Objectifs:

• Savoir échanger d'information via des objets Socket `a adapter au protocole voulu et à englober dans un thread suivant des cas.

• Savoir créer une application Client/Serveur entre deux machines permettant de communiquer et échanger des flux de données.

Matériel nécessaire :

• Postes informatiques sous Windows dotés de cartes réseaux.

Exercice 01 : Socket (TCP)

Travail demandé :

1. Commencer par éditer et commenter les lignes de la classe Server1.java

```
1  package reseau;
2
3  import java.io.IOException;
7
8  public class serveur1
9  {
10
11 public static void main( String [] args) throws IOException {
12     ServerSocket listener = new ServerSocket (9090);//à l'aide de la classe ServerrSocket on associe un objetà un port pour attendre la connexion
13     System.out.println("en attente de connection...");
14     try {
15         while (true) {
16             Socket socket = listener.accept();//essayer d'etablir la connexion en affectant le resultat à un objet de la classe socket
17             try{
18                 PrintWriter out =
19                 new PrintWriter( socket.getOutputStream(),true);//on cree un objet de la classe PrintWriter qui effectue l'envoie de la requete
20                 out.println("vous etesbien connecté au serveur "+socket.getLocalSocketAddress());
21             }
22             finally {
23                 socket.close ();//arrete l'envoie des requetes
24             }
25         }
26     }
27     finally {
28         listener.close ();//arreter la connexion en le serveur et le client
29     }
30     }
31 }
32
```

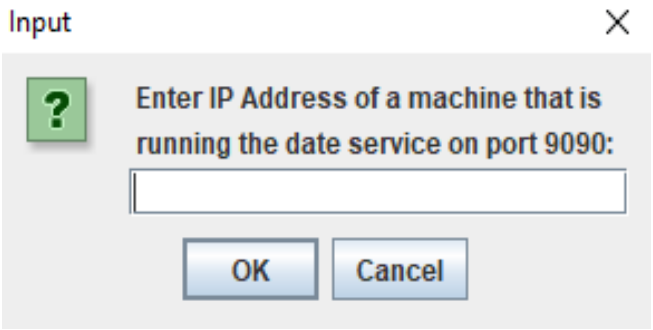
2.Compiler et Exécuter cette classe. Commenter le résultat.

⇨ **Après la compilation le serveur lance la connexion et attend la réponse du client**

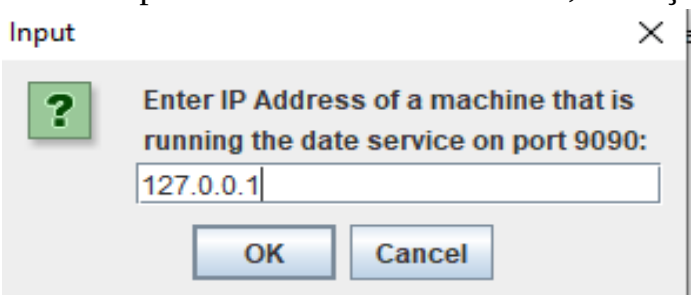
3.Editer et commenter les lignes de la classe Client1.java.

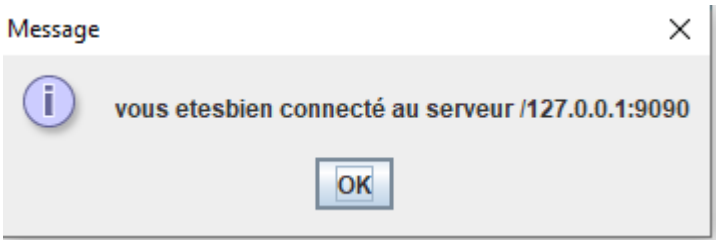
```
package reseau;

import java.io.BufferedReader;

public class client1 {
    private static Socket s;
//affichage du message dans un interface graphique qui demande de saisir l'adresse IP de la machine
    public static void main( String [] args) throws IOException {
        String serverAddress = JOptionPane.showInputDialog(
        " Enter IP Address of a machine that is\n" +
        " running the date service on port 9090: ");
        s = new Socket(serverAddress,9090);
        BufferedReader input =
        new BufferedReader(new InputStreamReader(s.getInputStream()));
        String answer = input . readLine ();
        JOptionPane.showMessageDialog(null , answer );
        System.exit(0);
    }
}
```

4. Compiler et Exécuter cette classe. Commenter le résultat.



⇨ Apres avoir taper localhost dans la console, on reçoit :





5. Est-ce-que le serveur peut traiter plusieurs clients simultanément ?

⇨ **Non, le serveur ne peut pas traiter plusieurs clients simultanément car on ne travaille pas avec la notion des threads.**

Exercice 02 : Socket (TCP)

Travail demandé :

1.Créer une classe TCPclient a comme attributs :serverPort,serverName ,et clientSocket

```
12 public class TCPClient {
13     private int portServer;
14     private String serverName;
15     private Socket client;
16     private Scanner scanner = new Scanner(System.in);
17     public TCPClient(int portServer, String serverName) {
18         this.portServer = portServer;
19         this.serverName = serverName;
20 
21     }
22     public void connect() {
23         try {
24             this.client = new Socket(this.serverName, this.portServer);
25             System.out.println("connected to : "+client.getRemoteSocketAddress());
26             client.getRemoteSocketAddress() ;
27         } catch (IOException e) {
28             e.printStackTrace();
29         }
30     }
31 
32     public void sendMessage() {
33         System.out.print("write a message : ");
34         try {
35             OutputStream out = client.getOutputStream();
36             DataOutputStream outData = new DataOutputStream(out);;
37             if(this.client.isConnected()) {
38                 String message = scanner.next();
39                 outData.writeUTF(message);
40             }else {
41                 scanner.close();
42             }
43 
44 
45         } catch (IOException e) {
46             // TODO Auto-generated catch block
47             e.printStackTrace();
48         }
49     }
```

```
    public void recieveMessage() {
        try {
            InputStream in = client.getInputStream();
            DataInputStream dataIn = new DataInputStream(in);// ici ou vient le message
            String message = dataIn.readUTF();
            System.out.println("recieved message : "+ message);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    public Socket getClient() {
        return client;
    }
    public void setClient(Socket client) {
        this.client = client;
    }


}
```

⇨ Menu principale :

```
    public static void main(String[] args) {
        TCPClient client = new TCPClient(1234,"localhost");
        TCPClient client1 = new TCPClient(1234, "localhost");

        client.connect();
        client1.connect();
        while( client.getClient().isConnected() ) {
            client.sendMessage();
            client.recieveMessage();
            client1.sendMessage();
            client1.recieveMessage();
        }


    }

}
```

2.Créer une classe TCPserver a comme attributs :Port,serverSocket ,Socket .

```java
public class TCPServer {
    private int port;
    private ServerSocket serverSocket;
    private Socket server;
    private Scanner scanner = new Scanner(System.in);
    private boolean isDone = false;
    public TCPServer(int port) {
        this.port = port;

        try {
            serverSocket = new ServerSocket(this.port);
            serverSocket.setSoTimeout(6000000);
            System.out.println("server running on address : "+serverSocket.getLocalSocketAddress());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public void serve() {
        try {
            this.server = serverSocket.accept();
        } catch (IOException e1) {
            e1.printStackTrace();
        }

        while(!isDone) {
            recieveMessage();
            sendMessage();
        }
        closeConnection();

    }

    public void sendMessage() {
        System.out.print("write a message : ");
```

```java
    public void sendMessage() {
        System.out.print("write a message : ");

        try {
            OutputStream out = server.getOutputStream();
            DataOutputStream outData = new DataOutputStream(out);
            String message = scanner.next();
            outData.writeUTF(message);

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void recieveMessage() {
        try {
            InputStream in = server.getInputStream();
            DataInputStream dataIn = new DataInputStream(in);
            String message = dataIn.readUTF();
            System.out.println("recieved message : "+ message);
            if(message.equalsIgnoreCase("Over&Out")) {
                this.isDone = true;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    public void closeConnection() {
        try {
            this.server.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

⇨ Menu principale :

```
    public static void main(String[] args) {
        TCPServer server = new TCPServer(1234);
        server.serve();

    }

}
```

5. Est-ce-que le serveur peut traiter plusieurs clients simultanément ?

⇨ **Non, le serveur ne peut pas traiter plusieurs clients simultanément car car il n'implémente pas les threads.**

Exercice 03 : Socket (UDP)

Travail demandé :

1.Commencer par éditer et commenter les lignes de la classe Send UDP.java.

```
package reseau;



import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class Send_UDP {

    public static void main(String[] args) throws Exception {
        /*on cree une socket afin d'etablir une connexion */
        DatagramSocket ds = new DatagramSocket();
        String str = " Welcome ";
        InetAddress ip = InetAddress.getByName("127.0.0.1");

        /* Ce constructeur de DatagramPacket est utilisé pour envoyer les paquets */
        /* creer un packet à envoyer qui prend en parametre le message envoyé en bits ,sa longuer ,l'adresse IP et le port
         */
        DatagramPacket dp=new DatagramPacket( str. getBytes (), str. length (),ip ,3000);
        ds. send(dp );//envoyer le packet à  l'aide de la socket créer
        ds. close ();
    }

}
```

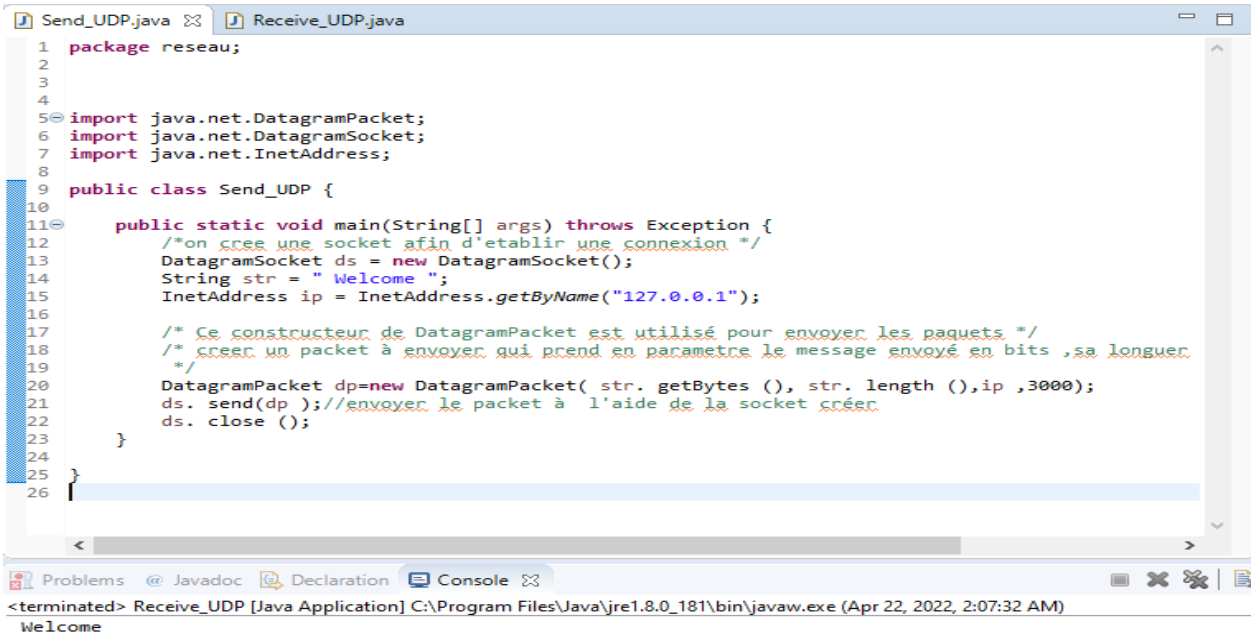
2. Compiler et Exécuter cette classe. Commenter le résultat.

Le client commence par la création du socket, après le remplissage du paquet á envoyer avec le message ‘’welcome’’ doté de l’adresse et le numéro de port du serveur qui est á l’écoute, ensuite est l’envoie et la fermeture du socket .

3. Editer et commenter les lignes de la classe Receive_UDP.java.

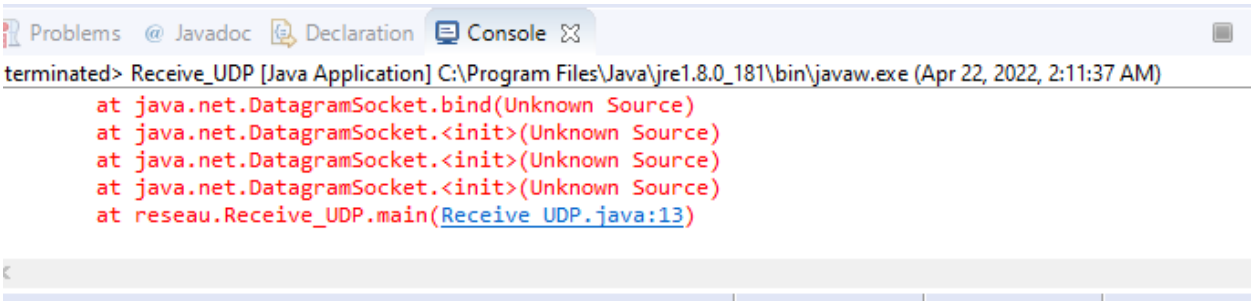
```java
package reseau;

import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class Receive_UDP {
    /* Classe Receive UDP.java. */


        public static void main(String[] args) throws Exception{
            /* il cree une prise datagram et la lie avec
            * le num´ero de port disponible sur la machine localhost.*/
            DatagramSocket ds = new DatagramSocket(3000);
            byte []buf = new byte[1024];

            /* Ce constructeur de DatagramPacket est
            * utilis´e pour recevoir les paquets .*/

            DatagramPacket dp = new DatagramPacket(buf,1024);
            ds.receive(dp);
            String str = new String(dp.getData(),0,dp.getLength());
            System.out.println(str);
            ds.close ();
        }
    }
```

4. Compiler et Exécuter cette classe. Commenter le résultat.



5. Est-ce-que le serveur peut traiter plusieurs clients simultanément ?

⇨ **Non, car le serveur n'implémente pas les threads.**



4 .Application

Travail demandé

Ecrire une application Client/Serveur permettant de calculer le prix TTC des articles Achetées par un client. Le serveur demande les informations suivantes :

1. Le nom & prénom du client.

2. Le couple (Prix Hors T axe Article, Nombre Article) de chaque article acheté.

Le programme retourne le prix hors taxe, le taux de la valeur ajoutée, et le prix TTC des achats du client, en précisant la date de l'opération. TVA =20% du prix de l'article, pour chaque article, on a : Prix TTC = Prix Hors T axe Article + T V A

⇨ Pour réaliser ce travail, on a créé une classe article, qu'on va instancier après dans le socket client afin de l'envoyer au serveur pour la traiter.

⇨ Coté Client :

```java
import java.net.Socket;
import java.util.Scanner;
import java.io.DataInputStream;
import java.io.DataOutputStream;


public class client {
    public static void main(String[] args) {
        try {

            Socket s=new Socket(InetAddress.getLocalHost(),2000);
            DataOutputStream out=new DataOutputStream(s.getOutputStream());
            Scanner sc=new Scanner(System.in);
            System.out.println("Entrez votre nom: ");
            String nom=sc.next();
            out.writeUTF(nom);
            System.out.println("Entrez votre prenom: ");
            String prenom=sc.next();
            out.writeUTF(prenom);
            System.out.println("Entrez le nombre d'article: ");
            int nbr=sc.nextInt();
            out.writeInt(nbr);
            System.out.println("Entrez le prix d'article d'hors taxe: ");
            float prix=sc.nextFloat();
            out.writeFloat(prix);
            out.flush();
            DataInputStream inp=new DataInputStream(s.getInputStream());
            String res=inp.readUTF();
            System.out.println(res);
            s.close();
        }
        catch(Exception e) {

        }

    }
}
```

⇨ Coté Serveur :

```java
package reseau;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Serveur {
private static long calcul(float prix,float tva)
{
    return(long)(prix+tva);
}
public static void main(String []args)throws IOException{
    ServerSocket ss=new ServerSocket(2000);
    System.out.println("attente de connexion");
    Socket client=ss.accept();
    DataInputStream inp=new DataInputStream(client.getInputStream());
    String nom=inp.readUTF();
    String prenom=inp.readUTF();
    int nbr=inp.readInt();
    float prix_taxe=inp.readFloat();
    float tva=(float)(prix_taxe*0.2);
    float prixtt=nbr*calcul(prix_taxe,tva);
    String str="Bonjour"+" "+nom+""+prenom+"\nle prix hors taxe est:"+prix_taxe+""+ "nombre d'article est:"+nbr+
    "\n et le taux de valeur ajoute est:"+tva+
    "\n le prix totale avec taxe est :"+prixtt+"\n";
    DataOutputStream out=new DataOutputStream(client.getOutputStream());
    out.writeUTF(str);
    client.close();
    ss.close();

}
}
```

⇨ Exécution :

Problems | Javadoc | Declaration | Console

```
<terminated> client [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Apr 23, 2022, 12:33:13 AM)
Entrez votre nom:
aya
Entrez votre prenom:
fechtali
Entrez le nombre d'article:
2
Entrez le prix d'article d'hors taxe:
150
Bonjour ayafechtali
le prix hors taxe est:150.0nombre d'article est:2
 et le taux de valeur ajoute est:30.0
 le prix totale avec taxe est :360.0
```