Mumbai University

Master Code

B.Sc.IT: Semester – VI

Practical (Master Code)

[CBSGS – 75:25 Pattern]

Internet Technology

Internet Technologies (Practical Questions) [Java Code]

```
===============
ArpClient.java
===============
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
class ArpClient
{
public static void main(String args[]) throws Exception
{
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
Socket clientSocket=new Socket("localhost",8000);
DataInputStream dataIn=new DataInputStream(clientSocket.getInputStream());
DataOutputStream dataOut=new DataOutputStream(clientSocket.getOutputStream());
System.out.println("Enter the Logical address(IP from 192.168.1.1 to192.168.1.5)
");
String str1=in.readLine();
dataOut.writeBytes(str1+'\n');
String str=dataIn.readLine();
System.out.println("The Physical Address is: "+str);
clientSocket.close();
}
}
```

```
==============
ArpServer.java
==============
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
class ArpServer
{
public static void main(String args[]) throws Exception
{
System.out.println(": : : : :Server Ready: : :");
{
ServerSocket  serverSocket=new ServerSocket(8000);
Socket socket=serverSocket.accept();
while(true)
{
DataInputStream din=new DataInputStream(socket.getInputStream());
DataOutputStream dout=new DataOutputStream(socket.getOutputStream());
String str=din.readLine();
String
ipAdress[]={"192.168.1.1","192.168.1.2","192.168.1.3","192.168.1.4","192.168.1.5"};
String
macAdress[]={"2A:07:AC:C2","CC:DD:E3:FA","9A:07:BB:CC","AA:BB:E3:FA","BB:CC:8E:EA"}
;
for(int i=0;i<ipAdress.length;i++)
{
if(str.equals(ipAdress[i]))
{
dout.writeBytes(macAdress[i]+'\n');
break;
}
}
}
}
}
}

===============
ChatClient.java
===============
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

class chatclient
{
private static int port=1001;
private static String host="local host";
public static void main(String[] args) throws IOException
{
Socket server;
PrintWriter out=null;
try
{
server=new Socket(host,port);
out=new PrintWriter(server.getOutputStream(),true);
}
catch(UnknownHostException e)
{
System.err.println(e);
System.exit(1);
}
BufferedReader stdln=new BufferedReader(new InputStreamReader(System.in));
String msg;
```

```
while((msg=stdln.readLine())!=null)
{
out.println(msg);
}
}
}

===============
ChatServer.java
===============
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.net.ServerSocket;

class ChatServer
{
private static int port=1001;
public static void main(String[] args) throws IOException
{
ServerSocket server=null;
try
{
server=new ServerSocket(port);
}
catch(IOException e)
{
System.err.println("Could not listen on port:"+port);
System.err.println(e);
System.exit(1);
}
Socket client=null;
try
{
client=server.accept();
}
catch(IOException e)
{
System.err.println("Accept failed ");
System.err.println(e);
System.exit(1);
}
BufferedReader in=new BufferedReader(new
InputStreamReader(client.getInputStream()));
String msg;
```

```
while((msg=in.readLine())!=null)
{
System.out.println("Client says:"+msg);
}
}
}

==================
DataClient7.java
==================
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class DataClient7
{
public static void main(String args[]) throws IOException
{
String host="localhost";
byte data[]=new byte[1024];
InetAddress address=InetAddress.getByName(host);
System.out.println("checking at:  " + address);
DatagramPacket dataPacket=new DatagramPacket (data,data.length,address,13);
DatagramSocket socket=new DatagramSocket();
socket.send(dataPacket);
dataPacket= new DatagramPacket (data,data.length);
socket.receive(dataPacket);
String time=new String(dataPacket.getData());
System.out.println("Time on the"+ host+"is:" + time);
}
}

==================
DataServer7.java
==================
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Date;
public class DataServer7
{
public static void main(String args[]) throws IOException
{
DatagramSocket socket=new DatagramSocket(13);
while(true)
{
byte buffer[]=new byte[256];
DatagramPacket packet=new DatagramPacket(buffer,buffer.length);
socket.receive(packet);
String date=new Date().toString();
buffer=date.getBytes();
InetAddress address=packet.getAddress();
int port=packet.getPort();
packet=new DatagramPacket(buffer,buffer.length,address ,port);
socket.send(packet);
}
}
}

==================
EchoClient4.java
==================
import java.net.*;
import java.io.*;
public class EchoClient4
{
public static void main(String[]args)throws IOException
{
String serHostname=new String("localhost");
if(args.length>0)
{
serHostname=args[0];
}
System.out.println("connection to host"+serHostname+"on port 8000.");
Socket socket=null;
PrintWriter out=null;
BufferedReader in= null;
try
{
socket=new Socket(serHostname,8000);
out=new PrintWriter(socket.getOutputStream(),true);
in=new BufferedReader(new InputStreamReader(socket.getInputStream()));
}
catch(UnknownHostException e)
{
System.err.println("Unknown Host:"+serHostname);
System.exit(1);
}
catch(IOException e)
{
System.err.println("couldnt get IO for"+"the connection to:"+serHostname+"please
open the server connection first");
System.exit(1);
}
BufferedReader bin =new BufferedReader(new InputStreamReader(System.in));
```

```
String userInput;
System.out.println("input:");
while((userInput=bin.readLine())!=null)
{
out.println(userInput);
System.out.println("echo:" + in.readLine());
System.out.println("input:");
}
out.close();
in.close();
bin.close();
socket.close();
}
}

==================
EchoServer4.java
==================
import java.net.*;
import java.io.*;
public class EchoServer4
{
public static void main(String[]args)throws IOException
{
ServerSocket serverSocket=null;
Socket clientSocket=null;
try
{
serverSocket=new ServerSocket(8000);
}
catch (IOException e)
{
System.err.println("someting went wrong on port:8000");
System.exit(1);
}
System.out.println("waitin for connection...");
try
{
clientSocket =serverSocket.accept();
}
catch(IOException e)
{
System.err.println("sorry failed...");
System.exit(1);
}
System.out.println("connection succesfull");
System.out.println("plz input value");
PrintWriter out=new PrintWriter(clientSocket.getOutputStream(),true);
```

```
BufferedReader in=new BufferedReader(new 
InputStreamReader(clientSocket.getInputStream()));
String inputLine;
while((inputLine=in.readLine())!=null)
{
System.out.println("server:"+inputLine);
out.println(inputLine);
if (inputLine.equals("bye"))
{
break;
}
}
out.close();
in.close();
clientSocket.close();
serverSocket.close();
}
}

===================
GreetingClient.java
===================
import java.net.*;
import java.io.*;

public class GreetingClient
{
public static void main(String [] args)
{
String serverName = args[0];
int port = Integer.parseInt(args[1]);
try
{
System.out.println("Connecting to " + serverName + " on port " + port);
Socket client = new Socket(serverName, port);
System.out.println("Just connected to " + client.getRemoteSocketAddress());
OutputStream outToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);
out.writeUTF("Hello from " + client.getLocalSocketAddress());
InputStream inFromServer = client.getInputStream();
DataInputStream in = new DataInputStream(inFromServer);
System.out.println("Server says " + in.readUTF());
client.close();
}
catch(IOException e)
{
e.printStackTrace();
}
```

```
}
}

===================
GreetingServer.java
===================
/ File Name GreetingServer.java

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

public class GreetingServer extends Thread
{
private ServerSocket serverSocket;
public GreetingServer(int port) throws IOException
{
serverSocket = new ServerSocket(port);
serverSocket.setSoTimeout(10000);
}
public void run()
{
while(true)
{
try
{
System.out.println("Waiting for client on port " + serverSocket.getLocalPort() +
"...");
Socket server = serverSocket.accept();
System.out.println("Just connected to " + server.getRemoteSocketAddress());
DataInputStream in = new DataInputStream(server.getInputStream());
System.out.println(in.readUTF());
DataOutputStream out = new DataOutputStream(server.getOutputStream());
out.writeUTF("Thank you for connecting to " + server.getLocalSocketAddress() +
"\nGoodbye!");
server.close();
}
catch(SocketTimeoutException s)
{
System.out.println("Socket timed out!");
break;
}
catch(IOException e)
{
e.printStackTrace();
break;
}
}
}
```

```
public static void main(String [] args)
{
int port = Integer.parseInt(args[0]);
try
{
Thread t = new GreetingServer(port);
t.start();
}
catch(IOException e)
{
e.printStackTrace();
}
}
}

=================
ShortestPath.java
=================
import java.io.*;
import java.util.*;
import java.net.*;
class ShortestPath {
public static int a1, a2,b1,b2,n,m;
public static int a[][]=new int[10][10],par[][]=new int[10][10],distance[][]=new
int[10][10],i,j;
public static void main(String s[]) throws IOException{
int x,y,num;
LinkedList list=new LinkedList();
LinkedList path=new LinkedList();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the value for n & m 1st matrix");
BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));
n=Integer.parseInt(bf.readLine());
m=Integer.parseInt(bf.readLine());
System.out.println("Enter the  element for 1st matrix(0 or 1)");
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
a[i][j]=Integer.parseInt(bf.readLine());
par[i][j]=-1;
distance[i][j]=n*m;
}
}
System.out.println("Enter the position of source(x,y)");
a1=Integer.parseInt(bf.readLine());
b1=Integer.parseInt(bf.readLine());
System.out.println("Enter the position of destination(x,y)");
```

```
a2=Integer.parseInt(bf.readLine());
b2=Integer.parseInt(bf.readLine());
list.add(getnumber(a1,b1));
distance[a1][b1]=0;
while (list.size() !=0) {
num=Integer.parseInt(list.removeLast().toString());
x=num/m;
y=num%m;
if(x>0 && a[x-1][y] !=1 &&(distance[x-1][y]>(distance[x][y] +1)) )
{
list.add(getnumber(x-1,y));
distance[x-1][y]=distance[x][y]+1;
par[x-1][y]=num;
}
if(y>0 && a[x][y-1] !=1 && (distance[x][y-1]>(distance[x][y] +1)) )
{
list.add(getnumber(x,y-1));
distance[x][y-1]=distance[x][y]+1;
par[x][y-1]=num;
}
if(x<n-1 && a[x+1][y] !=1 && (distance[x+1][y]>(distance[x][y] +1)) )
{
list.add(getnumber(x+1,y));
distance[x+1][y]=distance[x][y]+1;
par[x+1][y]=num;
}
if(y<m-1 && a[x][y+1] !=1 && (distance[x][y+1]>(distance[x][y] +1)) )
{
list.add(getnumber(x,y+1));
distance[x][y+1]=distance[x][y]+1;
par[x][y+1]=num;
}
}
path.add("("+ a2 +","+ b2 +")");
x=a2;
y=b2;
do {
num=par[x][y];
x=num/m;
y=num%m;
path.add("("+ x +","+ y +")");
}
while(!(x==a1 && y==b1));
while(path.size() !=0)  {
System.out.println(path.removeLast());
}
}
public static Integer getnumber(int x,int y)  {
```

```
return new Integer(x *m+y);
}
}

=================
tcpip_client.java
=================
import java.net.*;
import java.io.*;

class tcpip_client
{
public static void main(String args[]) throws IOException
{
Socket s=null;
BufferedReader b=null;
try
{
s=new Socket(InetAddress.getLocalHost(),98);
b=new BufferedReader(new InputStreamReader(s.getInputStream()));
}
catch(UnknownHostException u)
{
System.err.println("I don't know host");
System.exit(0);
}
String inp;
while((inp=b.readLine())!=null)
{
System.out.println(inp);
}
b.close();
s.close();
}
}

=================
tcpip_server.java
=================
import java.net.*;
import java.io.*;

class tcpip_server
{
public static void main(String args[]) throws IOException
{
ServerSocket n1=null;
try
```

```
{
n1=new ServerSocket(98);
}
catch(IOException e)
{
System.err.println("Port 98 could not be found");
System.exit(1);
}
Socket c=null;
try
{
c=n1.accept();
System.out.println("Connection from "+c);
}
catch(IOException e)
{
System.out.println("Accept failed");
System.exit(1);
}
PrintWriter out=new PrintWriter(c.getOutputStream(),true);
BufferedReader in=new BufferedReader(new InputStreamReader(c.getInputStream()));
String n;
BufferedReader sin=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Ready to type now");
while((n=sin.readLine())!=null)
{
out.println(n);
}
out.close();
c.close();
n1.close();
}
}

===============
TimeClient.java
===============
import java.io.*;
import java.net.*;
class TimeClient
{
public static void main(String[]args)throws Exception
{
Socket c=new Socket(InetAddress.getLocalHost(),1234);
BufferedReader br=new BufferedReader(new InputStreamReader(c.getInputStream()));
String userInput;
while((userInput=br.readLine())!=null)
{
```

```
System.out.println(userInput);
}
c.close();

}
}

===============
TimeServer.java
===============
import java.io.*;
import java.net.*;
import java.util.*;
class TimeServer
{
public static void main(String[]args)throws Exception
{
{
ServerSocket s=new ServerSocket(1234);
Socket c=s.accept();
Calendar cal=new GregorianCalendar();
PrintWriter out=new PrintWriter(c.getOutputStream());
out.println(new Date());
out.println("TIme: ");
out.println(cal.get(Calendar.HOUR)+"Hrs:"+cal.get(Calendar.MINUTE)+"Minu"+cal.get(C
alendar.SECOND)+"Sec");
out.flush();
s.close();
c.close();
}
}
}

==============
UDPClient.java
==============
import java.io.*;
import java.net.*;

class UDPClient
{
public static void main(String args[]) throws Exception
{
BufferedReader inFromUser =
new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName("localhost");
byte[] sendData = new byte[1024];
```

```
byte[] receiveData = new byte[1024];
String sentence = inFromUser.readLine();
sendData = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
IPAddress, 9876);
clientSocket.send(sendPacket);
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String modifiedSentence = new String(receivePacket.getData());
System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
}

==================
UDPEchoClient.java
==================
import java.io.*;
import java.net.*;
public class UDPEchoClient
{
public static void main(String args[])throws Exception
{
byte[]recvBuf=new byte[256];
byte[]sendBuf=new byte[256];
DatagramSocket socket=new DatagramSocket();
InetAddress serverAddr=InetAddress.getByName("server name");
BufferedReader In=new BufferedReader (new Input StreamReader(System.in));
String sendString=In.readLine();
sendBuf=sendString.getByte();
DatagramPacket sendPacket=new
DatagramPacket(sendBuf,sendBuf.length,serverAddr,serverPort);
socket.send(sendPacket);
DatagramPacket recvString=new String(recvPacket.getData());
System.out.println("Received from server:"+recvString);
socket.close()
}
}

==================
UDPEchoServer.java
==================
import java.io.*;
import java.net.*;
public class UDPEchoServer
{
public static void main(String args[])throws Exception
{
```

```
byte[]recvBuf=new byte[256];
byte[]sendBuf=new byte[256];
DatagramSocket socket= new DatagramSocket(7);
for(;;)
{
DatagramPacket receivePacket= new Datagrampacket(recvBuf,recvBuf.length);
socket.receive(receivePacket);
DatagramPacket
sendPacket=newDatagramPacket(sendBuf,sendBuf.length,receivepacket.getAddress(),receivePacket.getPort());
socket.(sendPacket);
}
}
}

==============
UDPServer.java
==============
import java.io.*;
import java.net.*;

class UDPServer
{
public static void main(String args[]) throws Exception
{
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] receiveData = new byte[1024];
byte[] sendData = new byte[1024];
while(true)
{
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
serverSocket.receive(receivePacket);
String sentence = new String( receivePacket.getData());
System.out.println("RECEIVED: " + sentence);
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
String capitalizedSentence = sentence.toUpperCase();
sendData = capitalizedSentence.getBytes();
DatagramPacket sendPacket =
new DatagramPacket(sendData, sendData.length, IPAddress, port);
serverSocket.send(sendPacket);
}
}
}
```