CN mini practical

1️⃣ HTTP Client

```
import java.io.*;import java.net.*;
class A{public static void main(String[]a)throws Exception{
Socket s=new Socket("example.com",80);
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
o.println("GET / HTTP/1.0\n");BufferedReader i=new BufferedReader(new I
nputStreamReader(s.getInputStream()));
String l;while((l=i.readLine())!=null)System.out.println(l);s.close();}}
```

Out:

`<html>Example Domain...</html>`

2️⃣ Echo Server / Client

Server

```
import java.io.*;import java.net.*;
class S{public static void main(String[]a)throws Exception{
ServerSocket ss=new ServerSocket(5000);
Socket s=ss.accept();BufferedReader i=new BufferedReader(new InputStre
amReader(s.getInputStream()));
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
String m;while((m=i.readLine())!=null)o.println("Echo:"+m);}}
```

Client

```
import java.io.*;import java.net.*;
class C{public static void main(String[]a)throws Exception{
Socket s=new Socket("localhost",5000);
BufferedReader c=new BufferedReader(new InputStreamReader(System.i
n));
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInput
Stream()));
String m;while(!(m=c.readLine()).equals("bye")){o.println(m);System.out.pri
ntln(i.readLine());}}}
```

Out:

`Hi → Echo:Hi`

3️⃣ Chat Program

Server

```
import java.io.*;import java.net.*;
class CS{public static void main(String[]a)throws Exception{
ServerSocket ss=new ServerSocket(5001);Socket s=ss.accept();
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInput
Stream()));
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
BufferedReader k=new BufferedReader(new InputStreamReader(System.i
n));
String m;while(!(m=i.readLine()).equals("bye")){System.out.println("Cli:"+
m);o.println(k.readLine());}}}
```

Client

```
import java.io.*;import java.net.*;
class CC{public static void main(String[]a)throws Exception{
Socket s=new Socket("localhost",5001);
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInput
Stream()));
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
BufferedReader k=new BufferedReader(new InputStreamReader(System.i
n));
String m;while(!(m=k.readLine()).equals("bye")){o.println(m);System.out.pri
ntln("Srv:"+i.readLine());}}}
```

Out:

`Client: hi → Server: hey!`

4️⃣ File Server

Server

```
import java.io.*;import java.net.*;
class FS{public static void main(String[]a)throws Exception{
ServerSocket ss=new ServerSocket(5002);Socket s=ss.accept();
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInput
Stream()));
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
File f=new File(i.readLine());
if(f.exists()){BufferedReader fr=new BufferedReader(new FileReader(f));
String l;while((l=fr.readLine())!=null)o.println(l);}else o.println("No file");}}
```

Client

```
import java.io.*;import java.net.*;
class FC{public static void main(String[]a)throws Exception{
Socket s=new Socket("localhost",5002);
PrintWriter o=new PrintWriter(s.getOutputStream(),true);
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInput
Stream()));
o.println("data.txt");String l;while((l=i.readLine())!=null)System.out.println
(l);}}
```

Out:

`Hello World`

5️⃣ DNS

```
import java.net.*;
class DNS{public static void main(String[]a)throws Exception{
System.out.println(InetAddress.getByName("google.com").getHostAddress
());}}
```

Out:

`142.250.x.x`

6️⃣ ARP

```
import java.util.*;
class ARP{public static void main(String[]a){
Map<String,String>m=Map.of("1.1.1.1","AA:BB","1.1.1.2","CC:DD");
Scanner sc=new Scanner(System.in);
System.out.println(m.getOrDefault(sc.next(),"Not found"));}}
```

Out:

`Enter IP:1.1.1.1 → AA:BB`

7️⃣ Distance Vector

```
class DVR{public static void main(String[]a){
int[][]c={{0,2,7},{2,0,1},{7,1,0}};
for(int i=0;i<3;i++){System.out.print("R"+i+": ");
for(int j=0;j<3;j++)System.out.print(c[i][j]+" ");System.out.println();}}}
```

Out:

`R0:0 2 7`

8️⃣ Link State (Dijkstra)

```
import java.util.*;
class LSR{public static void main(String[]a){
int[][]g={{0,4,0,0,8},{4,0,8,0,11},{0,8,0,7,0},{0,0,7,0,9},{8,11,0,9,0}};
int n=g.length,d[]=new int[n];boolean[]v=new boolean[n];
Arrays.fill(d,9999);d[0]=0;
for(int i=0;i<n;i++){int u=-1,m=9999;
for(int j=0;j<n;j++)if(!v[j]&&d[j]<m){m=d[j];u=j;}
v[u]=true;
for(int k=0;k<n;k++)if(g[u][k]>0&&!v[k]&&d[u]+g[u][k]<d[k])d[k]=d[u]+g[u]
[k];}
for(int i=0;i<n;i++)System.out.println("N"+i+":"+d[i]);}}
```

Out:

`N0:0 N1:4 N2:12 N3:19 N4:8`

🔟 Commands

Command	Primary Use	PDU Relation & Level
`tcpdump`	**Packet analyzer**. Intercepts and displays live network traffic.	Directly captures **PDUs** (Frames, Packets, Segments). (**L2, L3, L4**)
`netstat`	Displays **active connections** and port usage.	Reports on the **state** of Transport Layer connections (TCP Segments/UDP Datagrams). (**L4**)
`ifconfig`	Views/manages **network interface configuration** (IP, MAC, etc.).	Displays the essential addressing info used in **L2 Frame** and **L3 Packet** headers. (**L2, L3**)
`nslookup`	Queries DNS servers for **name-to-IP resolution**.	Initiates a **DNS Query/Response** encapsulated within a **UDP Datagram PDU**. (**L4, L7**)
`traceroute`	Determines the **path/hops** packets take to a destination.	Relies on sending **ICMP Packets** and receiving **ICMP Time Exceeded** messages from routers. (**L3**)