

## Experiment No. 8

Aim: To implement chat application using Socket programming.

Requirement: Windows/MAC/Linux OS in P.C and JAVA/Python programming Language in OS.

Theory:

A socket is one endpoint of a two way communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication take place.

Like 'Pipe' is used to create pipes and sockets is created using 'socket' system call. The socket provides bidirectional FIFO Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.

Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.

Types of Sockets:

There are two types of Sockets: the datagram socket and the stream socket.

Datagram Socket:

This is a type of network which has connection less point for sending and receiving packets. It is similar to mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).

Stream Socket:

In Computer operating system, a stream socket is type of interprocess communications socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well-defined mechanisms for creating and destroying connections and for detecting errors. It is similar to phone. A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.

Function Call	Description
create()	To create a socket
bind()	It's a socket identification like a telephone number to contact
listen()	Ready to receive a connection
connect()	Ready to act as a sender
accept()	Confirmation, it is like accepting to receive a call from a sender
write()	To send data
read()	To receive data
close()	To close a connection

Chat Application Code:Server Side:

```

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

public class Server{
    static HashMap<Integer,ClientHandler> acc = new HashMap<Integer,ClientHandler>();
    static int count = 0;
    public static void main(String args[]) throws IOException {
        ServerSocket ss = new ServerSocket(200);
        while(true){
            Socket s = ss.accept();
            ClientHandler ct = new ClientHandler("client "+count, new
DataInputStream(s.getInputStream()), new DataOutputStream(s.getOutputStream()),s);
            Thread t = new Thread(ct);
            acc.put(count, ct);
            count++;
            t.start();
        }
    }
}

class ClientHandler implements Runnable{
    private final String name;
    final DataInputStream dis;
    final DataOutputStream dos;
    public final Socket s;
    boolean loggedin;
    public ClientHandler(String name,DataInputStream dis,DataOutputStream dos,Socket s){
        this.name = name;
        this.dis = dis;
        this.dos = dos;
        this.s = s;
        this.loggedin = true;
    }
    @Override
    public void run(){
        while(true){
            String receive;
            try{
                receive = this.dis.readUTF();
                if(receive.equals("logout")){
                    this.loggedin = false;
                    this.s.close();
                    break;
                }
            }
            if(receive.contains("#")){
                StringTokenizer st = new StringTokenizer(receive,"#");
                String msg = st.nextToken();
            }
        }
    }
}

```

```

        String name = st.nextToken();
        ClientHandler cc =
Server.acc.get(Character.getNumericValue(name.charAt(name.length()-1)));
        if(cc == null || !cc.loggedin){
            this.dos.writeUTF("Client not found");
        }
        else{
            cc.dos.writeUTF(this.name+": "+msg);
        }
    }
    else{
        this.dos.writeUTF("Wrong Input");
    }
} catch (IOException e){
    e.printStackTrace();
}
}
}
}

```

#### Client Side:

```

package chatapp;
import java.util.*;
import java.io.*;
import java.net.*;
public class Client {
    public static void main(String[] args) throws IOException{
        InetAddress ip = InetAddress.getByName("localhost");
        Socket s= new Socket(ip,200);
        Thread sm = new Thread(new SendMessage(s, new
DataOutputStream(s.getOutputStream())));
        Thread rm = new Thread(new ReadMessage(s, new
DataInputStream(s.getInputStream())));
        sm.start();
        rm.start();
    }
}
class SendMessage implements Runnable{
    Scanner sc = new Scanner(System.in);
    final private Socket s;
    DataOutputStream dos;
    public SendMessage(Socket s, DataOutputStream dos){
        this.s = s;
        this.dos = dos;
    }
    @Override
    public void run(){
        while(true){
            try{
                this.dos.writeUTF(sc.nextLine());
            }

```

```

        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
}
}
class ReadMessage implements Runnable{
    final private Socket s;
    DataInputStream dis;
    public ReadMessage(Socket s, DataInputStream dis){
        this.s = s;
        this.dis = dis;
    }
    @Override
    public void run(){
        while(true){
            try{
                System.out.println(this.dis.readUTF());
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
}

```

### Output:

### Server:

```

C:\Users\adnan\OneDrive\Desktop\College\sem5\CN\Practical 8\Chat Application>java chatapp/Server

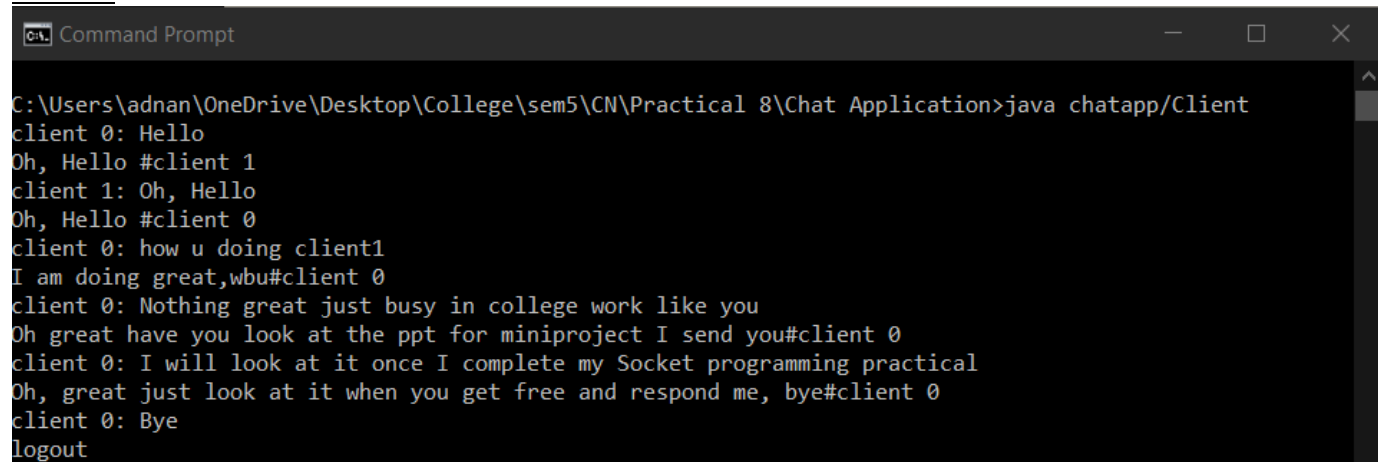
```

### Client 0:

```

C:\Users\adnan\OneDrive\Desktop\College\sem5\CN\Practical 8\Chat Application>java chatapp/Client
Hello#Client 1
client 1: Oh, Hello
how u doing client1#client 1
client 1: I am doing great,wbu
Nothing great just busy in college work like you#client 1
client 1: Oh great have you look at the ppt for miniproject I send you
I will look at it once I complete my Socket programming practical#client 3
Client not found
I will look at it once I complete my Socket programming practical#client 1
client 1: Oh, great just look at it when you get free and respond me, bye
Bye#client 1
logout

```

Client 1:

```
C:\Users\adnan\OneDrive\Desktop\College\sem5\CN\Practical 8\Chat Application>java chatapp/Client
client 0: Hello
Oh, Hello #client 1
client 1: Oh, Hello
Oh, Hello #client 0
client 0: how u doing client1
I am doing great, wbu #client 0
client 0: Nothing great just busy in college work like you
Oh great have you look at the ppt for miniproject I send you #client 0
client 0: I will look at it once I complete my Socket programming practical
Oh, great just look at it when you get free and respond me, bye #client 0
client 0: Bye
logout
```

Conclusion: We have successfully implemented Chat Application with the help of Socket Programming using TCP or UDP in JAVA.