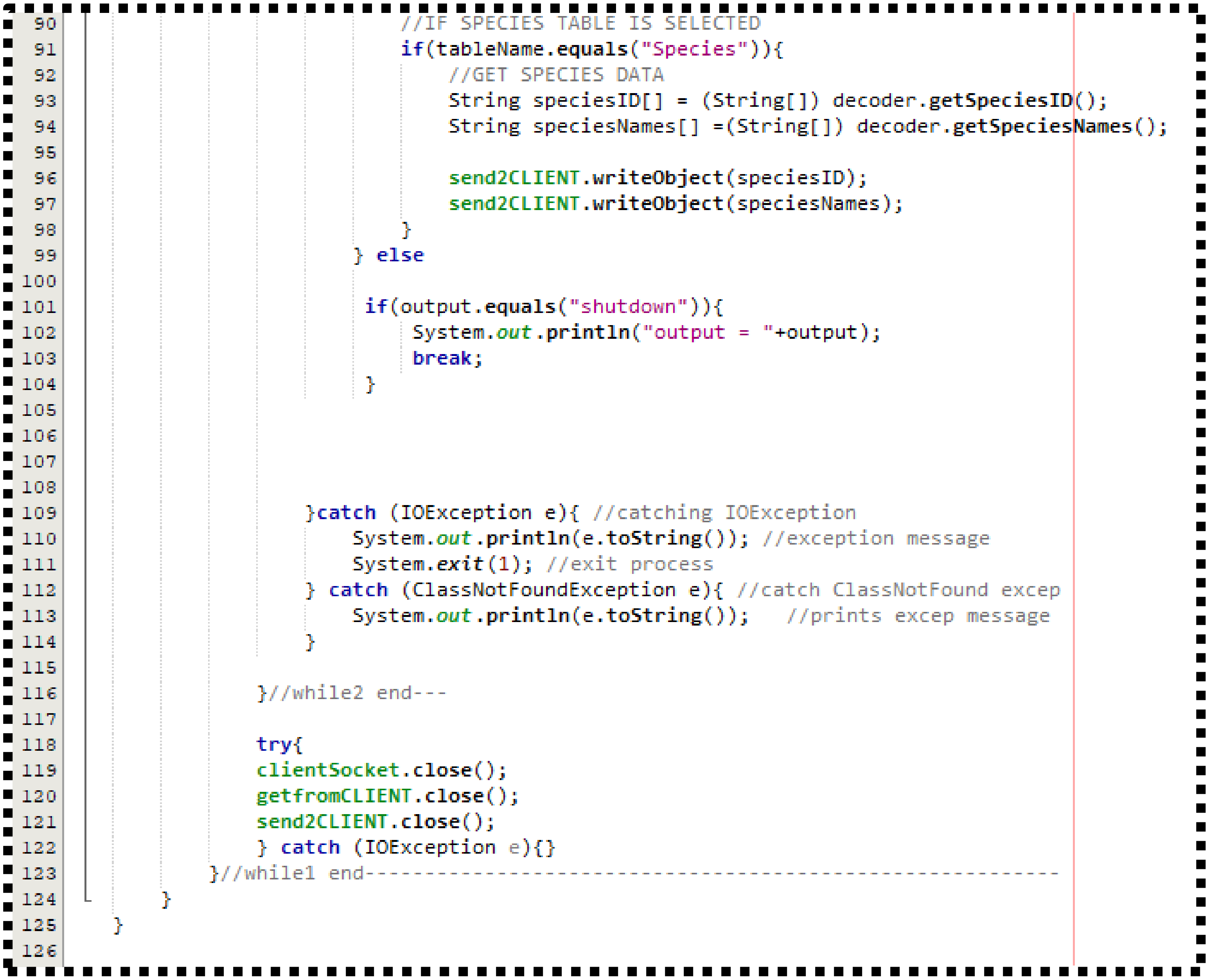
```java
/*
            Filename     :   Session
            Author       :   Aysham Hameed
            Created      :   15th October 2018
            OS           :   Windows 10
            Version      :   Netbeans IDE 8.2
            Desription   :   Allows multiple clients & creates a session for each.
 */


package UI.Server;  //contained in UI.Server package

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

public class Session implements Runnable { //Implementing the Runnable class.

    //VARIABLES FOR SOCKET, OBJECT STREAMS & THREAD
    public Socket clientSocket;
    public  ObjectOutputStream send2CLIENT = null;
    public ObjectInputStream getfromCLIENT = null;
    private Thread runner;


```



```java
    //SESSION CONSTRUCTOR RECIEVES A SOCKET AS PARAMETER
    public Session(Socket s){
        clientSocket = s;   //recives socket parameter

        //CREATING STREAM OBJECTS
        try{
            //instantiates new ObjectOutputStream
            send2CLIENT = new ObjectOutputStream(clientSocket.getOutputStream());
            //instantiates new ObjectInputStream
            getfromCLIENT = new ObjectInputStream(clientSocket.getInputStream());
        } catch (IOException e){ //catches IO exception
            System.out.println(e.toString());   //prints exceptio message
        }

        //STARTING THREAD
        if(runner==null){   //if runner thread is empty
            runner = new Thread(this);  //redefine thread object
            runner.start(); //start runner thread.
        }

    }
```

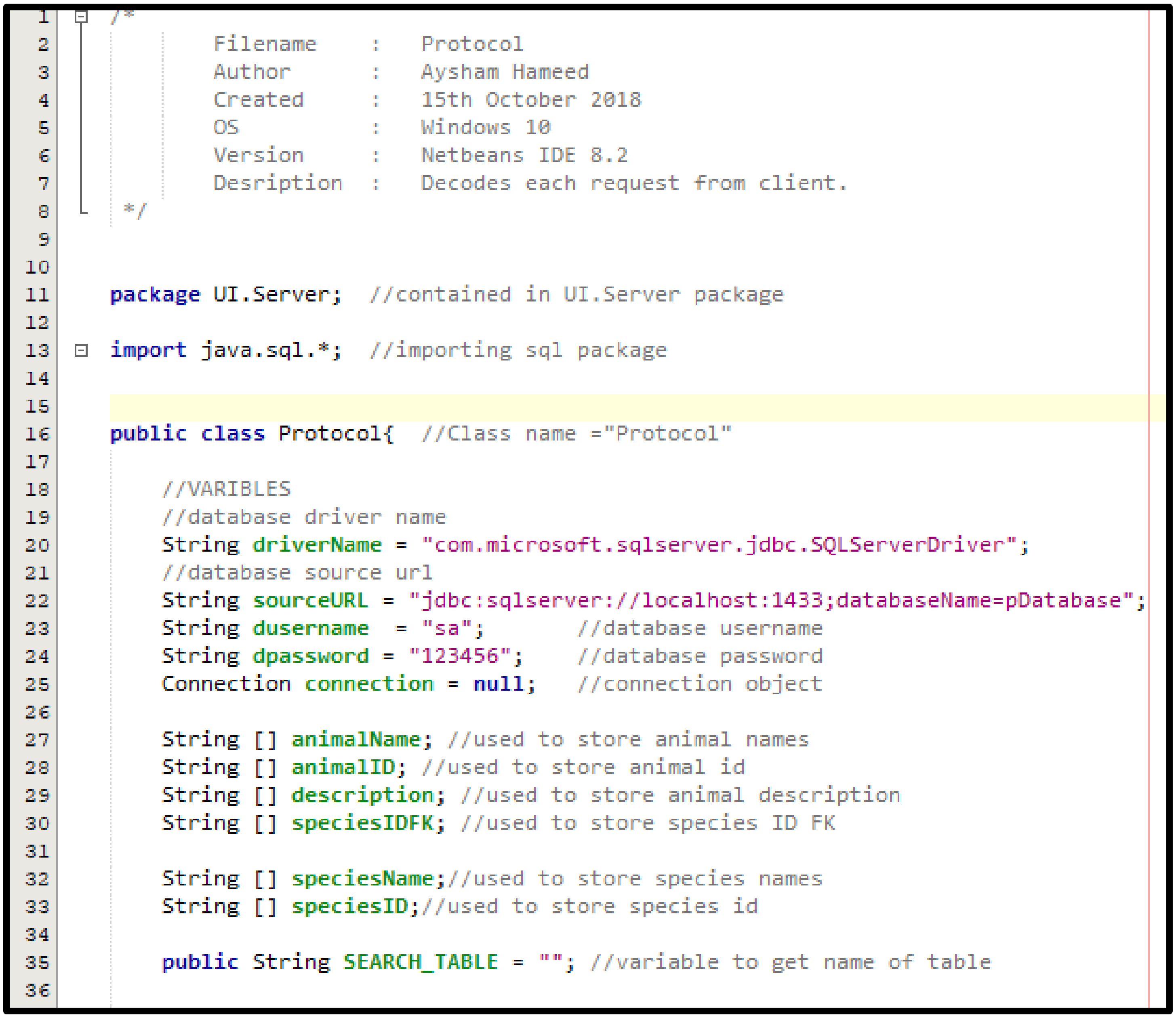


```
    public void run(){
        //WHILE IT IS BUSY WITH CURRENT THREAD
        while(runner==Thread.currentThread()){//while1 start------------------
            //PAUSING FOR 10 MILISECONDS
            try{
                Thread.sleep(10);
            }catch (InterruptedException e){System.out.println(e.toString());}

            //KEEP ON LISTENING
            while(true){
                try{
                    //prints out list of active connections.
                    System.out.println("Who is connected? : "+clientSocket.
                                                            getInetAddress());
                    //gets client request
                    String clientRequest = (String) getfromCLIENT.readObject();
                    System.out.println("Client : "+clientRequest);
                    //new Protocol class object
                    Protocol decoder = new Protocol();
                    //clientRequest being processed
                    String output = decoder.processInput(clientRequest);
                    //sends client output
                    send2CLIENT.writeObject(output);

                    String tableName = decoder.SEARCH_TABLE;
                    //if output when user tries searching is
                    if(output.equals("nameExists")){ //"nameExists"
                         //sending the client name of table
                         send2CLIENT.writeObject(tableName);
                        if(tableName.equals("Animals")){ //IF ANIMAL TABLE
                            //GET ANIMAL DATA   //SELECETED
                            String animalID[] = (String[]) decoder.getAnimalID();
                            String animalNames[]=(String[]) decoder.getAnimalNames();
                            String description[]=(String[]) decoder.getDescription();
                            String speciesIDFK[]=(String[]) decoder.getSpeciesIDFK();
                            send2CLIENT.writeObject(animalID);
                            send2CLIENT.writeObject(animalNames);
                            send2CLIENT.writeObject(description);
                            send2CLIENT.writeObject(speciesIDFK);
                        } else
```

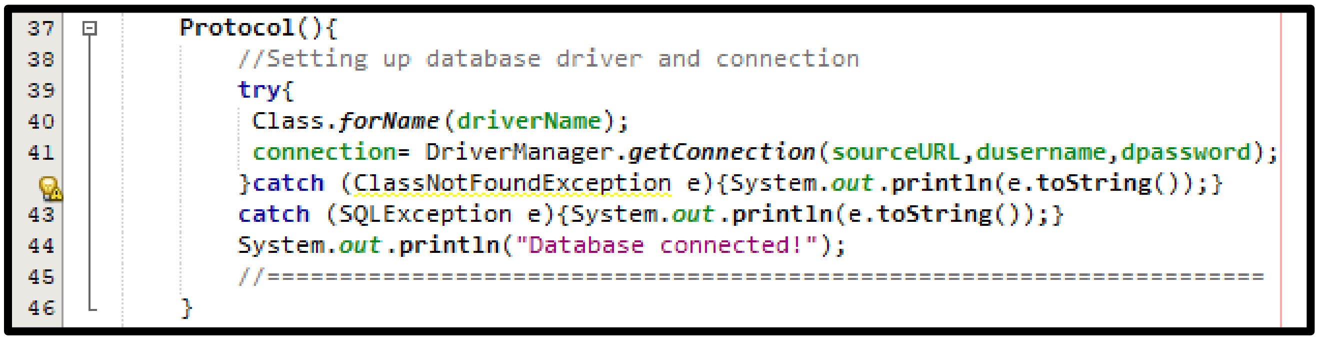


```
                            //IF SPECIES TABLE IS SELECTED
                            if(tableName.equals("Species")){
                                //GET SPECIES DATA
                                String speciesID[] = (String[]) decoder.getSpeciesID();
                                String speciesNames[] =(String[]) decoder.getSpeciesNames();

                                send2CLIENT.writeObject(speciesID);
                                send2CLIENT.writeObject(speciesNames);
                            }
                        } else

                         if(output.equals("shutdown")){
                             System.out.println("output = "+output);
                             break;
                         }




                    }catch (IOException e){ //catching IOException
                        System.out.println(e.toString()); //exception message
                        System.exit(1); //exit process
                    } catch (ClassNotFoundException e){ //catch ClassNotFound excep
                        System.out.println(e.toString());   //prints excep message
                    }

                }//while2 end---

                try{
                clientSocket.close();
                getfromCLIENT.close();
                send2CLIENT.close();
                } catch (IOException e){}
            }//while1 end----------------------------------------------------------
        }
    }
```

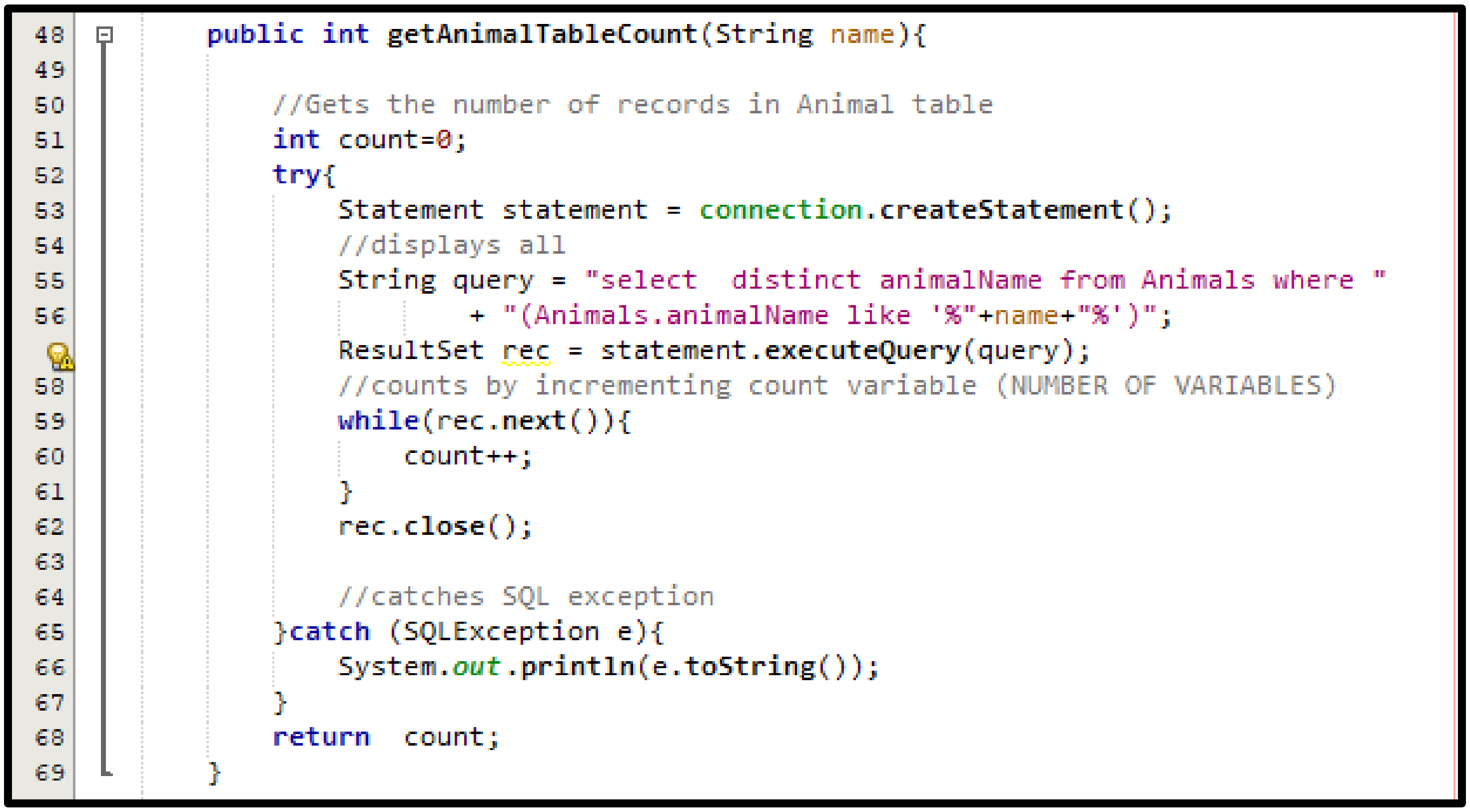
## Protocol Class



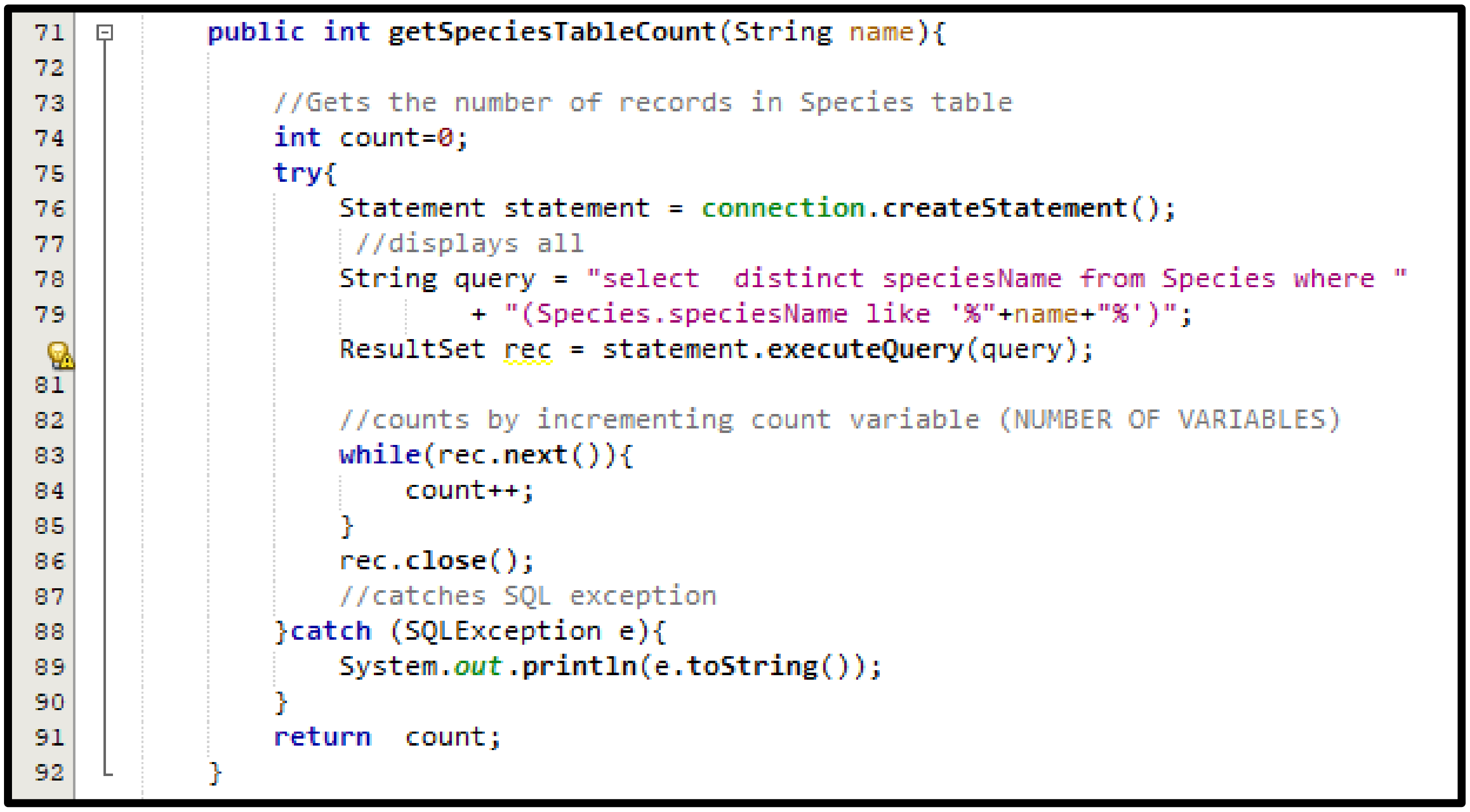
```java
/*
        Filename    :   Protocol
        Author      :   Aysham Hameed
        Created     :   15th October 2018
        OS          :   Windows 10
        Version     :   Netbeans IDE 8.2
        Desription  :   Decodes each request from client.
 */


package UI.Server;  //contained in UI.Server package

import java.sql.*;  //importing sql package


public class Protocol{  //Class name ="Protocol"

    //VARIBLES
    //database driver name
    String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    //database source url
    String sourceURL = "jdbc:sqlserver://localhost:1433;databaseName=pDatabase";
    String dusername  = "sa";       //database username
    String dpassword = "123456";    //database password
    Connection connection = null;   //connection object

    String [] animalName; //used to store animal names
    String [] animalID; //used to store animal id
    String [] description; //used to store animal description
    String [] speciesIDFK; //used to store species ID FK

    String [] speciesName;//used to store species names
    String [] speciesID;//used to store species id

    public String SEARCH_TABLE = ""; //variable to get name of table

```



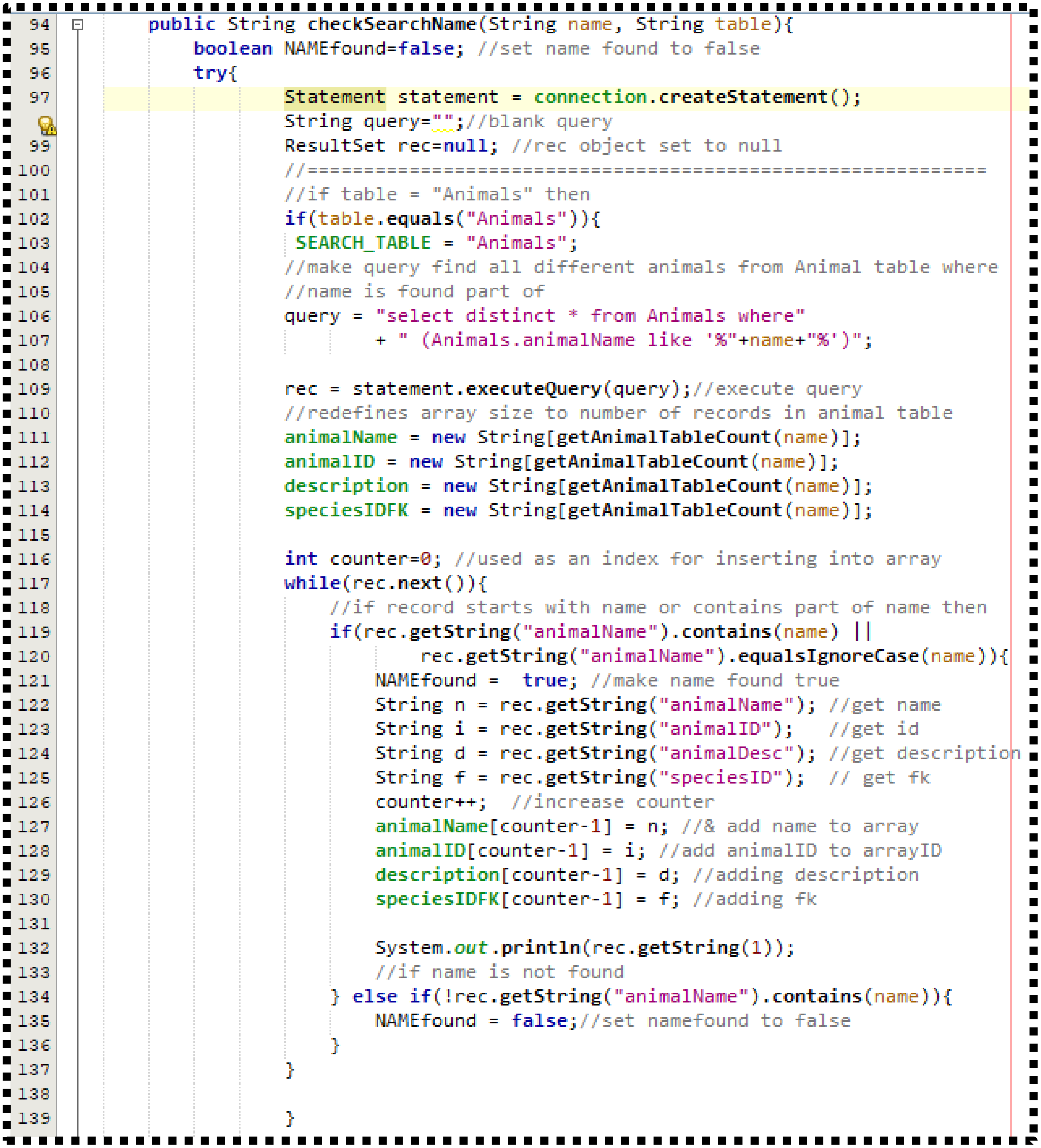
```java
    Protocol(){
        //Setting up database driver and connection
        try{
         Class.forName(driverName);
         connection= DriverManager.getConnection(sourceURL,dusername,dpassword);
        }catch (ClassNotFoundException e){System.out.println(e.toString());}
        catch (SQLException e){System.out.println(e.toString());}
        System.out.println("Database connected!");
        //======================================================================
    }
```



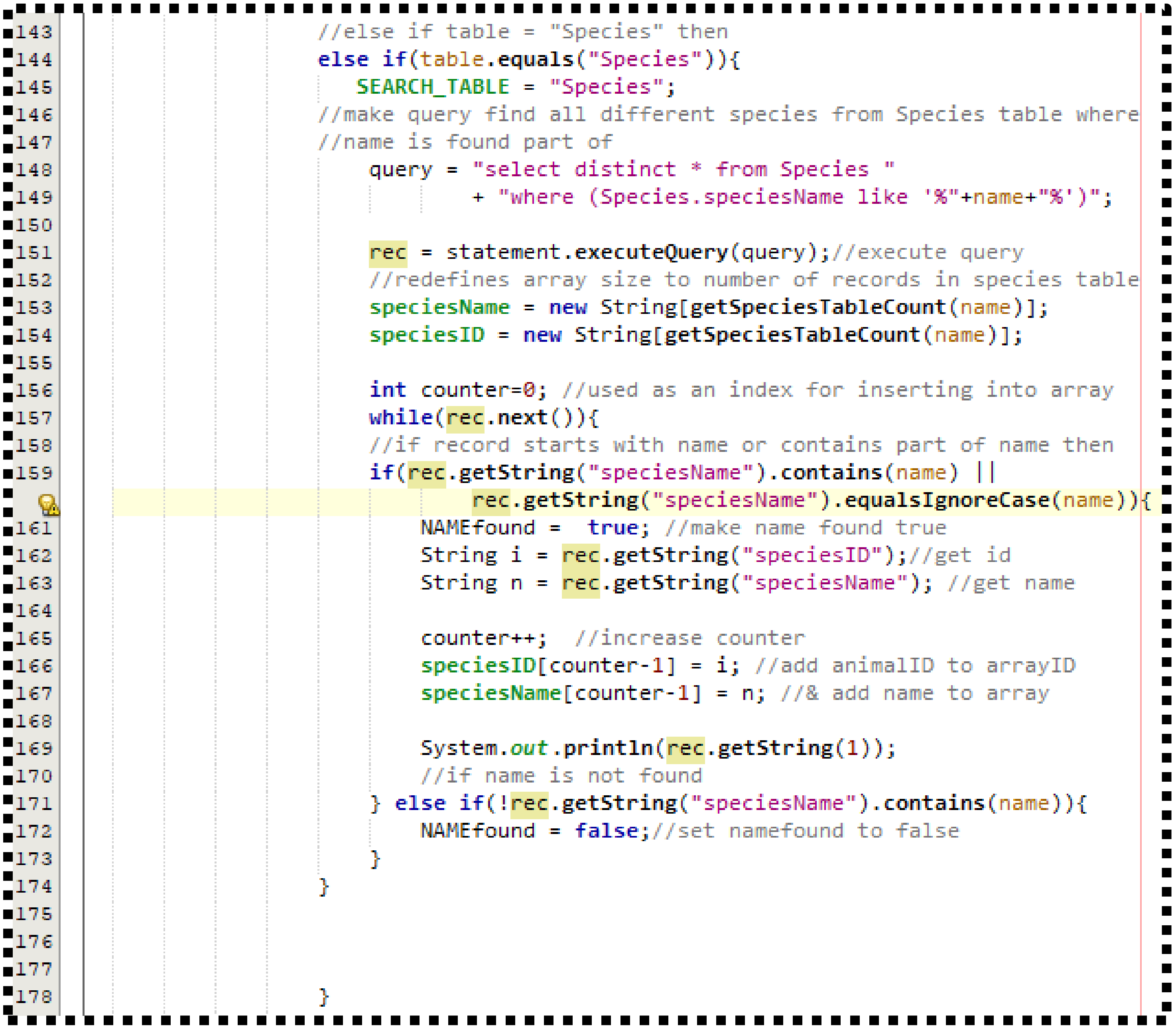
```java
    public int getAnimalTableCount(String name){

        //Gets the number of records in Animal table
        int count=0;
        try{
            Statement statement = connection.createStatement();
            //displays all
            String query = "select  distinct animalName from Animals where "
                    + "(Animals.animalName like '%"+name+"%')";
            ResultSet rec = statement.executeQuery(query);
            //counts by incrementing count variable (NUMBER OF VARIABLES)
            while(rec.next()){
                count++;
            }
            rec.close();

            //catches SQL exception
        }catch (SQLException e){
            System.out.println(e.toString());
        }
        return  count;
    }
```



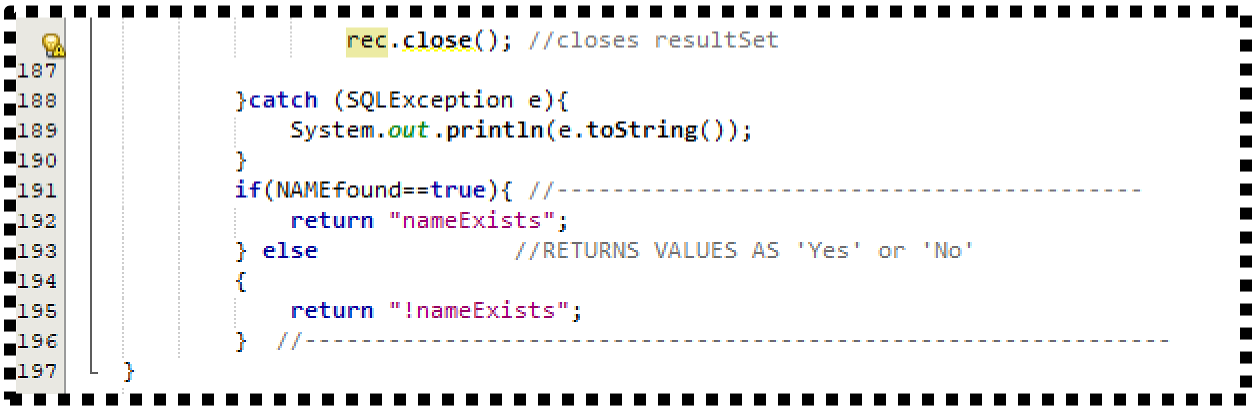
```java
    public int getSpeciesTableCount(String name){

        //Gets the number of records in Species table
        int count=0;
        try{
            Statement statement = connection.createStatement();
             //displays all
            String query = "select  distinct speciesName from Species where "
                    + "(Species.speciesName like '%"+name+"%')";
            ResultSet rec = statement.executeQuery(query);

            //counts by incrementing count variable (NUMBER OF VARIABLES)
            while(rec.next()){
                count++;
            }
            rec.close();
            //catches SQL exception
        }catch (SQLException e){
            System.out.println(e.toString());
        }
        return  count;
    }
```



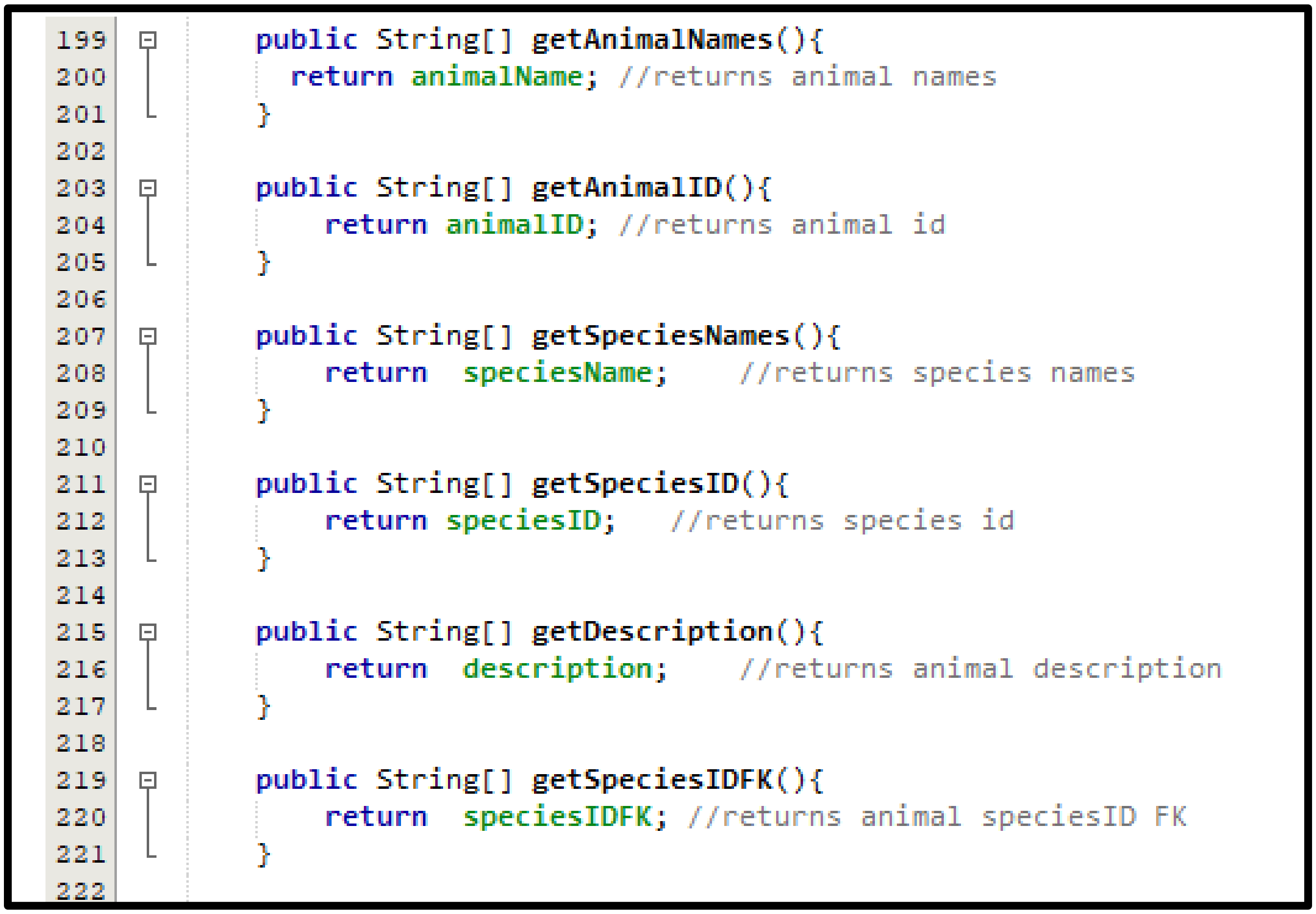
```java
    public String checkSearchName(String name, String table){
        boolean NAMEfound=false; //set name found to false
        try{
                Statement statement = connection.createStatement();
                String query="";//blank query
                ResultSet rec=null; //rec object set to null
                //=============================================================
                //if table = "Animals" then
                if(table.equals("Animals")){
                 SEARCH_TABLE = "Animals";
                //make query find all different animals from Animal table where
                //name is found part of
                query = "select distinct * from Animals where"
                        + " (Animals.animalName like '%"+name+"%')";

                rec = statement.executeQuery(query);//execute query
                //redefines array size to number of records in animal table
                animalName = new String[getAnimalTableCount(name)];
                animalID = new String[getAnimalTableCount(name)];
                description = new String[getAnimalTableCount(name)];
                speciesIDFK = new String[getAnimalTableCount(name)];

                int counter=0; //used as an index for inserting into array
                while(rec.next()){
                    //if record starts with name or contains part of name then
                    if(rec.getString("animalName").contains(name) ||
                            rec.getString("animalName").equalsIgnoreCase(name)){
                        NAMEfound =  true; //make name found true
                        String n = rec.getString("animalName"); //get name
                        String i = rec.getString("animalID");   //get id
                        String d = rec.getString("animalDesc"); //get description
                        String f = rec.getString("speciesID");  // get fk
                        counter++;  //increase counter
                        animalName[counter-1] = n; //& add name to array
                        animalID[counter-1] = i; //add animalID to arrayID
                        description[counter-1] = d; //adding description
                        speciesIDFK[counter-1] = f; //adding fk

                        System.out.println(rec.getString(1));
                        //if name is not found
                    } else if(!rec.getString("animalName").contains(name)){
                        NAMEfound = false;//set namefound to false
                    }
                }

                }
```



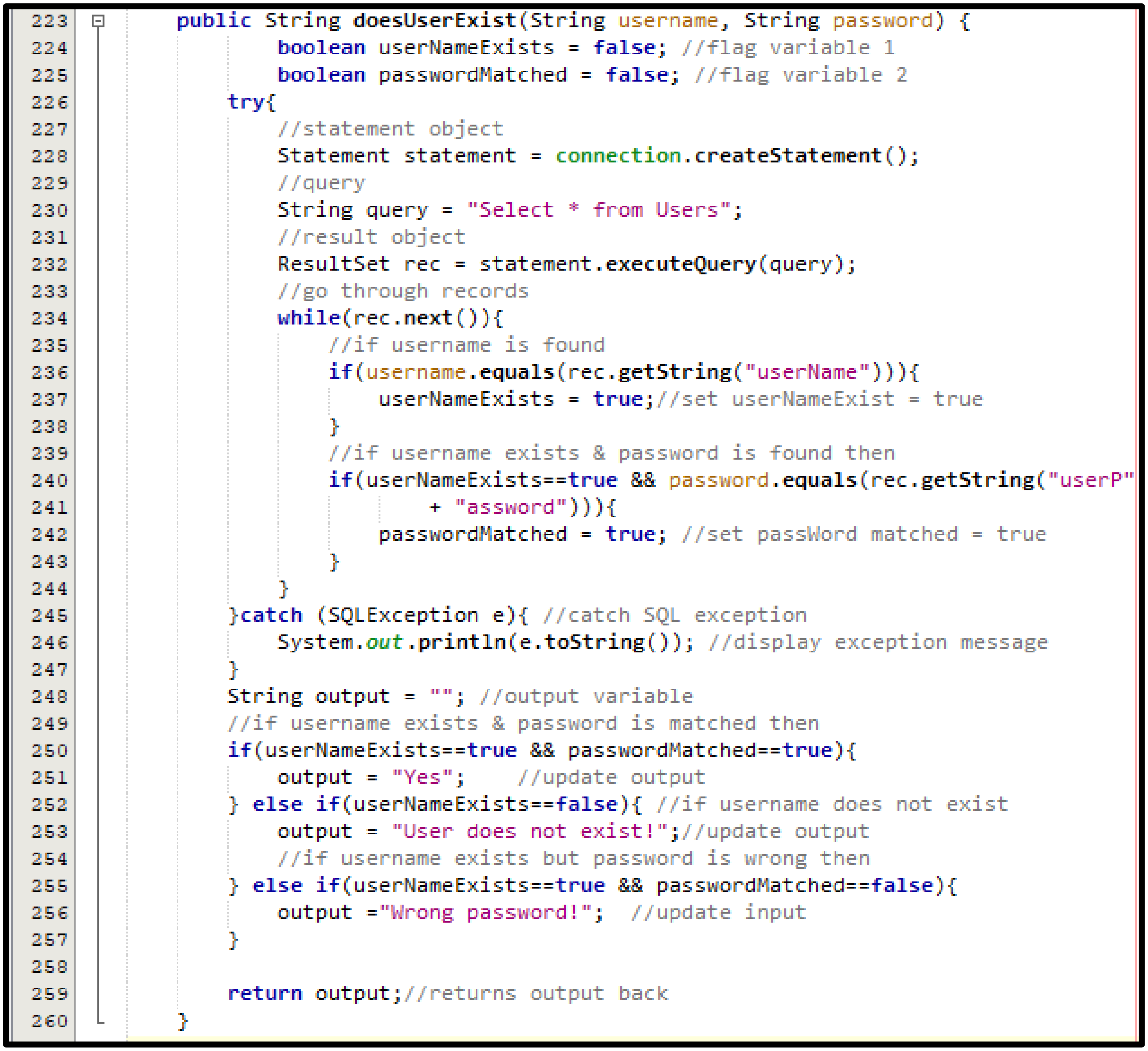
```java
                //else if table = "Species" then
                else if(table.equals("Species")){
                   SEARCH_TABLE = "Species";
                //make query find all different species from Species table where
                //name is found part of
                    query = "select distinct * from Species "
                            + "where (Species.speciesName like '%"+name+"%')";

                    rec = statement.executeQuery(query);//execute query
                    //redefines array size to number of records in species table
                    speciesName = new String[getSpeciesTableCount(name)];
                    speciesID = new String[getSpeciesTableCount(name)];

                    int counter=0; //used as an index for inserting into array
                    while(rec.next()){
                    //if record starts with name or contains part of name then
                    if(rec.getString("speciesName").contains(name) ||
                            rec.getString("speciesName").equalsIgnoreCase(name)){
                        NAMEfound =  true; //make name found true
                        String i = rec.getString("speciesID");//get id
                        String n = rec.getString("speciesName"); //get name

                        counter++;  //increase counter
                        speciesID[counter-1] = i; //add animalID to arrayID
                        speciesName[counter-1] = n; //& add name to array

                        System.out.println(rec.getString(1));
                        //if name is not found
                    } else if(!rec.getString("speciesName").contains(name)){
                        NAMEfound = false;//set namefound to false
                    }
                }



                }
```



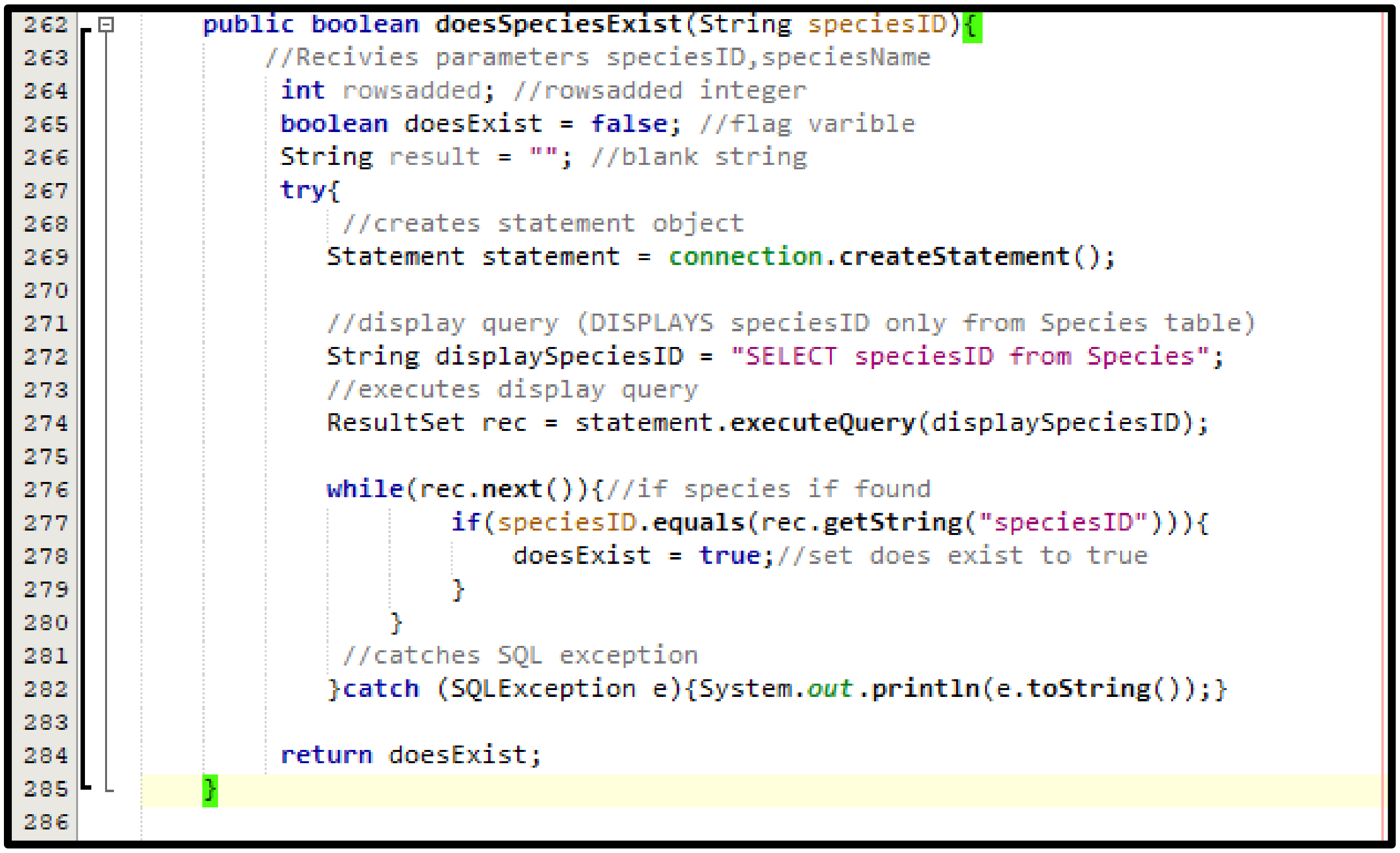
```java
                  rec.close(); //closes resultSet

          }catch (SQLException e){
              System.out.println(e.toString());
          }
          if(NAMEfound==true){ //------------------------------------------
              return "nameExists";
          } else              //RETURNS VALUES AS 'Yes' or 'No'
          {
              return "!nameExists";
          }  //--------------------------------------------------------------
  }
```



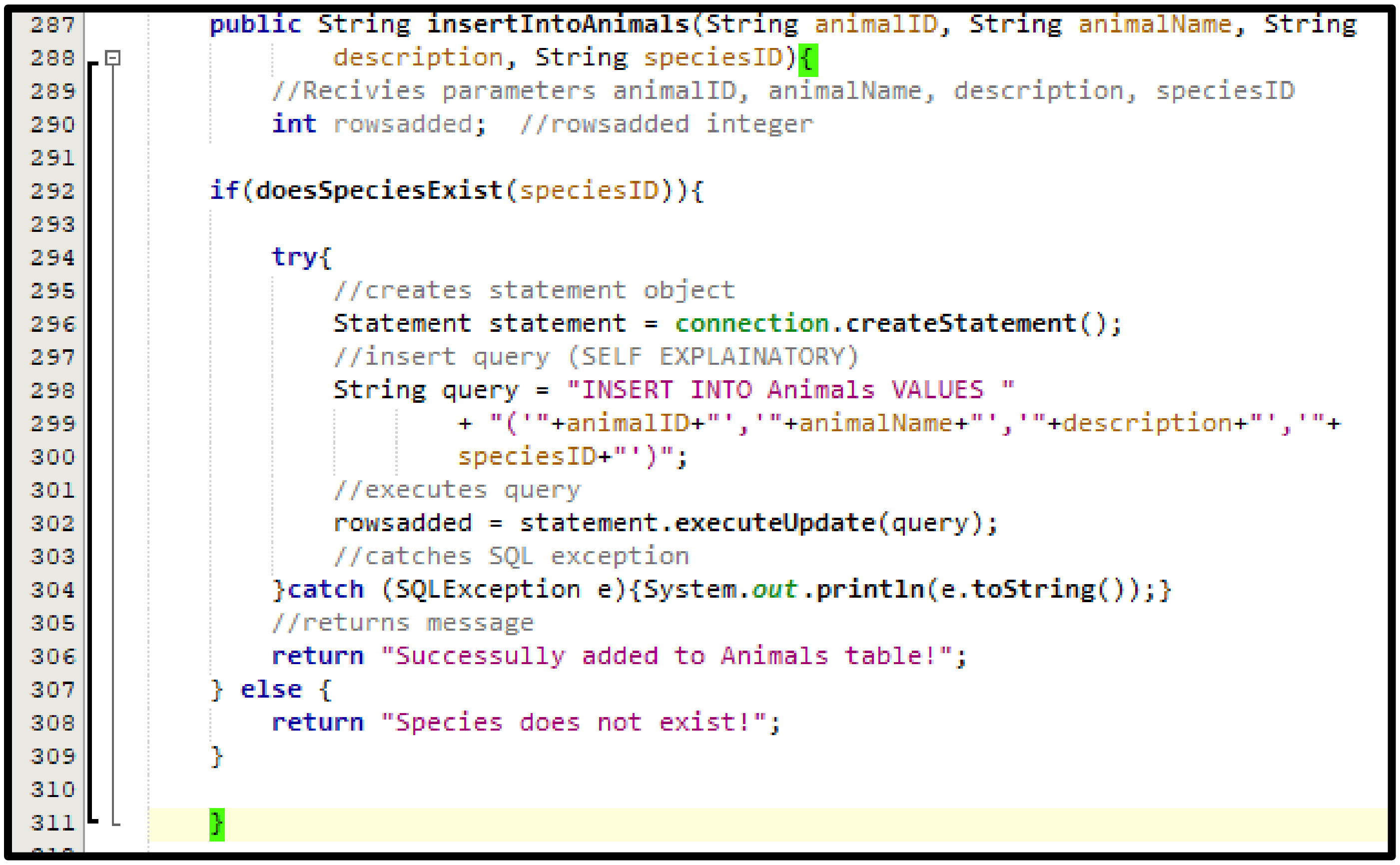
```
    public String[] getAnimalNames(){
     return animalName; //returns animal names
    }

    public String[] getAnimalID(){
        return animalID; //returns animal id
    }

    public String[] getSpeciesNames(){
        return  speciesName;    //returns species names
    }

    public String[] getSpeciesID(){
        return speciesID;   //returns species id
    }

    public String[] getDescription(){
        return  description;    //returns animal description
    }

    public String[] getSpeciesIDFK(){
        return  speciesIDFK; //returns animal speciesID FK
    }

```



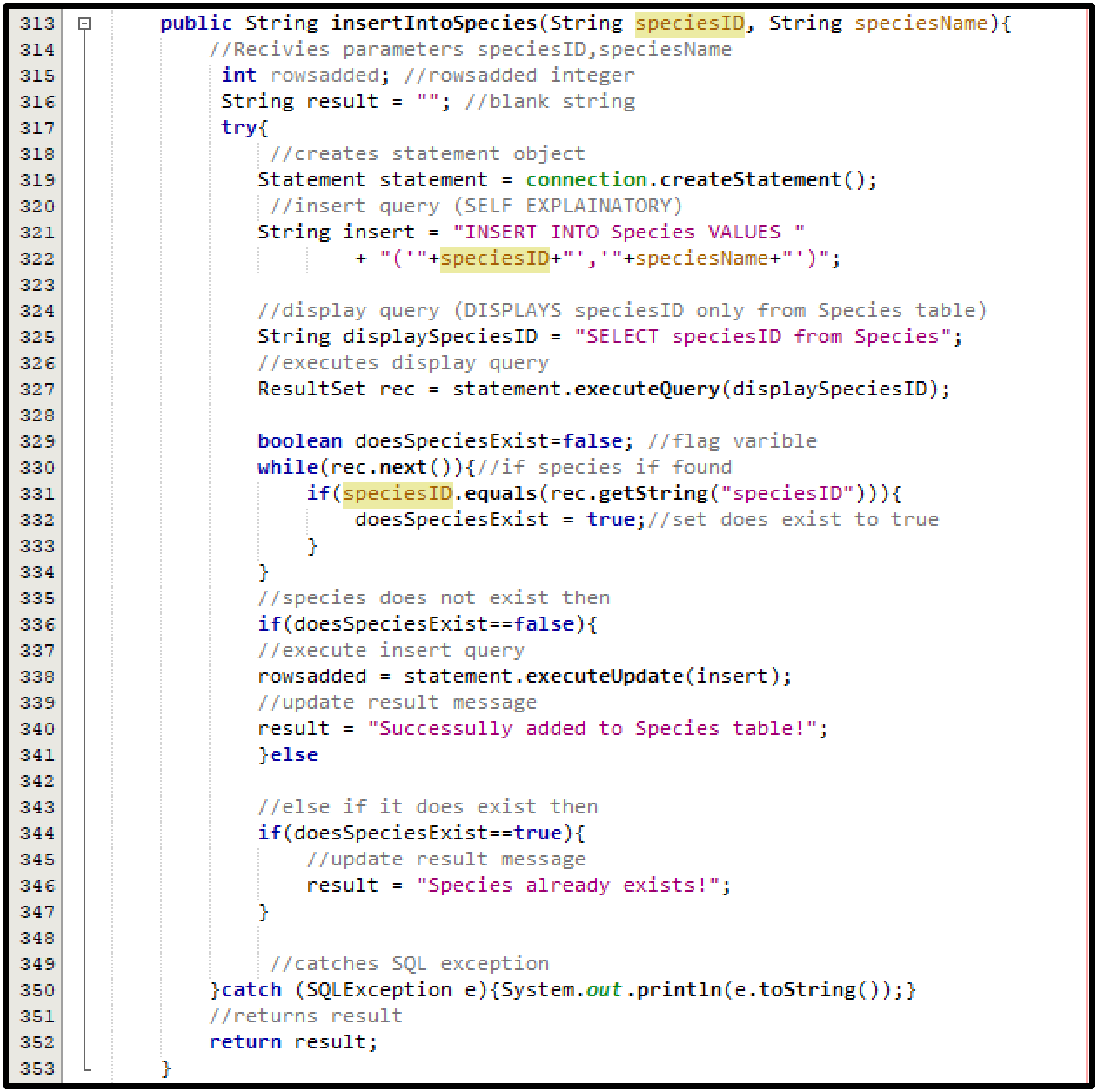
```
    public String doesUserExist(String username, String password) {
            boolean userNameExists = false; //flag variable 1
            boolean passwordMatched = false; //flag variable 2
        try{
            //statement object
            Statement statement = connection.createStatement();
            //query
            String query = "Select * from Users";
            //result object
            ResultSet rec = statement.executeQuery(query);
            //go through records
            while(rec.next()){
                //if username is found
                if(username.equals(rec.getString("userName"))){
                    userNameExists = true;//set userNameExist = true
                }
                //if username exists & password is found then
                if(userNameExists==true && password.equals(rec.getString("userP"
                        + "assword"))){
                    passwordMatched = true; //set passWord matched = true
                }
            }
        }catch (SQLException e){ //catch SQL exception
            System.out.println(e.toString()); //display exception message
        }
        String output = ""; //output variable
        //if username exists & password is matched then
        if(userNameExists==true && passwordMatched==true){
            output = "Yes";    //update output
        } else if(userNameExists==false){ //if username does not exist
            output = "User does not exist!";//update output
            //if username exists but password is wrong then
        } else if(userNameExists==true && passwordMatched==false){
            output ="Wrong password!";  //update input
        }

        return output;//returns output back
    }
```



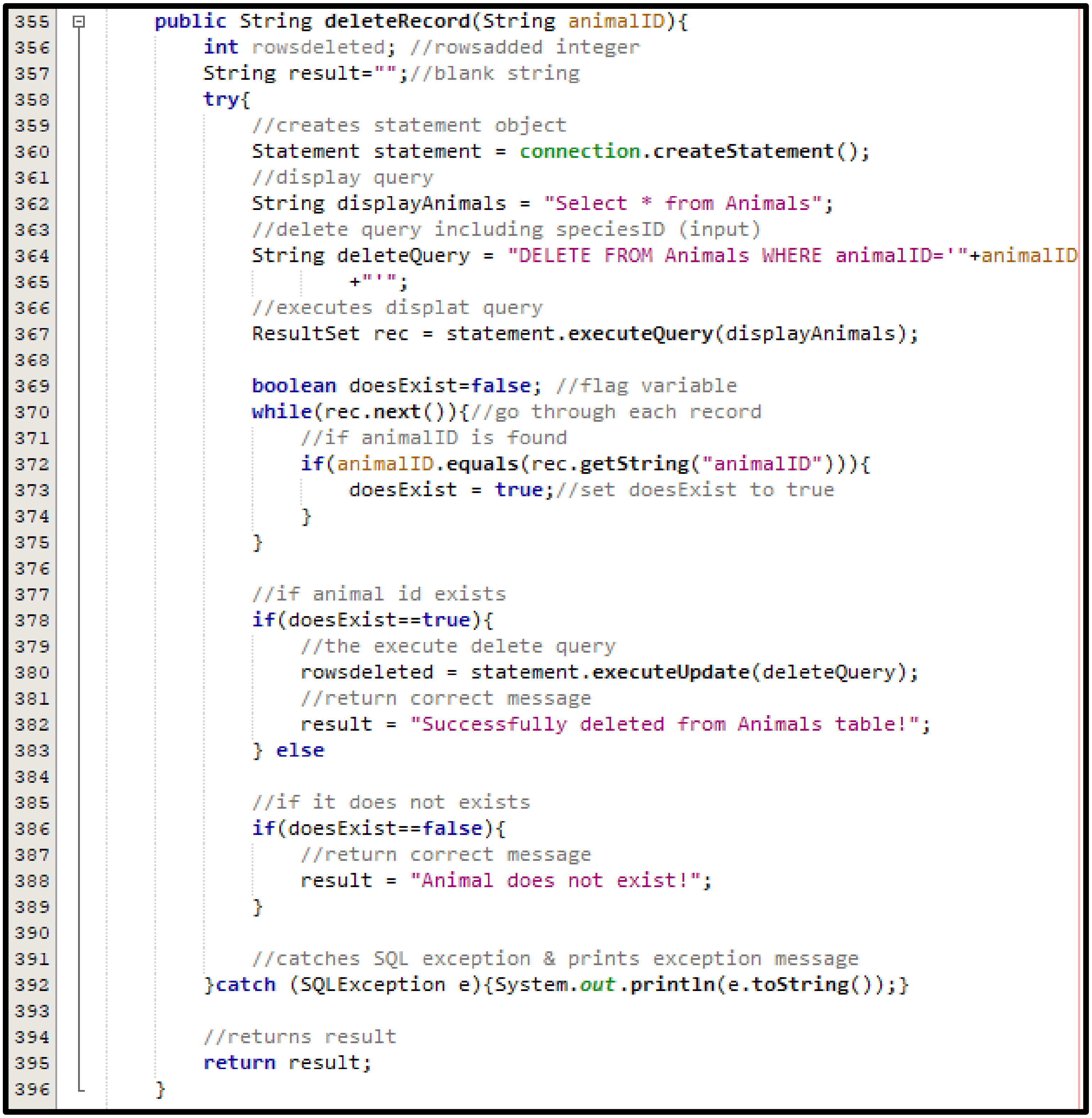
```
    public boolean doesSpeciesExist(String speciesID){
        //Recivies parameters speciesID,speciesName
         int rowsadded; //rowsadded integer
         boolean doesExist = false; //flag varible
         String result = ""; //blank string
         try{
            //creates statement object
            Statement statement = connection.createStatement();

            //display query (DISPLAYS speciesID only from Species table)
            String displaySpeciesID = "SELECT speciesID from Species";
            //executes display query
            ResultSet rec = statement.executeQuery(displaySpeciesID);

            while(rec.next()){//if species if found
                    if(speciesID.equals(rec.getString("speciesID"))){
                        doesExist = true;//set does exist to true
                    }
                }
            //catches SQL exception
            }catch (SQLException e){System.out.println(e.toString());}

         return doesExist;
    }

```



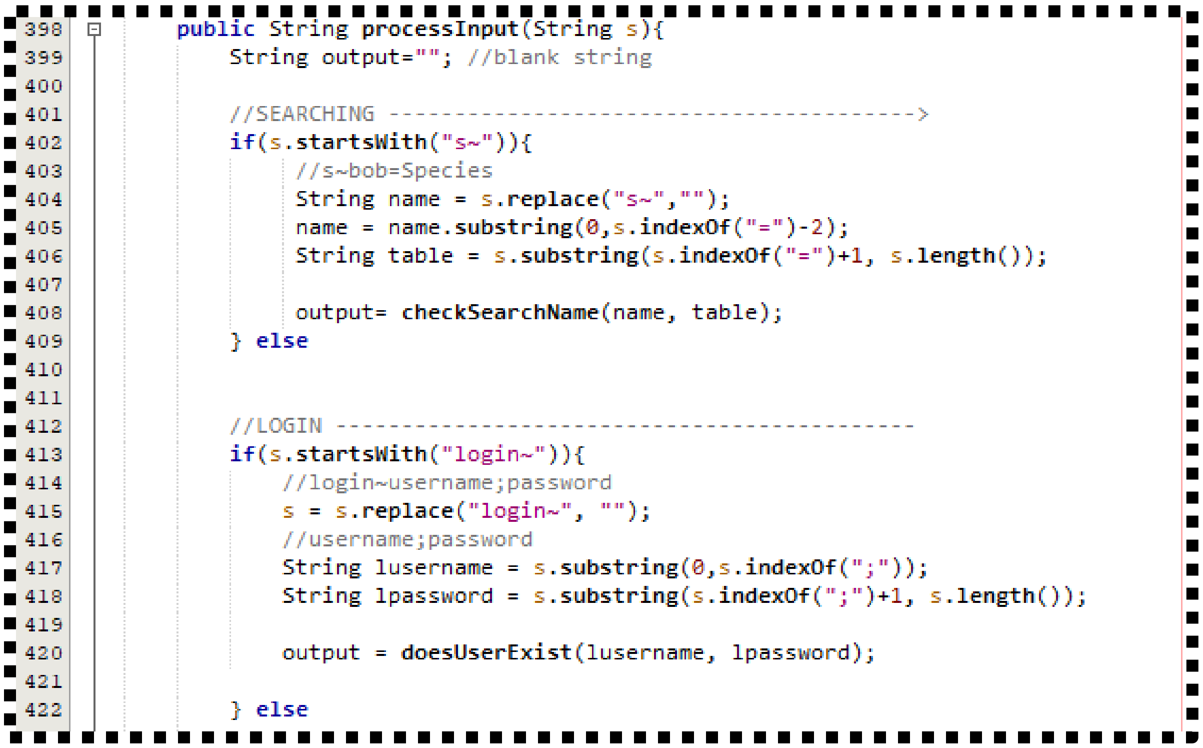
```
    public String insertIntoAnimals(String animalID, String animalName, String
            description, String speciesID){
        //Recivies parameters animalID, animalName, description, speciesID
        int rowsadded;  //rowsadded integer

    if(doesSpeciesExist(speciesID)){

        try{
            //creates statement object
            Statement statement = connection.createStatement();
            //insert query (SELF EXPLAINATORY)
            String query = "INSERT INTO Animals VALUES "
                    + "('"+animalID+"','"+animalName+"','"+description+"','"+
                    speciesID+"')";
            //executes query
            rowsadded = statement.executeUpdate(query);
            //catches SQL exception
        }catch (SQLException e){System.out.println(e.toString());}
        //returns message
        return "Successully added to Animals table!";
    } else {
        return "Species does not exist!";
    }

    }
```



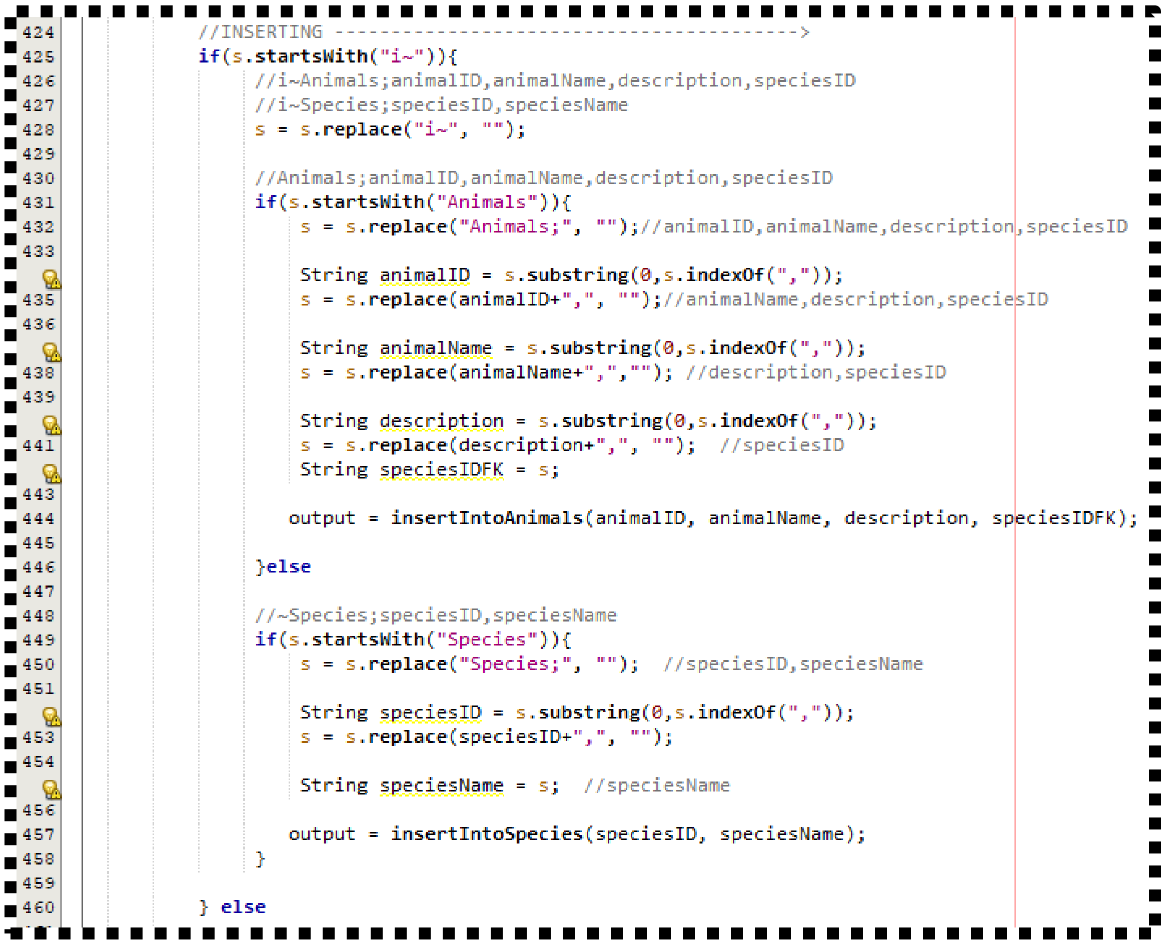
```java
    public String insertIntoSpecies(String speciesID, String speciesName){
        //Recivies parameters speciesID,speciesName
         int rowsadded; //rowsadded integer
         String result = ""; //blank string
         try{
            //creates statement object
            Statement statement = connection.createStatement();
            //insert query (SELF EXPLAINATORY)
            String insert = "INSERT INTO Species VALUES "
                    + "('"+speciesID+"','"+speciesName+"')";

            //display query (DISPLAYS speciesID only from Species table)
            String displaySpeciesID = "SELECT speciesID from Species";
            //executes display query
            ResultSet rec = statement.executeQuery(displaySpeciesID);

            boolean doesSpeciesExist=false; //flag varible
            while(rec.next()){//if species if found
                if(speciesID.equals(rec.getString("speciesID"))){
                    doesSpeciesExist = true;//set does exist to true
                }
            }
            //species does not exist then
            if(doesSpeciesExist==false){
            //execute insert query
            rowsadded = statement.executeUpdate(insert);
            //update result message
            result = "Successully added to Species table!";
            }else

            //else if it does exist then
            if(doesSpeciesExist==true){
                //update result message
                result = "Species already exists!";
            }

            //catches SQL exception
        }catch (SQLException e){System.out.println(e.toString());}
        //returns result
        return result;
    }
```



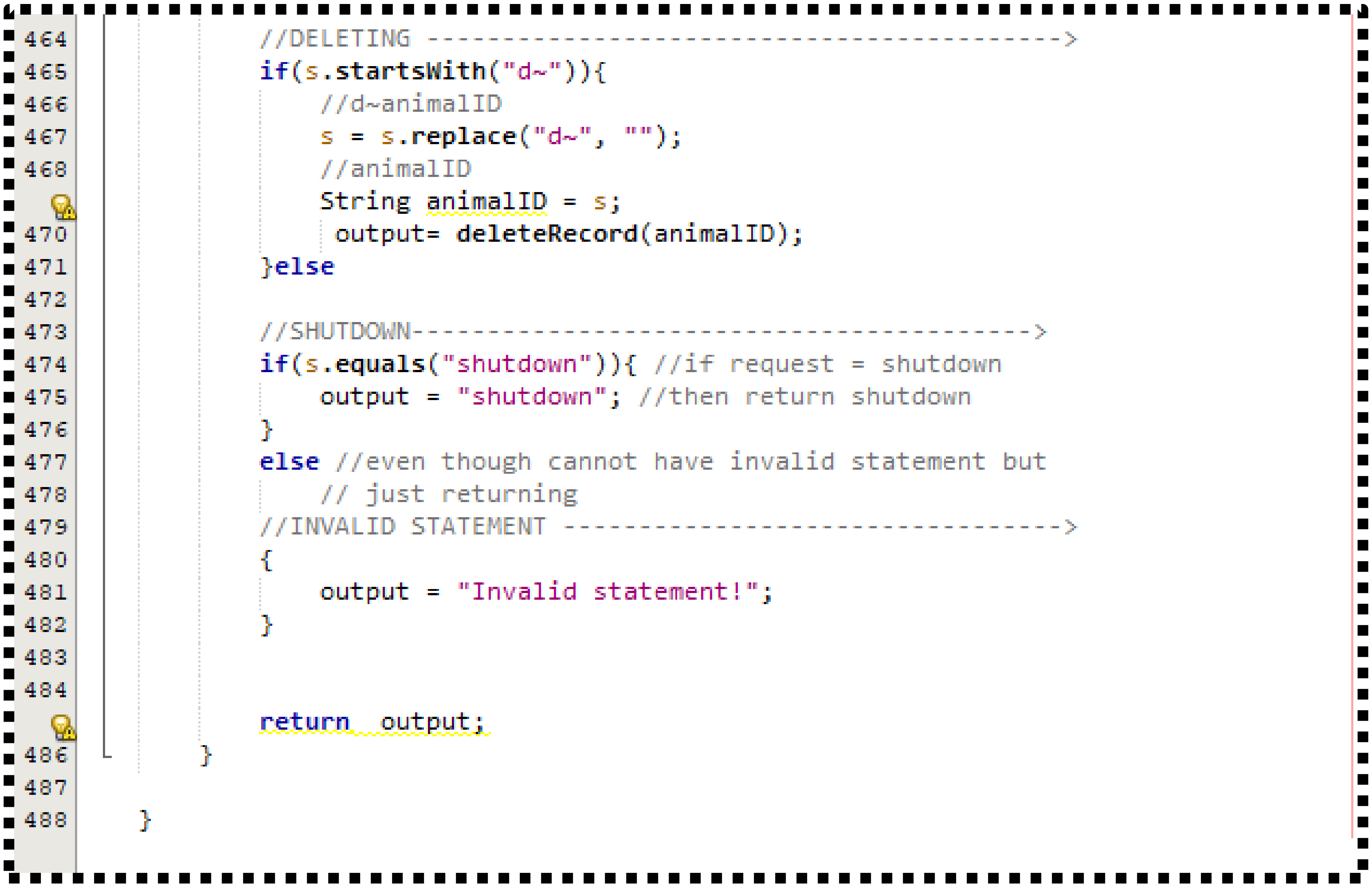
```java
    public String deleteRecord(String animalID){
        int rowsdeleted; //rowsadded integer
        String result="";//blank string
        try{
            //creates statement object
            Statement statement = connection.createStatement();
            //display query
            String displayAnimals = "Select * from Animals";
            //delete query including speciesID (input)
            String deleteQuery = "DELETE FROM Animals WHERE animalID='"+animalID
                    +"'";
            //executes displat query
            ResultSet rec = statement.executeQuery(displayAnimals);

            boolean doesExist=false; //flag variable
            while(rec.next()){//go through each record
                //if animalID is found
                if(animalID.equals(rec.getString("animalID"))){
                    doesExist = true;//set doesExist to true
                }
            }

            //if animal id exists
            if(doesExist==true){
                //the execute delete query
                rowsdeleted = statement.executeUpdate(deleteQuery);
                //return correct message
                result = "Successfully deleted from Animals table!";
            } else

            //if it does not exists
            if(doesExist==false){
                //return correct message
                result = "Animal does not exist!";
            }

            //catches SQL exception & prints exception message
        }catch (SQLException e){System.out.println(e.toString());}

        //returns result
        return result;
    }
```



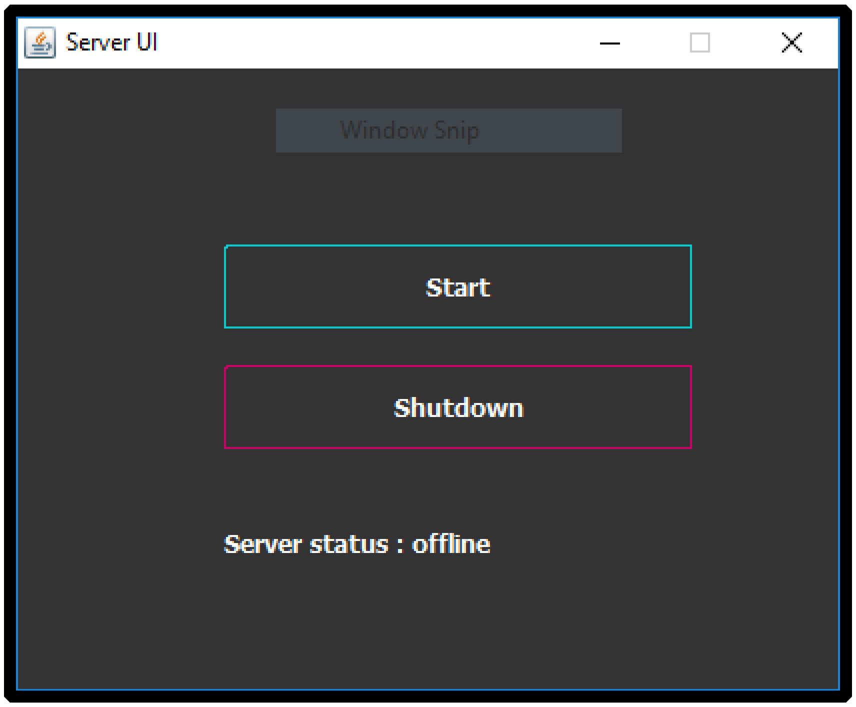
```
    public String processInput(String s){
        String output=""; //blank string

        //SEARCHING ---------------------------------------->
        if(s.startsWith("s~")){
             //s~bob=Species
             String name = s.replace("s~","");
             name = name.substring(0,s.indexOf("=")-2);
             String table = s.substring(s.indexOf("=")+1, s.length());

             output= checkSearchName(name, table);
        } else


        //LOGIN ---------------------------------------------
        if(s.startsWith("login~")){
            //login~username;password
            s = s.replace("login~", "");
            //username;password
            String lusername = s.substring(0,s.indexOf(";"));
            String lpassword = s.substring(s.indexOf(";")+1, s.length());

            output = doesUserExist(lusername, lpassword);

        } else
```



```
        //INSERTING ---------------------------------------->
        if(s.startsWith("i~")){
            //i~Animals;animalID,animalName,description,speciesID
            //i~Species;speciesID,speciesName
            s = s.replace("i~", "");

            //Animals;animalID,animalName,description,speciesID
            if(s.startsWith("Animals")){
                s = s.replace("Animals;", "");//animalID,animalName,description,speciesID

                String animalID = s.substring(0,s.indexOf(","));
                s = s.replace(animalID+",", "");//animalName,description,speciesID

                String animalName = s.substring(0,s.indexOf(","));
                s = s.replace(animalName+",",""); //description,speciesID

                String description = s.substring(0,s.indexOf(","));
                s = s.replace(description+",", "");  //speciesID
                String speciesIDFK = s;

               output = insertIntoAnimals(animalID, animalName, description, speciesIDFK);

            }else

            //~Species;speciesID,speciesName
            if(s.startsWith("Species")){
                s = s.replace("Species;", "");  //speciesID,speciesName

                String speciesID = s.substring(0,s.indexOf(","));
                s = s.replace(speciesID+",", "");

                String speciesName = s;  //speciesName

               output = insertIntoSpecies(speciesID, speciesName);
            }

        } else
```



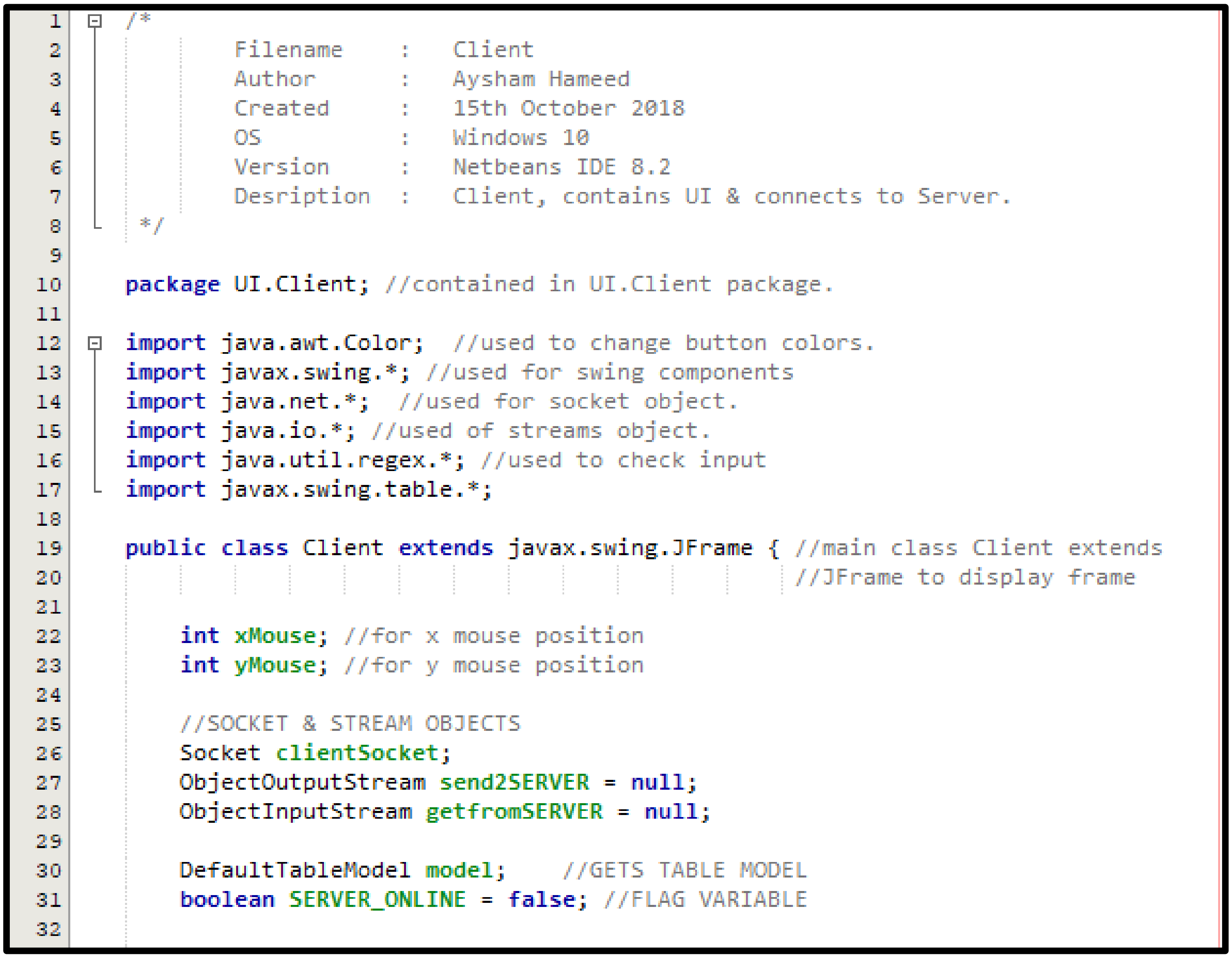
```
            //DELETING ---------------------------------------->
            if(s.startsWith("d~")){
                //d~animalID
                s = s.replace("d~", "");
                //animalID
                String animalID = s;
                 output= deleteRecord(animalID);
            }else

            //SHUTDOWN------------------------------------------>
            if(s.equals("shutdown")){ //if request = shutdown
                output = "shutdown"; //then return shutdown
            }
            else //even though cannot have invalid statement but
                // just returning
            //INVALID STATEMENT ---------------------------------->
            {
                output = "Invalid statement!";
            }


            return  output;
        }

    }
```



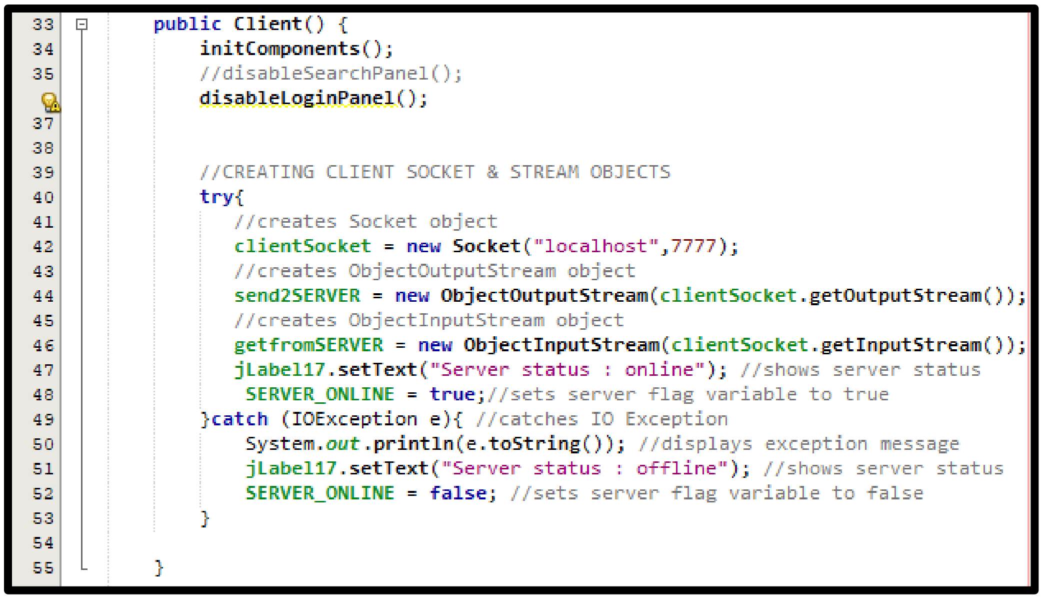
Server status : offline

# Client.

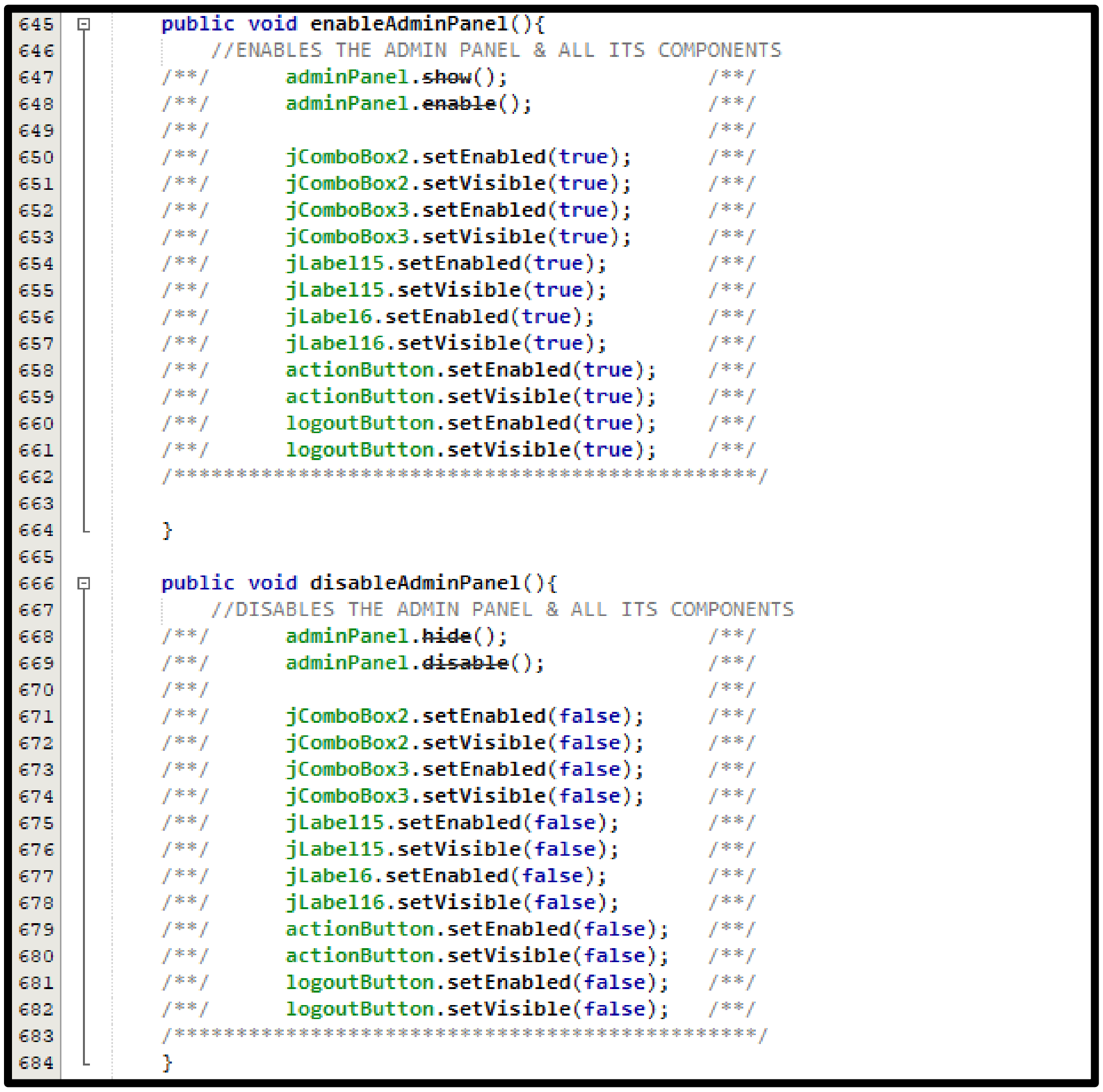
## Client UI Class



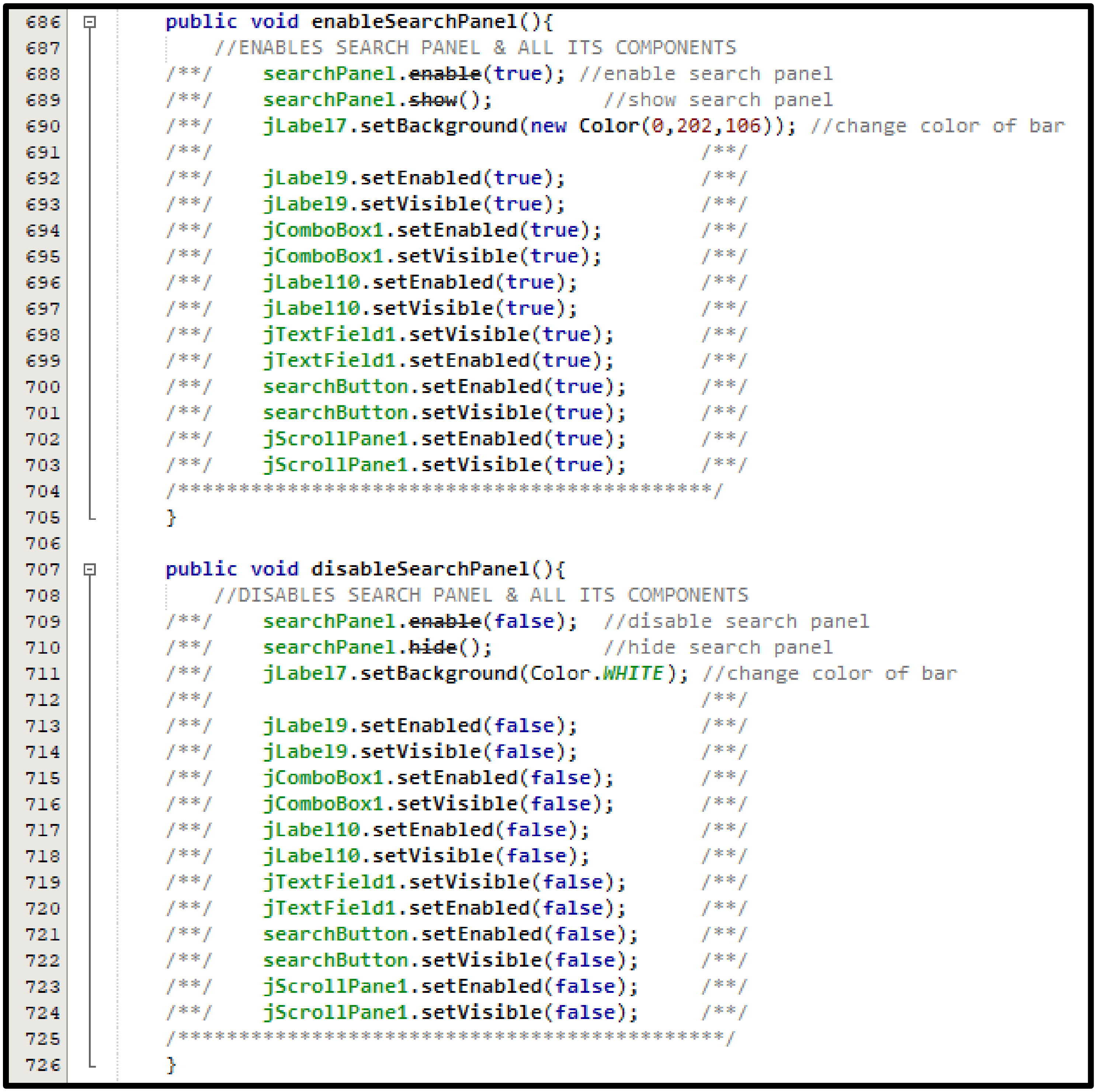
```java
/*
        Filename    :   Client
        Author      :   Aysham Hameed
        Created     :   15th October 2018
        OS          :   Windows 10
        Version     :   Netbeans IDE 8.2
        Desription  :   Client, contains UI & connects to Server.
 */

package UI.Client; //contained in UI.Client package.

import java.awt.Color;  //used to change button colors.
import javax.swing.*; //used for swing components
import java.net.*;  //used for socket object.
import java.io.*; //used of streams object.
import java.util.regex.*; //used to check input
import javax.swing.table.*;

public class Client extends javax.swing.JFrame { //main class Client extends
                                                 //JFrame to display frame

    int xMouse; //for x mouse position
    int yMouse; //for y mouse position

    //SOCKET & STREAM OBJECTS
    Socket clientSocket;
    ObjectOutputStream send2SERVER = null;
    ObjectInputStream getfromSERVER = null;

    DefaultTableModel model;    //GETS TABLE MODEL
    boolean SERVER_ONLINE = false; //FLAG VARIABLE

```



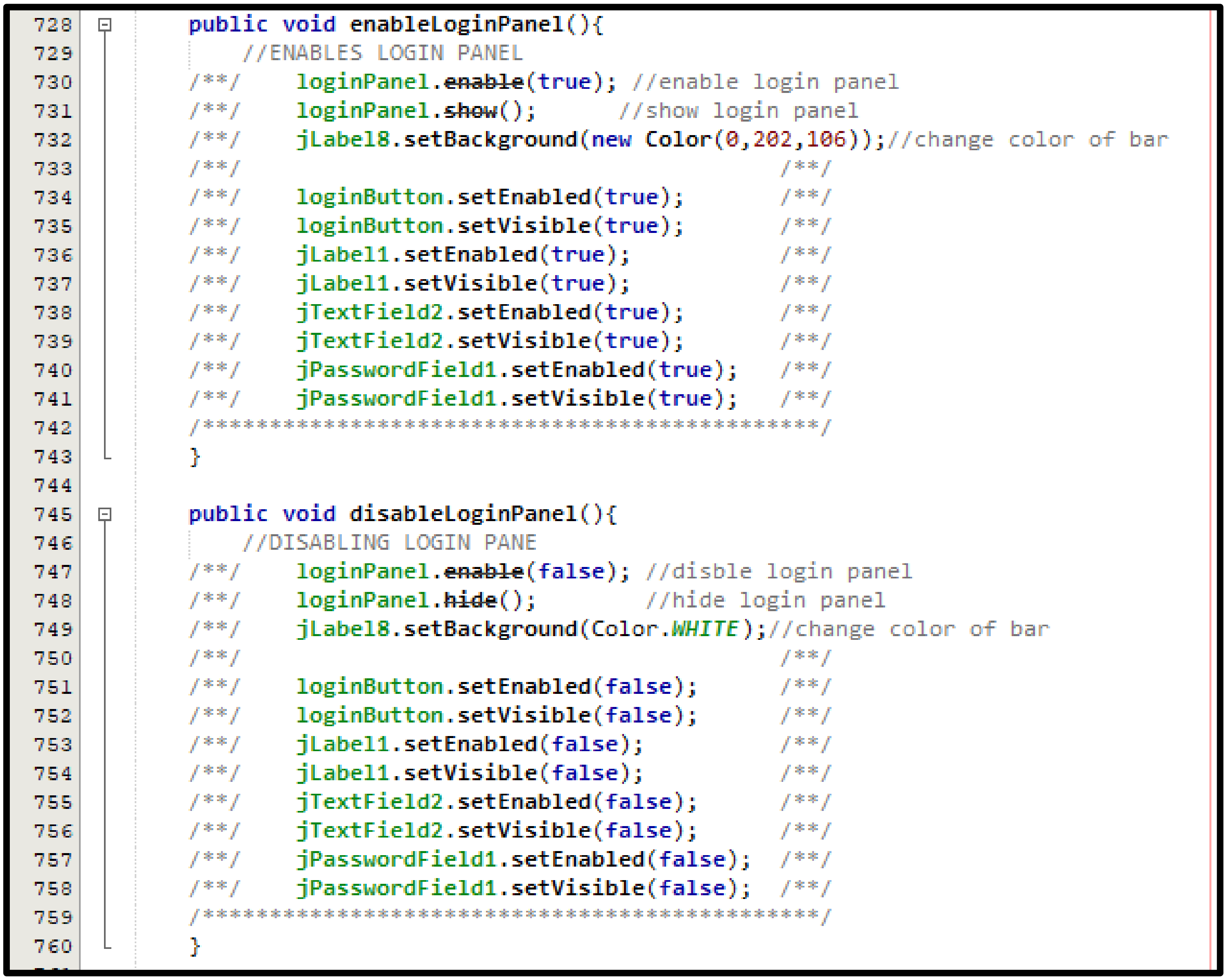
```java
    public Client() {
        initComponents();
        //disableSearchPanel();
        disableLoginPanel();


        //CREATING CLIENT SOCKET & STREAM OBJECTS
        try{
           //creates Socket object
           clientSocket = new Socket("localhost",7777);
           //creates ObjectOutputStream object
           send2SERVER = new ObjectOutputStream(clientSocket.getOutputStream());
           //creates ObjectInputStream object
           getfromSERVER = new ObjectInputStream(clientSocket.getInputStream());
           jLabel17.setText("Server status : online"); //shows server status
            SERVER_ONLINE = true;//sets server flag variable to true
        }catch (IOException e){ //catches IO Exception
            System.out.println(e.toString()); //displays exception message
            jLabel17.setText("Server status : offline"); //shows server status
            SERVER_ONLINE = false; //sets server flag variable to false
        }

    }
```



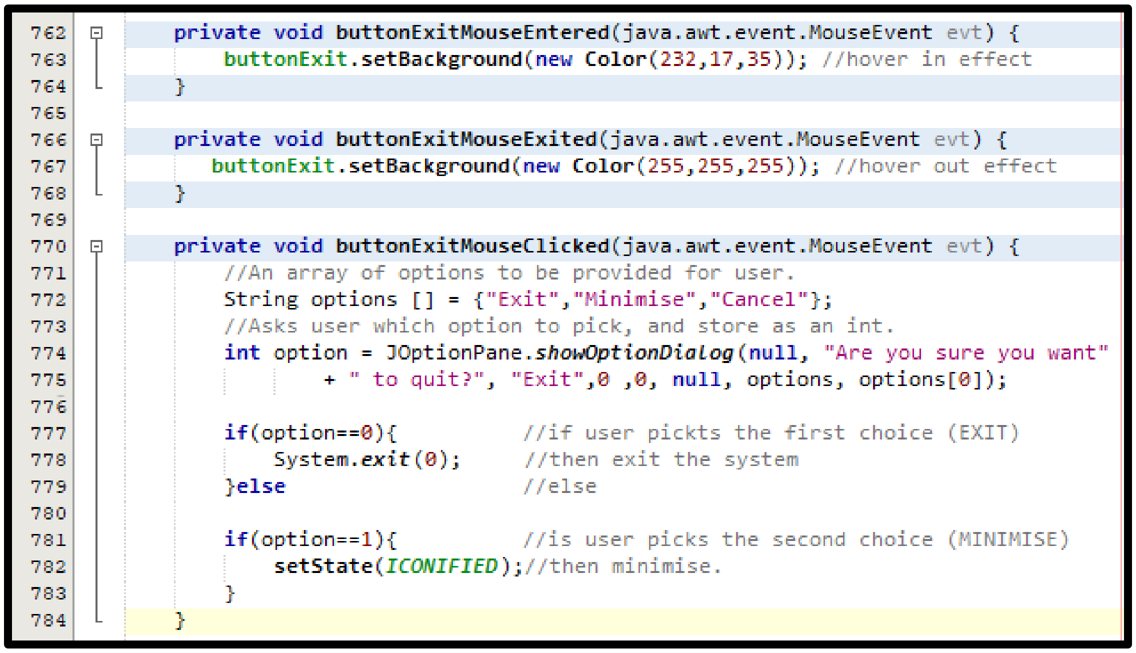
```java
    public void enableAdminPanel(){
        //ENABLES THE ADMIN PANEL & ALL ITS COMPONENTS
    /**/        adminPanel.~~show~~();                 /**/
    /**/        adminPanel.~~enable~~();               /**/
    /**/                                          /**/
    /**/        jComboBox2.setEnabled(true);      /**/
    /**/        jComboBox2.setVisible(true);      /**/
    /**/        jComboBox3.setEnabled(true);      /**/
    /**/        jComboBox3.setVisible(true);      /**/
    /**/        jLabel15.setEnabled(true);        /**/
    /**/        jLabel15.setVisible(true);        /**/
    /**/        jLabel6.setEnabled(true);         /**/
    /**/        jLabel16.setVisible(true);        /**/
    /**/        actionButton.setEnabled(true);    /**/
    /**/        actionButton.setVisible(true);    /**/
    /**/        logoutButton.setEnabled(true);    /**/
    /**/        logoutButton.setVisible(true);    /**/
    /**********************************************/

    }

    public void disableAdminPanel(){
        //DISABLES THE ADMIN PANEL & ALL ITS COMPONENTS
    /**/        adminPanel.~~hide~~();                 /**/
    /**/        adminPanel.~~disable~~();              /**/
    /**/                                          /**/
    /**/        jComboBox2.setEnabled(false);     /**/
    /**/        jComboBox2.setVisible(false);     /**/
    /**/        jComboBox3.setEnabled(false);     /**/
    /**/        jComboBox3.setVisible(false);     /**/
    /**/        jLabel15.setEnabled(false);       /**/
    /**/        jLabel15.setVisible(false);       /**/
    /**/        jLabel6.setEnabled(false);        /**/
    /**/        jLabel16.setVisible(false);       /**/
    /**/        actionButton.setEnabled(false);   /**/
    /**/        actionButton.setVisible(false);   /**/
    /**/        logoutButton.setEnabled(false);   /**/
    /**/        logoutButton.setVisible(false);   /**/
    /**********************************************/
    }
```



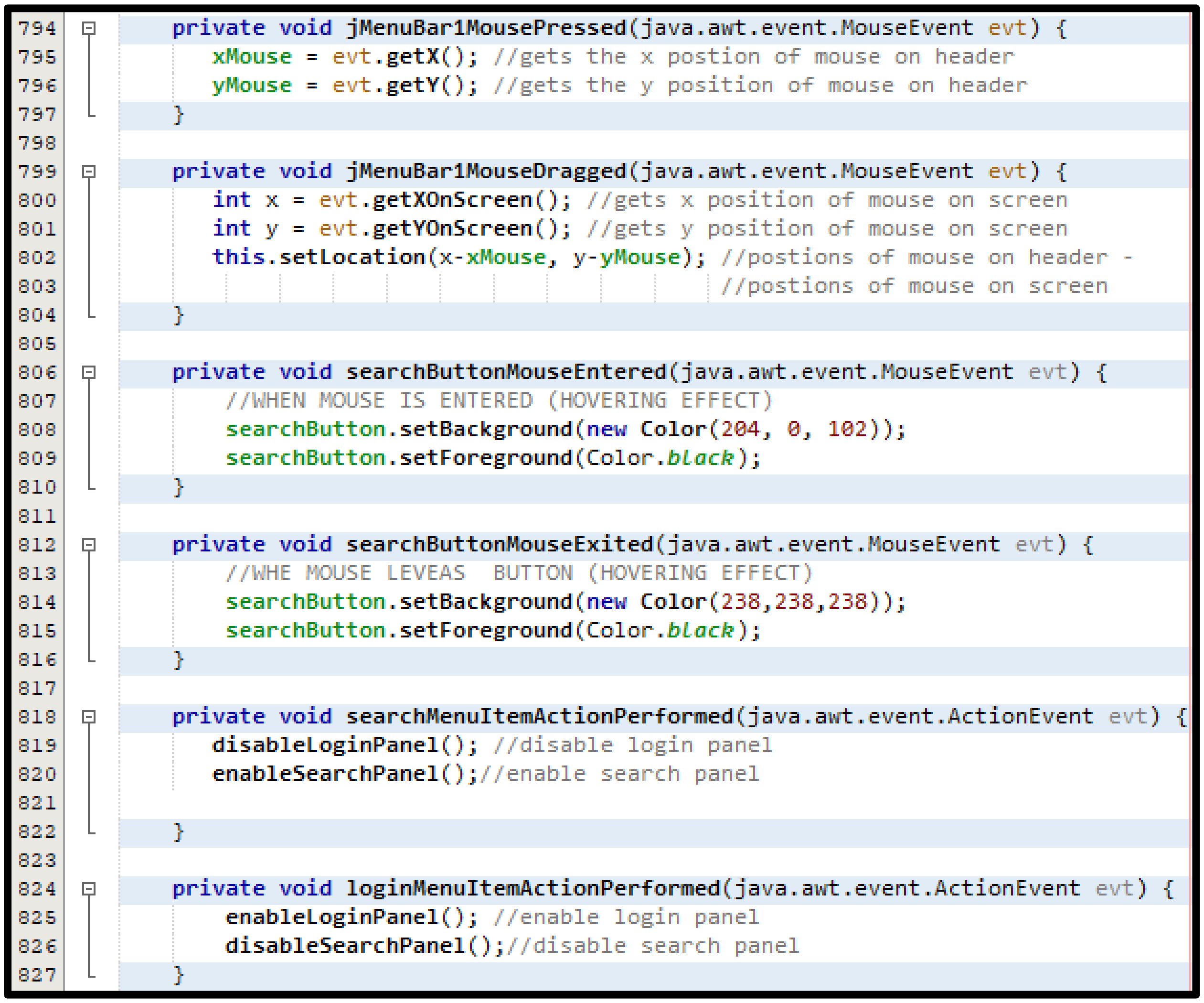
```
    public void enableSearchPanel(){
        //ENABLES SEARCH PANEL & ALL ITS COMPONENTS
    /**/    searchPanel.~~enable~~(true); //enable search panel
    /**/    searchPanel.~~show~~();         //show search panel
    /**/    jLabel7.setBackground(new Color(0,202,106)); //change color of bar
    /**/                                        /**/
    /**/    jLabel9.setEnabled(true);           /**/
    /**/    jLabel9.setVisible(true);           /**/
    /**/    jComboBox1.setEnabled(true);        /**/
    /**/    jComboBox1.setVisible(true);        /**/
    /**/    jLabel10.setEnabled(true);          /**/
    /**/    jLabel10.setVisible(true);          /**/
    /**/    jTextField1.setVisible(true);       /**/
    /**/    jTextField1.setEnabled(true);       /**/
    /**/    searchButton.setEnabled(true);      /**/
    /**/    searchButton.setVisible(true);      /**/
    /**/    jScrollPane1.setEnabled(true);      /**/
    /**/    jScrollPane1.setVisible(true);      /**/
    /********************************************/
    }

    public void disableSearchPanel(){
        //DISABLES SEARCH PANEL & ALL ITS COMPONENTS
    /**/    searchPanel.~~enable~~(false);  //disable search panel
    /**/    searchPanel.~~hide~~();         //hide search panel
    /**/    jLabel7.setBackground(Color.WHITE); //change color of bar
    /**/                                        /**/
    /**/    jLabel9.setEnabled(false);          /**/
    /**/    jLabel9.setVisible(false);          /**/
    /**/    jComboBox1.setEnabled(false);       /**/
    /**/    jComboBox1.setVisible(false);       /**/
    /**/    jLabel10.setEnabled(false);         /**/
    /**/    jLabel10.setVisible(false);         /**/
    /**/    jTextField1.setVisible(false);      /**/
    /**/    jTextField1.setEnabled(false);      /**/
    /**/    searchButton.setEnabled(false);     /**/
    /**/    searchButton.setVisible(false);     /**/
    /**/    jScrollPane1.setEnabled(false);     /**/
    /**/    jScrollPane1.setVisible(false);     /**/
    /*********************************************/
    }
```



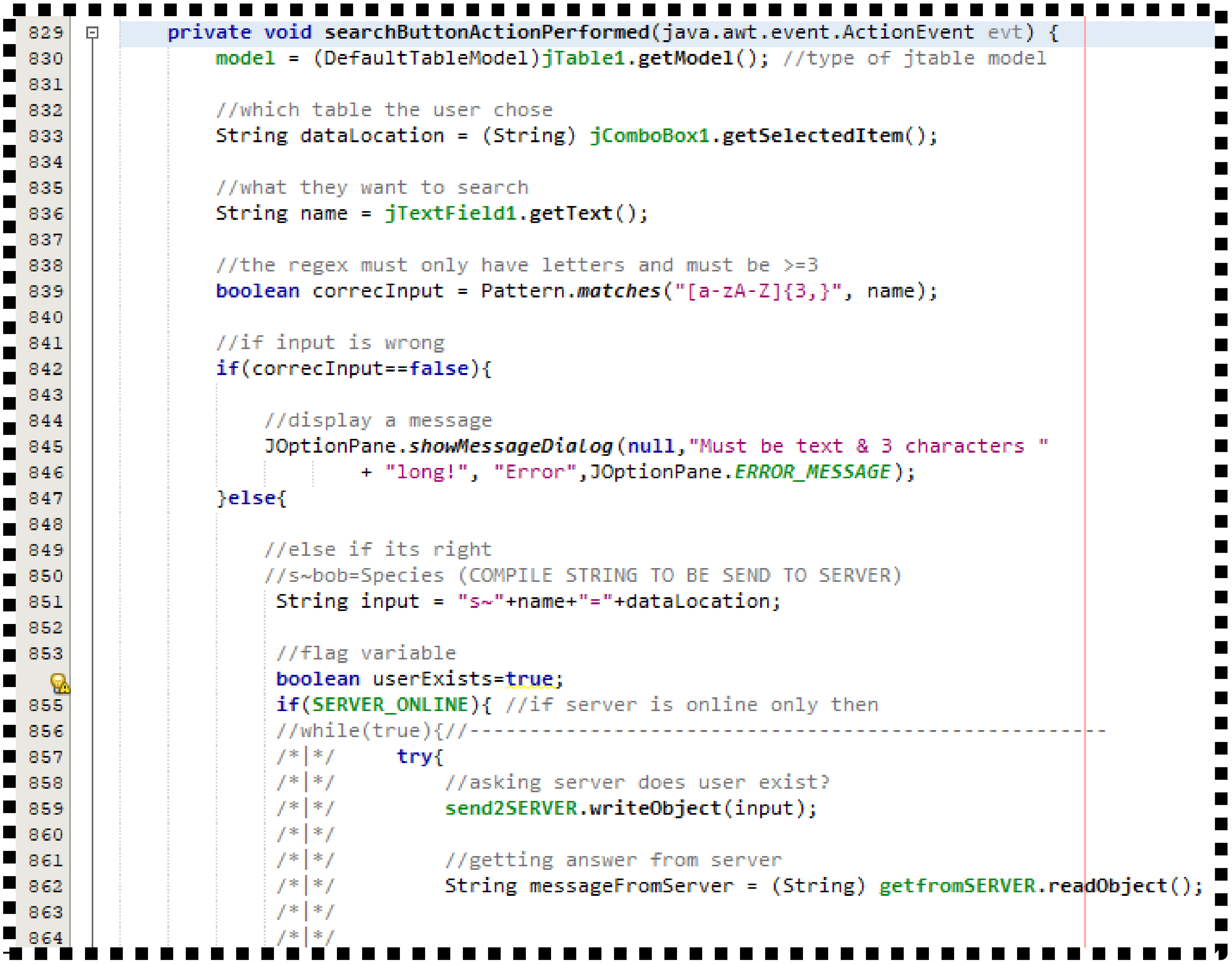
```
    public void enableLoginPanel(){
        //ENABLES LOGIN PANEL
    /**/    loginPanel.~~enable~~(true); //enable login panel
    /**/    loginPanel.~~show~~();      //show login panel
    /**/    jLabel8.setBackground(new Color(0,202,106));//change color of bar
    /**/                                        /**/
    /**/    loginButton.setEnabled(true);       /**/
    /**/    loginButton.setVisible(true);       /**/
    /**/    jLabel1.setEnabled(true);           /**/
    /**/    jLabel1.setVisible(true);           /**/
    /**/    jTextField2.setEnabled(true);       /**/
    /**/    jTextField2.setVisible(true);       /**/
    /**/    jPasswordField1.setEnabled(true);   /**/
    /**/    jPasswordField1.setVisible(true);   /**/
    /**********************************************/
    }

    public void disableLoginPanel(){
        //DISABLING LOGIN PANE
    /**/    loginPanel.~~enable~~(false); //disble login panel
    /**/    loginPanel.~~hide~~();        //hide login panel
    /**/    jLabel8.setBackground(Color.WHITE);//change color of bar
    /**/                                        /**/
    /**/    loginButton.setEnabled(false);      /**/
    /**/    loginButton.setVisible(false);      /**/
    /**/    jLabel1.setEnabled(false);          /**/
    /**/    jLabel1.setVisible(false);          /**/
    /**/    jTextField2.setEnabled(false);      /**/
    /**/    jTextField2.setVisible(false);      /**/
    /**/    jPasswordField1.setEnabled(false);  /**/
    /**/    jPasswordField1.setVisible(false);  /**/
    /**********************************************/
    }
```



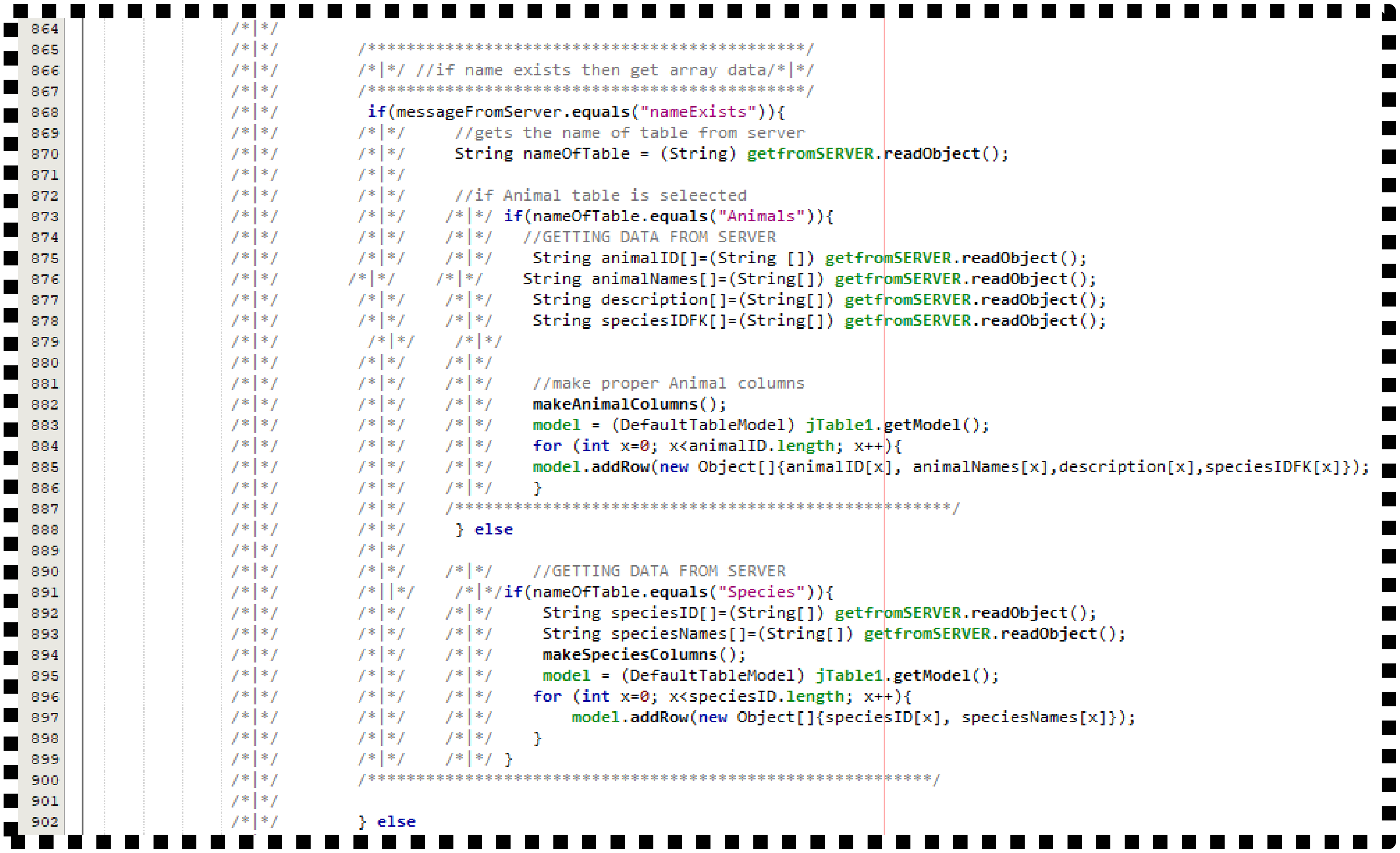
```
    private void buttonExitMouseEntered(java.awt.event.MouseEvent evt) {
        buttonExit.setBackground(new Color(232,17,35)); //hover in effect
    }

    private void buttonExitMouseExited(java.awt.event.MouseEvent evt) {
       buttonExit.setBackground(new Color(255,255,255)); //hover out effect
    }

    private void buttonExitMouseClicked(java.awt.event.MouseEvent evt) {
        //An array of options to be provided for user.
        String options [] = {"Exit","Minimise","Cancel"};
        //Asks user which option to pick, and store as an int.
        int option = JOptionPane.showOptionDialog(null, "Are you sure you want"
                + " to quit?", "Exit",0 ,0, null, options, options[0]);

        if(option==0){          //if user pickts the first choice (EXIT)
            System.exit(0);     //then exit the system
        }else                   //else

        if(option==1){          //is user picks the second choice (MINIMISE)
            setState(ICONIFIED);//then minimise.
        }
    }
```



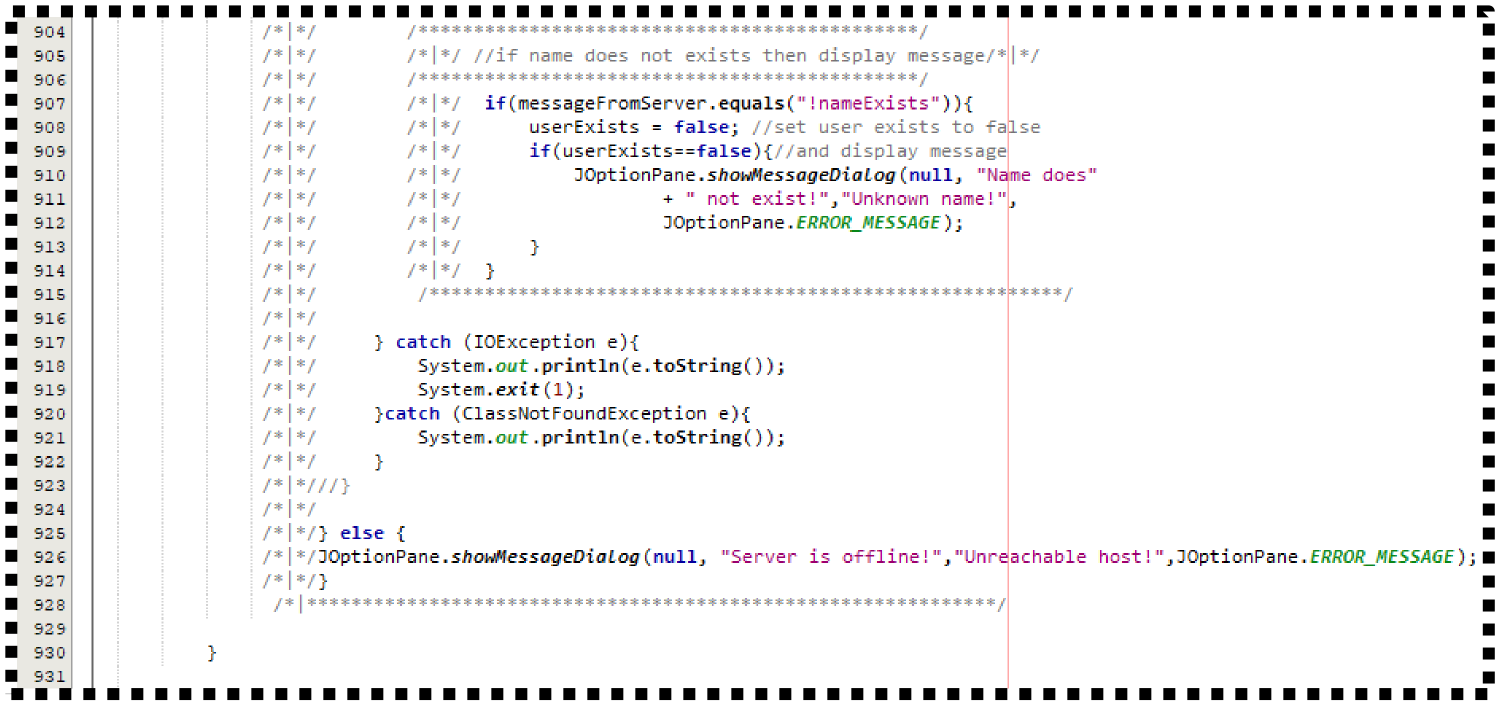
```
    private void jMenuBar1MousePressed(java.awt.event.MouseEvent evt) {
       xMouse = evt.getX(); //gets the x postion of mouse on header
       yMouse = evt.getY(); //gets the y position of mouse on header
    }

    private void jMenuBar1MouseDragged(java.awt.event.MouseEvent evt) {
       int x = evt.getXOnScreen(); //gets x position of mouse on screen
       int y = evt.getYOnScreen(); //gets y position of mouse on screen
       this.setLocation(x-xMouse, y-yMouse); //postions of mouse on header -
                                             //postions of mouse on screen
    }

    private void searchButtonMouseEntered(java.awt.event.MouseEvent evt) {
        //WHEN MOUSE IS ENTERED (HOVERING EFFECT)
        searchButton.setBackground(new Color(204, 0, 102));
        searchButton.setForeground(Color.black);
    }

    private void searchButtonMouseExited(java.awt.event.MouseEvent evt) {
        //WHE MOUSE LEVEAS  BUTTON (HOVERING EFFECT)
        searchButton.setBackground(new Color(238,238,238));
        searchButton.setForeground(Color.black);
    }

    private void searchMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
       disableLoginPanel(); //disable login panel
       enableSearchPanel();//enable search panel

    }

    private void loginMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        enableLoginPanel(); //enable login panel
        disableSearchPanel();//disable search panel
    }
```



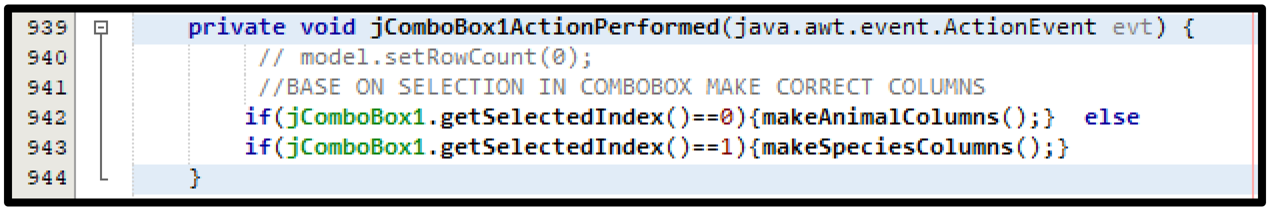
```
    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
        model = (DefaultTableModel)jTable1.getModel(); //type of jtable model

        //which table the user chose
        String dataLocation = (String) jComboBox1.getSelectedItem();

        //what they want to search
        String name = jTextField1.getText();

        //the regex must only have letters and must be >=3
        boolean correcInput = Pattern.matches("[a-zA-Z]{3,}", name);

        //if input is wrong
        if(correcInput==false){

            //display a message
            JOptionPane.showMessageDialog(null,"Must be text & 3 characters "
                    + "long!", "Error",JOptionPane.ERROR_MESSAGE);
        }else{

            //else if its right
            //s~bob=Species (COMPILE STRING TO BE SEND TO SERVER)
             String input = "s~"+name+"="+dataLocation;

             //flag variable
             boolean userExists=true;
             if(SERVER_ONLINE){ //if server is online only then
             //while(true){//----------------------------------------------------
             /*|*/     try{
             /*|*/         //asking server does user exist?
             /*|*/         send2SERVER.writeObject(input);
             /*|*/
             /*|*/         //getting answer from server
             /*|*/         String messageFromServer = (String) getfromSERVER.readObject();
             /*|*/
             /*|*/
```



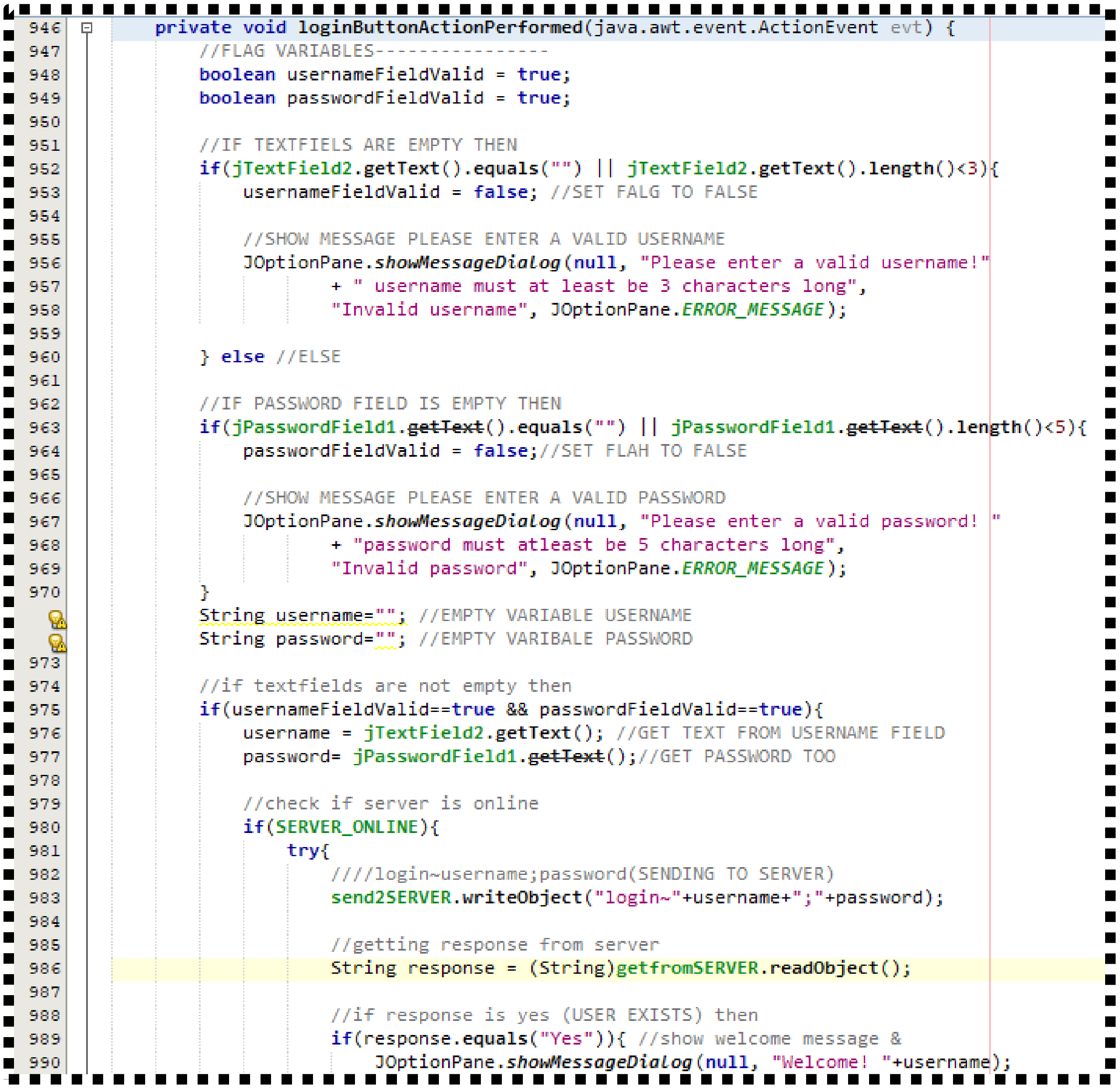
```
/*|*/
/*|*/        /***********************************************/
/*|*/        /*|*/ //if name exists then get array data/*|*/
/*|*/        /***********************************************/
/*|*/         if(messageFromServer.equals("nameExists")){
/*|*/        /*|*/     //gets the name of table from server
/*|*/        /*|*/     String nameOfTable = (String) getfromSERVER.readObject();
/*|*/        /*|*/
/*|*/        /*|*/     //if Animal table is seleceted
/*|*/        /*|*/    /*|*/ if(nameOfTable.equals("Animals")){
/*|*/        /*|*/    /*|*/   //GETTING DATA FROM SERVER
/*|*/        /*|*/    /*|*/    String animalID[]=(String []) getfromSERVER.readObject();
/*|*/       /*|*/    /*|*/    String animalNames[]=(String[]) getfromSERVER.readObject();
/*|*/        /*|*/    /*|*/    String description[]=(String[]) getfromSERVER.readObject();
/*|*/        /*|*/    /*|*/    String speciesIDFK[]=(String[]) getfromSERVER.readObject();
/*|*/         /*|*/    /*|*/
/*|*/        /*|*/    /*|*/
/*|*/        /*|*/    /*|*/    //make proper Animal columns
/*|*/        /*|*/    /*|*/    makeAnimalColumns();
/*|*/        /*|*/    /*|*/    model = (DefaultTableModel) jTable1.getModel();
/*|*/        /*|*/    /*|*/    for (int x=0; x<animalID.length; x++){
/*|*/        /*|*/    /*|*/    model.addRow(new Object[]{animalID[x], animalNames[x],description[x],speciesIDFK[x]});
/*|*/        /*|*/    /*|*/    }
/*|*/        /*|*/    /****************************************************/
/*|*/        /*|*/     } else
/*|*/        /*|*/
/*|*/        /*|*/    /*|*/    //GETTING DATA FROM SERVER
/*|*/        /*||*/    /*|*/if(nameOfTable.equals("Species")){
/*|*/        /*|*/    /*|*/     String speciesID[]=(String[]) getfromSERVER.readObject();
/*|*/        /*|*/    /*|*/     String speciesNames[]=(String[]) getfromSERVER.readObject();
/*|*/        /*|*/    /*|*/     makeSpeciesColumns();
/*|*/        /*|*/    /*|*/     model = (DefaultTableModel) jTable1.getModel();
/*|*/        /*|*/    /*|*/    for (int x=0; x<speciesID.length; x++){
/*|*/        /*|*/    /*|*/        model.addRow(new Object[]{speciesID[x], speciesNames[x]});
/*|*/        /*|*/    /*|*/    }
/*|*/        /*|*/    /*|*/ }
/*|*/        /************************************************************/
/*|*/
/*|*/        } else
```



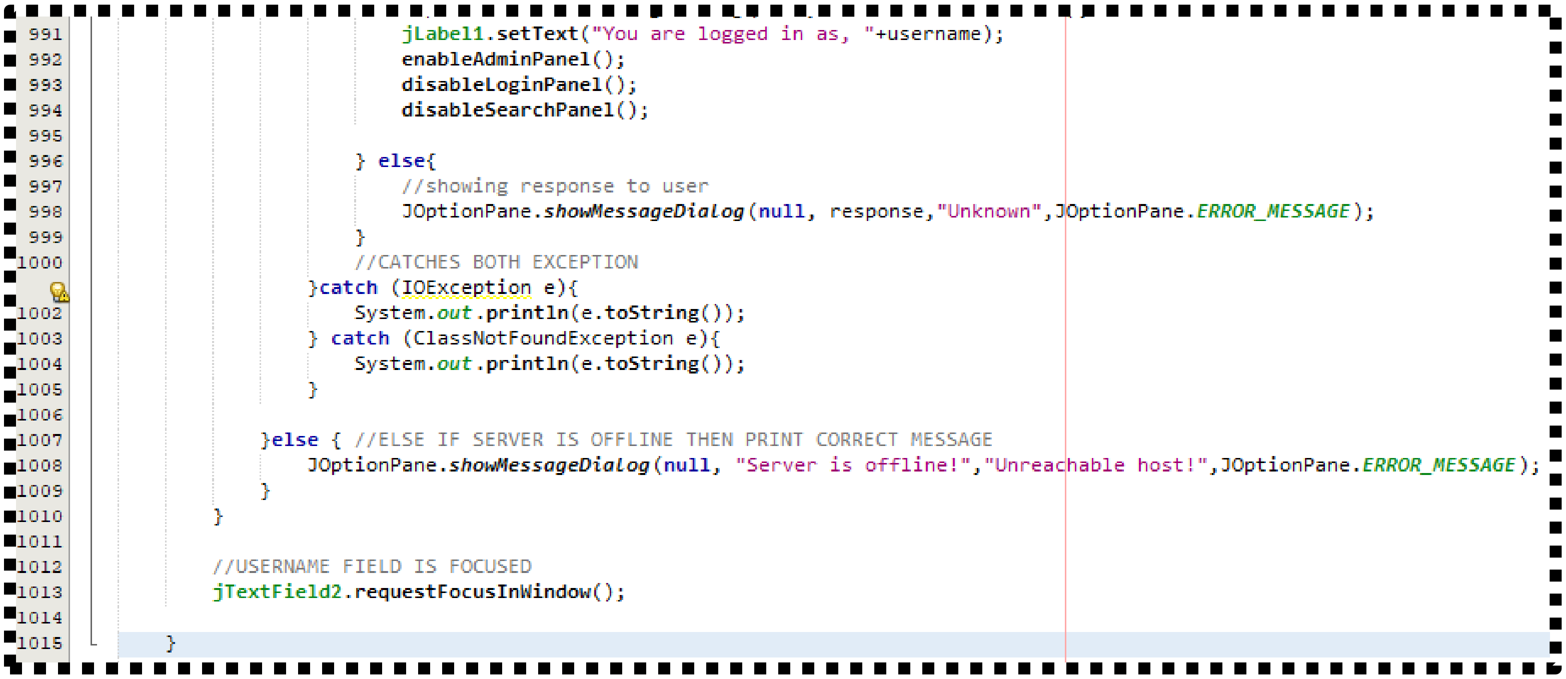
```
/*|*/        /***********************************************/
/*|*/        /*|*/ //if name does not exists then display message/*|*/
/*|*/        /***********************************************/
/*|*/        /*|*/  if(messageFromServer.equals("!nameExists")){
/*|*/        /*|*/      userExists = false; //set user exists to false
/*|*/        /*|*/      if(userExists==false){//and display message
/*|*/        /*|*/          JOptionPane.showMessageDialog(null, "Name does"
/*|*/        /*|*/                  + " not exist!","Unknown name!",
/*|*/        /*|*/                  JOptionPane.ERROR_MESSAGE);
/*|*/        /*|*/      }
/*|*/        /*|*/  }
/*|*/         /************************************************************/
/*|*/
/*|*/     } catch (IOException e){
/*|*/         System.out.println(e.toString());
/*|*/         System.exit(1);
/*|*/     }catch (ClassNotFoundException e){
/*|*/         System.out.println(e.toString());
/*|*/     }
/*|*///}
/*|*/
/*|*/} else {
/*|*/JOptionPane.showMessageDialog(null, "Server is offline!","Unreachable host!",JOptionPane.ERROR_MESSAGE);
/*|*/}
 /*|*************************************************************/

   }
```



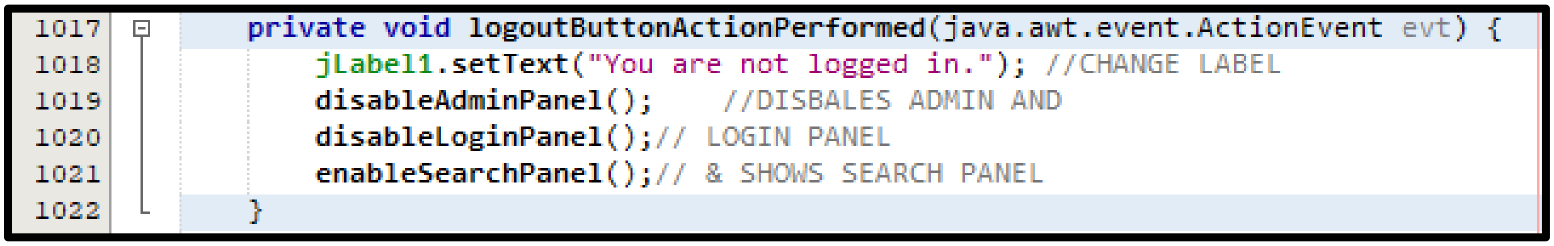
```
    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
         // model.setRowCount(0);
         //BASE ON SELECTION IN COMBOBOX MAKE CORRECT COLUMNS
        if(jComboBox1.getSelectedIndex()==0){makeAnimalColumns();}  else
        if(jComboBox1.getSelectedIndex()==1){makeSpeciesColumns();}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



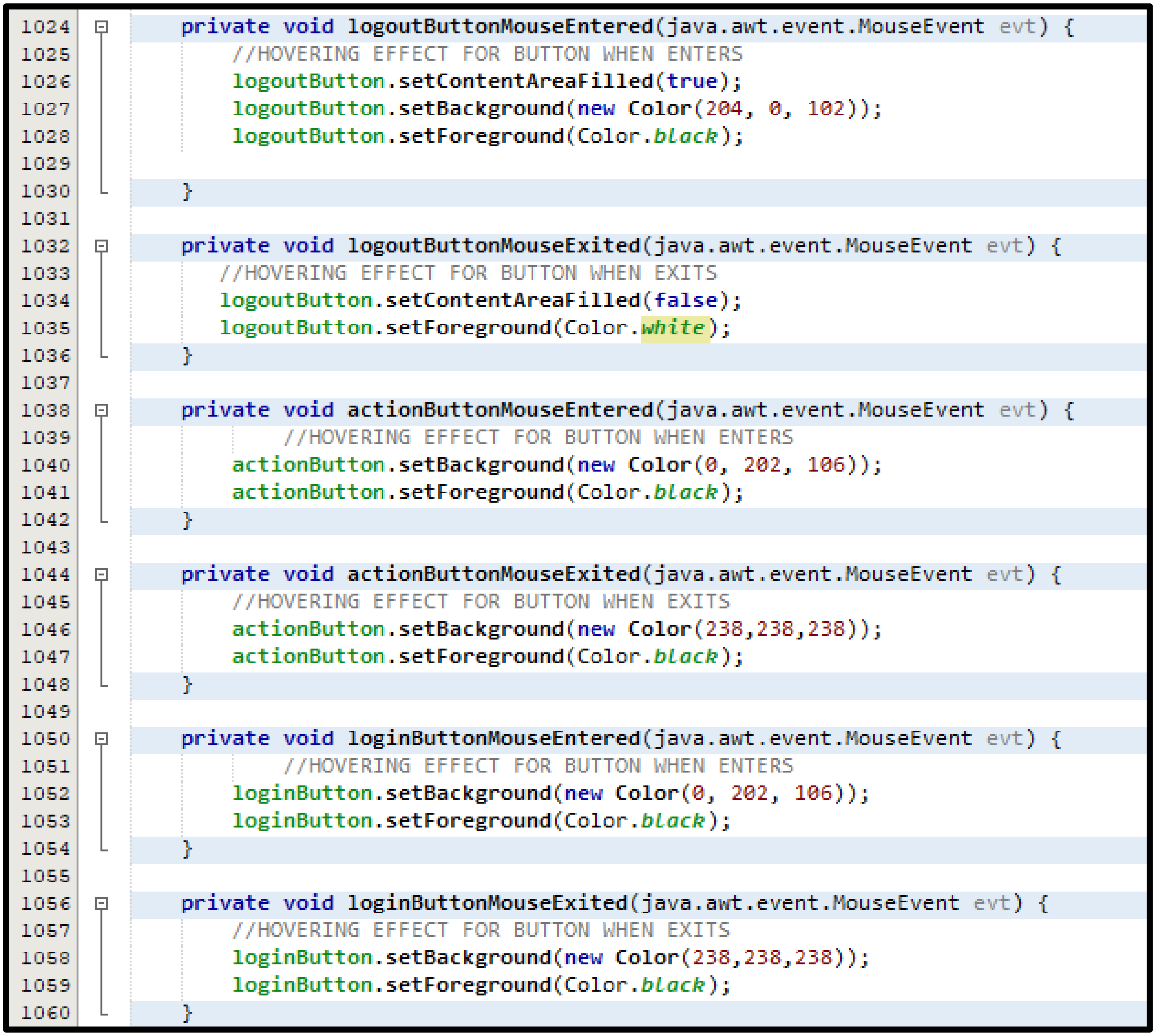
```
    private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //FLAG VARIABLES----------------
        boolean usernameFieldValid = true;
        boolean passwordFieldValid = true;

        //IF TEXTFIELS ARE EMPTY THEN
        if(jTextField2.getText().equals("") || jTextField2.getText().length()<3){
            usernameFieldValid = false; //SET FALG TO FALSE

            //SHOW MESSAGE PLEASE ENTER A VALID USERNAME
            JOptionPane.showMessageDialog(null, "Please enter a valid username!"
                    + " username must at least be 3 characters long",
                    "Invalid username", JOptionPane.ERROR_MESSAGE);

        } else //ELSE

        //IF PASSWORD FIELD IS EMPTY THEN
        if(jPasswordField1.~~getText~~().equals("") || jPasswordField1.~~getText~~().length()<5){
            passwordFieldValid = false;//SET FLAH TO FALSE

            //SHOW MESSAGE PLEASE ENTER A VALID PASSWORD
            JOptionPane.showMessageDialog(null, "Please enter a valid password! "
                    + "password must atleast be 5 characters long",
                    "Invalid password", JOptionPane.ERROR_MESSAGE);
        }
        String username=""; //EMPTY VARIABLE USERNAME
        String password=""; //EMPTY VARIBALE PASSWORD

        //if textfields are not empty then
        if(usernameFieldValid==true && passwordFieldValid==true){
            username = jTextField2.getText(); //GET TEXT FROM USERNAME FIELD
            password= jPasswordField1.~~getText~~();//GET PASSWORD TOO

            //check if server is online
            if(SERVER_ONLINE){
                try{
                    ////login~username;password(SENDING TO SERVER)
                    send2SERVER.writeObject("login~"+username+";"+password);

                    //getting response from server
                    String response = (String)getfromSERVER.readObject();

                    //if response is yes (USER EXISTS) then
                    if(response.equals("Yes")){ //show welcome message &
                        JOptionPane.showMessageDialog(null, "Welcome! "+username);
```



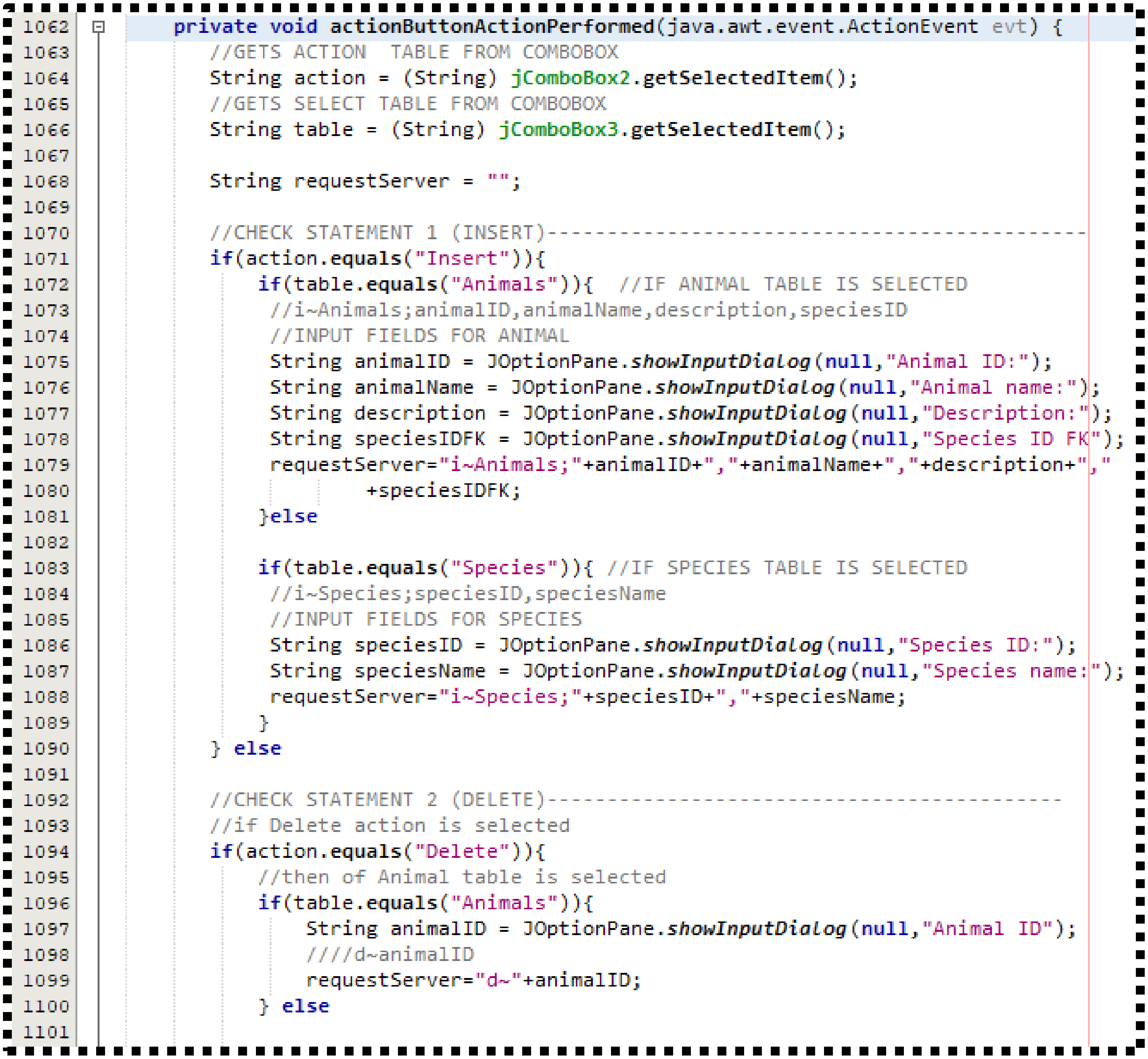
```
                        jLabel1.setText("You are logged in as, "+username);
                        enableAdminPanel();
                        disableLoginPanel();
                        disableSearchPanel();

                    } else{
                        //showing response to user
                        JOptionPane.showMessageDialog(null, response,"Unknown",JOptionPane.ERROR_MESSAGE);
                    }
                    //CATCHES BOTH EXCEPTION
                }catch (IOException e){
                    System.out.println(e.toString());
                } catch (ClassNotFoundException e){
                    System.out.println(e.toString());
                }

            }else { //ELSE IF SERVER IS OFFLINE THEN PRINT CORRECT MESSAGE
                JOptionPane.showMessageDialog(null, "Server is offline!","Unreachable host!",JOptionPane.ERROR_MESSAGE);
            }
        }

        //USERNAME FIELD IS FOCUSED
        jTextField2.requestFocusInWindow();

    }
```



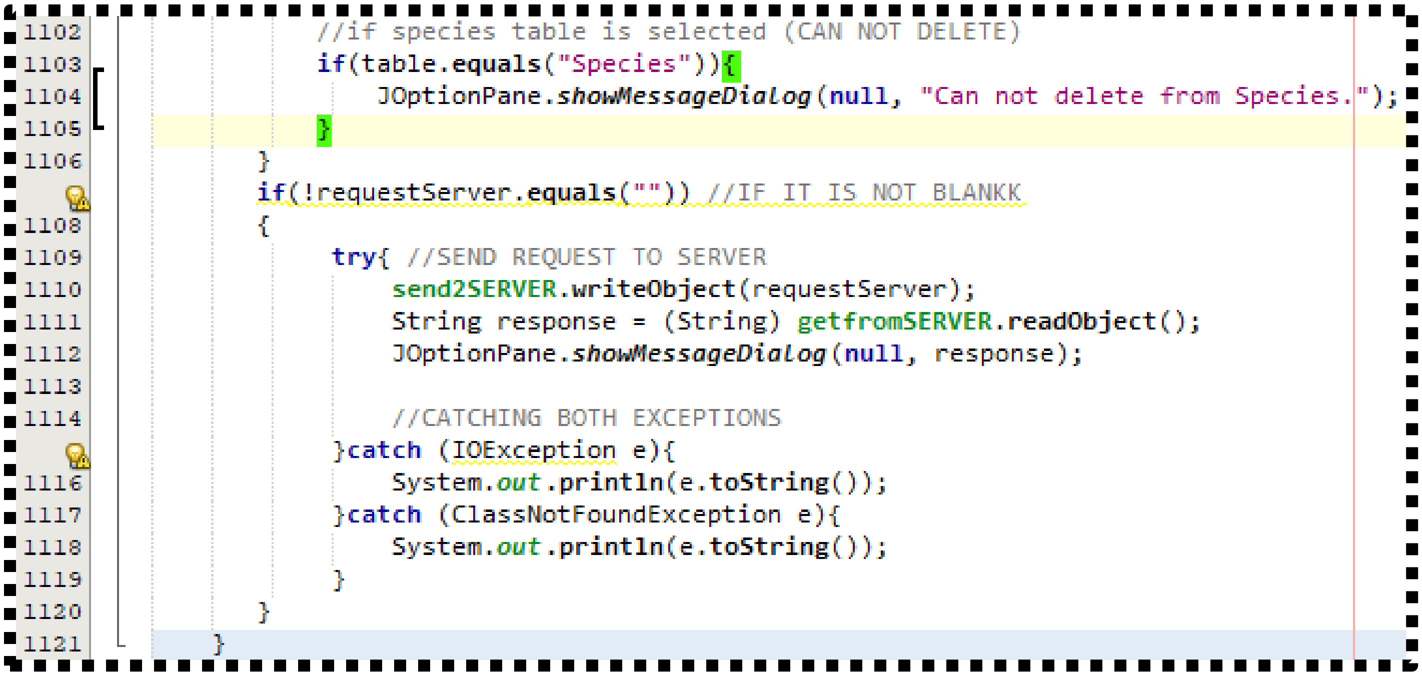
```
    private void logoutButtonActionPerformed(java.awt.event.ActionEvent evt) {
        jLabel1.setText("You are not logged in."); //CHANGE LABEL
        disableAdminPanel();    //DISBALES ADMIN AND
        disableLoginPanel();// LOGIN PANEL
        enableSearchPanel();// & SHOWS SEARCH PANEL
   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



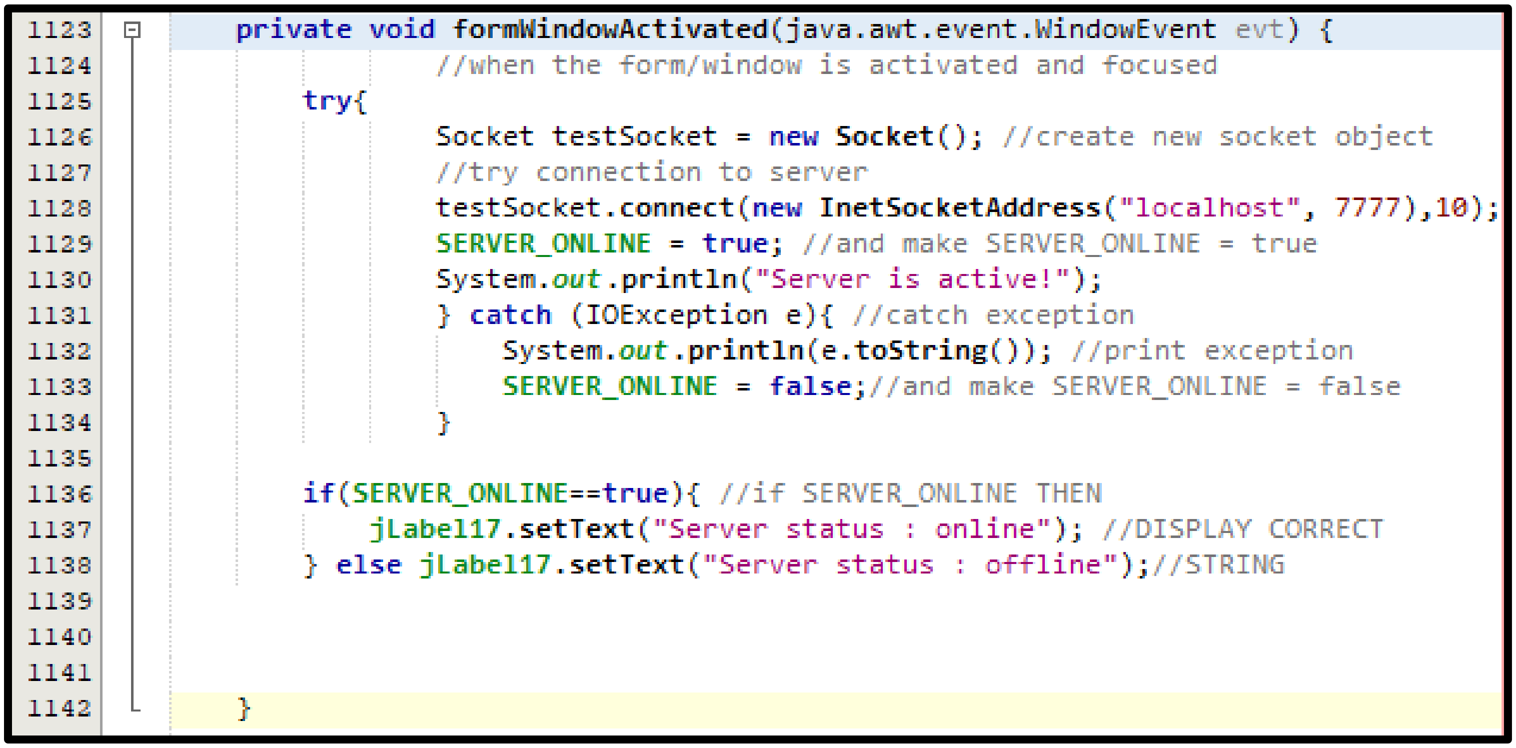
```java
    private void logoutButtonMouseEntered(java.awt.event.MouseEvent evt) {
        //HOVERING EFFECT FOR BUTTON WHEN ENTERS
        logoutButton.setContentAreaFilled(true);
        logoutButton.setBackground(new Color(204, 0, 102));
        logoutButton.setForeground(Color.black);

    }

    private void logoutButtonMouseExited(java.awt.event.MouseEvent evt) {
       //HOVERING EFFECT FOR BUTTON WHEN EXITS
       logoutButton.setContentAreaFilled(false);
       logoutButton.setForeground(Color.white);
    }

    private void actionButtonMouseEntered(java.awt.event.MouseEvent evt) {
            //HOVERING EFFECT FOR BUTTON WHEN ENTERS
        actionButton.setBackground(new Color(0, 202, 106));
        actionButton.setForeground(Color.black);
    }

    private void actionButtonMouseExited(java.awt.event.MouseEvent evt) {
        //HOVERING EFFECT FOR BUTTON WHEN EXITS
        actionButton.setBackground(new Color(238,238,238));
        actionButton.setForeground(Color.black);
    }

    private void loginButtonMouseEntered(java.awt.event.MouseEvent evt) {
            //HOVERING EFFECT FOR BUTTON WHEN ENTERS
        loginButton.setBackground(new Color(0, 202, 106));
        loginButton.setForeground(Color.black);
    }

    private void loginButtonMouseExited(java.awt.event.MouseEvent evt) {
        //HOVERING EFFECT FOR BUTTON WHEN EXITS
        loginButton.setBackground(new Color(238,238,238));
        loginButton.setForeground(Color.black);
    }
```



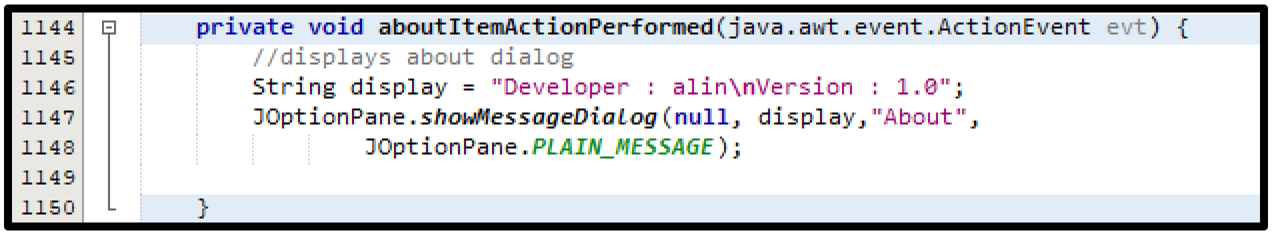
```
    private void actionButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //GETS ACTION  TABLE FROM COMBOBOX
        String action = (String) jComboBox2.getSelectedItem();
        //GETS SELECT TABLE FROM COMBOBOX
        String table = (String) jComboBox3.getSelectedItem();

        String requestServer = "";

        //CHECK STATEMENT 1 (INSERT)----------------------------------------------
        if(action.equals("Insert")){
            if(table.equals("Animals")){  //IF ANIMAL TABLE IS SELECTED
             //i~Animals;animalID,animalName,description,speciesID
             //INPUT FIELDS FOR ANIMAL
             String animalID = JOptionPane.showInputDialog(null,"Animal ID:");
             String animalName = JOptionPane.showInputDialog(null,"Animal name:");
             String description = JOptionPane.showInputDialog(null,"Description:");
             String speciesIDFK = JOptionPane.showInputDialog(null,"Species ID FK");
             requestServer="i~Animals;"+animalID+","+animalName+","+description+","
                     +speciesIDFK;
            }else

            if(table.equals("Species")){ //IF SPECIES TABLE IS SELECTED
             //i~Species;speciesID,speciesName
             //INPUT FIELDS FOR SPECIES
             String speciesID = JOptionPane.showInputDialog(null,"Species ID:");
             String speciesName = JOptionPane.showInputDialog(null,"Species name:");
             requestServer="i~Species;"+speciesID+","+speciesName;
            }
        } else

        //CHECK STATEMENT 2 (DELETE)---------------------------------------------
        //if Delete action is selected
        if(action.equals("Delete")){
            //then of Animal table is selected
            if(table.equals("Animals")){
               String animalID = JOptionPane.showInputDialog(null,"Animal ID");
               ////d~animalID
               requestServer="d~"+animalID;
            } else

```



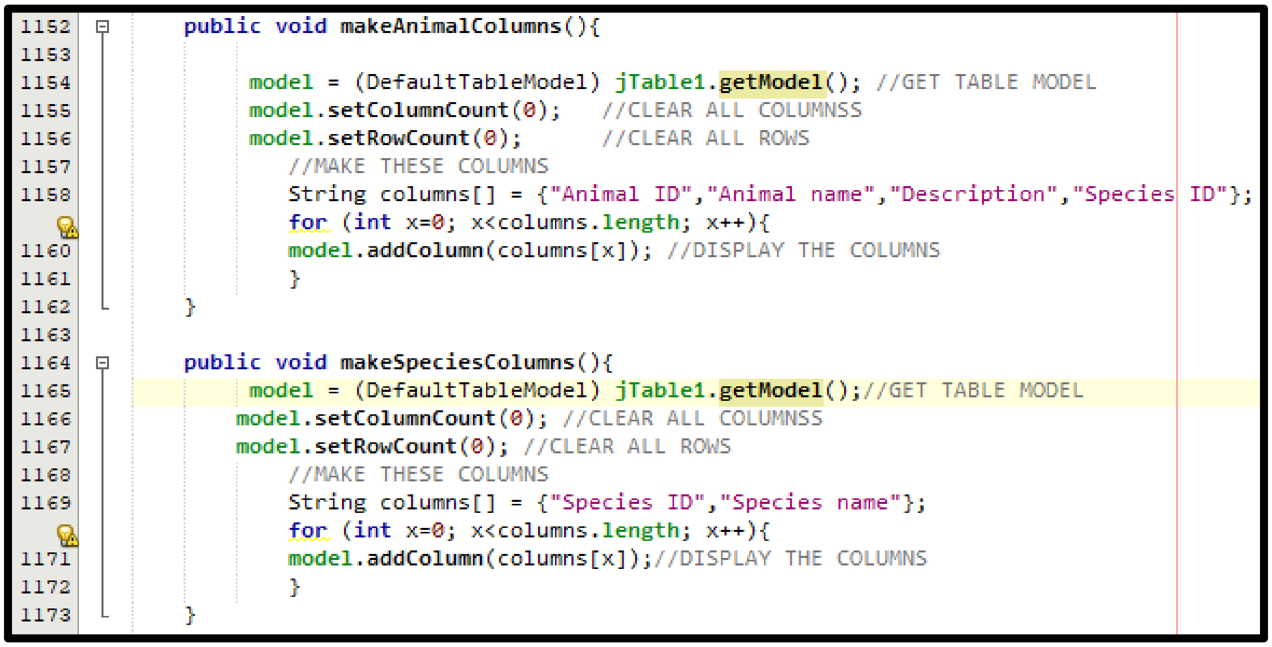
```
            //if species table is selected (CAN NOT DELETE)
            if(table.equals("Species")){
                JOptionPane.showMessageDialog(null, "Can not delete from Species.");
            }
        }
        if(!requestServer.equals("")) //IF IT IS NOT BLANKK
        {
            try{ //SEND REQUEST TO SERVER
                send2SERVER.writeObject(requestServer);
                String response = (String) getfromSERVER.readObject();
                JOptionPane.showMessageDialog(null, response);

                //CATCHING BOTH EXCEPTIONS
            }catch (IOException e){
                System.out.println(e.toString());
            }catch (ClassNotFoundException e){
                System.out.println(e.toString());
            }
        }
    }
```



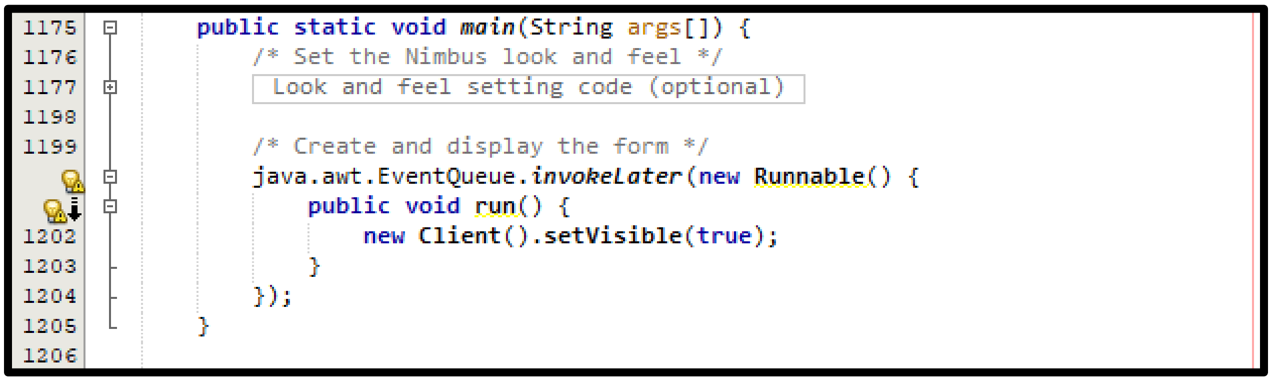
```
    private void formWindowActivated(java.awt.event.WindowEvent evt) {
                //when the form/window is activated and focused
        try{
                Socket testSocket = new Socket(); //create new socket object
                //try connection to server
                testSocket.connect(new InetSocketAddress("localhost", 7777),10);
                SERVER_ONLINE = true; //and make SERVER_ONLINE = true
                System.out.println("Server is active!");
                } catch (IOException e){ //catch exception
                    System.out.println(e.toString()); //print exception
                    SERVER_ONLINE = false;//and make SERVER_ONLINE = false
                }

        if(SERVER_ONLINE==true){ //if SERVER_ONLINE THEN
            jLabel17.setText("Server status : online"); //DISPLAY CORRECT
        } else jLabel17.setText("Server status : offline");//STRING



    }
```



```java
    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);

    }
```

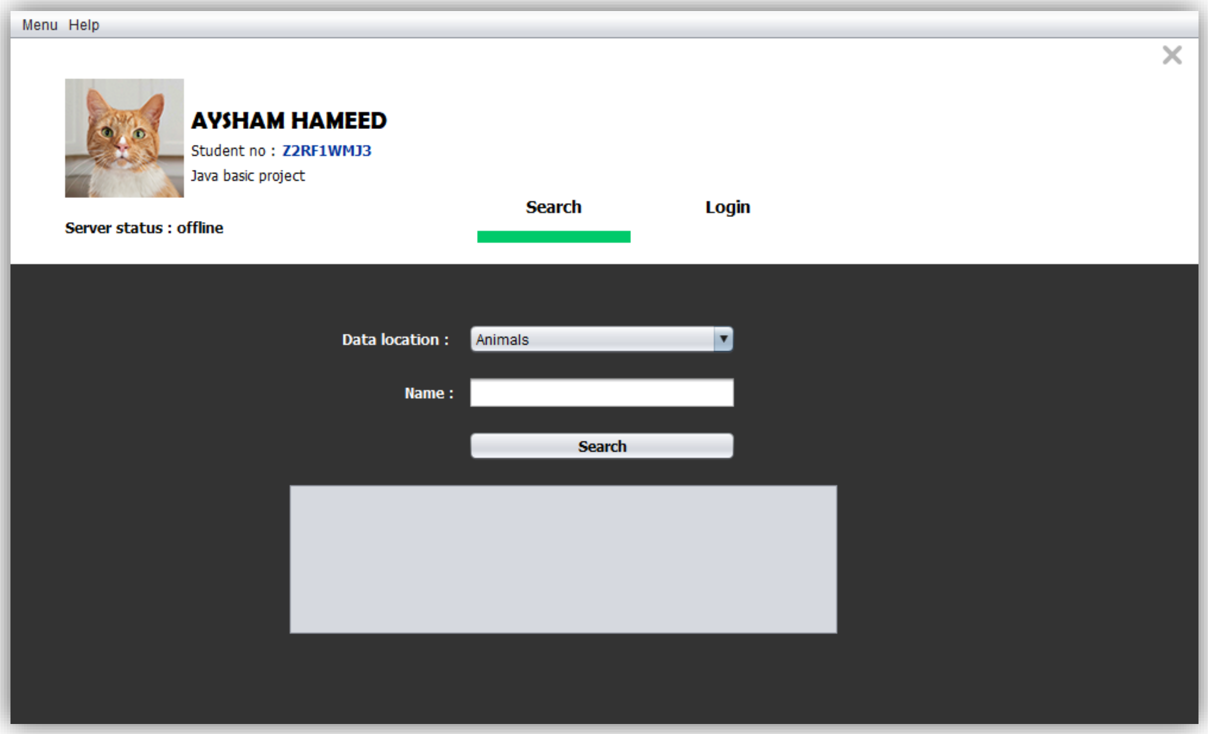


```java
    public void makeAnimalColumns(){

         model = (DefaultTableModel) jTable1.getModel(); //GET TABLE MODEL
         model.setColumnCount(0);    //CLEAR ALL COLUMNSS
         model.setRowCount(0);       //CLEAR ALL ROWS
            //MAKE THESE COLUMNS
            String columns[] = {"Animal ID","Animal name","Description","Species ID"};
            for (int x=0; x<columns.length; x++){
            model.addColumn(columns[x]); //DISPLAY THE COLUMNS
            }
    }

    public void makeSpeciesColumns(){
         model = (DefaultTableModel) jTable1.getModel();//GET TABLE MODEL
        model.setColumnCount(0); //CLEAR ALL COLUMNSS
        model.setRowCount(0); //CLEAR ALL ROWS
            //MAKE THESE COLUMNS
            String columns[] = {"Species ID","Species name"};
            for (int x=0; x<columns.length; x++){
            model.addColumn(columns[x]);//DISPLAY THE COLUMNS
            }
    }
```



```java
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Client().setVisible(true);
            }
        });
    }

```

## User Interface.



AYSHAM HAMEED

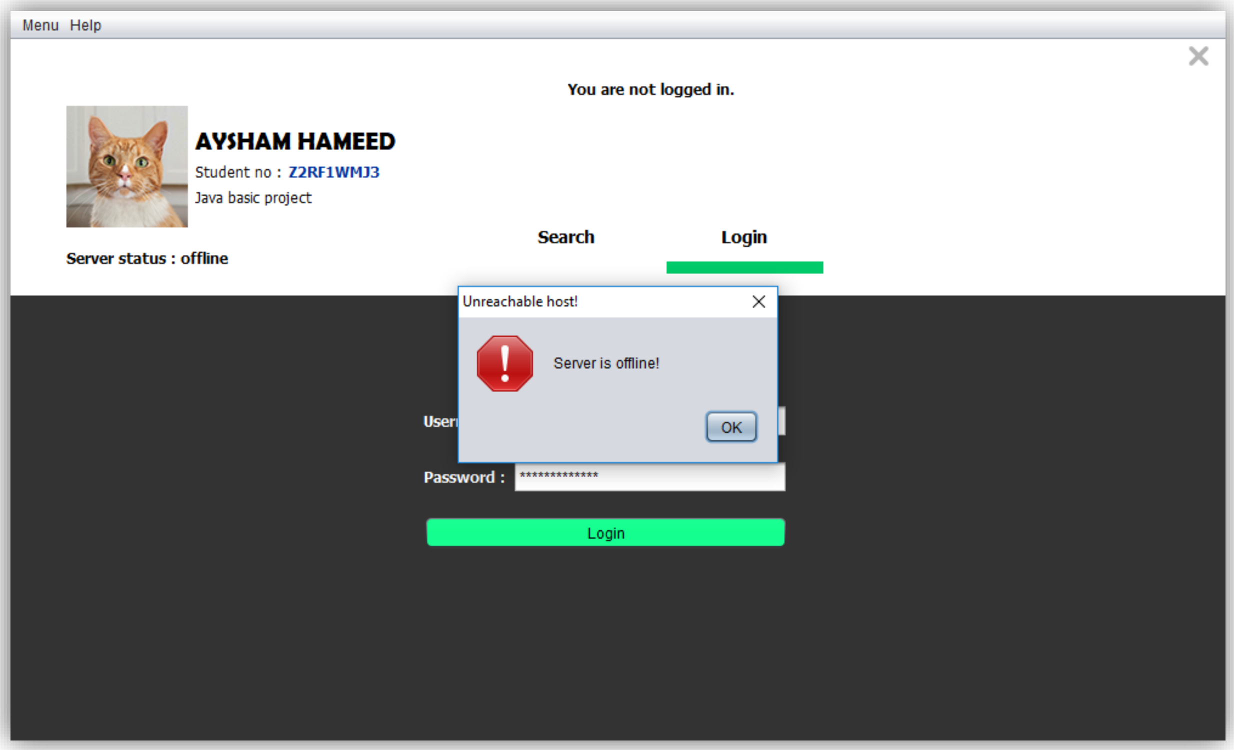
Student no : Z2RF1WMJ3

Java basic project

Server status : offline

Data location : Animals

Name :



You are not logged in.

AYSHAM HAMEED

Student no : Z2RF1WMJ3

Java basic project

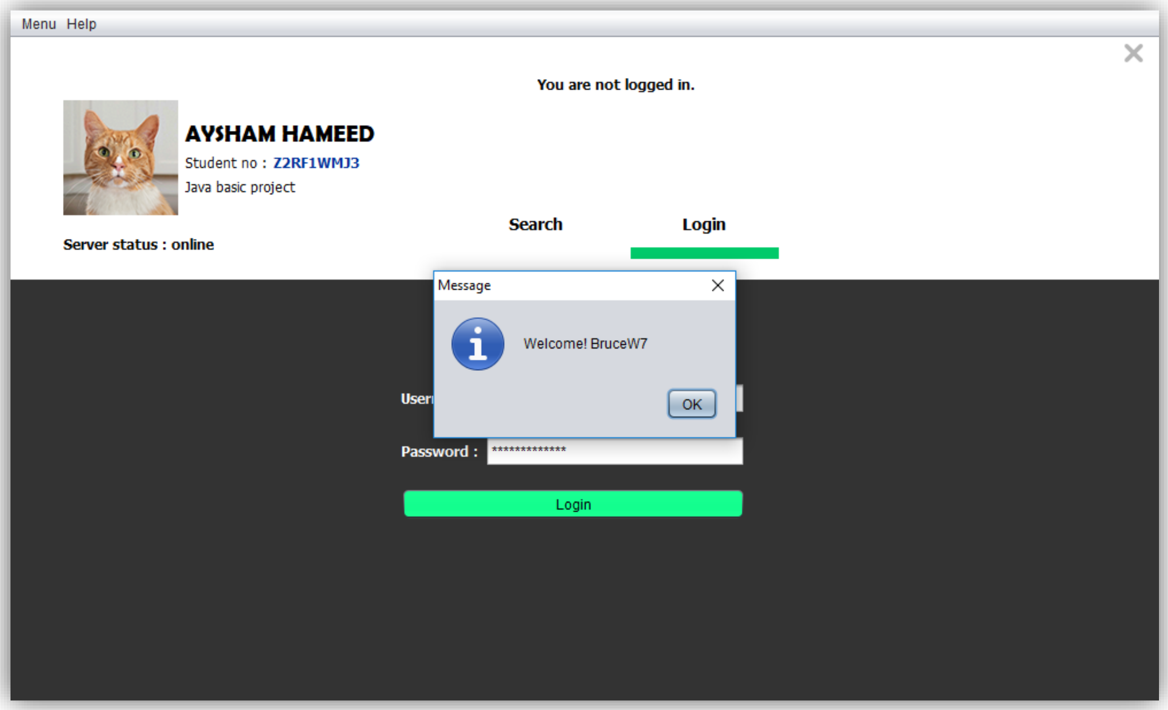
Server status : offline

Unreachable host!

Server is offline!

User

Password : *************



You are not logged in.

AYSHAM HAMEED

Student no : Z2RF1WMJ3

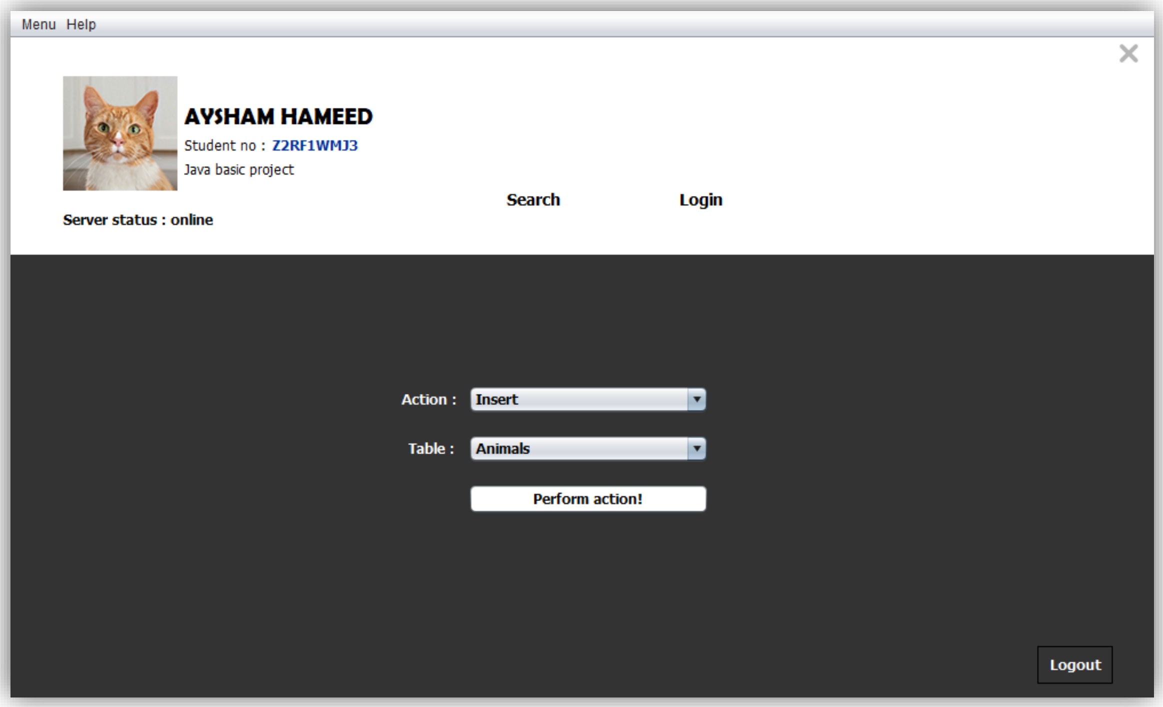
Java basic project

Server status : online

Welcome! BruceW7

User

Password : *************



AYSHAM HAMEED

Student no : Z2RF1WMJ3

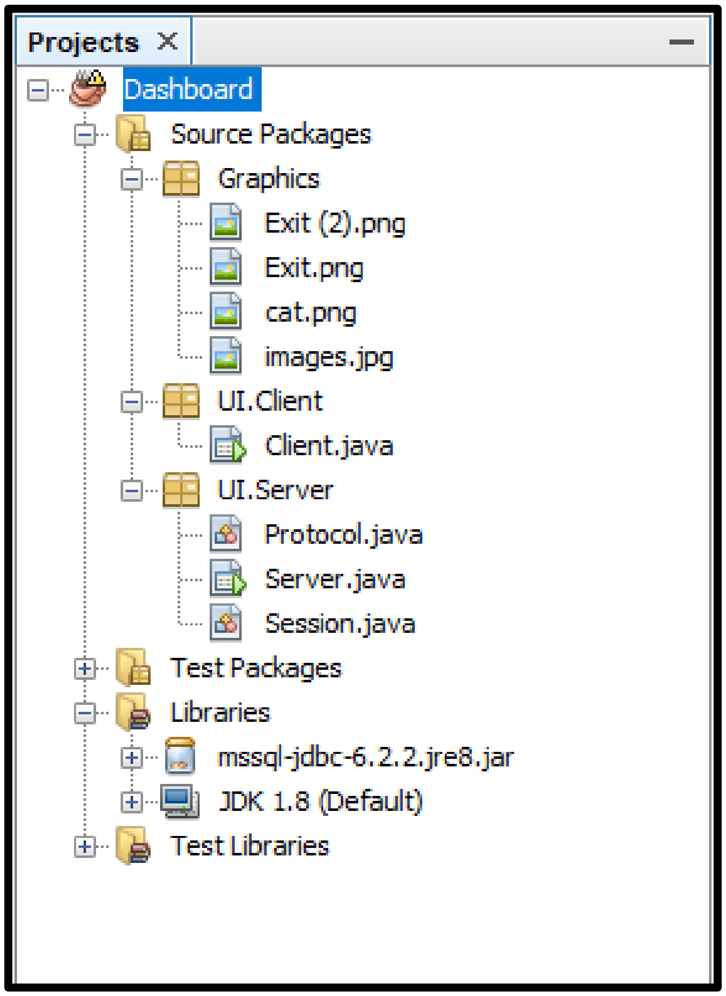
Java basic project

Server status : online

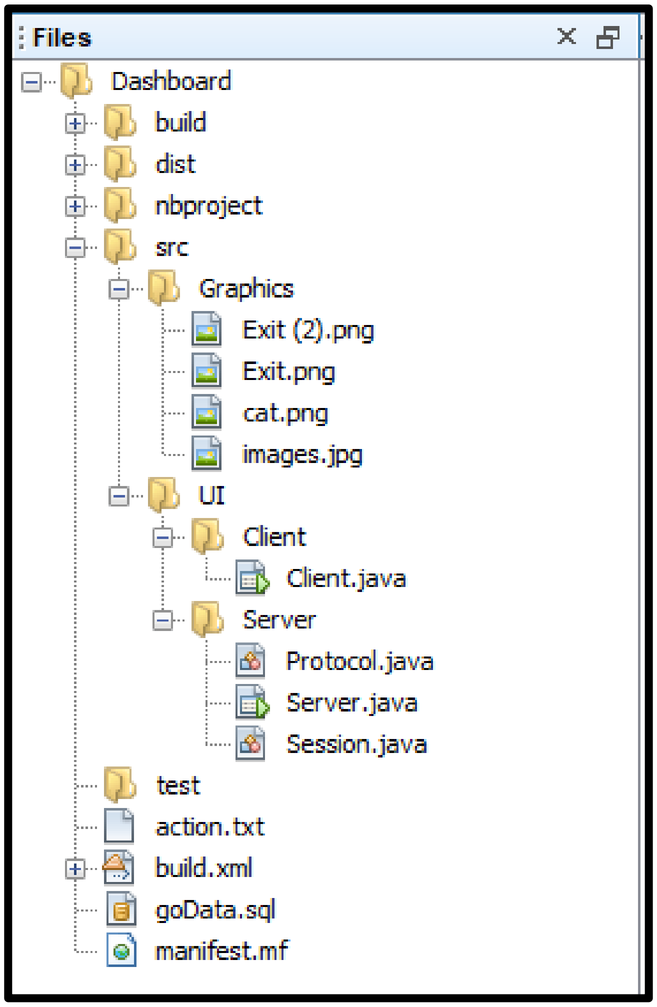
Action : Insert

Table : Animals

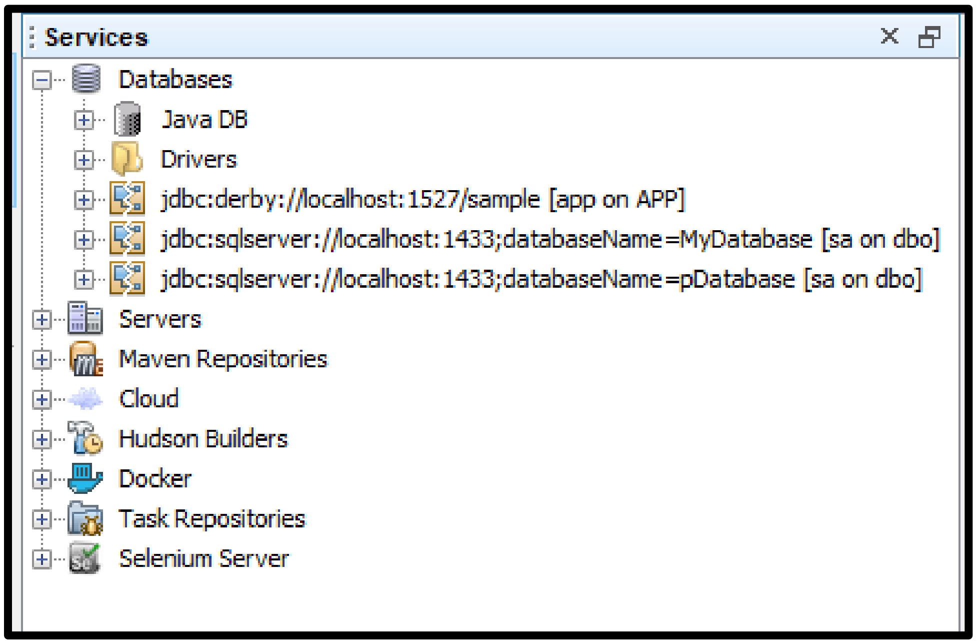
## Project Structure.



```
Projects ×
Dashboard
    Source Packages
        Graphics
            Exit (2).png
            Exit.png
            cat.png
            images.jpg
        UI.Client
            Client.java
        UI.Server
            Protocol.java
            Server.java
            Session.java
    Test Packages
    Libraries
        mssql-jdbc-6.2.2.jre8.jar
        JDK 1.8 (Default)
    Test Libraries
```



```
Files
Dashboard
    build
    dist
    nbproject
    src
        Graphics
            Exit (2).png
            Exit.png
            cat.png
            images.jpg
        UI
            Client
                Client.java
            Server
                Protocol.java
                Server.java
                Session.java
    test
    action.txt
    build.xml
    goData.sql
    manifest.mf
```



```
Services
Databases
    Java DB
    Drivers
    jdbc:derby://localhost:1527/sample [app on APP]
    jdbc:sqlserver://localhost:1433;databaseName=MyDatabase [sa on dbo]
    jdbc:sqlserver://localhost:1433;databaseName=pDatabase [sa on dbo]
Servers
Maven Repositories
Cloud
Hudson Builders
Docker
Task Repositories
Selenium Server
```