EXP NO: 7 **FULL DUPLEX CHAT USING TCP/IP**

AIM:

To implement a chat server and client in java using TCP sockets.

DESCRIPTION:

TCP Clients send requests to the server and the server will receive the request and response with acknowledgement. Every time the client communicates with the server and receives a response from it.

ALGORITHM:

Server

1. Create a server socket and bind it to the port.
2. Listen for new connections and when a connection arrives, accept it.
3. Read Client's message and display it
4. Get a message from user and send it to client
5. Repeat steps 3-4 until the client sends "exit"
6. Close all streams
7. Close the server and client socket
8. Stop

Client

1. Create a client socket and connect it to the server's port number
2. Get a message from user and send it to server
3. Read server's response and display it
4. Repeat steps 2-3 until chat is terminated with "exit" message
5. Close all input/output streams
6. Close the client socket
7. Stop

Server

```
package com;

import java.io.IOException;
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Server{
	static ServerSocket serverSocket;
	public static void main(String[] args) {
		try {
```

```
                        serverSocket = new ServerSocket(1515);
                        while(true) {
                                Socket accept_client = serverSocket.accept();
                                new Thread(new ServerIn(accept_client)).start();
                                new Thread(new ServerOut(accept_client)).start();
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                        try {
                                serverSocket.close();
                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }
        }
}
//Accepted threads
class ServerIn implements Runnable{
        Socket socket;

        ServerIn(Socket socket){
                this.socket = socket;
        }
        @Override
        public void run() {
                try {
                        InputStream in = socket.getInputStream();
                        while(true) {
                                byte infile[] = new byte[1024];
                                int size  = in.read(infile);
                                String string = new String(infile,0,size);
                                if(!string.equals("") && !string.equals("\n"))
System.out.println("message from client: "+ string);
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                        try {
                                socket.close();
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                }
        }
}
//Thread to send
class ServerOut implements Runnable{
        Socket socket;
        Scanner reader = new Scanner(System.in);
        ServerOut(Socket socket){
```

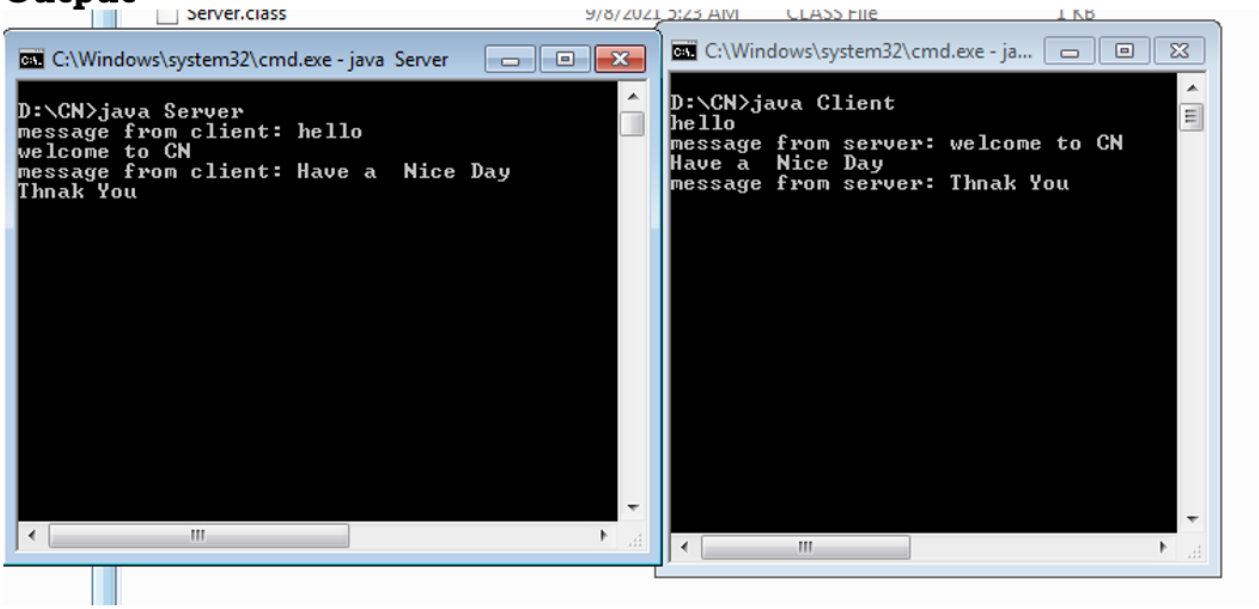
```
		this.socket = socket;
	}
	public void run() {
		try {
			OutputStreamWriter out = new
OutputStreamWriter(socket.getOutputStream());
			while(true) {
				String string  = reader.nextLine();
				out.write(string);
				out.flush();
			}
		} catch (IOException e) {
			e.printStackTrace();
			try {
				socket.close();
				reader.close();
			} catch (IOException e1) {
				e1.printStackTrace();
			}
		}
	}
}
```

Client

```
package com;

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

public class Client{
	public static void main(String[] args) {
		try {
			Socket client = new Socket("127.0.0.1",1515);
			new Thread(new ClientIn(client)).start();
			new Thread(new ClientOut(client)).start();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
//Accepted threads
class ClientIn implements Runnable{
	Socket socket;
	ClientIn(Socket socket){
		this.socket = socket;
	}
```

```
	@Override
	public void run() {
		try {
			InputStream in = socket.getInputStream();
			while(true) {
				byte infile[] = new byte[1024];
				int size = in.read(infile);
				String string = new String(infile,0,size);
				if(!string.equals("") && !string.equals("\n"))
System.out.println("message from server: "+ string);
			}
		} catch (IOException e) {
			e.printStackTrace();
			try {
				socket.close();
			} catch (IOException e1) {
				e1.printStackTrace();
			}
		}
	}

}
//Thread to send
class ClientOut implements Runnable{
	Socket socket;
	Scanner reader = new Scanner(System.in);
	ClientOut(Socket socket){
		this.socket = socket;
	}
	public void run() {
		try {
			OutputStreamWriter out = new
OutputStreamWriter(socket.getOutputStream());
			while(true) {
				String string  = reader.nextLine();
				out.write(string);
				out.flush();
			}
		} catch (IOException e) {
			e.printStackTrace();
			try {
				socket.close();
				reader.close();
			} catch (IOException e1) {
				e1.printStackTrace();
			}
		}
	}
}
```

Output



Result: Thus the Chat Using TCP/IP has been executed using Java programming