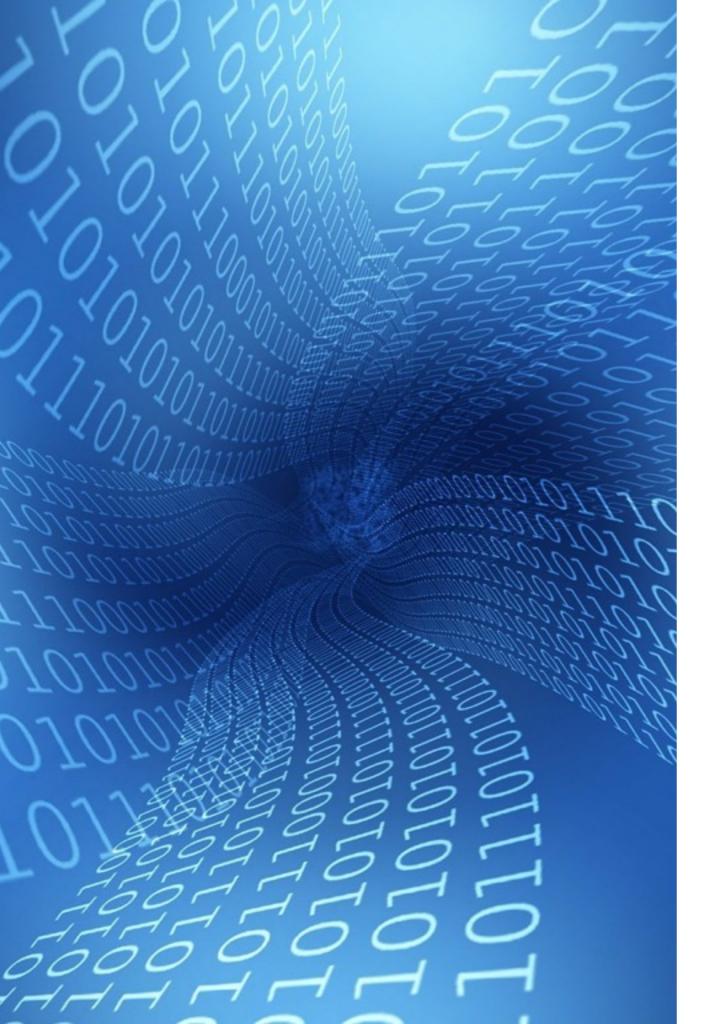


# TECHNIQUES DE PROGRAMMATION RÉSEAU

7 7 7

Hiver 2017 - Cours 6



# PLAN DE LA LEÇON

- + Retour sur l'examen sur les sockets
- + JavaMail
- + Exercices:
  - + envoi de courriels avec JavaMail
  - + gestion des utilisateurs du projet

#### LES SOCKETS: RETOUR SUR L'EXAMEN 1

- + Question 1 : Les sockets : côté serveur
  - (a) Lancement du serveur et attente de connexion client
    - + Ouverture d'un port serveur (ServerSocket)
    - + Lancement d'un thread pour l'attente de connexion client
  - (b) Connexion d'un client
    - + Ouverture d'un socket entre le client et le serveur (Socket)
    - + Création d'un buffer d'écriture pour parler au client
    - + Création d'un buffer de lecture pour écouter le client
    - + Si c'est le premier client :
      - + Lancement d'un thread pour écouter les communications des clients (paquets / datagrammes)

#### LES SOCKETS: RETOUR SUR L'EXAMEN 1

- + Question 2 : Les sockets : côté client
  - (a) Lancement du client et attente de paquets du serveur
    - + Ouverture d'un port client (Socket)
    - + Création d'un buffer d'écriture pour parler au serveur
    - + Création d'un buffer de lecture pour recevoir les paquets du serveur
    - + Lancement d'un thread (ou d'une boucle) qui vérifie la réception de messages (paquets / datagrammes) du serveur

## LES SOCKETS : CÔTÉ SERVEUR

+ Convertisseur de mesures

```
VerifierServeur

- ref:Serveur

+ VerifierServeur(Serveur cs)

+ run()
```

Thread de lecture

```
VerifierConnexion
- ref:Serveur
+ VerifierConnexion(Serveur cs)
+ run()
```

Thread de connexion

```
Serveur
 serveurSock:ServerSocket
 port:int
 maxConnexions:int
 connexions:Socket[maxConnexions]
 nbConnexions:int
os:PrintWriter[maxConnexions]
 is:BufferedInputStream[maxConnexions]
- vt:VerifierServeur
+ Serveur(int port)
+ Serveur()
+ getPort():int
+ setPort(int port)
 envoyer(String msg, int i)
 connecter()
 deconnecterClient(int client)
 traiterCommande(String commande,
                  double mesure)
+ attente()
+ lire()
+ main(String args[])
```

# ConvertisseurMesure

- + metresEnPieds(double
   metres):double
  + piedsEnMetres(douple
   pieds):double
  + poucesEnCentimetres(double
- + poucesEnCentimetres(double pouce):double
- + centimetresEnPouces(double centimetres):double
- + millimetresEnPouces(double millimetres):double
- + poucesEnMillimetres(double pouces):double
- + poucesEnPieds(double
   pouces):double
- + piedsEnPouces(double

## LES SOCKETS : CÔTÉ CLIENT

+ Client convertisseur de mesures

```
Client
- clientSock:Socket
- os:PrintWriter
- is:BufferedInputStream
- adr:String
- port:int
+ Client(String adr, int port)
+ envoyer(String message)
+ connecter():boolean
+ deconnecter():boolean
+ lire()
```

### LES SOCKETS : CÔTÉ SERVEUR

+ Client et serveur du convertisseur de mesures

```
is[num] = new BufferedInputStream(
                                                                                        connexions[num].getInputStream());
os = new PrintWriter(clientSock.getOutputStream(), true);
                                                                                    is[i].read(buf);
os.print(message);
                                                                                    texte = (new String(buf)).trim();
os.flush();
                                     clientSock = new Socket(127.0.0.1, 5555);
                                    connexions[num] = Socket sk = server.accept();
                                                  commande mesure
                                                      résultat
                                                     m2ft 1,83
                                                                                                     Serveur
      Client
                                                       6,00 ft
                                                                                              TimeServer\ ts =
                                                                                                 new TimeServer(host, port);
                                                                                              ts.open();
                                                        STOP
is = new BufferedInputStream(clientSock.getInputStream());
                                                                                   os[num] = new PrintWriter(
is.read(buf);
                                                                                      connexions[num].getOutputStream());
msg = (new String(buf)).trim();
                                                                                   os[i].print(msg);
```

os[i].flush();

#### API CLIENT-SERVEUR : JAVAMAIL

+ Théorie:

https://www.jmdoudoux.fr/java/dej/chap-javamail.htm
http://www.commentcamarche.net/contents/536-pop3-smtp-imap-protocoles-de-messagerie

- + Exercice:
  - + L'envoi de courriel avec JavaMail : <a href="http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/javamail/javamail.html">http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/javamail/javamail.html</a>
  - + Intercepteur d'envoi de courriel (FakeSMTP) : <a href="http://nilhcem.com/FakeSMTP/">http://nilhcem.com/FakeSMTP/</a>
- + Projet étape 2 :
  - + Vous devrez ajouter un système d'inscription et de gestion des utilisateurs. Le système devra :
    - + envoyer un courriel avec un lien de validation lors de l'inscription d'un nouvel utilisateur;
    - + permettre la réinitialisation de l'identifiant et du mot de passe par l'envoi d'un nouveau lien de validation à l'adresse courriel de l'utilisateur en cas d'oubli de ceux-ci ;