

## Server (TCP)

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

public class SimpleServer {
    public static final int PORT = 8080;
    public static void main(String[] args) throws IOException {
        ServerSocket s = new ServerSocket(PORT);
        System.out.println("Started: " + s);
        try {
            Socket socket = s.accept();
            try {
                System.out.println("Connection accepted: "+ socket);
                BufferedReader in =
                    new BufferedReader(
                        new InputStreamReader(
                            socket.getInputStream()));
                PrintWriter out =
                    new PrintWriter(
                        new BufferedWriter(
                            new OutputStreamWriter(
                                socket.getOutputStream()),true));
                while (true) {
                    String str = in.readLine();
                    if (str.equals("END")) break;
                    System.out.println("Echoing: " + str);
                    out.println(str);
                }
            } finally {
                System.out.println("closing...");
                socket.close();
            }
        } finally {
            s.close();
        }
    }
}
```

## Client (TCP)

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

public class SimpleClient {
    public static void main(String[] args) throws IOException {
        InetAddress addr = InetAddress.getByName(null);
        System.out.println("addr = " + addr);
        Socket socket = new Socket(addr, SimpleServer.PORT);
        try {
            System.out.println("socket = " + socket);
            BufferedReader in =
                new BufferedReader(
                    new InputStreamReader(
                        socket.getInputStream()));
            PrintWriter out =
                new PrintWriter(
                    new BufferedWriter(
                        new OutputStreamWriter(
                            socket.getOutputStream()),true));

            for(int i = 0; i < 10; i++) {
                out.println("ciao " + i);
                String str = in.readLine();
                System.out.println(str);
            }
            out.println("END");
        } finally {
            System.out.println("closing...");
            socket.close();
        }
    }
}
```

## Classe Dgram

```
import java.net.*;
public class Dgram {
    static DatagramPacket toDatagram(String s,
                                     InetAddress destAddr, int destPort) {
        byte[] buf = s.getBytes();
        return new DatagramPacket(buf, buf.length, destAddr, destPort);
    }

    static String toString(DatagramPacket p) {
        return new String(p.getData(), 0, p.getLength());
    }
}
```

## Server (UDP)

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

public class DgramServer {
    static final int INPORT = 1711;

    public static void main(String[] args) {
        byte[] buf = new byte[1000];
        DatagramPacket dp = new DatagramPacket(buf, buf.length);
        DatagramSocket socket;

        try {
            socket = new DatagramSocket(INPORT);
            System.out.println("Server started");
            while(true) {
                socket.receive(dp);
                String rcvd = Dgram.toString(dp) + ", from address: " +
                    dp.getAddress() + ",
                    port: " + dp.getPort();

                System.out.println(rcvd);
                String echoString = "Echoed: " + rcvd;
                DatagramPacket echo = Dgram.toDatagram(echoString,
                    dp.getAddress(), dp.getPort());

                socket.send(echo);
            }
        } catch(SocketException e) {
            System.err.println("Can't open socket");
            System.exit(1);
        } catch(IOException e) {
            System.err.println("Communication error");
            e.printStackTrace();
        }
    }
}
```

## Client (UDP)

```
import java.net.*;
import java.io.*;
public class DgramClient {
    public static void main(String[] args) {
        DatagramSocket s;
        InetAddress hostAddress;
        byte[] buf = new byte[1000];
        DatagramPacket dp = new DatagramPacket(buf, buf.length);

        try {
            s = new DatagramSocket();
            hostAddress = InetAddress.getByName("localhost");
            System.out.println("Client starting");
            for(int i = 0; i < 5; i++) {
                String outMessage = "Client #" + i, message #" + i;
                s.send(Dgram.toDatagram(outMessage,
                    hostAddress,
                    DgramServer.INPORT));

                s.receive(dp);
                String rcvd = "Client #" + i, rcvd from " +
                    dp.getAddress() + ", " +
                    dp.getPort() + ": " + Dgram.toString(dp);

                System.out.println(rcvd);
            }
        } catch(UnknownHostException e) {
            System.err.println("Cannot find host");
            System.exit(1);
        } catch(SocketException e) {
            System.err.println("Can't open socket");
            e.printStackTrace();
            System.exit(1);
        } catch(IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
} // main
} // class
```

## Prods & Cons: messaggi

```
import java.io.*;

class Message implements Serializable {
    private int type;
    private String body;

    Message(int aType, String aBody) {
        type = aType;
        body = aBody;
    }

    public int getType() {
        return type;
    }

    public String getBody() {
        return body;
    }

    public String toString() {
        return "Message[type=" + type + ", body=" + body + "];"
    }
}
```

## Prods&Cons: Buffer

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

public class Buffer {
    public static final int PORT = 8080;
    public static void main(String[] args) throws IOException,
        ClassNotFoundException {
        LinkedList<Socket> sockqueue = new LinkedList<Socket>();
        LinkedList<Message> msgqueue = new LinkedList<Message>();
        ServerSocket servsock = new ServerSocket(PORT);
        try {
            while(true) {
                Socket sock = servsock.accept();
                try {
                    ObjectInputStream in = new ObjectInputStream(
                        sock.getInputStream());
                    Message m = (Message)in.readObject();
                    if (m.getType() == 0) // dati
                        if (sockqueue.isEmpty()) // no waiting cons
                            msgqueue.addLast(m);
                        else { // waiting cons available
                            Socket appsock = (Socket)sockqueue.removeFirst();
                            try {
                                ObjectOutputStream o = new
                                    ObjectOutputStream(
                                        appsock.getOutputStream());
```

```
                                o.writeObject(m);
                            } finally {
                                appsock.close();
                            }
                        }
                    else // pronto
                        if (msgqueue.isEmpty()) // no available data
                            sockqueue.addLast(sock);
                        else { // available data
                            ObjectOutputStream out = new ObjectOutputStream(
                                sock.getOutputStream());
                            Message msg = (Message)msgqueue.removeFirst();
                            out.writeObject(msg);
                            sock.close();
                        }
                } catch(IOException e) {
                    sock.close();
                }
            }
        } finally {
            servsock.close();
        }
    }
}
```

## Prods & Cons: Producer

```
import java.net.*;
import java.io.*;
public class Producer {
    public static void main(String[] args) throws IOException,
        ClassNotFoundException {
        InetAddress addr = InetAddress.getByName(null);
        Socket socket = new Socket(addr, Buffer.PORT);
        System.out.println("Connected to " + socket);
        try {
            ObjectOutputStream out = new ObjectOutputStream(
                socket.getOutputStream());
            System.out.print("Sending data message... ");
            Message data = new Message(0, "pippo");
            out.writeObject(data);
            System.out.println("done");
        } finally {
            socket.close();
        }
    }
}
```

## Prods & Cons: Consumer

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

public class Consumer {
    public static void main(String[] args) throws IOException,
        ClassNotFoundException {
        InetAddress addr = InetAddress.getByName(null);
        Socket socket = new Socket(addr, Buffer.PORT);
        System.out.println("Connected to " + socket);
        try {
            ObjectOutputStream out = new ObjectOutputStream(
                socket.getOutputStream());

            Message pronto = new Message(1, "ready");
            out.writeObject(pronto);
            ObjectInputStream in = new ObjectInputStream(
                socket.getInputStream());

            Message dati = (Message)in.readObject();
            System.out.println("received "+ dati);
        } finally {
            socket.close();
        }
    }
}
```