Government Engineering College, Rajkot Computer Engineering Department

Advance Java Programming

(Subject Code –3160707)

# LAB

# MANUAL

**6th Semester Computer Engineering**

*YASH A. LATHIYA*

*CE\_K2*

*200200107095*

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **List of Practical** | **Pag e**  **No.** | **Date of Submission** | **Signature** |
| **1.1** | Write java program(s) which accepts names of website from user and prints IP Address of it. The program should be stopped when user enters “exit” as an input. | **5** | **31/01/2023** |  |
| **1.2** | Write a java program (client) which accepts integer from user, send it to the server, server sorts all the numbers and returns the sorted numbers to the clients. Example: Client enters and sends following number => 10, 25, 4, 29, and 15, exit. Output on client program should be => 4, 10, 15, 25,29. | **6** | **31/01/2023** |  |
| **1.3** | Write a menu based java program which performs following.  1. Reverse of the given string  2. Converts string into upper case  3. Converts string into lower case  4. Counts string length | **9** | **07/02/2023** |  |
| **1.4** | Write a java program(s) to implement simple chat application. | **11** | **07/02/2023** |  |
| **1.5** | Write java program to send hello message to the server using UDP. | **14** | **14/02/2023** |  |
| **2.1** | Write a java program to insert a student record in Student Table. | **16** | **14/02/2023** |  |
| **2.2** | Write a java program to insert a student record in Student Table using Prepared Statement. | **18** | **21/02/2023** |  |
| **2.3** | Write a java program to call a stored procedure using Callable Statement Interface. | **20** | **21/02/2023** |  |
| **3.1** | Write a servlet application to print the current date and time. |  | **28/02/2023** |  |
| **3.2** | Write a Servlet application to count the total number of visits on your website. |  | **28/02/2023** |  |
| **3.3** | Write a Servlet application to authenticate user. |  | **28/02/2023** |  |
| **3.4** | Write a Servlet application to perform sign up on website. Fields : username, password, Name, Email and Mobile number |  | **07/03/2023** |  |
| **3.5** | Write a Servlet application to fetch Mark sheet of a given enrollment number. |  | **07/03/2023** |  |
| **3.6** | To develop a web application using servlet event handling and filters. |  | **14/03/2023** |  |
| **3.7** | Develop a web application that demonstrates the use of session level events.  Implement the servlet that adds two numbers. Use the two filters in a chain mapped to servlet such that first filter checks the number for valid format and second checks them for a range. If both conditions are fulfilled then only servlet perform the operation and displays the result, otherwise appropriate message from respective filter. |  | **14/03/2023** |  |
| **4.1** | To develop a web application using JSP and JDBC Write two HTML pages named as register.html and login.html.  Where, “register.html‟ has one UI-form, which collects user’s details like full name, age, mobile number, email address, password and confirm password. This register.html page will submit data to servlet named as “RegisterServlet.java”. RegisterServlet will insert user’s details to database and on successful insertion, it will redirect user tologin.html with appropriate message otherwise show appropriate error message.  “login.html” asks either mobile number OR email address and password details for login.  For valid user, redirect him/her to welcome.jsp page and also start new session.  In welcome.jsp,  1. Check whether user is eligible for Vote or not using session data and display appropriate message.  2. Display logout button, so that once user will click on logout, it will close the user's session and redirect to login.html.  For invalid user, display appropriate message like at login page  1. Invalid mobile Number/email address or password. OR  2. Mobile Number/email address doesn't exist. |  | **21/03/2023** |  |
| **4.2** | Write a JSP page to demonstrate various tag of SQL tag library |  | **21/03/2023** |  |
| **5.1** | Write a program to demonstrate the use of JSF Convertor Tag and Validation Tag. |  | **28/03/2023** |  |
| **5.2** | Write a program that demonstrates the use of JSF Event Handling and Database Access. |  | **04/04/2023** |  |
| **6.1** | Write a hibernate application to save object of Employee class into database.  Employee class contains following properties  • EID  • ENAME  • ESalary  Write SQL query for table as well. |  | **11/04/2023** |  |
| **6.2** | Write a menu based program to perform following operations using HQL (consider above class and table)  a. Insert new record  b. Update Existing Record  c. Delete Record |  | **02/05/2023** |  |
| **7.1** | Create a MVC Spring Application which demonstrates Dependency Injection using constructor Consider an Employee(E-id, E\_name) class. Write all necessary classes/files. |  | **09/05/2023** |  |

***Unit 1 :Java Networking***

**Practical 1.1**

**Write java program(s) which accepts names of website from user and prints IP Address of it. The program should be stopped when user enters “exit” as an input.**

import java.util.Scanner;

import java.net.\*;

public class **Practical1** {

/\*\*

\* @param args the command line arguments

\* @param {str} URL of Website

\*/

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

String str = "";

while(str != "exit"){

str = sc.nextLine();

try{

InetAddress ip = InetAddress.getByName(new URL(str).getHost());

System.out.println(ip);

}

catch(Exception e){

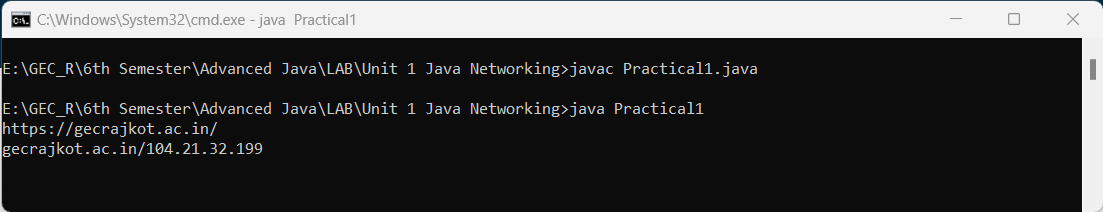
System.out.println("Please enter correct URL");

}

}

}

}



**Practical 1.2**

**Write a java program (client) which accepts integer from user, send it to the server, server sorts all the numbers and returns the sorted numbers to the clients. Example: Client enters and sends following number =>10, 25, 4, 29, and 15, exit. Output on client program should be => 4, 10, 15, 25,29.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class **Practical2Client** {

/\*\*

\* @param args the command line arguments

\* @param {numbers} List of num

\* @param {num} a number which is entered by client at command line

\*/

public static void main(String[] args) {

try {

Socket socket = new Socket("localhost", 3000); // connect to server

System.out.println("Connected to server.");

// create input/output streams

InputStream is = socket.getInputStream();

DataInputStream dis = new DataInputStream(is);

OutputStream os = socket.getOutputStream();

DataOutputStream dos = new DataOutputStream(os);

Scanner sc = new Scanner(System.in);

List<Integer> numbers = new ArrayList<Integer>();

System.out.println("Enter numbers to be sorted (enter -1 to stop):");

// read numbers from user input and send to server

while (true) {

int num = sc.nextInt();

if (num == -1) {

break;

}

numbers.add(num);

dos.writeInt(num);

}

dos.writeInt(-1); // signal end of input

// read sorted numbers from server and print

System.out.println("Sorted numbers:");

while (true) {

try {

int num = dis.readInt();

if (num == -1) {

break;

}

System.out.print(num + " ");

} catch (Exception e) {

e.printStackTrace();

break;

}

}

socket.close(); // close socket

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class **Practical2Server** {

/\*\*

\* @param args the command line arguments

\* @param {numbers} List of num

\* @param {num} a number which is entered by client at command line

\*/

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(3000); // create server socket

System.out.println("Server started. Waiting for client...");

Socket socket = serverSocket.accept(); // accept client connection

System.out.println("Client connected.");

// create input/output streams

InputStream is = socket.getInputStream();

DataInputStream dis = new DataInputStream(is);

OutputStream os = socket.getOutputStream();

DataOutputStream dos = new DataOutputStream(os);

List<Integer> numbers = new ArrayList<Integer>();

while (true) {

try {

int num = dis.readInt(); // read number sent by client

if (num == -1) { // client signals end of input

break;

}

numbers.add(num);

} catch (Exception e) {

e.printStackTrace();

break;

}

}

Collections.sort(numbers); // sort the numbers

// send sorted numbers back to client

for (int num : numbers) {

dos.writeInt(num);

}

dos.writeInt(-1); // signal end of output

System.out.println("Numbers sorted and sent back to client.");

serverSocket.close(); // close server socket

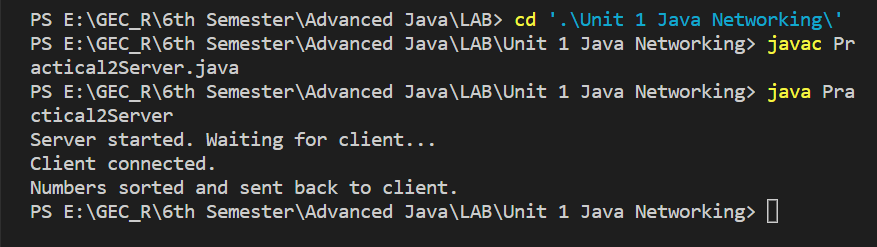
} catch (Exception e) {

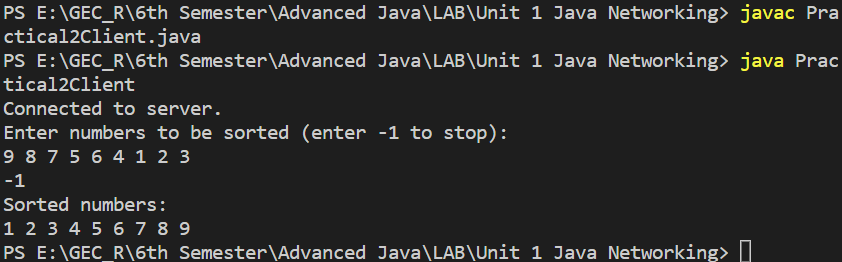
e.printStackTrace();

}

}

}





**Practical 1.3**

**Write a menu based java program which performs following.**

**1. Reverse of the given string**

**2. Converts string into upper case**

**3. Converts string into lower case**

**4. Counts string length**

import java.util.Scanner;

public class **Practical3** {

/\*\*

\* @param args the command line arguments

\* @param {str} string on which operations to be performed

\* @param {operation} operation number {1 , 2, 3, 4} details mentioned on line 30- 33

\*/

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

while(true){

System.out.println("Enter String -- enter exit to terminate program");

String str = sc.nextLine();

System.out.println("enter 1 for reverse the "+str);

System.out.println("enter 2 for converting the "+str+ " into upper case");

System.out.println("enter 3 for converting the "+str+ " into lower case");

System.out.println("enter 4 for count length of the "+str);

int operation = sc.nextInt();

sc.nextLine();

switch(operation){

case 1: StringBuilder sb = new StringBuilder(); //reverse the string

sb.append(str);

sb.reverse();

System.out.println(sb);

break;

case 2: System.out.println(str.toUpperCase()); // to upper case

break;

case 3: System.out.println(str.toLowerCase()); //to lower case

break;

case 4: System.out.println("length of string is "+str.length()); //length of string

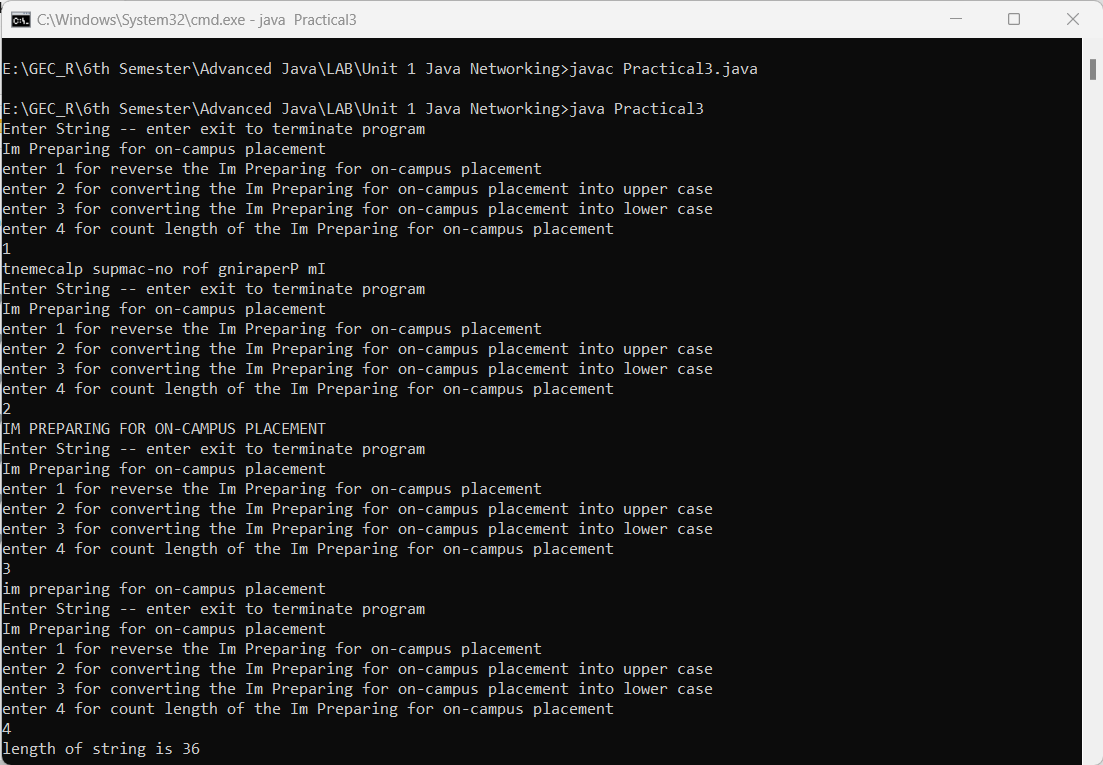
break;

}

}

}

}



**Practical 1.4**

**Write a java program(s) to implement simple chat application.**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class **Practical4Client** {

/\*\*

\* @param args the command line arguments

\* @param {str} message which will be send to server

\* @param {msg} message which is send by server

\*/

public static void main(String[] args) {

try {

Socket socket = new Socket("localhost", 3000); // connect to server

System.out.println("Connected to server.");

// create input/output streams

InputStream is = socket.getInputStream();

DataInputStream dis = new DataInputStream(is);

OutputStream os = socket.getOutputStream();

DataOutputStream dos = new DataOutputStream(os);

Scanner sc = new Scanner(System.in);

System.out.println("Enter Message : ");

// read numbers from user input and send to server

while (true) {

String str = sc.nextLine();

dos.writeUTF(str);

try{

String msg = dis.readUTF();

}

catch(Exception e){

System.out.println("");

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class **Practical4Server** {

/\*\*

\* @param args the command line arguments

\* @param {str} message which will be send to client

\* @param {msg} message which is send by client

\*/

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

try {

ServerSocket serverSocket = new ServerSocket(3000); // create server socket

System.out.println("Server started. Waiting for client...");

Socket socket = serverSocket.accept(); // accept client connection

System.out.println("Client connected.");

// create input/output streams

InputStream is = socket.getInputStream();

DataInputStream dis = new DataInputStream(is);

OutputStream os = socket.getOutputStream();

DataOutputStream dos = new DataOutputStream(os);

while (true) {

try {

String str = dis.readUTF();

System.out.println(str);

try{

String msg = sc.nextLine();

dos.writeUTF(msg);

}

catch(Exception e){

System.out.println("");

}

} catch (Exception e) {

e.printStackTrace();

break;

}

}

serverSocket.close(); // close server socket

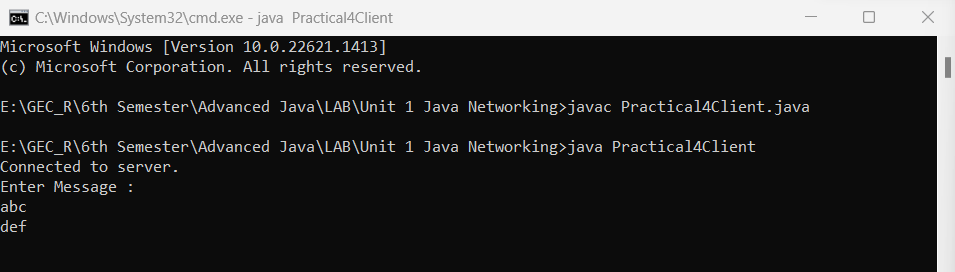
} catch (Exception e) {

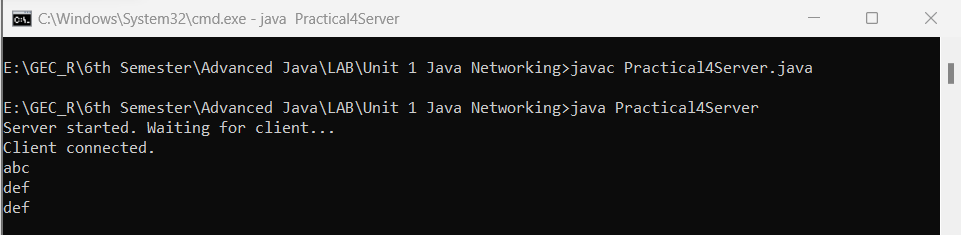
e.printStackTrace();

}

}

}





**Practical 1.5**

**Write java program to send hello message to the server using UDP.**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class **Practical5Client** {

public static void main(String[] args) {

String message = "Hello";

byte[] buffer = message.getBytes();

int serverPort = 12345;

try (DatagramSocket socket = new DatagramSocket()) {

InetAddress serverAddress = InetAddress.getByName("localhost");

DatagramPacket packet = new DatagramPacket(buffer, buffer.length, serverAddress, serverPort);

socket.send(packet);

System.out.println("Sent message: " + message);

}

catch (IOException e) {

System.err.println("Error sending message: " + e.getMessage());

}

}

}

public class **Practical5Server** {

public static void main(String[] args) {

int serverPort = 12345;

byte[] buffer = new byte[1024];

try (DatagramSocket socket = new DatagramSocket(serverPort)) {

System.out.println("Server started and listening on port " + serverPort);

while (true) {

DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

socket.receive(packet);

String message = new String(packet.getData(), 0, packet.getLength());

System.out.println("Received message: " + message);

}

}

catch (SocketException e) {

System.err.println("Error creating socket: " + e.getMessage());

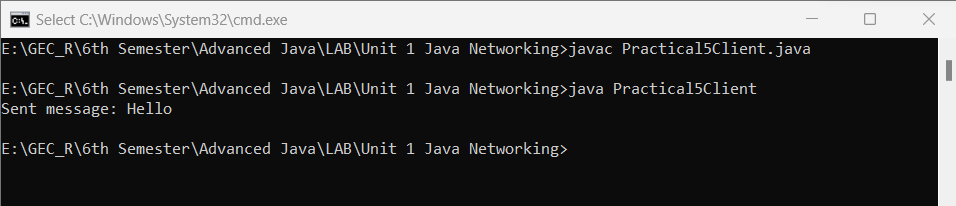
} catch (IOException e) {

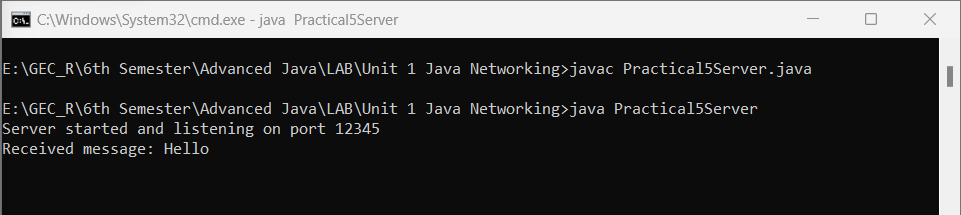
System.err.println("Error receiving message: " + e.getMessage());

}

}

}





***Unit 2 : Java Database Connectivity***

**Practical 2.1**

**Write a java program to insert a student record in Student Table.**

package unit2labpractical;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

\* @author Yash

\*/

public class Unit2LABPractical {

/\*\*

\* @param args the command line arguments

\* @param {con} object of Connection class

\* @param {stmt} object of statement class

\* @param {sq} Insert SQL Query

\*/

public static void main(String[] args) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/advancejavapractical","root","Yash@2233");

Statement stmt = con.createStatement();

String sq = "INSERT INTO student VALUES ('yash', 'ya', 'yash@gmail.com', 'surat',7046985354, 1)";

stmt.executeUpdate(sq);

} catch (ClassNotFoundException ex) {

Logger.getLogger(Unit2LABPractical.class.getName()).log(Level.SEVERE, null, ex);

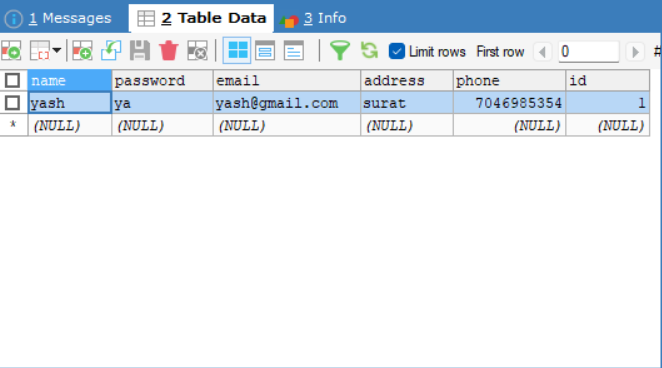
} catch (SQLException ex) {

Logger.getLogger(Unit2LABPractical.class.getName()).log(Level.SEVERE, null, ex);

}

}

}



**Practical 2.2**

**Write a java program to insert a student record in Student Table using Prepared Statement.**

package unit2practical2;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.PreparedStatement;

/\*\*

\* @author Yash

\*/

public class Unit2Practical2 {

/\*\*

\* @param args the command line arguments

\* @param {con} object of Connection class

\* @param {stmt} object of PreparedStatement class

\* @param {sq} Insert SQL Query

\*/

public static void main(String[] args) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/advancejavapractical","root","Yash@2233");

String sq = "INSERT INTO student (name, password, email, address, phone, id) VALUES (?,?,?,?,?,?)";

PreparedStatement stmt = con.prepareStatement(sq);

stmt.setString(1, "Nishant");

stmt.setString(2, "nishu");

stmt.setString(3, "n@gmail.com");

stmt.setString(4, "Ahemdabad");

stmt.setLong(5, 8401055354L);

stmt.setInt(6, 2);

stmt.executeUpdate();

} catch (ClassNotFoundException ex) {

Logger.getLogger(Unit2Practical2.class.getName()).log(Level.SEVERE, null, ex);

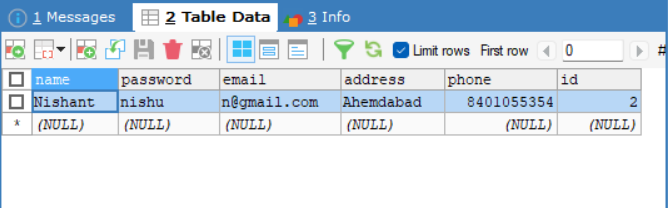
} catch (SQLException ex) {

Logger.getLogger(Unit2Practical2.class.getName()).log(Level.SEVERE, null, ex);

}

}

}



**Practical 2.3**

**Write a java program to call a stored procedure using Callable Statement Interface.**

package unit2practical3;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.CallableStatement;

/\*\*

\* @author Yash

\*/

public class Unit2Practical3 {

/\*\*

\* @param args the command line arguments

\* @param {con} object of Connection class

\* @param {stmt} object of CallableStatement class

\* @param {sql} Calls stored procedure add\_studentlocation

\*/

public static void main(String[] args) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/advancejavapractical","root","Yash@2233");

String sql = "{CALL add\_studentlocation(?,?)}";

CallableStatement stmt = con.prepareCall(sql);

stmt.setString(1, "Satyam");

stmt.setString(2, "Bhavnagar");

stmt.executeUpdate();

}

catch (ClassNotFoundException ex) {

Logger.getLogger(Unit2Practical3.class.getName()).log(Level.SEVERE, null, ex);

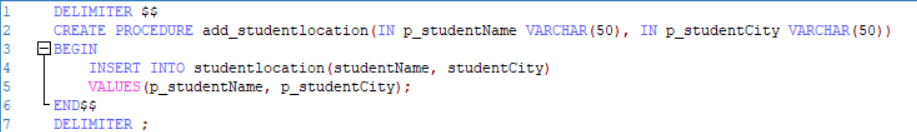
} catch (SQLException ex) {

Logger.getLogger(Unit2Practical3.class.getName()).log(Level.SEVERE, null, ex);

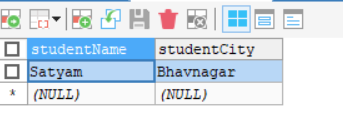
}

}

}



[stored procedure – add\_studentlocation]



***Unit 3 : Servlet***

**Practical 3.1**

**Write a servlet application to print the current date and time.**

**Practical 3.2**

**Write a Servlet application to count the total number of visits on your website.**

**Practical 3.3**

**Write a Servlet application to authenticate user.**

**Practical 3.4**

**Write a Servlet application to perform sign up on website.**

**Fields : username, password, Name ,Email and Mobile number**

**Practical 3.5**

**Write a Servlet application to fetch Mark sheet of a given enrollment number.**

**Practical 3.6**

**To develop a web application using servlet event handling and filters.**

**Practical 3.7**

**Develop a web application that demonstrates the use of session level events.**

**Implement the servlet that adds two numbers. Use the two filters in a chain mapped to servlet such that first filter checks the number for valid format and second checks them for a range. If both conditions are fulfilled then only servlet perform the operation and displays the result, otherwise appropriate message from respective filter.**

***Unit 4 : Java Server Page***

**Practical 4.1**

**To develop a web application using JSP and JDBC.**

**Write two HTML pages named as register.html and login.html.**

**Where, “register.html‟ has one UI-form, which collects user’s details likefull name, age, mobile number, email address,password and confirm password. This register.html page will submit data to servlet named as “RegisterServlet.java”. RegisterServlet will insert user‟s details to database and on successful insertion, it will redirect user tologin.html with appropriate message otherwise show appropriate error message.**

**“login.html” asks either mobile number OR email address and password details for login.**

**For valid user, redirect him/her to welcome.jsp page and also start new session. In welcome.jsp,**

**1. Check whether user is eligible for Vote or not using session data and display appropriate message.**

**2. Display logout button, so that once user will click on logout, it will close the user's session and redirect to login.html.**

**For invalid user, display appropriate message like at login page**

**1. Invalid mobile Number/email address or password. OR**

**2. Mobile Number/email address doesn't exist.**

**Practical 4.2**

**Write a JSP page to demonstrate various tag of SQL tag library.**

***Unit 5 : Java Server Faces***

**Practical 5.1**

**Write a program to demonstrate the use of JSF Convertor Tag and Validation Tag.**

**Practical 5.2**

**Write a program that demonstrates the use of JSF Event Handling and Database Access.**

***Unit 6 : Hibernate***

**Practical 6.1**

**Write a hibernate application to save object of Employee class into database.**

**Employee class contains following properties.**

**• EID**

**• ENAME**

**• ESalary**

**Write SQL query for table as well.**

**Practical 6.2**

**Write a menu based program to perform following operations using HQL (consider above class and table).**

**a. Insert new record**

**b. Update Existing Record**

**c. Delete Record**

***Unit 7 : Spring MVC – Java Web Framework***

**Practical 7.1**

**Create a MVC Spring Application which demonstrates Dependency Injection using constructor Consider an Employee (E-id, E\_name) class. Write all necessary classes/files.**