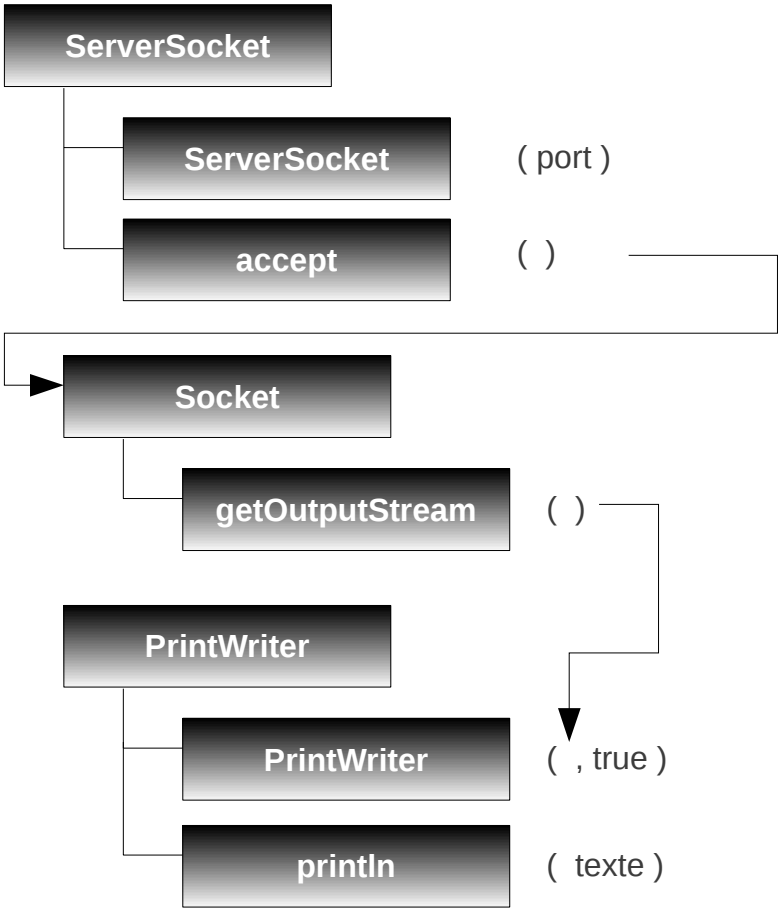
Client-Server

Course Goals

- Server initialisations
- Client initialisations
- Discussions

Server

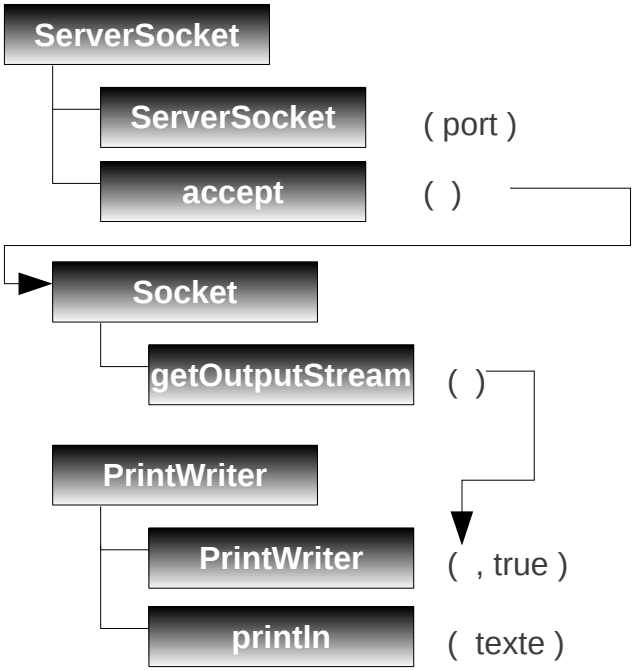
First server Write method



- We instantiate a ServerSocket with a port.
- We listen to this Instantiation with accept() method. It returns a Socket as soon as a connection occurs.
- We instantiate a PrintWriter with this socket's getOutputStream() (2nd parameter is to auto-flush streams)
- We send messages with PrintWriter.println() method

First server – Write method

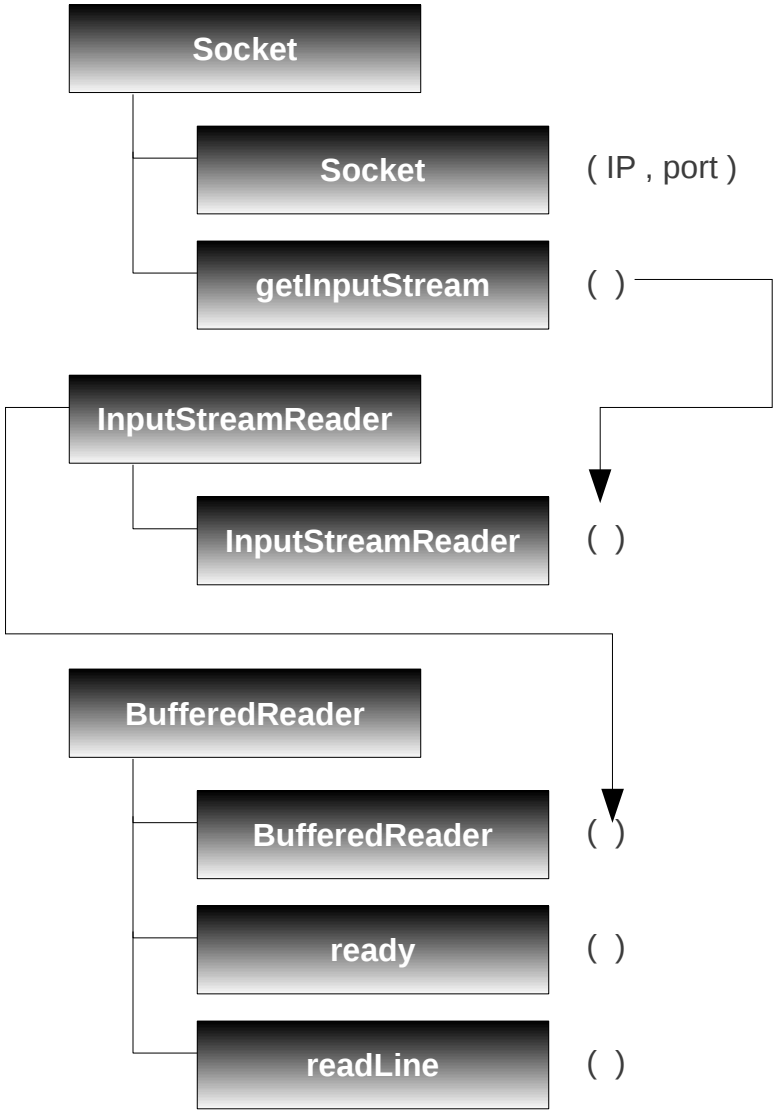
- We instantiate a ServerSocket with a port.
- We listen to this Instantiation with accept() method. It returns a Socket as soon as a connection occurs.
- We instantiate a PrintWriter with this socket's getOutputStream() (2nd parameter is to auto-flush streams)
- We send messages with PrintWriter.println() method




```
package com.ingesup.java.cliserv;
import java.io.*;
import java.net.*;

public class Serveur1
{
	public Serveur1(int numPort,String chaineAEnvoyer)
	{
		try {
			ServerSocket srvr = new ServerSocket(numPort);
			System.out.println("Serveur à l'écoute...");
			Socket skt = srvr.accept();
			System.out.println("Un client est connecté !");
			PrintWriter out = new PrintWriter(skt.getOutputStream(),true);
			System.out.println("Envoi de la chaine '" + chaineAEnvoyer+ "'.");
			out.println(chaineAEnvoyer);
			out.close();
			skt.close();
			srvr.close();
		}
		catch(Exception e) {
		System.out.println("Const. Serveur1 -> Problème : "+e.getMessage());
		}
	}

	public static void main(String args[])
	{
		if(args.length!=2)
			new Serveur1(1234,"Bonjour du serveur !");
		else
			new Serveur1(Integer.parseInt(args[0]),args[1]);
	}
}
```

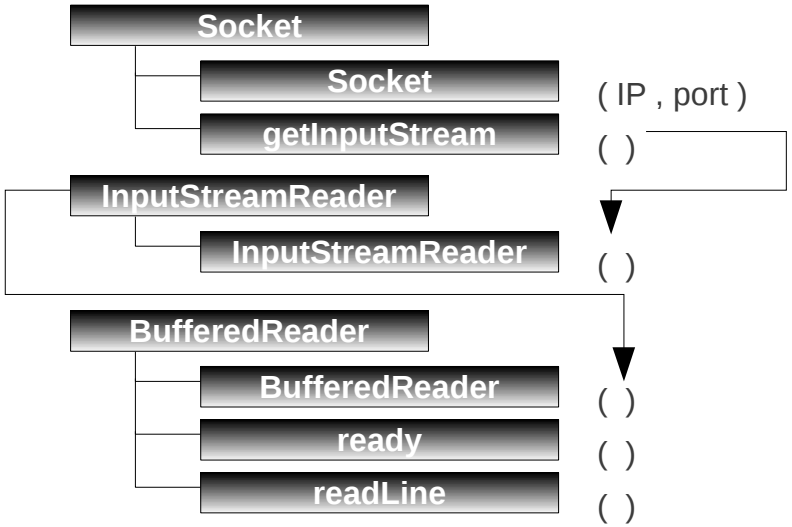
Client

First client – Read method



- We instantiate a Socket with an IP and a port.
- We instantiate an InputStreamReader with this socket's getInputStream()
- We instantiate a BufferedReader with previous instance.
- We loop while BufferedReader instance is not ready
- We retrieve stream text with readLine() method

First client – Read method



- We instantiate a Socket with an IP and a port.
- We instantiate an InputStreamReader with this socket's getInputStream()
- We instantiate a BufferedReader with previous instance.
- We loop while BufferedReader instance is not ready
- We retrieve stream text with readLine() method

```
package com.ingesup.java.cliserv;

import java.io.*;
import java.net.Socket;

public class Client1
{
	public Client1(String ipServeur,int port) {
		try {
			Socket skt = new Socket(ipServeur,port);
			BufferedReader in = new BufferedReader(new   InputStreamReader(skt.getInputStream()));
			System.out.println("Chaine reçue du serveur : '"+in.readLine()+"'.");
			in.close();
		}
		catch(Exception e) { System.out.println("Problème client : "+e.getMessage());   }
	}

	public static void main(String args[]) {
		if(args.length!=2)
			new Client1("localhost", 1234);
		else
			new Client1(args[0], Integer.parseInt(args[1]));
	}
}
```

First job

First job

- Create a server :
  - private attribute : a static String array with some stuff
  - main :
    - listens to client demand
    - an infinite loop
    - transform received String into an integer
    - if integer is out of array's bounds, send “Tchao” and break
    - if not, send array's corresponding String

First job

- Create a client :
  - main :
    - a do... while loop until received String is “Tchao”
    - accept from keyboard an integer
    - send this integer to socket
    - read socket answer and print it

First job result

```
Tableau ← tableau de String avec des valeurs
Instancier socketServeur
SocketComm ← écoute
In ← flux de lecture du socketComm
Out ← flux d'écriture du socketComm
Tant que vrai
       Nb ← chaine reçue de "in" transfo en entier
       Si nb>=taille du tableau
              Envoyer "Tchao" sur "out"
              Casser la boucle
       Sinon
              Envoyer tableau[nb] sur "out"
       Fin si
Fin tant que
```

```
SocketComm ← appel serveur
In ← flux de lecture du socketComm
Out ← flux d'écriture du socketComm
Clavier ← flux de lecture du clavier
Tant que vrai
       Nb ← entier reçu de "clavier"
       Envoyer Nb sur "out"
       Reponse ← chaine reçue de "in"
       Afficher reponse
       Si reponse = "Tchao"
              Casser la boucle
       Fin si
Fin tant que
```

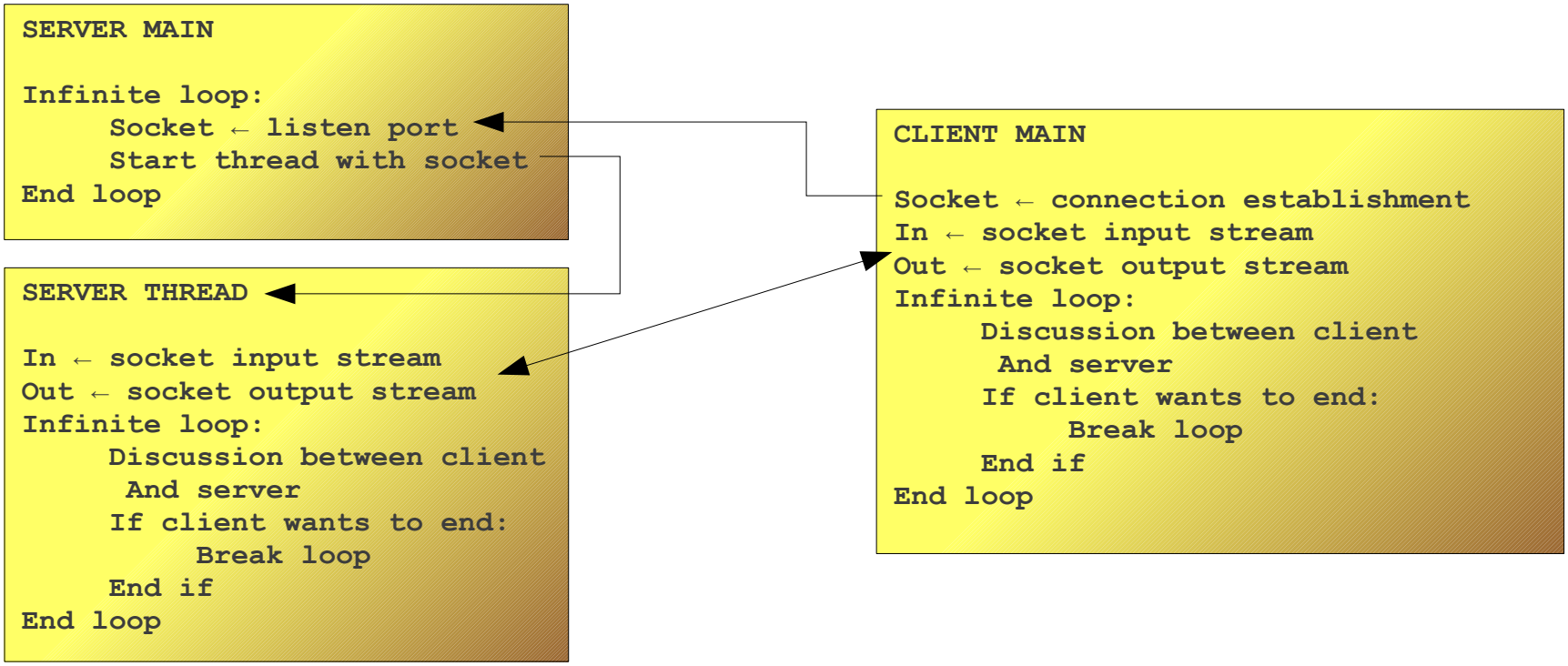
```
$ java Serveur
Le serveur est à l'écoute...
Le serveur a une connexion !
Le serveur a reçu '0'
Il renvoie 'Va faire le ménage'


Le serveur a reçu '1'
Il renvoie 'Va faire les courses'


Le serveur a reçu '2'
Il renvoie 'Va préparer à manger'


Le serveur a reçu '3'
Indice en dehors du tableau. J'arrête !
```

```
$ java Client
Entrez un nombre :
0

Le client a reçu 'Va faire le ménage'
Entrez un nombre :
1

Le client a reçu 'Va faire les courses'
Entrez un nombre :
2

Le client a reçu 'Va préparer à manger'
Entrez un nombre :
3


Le client a reçu 'Tchao'
```

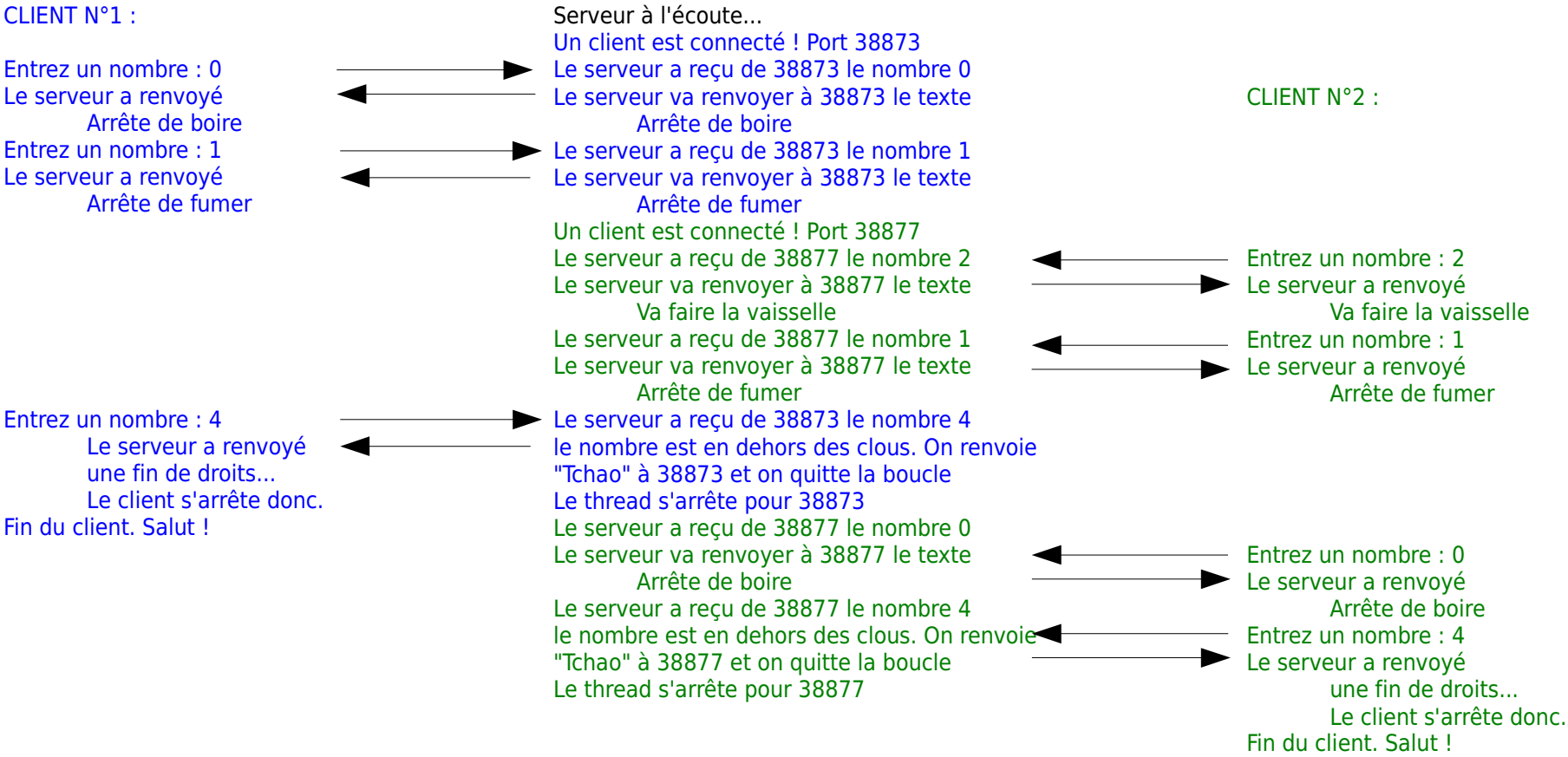
Threaded Server

Threaded server

- Now, when the server receives a connection (i.e. a socket from listen() method), it starts a runnable thread (with socket parameter) which treats with client's demands.
- So now server can continue listening other clients connections...



Threaded server results



Solutions

First job 1/2

```
package com.ingesup.java.cliserv;
import java.io.*;
import java.net.*;

public class Serveur2
{
	public static void main(String args[])
	{
		String[] chaines={"Arrête de boire", "Arrête de fumer","Va faire la vaisselle"};
		try {
			ServerSocket srvr = new ServerSocket(1234);
			System.out.println("Serveur à l'écoute...");
			Socket skt = srvr.accept();
			System.out.println("Un client est connecté !");
			// on initialise les flux de lecture et d'écriture
			PrintWriter out = new PrintWriter(skt.getOutputStream(),    true);
			BufferedReader in = new BufferedReader(New InputStreamReader(skt.getInputStream()));
			// boucle infinie
			while(true){
				// le serveur reçoit un chiffre
				int nb=Integer.parseInt(in.readLine());
				System.out.println("Serveur : réception "+nb);
				if(nb<0 || nb>=chaines.length){
					System.out.println("le nombre est en dehors des clous. On renvoie \"Tchao\" et on quitte la boucle");
					out.println("Tchao");
					break;
				}
				// sinon renvoi contenu indice du tableau
				System.out.println("Le serveur va renvoyer "+chaines[nb]);
				out.println(chaines[nb]);
			}
			System.out.println("Le serveur s'arrête...");
			out.close();
			skt.close();
			srvr.close();
		}
		catch(Exception e) {
			System.out.println("Problème : "+e.getMessage());
		}
	}
}
```

First job 2/2

```java
package com.ingesup.java.cliserv;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client2
{
	public static void main(String args[]) {
		try {
			Socket skt = new Socket("localhost", 1234);
			// on initialise les flux de lecture et d'écriture
			PrintWriter out = new PrintWriter(skt.getOutputStream(),    true);
			BufferedReader in = new BufferedReader(new   InputStreamReader(skt.getInputStream()));
			Scanner clavier=new Scanner(System.in);

			while(true)
			{
				System.out.print("Entrez un nombre : ");
				int nb=clavier.nextInt();
				// on envoie ce nombre au serveur
				out.println(nb);
				// on attend maintenant la chaine
				String reponse=in.readLine();
				if(reponse.equals("Tchao"))
				{
					System.out.println("Le serveur a renvoyé une fin de droits... Le client s'arrête donc.");
					break;
				}
				System.out.println("Le serveur a renvoyé "+reponse);
			}
			System.out.println("Fin du client. Salut !");
			in.close();
		}
		catch(Exception e)
		{
			System.err.println("Problème client : "+e.getMessage());
		}
	}
}
```

Threaded server 1/2

- Only server has to be changed. Here : main thread

```
package com.ingesup.java.cliserv;
import java.net.ServerSocket;
import java.net.Socket;

public class Serveur3
{
	public static void main(String args[])
	{
		try {
			ServerSocket srvr = new ServerSocket(1234);
			System.out.println("Serveur à l'écoute...");
			while(true)
			{
				Socket skt = srvr.accept();
				System.out.println("Un client est connecté ! Port "+skt.getPort());
				new Thread(new ServeurThread(skt)).start();
			}
		}
		catch(Exception e) {   System.out.println("Problème : "+e.getMessage());}
	}
}
```

Threaded server 2/2

```java
package com.ingesup.java.cliserv;

import java.io.*;
import java.net.Socket;

public class ServeurThread implements Runnable
{
	private Socket socket;
	private static String[] chaines={"Arrête de boire", "Arrête de fumer","Va faire la vaisselle"};

	public ServeurThread(Socket _socket)
	{
		socket = _socket;
	}
	@Override
	public void run()
	{
		try
		{
			// on initialise les flux de lecture et d'écriture
			PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
			BufferedReader in = new BufferedReader(new   InputStreamReader(socket.getInputStream()));

			// boucle infinie
			while(true)
			{
				// le serveur reçoit un chiffre
				int nb=Integer.parseInt(in.readLine());
				System.out.println("Le serveur a reçu de "+socket.getPort()+" le nombre "+nb);
				if(nb<0 || nb>=chaines.length)
				{
					System.out.println("le nombre est en dehors des clous. On renvoie \"Tchao\" à "+
						socket.getPort()+" et on quitte la boucle");
					out.println("Tchao");
					break;
				}
				// sinon on renvoie le contenu de l'indice du tableau
				System.out.println("Le serveur va renvoyer à "+socket.getPort()+" le texte "+chaines[nb]);
				out.println(chaines[nb]);
			}
			System.out.println("Le thread s'arrête pour "+socket.getPort());
		} catch (IOException e){	System.out.println("Problème : "+e.getMessage());}
	}
}
```


Fluid Science