CN LAB CODES

1. ONE WAY COMMUNICATION VIA CLIENT-SERVER

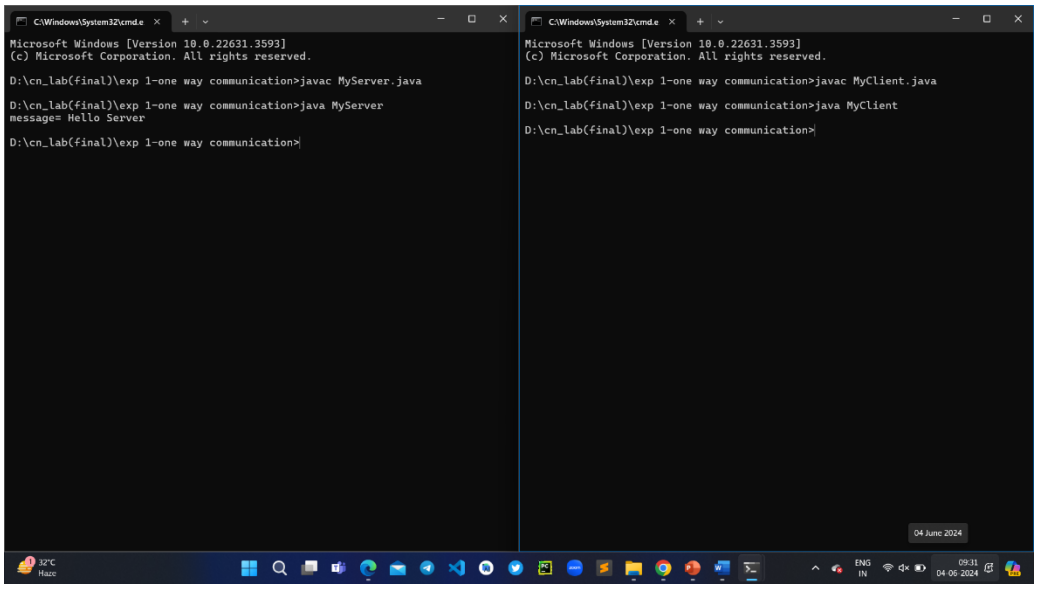
MyServer

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
```

MyClient

```
import java.io.*;
import java.net.*;
public class MyClient {
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello Server");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
```

}



1ii. **BOTH WAY COMMUNICATION VIA CLIENT-SERVER.**

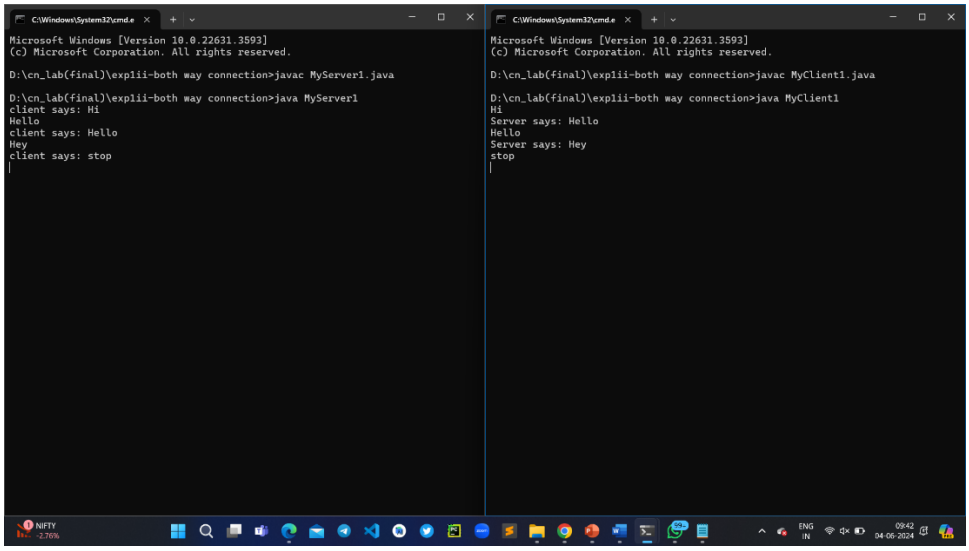
MyServer1.java

```
import java.net.*;

import java.io.*;

class MyServer1{

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

ServerSocket ss=new ServerSocket(3333);

Socket s=ss.accept();

DataInputStream din=new DataInputStream(s.getInputStream());

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";

while(!str.equals("stop")){

str=din.readUTF();

System.out.println("client says: "+str);

str2=br.readLine();

dout.writeUTF(str2);

dout.flush();

}

din.close();
```

```
s.close();
ss.close();
}}
```

MyClient1

```
import java.net.*;
import java.io.*;
class MyClient1{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String str="",str2="";
while(!str.equals("stop")){
str=br.readLine();
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Server says: "+str2);
}
dout.close();
s.close();
}}
```



1. BOTH WAY CONNECTION TO DO A STRING REVERSE PROGRAM

MyServer1

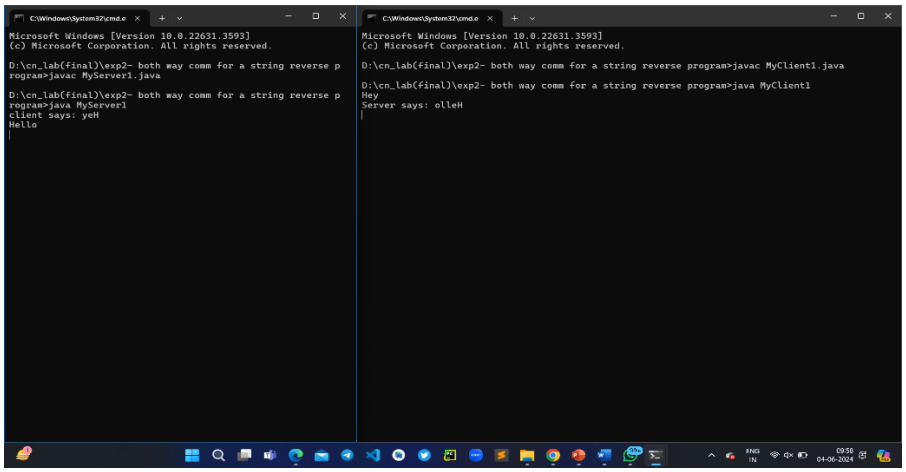
```
import java.net.*;
import java.io.*;
class MyServer1{
public static void main(String args[])throws Exception{
ServerSocket ss=new ServerSocket(3333);
Socket s=ss.accept();
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String str="",str2="";
while(!str.equalsIgnoreCase("stop")){
str=din.readUTF();
System.out.println("client says: "+str);
str2=reverseString(br.readLine());
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
```

```
}
private static String reverseString(String s){
StringBuilder sb=new StringBuilder(s);
return sb.reverse().toString();
}
}
```

MyClient1

```
import java.net.*;
import java.io.*;
class MyClient1{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String str="",str2="";
while(!str.equalsIgnoreCase("stop")){
str=br.readLine();
str2=reverseString(str);
dout.writeUTF(str2);
dout.flush();
str=din.readUTF();
System.out.println("Server says: "+str);
}
dout.close();
s.close();
}
private static String reverseString(String s){
StringBuilder sb=new StringBuilder(s);
return sb.reverse().toString();
}
```

}



2. IMPLEMENT CALCULATOR USING BOTH WAY CONN VIA CLIENT-SERVER

Calserver

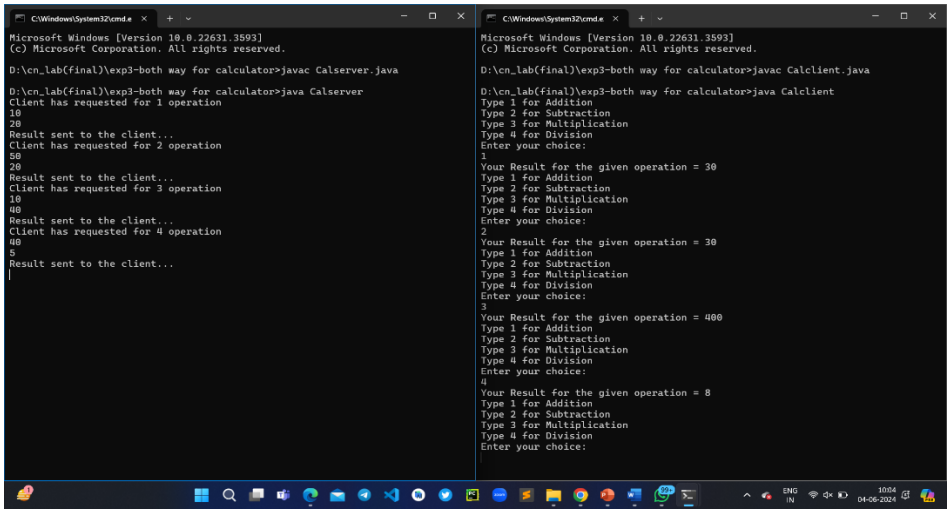
```
import java.io.*;
import java.net.*;
import java.util.*;
public class Calserver {
public static void main(String args[]) throws IOException {
ServerSocket Serve = new ServerSocket(6666);
Socket sock = Serve.accept();
DataInputStream inpStrm = new DataInputStream(sock.getInputStream());
DataOutputStream outpStrm = new DataOutputStream(sock.getOutputStream());
Scanner sc = new Scanner(System.in);
try {
while (true) {
int oprtr = inpStrm.readInt();
System.out.println("Client has requested for " + oprtr + " operation");
int res = 0;
int data1 = sc.nextInt();
int data2 = sc.nextInt();
switch(oprtr) {
case 1 :
res = data1 + data2;
```

```
outpStrm.writeUTF(Integer.toString(res));
break;
case 2 :
res = data1 - data2;
outpStrm.writeUTF(Integer.toString(res));
break;
case 3 :
res = data1 * data2;
outpStrm.writeUTF(Integer.toString(res));
break;
case 4 :
res = data1 / data2;
outpStrm.writeUTF(Integer.toString(res));
break;
default :
outpStrm.writeUTF(" You have given invalid choice! ");
break;
}
System.out.println("Result sent to the client...");
}
}
catch(Exception exp) {
System.out.println(exp);
}
}
}
```

Calclient

```
import java.io.*;
import java.net.*;
import java.util.*;
public class Calclient {
```

```
public static void main(String[] args) throws IOException {
InetAddress addr = InetAddress.getLocalHost();
Scanner inp = new Scanner(System.in);
Socket sock = new Socket(addr, 6666);
DataInputStream inpStrm = new DataInputStream(sock.getInputStream());
DataOutputStream outpStrm = new DataOutputStream(sock.getOutputStream());
try {
while (true) {
System.out.println("Type 1 for Addition");
System.out.println("Type 2 for Subtraction");
System.out.println("Type 3 for Multiplication");
System.out.println("Type 4 for Division");
System.out.println("Enter your choice: ");
int oprtr = inp.nextInt();
if (oprtr == 0) {
break;
}
outpStrm.writeInt(oprtr);
String res = inpStrm.readUTF();
System.out.println("Your Result for the given operation = " + res);
}
}
catch(Exception exp) {
System.out.println(exp);
}
}
}
```



3. STOP AND WAIT ARQ USING BOTH WAY VIA CLIENT-SERVER

Serversnw

```
import java.io.*;
import java.net.*;
import java.util.*;
public class Serversnw {
private static ServerSocket serverSocket;
private static Socket socket;
private static DataInputStream dataInputStream;
private static DataOutputStream dataOutputStream;
public static void main(String[] args) {
try {
serverSocket = new ServerSocket(8080);
System.out.println("Server is listening on port 8080...");
socket = serverSocket.accept();
System.out.println("Client connected.");
dataInputStream = new DataInputStream(socket.getInputStream());
dataOutputStream = new DataOutputStream(socket.getOutputStream());
```

```
String receivedMessage;
String sentMessage;
do {
// Receive message from client
receivedMessage = dataInputStream.readUTF();
System.out.println("Received: " + receivedMessage);
// Send ACK to client
dataOutputStream.writeUTF("ACK");
dataOutputStream.flush();
System.out.println("Sent ACK.");
// Send message to client
System.out.print("Enter message to send: ");
sentMessage = new Scanner(System.in).nextLine();
dataOutputStream.writeUTF(sentMessage);
dataOutputStream.flush();
System.out.println("Sent: " + sentMessage);
} while (!sentMessage.equalsIgnoreCase("stop") &&
!receivedMessage.equalsIgnoreCase("stop"));
// Close resources
dataInputStream.close();
dataOutputStream.close();
socket.close();
serverSocket.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

Clientsnw

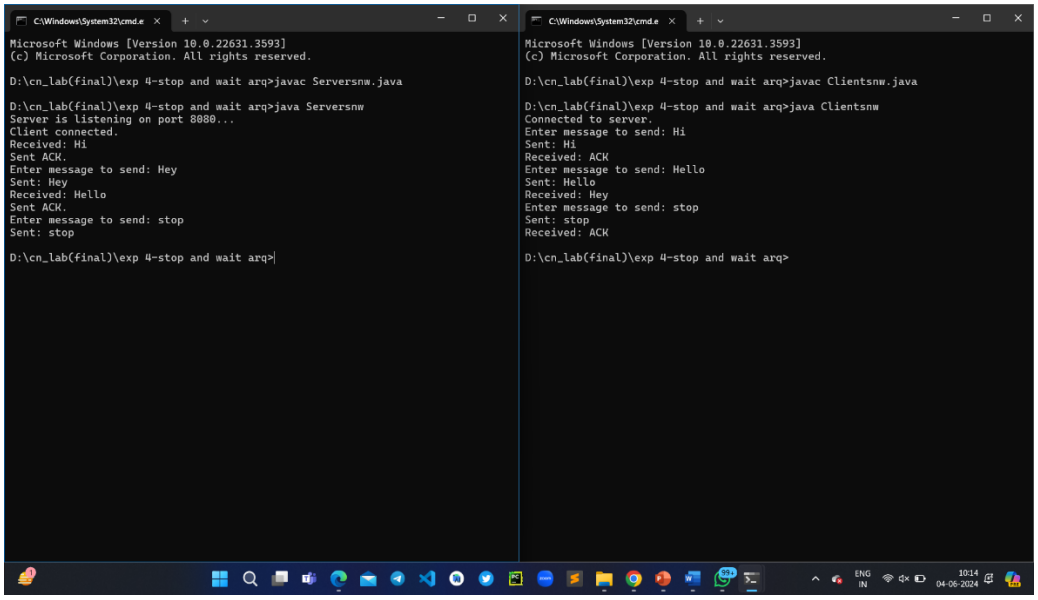
```
import java.io.*;
import java.net.*;
```

```
import java.util.*;
public class Clientsnw {
private static Socket socket;
private static DataInputStream dataInputStream;
private static DataOutputStream dataOutputStream;
public static void main(String[] args) {
try {
socket = new Socket("localhost", 8080);
System.out.println("Connected to server.");
dataInputStream = new DataInputStream(socket.getInputStream());
dataOutputStream = new DataOutputStream(socket.getOutputStream());
String receivedMessage;
String sentMessage;
do {
// Send message to server
System.out.print("Enter message to send: ");
sentMessage = new Scanner(System.in).nextLine();
dataOutputStream.writeUTF(sentMessage);
dataOutputStream.flush();
System.out.println("Sent: " + sentMessage);
// Receive ACK from server
receivedMessage = dataInputStream.readUTF();
System.out.println("Received: " + receivedMessage);
} while (!sentMessage.equalsIgnoreCase("stop") &&
!receivedMessage.equalsIgnoreCase("stop"));
// Close resources
dataInputStream.close();
dataOutputStream.close();
socket.close();
} catch (IOException e) {
e.printStackTrace();
```

}

}

}



4. PATTERN OF ALPHABETS AND NUMBERS SEQUENTIALLY USING CLIENT-SERVER BOTH WAY.

Serverabcd

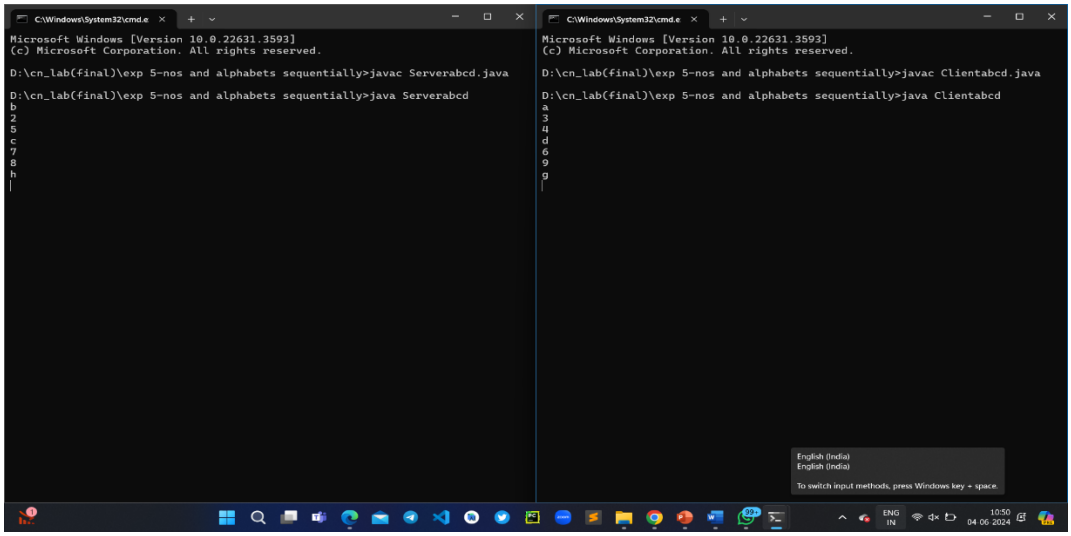
```
import java.net.*;
import java.io.*;
public class Serverabcd{
public static void main(String arg[]) throws Exception{
ServerSocket ss = new ServerSocket(1234);
Socket s = ss.accept();
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = "", str2 = "";
while(!str.equals("Stop")){
str = din.readUTF();
if(str.equals("Stop"))
{
str2 = str;
}
```

```
else
{
char ch = str.charAt(0);
ch+=1;
System.out.println(ch);
str2 = br.readLine();
}
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
}
}
```

Clientabcd

```
import java.net.*;
import java.io.*;
public class Clientabcd{
public static void main(String arg[]) throws Exception{
//InetAddress ia = InetAddress.getLocalHost();
Socket s = new Socket("localhost",1234);
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = "", str2 = "";
while(!str.equals("Stop")){
str = br.readLine();
dout.writeUTF(str);
dout.flush();
str2 = din.readUTF();
```

```
if(str2.equals("Stop"))
break;
char ch = str2.charAt(0);
ch +=1;
System.out.println(ch);
}
dout.flush();
dout.close();
s.close();
}
}
```



5. PATTERN OF ALPHABETS AND NUMBERS SEQUENTIALLY IN REVERSE ORDER USING CLIENT-SERVER BOTH WAY.

Serverabcdrev

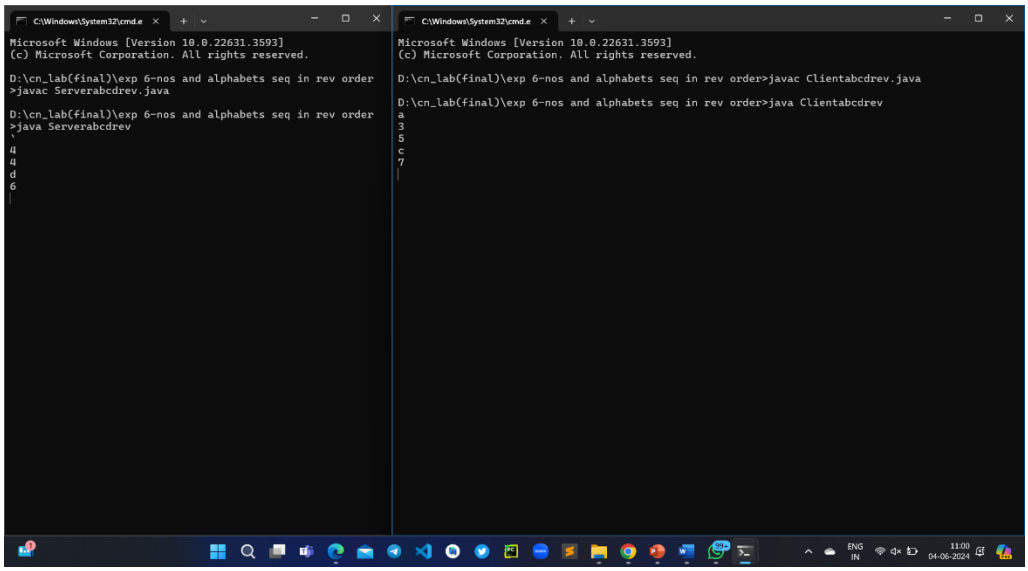
```
import java.net.*;
import java.io.*;
public class Serverabcdrev{
public static void main(String arg[]) throws Exception{
ServerSocket ss = new ServerSocket(1234);
Socket s = ss.accept();
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = "", str2 = "";
while(!str.equals("Stop")){
str = din.readUTF();
if(str.equals("Stop"))
{
str2 = str;
}
else
{
char ch = str.charAt(0);
ch-=1;
System.out.println(ch);
str2 = br.readLine();
}
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
}
}
```

Clientabcdrev

```
import java.net.*;
import java.io.*;
public class Clientabcdrev{
public static void main(String arg[]) throws Exception{
//InetAddress ia = InetAddress.getLocalHost();
Socket s = new Socket("localhost",1234);
DataInputStream din = new DataInputStream(s.getInputStream());
```

```
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = "", str2 = "";
while(!str.equals("Stop")){
str = br.readLine();
dout.writeUTF(str);
dout.flush();
str2 = din.readUTF();
if(str2.equals("Stop"))
break;
char ch = str2.charAt(0);
ch -=1;
System.out.println(ch);
}
dout.flush();
dout.close();
s.close();
}
}
```



6. SELECTIVE REPEAT ARQ USING BOTH WAY CLIENT-SERVER.

SRs

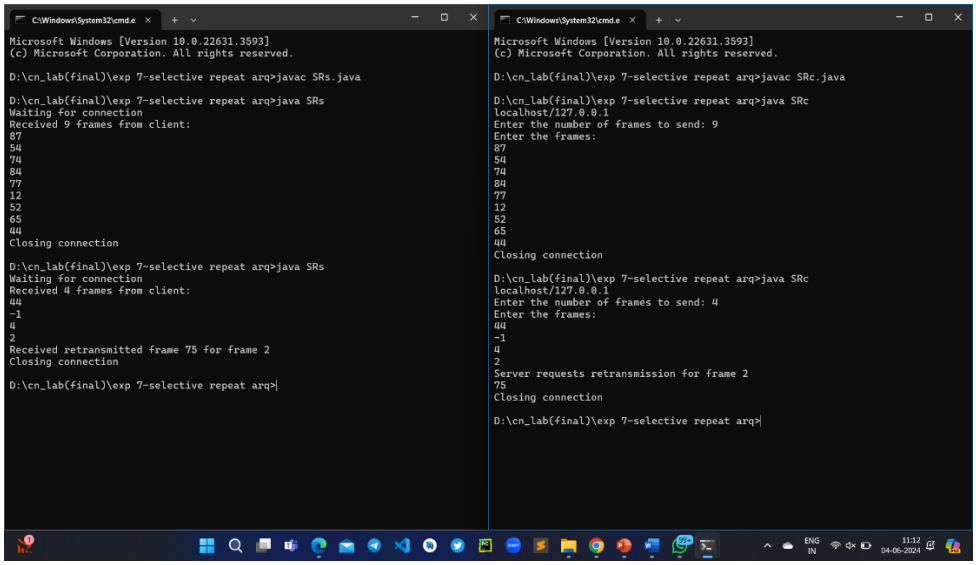
```
import java.io.*;
import java.net.*;
public class SRs {
static ServerSocket serverSocket;
static DataInputStream dis;
static DataOutputStream dos;
static int[] frames;
public static void main(String[] args) {
try {
serverSocket = new ServerSocket(8011);
System.out.println("Waiting for connection");
Socket client = serverSocket.accept();
dis = new DataInputStream(client.getInputStream());
dos = new DataOutputStream(client.getOutputStream());
int numOfFrames = dis.readInt();
frames = new int[numOfFrames];
System.out.println("Received " + numOfFrames + " frames from client:");
for (int i = 0; i < numOfFrames; i++) {
frames[i] = dis.readInt();
System.out.println(frames[i]);
}
int request = requestRetransmission();
dos.writeInt(request);
dos.flush();
if (request != -1) {
int retransmitFrame = dis.readInt();
frames[request] = retransmitFrame;
System.out.println("Received retransmitted frame " + retransmitFrame + " for frame " + (request
+ 1));
}
System.out.println("Closing connection");
```

```
} catch (IOException e) {
System.out.println(e);
} finally {
try {
dis.close();
dos.close();
serverSocket.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
static int requestRetransmission() {
for (int i = 0; i < frames.length; i++) {
if (frames[i] == -1) {
return i;
}
}
return -1;
}
}
```

SRc

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class SRc {
static Socket connection;
public static void main(String a[]) {
try {
Scanner scanner = new Scanner(System.in);
InetAddress addr = InetAddress.getByName("localhost");
```

```
System.out.println(addr);
connection = new Socket(addr, 8011);
DataOutputStream out = new DataOutputStream(connection.getOutputStream());
DataInputStream in = new DataInputStream(connection.getInputStream());
System.out.print("Enter the number of frames to send: ");
int numOfFrames = scanner.nextInt();
out.writeInt(numOfFrames);
out.flush();
System.out.println("Enter the frames:");
for (int i = 0; i < numOfFrames; i++) {
int frame = scanner.nextInt();
out.writeInt(frame);
out.flush();
}
int request = in.readInt();
if (request != -1) {
System.out.println("Server requests retransmission for frame " + (request + 1));
int retransmitFrame = scanner.nextInt();
out.writeInt(retransmitFrame);
out.flush();
}
System.out.println("Closing connection");
connection.close();
scanner.close();
} catch (IOException e) {
System.out.println(e);
}
}
}
```



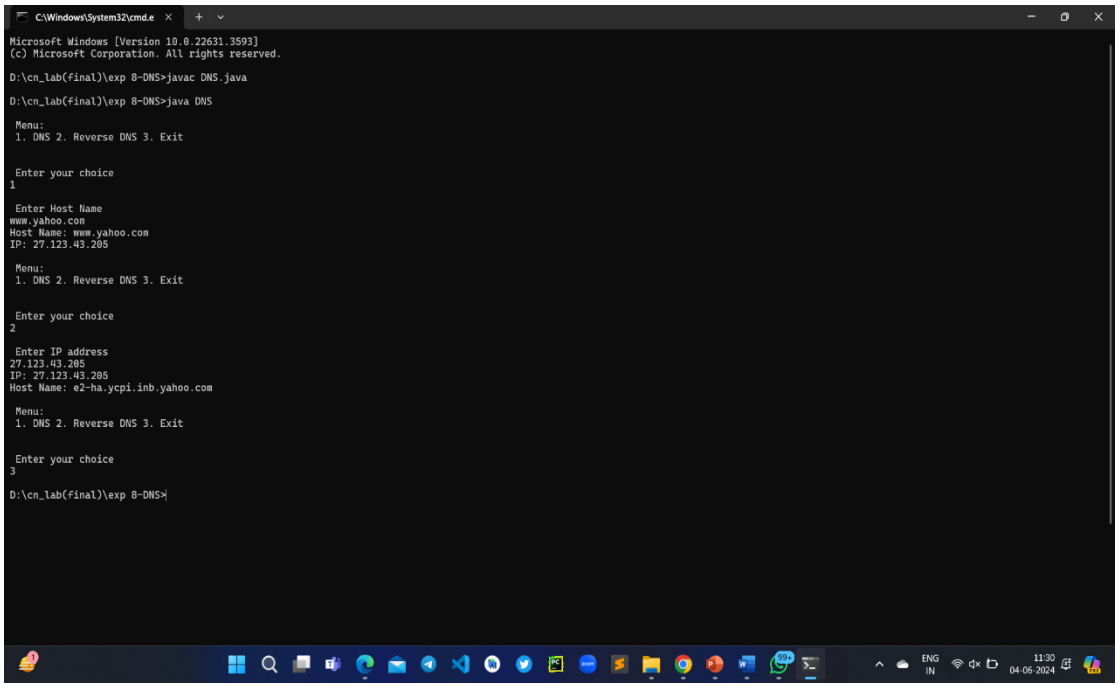
7. DNS PROGRAM

DNS

```
import java.net.*;
import java.io.*;
import java.util.*;
public class DNS
{
public static void main(String[] args)
{
int n;
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
do
{
System.out.println("\n Menu: \n 1. DNS 2. Reverse DNS 3. Exit \n");
System.out.println("\n Enter your choice");
n = Integer.parseInt(System.console().readLine());
if(n==1)
{
```

```
try
{
System.out.println("\n Enter Host Name ");
String hname=in.readLine();
InetAddress address;
address = InetAddress.getByName(hname);
System.out.println("Host Name: " + address.getHostName());
System.out.println("IP: " + address.getHostAddress());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
if(n==2)
{
try
{
System.out.println("\n Enter IP address");
String ipstr = in.readLine();
InetAddress ia = InetAddress.getByName(ipstr);
System.out.println("IP: "+ipstr);
System.out.println("Host Name: " +ia.getHostName());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
}while(!(n==3));
}
```

}



8. STRING REVERSE, CONCATENATION, UPPERCASE, LOWERCASE, SENTENCE USING BOTH WAY CLIENT

Serverstr

```
import java.net.ServerSocket;

import java.net.Socket;

import java.io.*;

class Serverstr {

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

ServerSocket ss = new ServerSocket(8888);

Socket s = ss.accept();

DataInputStream din = new DataInputStream(s.getInputStream());

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

System.out.println("Connected successfully...");

int op;

do {

String sop = din.readUTF();

op = Integer.parseInt(sop);

System.out.println("Option given by the client: " + op);

String str1 = "", str2 = "";
```

```
switch (op) {
case 1:
str1 = din.readUTF();
System.out.println("String given by the client: " + str1);
for (int i = 0; i < str1.length(); i++) {
str2 = str1.charAt(i) + str2;
}
dout.writeUTF(str2);
dout.flush();
break;
case 2:
str1 = din.readUTF();
System.out.println("String given by the client: " + str1);
str2 = din.readUTF();
System.out.println("String given by the client: " + str2);
str2 = str1 + " " + str2;
dout.writeUTF(str2);
dout.flush();
break;
case 3:
str1 = din.readUTF();
System.out.println("String given by the client: " + str1);
str2 = str1.toUpperCase();
dout.writeUTF(str2);
dout.flush();
break;
case 4:
str1 = din.readUTF();
System.out.println("String given by the client: " + str1);
str1 = str1.toLowerCase();
char[] arr = str1.toCharArray();
```

```
for (int i = 0; i < arr.length; i++) {
arr[i] = Character.toLowerCase(arr[i]);
}
str2 = new String(arr);
dout.writeUTF(str2);
dout.flush();
break;
case 5:
str1 = din.readUTF();
System.out.println("String given by the client: " + str1);
if (!str1.isEmpty()) {
str2 = Character.toUpperCase(str1.charAt(0)) + str1.substring(1).toLowerCase();
}
dout.writeUTF(str2);
dout.flush();
break;
case 6:
break;
default:
System.out.println("WRONG INPUT");
break;
}
} while (op != 6);
din.close();
s.close();
ss.close();
}
}
```

Clientstr

```
import java.net.Socket;
import java.io.*;
```

```
class Clientstr {
public static void main(String args[]) throws Exception {
Socket s = new Socket("localhost", 8888);
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int op;
do {
System.out.println("1-Reverse");
System.out.println("2-Concate");
System.out.println("3-Uppercase");
System.out.println("4-Lowercase");
System.out.println("5-Sentence");
System.out.println("6-Exit");
System.out.println("Enter your Option");
String sop = br.readLine();
op = Integer.parseInt(sop);
dout.writeUTF(sop);
String str1 = "", str2 = "";
switch (op) {
case 1:
System.out.println("Enter a string");
str1 = br.readLine();
dout.writeUTF(str1);
str2 = din.readUTF();
System.out.println("Reverse: " + str2);
dout.flush();
break;
case 2:
System.out.println("Enter a string");
str1 = br.readLine();
```

```
System.out.println("Enter a string");
str2 = br.readLine();
dout.writeUTF(str1);
dout.writeUTF(str2);
dout.flush();
str2 = din.readUTF();
System.out.println("Concate: " + str2);
break;
case 3:
System.out.println("Enter a String: ");
str1 = br.readLine();
dout.writeUTF(str1);
str2 = din.readUTF();
System.out.println("Uppercase: " + str2);
break;
case 4:
System.out.println("Enter a String: ");
str1 = br.readLine();
dout.writeUTF(str1);
str2 = din.readUTF();
System.out.println("Lowercase: " + str2);
break;
case 5:
System.out.println("Enter a String: ");
str1 = br.readLine();
dout.writeUTF(str1);
str2 = din.readUTF();
System.out.println("Sentence: " + str2);
break;
case 6:
break;
```

```
default:
System.out.println("WRONG INPUT");
break;
}
} while (op != 6);
dout.close();
s.close();
}
}
```