# TP1: Synchronous Java Socket (Blocking Communication)

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

public class TCPClient {

public static void main(String argv[]) throws Exception
{
String sentence;
String modifiedSentence;
//Crée flot d'entrée
BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
//Crée socket client, se connecte au serveur
Socket clientSocket = new Socket("localhost", 3456);
//Crée flot de sortie lié au socket
DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
//Crée flot d'entrée lié au socket
BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
sentence = inFromUser.readLine ();
//envoie ligne au serveur
outToServer.writeBytes (sentence + '\n');
//lit ligne du serveur
modifiedSentence = inFromServer.readLine ();
System.out.println (" Recu du serveur : " + modifiedSentence);
clientSocket.close ();
}
}
```

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

public class TCPServer {

public static void main(String argv[]) throws Exception
{
String clientSentence;
String capitalizedSentence;
//Crée socket de démarrage au port 3456
ServerSocket welcomeSocket = new ServerSocket(3456);
//Attend le contact du client, sur le socket de démarrage
Socket connectionSocket = welcomeSocket.accept();
while(true) {
//Crée flot d'entrée lié au socket
BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
//Crée flot de sortie lié au socket
DataOutputStreamout ToClient = new
DataOutputStream(connectionSocket.getOutputStream());
//Lit ligne d'entrée du socket
clientSentence = inFromClient.readLine ();
capitalizedSentence = clientSentence.toUpperCase () + '\n';
//Écrit ligne de sortie dans le socket
outToClient.writeBytes (capitalizedSentence);
}//Fin de la boucle while, boucle avec retour et attend une autre
connexion avec un client
}
}
```

**Questions**

1. Analyser les situations suivantes.
    a. Lancer le client avant le serveur.
    b. Lancer le serveur avant un client.
    c. Lancer un serveur et plusieurs clients.
2. Modifier le code de serveur pour qu'il serve plusieurs clients
3. Version UDP
    a. Avec les datagrammes UDP, modifier le code Java et répondez à la question 1.
    b. Que concluez-vous ?

Help UDP Datagram Client Side

```
DatagramSocketclientSocket = new DatagramSocket ();
InetAddressIPAddress = InetAddress.getByName ("localhost");

byte [] sendData = new byte[1024];
byte [] receiveData = new byte[1024];

DatagramPacketsendPacket = new DatagramPacket (sendData, sendData.length, IPAddress, 9876);
clientSocket.send (sendPacket);
DatagramPacketreceivePacket = new DatagramPacket (receiveData, receiveData.length);
clientSocket.receive (receivePacket);
String modifiedSentence = new String (receivePacket.getData ());
clientSocket.close ();
```

4. Concevez et implémentez une version avec les threads de Java.

**Référence** : Ajay D. Kshemkalyani, MukeshSinghal, **Distributed Computing Principles Algorithms and Systems** CAMBRIDGE UNIVERSITY PRESS, 2008 .1.6 Primitives for distributed communication (pages 14-19)