

DSCC**Practical No: 01****1. Develop a JAVA program for multi-client chat server.**

```
ChatServer.java package multiclient; import  
java.net.ServerSocket; import java.net.Socket;
```

```
public class ChatServer {  
    int port;  
    ServerSocket serverSocket;  
    Socket socket;  
  
    public ChatServer(int port) {  
        super();  
        this.port = port;  
    }  
  
    public void listen() {  
        try {  
            serverSocket = new ServerSocket(port);  
            System.out.println("Listening on ip:" +  
serverSocket.getInetAddress().getHostAddress() + " and port:" + port);  
            while(true)  
            {  
                socket = serverSocket.accept();  
                System.out.println("Client Accepted " + socket);  
                ServRequest sr=new ServRequest(socket,this);  
                sr.start();  
            }  
  
        } catch (Exception e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

```
    }  
}  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        ChatServer cs = new ChatServer(5000);  
        cs.listen();  
    }  
  
}
```

ChatClient.java package

```
multiclient; import  
java.io.BufferedReader; import  
java.io.DataInputStream; import  
java.io.DataOutputStream; import  
java.io.InputStreamReader; import  
java.net.InetAddress; import  
java.net.Socket;
```

```
public class ChatClient {  
    Socket socket;  
    int port;  
  
    public ChatClient(int port) {  
        super();  
        this.port = port;  
    }  
  
    public void request() {  
        try {
```

```
        InetAddress host = InetAddress.getLocalHost();

        socket = new Socket(host.getHostName(), port);

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

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

        System.out.println("Connected");

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

        String line = "";

        while(!line.equals("bye")) {

            line = keyRead.readLine();

            dos.writeUTF(line);

dos.flush();

            line = dis.readUTF();

            System.out.println("Server reply - " + line);

        }

keyRead.close();

dos.close();

socket.close();

    }

    catch(Exception e) {

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

    }

}

public static void main(String[] ar) {

    ChatClient cc = new ChatClient(5000);
```

```
        cc.request();  
    }  
  
}
```

ServRequest.java

```
package multicient; import  
java.io.BufferedReader; import  
java.io.BufferedInputStream; import  
java.io.BufferedOutputStream; import  
java.io.DataInputStream; import  
java.io.DataOutputStream; import  
java.io.InputStreamReader; import  
java.net.Socket;  
  
public class ServRequest extends Thread{  
    private Socket socket;  
    @SuppressWarnings("unused")  
    private ChatServer chatServer;  
  
    public ServRequest(Socket socket, ChatServer chatServer) {  
        this.socket=socket;  
        this.chatServer=chatServer;  
    }  
  
    public void run()  
    {  
        try {  
            DataInputStream dis = new DataInputStream(new  
BufferedReader(new  
BufferedInputStream(socket.getInputStream())));  
            DataOutputStream dos = new DataOutputStream(new  
BufferedOutputStream(socket.getOutputStream()));
```

```
        BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
        boolean done = false;

        while (!done) {
            String line = dis.readUTF();

            System.out.println(" Client Msg - "+ line + "\n");

            done = line.equals("bye");

            line = keyRead.readLine();                dos.writeUTF(line);

            dos.flush();
        }

        dis.close();
        socket.close();

    }

    catch(Exception e)
    {
        System.out.println(e.getMessage());
    }

}

}
```

Output:

```
Problems Javadoc Declaration Console X
ChatClient (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (30-Nov-2022, 9:59:42 am)
Connected
Hii
Server reply - Hii
Good Morning
Server reply - This is a Test
Bye
Server reply - Bye
```

```
Problems Javadoc Declaration Console X
ChatServer (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (30-Nov-2022, 9:59:36 am)
Listening on ip:0.0.0.0 and port:5000
Client Accepted Socket[addr=/172.16.9.83,port=60964,localport=5000]
Client Msg - Hii

Hii
Client Msg - Good Morning

This is a Test
Client Msg - Bye

Bye
```

2. Write a java program to implement mutual exclusion using Token ring algorithm.

UDPChatClient2.java package tokenring; import java.io.BufferedReader; import
java.io.InputStreamReader; import java.net.DatagramPacket; import
java.net.DatagramSocket; import java.net.InetAddress;

```
public class UDPChatClient {  
    DatagramSocket udpClientSocket;  
    int port;  
  
    public UDPChatClient(int port) {  
        this.port = port;  
    }  
}
```

```
public void sendReq() {  
    InetAddress serverAddress;  
    String in;  
  
    try {  
        udpClientSocket = new DatagramSocket();           InetAddress  
        host = InetAddress.getLocalHost();           serverAddress =  
        InetAddress.getByName(host.getHostName());  
  
        BufferedReader keyRead = new BufferedReader(new  
        InputStreamReader(System.in));  
        System.out.println("UDP Client-1 started at " + InetAddress.getLocalHost());  
  
        while (true) {  
            System.out.println("Enter message for server: ");           in =  
            keyRead.readLine();  
            DatagramPacket sndPacket = new DatagramPacket(in.getBytes(),  
            in.getBytes().length,           serverAddress,           port);  
            udpClientSocket.send(sndPacket);  
  
            if(in.equalsIgnoreCase("bye"))  
                break;  
            byte[]  
            buf = new byte[1024];  
            DatagramPacket recPacket = new DatagramPacket(buf, buf.length);  
            udpClientSocket.receive(recPacket);  
            String msg = new String(recPacket.getData()).trim();  
  
            System.out.println("Message from " +  
            recPacket.getAddress().getHostAddress() + ": " + msg);  
        }  
    }  
}
```

```
    }  
    catch(Exception e) {  
        System.out.println(e.getMessage());  
    }  
    finally {  
        udpClientSocket.close();  
    }  
}  
  
public static void main(String[] args) {  
    UDPChatClient sender = new UDPChatClient(5000);  
    sender.sendReq();  
  
}  
  
}
```

UDPChatSrv.java package

```
tokenring; import  
java.io.BufferedReader; import  
java.io.InputStreamReader; import  
java.net.DatagramPacket; import  
java.net.DatagramSocket; import  
java.net.InetAddress;  
  
public class UDPChatSrv {  
    public DatagramSocket udpSrvSocket; public int  
    port;  
    String in;
```



```
public UDPChatSrv(int port) {
    this.port = port;
}

private void listen() {
    try {
        udpSrvSocket = new DatagramSocket(port);
        BufferedReader keyRead = new
        BufferedReader(new InputStreamReader(System.in));
        String msg;

        int [] clientPortA = new int[2];
        InetAddress clientAddress;          int
        clientPort, clientCnt = 0;          int tokenTo
        = -1, currentClient = -1;

        DatagramPacket recPacket, sndPacket;

        System.out.println("Server started at " + InetAddress.getLocalHost());

        while (true) {
            byte[] buf = new byte[1024];
            //System.out.println("while @server");          recPacket
            = new DatagramPacket(buf, buf.length);

            // blocks until a packet is received
            udpSrvSocket.receive(recPacket);          msg = new
            String(recPacket.getData()).trim();
            clientAddress = recPacket.getAddress();
            clientPort = recPacket.getPort();

            boolean clientPortPresent = false;
```

```
int i;

for(i = 0; i < clientPortA.length; i++) {
if(clientPortA[i] == clientPort) {
    clientPortPresent = true;
currentClient = i;      break;
    }
    }

    if(clientPortPresent == false) {
clientPortA[clientCnt] = clientPort;
currentClient = clientCnt;      clientCnt++;
    }

    //System.out.println("Message from client " + currentClient + ": " + msg);

    if(tokenTo == -1 && clientCnt == 1) {
        tokenTo = 0; //Assign token to 1st client in d list
        System.out.println("send Message :- Token assigned to client " + currentClient);
        in = "Token assigned";

        sndPacket = new DatagramPacket(in.getBytes(), in.getBytes().length, clientAddress,
clientPortA[currentClient]);      udpSrvSocket.send(sndPacket);
    }

    //1. token is with 0th client in d list => clientCnt=1
    //2. client send message token => either he wants it or return it.
    //3. if he wants check who has d token n reply accordingly
    //3.1 if token is with 1 n 2 wants then deny
    //3.2 token message arrived at server, current client n tokenTo are same
        //3.2 so remove token from current n assign it to next
    //4. if he returns it then assign it next in list
```

```
        if(msg.contains("token")) {
if(tokenTo == currentClient) {
            if(clientPortA.length == tokenTo)
                tokenTo = 0;
            else
                tokenTo++;

            System.out.println("send Message :- Token assigned to client " + currentClient);
            //in = keyRead.readLine();
in = "Token assigned";

            sndPacket = new DatagramPacket(in.getBytes(), in.getBytes().length,
clientAddress, clientPortA[tokenTo]);      udpSrvSocket.send(sndPacket);
        }
        else {
            System.out.println("send Message :- ");
            //in = keyRead.readLine();      in = "Token is with Client
- " + tokenTo + ". Wait for your turn.";

            sndPacket = new DatagramPacket(in.getBytes(), in.getBytes().length,
clientAddress, clientPortA[currentClient]);      udpSrvSocket.send(sndPacket);
        }
    }
    else {
        if(currentClient == tokenTo) {
            System.out.println("send Message :- ");
            in = keyRead.readLine();      //in =
"Token assigned";

            sndPacket = new DatagramPacket(in.getBytes(), in.getBytes().length,
clientAddress, clientPortA[currentClient]);

            udpSrvSocket.send(sndPacket);
        }
        else{
```

```
        System.out.println("send Message :- ");

        //in = keyRead.readLine();          in = "Token is with Client
- " + tokenTo + ". Wait for your turn.";

        sndPacket = new DatagramPacket(in.getBytes(), in.getBytes().length,
clientAddress, clientPortA[currentClient]);      udpSrvSocket.send(sndPacket);
    }

}

/*if(msg.equalsIgnoreCase("bye"))
    clientCnt--;
*/
}

}

catch(Exception e) {
    System.out.println(e.getMessage());
}

finally {
udpSrvSocket.close();
}

}

public static void main(String[] args) {
    UDPChatSrv client = new UDPChatSrv(5000);
    client.listen();
}

}
```

Output:

```
Problems Javadoc Declaration Console ×
<terminated> UDPChatClient (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (30-Nov-2022, 10:16:33 a
UDP Client-1 started at Hajisab07/172.16.9.83
Enter message for server:
Hii
Message from 172.16.9.83: Token assigned
Enter message for server:
Good Morining
Message from 172.16.9.83: Hii
Enter message for server:
Bye
```

```
Problems Javadoc Declaration Console ×
UDPChatSrv (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (30-Nov-2022, 10:16:22 am)
Server started at Hajisab07/172.16.9.83
send Message :- Token assigned to client 0
send Message :-
Hii
send Message :-
token
send Message :-
Bye
```