****

**Tribhuvan University**

**Faculty of Humanities and social sciences**

**Lab Report on**

**“Network Programming”**

**Submitted To:**

Department of Computer Application

Padmashree International College

***In partial fulfillment of the requirements for the bachelor’s in Computer Application***

**Submitted By:**

Ratna Katuwal

BCA 6th Semester

University Registration No: -

6-2-622-47-2020

Bhadra 28, 2081

Under the Supervision of

Rajiv Bikram Shah



**Tribhuvan University**

**Faculty of humanities and Social Sciences**

**Padmashree International College**

**Tinkune, Kathmandu**

**Supervisor’s Recommendations**

I hereby recommend that this Lab report prepared under our supervision by **Ratna Katuwal** entitled **"Lab Report on Network Programming"** is a partial fulfillment of the requirements for the degree of Bachelor of computer application is recommended for the final evaluation.

……………………………………

Mr. Rajiv Bikram Shah

Lab Supervisor

Department of BCA

Padmashree International College

Tinkune, Kathmandu



**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Padmashree International College**

**LETTER OF APPROVAL**

This is to certify that this lab report prepared by Ratna KAtuwal entitled **“Network Programming”** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| ……………………………………  Mr. Rajiv Bikram Shah  Project Supervisor  Department of Computer Application  Tinkune, Kathmandu | …………………………………..  Mr. Ramesh Kumar Pudasaini  BCA Coordinator  Department of Computer Application  Padmashree International College |
| …………………………………..  Mr. Rajiv Bikram Shah  Internal Examiner | ……………………………….  Mr.  External Examiner |

# ACKNOWLEDGEMENT

We would like to take this opportunity to express our sincere thanks to the Development of Computer Application, Tribhuvan University for providing us this opportunity to explore our interest and idea in the field of computer software through the **“Networking Programming”**. In addition, we would also like to thank, FOHSS, which include this topic on the curriculum of the 6th semester.

It gives us immense pleasure to explore our deepest sense of gratitude and sincere thanks to our supervisor **Mr. Rajiv Bikram Shah** who gave us this golden opportunity to this wonderful project on the topic **“Lab Report on Networking Programming”,** which also helped us in doing a lot of research and we came to know about many new tools and technologies.

Moreover, we would like to thank our coordinator **Mr. Ramesh Kumar Pudasaini** for arranging a good favorable environment to complete our project. At last, our parents and friends who helped us a lot in finalizing this project the limited time frame are also highly thanked.

# Lab 1: URL

**Q1. Write a Java program that extracts and prints the host name, port number, protocol, and file path from a given URL.**

**Source Code:**

import java.net.MalformedURLException;  
import java.net.URL;  
  
public class Lab1 {  
 public static void main(String[] args) throws MalformedURLException {  
 URL url = new  
 URL("https://www.youtube.com/watch?v=OfM0GqsIB6c&ab\_channel=Mrwhosetheboss");  
 try  
 {  
 System.*out*.println("host name is " + url.getHost());  
 System.*out*.println("port no. is " + url.getPort());  
 System.*out*.println("protocol used is " + url.getProtocol());  
 System.*out*.println("Your file is: " + url.getFile());  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("error"+e);  
 }  
 }  
}

**Output**:

# Lab 2: URL Connection

**Q2. How can you use the URLConnection class in Java to retrieve and display the content of a webpage from a specified URL?**

**Source Code:**

import java.io.InputStream;

import java.net.URL;

import java.net.URLConnection;

public class Lab2 {

public static void main(String[] args)

{

try

{

URL url=new URL("file:///D:/Web%20assingement/Qn-10/index.html");

URLConnection con=url.openConnection();

InputStream stream=con.getInputStream();

int i;

while((i=stream.read())!=-1)

{

System.out.print((char)i);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output**

# Lab 3: HttpURLConnection

**Q3. Write a Java program that uses HttpURLConnection to connect to a URL and**

**prints the HTTP header fields and their values.**

**Source Code:**

import java.net.HttpURLConnection;  
import java.net.URL;  
  
public class Lab3 {  
 public static void main(String[] args)  
 {  
 int i;  
 try{  
 URL url=new URL("http://www.javatpoint.com/java-tutorial");  
 HttpURLConnection con=(HttpURLConnection) url.openConnection();  
 for(i=1;i<=8;i++)  
 {  
 System.*out*.println(con.getHeaderFieldKey(i)+" = "+con.getHeaderField(i));  
 }  
 con.disconnect();  
 }  
 catch(Exception e)  
 {  
 System.*out*.println(e);  
 }  
 }  
}

**Output:**

# Lab 4: InetAddress

**Q4. Write a Java program that uses the InetAddress class to find and print the hostname and IP address of a given website.**

**Source Code:**

import java.net.InetAddress;

public class Lab4 {

public static void main(String[] args)

{

try

{

InetAddress ip= InetAddress.getByName("www.instagram.com");

System.out.println("HostName: "+ip.getHostName());

System.out.println("IP address: "+ip.getHostAddress());

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output**:

# Lab 5: DatagramSocket

**Q5. Write Java programs to implement a UDP sender and receiver using DatagramSocket where the sender sends a message to a specified IP and port, and the receiver listens on that port and prints the received message.**

**Source Code:**

**Receiver.java**

package DatagramSocket;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class Receiver {

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

DatagramSocket ds = new DatagramSocket(3000);

byte[] buff = new byte[1024];

DatagramPacket dp = new DatagramPacket(buff, 1024);

ds.receive(dp);

String str = new String(dp.getData(), 0, dp.getLength());

System.out.println("Received Message: " + str);

ds.close();

}

}

**Sender.java**

package DatagramSocket;  
  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
  
public class Sender {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket();  
 String str = "Hello Receiver! This is a message sent from the Sender. Hope you receive it well!";  
 InetAddress ip = InetAddress.*getByName*("localhost");  
 DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);  
 ds.send(dp);  
 ds.close();  
 }  
}

**Output:**

# Lab 6: One-way communication with UDP

Q6. Write Java programs to implement a UDP-based client-server chat application using DatagramSocket where the client sends messages to the server, and the server listens on a specific port, displays the received messages, and exits when it receives 'bye'.

**Source Code:**

**UDPClient.java**

package OneWayCommunication;  
  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.util.Scanner;  
  
public class UPDClient {  
 public static void main(String[] args) throws Exception  
 {  
 DatagramSocket ds = new DatagramSocket();  
 Scanner scan=new Scanner(System.*in*);  
 while(true)  
 {  
 System.*out*.println("Enter your message:");  
 String str=scan.nextLine();  
 InetAddress ip= InetAddress.*getByName*("localhost");  
 DatagramPacket dp=new DatagramPacket(str.getBytes(),str.length(),ip,2335);  
 ds.send(dp);  
 if(str.equals("bye"))  
 {  
 ds.close();  
 break;  
 }  
 }  
 }  
}

**UDPServer.java**

package OneWayCommunication;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class UDPServer {

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

DatagramSocket ds = new DatagramSocket(2335);

byte[] buff = new byte[1024];

while (true) {

DatagramPacket dpreceive = new DatagramPacket(buff, buff.length);

ds.receive(dpreceive);

String str = new String(dpreceive.getData(), 0, dpreceive.getLength());

System.out.println("Client Messaged--> " + str);

if (str.equals("bye")) {

System.out.println("Server Is Exiting .... BYE");

break;

}

buff = new byte[1024];

}

}

}

**Output:**

# Lab 7: Two-way communication using UDP

**Q7. Write Java programs to implement a two-way UDP communication system where a client and server can both send and receive messages concurrently using separate threads.**

**Source Code:**

**UDPClientTwoWay.java**

package TwoWayCommunication;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.util.Scanner;  
public class UDPClientTwoWayCommunication {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket();  
 DatagramSocket ds1 = new DatagramSocket(6001);  
 Scanner scan = new Scanner(System.*in*);  
 Thread receiveThread = new Thread(() -> {  
 try {  
 while (true) {  
 byte[] buff = new byte[1024]; DatagramPacket dpreceive = new DatagramPacket(buff, buff.length);  
 ds1.receive(dpreceive);  
 String str1 = new String(dpreceive.getData(), 0, dpreceive.getLength());  
 System.*out*.println("Server Messaged--> " + str1);  
 if (str1.equals("bye")) {  
 ds1.close();  
 break;  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 receiveThread.start();  
 while (true) {  
 System.*out*.println("Enter your message:");  
 String str = scan.nextLine();  
 InetAddress ip = InetAddress.*getByName*("localhost");  
 DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 2336);  
 ds.send(dp);  
 if (str.equals("bye")) {  
 ds.close();  
 break;  
 }  
 }  
 }  
}

**UDPServerTwoWay.java**

package TwoWayCommunication;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.util.Scanner;  
public class UDPServerTwoWayCommunication {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket(2336);  
 Scanner scan = new Scanner(System.*in*);  
 Thread receiveThread = new Thread(() -> {  
 try {  
 while (true) {  
 byte[] buff = new byte[1024];  
 DatagramPacket dpreceive = new DatagramPacket(buff, buff.length);  
 ds.receive(dpreceive);  
 String str = new String(dpreceive.getData(), 0, dpreceive.getLength());  
 System.*out*.println("Client Messaged--> " + str);  
 if (str.equals("bye")) {  
 System.*out*.println("Server Is Exiting .... BYE");  
 ds.close();  
 break;  
 }  
 buff = new byte[1024];  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 receiveThread.start();while (true) {  
 System.*out*.println("Enter your message:");  
 String str1 = scan.nextLine();  
 InetAddress ip = InetAddress.*getByName*("localhost");  
 DatagramPacket dp = new DatagramPacket(str1.getBytes(), str1.length(), ip, 6001);  
 ds.send(dp);  
 if (str1.equals("bye")) {  
 break;  
 }  
 }  
 }  
}

**Output:**

# Lab 8: ServerSocket

**Q8. Write Java programs to implement a TCP client-server communication system where the client sends a message to the server, and the server receives and displays the message using Socket and ServerSocket classes.**

**ServerSocketClass.java**

package ServerSocket;  
import java.io.DataInputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
public class ServerSocketClass {  
 public static void main(String[] args) throws Exception {  
 ServerSocket ss = new ServerSocket(9000);  
 Socket s = ss.accept();  
 DataInputStream dis = new DataInputStream(s.getInputStream());  
 String str = dis.readUTF();  
 System.*out*.println("Message:-> " + str);  
 ss.close();  
 }  
}

**ServerSocket.java**

package ServerSocket;  
import java.io.DataOutputStream;  
import java.net.Socket;  
public class ServerSocket {  
 public static void main(String[] args) throws Exception  
 {  
 Socket s=new Socket("localhost",9000);  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 dout.writeUTF("Hello my name is pem");  
 dout.flush();  
 dout.close();  
 s.close();  
 }  
}

**Output:**

# Lab 9: Communication with TCP

**Q9. Write Java programs to implement a TCP communication system where a client sends messages to a server, and the server receives and displays the messages, with the connection closing when 'bye' is sent.**

**Source Code:**

**TCPClient.java**

package CommunicationWithTCP;  
  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class TCPClient {  
 public static void main(String[] args) throws Exception  
 {  
 Scanner scan=new Scanner(System.*in*);  
 Socket s=new Socket("localhost",5656);  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 DataInputStream dis=new DataInputStream(s.getInputStream());  
 while(true)  
 {  
 System.*out*.println("Write Your message");  
 String str=scan.nextLine();  
 dout.writeUTF(str);  
 dout.flush();  
 if(str.equals("bye"))  
 {  
 dout.close();  
 s.close();  
 break;  
 }  
 }  
 }  
}

**TCPServer.java**

package CommunicationWithTCP;  
  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.ServerSocket; *// Correct import*import java.net.Socket;  
import java.util.Scanner;  
  
public class TCPServer {  
 public static void main(String[] args) throws Exception {  
 int i = 0;  
 ServerSocket ss = new ServerSocket(5656); *// Bind server to port 5656* Socket s = ss.accept(); *// Accept client connection* Scanner scan = new Scanner(System.*in*);  
 DataOutputStream dout = new DataOutputStream(s.getOutputStream());  
 DataInputStream dis = new DataInputStream(s.getInputStream());  
  
 while (true) {  
 String str = dis.readUTF(); *// Read message from client* System.*out*.println("Client---> " + str);  
  
 if (str.equals("bye")) {  
 System.*out*.println("Client Messaged .... BYE..Exiting");  
 dis.close();  
 s.close(); *// Close socket after client says "bye"* ss.close(); *// Close server socket* break;  
 }  
 }  
 }  
}

**Output:**

# Lab 10:Two-way communication with TCP

**Q10. Write Java programs to implement a two-way TCP communication system where a client and server can both send and receive messages, with the communication terminating when 'bye' is sent.**

**Source Code:**

**TCPClient.java**

package TwoWayCommunicationTCP;  
  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.Socket;  
import java.util.Scanner;  
public class TCPClient {  
 public static void main(String[] args) throws Exception  
 {  
 Scanner scan=new Scanner(System.*in*);  
 Socket s=new Socket("localhost",5656);  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 DataInputStream dis=new DataInputStream(s.getInputStream());  
 while(true)  
 {  
 System.*out*.println("Write Your message");  
 String str=scan.nextLine();  
 dout.writeUTF(str);  
 dout.flush();  
 if(str.equals("bye"))  
 {  
 dout.close();  
 s.close();  
 break;  
 }  
 String str1=(String)dis.readUTF();  
 System.*out*.println("Server Messaged--->"+str1);  
 }  
 }  
}

**TCPServer.java**

package TwoWayCommunicationTCP;  
  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.Scanner;  
public class TCPServer {  
 public static void main(String[] args) throws Exception  
 {  
 int i=0;  
 ServerSocket ss=new ServerSocket(5656);  
 Socket s=ss.accept();  
 Scanner scan=new Scanner(System.*in*);  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 DataInputStream dis=new DataInputStream(s.getInputStream());  
 while(true)  
 {  
 String str=dis.readUTF();  
 System.*out*.println("Client--->"+str);  
 if(str.equals("bye"))  
 {  
 System.*out*.println("Client Messaged .... BYE..Exiting");  
 dis.close();  
 s.close();  
 ss.close();  
 break;  
 }  
 System.*out*.println("Enter your message:");  
 String str1=scan.nextLine();  
 dout.writeUTF(str1);  
 dout.flush();  
 }  
 }  
}

**Output:**

# Lab 11: Two-way communication Local

**Q11. Write Java programs to implement a two-way TCP communication system where a client and server communicate by sending and receiving messages, with communication terminating when 'bye' is sent.**

**Source Code:**

**MyClient.java**

package TwowaycommunicationLocal;  
  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.Socket;  
import java.util.Scanner;  
public class MyClient {  
 public static void main(String[] args)throws Exception{  
 Socket s=new Socket("192.168.43.153",6666);  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 DataInputStream din=new DataInputStream(s.getInputStream());  
 Scanner scan=new Scanner(System.*in*);  
 while(true)  
 {  
 System.*out*.println("Enter Your message.");  
 String str=scan.nextLine();  
 dout.writeUTF(str);  
 dout.flush();  
 if(str.equals("bye"))  
 {  
 dout.close();  
 s.close();  
 break;  
 }  
 String str1=(String)din.readUTF();  
 System.*out*.println("Server--->"+str1);  
 }  
 }  
}

**MyServer.java**

package TwowaycommunicationLocal;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.Scanner;  
  
public class MyServer {  
 public static void main(String[] args) throws Exception  
 {  
 ServerSocket ss=new ServerSocket(6666);  
 Socket s=ss.accept();  
 DataInputStream dis=new DataInputStream(s.getInputStream());  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 Scanner scan=new Scanner(System.*in*);  
 while(true)  
 {  
 String str=(String)dis.readUTF();  
 System.*out*.println("Client--->"+str);  
 if(str.equals("bye"))  
 {  
 System.*out*.println("Client Said Bye exiting");  
 dis.close();  
 dout.close();  
 s.close();  
 ss.close();  
 break;  
 }  
 System.*out*.println("Enter Your Message");  
 String str1=scan.nextLine();  
 dout.writeUTF(str1);  
 dout.flush();  
 }  
 }  
}

**Output:**

# Lab 12: Multithreaded Server

**Q12. Write Java programs to implement a multithreaded TCP server that can handle multiple clients concurrently, with each client communicating with the server through separate threads, and a client application that sends and receives messages to/from the server.**

**Source Code:**

**Client.java**

package MultithreadedServer;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.Socket;  
import java.util.Scanner;  
public class Client {  
 public static void main(String[] args) throws Exception {  
 Socket s=new Socket("localhost",6666);  
 DataOutputStream dout= new DataOutputStream(s.getOutputStream());  
 DataInputStream din=new DataInputStream(s.getInputStream());  
 Scanner scan=new Scanner(System.*in*);  
 while(true) {  
 String strrecived=din.readUTF();  
 System.*out*.println("Server Messaged:"+strrecived);  
 System.*out*.println("Write Your Message");  
 String strtosend=scan.nextLine();  
 dout.writeUTF(strtosend);  
 dout.flush();  
 if(strtosend.equals("Exit")) {  
 System.*out*.println("Client "+s+" is exiting");  
 s.close();  
 din.close();  
 dout.close();  
 break;  
 }  
 }  
 }  
}

**Server.java**

package MultithreadedServer;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
public class Server {  
 public static void main(String[] args) throws Exception {  
 ServerSocket ss=new ServerSocket(6666);  
 while(true) {  
 Socket s=ss.accept();  
 System.*out*.println("A new client is connected via "+s);  
 DataInputStream dis=new DataInputStream(s.getInputStream());  
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
 dout.writeUTF("Enter the name of the client");  
 String name=dis.readUTF();  
 System.*out*.println("Assigning a new thread to the client");  
 Thread t1=new ClientHandler(s,name,dis,dout);  
 t1.start();  
 }  
 }  
}  
class ClientHandler extends Thread {  
 Socket s;  
 DataInputStream dis;  
 DataOutputStream dout;  
 String strrecieved="",strsent="",name;  
 ClientHandler(Socket s,String name,DataInputStream dis,DataOutputStream dout) {  
 this.s=s;  
 this.dis=dis;  
 this.dout=dout;  
 this.name=name;  
 }  
 public void run() {  
 try {  
 while(true) {  
 dout.writeUTF("What is your name? Written already write Exit.");  
 strrecieved=dis.readUTF();  
 if(strrecieved.equals("Exit")) {  
 System.*out*.println("Client "+this.s+" EXITED");  
 this.s.close();  
 this.dis.close();  
 this.dout.close();  
 break;  
 }  
 System.*out*.println("Client Messaged "+strrecieved);  
 }  
 }  
 catch(Exception e) {  
 System.*out*.println(e);  
 }  
 }  
}

**Output:**

# Conclusion

In conclusion, this lab exercise effectively demonstrates key network programming concepts using Java. By implementing both UDP and TCP communication protocols, we gained hands-on experience with different types of data transmission and connection methods. The single-threaded and multithreaded server implementations provided insights into managing multiple clients and handling concurrent interactions. Through these exercises, we developed a solid understanding of how to establish reliable and efficient communication channels in networked applications. Overall, the lab highlighted the importance of protocol choice, message handling, and concurrency in building robust networked systems.