QUALITÉ DE DÉVELOPPEMENT

LA PROGRAMMATION RÉSEAU AVEC JAVA

🎓 2A - Bachelor Universitaire de Technologie
🏛 IUT d'Orsay - Université Paris-Saclay - 2024/2025

Idir AIT SADOUNE
idir.ait-sadoune@universite-paris-saclay.fr

PLAN

- Notions de base sur le réseau
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Exemple

Retour au plan - Retour à l'accueil

PLAN

- Notions de base sur le réseau
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Exemple

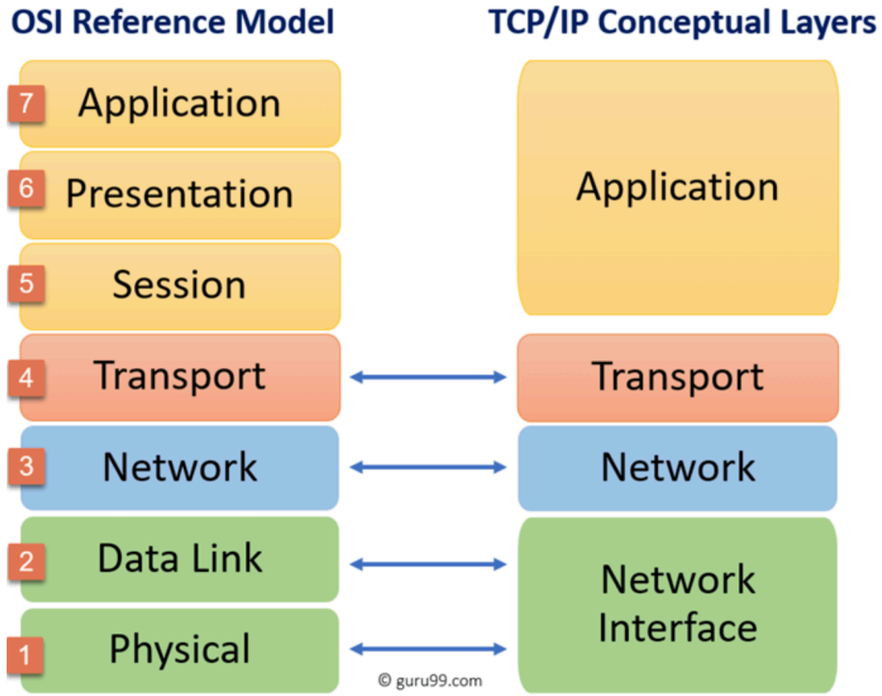
Retour au plan - Retour à l'accueil

LA SOCKET

- Au coeur de la prise en charge du réseau en **Java** se trouve le concept de **socket**.
  - une **socket** identifie un point de terminaison dans un réseau.
- Une **Socket** permet à un seul ordinateur de servir plusieurs clients à la fois, ainsi que de servir différents types d'informations.
- Cela est accompli grâce à l'utilisation d'un port, qui est une socket numéroté sur une machine particulière.
  - on dit qu'un processus serveur écoute un port jusqu'à ce qu'un client s'y connecte.
- Un serveur est autorisé à accepter plusieurs clients connectés au même numéro de port, bien que chaque session soit unique.
  - pour gérer plusieurs connexions client, un processus serveur doit être **multithread**.

LES PROTOCOLS TCP ET IP

La communication de socket s'effectue via un protocole.



LES PROTOCOLS TCP ET IP

- Internet Protocol (IP) est un protocole de routage de bas niveau qui divise les données en petits paquets et les envoie à une adresse sur un réseau.
- Transmission Control Protocol (TCP) est un protocole de haut niveau qui parvient à enchaîner de manière robuste ces paquets, en les ordonnant et en les retransmettant si nécessaire pour transférer les données de manière fiable.

UNE ADRESSE INTERNET

- **Une adresse Internet** est un numéro qui identifie de manière unique chaque ordinateur sur le **Net**.
- Toutes les adresses Internet sont constituées de valeurs sur 32 bits, organisées en quatre valeurs de 8 bits (**IPv4**, Internet Protocol version 4).
  - `192.54.12.68`
- Un nouveau schéma d'adressage, appelé **IPv6**, est entré en jeu (Valeurs de 128 bits organisées en huit blocs de 16 bits).
  - `2001:0db8:0000:85a3:0000:0000:ac1f:8001`

LE NOM D'UNE ADRESSE INTERNET

- Le nom d'une adresse Internet, appelé nom de domaine, décrit l'emplacement d'une machine dans un espace de noms.
- Par exemple, www.HerbSchildt.com est dans le domaine de premier niveau COM (réservé aux sites commerciaux américains);
  - il s'appelle HerbSchildt, et www identifie le serveur pour les requêtes Web.
- Un nom de domaine Internet est mappé à une adresse IP par le **Domain Naming Service** (**DNS**).
- Cela permet aux utilisateurs de travailler avec des noms de domaine, alors qu'Internet fonctionne avec des adresses IP.

NUMÉRO DE PORT

- Une fois la connexion établie, un protocole de haut-niveau, qui dépend du numéro de port que vous utilisez, s'ensuit.
- TCP/IP réserve les premiers 1 024 ports à des protocoles spécifiques :
  - 21 pour FTP; 23 pour Telnet; 25 pour e-mail; 80 pour HTTP; ...
- Il appartient à chaque protocole de déterminer comment un client doit interagir avec le port. .
- Par exemple, HTTP est le protocole utilisé par les navigateurs Web et les serveurs pour transférer des pages hypertextes et des images.

PROGRAMMATION RÉSEAU

- Les sockets TCP/IP sont utilisées pour implémenter des connexions fiables, bidirectionnelles, persistantes, point à point et basées sur des flux entre les hôtes sur Internet.
- Une socket peut être utilisé pour connecter le **système d'E/S de Java** à d'autres programmes qui peuvent résider soit sur la même machine, soit sur n'importe quelle autre machine sur Internet.

JAVA ET LA PROGRAMMATION RÉSEAU

- L'une des raisons pour lesquelles **Java** est le premier langage pour la programmation réseau sont les classes définies dans le package `java.net`.

- Voici deux types de sockets TCP en **Java** :
  1. La classe `ServerSocket` est conçue pour être un « écouteur », qui attend que des clients se connectent avant de faire quoi que ce soit
     $\rightarrow$ Ainsi, `ServerSocket` est destiné aux serveurs.
  2. La classe `Socket` est destinée aux clients. Elle est conçue pour se connecter aux sockets du serveur et initier des échanges de protocole.

PLAN

- Notions de base sur le réseau
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Exemple

Retour au plan - Retour à l'accueil

TCP/IP CLIENT SOCKET

- La création d'un objet `Socket` établit implicitement une connexion entre le client et le serveur.
- Voici deux constructeurs utilisés pour créer des sockets client:

`Socket(String hostName, int port)`	Creates a socket connected to the named host and port.
`Socket(InetAddress ipAddress, int port)`	Creates a socket using a preexisting InetAddress object and a port.

TCP/IP CLIENT SOCKET

- Une `Socket` peut être examiné pour les informations d'adresse et de port qui lui sont associées, en utilisant les méthodes suivantes:

`InetAddress getInetAddress()`	Returns the InetAddress associated with the Socket object. It returns null if the socket is not connected.
`int getPort()`	Returns the remote port to which the invoking Socket object is connected. It returns 0 if the socket is not connected.

TCP/IP CLIENT SOCKET

- Vous pouvez accéder aux flux d'entrée et de sortie associés à une `Socket`.

`InputStream getInputStream()`	Returns the InputStream associated with the invoking socket.
`OutputStream getOutputStream()`	Returns the OutputStream associated with the invoking socket.

EXEMPLE

```java
public class Whois {
  public static void main(String[] args) throws IOException {
    int c;
    // Create a socket connected to internic.net, port 43.
    Socket s = new Socket("whois.internic.net", 43);
    // Obtain input and output streams.
    InputStream in = s.getInputStream();
    OutputStream out = s.getOutputStream();
    // Construct a request string.
    String str = (args.length == 0 ? "MHProfessional.com" : args[0]) + "\n";
    byte buf[] = str.getBytes();
    // Send request.
    out.write(buf);
    // Read and display response.
    while ((c = in.read()) != -1) {
      System.out.print((char) c);
    }
    s.close();
  }
}
```

PLAN

- Notions de base sur le réseau
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Exemple

Retour au plan - Retour à l'accueil

TCP/IP SERVER SOCKETS

- La classe `ServerSocket` est utilisée pour créer des serveurs qui écoutent des programmes clients souhaitant se connecter à eux sur des ports publiés.
- Lorsque vous créez un `ServerSocket`, il s'enregistre auprès du système comme ayant un intérêt dans les connexions client.

`ServerSocket(int port)`	Creates server socket on the specified port with a queue length of 50.
`ServerSocket(int port, int maxQueue)`	Creates a server socket on the specified port with a maximum queue length of `maxQueue`.

EXEMPLE

```java
public class Serveur {
  public static void main(String[] args) throws Exception {
    ServerSocket s = new ServerSocket(9999);
    Socket soc = s.accept();

    ObjectOutputStream out = new ObjectOutputStream(soc.getOutputStream());
    ObjectInputStream in = new ObjectInputStream(soc.getInputStream());

    int[] tableauAEmettre = { 7, 8, 9 };
    out.flush();
    out.writeObject(tableauAEmettre);
    out.flush();

    Object objetRecu = in.readObject();
    int[] tableauRecu = (int[]) objetRecu;

    in.close();
    out.close();
    soc.close();
  }
}
```

EXEMPLE

```java
public class Client {
  public static void main(String[] args) throws Exception {
    Socket socket = new Socket("localhost", 9999);

    ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

    int[] tableauAEmettre = { 1, 2, 3 };
    out.flush();
    out.writeObject(tableauAEmettre);
    out.flush();

    Object objetRecu = in.readObject();
    int[] tableauRecu = (int[]) objetRecu;

    in.close();
    out.close();
    socket.close();
  }
}
```

PLAN

- Notions de base sur le réseau
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Exemple

Retour au plan - Retour à l'accueil

EXEMPLE

```
1  public class ServerThread extends Thread {
2    private Socket s;
3
4    public ServerThread(Socket s) {
5      this.s = s;
6    }
7
8    public void run() {
9      try {
10       BufferedReader readData = new BufferedReader(new InputStreamReader(this.s.getInputStream()));
11       PrintWriter writeData = new PrintWriter(new OutputStreamWriter(this.s.getOutputStream()));
12
13       Random rng = new Random();
14       int readValue;
15       int response = -1;
16       int total = 0;
17
18       do {
19         int v1 = rng.nextInt(100) + 1;
20         int v2 = rng.nextInt(100) + 1;
21         int op = rng.nextInt(2);
22         String ops = (op == 0) ? "+" : "-";
23
24         writeData.println(v1 + "," + ops + "," + v2);
```

EXEMPLE

```java
public class Server {
  public static void main(String[] args) {
    try {
      ServerSocket server = new ServerSocket(5555);
      int cpt = 0;
      do {
        Socket client = server.accept();
        ServerThread c = new ServerThread(client);
        c.start();
        cpt++;
      } while (cpt<50);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

EXEMPLE

```java
public class Client {
  public static void main(String[] args) {
    try {
      Socket socket = new Socket("localhost", 5555);

      PrintWriter writeData = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));
      BufferedReader buf = new BufferedReader(new InputStreamReader(socket.getInputStream()));

      Scanner sc = new Scanner(System.in);

      String response = "";
      do {
        String readM = buf.readLine();
        String[] message = readM.split(",");

        int v1 = Integer.valueOf(message[0]).intValue();
        String op = message[1];
        int v2 = Integer.valueOf(message[2]).intValue();

        System.out.print(v1 + " " + op + " " + v2 + " = ");
        int result = sc.nextInt();
        sc.nextLine();

        String resultMessage = String.valueOf(result);
```

MERCI

Version PDF des slides

Retour à l'accueil - Retour au plan