# Practical 1

**Objective:**

1. Write a program to implement the connection oriented echo client server application

**Software Environment: Notepad++, jdk1.8**

**Implementation:**

**SERVER:**

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.UnknownHostException;

//Write a program to implement the connection oriented echo client server application

public class server {

static Socket socket;

static ServerSocket serverSocket;

DataInputStream inputStream;

DataOutputStream outputStream;

public server() throws UnknownHostException, IOException{

serverSocket=new ServerSocket(4523);

System.out.println("Server is On");

socket=serverSocket.accept();

outputStream=new DataOutputStream(socket.getOutputStream());

inputStream=new DataInputStream(socket.getInputStream());

String msg=inputStream.readUTF();

//System.out.println(msg);

outputStream.writeUTF(msg);

outputStream.flush();

socket.close();

serverSocket.close();

}

public static void main(String[] args) throws UnknownHostException, IOException {

server serverClass=new server();

}}

**CLIENT:**

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.\*;

//Write a program to implement the connection oriented echo client server application

public class client {

static Socket socket;

DataInputStream inputStream;

DataOutputStream outputStream;

public client() throws IOException{

socket=new Socket("172.16.1.28",4523);

System.out.println("Connected");

outputStream=new DataOutputStream(socket.getOutputStream());

inputStream=new DataInputStream(socket.getInputStream());

String msg="i m client";

outputStream.writeUTF(msg);

String msg2=inputStream.readUTF();

System.out.println("Server side msg (echo) : "+msg2);

socket.close();

}

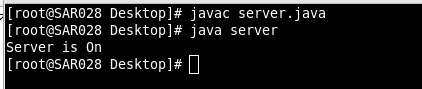
public static void main(String[] args) throws UnknownHostException, IOException {

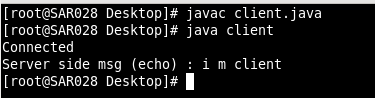
client clientClass=new client();

}

}

**OUTPUT:**





**Objective:**

1. Write a program to implement the connection oriented echo client server application

**Software Environment: Notepad++, jdk1.8**

**Implementation:**

**SERVER:**

//Server Program for echo application

import java.net.\*;

import java.io.\*;

public class Server{

static DatagramSocket socket;

static DatagramPacket rdp,sdp;

public static void main(String args[]) throws IOException{

System.out.println("Server Started......");

try{

socket = new DatagramSocket(4444);

String str;

while(socket!=null){

rdp = new DatagramPacket(new byte[512],512);

socket.receive(rdp);

sdp = new DatagramPacket(rdp.getData(),rdp.getLength(),rdp.getAddress(),rdp.getPort());

socket.send(sdp);

}

socket.close();

}catch(Exception e){

e.printStackTrace();

}

}}

**CLIENT:**

import java.net.\*;

import java.io.\*;

public class Client{

static DatagramSocket socket;

static DatagramPacket sdp,rdp;

static BufferedReader cl\_in;

static InetAddress address;

static int port;

public static void main(String args[]){

System.out.println("Initiate Communication.....");

try{

socket = new DatagramSocket();

cl\_in = new BufferedReader(new InputStreamReader(System.in));

String str1;

address = InetAddress.getLocalHost();

port = 4444;

str1 = cl\_in.readLine();

while(!str1.equalsIgnoreCase("stop")){

sdp = new DatagramPacket(str1.getBytes(),str1.length(),address,port);

socket.send(sdp);

rdp = new DatagramPacket(new byte[str1.length()],str1.length());

socket.receive(rdp);

String str2 = new String(rdp.getData());

System.out.println(str2);

str1 = cl\_in.readLine();

}

socket.close();

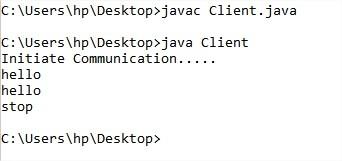
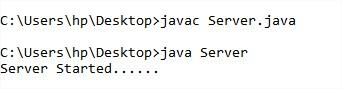
}catch(Exception e){

e.printStackTrace();

}

}}

**Output:**



**Practical 2**

**Objective:**

1. Develop chat application using either TCP or UDP protocol.

**Software Environment: Notepad++, jdk1.8**

Implementation:

SERVER:

import java.net.\*;

import java.io.\*;

public class Server{

static ServerSocket serverSocket;

static Socket socket;

static BufferedReader cl\_in, ser\_in;

static DataOutputStream ser\_out;

public static void main(String args[]) throws IOException{

System.out.println("Server Started......");

try{

serverSocket = new ServerSocket(1234);

socket = serverSocket.accept();

ser\_in = new BufferedReader(new InputStreamReader(System.in));

cl\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

ser\_out = new DataOutputStream(socket.getOutputStream());

String str1,str2;

str1 = cl\_in.readLine();

while(str1!=null){

System.out.println(str1);

str2 = ser\_in.readLine();

ser\_out.writeBytes(str2+"\n");

str1 = cl\_in.readLine();

}

socket.close();

serverSocket.close();

}catch(Exception e){

e.printStackTrace();

}

}}

**CLIENT:**

**import java.net.\*;**

**import java.io.\*;**

**public class Client{**

**static Socket socket;**

**static BufferedReader cl\_in, ser\_in;**

**static DataOutputStream cl\_out;**

**public static void main(String args[]){**

**System.out.println("Initiate Communication.....");**

**try{**

**socket = new Socket("127.0.0.1",1234);**

**cl\_in = new BufferedReader(new InputStreamReader(System.in));**

**ser\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**

**cl\_out = new DataOutputStream(socket.getOutputStream());**

**String str1,str2;**

**str1 = cl\_in.readLine();**

**while(!str1.equalsIgnoreCase("stop")){**

**cl\_out.writeBytes(str1+"\n");**

**str2 = ser\_in.readLine();**

**System.out.println(str2);**

**str1 = cl\_in.readLine();**

**}**

**socket.close();**

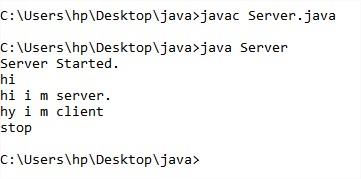
**}catch(Exception e){**

**e.printStackTrace();**

**}**

**}}**

**OUTPUT:**



**Objective:**

1. Implement TCP Server for transferring files using Socket and ServerSocket .

**Software Environment: Notepad++, jdk1.8**

**Implementation:**

**SERVER:**

import java.net.\*;

import java.io.\*;

public class Server{

static ServerSocket serverSocket;

static Socket socket;

static BufferedReader cl\_in, ser\_in;

static DataOutputStream ser\_out;

public static void main(String args[]) throws IOException{

System.out.println("Server Started.");

FileInputStream in = null;

try{

serverSocket = new ServerSocket(1234);

socket = serverSocket.accept();

ser\_in = new BufferedReader(new InputStreamReader(System.in));

cl\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

ser\_out = new DataOutputStream(socket.getOutputStream());

String str1;

str1 = cl\_in.readLine();

System.out.println(str1);

in = new FileInputStream(str1);

int c;

ser\_out.writeBytes("File Found");

while((c = in.read())!=-1){

ser\_out.write(c);

}

c=-1;

ser\_out.write(c);

System.out.println("File transferred Successfully");

}catch(SocketException e){

e.printStackTrace();

}catch(FileNotFoundException e){

ser\_out.writeBytes("File Not Found");

}finally{

socket.close();

serverSocket.close();

}

}

}

**CLIENT:**

import java.net.\*;

import java.io.\*;

public class Client{

static Socket socket;

static BufferedReader cl\_in, ser\_in;

static DataOutputStream cl\_out;

public static void main(String args[]) throws IOException{

System.out.println("Communication started.");

FileOutputStream out = null;

try{

socket = new Socket("PARTH",1234);

cl\_in = new BufferedReader(new InputStreamReader(System.in));

ser\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

cl\_out = new DataOutputStream(socket.getOutputStream());

String str1;

System.out.print("Enter FileName: ");

str1 = cl\_in.readLine();

cl\_out.writeBytes(str1+"\n");

out = new FileOutputStream("output.txt");

int c;

String str2 = ser\_in.readLine();

if(str2.equals("File Not Found")){

System.out.println("File Not Found");

}else{

while((c = ser\_in.read())!=-1){

out.write(c);

} }

}catch(SocketException e){

e.printStackTrace();

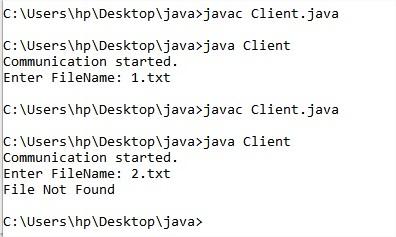
}finally{

socket.close();

}}}

**OUTPUT:**

****



# Practical 3

**Objective:**

1. Implement any one sorting algorithm using TCP/UDP on Server application and

Give Input On Client side and client should sorted output from server and display

sorted on input side.

**Software Environment: Notepad++, jdk1.8**

**Implementation:**

**SERVER:**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Server{

static ServerSocket serverSocket;

static Socket socket;

static BufferedReader cl\_in, ser\_in;

static DataOutputStream ser\_out;

public static void main(String args[]) throws IOException{

System.out.println("Server Started.");

try{

serverSocket = new ServerSocket(1234);

socket = serverSocket.accept();

ser\_in = new BufferedReader(new InputStreamReader(System.in));

cl\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

ser\_out = new DataOutputStream(socket.getOutputStream());

String str1,str2;

str1 = cl\_in.readLine();

String[] a = str1.split(" ");

int[] b = new int[a.length];

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

b[i] = Integer.parseInt(a[i]);

}

int min;

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

for(int j=i+1;j<b.length;j++){

if(b[i]>b[j]){

min = b[i];

b[i] = b[j];

b[j] = min;

}

}

}

String outputstr = Arrays.toString(b);

ser\_out.writeBytes(outputstr);

socket.close();

serverSocket.close();

}catch(Exception e){

e.printStackTrace();

}}}

**CLIENT:**

import java.net.\*;

import java.io.\*;

public class Client{

static Socket socket;

static BufferedReader cl\_in, ser\_in;

static DataOutputStream cl\_out;

public static void main(String args[]){

System.out.println("Communication started.");

try{

socket = new Socket("PARTH",1234);

cl\_in = new BufferedReader(new InputStreamReader(System.in));

ser\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

cl\_out = new DataOutputStream(socket.getOutputStream());

String str1,str2;

System.out.println("Enter Array");

str1 = cl\_in.readLine();

cl\_out.writeBytes(str1+"\n");

str2 = ser\_in.readLine();

System.out.println(str2);

socket.close();

}catch(Exception e){

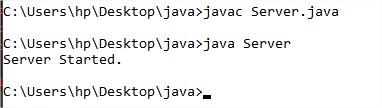
e.printStackTrace();

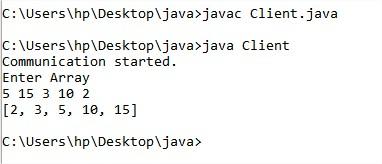
}

}

}

**OUTPUT:**

****

****

**Objective:**

1. Implement Concurrent TCP Server programming in which more than one client can

connect and communicate with Server for sending the string and server returns the

reverse of string to each of client

**Software Environment: Notepad++, jdk1.8**

**Implementation:**

**SERVER:**

**import java.net.\*;**

**import java.io.\*;**

**public class Server extends Thread{**

**static ServerSocket serverSocket;**

**Server() throws IOException{**

**serverSocket = new ServerSocket(1234);**

**}**

**public void run(){**

**try{**

**while(true){**

**Socket socket = serverSocket.accept();**

**BufferedReader cl\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**

**DataOutputStream ser\_out = new DataOutputStream(socket.getOutputStream());**

**String str1;**

**str1 = cl\_in.readLine();**

**System.out.println(str1);**

**StringBuilder str = new StringBuilder();**

**str.append(str1);**

**str = str.reverse();**

**ser\_out.writeBytes(str+"\n");**

**}**

**}catch(SocketException e){**

**System.out.println("Socket timed out");**

**}catch(IOException e){**

**e.printStackTrace();**

**}}**

**public static void main(String args[]){**

**System.out.println("Server Started");**

**try{**

**Thread t = new Server();**

**t.start();**

**}catch(IOException e){**

**e.printStackTrace();**

**}**

**}**

**}**

**CLIENT:**

**import java.net.\*;**

**import java.io.\*;**

**public class Client{**

**static Socket socket;**

**static BufferedReader cl\_in, ser\_in;**

**static DataOutputStream cl\_out;**

**public static void main(String args[]){**

**System.out.println("Communication started.");**

**try{**

**socket = new Socket("PARTH",1234);**

**cl\_in = new BufferedReader(new InputStreamReader(System.in));**

**ser\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**

**cl\_out = new DataOutputStream(socket.getOutputStream());**

**String str1,str2;**

**str1 = cl\_in.readLine();**

**cl\_out.writeBytes(str1+"\n");**

**str2 = ser\_in.readLine();**

**System.out.println(str2);**

**socket.close();**

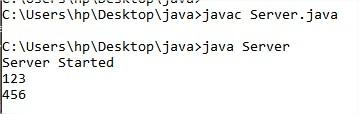
**}catch(Exception e){**

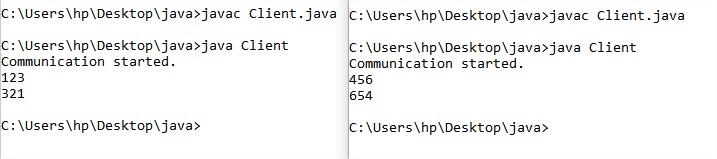
**e.printStackTrace();**

**}**

**}}**

**OUTPUT:**

****

****

# Practical 4

**Objective:**

|  |
| --- |
| 1. write a program which prints the student\_name and id\_no from the database in reverse order and also insert the two new rows. |

**Software Environment: Eclipse, jdk1.8**

**Implementation:**

**import java.sql.\*;**

**public class LoginForm {**

**public static void main(String[] args) {**

**try{**

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

**Connection con=DriverManager.getConnection("jdbc:mysql://localhost/mysql","root","");**

**Statement stmt=con.createStatement();**

**String sql="abc",b="comp";**

**ResultSet rs=stmt.executeQuery("select \* from student");**

**rs.setFetchDirection(ResultSet.FETCH\_REVERSE);**

**rs.last();**

**while(rs.previous())**

**{**

**System.out.println("Id : "+rs.getString(1)+" Name : "+rs.getString(2)+" Branch : "+rs.getString(3));**

**}**

**}**

**catch (ClassNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

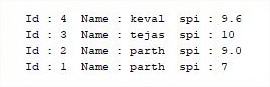
**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}}}**

**Output:**

****

**Objective:**

B. Develop an user interface that perform the following SQL operations :(i) Select (ii) Insert (iii) Update (iv) Delete

**Software Environment: Eclipse, jdk1.8**

**Implementation:**

**Main.java**

package database;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class Main {

static DBswing swingobj;

public static Connection makecoonnection() throws ClassNotFoundException,

SQLException {

Class.forName("com.mysql.jdbc.Driver"); //for mysql

Connection con = DriverManager.getConnection(

"jdbc:mysql://localhost/test", "root", "");

System.out.println("Connection Established...");

return con;

}

// for oracle s16cos153 student

public static void action() {

swingobj.btnshow.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String select = "select \* from student";

try {

Connection con = makecoonnection();

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(select);

while (rs.next()) {

System.out.println(rs.getString(1)+" "+ rs.getString(2)+" "

+ rs.getString(3));

}

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

} catch (SQLException e1) {

e1.printStackTrace();

}

}

});

swingobj.btninsert.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String id=swingobj.textid.getText();

String name=swingobj.textname.getText();

String spi=swingobj.textspi.getText();

String insert = "insert into student(id,name,spi) values(\""+id+"\",\""+name+"\",\""+spi+"\")";

try {

Connection con = makecoonnection();

Statement st = con.createStatement();

int val=st.executeUpdate(insert);

System.out.println("Successfully Inserted...");

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

} catch (SQLException e1) {

e1.printStackTrace();

}

}

});

swingobj.btnupdate.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String id=swingobj.textid.getText();

String name=swingobj.textname.getText();

String spi=swingobj.textspi.getText();

String update = "UPDATE student SET name =\"" + name + "\","

+ "spi=\"" + spi + "\" WHERE id=\"" + id + "\"";

try {

Connection con = makecoonnection();

Statement st = con.createStatement();

int val=st.executeUpdate(update);

System.out.println("Successfully Updated...");

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

} catch (SQLException e1) {

e1.printStackTrace();

}

}

});

swingobj.btndelete.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String id=swingobj.textid.getText();

String name=swingobj.textname.getText();

String spi=swingobj.textspi.getText();

String delete = "DELETE FROM student WHERE id=\""+id+"\"";

try {

Connection con = makecoonnection();

Statement st = con.createStatement();

int val=st.executeUpdate(delete);

System.out.println("Successfully Deleted...");

} catch (ClassNotFoundException e1) {

e1.printStackTrace();

} catch (SQLException e1) {

e1.printStackTrace();

}}

});

}

public static void main(String[] args) throws ClassNotFoundException,

SQLException {

String result;

swingobj = new DBswing();

action();

}

}

**DBswing.java**

package database;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class DBswing {

JFrame mainFrame;

JLabel title;

JLabel id;

JLabel name;

JLabel spi;

JTextField textid;

JTextField textname;

JTextField textspi;

JButton btninsert;

JButton btnshow;

int flag=0;

JButton btnupdate;

JButton btndelete;

public DBswing(){

mainFrame = new JFrame("Database");

mainFrame.setSize(500,500);

mainFrame.setLayout(null);

title=new JLabel("Student Table");

title.setBounds(50,00,150,20);

id=new JLabel("ID");

id.setBounds(10,40,100,20);

textid=new JTextField("");

textid.setBounds(100,40,120,20);

name=new JLabel("Name");

name.setBounds(10,70,100,20);

textname=new JTextField("");

textname.setBounds(100,70,120,20);

spi=new JLabel("SPI");

spi.setBounds(10,100,100,20);

textspi=new JTextField("");

textspi.setBounds(100,100,120,20);

btninsert=new JButton("INSERT");

btninsert.setBounds(10,130,100,20);

btnupdate=new JButton("UPDATE");

btnupdate.setBounds(10,160,100,20);

btnshow=new JButton("SHOW ALL");

btnshow.setBounds(10,190,100,20);

btndelete=new JButton("DELETE");

btndelete.setBounds(10,220,100,20);

mainFrame.add(title);

mainFrame.add(id);

mainFrame.add(textid);

mainFrame.add(name);

mainFrame.add(textname);

mainFrame.add(spi);

mainFrame.add(textspi);

mainFrame.add(btninsert);

mainFrame.add(btnupdate);

mainFrame.add(btnshow);

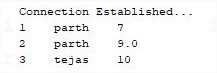
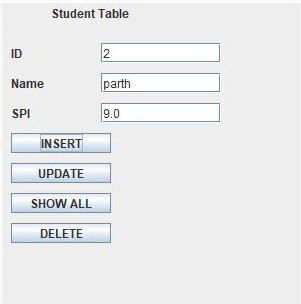
mainFrame.add(btndelete);

mainFrame.setVisible(true);

}

}

**Output:**

****