SERVER UDP

LIBRERIE

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
```

CREAZIONE SOCKET

```
        DatagramSocket ds = new DatagramSocket(PORT);
```

ds = Nome del socket

BUFFER PER INPUT

```
        byte[] buffer = new byte[256];
```

buffer = Nome del buffer
256 = Dimensione del buffer

RICEZIONE MESSAGGIO

```
        DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
```

dp = Nome del DatagramPacket
buffer = Nome del buffer precedentemente creato
buffer.length = Funzione rapida per calcolare la lunghezza del buffer

Dopo aver creato il nostro “contenitore” procediamo con la cattura dell’input dell’utente

```
        ds.buffer(dp);
```

Funzione che attende un input da un client. Il messaggio verrà inserito dentro al buffer stabilito nel dp. dp.getData() mostrerà il messaggio in esadecimale

MESSAGGIO IN STRINGA

```
        String msg = new String(dp.getData());
        msg = msg.substring(0, dp.getLength() - 1);
```

Nella variabile “msg” verrà inserita la stringa ricevuta in input (prima riga) e con la seconda riga di codice inserisco solamente il messaggio inviato, togliendo tutti i “caratteri vuoti” del buffer

EQUALS

```
        msg.equals("bye")
```

Funzione utilizzata per controllare il contenuto di “msg”

STRING TO INT

```
        int x = Integer.parseInt(msg);
```

Funzione rapida per trasformare una stringa in intero

CONNESSIONE

```
        nc -u localhost 8888
```

-u = Indica la modalità UDP

CLIENT UDP

LIBRERIE

```
    import java.io.IOException;
    import java.io.UnsupportedEncodingException;
    import java.net.DatagramPacket;
    import java.net.DatagramSocket;
    import java.net.InetAddress;
    import java.net.SocketException;
    import java.net.UnknownHostException;
    import java.util.Scanner;
```

CREAZIONE SOCKET

```
        DatagramSocket ds =new DatagramSocket();
```

La differenza è nel fatto che non si deve passare la porta

INDIRIZZO + PORTA DI CONNESSIONE

```
        InetAddress indirizzo = InetAddress.getByName("localhost");
        int porta = 8888;
```

DATAGRAM PACKET

```
        DatagramPacket dp =new DatagramPacket(buffer, buffer.length, indirizzo, p
```

La differenza è nel fatto che viene passato anche l'indirizzo e la porta

MESSAGGIO DA INVIARE

```
        Scanner input = new Scanner(System.in);
        System.out.println("Messaggio da inviare: ");
        String msg = in.nextLine();
```

INVIO MESSAGGIO AL SERVER

```
        byte[] buffer =new  byte[256];
        buffer = msg.getBytes("UTF-8");
        dp.setData(buffer);
        dp.setLength(buffer.length);
        ds.send(dp);
```

Dentro al "buffer" verrà inserita la stringa ricevuta in input, viene inserito dentro al DatagramPacket il contenuto e la lunghezza del messaggio e infine viene spedito al server il messaggio con la funzione "send"

SERVER TCP

LIBRERIE

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

VARIABILI GENERALI

```
BufferedReader inDalClient;
DataOutputStream outVersoClient;
```

CREAZIONE SOCKET

```
ServerSocket server = new ServerSocket(8888);
```

server = Nome socket
8888 = Porta di hosting del server

ATTESA CONNESSIONE

```
 Socket client = server.accept();
```

Attende la connessione di un client

CHIUSURA SOCKET

```
server.close();
```

Chiude la connessione al Socket

ASSOCIAZIONE I/O

```
inDalClient = new BufferedReader(new InputStreamReader (client.getInputSt
outVersoClient = new DataOutputStream(client.getOutputStream());
```

Associo all'input e all'output il client appena connesso

INPUT DA CLIENT

```
String msg = inDalClient.readLine();
```

msg = Nome della variabile
inDalClient = Buffer inizializzato precedentemente

OUTPUT AL CLIENT

```
        outVersoClient.writeBytes("MSG");
```

MSG = stringa da inviare al client
outVersoClient = Buffer inizializzato precedentemente

CHIUSURA CONNESSIONE

```
        client.close();
        inDalClient.close();
        outVersoClient.close();
```

E' necessario chiudere sia la connessione con il client, sia i 2 buffer utilizzati per l'input e l'output

ARRAY DINAMICO + POPOLAMENTO

```
        static ArrayList<String> Array = new ArrayList<String>();
```

Array dinamico di Stringhe

Per popolarlo utilizzare

```
        Array.add(client.toString());
```

client è la variabile utilizzata nel momento dell'attesa di una connessione (Socket client = server.accept();)

IMPLEMENTAZIONE WEBSERVER

```
        out.writeBytes("HTTP/1.1\nContent-Type: text/html; charset=UTF-8\n");
```

Questo deve essere inviato come PRIMO messaggio. Successivamente si utilizza la classica sintassi html

```
        out.writeBytes("<!DOCTYPE html> <html> <body> SOME TEXT </body> </html>")
```