QUALITÉ DE DÉVELOPPEMENT

LA PROGRAMMATION RÉSEAU AVEC JAVA

🎓 2A - Bachelor Universitaire de Technologie
🏛 IUT d'Orsay - Université Paris-Saclay - 2023/2024

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

PLAN

- Networking Basics
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Example

Retour au plan - Retour à l'accueil

PLAN

- Networking Basics
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Example

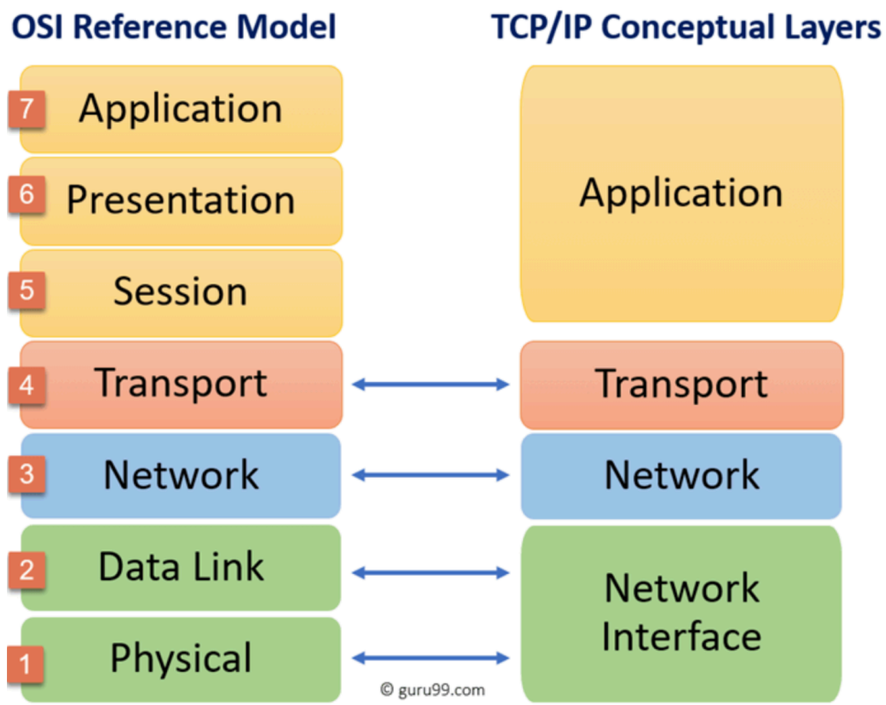
Retour au plan - Retour à l'accueil

A SOCKET

- At the core of **Java**'s networking support is the concept of a **socket**.
  - A **socket** identifies an endpoint in a network.
- **Socket** allows a single computer to serve many different clients at once, as well as to serve many different types of information.
- This is accomplished through the use of a port, which is a numbered socket on a particular machine.
  - A server process is said to listen to a port until a client connects to it.
- A server is allowed to accept multiple clients connected to the same port number, although each session is unique.
  - To manage multiple client connections, a server process must be multithreaded.

IP AND TCP PROTOCOLS

Socket communication takes place via a protocol.



IP AND TCP PROTOCOLS

- Internet Protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network.
- Transmission Control Protocol (TCP) is a higher-level protocol that manages to robustly string together these packets, sorting and retransmitting them as necessary to reliably transmit data.

AN INTERNET ADDRESS

- A key component of the Internet is the address. An Internet address is a number that uniquely identifies each computer on the Net.
- All Internet addresses consisted of 32-bit values, organized as four 8-bit values (**IPv4**, Internet Protocol version 4).
  - `192.54.12.68`
- A new addressing scheme, called **IPv6** has come into play (128-bit values organized into eight 16-bit chunks).
  - `2001:0db8:0000:85a3:0000:0000:ac1f:8001`

THE NAME OF AN INTERNET ADDRESS

- The name of an Internet address, called its domain name, describes a machine's location in a name space.
- For example, www.HerbSchildt.com is in the COM top-level domain (reserved for U.S. commercial sites);
  - it is called HerbSchildt, and www identifies the server for web requests.
- An Internet domain name is mapped to an IP address by the Domain Naming Service (**DNS**).
- This enables users to work with domain names, but the Internet operates on IP addresses.

PORT NUMBER

- Once a connection has been established, a higher-level protocol ensues, which is dependent on which port you are using.
- TCP/IP reserves the lower 1,024 ports for specific protocols :
  - 21 is for FTP; 23 is for Telnet; 25 is for e-mail; 80 is for HTTP; ...
- It is up to each protocol to determine how a client should interact with the port.
- For example, HTTP is the protocol that web browsers and servers use to transfer hypertext pages and images.

NETWORK PROGRAMMING

- TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet.
- A socket can be used to connect **Java's I/O system** to other programs that may reside either on the local machine or on any other machine on the Internet.

JAVA AND NETWORK PROGRAMMING

- One of the most important reasons that **Java** is the premier language for network programming are the classes defined in the `java.net` package.
- There are two kinds of TCP sockets in **Java**.
  1. The `ServerSocket` class is designed to be a "listener," which waits for clients to connect before doing anything.
     - ➠ Thus, `ServerSocket` is for servers.
  2. The `Socket` class is for clients. It is designed to connect to server sockets and initiate protocol exchanges.

PLAN

- Networking Basics
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Example

Retour au plan - Retour à l'accueil

TCP/IP CLIENT SOCKET

- The creation of a `Socket` object implicitly establishes a connection between the client and server.
- Here are two constructors used to create client sockets:

`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

- A `Socket` can be examined for the address and port information associated with it, by use of the following methods:

`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

- You can access to the input and output streams associated with a `Socket`.

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

EXAMPLE

```
1  public class Whois {
2      public static void main(String[] args) throws IOException {
3          int c;
4          // Create a socket connected to internic.net, port 43.
5          Socket s = new Socket("whois.internic.net", 43);
6          // Obtain input and output streams.
7          InputStream in = s.getInputStream();
8          OutputStream out = s.getOutputStream();
9          // Construct a request string.
10         String str = (args.length == 0 ? "MHProfessional.com" : args[0]) + "\n";
11         byte buf[] = str.getBytes();
12         // Send request.
13         out.write(buf);
14         // Read and display response.
15         while ((c = in.read()) != -1) {
16             System.out.print((char) c);
17         }
18         s.close();
19     }
20 }
```

PLAN

- Networking Basics
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Example

Retour au plan - Retour à l'accueil

TCP/IP SERVER SOCKETS

- The `ServerSocket` class is used to create servers that listen for either local or remote client programs to connect to them on published ports.
- When you create a `ServerSocket`, it will register itself with the system as having an interest in client connections.

`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`.

EXAMPLE

```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();
    }
}
```

EXAMPLE

```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

- Networking Basics
- TCP/IP Client Sockets
- TCP/IP Server Sockets
- Example

Retour au plan - Retour à l'accueil

EXAMPLE

```
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.ge
11             PrintWriter writeData = new PrintWriter(new OutputStreamWriter(this.s.getOut
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;
```

EXAMPLE

```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();}
    }
}
```

EXAMPLE

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

            PrintWriter writeData = new PrintWriter(new OutputStreamWriter(socket.getOut
            BufferedReader buf = new BufferedReader(new InputStreamReader(socket.getInpu

            Scanner sc = new Scanner(System.in);

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

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

                System.out.print(v1 + " " + op + " " + v2 + " = ");
```

MERCI

Retour à l'accueil - Retour au plan