valtech_

Java et la Conception Objet

Lecture et écriture à l'aide des streams

Présentation

Java utilise des streams (flux) pour les entrées et sorties élémentaires

Ces streams sont semblables à ceux du C++

Objectifs de ce module :

- Utiliser les streams standard
- Lire les entrées du clavier
- Emballer les streams pour obtenir des fonctionnalités supplémentaires

Qu’est-ce qu’un stream ?

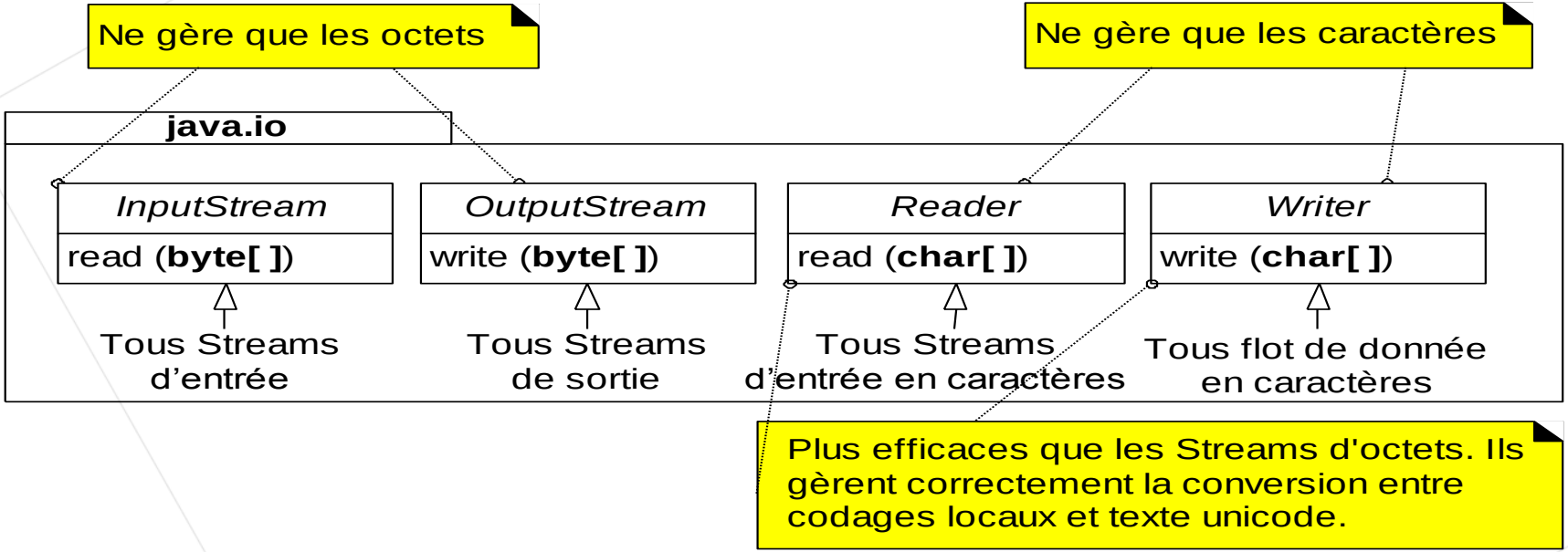
Un chemin de communication entre la source d’une information et sa destination

Trois streams standard sont disponibles

Tous sont des champs statiques publics de la classe java.lang.System

- in - InputStream pour la lecture depuis le clavier
- out - PrintStream pour l'écriture sur la console
  - System.out.println (« Ah, oui, on a déjà fait ça »)
- err - PrintStream pour l'écriture sur la console
  - System.err.println (« Certains systèmes d'exploitation permettent la redirection des erreurs »)
  - Unix C Shell prompt% java MyProgram >& error.txt
  - Toute sortie System.err est écrite sur le fichier file error.txt et toute sortie System.out est écrite sur la console

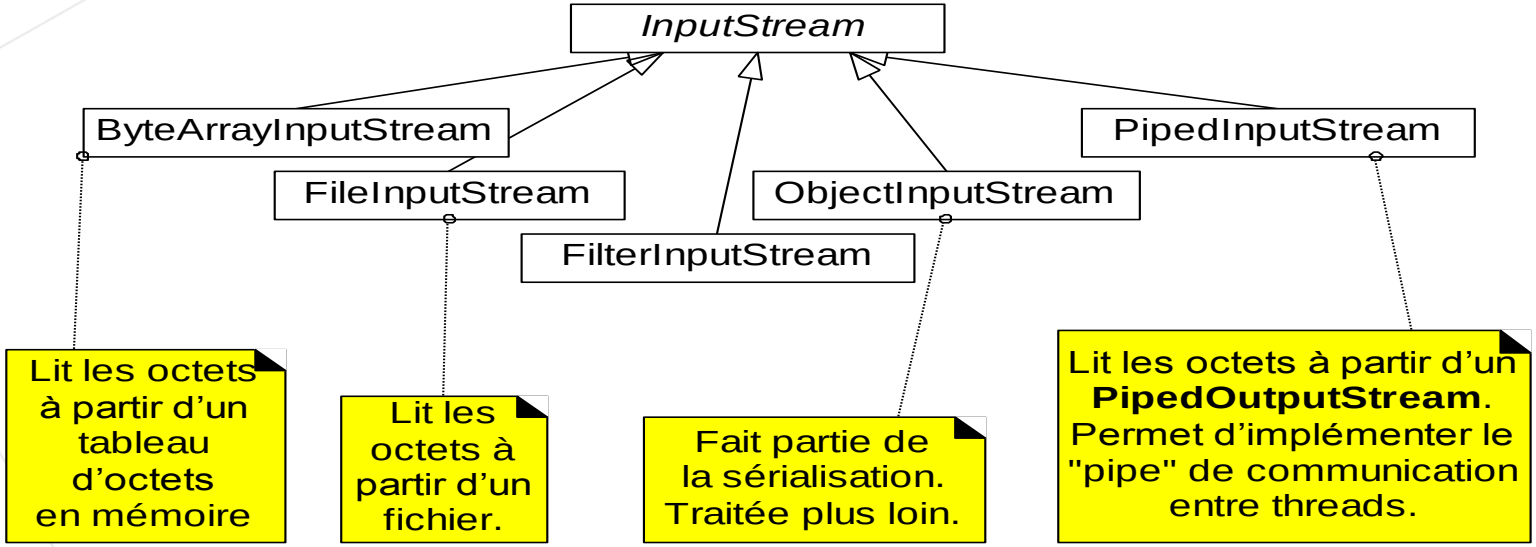
La hiérarchie java.io (simplifiée)



InputStream et Reader sont spécialisés pour lire à partir de sources

OutputStream et Writer sont spécialisés pour écrire sur des sources

Hiérarchie InputStream partielle

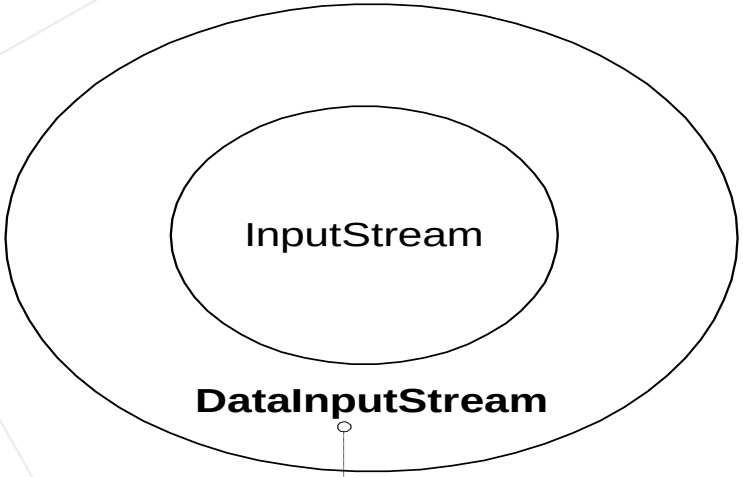


InputStream est spécialisé pour lire à partir de sources diverses

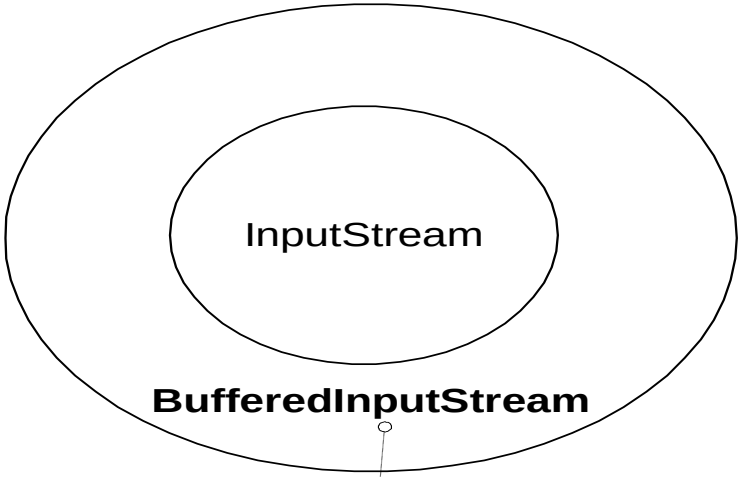
Une hiérarchie semblable existe pour OutputStream, Reader, …

Jusqu'à présent, rien ne nous permet de lire un float, int, long, double, ...

Emballage de streams

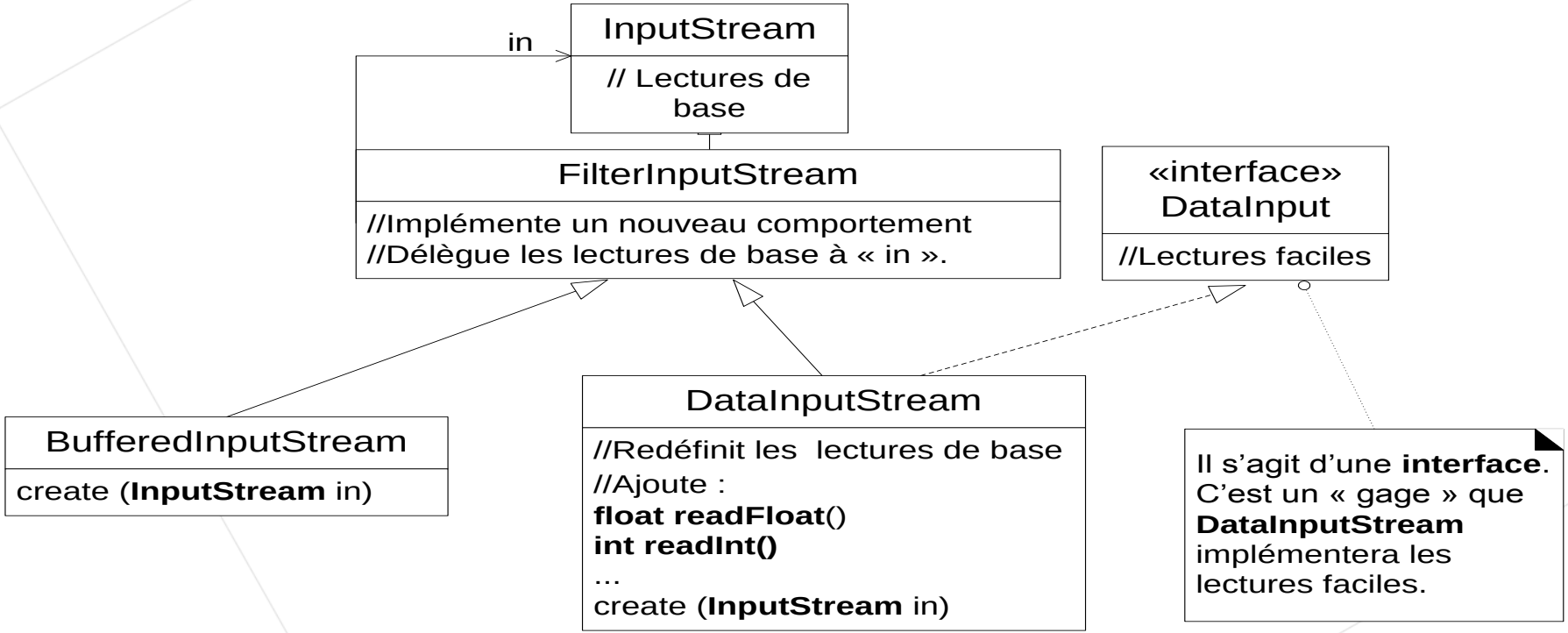


Redéfinit les opérations de lecture de base d'**InputStream** et ajoute la capacité de lire les **float**, **int**, **double**,...



Ne permet pas de lire les **float** (et autres), mais augmente l'efficacité par la lecture et la mise en mémoire tampon des données.

Les FilteredInputStreams



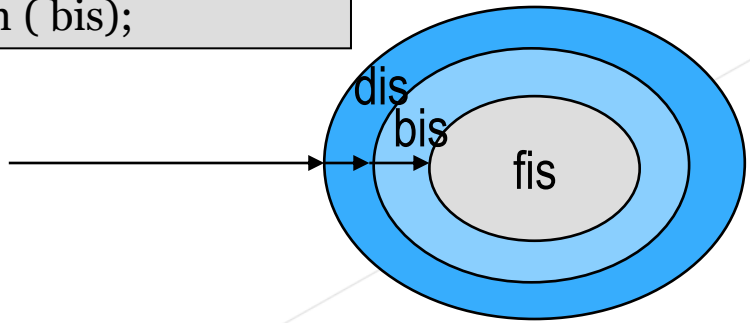
Utilisation de « wrappers »

« Emballez » des streams dans d'autres streams pour acquérir de nouvelles fonctionnalités

- Appliquez ce concept pour la communication par sockets et la sérialisation des objets

L'emballage de streams se produit généralement à la construction

```
FileInputStream fis = new FileInputStream ("file.dat");
BufferedInputStream bis = new BufferedInputStream ( fis);
DataInputStream dis = new DataInputStream ( bis);
```



Lecture/écriture de types primitifs

Comment lire un float à partir d'un fichier binaire ?

```
FileInputStream fs = new FileInputStream ("file.txt");
DataInputStream in = new DataInputStream (fs);
    float n = in.readFloat();
```

Utilisez DataInputStream avec DataOutputStream

Canaux UNICODE Readers & Writers

Flots de caractères UNICODE-2.

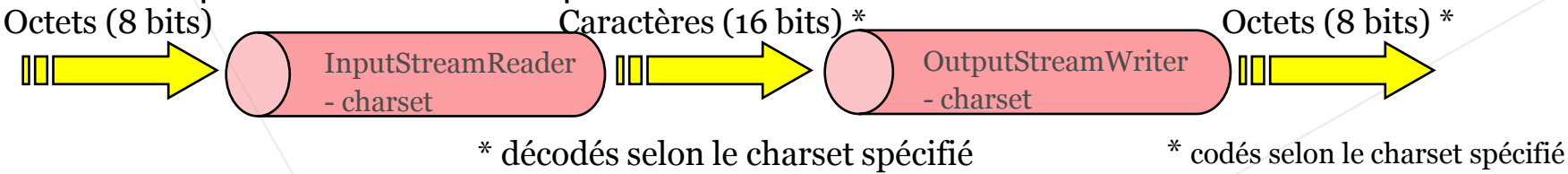
- 2 octets (16 bits) par caractère.

Deux classes de base :

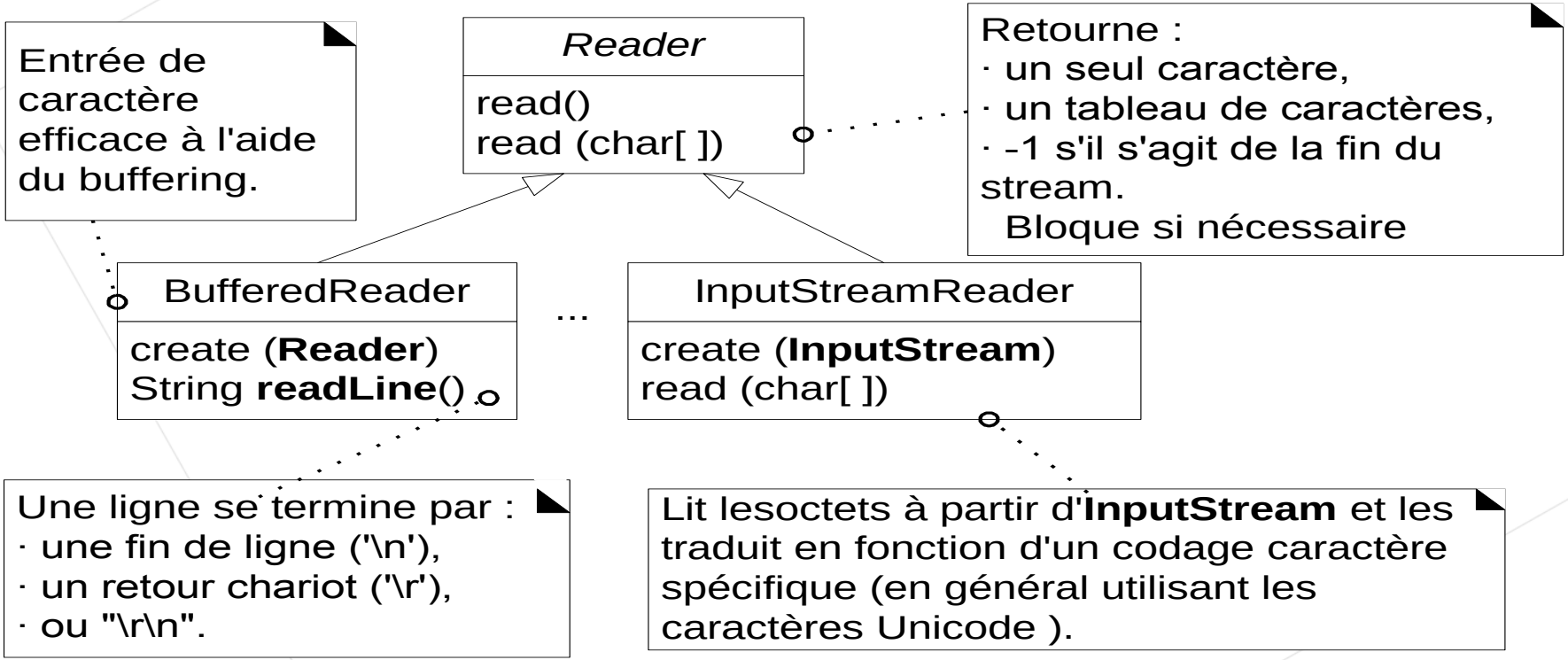
- java.io.Reader : équivalent de InputStream.
- java.io.Writer : équivalent de OutputStream.

Vous pouvez connecter un canal UNICODE sur un canal par octets.

- OutputStreamWriter et InputStreamReader.



Readers : les entrées caractères



Lecture de fichiers texte

Comment lire une ligne à partir d'un fichier texte ?

Emballez FileReader dans un BufferedReader

- BufferedReader a une méthode
  - String readLine();

```
FileReader fr = new FileReader ("file.txt");
BufferedReader br = new BufferedReader( fr );
String s = br.readLine();
```

Conversion de chaînes en nombres

Les classes d'encapsulation des types primitifs contiennent la méthode statique parse<Type>

Exemple : la classe Integer

- encapsule une valeur primitive de type int dans un objet
- dispose d’une méthode parseInt
    - public static int parseInt(String s)
    - throws NumberFormatException

Analyse (parsing) d’une chaîne

Il peut être nécessaire d’analyser la chaîne

Par exemple, un ensemble de nombres délimités par virgules

- String s = “3.1, 5.66, -87, 14.1”;

Utilisez java.util.StringTokenizer pour procéder à l’analyse

- Le constructeur prend un ensemble de délimiteurs
  - StringTokenizer tokenizer = new StringTokenizer(s, “,”);
  - String t = tokenizer.nextToken();
  - Autres…

Ou à la méthode split de la classe String

- String[] splits = line.split(",");

Étapes pour lire les entrées du clavier

Emballez System.in dans le type approprié de stream d'entrée

```
InputStreamReader rdr = new InputStreamReader(System.in);
BufferedReader in = new BufferedReader(rdr);
```

Lisez ligne par ligne

```
try {
    String name = in.readLine();
}
catch (IOException e)
{ ... }
```

- La chaîne obtenue peut alors être analysée, convertie,…

Récapitulatif

```
BufferedReader in = new BufferedReader
          (new InputStreamReader(System.in));

try {
       System.out.println("Entrer nom :");
       String name = in.readLine();

       System.out.println(“Bonjour “ + name);

       System.out.println(“Veuillez entrer votre age:");
      int age  = Integer.parseInt(in.readLine());
}
catch (IOException e) { System.err.println(e); }
```

Le package java.nio (pour info)

Pour information:

- Avec la sortie de Java 1.4, un nouveau package d'E/S a été ajouté

Les classes de java.nio apportent :

- De meilleures performances
  - traitements des données par blocs
  - en en déléguant le maximum à l'OS
  - Buffers réutilisables, évitant la création d'objets et la garbage collection
- Des E/S non bloquantes, ou permettant à une tâche de bloquer sur plusieurs E/S à la fois (multiplexage)
- . . .

Ceci est au-delà de notre programme