Sockets

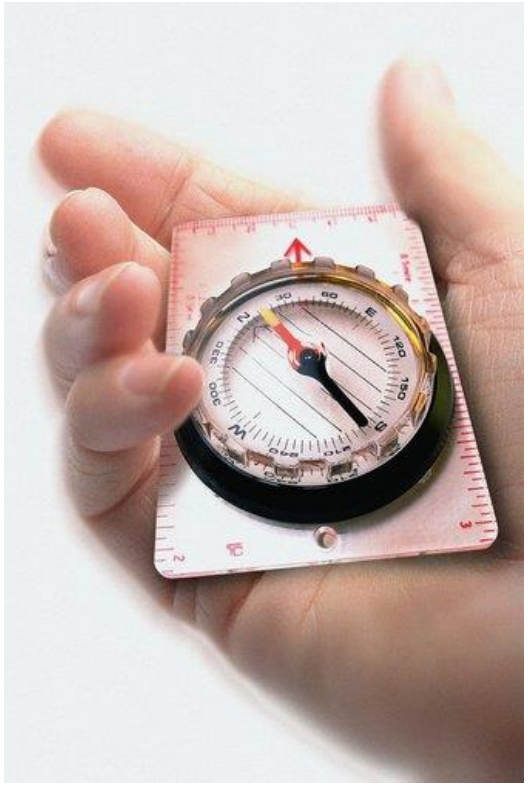
Application Client/Server

Course objectives

En complétant ce cours, vous serez en mesure de:

- Expliquer ce que sont les sockets
- Les utiliser
- Créer une application client/serveur

Plan du cours



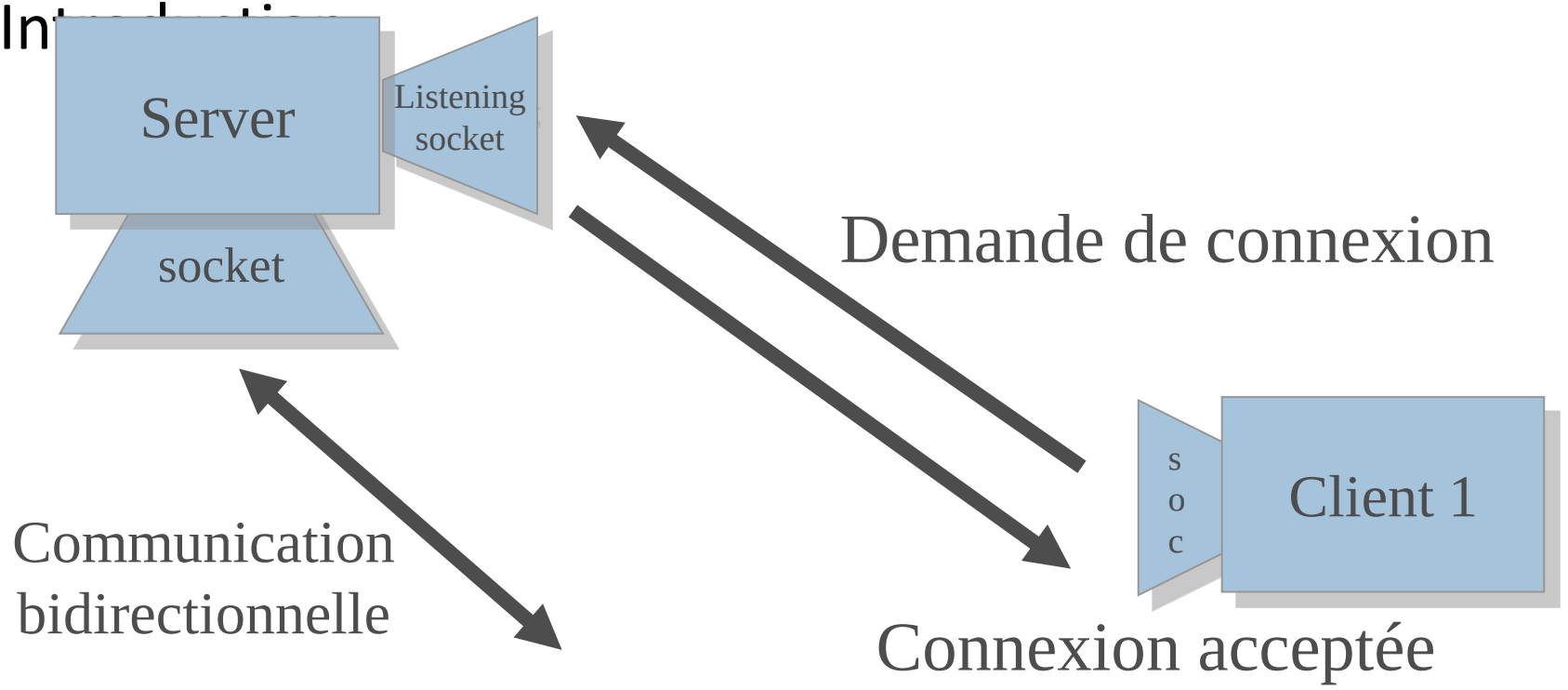
- Introduction
- Classe InetAddress
- Classe Socket
- Classe ServerSocket

Sockets

INTRODUCTION

Client/Server fonctionnement

- Introduction



Le paquet java.net

- Peut gérer les opérations réseau :
  - TCP (protocole de contrôle de transmission)
  - UDP (protocole de datagramme utilisateur)
- Classes standards :
  - InetAddress
  - Socket
  - ServeurSocket

Exemple concret

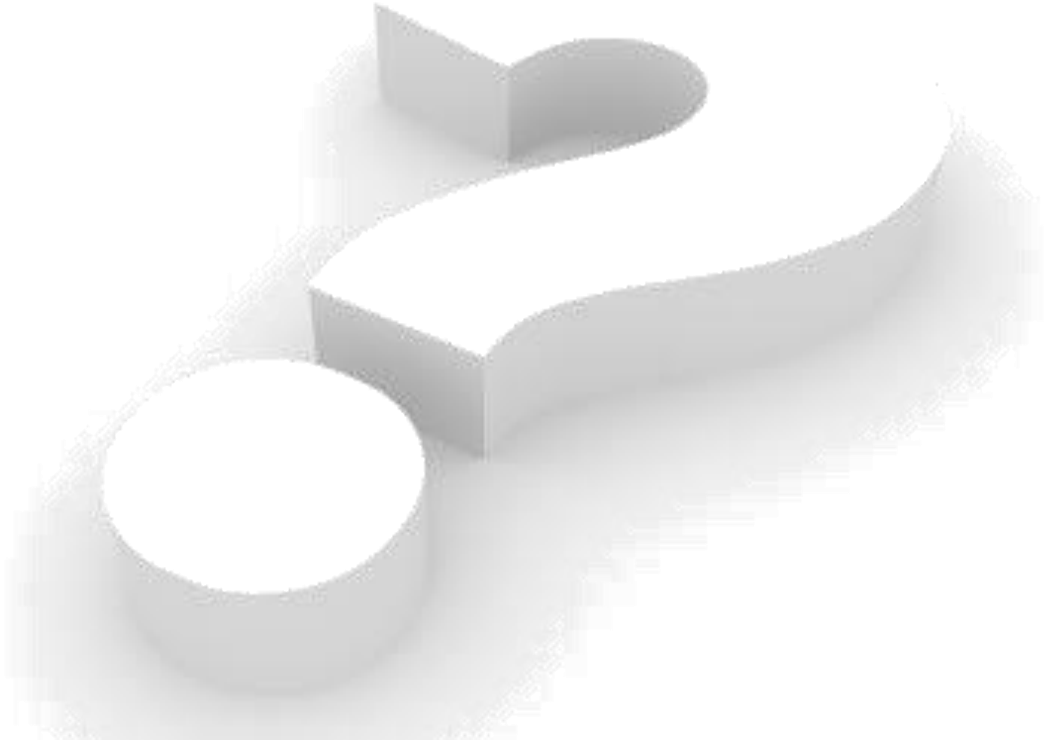
- Nous utilisons des sockets :
  - Lorsqu'un client souhaite télécharger un fichier depuis un serveur
  - Lorsqu'un utilisateur se connecte à une messagerie instantanée

Définition

- Une socket est le point de communication par lequel un thread
  - Peut émettre ou recevoir des informations
  - Permet la communication entre deux applications aussi bien que sur une même machine qu'à travers un réseau **TCP/IP**

- Il s'agit d'un modèle permettant la communication inter processus

Questions ?



Sockets

CLASSE INETADDRESS

La classe InetAddress

- Avec la Classe **InetAddress** vous pouvez :
  - Utiliser le nom de l'ordinateur au lieu de l'adresse IP
  - Obtenir le nom de l'ordinateur avec une adresse IP en utilisant Reverse DNS
  - Utiliser les services DNS et NIS pour la résolution de noms de domaine

- Les services de résolution de noms ont un cache qui stocke les résolutions de noms réussies et non réussies

- Cette classe a un constructeur caché

Obtenir un objet InetAddress

- Vous obtenez une InetAddress en appelant des méthodes statiques

- Les méthodes statiques sont :
  - InetAddress getLocalHost()
  - InetAddress getByAddress(byte[] addr)
  - InetAddress getByName(String hostName)

- Ces méthodes lèvent UnknownHostException

Méthodes

- Il existe des méthodes d'instance pour obtenir des informations sur InetAddress :
  - String getHostName()
  - byte[] getAddress()
  - boolean isLoopbackAddress()

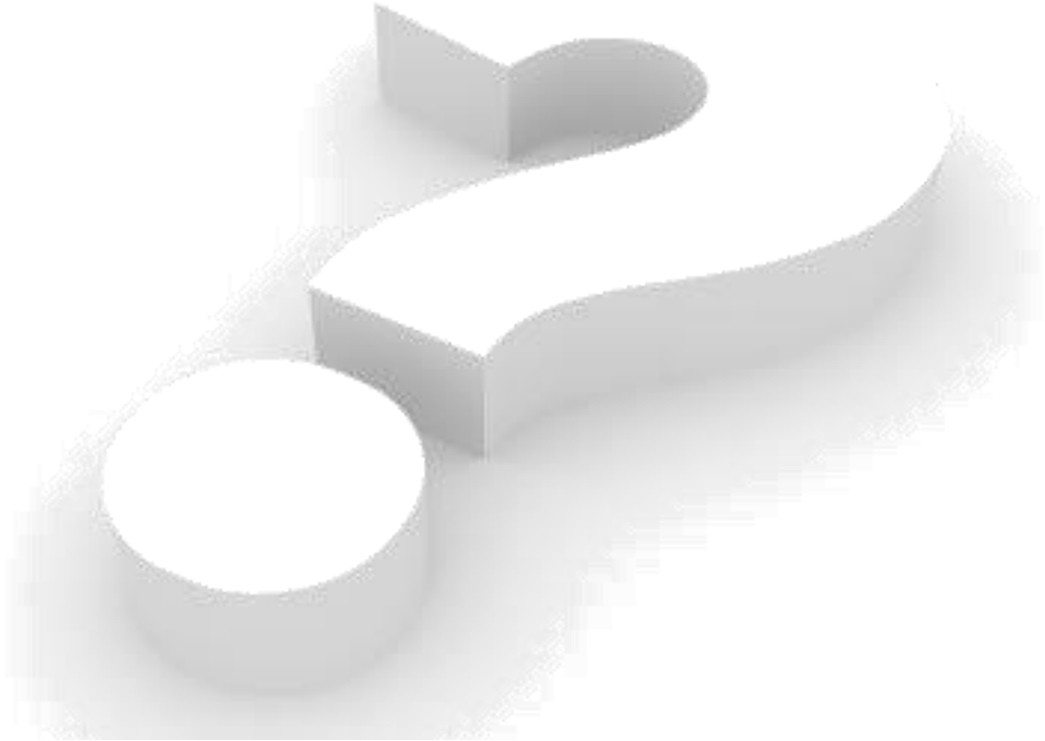
Exemple

```
try {
  //Get an InetAddress by a HostName
  InetAddress inet = InetAddress.getByName("localhost");
  // Display the Inet Address : IP, Name and Loop
  System.out.println("IP : " + inet.getHostAddress());
  System.out.println("Name : " + inet.getHostName());
  System.out.println("Loop : "+inet.isLoopbackAddress());
} catch (UnknownHostException e)  {
  // If an exception happens ...
  e.printStackTrace();
}
```

Exemple

```
try {
   //Get an InetAddress by a address
InetAddress inet = InetAddress.getByAddress(new byte[]{
          (byte) 10, (byte) 20, (byte) 72, (byte) 5});
   // Display the Inet Address : IP, Name and Loop
   System.out.println("IP : " + inet.getHostAddress());
   System.out.println("Name : " + inet.getHostName());
   System.out.println("Loop :
"+inet.isLoopbackAddress());
} catch (UnknownHostException e) {
// If an exception happens ...
e.printStackTrace();
}
```

Questions ?



Sockets

CLASSE SOCKET

Constructeurs

- De nombreux constructeurs pour créer un Socket :
  - Socket()
  - Socket(InetAddress addr, int port) throws IOException
  - Socket(String host, int port) throws IOException, UnknownHostException

Méthodes

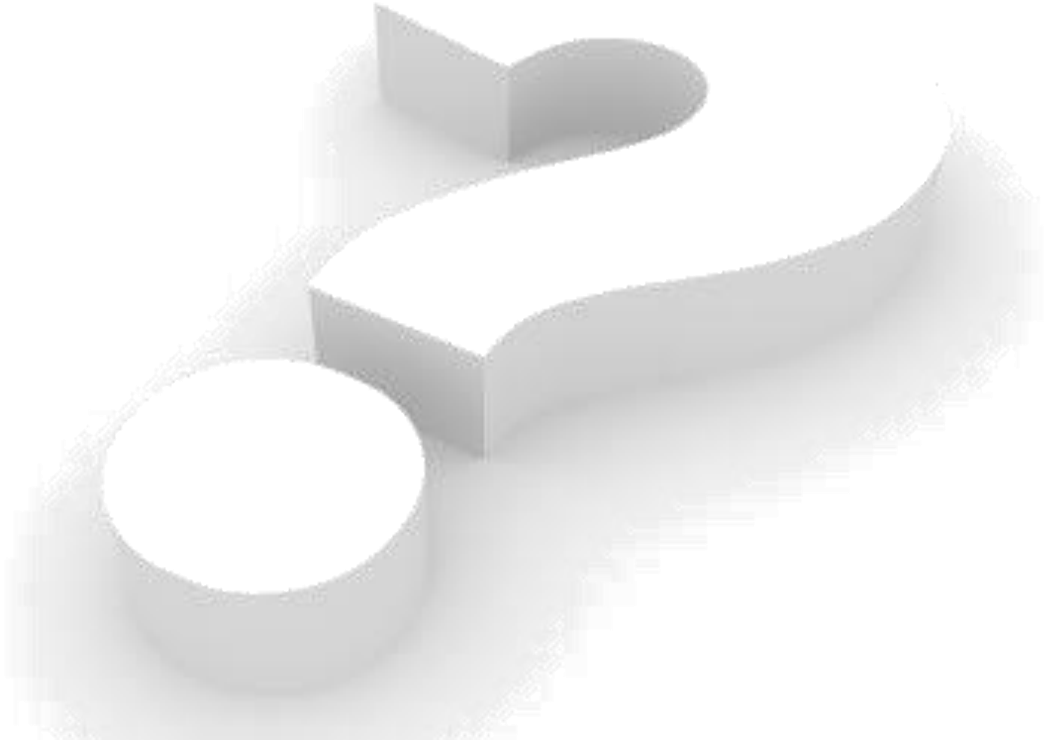
- InputStream getInputStream() :
  - Le InputStream contient ce que le socket reçoit
- OutputStream getOutputStream() :
  - Le OutputStream contient ce que le socket envoie
- InetAddress getInetAddress() :
  - L'adresse à laquelle le socket est lié
- int getPort() :
  - Le port sur lequel le socket est connecté
- void close() :
  - Fermer le socket

Exemple

- Le socket écrit ce qu'il reçoit dans un fichier :

```
Socket s = new Socket("localhost", 18000);
OutputStream out = new FileOutputStream ("file");
InputStream in = s.getInputStream();
byte[] buffer = new byte[256];
// While there is bytes to read
while (in.read (buffer) != -1) {
out.write (buffer);
}
// Close the socket
s.close();
out.flush();
out.close();
```

Questions ?



Sockets

CLASSE SERVERSOCKET

La Classe ServerSocket

- Crée un thread d'écoute des connexions clients

- Lorsqu'un client se connecte, un Socket est retourné :
  - Il représente la connexion client/serveur

Constructeurs

- ServerSocket()
- ServeurSocket(int port) :
  - Le port que le ServerSocket écoutera
- ServerSocket(int port, int backlog) :
  - Spécifie en outre la longueur de la file d'attente

- ServerSocket (int port, int backlog, InetAddress bindAddr)

Méthodes

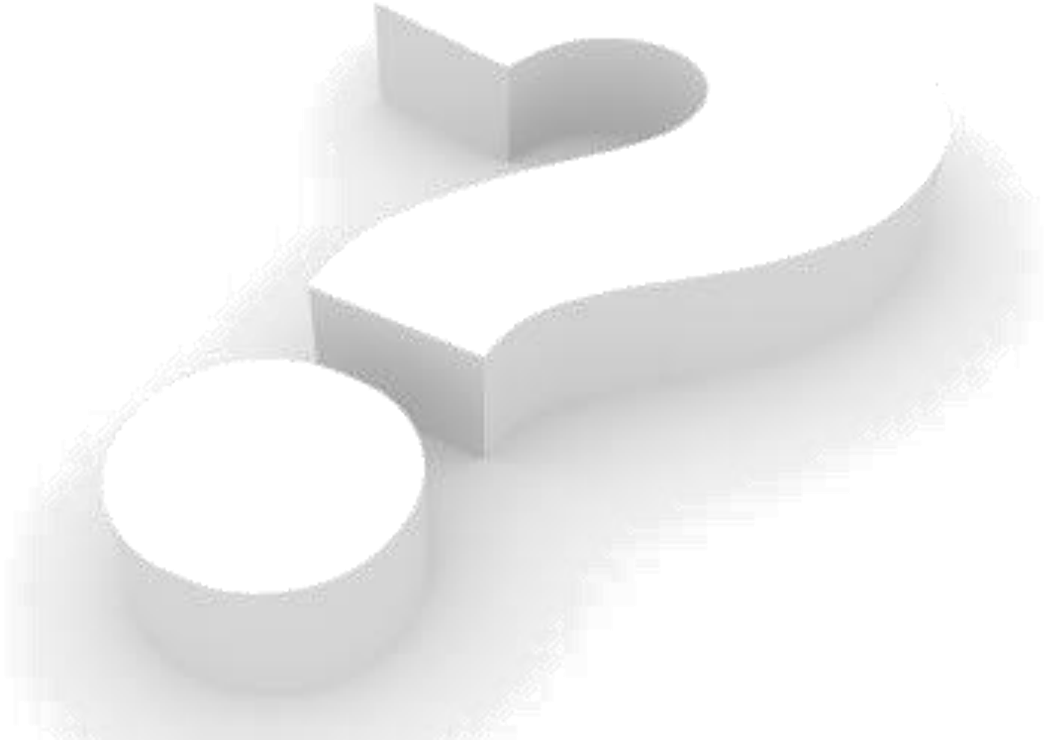
- Socket accept() throws IOException :
  - Accepter la connexion du client
- boolean isBound() :
  - Renvoie si le ServerSocket est lié avec succès à une adresse
- boolean isClosed() :
  - Renvoie si le ServetSocket a été fermé
- InetAddress getInetAddress() :
  - Renvoie l'adresse locale de ce ServerSocket
- void close() :
  - Ferme le ServerSocket

Exemple

- Le serveur enverra le contenu d'un fichier à un client :

```
// Listening Port : 18000 || Max connection queue: 5
ServerSocket listen = new ServerSocket(18000, 5);
Socket service;
while (true) {
// Ready to accept client connection
service = listen.accept();
OutputStream out = service.getOutputStream();
InputStream in = new FileInputStream ("file");
byte[] buffer = new byte[256];
// While there is byte to read
while (in.read (buffer) != -1) {
    out.write (buffer);
}
/* Close all the streams and the socket */
```

Questions ?



Exercises (1/3)

- Créez un nouveau projet Java et nommez-le BinaryConverter
- Créez un package com.cci.binaryconverter.server :
  - Créez une classe **ServerLauncher** :
    - Avec une méthode **main** qui contient un ServerSocket qui écoute les tentatives de connexion et crée de nouveaux Threads pour gérer chacune d'elles
  - Créez une classe **BinaryConverterService** implémentant **Runnable** :
    - Cette classe exécutable doit récupérer les données envoyées par le socket qu'elle gère
    - Convertir en chaîne binaire :
      - Utilisez Integer. toBinaryString(int)
    - Renvoyer la nouvelle chaîne au client à l'aide du Socket

Exercises (2/3)

- Créer un package com.cci.binaryconverter.client
  - Créez une classe CreateLauncher :
    - Avec une méthode principale (main) qui :
      - Demande à l'utilisateur d'entrer la phrase qu'il veut traduire en binaire
      - Créer un Socket et envoie la phrase au serveur
      - Affiche le résultat envoyé par le serveur

Exercises (3/3)

- Vous pouvez essayer de connecter votre client avec le serveur d'un de vos voisins
- Essayez d'exécuter un grand nombre de requêtes simultanées sur le même serveur
  - Que ce passe-t-il ? Pourquoi ?
    - Vous pouvez utiliser VisualVM pour vous aider
- Regardez le Javadoc de la classe ThreadPoolExecutor :
  - Expliquez pourquoi cette classe peut vous aider à résoudre le problème
  - Refactorisez votre serveur pour utiliser cette classe

Fin

Merci de votre attention