Assignment -3

**1. Write a TCP based program to read a file from another machine.**

**#TCPFileServer.java :-**

package Assignment3.Question1;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileReader;

import java.net.ServerSocket;

import java.net.Socket;

public class TCPFileServer

{

public static void main(String[] args)

{

try

{

// Connection With Client

ServerSocket ss= new ServerSocket(5001);

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

Socket stk=ss.accept();

System.out.println("Connection Established.");

DataInputStream dis=new DataInputStream(stk.getInputStream());

DataOutputStream dos=new DataOutputStream(stk.getOutputStream());

String filePath = "";

while(!filePath.equalsIgnoreCase("Quit"))

{

// Get File

filePath=dis.readUTF();

if(filePath.equalsIgnoreCase("Quit")) break;

System.out.println(filePath+"...");

File f=new File(filePath);

if (f.isFile())

{

System.out.println("file found");

}

// Read & Send File

BufferedReader reader = new BufferedReader(new FileReader(f));

String line;

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

{

dos.writeUTF(line);

}

dos.writeUTF("EOF");

reader.close();

}

// Close

dis.close();

dos.close();

System.out.print("Connection Ended...");

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**#Output :-**

Waiting for Client ...

Connection Established.

book\_list.txt...

file found

LICENSE...

file found

Connection Ended...

**#TCPFileClient.java :-**

package Assignment3.Question1;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.Socket;

import java.util.Scanner;

public class TCPFileClient

{

public static void main(String[] args)

{

try

{

Socket socket = new Socket("localhost",5001);

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

System.out.println("Connection Established");

Scanner scan = new Scanner(System.in);

// Send File Name to Server

String fileName = "";

while(!fileName.equals("Quit"))

{

System.out.print("\nEnter the File Name : ");

fileName = scan.next() ;

if(fileName.equalsIgnoreCase("Quit")) break;

dos.writeUTF(fileName);

// Read File from Server

System.out.println("File Content....");

String fileContent = dis.readUTF();

while(!fileContent.equals("EOF"))

{

System.out.println(fileContent);

fileContent = dis.readUTF();

}

System.out.println("End Of File....\n");

}

dos.writeUTF("Quit");

// Close

dos.close();

dis.close();

socket.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**#Output :-**

Connection Established

Enter the File Name : book\_list.txt

File Content....

1\*Advance Java\*Mammta Padole\*MSU\*31-12-2024\*5000.0\*5\*5500.0

2\*Life of CR\*Sahil Bhanderi\*Bhanderi Publication\*01-03-2030\*0.0\*200\*1000.0

3\*Data Structures\*Anjali Sharma\*R.K. Publishers\*15-06-2025\*2000.0\*10\*3000.0

4\*Algorithms\*Rajesh Patel\*Patel & Sons\*25-12-2027\*1500.0\*8\*1800.0

5\*Web Development\*Neha Mehta\*MSU\*10-11-2026\*0.0\*12\*2500.0

6\*Database Systems\*Karan Joshi\*Joshi Enterprises\*02-02-2031\*500.0\*20\*1500.0

7\*Computer Networks\*Suresh Chavda\*National Book House\*18-07-2028\*3500.0\*15\*4000.0

8\*Software Engineering\*Pooja Desai\*Desai Print\*30-04-2029\*1000.0\*5\*1200.0

9\*Operating Systems\*Ravi Shah\*Shah Publications\*21-08-2024\*700.0\*7\*800.0

10\*Artificial Intelligence\*Priya Thakkar\*MSU\*03-05-2027\*0.0\*25\*2800.0

11\*Machine Learning\*Vishal Gupta\*Gupta Bookstore\*16-09-2025\*4500.0\*12\*5000.0

12\*Cloud Computing\*Meera Rathi\*Modern Prints\*29-01-2032\*900.0\*6\*1100.0

13\*Networking Fundamentals\*Arjun Jain\*Jain Press\*07-10-2026\*300.0\*18\*1300.0

14\*Computer Architecture\*Swati Kumar\*Kumar Publishing\*12-03-2028\*2200.0\*9\*2600.0

15\*Cyber Security\*Manoj Solanki\*Solanki Distributors\*08-12-2030\*5000.0\*4\*5500.0

16\*Digital Marketing\*Simran Kaur\*MSU\*23-02-2029\*0.0\*30\*3200.0

17\*Blockchain Technology\*Akash Verma\*Verma & Co.\*04-06-2024\*1800.0\*11\*2100.0

18\*Business Intelligence\*Divya Nair\*Nair Publications\*11-07-2025\*3500.0\*14\*3800.0

19\*Web Security\*Nikhil Soni\*Soni Print Hub\*28-09-2027\*750.0\*13\*1050.0

20\*Game Development\*Ritika Jain\*Jain Enterprises\*09-11-2029\*2500.0\*8\*2900.0

21\*Reignite\*Abdul Kalam\*Science Books\*12-02-2011\*500.0\*2\*1200.0

End Of File....

Enter the File Name : LICENSE

File Content....

MIT License

Copyright (c) 2024 Khushi 💜

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

End Of File....

Enter the File Name : quit

**2. Write a UDP based program to read a file from another machine.**

**#UDPFileServer.java :-**

package Assignment3.Question2;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import org.omg.CosNaming.NamingContextPackage.NotFound;

public class UDPFileServer

{

public static void main(String[] args)

{

try

{

DatagramSocket server = new DatagramSocket(2001);

System.out.println("Server is running...");

DatagramPacket packet;

String filePath = "";

while (true)

{

// Receive File Path

byte[] filePathByte = new byte[1024];

packet = new DatagramPacket(filePathByte, filePathByte.length);

server.receive(packet);

filePath = new String(packet.getData(), 0, packet.getLength()).trim();

if (filePath.equalsIgnoreCase("Quit")) break;

System.out.println("Requested file: " + filePath);

File f = new File(filePath);

if (f.isFile())

{

System.out.println("File found. Sending...");

BufferedReader reader = new BufferedReader(new FileReader(f));

String line;

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

{

byte[] fileContentByte = line.getBytes();

packet = new DatagramPacket(fileContentByte, fileContentByte.length,

packet.getAddress(), packet.getPort());

server.send(packet);

}

reader.close();

// Send EOF signal

String eof = "EOF";

byte[] eofByte = eof.getBytes();

packet = new DatagramPacket(eofByte, eofByte.length, packet.getAddress(), packet.getPort());

server.send(packet);

}

else

{

System.out.println("File Not Found");

String notFound = "File not Found...!";

byte[] notFoundByte = notFound.getBytes();

packet = new DatagramPacket(notFoundByte, notFoundByte.length, packet.getAddress(), packet.getPort());

server.send(packet);

}

}

System.out.println("Server shutting down...");

server.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**# Output :-**

Server is running...

Requested file: README.md

File found. Sending...

Server shutting down...

**#UDPFileClient.java :-**

package Assignment3.Question2;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPFileClient

{

public static void main(String[] args)

{

try

{

DatagramSocket client = new DatagramSocket(1304);

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

Scanner scan = new Scanner(System.in);

String filePath = "";

while (true)

{

// Get file name from user

System.out.print("\nEnter the File Name (or 'Quit' to exit): ");

filePath = scan.next();

if (filePath.equalsIgnoreCase("Quit")) break;

// Send file name to server

byte[] filePathByte = filePath.getBytes();

DatagramPacket packet = new DatagramPacket(filePathByte, filePathByte.length, ip, 2001);

client.send(packet);

// Receive file content

System.out.println("Receiving file content...");

while (true)

{

byte[] fileContentByte = new byte[1024];

packet = new DatagramPacket(fileContentByte, fileContentByte.length);

client.receive(packet);

String fileContent = new String(packet.getData(), 0, packet.getLength()).trim();

if (fileContent.equalsIgnoreCase("EOF"))

{

System.out.println("End Of File...\n");

break;

}

if(fileContent.equalsIgnoreCase("File not Found...!"))

{

System.out.println("File Not Found...\n");

break;

}

System.out.println(fileContent);

}

}

// Send quit signal to server

String quit = "Quit";

byte[] quitByte = quit.getBytes();

DatagramPacket quitPacket = new DatagramPacket(quitByte, quitByte.length, ip, 2001);

client.send(quitPacket);

System.out.println("Client shutting down...");

client.close();

scan.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Enter the File Name (or 'Quit' to exit): README.md

Receiving file content...

# Advance\_java\_Assignments

End Of File...

Enter the File Name (or 'Quit' to exit): quit

Client shutting down...

**3. Write a Java program to open and read a file on local machine. Encrypt that file and transmit to another machine. At the receiving end, the file should be opened and read again, after decrypting the data.**

**# EncryptFile.java :-**

package Assignment3.Question3;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Base64;

public class EncryptFile

{

public static String encrypt(String content)

{

return Base64.getEncoder().encodeToString(content.getBytes());

}

public static void main(String[] args)

{

try

{

ServerSocket serverSocket = new ServerSocket(5421);

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

Socket socket = serverSocket.accept();

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

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

PrintStream ps = new PrintStream(socket.getOutputStream());

String fileName = br.readLine();

File file = new File(fileName);

FileReader fileReader = new FileReader(file);

BufferedReader brFile = new BufferedReader(fileReader);

String line = "";

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

{

String encryptedLine = encrypt(line);

ps.println(encryptedLine);

}

ps.println("EOF");

brFile.close();

fileReader.close();

ps.close();

br.close();

socket.shutdownOutput();

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Waiting for client....

Client Connected

**# DecryptFile.java :-**

package Assignment3.Question3;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.Socket;

import java.util.Base64;

public class DecryptFile

{

public static String decrypt(String encryptedContent)

{

return new String(Base64.getDecoder().decode(encryptedContent));

}

public static void main(String[] args)

{

try

{

Socket socket = new Socket("localhost",5421);

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

PrintStream ps = new PrintStream(socket.getOutputStream());

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

System.out.print("Enter the File Name : ");

String fileName = brFile.readLine();

ps.println(fileName);

brFile.close();

String line = "";

while(!(line=br.readLine()).equalsIgnoreCase("EOF"))

{

String decrLine = decrypt(line);

System.out.println(decrLine);

}

ps.close();

br.close();

socket.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Enter the File Name : LICENSE

MIT License

Copyright (c) 2024 Khushi 💜

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

**4. Write a Java program to implement EchoServer**

**# TCPEchoServer.java :-**

package Assignment3.Question4;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

public class TCPEchoServer

{

public static void main(String[] args)

{

try

{

ServerSocket serverSocket = new ServerSocket(2323);

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

Socket socket = serverSocket.accept();

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

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

String line = "";

while(!(line=br.readLine()).equalsIgnoreCase("END") && !socket.isClosed())

{

System.out.println(line);

}

br.close();

// socket.shutdownOutput();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Waiting for client....

Client Connected

Hello World...!

Tahiti a magical place.

I am Iron-man

quit

**# TCPEchoClient.java :-**

package Assignment3.Question4;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.Socket;

public class TCPEchoClient

{

public static void main(String[] args)

{

try

{

Socket socket = new Socket("localhost",2323);

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

PrintStream ps = new PrintStream(socket.getOutputStream());

String line = "";

System.out.print("Enter Message : ");

while(!(line=br.readLine()).equalsIgnoreCase("END"))

{

ps.println(line);

System.out.print("Enter Message : ");

}

ps.println("END");

ps.close();

br.close();

socket.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Enter Message : Hello World...!

Enter Message : Tahiti a magical place.

Enter Message : I am Iron-man

Enter Message : quit

Enter Message : end

**5. Write a Java program to implement ChatServer**

**# UDPChatSender.java :-**

package Assignment3.Question5;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPChatSender

{

public static void main(String[] args)

{

try

{

DatagramSocket datagramSocket = new DatagramSocket(1234);

DatagramPacket datagramPacket = null;

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

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

String message = "";

while(!message.equalsIgnoreCase("END"))

{

// Read From Client

byte b[] = new byte[1024];

datagramPacket = new DatagramPacket(b,b.length);

datagramSocket.receive(datagramPacket);

message = new String(datagramPacket.getData()).trim();

if(message.equalsIgnoreCase("END")) break;

System.out.println("Client Request : "+message);

// Write To Client

System.out.print("Serever Response : ");

message = br.readLine();

datagramPacket = new DatagramPacket(message.getBytes(), message.length(),inetAddress,4321);

datagramSocket.send(datagramPacket);

if(message.equalsIgnoreCase("END")) break;

}

br.close();

datagramSocket.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Client Request : Hello

Serever Response : Hey, how are you ?

Client Request : Can i take you out for coffee ?

Serever Response : Yeah,I would love that.

Client Request : When are you free ?

Serever Response : Weekend probabily

**# UDPChatClient.java :-**

package Assignment3.Question5;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPChatReceiver

{

public static void main(String[] args)

{

try

{

DatagramSocket datagramSocket = new DatagramSocket(4321);

DatagramPacket datagramPacket = null;

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

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

String message = "";

while(!message.equalsIgnoreCase("END"))

{

// Write To Client

System.out.print("Client Request : ");

message = br.readLine();

datagramPacket = new DatagramPacket(message.getBytes(), message.length(),inetAddress,1234);

datagramSocket.send(datagramPacket);

if(message.equalsIgnoreCase("END")) break;

// Read From Client

byte b[] = new byte[1024];

datagramPacket = new DatagramPacket(b,b.length);

datagramSocket.receive(datagramPacket);

message = new String(datagramPacket.getData()).trim();

if(message.equalsIgnoreCase("END")) break;

System.out.println("Serever Response : "+message);

}

br.close();

datagramSocket.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Client Request : Hello

Serever Response : Hey, how are you ?

Client Request : Can i take you out for coffee ?

Serever Response : Yeah,I would love that.

Client Request : When are you free ?

Serever Response : Weekend probabily

Client Request : end

**6. Write a Java program to implement ProxyServer**

**# HandleClientRequest.java :-**

package Assignment3.Question6;

import java.io.DataInputStream;

import java.io.DataOutputStream;

class HandleClientRequest

{

DataInputStream serverInputStream;

DataOutputStream serverOutputStream;

HandleClientRequest(DataInputStream serverInputStream, DataOutputStream serverOutputStream)

{

this.serverInputStream = serverInputStream;

this.serverOutputStream = serverOutputStream;

}

synchronized String writeToServer(String req)

{

String data = "";

try

{

this.serverOutputStream.writeUTF(req);

System.out.println("Request sent for file: " + req);

data = this.serverInputStream.readUTF();

return data;

}

catch(Exception e)

{

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

System.out.println(e);

data = "Some Error Occurred";

}

return data;

}

}

**# ClientRequest.java :-**

package Assignment3.Question6;

import java.net.\*;

import java.io.\*;

class ClientRequest extends Thread

{

Socket client;

DataInputStream clientInputStream;

DataOutputStream clientOutputStream;

HandleClientRequest handleClientRequest;

ClientRequest(Socket client, HandleClientRequest handleClientRequest)

{

try

{

this.client = client;

this.handleClientRequest = handleClientRequest;

this.clientInputStream = new DataInputStream(client.getInputStream());

this.clientOutputStream = new DataOutputStream(client.getOutputStream());

}

catch(Exception e)

{

System.out.println("Error in processing client request ...");

System.out.println(e);

}

}

public void run()

{

try

{

String req = "";

String data = "";

while(true)

{

req = clientInputStream.readUTF();

if(!req.equals("exit"))

{

data = this.handleClientRequest.writeToServer(req);

}

else

{

data = "Disconnecting the client ...";

}

clientOutputStream.writeUTF(data);

}

}

catch(Exception e)

{

System.out.println("Error in processing client request ...");

System.out.println(e);

}

finally

{

try

{

this.clientInputStream.close();

this.clientOutputStream.close();

this.client.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

}

**# ProxyServer.java :-**

package Assignment3.Question6;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.\*;

class ProxyServer

{

public static void main(String args[])

{

ServerSocket serverSocket = null;

Socket server = null;

Socket client = null;

DataInputStream serverInputStream = null;

DataOutputStream serverOutputStream = null;

HandleClientRequest handleClientRequest = null;

try

{

serverSocket = new ServerSocket(8081);

server = new Socket("localhost", 8082);

serverInputStream = new DataInputStream(server.getInputStream());

serverOutputStream = new DataOutputStream(server.getOutputStream());

handleClientRequest = new HandleClientRequest(serverInputStream, serverOutputStream);

while(true)

{

client = serverSocket.accept();

ClientRequest newClient = new ClientRequest(client, handleClientRequest);

newClient.start();

System.out.println("New Client Connected ...");

}

}

catch(Exception e)

{

System.out.println("Error on Proxy Server ...");

System.out.println(e);

}

finally

{

try

{

serverOutputStream.close();

serverInputStream.close();

server.close();

serverSocket.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

}

**# Output :-**

New Client Connected ...

Request sent for file: README.md

Request sent for file: LICENSE

Error in processing client request ...

java.io.EOFException

**# Server.java :-**

package Assignment3.Question6;

import java.net.\*;

import java.io.\*;

class Server

{

public static void main(String args[])

{

ServerSocket serverSocket = null;

Socket socket = null;

InputStream inputStream = null;

OutputStream outputStream = null;

DataInputStream dataInputStream = null;

DataOutputStream dataOutputStream = null;

File file = null;

FileInputStream fileInputStream = null;

String req = "";

String data = "";

byte[] arr = new byte[1024];

try

{

serverSocket = new ServerSocket(8082);

socket = serverSocket.accept();

inputStream = socket.getInputStream();

outputStream = socket.getOutputStream();

dataInputStream = new DataInputStream(inputStream);

dataOutputStream = new DataOutputStream(outputStream);

while(true)

{

try

{

req = dataInputStream.readUTF();

System.out.println("Request received for file: " + req);

file = new File(req);

if(file.exists())

{

fileInputStream = new FileInputStream(file);

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();

byte[] buffer = new byte[1024];

int bytesRead;

while ((bytesRead = fileInputStream.read(buffer)) != -1)

{

byteArrayOutputStream.write(buffer, 0, bytesRead);

}

arr = byteArrayOutputStream.toByteArray();

fileInputStream.close();

data = new String(arr);

}

else

{

data = "File Not Found ...";

}

}

catch(Exception e)

{

System.out.println("Error while processing request ...");

System.out.println(e);

data = "Internal Server Error ...";

}

finally

{

dataOutputStream.writeUTF(data);

}

}

}

catch(Exception e)

{

System.out.println("Error on Server ...");

System.out.println(e);

}

finally

{

try

{

dataOutputStream.close();

dataInputStream.close();

outputStream.close();

inputStream.close();

socket.close();

serverSocket.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

}

**# Output :-**

Request received for file: README.md

Request received for file: LICENSE

**# Client.java :-**

package Assignment3.Question6;

import java.net.\*;

import java.util.\*;

import java.io.\*;

class Client

{

public static void main(String args[])

{

try

{

Scanner sc = new Scanner(System.in);

Socket socket = new Socket("localhost", 8081);

DataInputStream dataInputStream = new DataInputStream(socket.getInputStream());

DataOutputStream dataOutputStream = new DataOutputStream(socket.getOutputStream());

String str = "";

String data = "";

do

{

System.out.print("Enter the file name: ");

str = sc.next();

dataOutputStream.writeUTF(str);

System.out.println("Request sent ...");

data = dataInputStream.readUTF();

System.out.println("Server Response: ");

System.out.println(data);

System.out.println();

}

while(!str.equals("exit"));

dataOutputStream.close();

dataInputStream.close();

socket.close();

sc.close();

}

catch(Exception e)

{

System.out.println("Error on Client ...");

System.out.println(e);

}

}

}

**# Output :-**

Enter the file name: README.md

Request sent ...

Server Response:

# Advance\_java\_Assignments

Enter the file name: LICENSE

Request sent ...

Server Response:

MIT License

Copyright (c) 2024 Khushi 💜

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

Enter the file name: exit

Request sent ...

Server Response:

Disconnecting the client ...