T5 Programmation Réseau

Le but de ce TP est d'apprendre les routines essentielles pour créer des applications réseau:

Chaque machine sur Internet est identifiée par une adresse ou un nom unique. Ces deux entités sont gérées sous Java par la classe InetAddress dont voici quelques méthodes :

- String getHostAddress() donne l'adresse IP de l'instance InetAddress courante
- String getHostName() donne le nom Internet de l'instance InetAddress courante
- String toString() donne l'identité adresse IP/ nom internet de l'instance InetAddress courante
- InetAddress getByName(String Host) crée l'instance InetAddress de la machine désignée par Host. Génère une exception si Host est inconnu. Host peut être le nom internet d'une machine ou son adresse IP sous la forme

1. Créer un nouveau programme java (sock.java) dont le code est le suivant :

```
import java.net.*; //package réseau
public class sock{ //classe sock
public static void main (String arg[]){
try{
InetAddress adresse=InetAddress.getLocalHost();
System.out.println("adresse="+adresse.toString());
System.out.println("adresse="+adresse.getHostAddress());
System.out.println("nom="+adresse.getHostName());
System.out.println("identite="+adresse);
System.out.println("nom:"+adresse.getByName("mail.google.com"));
} catch (UnknownHostException e){
System.out.println ("Erreur getLocalHost : "+e);
}
}
}
```

2. Compiler et exécuter le programme

   javac –d . sock.java

   java –cp sock

3. Interpréter les résultats.

L'outil de base utilisé par les programmes communiquant sur Internet est la *socket*. Pour qu'une application puisse envoyer et recevoir des informations sur le réseau Internet, il lui faut une prise de réseau « *socket* ». Dans la suite on vous demande de créer 2 programmes java : client.java et serveur.java, dont voici les codes :

Client.java

```
import java.net.*;
import java.io.*;
public class client{
public static void main (String arg[]){
String machine="localhost";
InetAddress serveurAddress=null;
try{
serveurAddress=InetAddress.getByName(machine);
} catch (Exception e){
erreur("Machine "+machine+" inaccessible (" + e +")",2);
}
int port=1050;
Socket sClient=null;
try{
sClient=new Socket(machine,port);
} catch (Exception e){
erreur("Erreur lors de la création de la socket de communication ("+e+")",4);
}
try{
System.out.println("Client : Client ["+identifie(InetAddress.getLocalHost())+","+
sClient.getLocalPort()+"] connecté au serveur [" + identifie (sClient.getInetAddress()) + "," +
sClient.getPort() + "]");
} catch (Exception e) {
erreur("identification liaison ("+e+")",5);
}
// création du flux de lecture des lignes tapées au clavier
BufferedReader IN=null;
try{
IN=new BufferedReader(new InputStreamReader(System.in));
} catch (Exception e){
erreur("Création du flux d'entrée clavier ("+e+")",6);
}
// création du flux d'entrée associée à la socket client
BufferedReader in=null;
try{
in=new BufferedReader(new InputStreamReader(sClient.getInputStream()));
} catch (Exception e){
erreur("Création du flux d'entrée de la socket client("+e+")",7);
}
// création du flux de sortie associée à la socket client
PrintWriter out=null;
try{
out=new PrintWriter(sClient.getOutputStream(),true);
} catch (Exception e){
erreur("Création du flux de sortie de la socket ("+e+")",8);
}
// boucle demandes - réponses
boolean serviceFini=false;
String demande=null;
String reponse=null;
// on lit le message envoyé par le serveur juste après la connexion
try{
```

```
reponse=in.readLine();
} catch (IOException e){
erreur("Lecture réponse ("+e+")",4);
}
// affichage réponse
System.out.println("Serveur : " +reponse);
while (! serviceFini){
// lecture d'une ligne tapée au clavier
System.out.print("Client : ");
try{
demande=IN.readLine();
} catch (Exception e){
erreur("Lecture ligne ("+e+")",9);
}
// envoi demande sur le réseau
try{
out.println(demande);
} catch (Exception e){
erreur("Envoi demande ("+e+")",10);
}
try{
reponse=in.readLine();
} catch (IOException e){
erreur("Lecture réponse ("+e+")",4);
}
// affichage réponse
System.out.println("Serveur : " +reponse);
// est-ce fini ?
if(demande.trim().toLowerCase().equals("fin")) serviceFini=true;
}
try{
sClient.close();
} catch(Exception e){
erreur("Fermeture socket ("+e+")",11);
}
}// main
// affichage des erreurs
public static void erreur(String msg, int exitCode){
System.err.println(msg);
System.exit(exitCode);
}
private static String identifie(InetAddress Host){
String ipHost=Host.getHostAddress();
String nomHost=Host.getHostName();
String idHost;
if (nomHost == null) idHost=ipHost;
else idHost=ipHost+","+nomHost;
return idHost;
}
}// fin class
```

Serveur.java

```
import java.net.*;
import java.io.*;
public class serveur{
public final static int nbConnexions=2;
public static void main (String arg[]){
int port=1050;
// on crée la socket d'écoute
ServerSocket ecoute=null;
```

```
try{
ecoute=new ServerSocket(port,nbConnexions);
} catch (Exception e){
erreur("Erreur lors de la création de la socket d'écoute ("+e+")",3);
}
// suivi
System.out.println("Serveur d'écho lancé sur le port " + port);
// boucle de service
boolean serviceFini=false;
Socket service=null;
while (! serviceFini){
// attente d'un client
try{
service=ecoute.accept();
} catch (IOException e){
erreur("Erreur lors de l'acceptation d'une connexion ("+e+")",4);
}
// on identifie la liaison
try{
System.out.println("Client ["+identifie(service.getInetAddress())+","+
service.getPort()+"] connecté au serveur [" + identifie (InetAddress.getLocalHost())
+ "," + service.getLocalPort() + "]");
} catch (Exception e) {
erreur("identification liaison",1);
}
// le service est assuré par une autre tâche
new traiteClientEcho(service).start();
}// fin while
}// fin main
// affichage des erreurs
public static void erreur(String msg, int exitCode){
System.err.println(msg);
System.exit(exitCode);
}
// identifie
private static String identifie(InetAddress Host){
// identification de Host
String ipHost=Host.getHostAddress();
String nomHost=Host.getHostName();
String idHost;
if (nomHost == null) idHost=ipHost;
else idHost=ipHost+","+nomHost;
return idHost;
}
}// fin class
// assure le service à un client du serveur d'écho
class traiteClientEcho extends Thread{
private Socket service; // socket de service
private BufferedReader in; // flux d'entrée
private PrintWriter out; // flux de sortie
// constructeur
public traiteClientEcho(Socket service){
this.service=service;
}
// méthode run
public void run(){
// création des flux d'entrée et de sortie
try{
in=new BufferedReader(new InputStreamReader(service.getInputStream()));
} catch (IOException e){
```

```
erreur("Erreur lors de la création du flux déentrée de la socket de service ("+e+")",1);
}// fin try
try{
out=new PrintWriter(service.getOutputStream(),true);
} catch (IOException e){
erreur("Erreur lors de la création du flux de sortie de la socket de service ("+e+")",1);
}// fin try
// l'identification de la liaison est envoyée au client
try{
out.println("Client ["+identifie(service.getInetAddress())+","+
service.getPort()+"] connecté au serveur [" + identifie (InetAddress.getLocalHost())
+ "," + service.getLocalPort() + "]");
} catch (Exception e) {
erreur("identification liaison",1);
}
// boucle lecture demande/écriture réponse
String demande,reponse;
try{
// le service s'arrête lorsque le client envoie une marque de fin de fichier
while ((demande=in.readLine())!=null){
// écho de la demande
reponse="["+demande+"]";
out.println(reponse);
// le service s'arrête lorsque le client envoie "fin"
if(demande.trim().toLowerCase().equals("fin")) break;
}// fin while
} catch (IOException e){
erreur("Erreur lors des échanges client/serveur ("+e+")",3);
}// fin try
// on ferme la socket
try{
service.close();
} catch (IOException e){
erreur("Erreur lors de la fermeture de la socket de service ("+e+")",2);
}// fin try
}// fin run
// affichage des erreurs
public static void erreur(String msg, int exitCode){
System.err.println(msg);
System.exit(exitCode);
}// fin erreur
// identifie
private String identifie(InetAddress Host){
// identification de Host
String ipHost=Host.getHostAddress();
String nomHost=Host.getHostName();
String idHost;
if (nomHost == null) idHost=ipHost;
else idHost=ipHost+","+nomHost;
return idHost;
}
}// fin class
```

4. Compiler les 2 programmes, lancer d'abord le serveur, puis le client
5. Interpréter les résultats.