Objectifs :

- Savoir échanger d'information via des objets Socket à 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.

1 .Threads

Travail demandé :

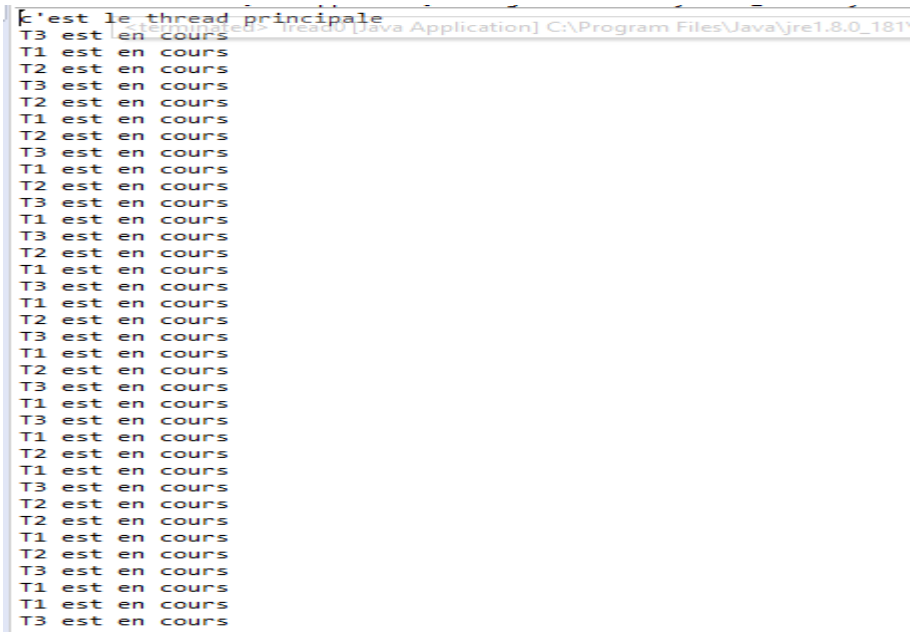
1. Compiler et Exécuter le programme suivant, commenter les résultats obtenus.

```java
package reseau;

public class Tread0 extends Thread {
    String name;

    public Tread0(String name) {
        super();
        this.name = name;
    }
    public void run() {
        for(int i = 1; i < 100 ; i++){
            try{
                sleep((long)(Math.random()*100));
            }catch (InterruptedException ie) {
                ie.getStackTrace();
            }
            System.out.println(name);
        }
    }
    public static void main(String[] args) {
        Tread0 T1 = new Tread0("T1 est en cours");
        Tread0 T2 = new Tread0("T2 est en cours");
        Tread0 T3 = new Tread0("T3 est en cours");
        T1.start();
        T2.start();
        T3.start();
        System.out.println("c'est le thread principale");
    }
}
```

⇨ Exécution :



Commentaire : les threads sont traités d'une manière aléatoire qui depend de la valeur passé en paramètre à la fonction sleep qui permet de retarder le lancement de l'exécution du thread.

2 .Threads et Suites :

```java
package reseau;

public class Suite extends Thread implements Runnable  {
int n ;
public Suite ( int n ) {
this.n = n ;
}
public void run () {
    float u=2;
for (int i=1;i <n ;i ++){
        u=5*u+5;
        System .out . println ("calcule de la "+n+" itération de la suite u"+n );
        try {
            sleep ((long)( Math . random () * 100));
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System .out . println ("la valeur de la suite à la "+i+" itération de u"+n+" est:"+u );
        }

        }
public static void main ( String args []) {
    Suite a = new Suite (2);
    Suite b = new Suite (5);
    Suite c = new Suite (8);
a. start ();
b. start ();
c. start ();
System.out.println("c'est le threads principal.");
}}
```

⇨ Exécution :

```
<terminated> Suite [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Apr 23, 2022, 11:23:11 PM)
calcule de la 2 itération de la suite u2
c'est le threads principal.
calcule de la 5 itération de la suite u5
calcule de la 8 itération de la suite u8
la valeur de la suite à la 1 itération de u5 est:15.0
calcule de la 5 itération de la suite u5
la valeur de la suite à la 1 itération de u8 est:15.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 1 itération de u2 est:15.0
la valeur de la suite à la 2 itération de u5 est:80.0
calcule de la 5 itération de la suite u5
la valeur de la suite à la 2 itération de u8 est:80.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 3 itération de u5 est:405.0
calcule de la 5 itération de la suite u5
la valeur de la suite à la 3 itération de u8 est:405.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 4 itération de u5 est:2030.0
la valeur de la suite à la 4 itération de u8 est:2030.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 5 itération de u8 est:10155.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 6 itération de u8 est:50780.0
calcule de la 8 itération de la suite u8
la valeur de la suite à la 7 itération de u8 est:253905.0
```

3.Serveur multi-threads :

Travail demandé :

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

2. Compiler et Exécuter ces classes. Commenter les résultats.

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

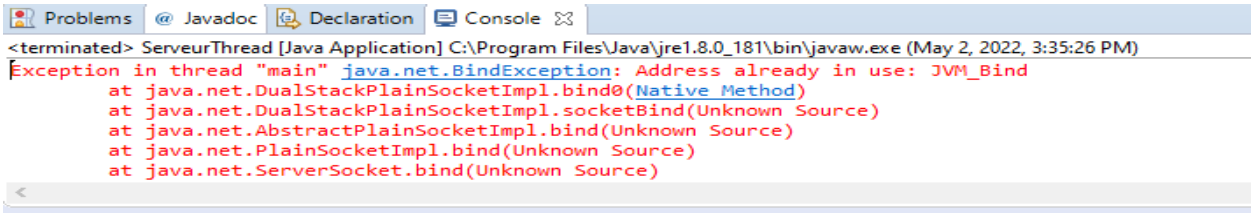
```
1  package reseau;
2  import java.io.IOException;
3  import java.net.ServerSocket;
4  import java.net.Socket;
5  public class ServeurThread {
6      public static void main(String[] args) throws IOException {
7          try //creer un nouveau serverSocket qui entend entend sur le port 4200
8          (ServerSocket serverSocket = new ServerSocket (4200);){
9              System .out . println ("En attent de connection ... ");
0              // accepter les connexions pour toujours
1              while (true) {
2                  Socket socket = serverSocket.accept();
3                  System.out.println (" Connection établie ");
4                  new Thread (new ClientHandler(socket)).start();
5              }
6          }
7      }
8
9  }
0
1
2
```

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

public class ClientHandler implements Runnable{
    private Socket socket ;

    public ClientHandler(Socket socket ){
        this.socket = socket ;
    }
    public void run (){
    // acceptation du flus entrant
        DataInputStream in1 ;
        try {
            in1 = new DataInputStream(this.socket.getInputStream());
            String nomClient = in1.readUTF ();
            // traitement de la donn´ee
            String str = "Bienvenu "+ nomClient +", t'es bien connecté ";
            /// envoie de la reponse
            DataOutputStream out = new DataOutputStream(this.socket.getOutputStream());
            out.writeUTF(str );
        } catch ( IOException e) {
            e.printStackTrace();
        }
    }
}
```

⇨ Exécution :



❖ On constate que l'exécution génère des exceptions, parceque le client Handler permet de gérer les clients et les deux classes qu'on a exécuté font parties du fichier serveur .

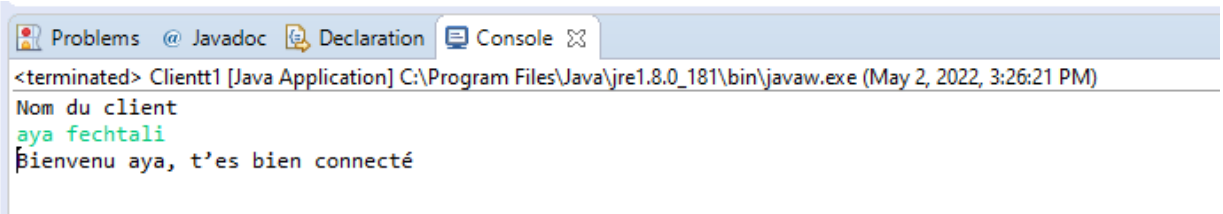
```java
package reseau;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class Clientt1 {

    public static void main(String[] args) throws UnknownHostException ,IOException {
        Socket client = new Socket("127.0.0.1",4200);
        System.out.println("Nom du client ");
        Scanner sc = new Scanner(System.in);
        String nomClient = sc.next();
        // envoie des donnees au serveur
        DataOutputStream out = new DataOutputStream(client.getOutputStream());
        out.writeUTF(nomClient);
        // recuperation des donnees envoyees par le serveur
        DataInputStream in = new DataInputStream(client.getInputStream());
        String resp = in.readUTF();
    }

}
```

⇨ Exécution :



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

⇨ **Oui ,le Serveur peut traiter plusieurs client grâce au principe des threads**

6.Créer plusieurs clients et tester la capacité du serveur à gérer plusieurs clients. Commenter les résultats trouvés

⇨ **Avec le principe des thread avec un seul serveur on peut traiter plusieurs Clients.**