

# Java Avancé

Cours 6 : Programmation Réseau

Arsène Lapostolet

15 Février 2024

# Introduction



### Socket

Thread: flux d'entrée/sortie réseau.

- Même interface que les autres entrées/sortie (fichiers, console)
- Utilise les couches réseau TCP ou UDP pour communiquer avec d'autres machines
- Le cours couvre le cas des sockets TCP



## **Socket Java**

Modèle client-serveur.

#### Classes:

- ServerSocket : se lie à un port sur la machine serveur
- Socket : se connecte à un port sur une IP

ServerSocket.accept bloque jusqu'à la connexion d'un client et retourne une Socket ouverte vers le client à chaque connexion.



#### Coté Serveur

Attente d'un client et ouverture des flux :

```
1 final var serverSocket = new ServerSocket(3000);
2 final var socket = serverSocket.accept();
3
4 final var streamFromClient = new BufferedReader(
5     new InputStreamReader(socket.getInputStream())
6 );
7
8 final var streamToClient = new
PrintWriter(socket.getOutputStream(), true);
```



#### Coté Client

Connexion au serveur et ouverture des flux :

```
final var clientSocket = new Socket(3000, "localhost");

final var streamFromClient = new BufferedReader(
    new InputStreamReader(clientSocket.getInputStream())

);

final var streamToClient = new PrintWriter(
    clientSocket.getOutputStream(),
    true

10);
```



### Socket et threads

println et readLine sont bloquants :

- Un println côté client bloque jusqu'au prochain readLine côté serveur
- Un readLine côté client bloque jusqu'au prochain println côté serveur

Et vice-versa.

Il faut utiliser des threads!



## Exemple : serveur de chat

```
public class ChatServer {
                                                            Java
  private final int port;
  private final List<PrintWriter> connectedClients;
  public ChatServer(int port) {
    this.port = port;
    this connectedClients = new ArrayList<>();
```



```
public void start() {
        final var serverSocket = new ServerSocket(port);
        while (true) {
          final var client = serverSocket.accept();
          connectedClients.add(
20
            new PrintWriter(client.getOutputStream(),true)
22
          new Thread(() -> listenForMessage(connectedClient))
                   .start();
25
```



```
private void listenForMessage(Socket client) {
      final BufferedReader messages = new BufferedReader(
          new InputStreamReader(client.getInputStream())
29
30
31
32
      while (true) {
33
          final var message = messages.readLine();
34
35
          for (final var connectedClient : connectedClients) {
36
               connectedClient.println(messageFromClient);
37
38
39 }
```



## Exemple : client de chat

```
public class ChatServer {
                                                            Java
  private final int port;
  private final String address;
  private BufferedReader messagesFromServerStream;
  private PrintWriter messagesToServerStream, output;
  public ChatClient(int port, String address, PrintWriter out) {
        this port = port;
        this address = address;
        this.output = out;
```



```
public void connect() {
      final var socket = new Socket(address, port);
      messagesFromServerStream = new BufferedReader(
          new InputStreamReader(socket.getInputStream())
      );
      messagesToServerStream = new PrintWriter(
20
          socket.getOutputStream(),
          true
22
23
      new Thread(this::listenForMessages).start();
25
```



```
private void listenForMessages(){
      while (true){
         final var message = messagesFromServerStream.readLine();
30
         output.println(message);
31
32
33
34
    public void sendMessage(String message){
35
      messagesToServerStream.println(message);
36
37
38 }
```