



TP : Programmation réseau

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 Treads

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

```
public class Tread0 extends Thread /*implements Runnable*/ {  
    String name;  
    public Tread0(String name) {  
        this.name = name;  
    }  
    public void run() {  
        for (int i=1; i<100 ; i++){  
            try {  
                sleep((long)(Math.random() * 100));  
            } catch (InterruptedException ie) {  
                ie.printStackTrace();  
            }  
            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 threads principal.");  
    }  
}
```

2 Thread et Suites

Écrire un programme java permettant de calculer trois termes différents de la suite $(U_n)_n$ en utilisant les threads

$$(U_n)_n : \begin{cases} U_0 = 2; \\ U_{n+1} = 5 * U_n + 5 \end{cases} \quad (1)$$

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. Éditer et commenter les lignes de la classe Client1.java.
4. Compiler et Exécuter cette classe. Commenter le résultat.
5. Est-ce-que le serveur peut traiter plusieurs clients simultanément?

```
public class ServeurThread {
    public static void main(String[] args) throws IOException {
        try {
            // create a new ServerSocket which listens on port 4200
            ServerSocket serverSocket = new ServerSocket(4200);
        } {
            System.out.println("En attent de connection...");
            // accept connections forever
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("Connection établie");
                new Thread(new ClientHandler(socket)).start();
            }
        }
    }
}
```

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

```

public class Client1 {

    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();
        //envoi des données au serveur
        DataOutputStream out= new DataOutputStream(client.getOutputStream());
        out.writeUTF(nomClient);
        //récupération des données envoyées par le serveur
        DataInputStream in=new DataInputStream(client.getInputStream());
        String resp= in.readUTF();
        System.out.println(resp);
    }
}

```

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