Implementing a Server (Java) 1. Open the Server Socket: ServerSocket server; DataOutputStream os; DataInputStream is; server = new ServerSocket( PORT ); 2. Wait for the Client Request: Socket client = server.accept(); 3. Create I/O streams for communicating to the client is = new DataInputStream( client.getInputStream() ); os = new DataOutputStream( client.getOutputStream() ); 4. Perform communication with client Receive from client: String line = is.readLine(); Send to client: os.writeBytes("Hello\n"); 5. Close sockets: client.close()

Implementing a Client (Java) 1. Create a Socket Object: client = new Socket( server, port\_id ); 2. Create I/O streams for communicating with the server. is = new DataInputStream(client.getInputStream() ); os = new DataOutputStream( client.getOutputStream() ); 3. Perform I/O or communication with the server: – Receive data from the server: String line = is.readLine(); – Send data to the server: os.writeBytes("Hello\n"); 4. Close the socket when done: client.close()

E/O

Java possède une grosse bibliothèque de classes et d'interfaces pour les entrées sorties. Cette bibliothèque se trouve dans le paquetage java.io. On l'utilise par la suite pour lire et écrire à partir de toute source ou destination de lecture ou d'écriture comme les fichiers, les URL, les sockets, etc. Une partie de l'arborescence des classes est :

Les classes de base ne permettent que de lire ou écrire des octets. D’autres classes XXXputStream (avec XXX = In ou Out) contiennent les méthodes pour lire ou écrire des type primitif de données ou des objets de classes.

File : classe modélisant un fichier de manière indépendante du système. Cette classe fournit des méthodes pour lister les répertoires, interroger des attributs de fichiers (droits en lecture, en écriture), renommer ou détruire des fichiers. FileNameFilter : interface possédant une méthode accept() (à implanter) permettant de n'accepter que certains noms de fichiers : utile pour implanter des filtres. RandomAccessFile : permet de lire ou d'écrire dans un fichier et de s'y déplacer. InputStream et OutputStream : classes abstraites qui définissent des méthodes pour lire et écrire. FileInputStream et FileOutputStream : permet de lire et écrire sur fichiers

FilterInputStream et FilterOuputStream sont des classes de base permettant de filtrer les entrées dans un InputStream ou un OutputStream. On se sert en fait de leurs classes dérivées. BufferedInputStream et BufferedOutputStream font de la bufferisation sur les E/S de fichiers. DataInputStream lit les octets (du binaire) et le renvoie dans le format indiqué (i.e. readDouble() renvoie un double). Peut être la classe la plus utilisée. DataOuputStream écrit du format binaire. PrintStream écrit sous forme ASCII ce qu'on lui passe comme du binaire (formatage). PushbackInputStream permet de remettre un byte dans l'InputStream associé. LineNumberInputStream indique et positionne le numéro de ligne dans un fichier texte

Les E/S texte en Java Des arborescences Reader et Writer ont été ajoutés en Java 1.1 pour traiter les fichiers textes en accord avec l'internationalisation pris en compte dans Java 1.1.

Manipuler les fichiers texte BufferedReader est une classe possédant la méthode readLine() ainsi que le constructeur BufferedReader(Reader). Pour lire une ligne d’un fichier texte, voici un code

Pour écrire on utilise plutôt la classe PrintWriter qui fournit les méthodes print(…), ou println(..) surchargées avec différents types (print(int), print(double), …) ainsi que le constructeur PrintWriter(Writer) et on peut écrire

Pour les chaînes de caractères, on peut aussi utiliser les méthodes : void writeUTF(String) de la classe DataOutputStream pour écrire, String readUTF() de la classe DataInputStream pour lire.