TP : Serveur/MultiClients

Un serveur TCP/IP multiclients (1)

- le serveur précédent accepte plusieurs connexions simultanées, mais ne traite qu'un client à la fois, les autres sont mis en attente
- pour y remédier, utiliser les *threads* java (**java.lang.Thread**)

```
try{
    ServerSocket serveur = new ServerSocket(PORT);  while
    (true) {
            //accepter une connexion
                    Socket socket = serveur.accept();
    // créer un thread : pour échanger les données avec le client
                    Connexion c = new Connexion(socket);
                    Thread processus_connexion = new Thread(c);
                    processus_connexion.start();
    }
} catch (IOExeption e) {…}
```

Un serveur TCP/IP multiclients (2)

```
class Connexion implements Runnable
{……
 public Connexion (Socket s)
   {  this.s = s;
   try{
   in=new BufferedReader(
                                          new
   InputStreamReader(s.getInputStream())
   );

   out = new
   PrintWriter(s.getOutputStream());
  } catch (IOExeption e) {…}
```

Un serveur TCP/IP multiclients (3)

```
public void run() {  try{
    while (true) {
          String ligne = in.readLine();
     if (ligne == null) break; //fin de connexion côté client
          output.println(ligne);
          out.flush();
     }
  } catch (IOExeption e) {…}
 finally {try{s.close();}catch (IOException e){}}
```

Un serveur TCP/IP multiclients (4)

- le serveur utilise une classe Connexion implémentant l'interface Runnable (*thread*) pour gérer les échanges de données en tâche de fond. C'est ce *thread* qui réalise le service demandé