2. Socket programming : implementation of connection- oriented services using standard ports.

SERVER

import java.net.\*;

import java.io.\*;

class Asimpleserver {

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

ServerSocket s=new ServerSocket(1234);

Socket s1=s.accept();

OutputStream s1Out=s1.getOutputStream();

DataOutputStream dos=new DataOutputStream(s1Out);

dos.writeUTF("hi there");

dos.close();

s1Out.close();

s1.close();

}

}

CLIENT:

import java.net.\*;

import java.io.\*;

class Asimpleclient{

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

Socket s1=new Socket("LocalHost",1234);

InputStream s1In=s1.getInputStream();

DataInputStream dis=new DataInputStream(s1In);

String st=new String(dis.readUTF());

System.out.println(st);

dis.close();

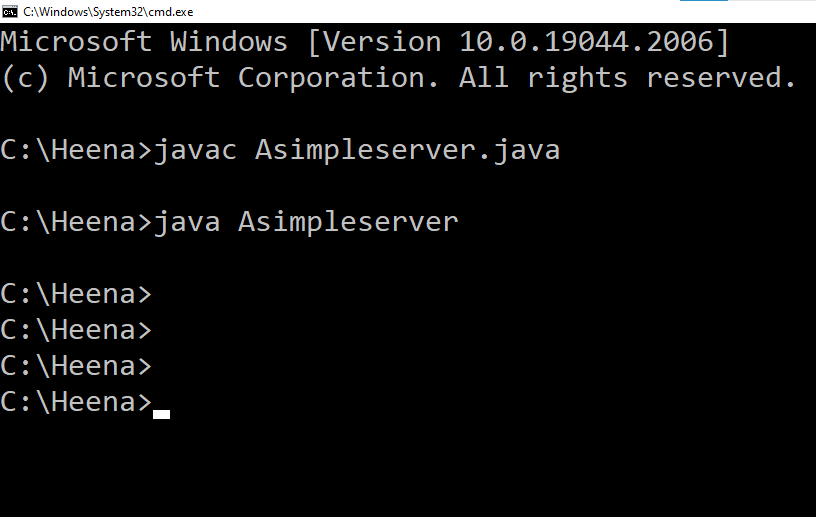
s1In.close();

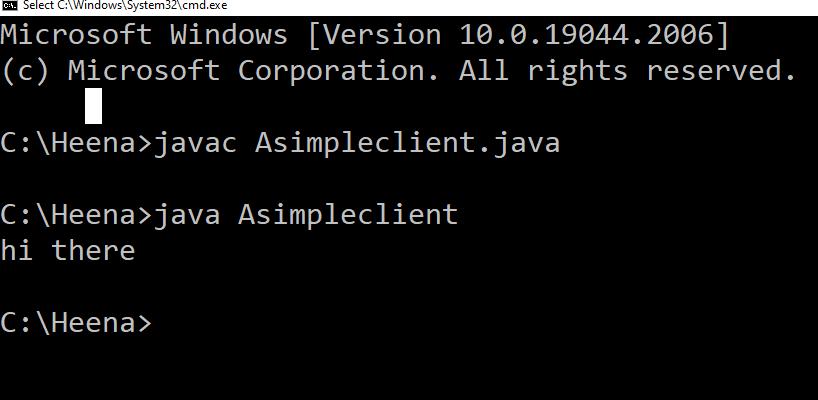
s1.close();

}

}

OUTPUT:





3. Implementation of connection less service using standard ports.

Server:

import java.net.\*;

public class Dsender{

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

DatagramSocket ds=new DatagramSocket();

String str="welcome java";

InetAddress ip=InetAddress.getByName("127.0.0.1");

DatagramPacket dp=new DatagramPacket(str.getBytes(),str.length(),ip,3000);

ds.send(dp);

ds.close();

}

}

CLIENT:

import java.net.\*;

public class Dreceiver {

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

DatagramSocket ds= new DatagramSocket(3000);

byte[] buf=new byte[1024];

DatagramPacket dp =new DatagramPacket(buf,1024);

ds.receive(dp);

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

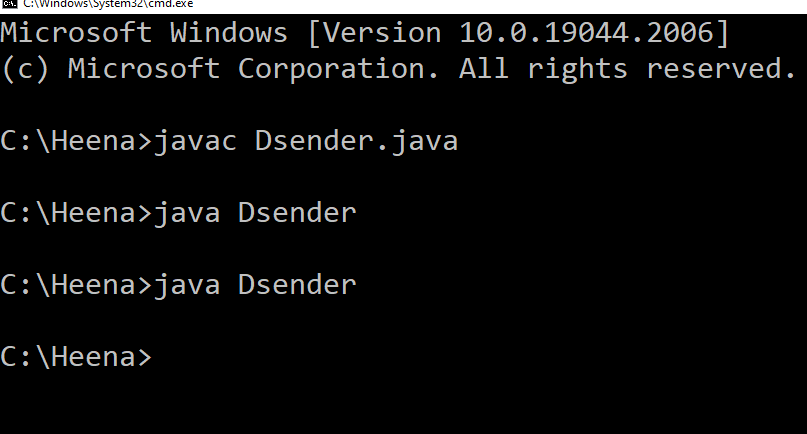
System.out.println(str);

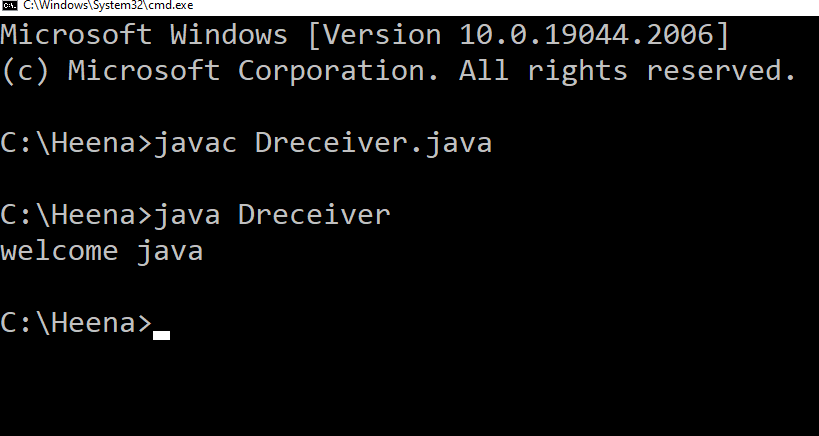
ds.close();

}

}

OUTPUT:





4. Implementation of connection-oriented iterative Echo-Server, date and time, character generation using user-defined ports.

SERVER:

import java .io.\*;

import java.net.\*;

import java.util.\*;

class DataServer

{

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

{

ServerSocket s=new ServerSocket(5217);

while(true)

{

System.out.println("waiting for connection.....");

Socket soc=s.accept();

DataOutputStream out= new DataOutputStream(soc.getOutputStream());

out.writeBytes("Server Date"+(new Date().toString()+"\n"));

out.close();

soc.close();

}

}

}

CLIENT:

import java .io.\*;

import java.net.\*;

class DataClient

{

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

{

Socket soc=new Socket(InetAddress.getLocalHost(),5217);

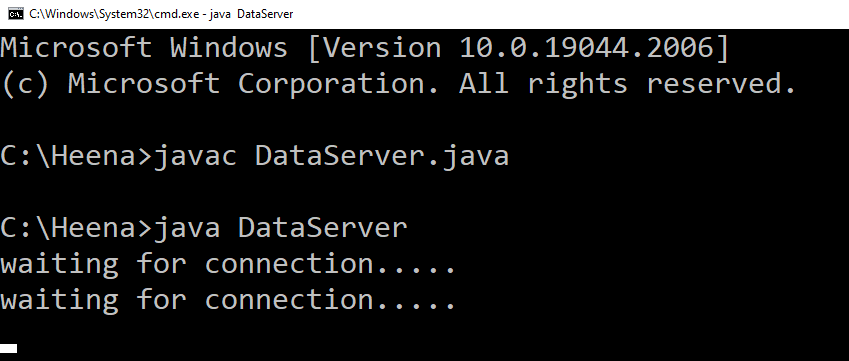
BufferedReader in=new BufferedReader(new InputStreamReader(soc.getInputStream()));

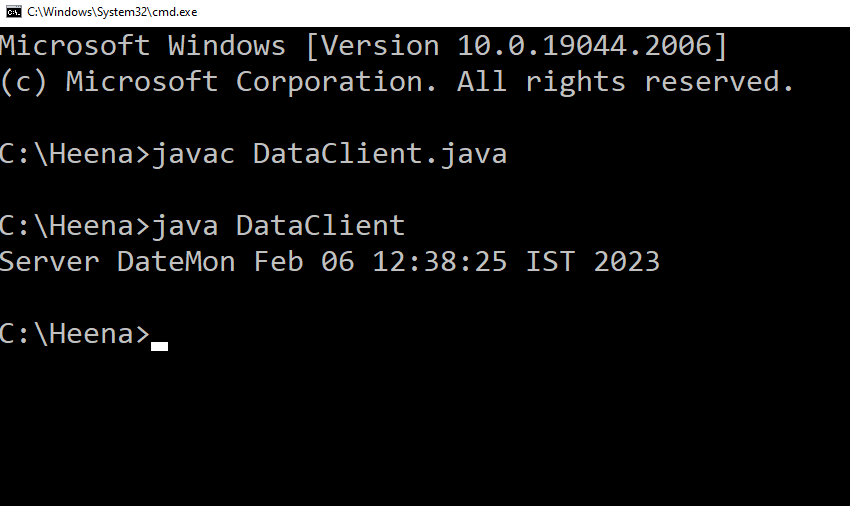
System.out.println(in.readLine());

}

}

OUTPUTS:





5. Implementation of Connectionless Iterative Echo-server, date and time, character generation using user-defined ports.

SERVER:

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Servernew

{

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

{

DatagramSocket ss=new DatagramSocket(1234);

while(true)

{

System.out.println("server is up......");

byte[] rd=new byte[100];

byte[] sd=new byte[100];

DatagramPacket rp=new DatagramPacket(rd,rd.length);

ss.receive(rp);

InetAddress ip=rp.getAddress();

int port=rp.getPort();

Date d=new Date();

String time=d+" ";

sd= time.getBytes();

DatagramPacket sp=new DatagramPacket(sd,sd.length,ip,port);

ss.send(sp);

rp=null;

System.out.println("Done!!");

}

}

}

CLIENT:

import java.net.\*;

import java.io.\*;

class Clientnew

{

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

{

System.out.println("server time>>>>>");

DatagramSocket cs=new DatagramSocket();

InetAddress ip= InetAddress.getByName("localhost");

byte[] rd=new byte[100];

byte[] sd=new byte[100];

DatagramPacket sp=new DatagramPacket(sd,sd.length,ip,1234);

DatagramPacket rp=new DatagramPacket(rd,rd.length);

cs.send(sp);

cs.receive(rp);

String time=new String(rp.getData());

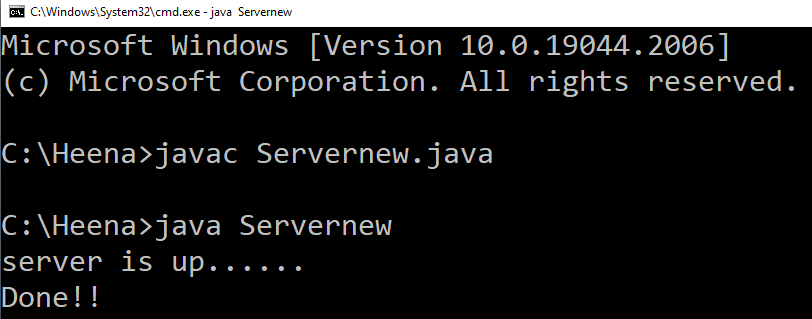
System.out.println(time);

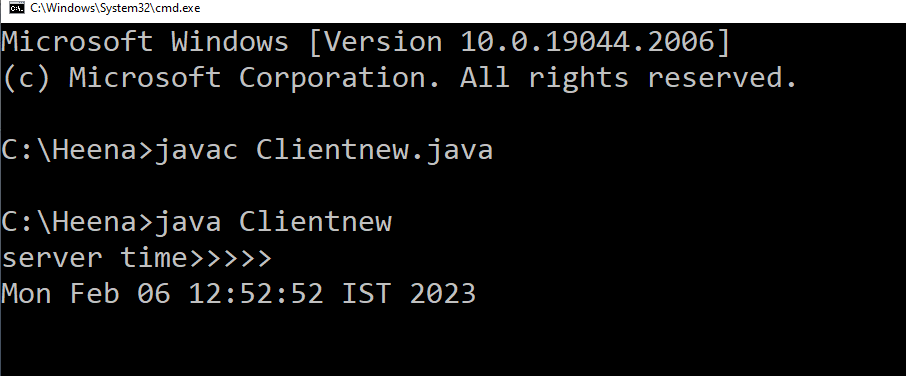
cs.close();

}

}

OUTPUTS:





6. Implementation of Connection-Oriented Concurrent Echo-server, date and time , character generation using user-defined ports.

SERVER:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class DataServer

{

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

{

ServerSocket ssocket = new ServerSocket(1234);

Socket socket = ssocket.accept();

try //(ServerSocket listener = new ServerSocket(5909));

{

System.out.println("The data server is running ....");

while(true)

{

try //(Socket socket = new listener.accept())

{

PrintWriter out = new PrintWriter(socket.getOutputStream(),true);

System.out.println(new Date().toString());

}

catch(Exception e)

{ System.out.println("Error : "+e.getMessage()); }

}

}

catch(Exception e)

{ System.out.println("Error : "+e.getMessage()); }

}

}

CLIENT:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class DataClient

{

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

{

if(args.length != 1)

{

System.err.println("pass the server IP Address to the sole command line argument");

return;

}

Socket ss = new Socket(args[0],1234);

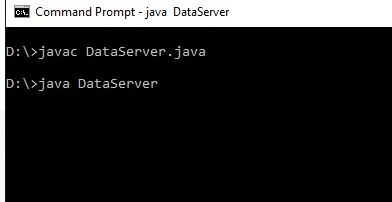
Scanner in = new Scanner(ss.getInputStream());

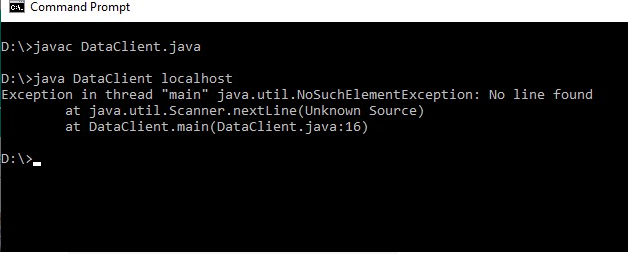
System.out.println("Server response : "+in.nextLine());

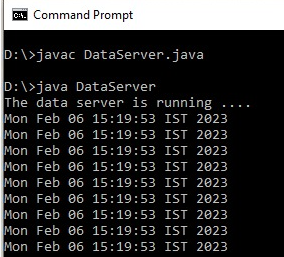
}

}

OUTPUTS:







7. Program for connection-oriented Iterative service in which server reverses the string send by client and sends it back.

SERVER:

import java.io.\*;

import java.net.\*;

class OrderReverseTCPServer {

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

{

String clientSentence;

String capitalizedSentence=null;

ServerSocket welcomeSocket=new ServerSocket(6333);

while(true){

Socket connectionSocket=welcomeSocket.accept();

BufferedReader inFromClient=new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

DataOutputStream outToClient=new DataOutputStream(connectionSocket.getOutputStream());

clientSentence=inFromClient.readLine();

String sendToClient=new StringBuilder(clientSentence).reverse().toString()+'\n';

outToClient.writeBytes(sendToClient);

}

}

}

CLIENT:

import java.io.\*;

import java.net.\*;

class OrderReverseTCPClient {

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

{

String sentence;

String modifiedSentence;

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

Socket clientSocket;

clientSocket=new Socket("localhost",6333);

DataOutputStream outToServer=new DataOutputStream(clientSocket.getOutputStream());

BufferedReader inFromServer=new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

sentence=inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence=inFromServer.readLine();

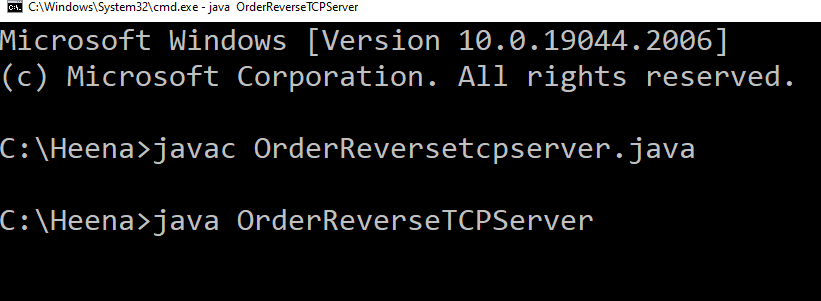
System.out.println("FROM SERVER:" +modifiedSentence);

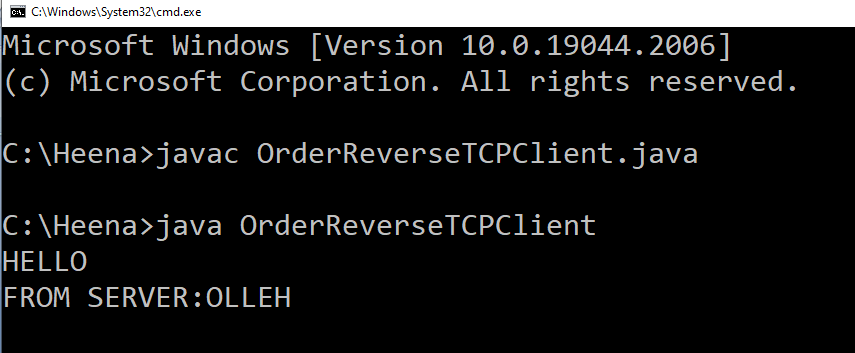
clientSocket.close();

}

}

OUTPUTS:





8. Program for connection-oriented Iterative service in which server changes the case of the strings send by client and sends back(Case Server).

SERVER:

import java.io.\*;

import java.net.\*;

class Server1

{

public static void main(String args[])

{

String messageIn;

String messageOut;

try {

ServerSocket ssock=new ServerSocket(6789);

while(true)

{

Socket connsock=ssock.accept();

InputStreamReader inStr=new InputStreamReader(connsock.getInputStream());

BufferedReader inNet=new BufferedReader(inStr);

DataOutputStream outNet=new DataOutputStream(connsock.getOutputStream());

messageIn=inNet.readLine();

messageOut=messageIn.toUpperCase()+"\n";

outNet.writeBytes(messageOut);

}

}

catch(IOException e) {

System.out.println(e.getMessage());

}

}

}

CLIENT:

import java.io.\*;

import java.net.\*;

class Client1

{

public static void main(String args[])

{

String message=null;

try {

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

Socket csock=new Socket("LocalHost",6789);

DataOutputStream outNet=new DataOutputStream(csock.getOutputStream());

BufferedReader inNet=new BufferedReader(new InputStreamReader(csock.getInputStream()));

message=kbd.readLine();

outNet.writeBytes(message+"\n");

message=inNet.readLine();

csock.close();

System.out.println("server sent:"+ message);

}

catch(IOException e) {

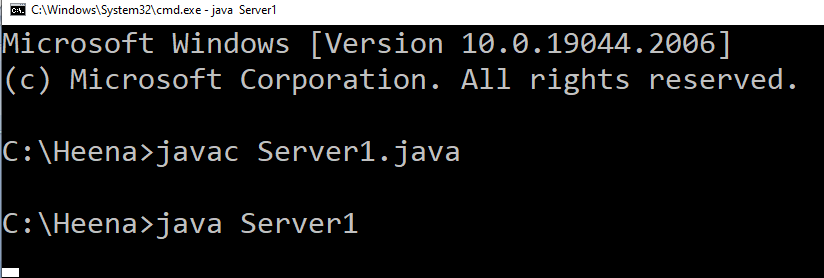
System.out.println(e.getMessage());

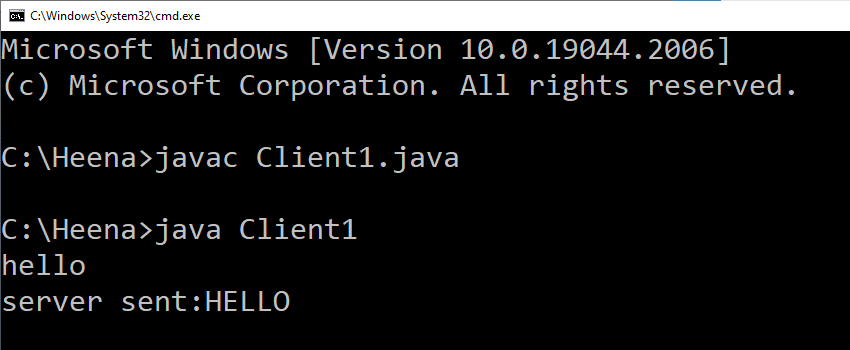
}

}

}

OUTPUTS:





9. Program for Connection-Oriented iterative service in which server calculates the net – salary of an employee based on the following details send by the clients

( i) Basic ii) hra iii)da iv)pt v)epf vi) net-salary-basic+hra+da-pt-epf)

Server:

import java.io.\*;

import java.net.\*;

class connectionServer

{

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

{

try

{

int serverListenPortNum=Integer.parseInt(args[0]);

ServerSocket connectionSocket=new ServerSocket(serverListenPortNum);

Socket dataSocket = connectionSocket.accept();

ObjectInputStream ois=new ObjectInputStream(dataSocket.getInputStream());

EmployeeData eData=(EmployeeData)ois.readObject();

System.out.println("Employee id:" + eData.getID());

System.out.println("Employee name:" + eData.getName());

System.out.println("Employee salary:" + eData.getSalary());

ois.close();

dataSocket.close();

connectionSocket.close();

}

catch(Exception e){ e.printStackTrace(); }

}

}

Client:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class connectionClient

{

public static void main(String args[])

{

try

{

InetAddress serverHost=InetAddress.getByName(args[0]);

int serverPortNum=Integer.parseInt(args[1]);

Socket clientSocket=new Socket(serverHost,serverPortNum);

EmployeeData empData=new EmployeeData();

Scanner input=new Scanner(System.in);

System.out.println("Enter employee id:");

int id=input.nextInt();

System.out.println("Enter employee name:");

String name = input.next();

System.out.println("Enter employee salary:");

double salary=input.nextDouble();

empData.setID(id);

empData.setName(name);

empData.setSalary(salary);

ObjectOutputStream oos=new ObjectOutputStream(clientSocket.getOutputStream());

oos.writeObject(empData);

oos.close();

clientSocket.close();

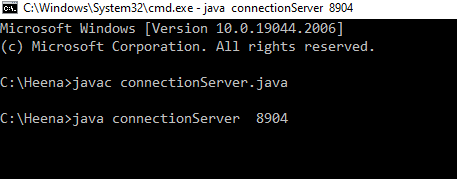
}

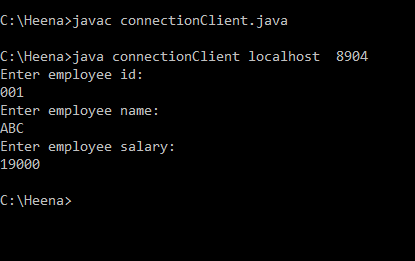
catch(Exception e) { e.printStackTrace(); }

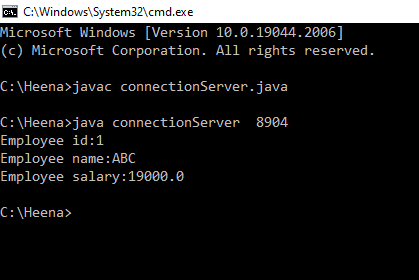
}

}

OUTPUT:







10. Program for file access using Sockets.

SERVER:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Server2

{

private static ServerSocket server;

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

{

server=new ServerSocket(1234);

Socket socket = server.accept();

BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter Out=new PrintWriter(socket.getOutputStream());

String request;

while((request=in.readLine())!=null)

{

boolean pinCorrect=checkpin(request);

Out.println(pinCorrect? "yes":"no");

Out.flush();

}

Out.close();

socket.close();

}

private static boolean checkpin(String pin)throws IOException

{

boolean results=false;

File file=new File("bold.tnt");

BufferedReader in=new BufferedReader(new InputStreamReader(new FileInputStream(file)));

String line;

while((line=in.readLine())!=null)

{

results =line.equals(pin);

}

in.close();

return results;

}

}

CLIENT:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Client2

{

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

{

int pinsize=0;

Socket clientsocket = new Socket(InetAddress.getLocalHost(),1234 );

BufferedReader in=new BufferedReader(new InputStreamReader(clientsocket.getInputStream()));

PrintWriter Out=new PrintWriter(clientsocket.getOutputStream());

Scanner scanner=new Scanner(System.in);

System.out.println("Enter pin:");

String password=scanner.next();

pinsize=password.length();

Out.println(password);

if(pinsize!=4)

{

System.out.println("pin must be 4digits");

}

else

{

System.out.println("checking....");

}

Out.flush();

String response = in.readLine();

System.out.println("response");

in.close();

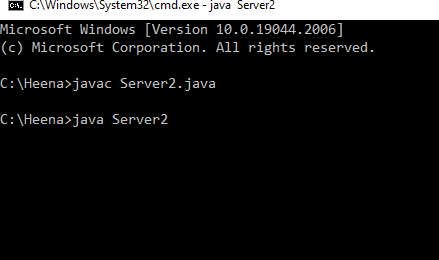
Out.close();

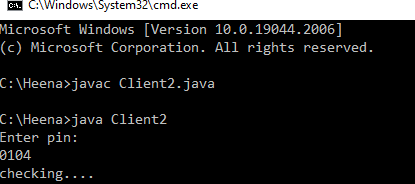
clientsocket.close();

}

}

OUTPUT:





11. Program for Remote Command Execution using sockets.

SERVER:

import java.io.\*;

import java.net.\*;

class RemoteServer

{

public static void main(String args[])

{

try

{

int port;

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

System.out.println("Enter the Port Address : ");

port = Integer.parseInt(buf.readLine());

ServerSocket ss = new ServerSocket(port);

System.out.println("Waiting ....");

Socket s = ss.accept();

if(s.isConnected() == true)

{

System.out.println("Client Socket is Connected Successfully. ");

}

InputStream in = s.getInputStream();

OutputStream ou = s.getOutputStream();

//BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));

String cmd = buf.readLine();

PrintWriter pr = new PrintWriter(ou);

pr.println(cmd);

Runtime H = Runtime.getRuntime();

//Process p = H.excel(cmd);

System.out.println("The "+cmd+"Command is executed successfully");

pr.flush();

pr.close();

ou.close();

in.close();

}

catch(Exception e)

{

System.out.println("Error : "+e.getMessage());

}

}

}

CLIENT:

import java.io.\*;

import java.net.\*;

class RemoteClient

{

public static void main(String args[])

{

try

{

int port;

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

System.out.println("Enter the port Address : ");

port = Integer.parseInt(buf.readLine());

Socket s = new Socket("localhost",port);

if(s.isConnected() == true)

{

System.out.println("Server Socket is connected Successfully. ");

}

InputStream in = s.getInputStream();

OutputStream ou = s.getOutputStream();

//BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));

//BufferedReader buf = new BufferedReader(new InputStreamReader(in));

PrintWriter pr = new PrintWriter(ou);

System.out.println("Enter the command to be Executed : ");

pr.println(buf.readLine());

pr.flush();

String str = buf.readLine();

System.out.println(" "+str+" Command is Executed Successfully. ");

pr.close();

ou.close();

in.close();

}

catch(Exception e)

{

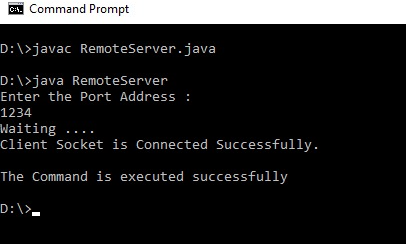
System.out.println("Error : "+e.getMessage());

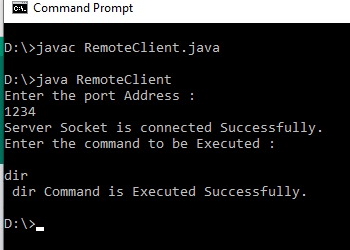
}

}

}

OUTPUTS:





12. Implementation of DNS.

import java.net.\*;

import java.io.\*;

import java.util.\*;

class DNS

{

public static void main(String args[])

{

int n;

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

do

{

System.out.println("\nMenu:\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("Enter HostName ");

String hname=in.readLine();

InetAddress address=InetAddress.getByName(hname);

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

}

catch (IOException ioe)

{

ioe.printStackTrace();

}

}

}

while(!(n==3));

}

}

OUTPUT:

