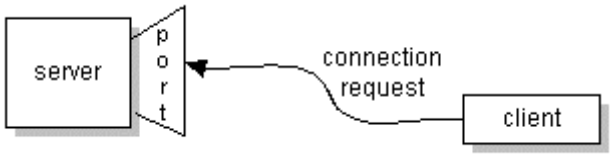
TP N°6 : Programmation Réseau avec les sockets TCP

Objectifs du TP :

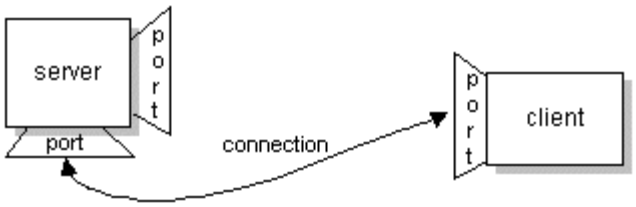
Ce TP a pour objectif de manipuler la communication client/serveur via les sockets avec java.

Les Sockets

Un socket est un point de terminaison dans une communication *bidirectionnelle* entre deux programmes fonctionnant sur un réseau.
Un socket est associé à un *numéro de port* afin que la couche TCP puisse identifier l'application vers laquelle les données doivent être transmises.
En fonctionnement normal, une application serveur fonctionne sur un ordinateur et possède un socket d'écoute associée à un port d'écoute. Le serveur attend une demande de connexion de la part d'un client sur ce port.



Si tout se passe bien, le serveur accepte la connexion. Suite à cette acceptation, le serveur crée un nouveau socket associé à un nouveau port appelé Socket d'échange. Ainsi il pourra communiquer avec le client, tout en continuant l'écoute sur le socket initial appelé Socket d'écoute en vue d'autres connexions.



Exercice 1 :

L'exercice a pour but de développer un serveur TCCP/IP qui écoute sur le port 10000 et qui accepte des connexions clientes. Le programme client se connecte au serveur, puis lui envoie un message. Le message est analysé par le serveur, qui lui répond.

L'exemple suivant montre deux classes :

- L'un du coté serveur : *SocketServeur*
- L'autre du coté client : *SocketClient*

Coté serveur :

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

public class SocketServeur {

    public static void main(String argv[]) {
        int port = 0;
        Scanner keyb = new Scanner(System.in);

        // ..............................................................

        System.out.print("Port d'écoute : ");

        try {
            port = keyb.nextInt();
        } catch (NumberFormatException e) {
            System.err.println("Le paramètre n'est pas un entier.");
            System.err.println("Usage : java ServeurUDP port-serveur");
            System.exit(-1);
        }

        try {

        // ..............................................................

            ServerSocket serverSocket = new ServerSocket(port);

            // ..........................................................

            Socket socket = serverSocket.accept();

            // ..........................................................

            ObjectOutputStream output =
                    new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream input =
                    new ObjectInputStream(socket.getInputStream());
            // ..........................................................
            String chaine = (String) input.readObject();
            System.out.println(" recu : " + chaine);

            // ..........................................................
            System.out.println(" ca vient de : " + socket.getInetAddress() +
":" + socket.getPort());

            // ..........................................................
            output.writeObject(new String("bien recu"));

        } catch (Exception e) {
            System.err.println("Erreur : " + e);
        }
    }
}
```

Coté client :

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

class SocketClient {

    public static void main(String argv[]) {
       int port = 0;
        String host = "";
        Scanner keyb = new Scanner(System.in);

        // ..............................................................

        System.out.print("Nom du serveur : ");
        host = keyb.next();
        System.out.print("Port du serveur : ");

        try {
            port = keyb.nextInt();
        } catch (NumberFormatException e) {
            System.err.println("Le second paramètre n'est pas un entier.");
            System.exit(-1);
        }

        // ..............................................................
        try {

            // ..........................................................
            InetAddress adr = InetAddress.getByName(host);

            // ..........................................................
            Socket socket = new Socket(adr, port);

            // ..........................................................
            ObjectOutputStream output =
                    new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream input =
                    new ObjectInputStream(socket.getInputStream());

            // ..........................................................
            output.writeObject(new String("ma  première socket"));

            // ..........................................................
            String chaine = (String) input.readObject();
            System.out.println(" recu du serveur : " + chaine);

        } catch (Exception e) {

            System.err.println("Erreur : " + e);

        }
    }
}
```

1. Commenter les lignes de code des classes : *SocketClient* , *SocketServeur* .
2. Testez ces deux programmes échangeant la phrase « ma première socket » dans les deux cas suivants :
    a. Les deux programmes se trouvant dans la même machine.
    b. Les deux programmes dans deux machines distants.

    Note : Il faut d'abord lancer le serveur pour qu'il soit à l'écoute.

3. Effectuer un schéma indiquant les flux de messages échangés

Exercice 2 :

On souhaite échanger entre le client/serveur les objets d'une classe voiture via les sockets.
la classe voiture

Pour cela le client Crée un objet voiture et l'envoi au serveur pour lui fixer une quantité de carburant avec la méthode *setCarburant().*

```
import java.io.*;

class voiture implements Serializable {

    private int carburant;
    private String model;
    private String type;
    private static int capacite = 300;

    voiture(String _type, String _model) {
        type = _type;
        model = _model;
        carburant = 0;
    }

    public void setCarburant(int c) {
        int maxi = capacite - carburant;
        if (c < maxi) {
            carburant += c;
            System.out.println("Le remplissage a été effectué sans problème.");
        } else {
            carburant = capacite;
            System.out.println((c - maxi) + " litre(s) de carburant ont
débordé.");
        }
    }
    public int getCarburant() {
        return carburant;
    }
    public int getCapacite() {
        return capacite;
    }
}
```

1. Ecrire la partie client de l'application et la partie serveur.
2. Tester cette application sur la même machine.
3. Tester cette application sur deux machines reliées par réseaux.

Exercice 3 :

L'application distribuée à réaliser est basique. Elle est composée d'un client et d'un serveur.

La partie client envoie des données concernant des personnes au serveur. Ces données sont composées de 2 informations : un entier (*int*) pour l'âge de la personne et une chaîne de caractères (*String*) pour son nom. A chaque envoi d'informations sur une personne de la part d'un client, le serveur retourne au client un entier correspondant à l'identificateur du client