JAVA

Les sockets

Les sockets

- `import java.net.*;`
- Principe
- La classe Socket
- Exemple d'un client
- La classe Serveur
- Exemple d'un serveur
- Système d'adressage

Principe

- Socket = flux de caractères entre deux applications distantes
  - Pas de protocole particulier entre les applications (au développeur de le définir)
  - Pas de type de transfert particulier (forcément des caractères)
  - Une socket est composée de deux flux : un pour les entrées (InputStream), un pour les sorties(OutputStream)
- Une application doit être le serveur
  - Il est identifiée par l'adresse IP de la machine locale
  - Il « attend» des clients sur un port déterminé par le développeur
- Les autres applications sont des clients
  - Un client doit connaître l'adresse et le port du serveur
  - Le flux de sortie du client correspond au flux d'entrée du serveur et vice et versa

La classe Socket

- **Socket**(String host, int port) ou
  Socket(InetAddress address, int port)
  - Crée une socket connecté au serveur *host* sur le port *port*

- void **close**()
  - Deconnecte la socket

La classe Socket

- InetAddress **getLocalAddress**()
  - Renvoie l'adresse du serveur
- int **getLocalPort**()
  - Renvoie le port de communication
- InputStream **getInputStream**()
  - Renvoie le flux d'entrée

```
BufferedReader  in = new BufferedReader(new InputStreamReader(s.getInputStream()));
String Msg = in.readLine();
```

- OutputStream **getOutputStream**()
  - Renvoie le flux de sortie

```
PrintWriter     out= new PrintWriter(s.getOutputStream(),true);
out.println("un message");
```

La classe ServerSocket

- **ServerSocket**(int port)
- void **close**()
- Socket **accept**()
- InetAddress **getInetAddress**()
- int **getLocalPort**()

Exemple d'un serveur simple client

```
/**
 * La classe ServeurSimple attend un client (un seul). Lorsque ce dernier
 * est connecté, il écoute la requête du client. Si c'est "DATE", il renvoie
 * la date système. Si c'est "HEURE", il renvoie l'heure...
 */
class ServeurSimple
{
   /**
    * Numéro de port arbitraire, mais ne correspondant à aucun service
    * standard.
    */
   final   int     Port    = 4567;

   public ServeurSimple()
   {
       try
       {
           /**
            * Création du serveur
            */
           ServerSocket    serveur = new ServerSocket(Port);
           System.out.println("Serveur opérationnel...");

           /**
            * On attend le client...
            */
           Socket  client = serveur.accept();

           /**
            * On crée les flux d'entrée et de sortie
            */
           BufferedReader  in = new BufferedReader(new InputStreamReader(client
           PrintWriter     out= new PrintWriter(client.getOutputStream(),true);
```

Exemple d'un serveur multiple clients

```
/**
 * La classe ServeurMultiple attend indéfiniment des clients. Lorsqu'un
 * client se connecte, un thread lui est consacré.
 */
class ServeurMultiple
{
   /**
    * Numéro de port arbitraire, mais ne correspondant à aucun service
    * standard.
    */
   final   int     Port    = 4567;

   public ServeurMultiple()
   {
       try
       {
           /**
            * Création du serveur
            */
           ServerSocket    serveur = new ServerSocket(Port);
           System.out.println("Serveur opérationnel...");

           /**
            * On attend les clients...
            */
           while (true)
           {
               Socket  client = serveur.accept();
               new ClientDuServeurMultiple(client);
           }
       }
```

Exemple d'un client (ou telnet)

```
class ClientSimple
{
   final   int     Port    = 4567;
   final   String  Host    = "localhost";

   public ClientSimple()
   {
       try
       {
           /**
            * Ouverture de la socket
            */
           Socket  s = new Socket(Host,Port);

           /**
            * Création des flux d'entrées/sorties
            */

           BufferedReader  in = new BufferedReader(new InputStreamReader(s.get
           PrintWriter     out= new PrintWriter(s.getOutputStream(),true);

           /**
            * Lecture des messages d'aides (5 lignes)
            */
           String  msg;
           msg = in.readLine();    System.out.println(msg);
           msg = in.readLine();    System.out.println(msg);
           msg = in.readLine();    System.out.println(msg);
           msg = in.readLine();    System.out.println(msg);
           msg = in.readLine();    System.out.println(msg);
```

Système d'adressage : InetAdress

- String **getHostAddress**()
  - Renvoie l'adresse IP au format « %d.%d.%d.%d »
- String **getHostName**()
  - Renvoie simplement le nom de la machine locale
- static InetAddress **getByName**(String host)
  - Sert de constructeur…
- static InetAddress **getLocalHost**()
  - Renvoie l'InetAdress de la machine locale