Annexe

Les sockets

Exemple d'une application typique comportant de la répartition à l'aide des Sockets :

Le processus client commence par émettre un message et le serveur lui répond par un écho de cette ligne en majuscule.

// Classe Serveur

```
import java.io.*;
import java.net.*;

public class Serveur {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {
      // Création d'un objet s à l'écoute du port spécifié
      ServerSocket s = new ServerSocket(port);
      System.out.println("En attente de connexion");
      Socket socClient = s.accept(); // L'acceptation d'une connexion d'un client
      System.out.println("Connexion établie");
      // On crée maintenant les flux d'entrée-sortie du Serveur
            BufferedReader entreeServeur = new BufferedReader(
                        new InputStreamReader(socClient.getInputStream())
                       );  // Un BufferedReader permet de lire par ligne.
             PrintWriter sortieServeur = new PrintWriter(
                       new BufferedWriter(
                         new OutputStreamWriter(socClient.getOutputStream())),
                       true);  // Un PrintWriter possède toutes les opérations print classiques.
      String str = entreeServeur.readLine(); // lecture du message envoyé par le client
      String strMajuscule=str.toUpperCase();
      System.out.println("Le message en majuscule est " + strMajuscule);
      // Fermeture des flux d'entrée-sortie du Serveur
      entreeServeur.close();
      sortieServeur.close();
      // Fermeture du socket client
      socClient.close();
  }
}
```

// Classe Client

```
import java.io.*;
import java.net.*;

public class Client {
  static final int port = 1200;
  public static void main(String[] args) throws Exception {

      System.out.println("Demande de connexion");
      Socket socket = new Socket("127.0.0.1", port); // construit un socket client connecté à la
                                                     // machine et le port spécifié
      System.out.println("Connexion établie");
       // On crée maintenant les flux d'entrée-sortie du Client
       BufferedReader entreeClient = new BufferedReader( new
InputStreamReader(socket.getInputStream())  ); //Un BufferedReader permet de lire par ligne.
       PrintWriter sortieClient = new PrintWriter( new BufferedWriter( new
OutputStreamWriter(socket.getOutputStream())), true); //Un PrintWriter possède toutes les
                                                       // opérations print classiques.
       String str = "bonjour";
       sortieClient.println(str); // envoi d'un message
       // Fermeture des flux d'entrée-sortie du Client
       entreeClient.close();
       sortieClient.close();
       // Fermeture du socket client
       socket.close();
  }
}
```