I/O en Java

Introduction

- Compléments de programmation Java
- Étude de 3 mécanismes nécessaires pour les systèmes distribués
  - Flots de données
  - Threads et synchronisation
  - New I/O
- Très inspiré du cours de Richard Grin (Master 1 Informatique)

1- Java io

Flots

- Représentent un canal de communication
- Les données peuvent y être lues ou écrites séquentiellement
  - Type FIFO
  - Pas d'accès aléatoire comme dans un tableau
- Pas de notion de limites de données
  - Des données écrites en 2 fois peuvent être lues en 1 fois
- Package java.io.
- 2 types de flots
  - Manipulation d'octets
  - Manipulation de caractères
- Tous les flots lèvent une IOException

Java.io

InputStream	Lecture de flots d'octets
OutputStream	Écriture de flots d' octets
Reader	Lecture de flots de caractères
Writer	Écriture de flots de caractères
File	Fichiers et répertoires
StreamTokenizer	Analyse lexicale

Types de flots, Types de classes

- Flots d'octets
  - Lecture/Écriture d'octets
- Flots de caractères
  - Manipule des caractères Unicode
  - Codés en Java sur 2 octets
- 2 Types de classes
  - Classes de base liés à une source ou une destination (Ex: FileReader)
  - Classes de décoration (Ex: BufferedReader)

Sources et destinations concrètes

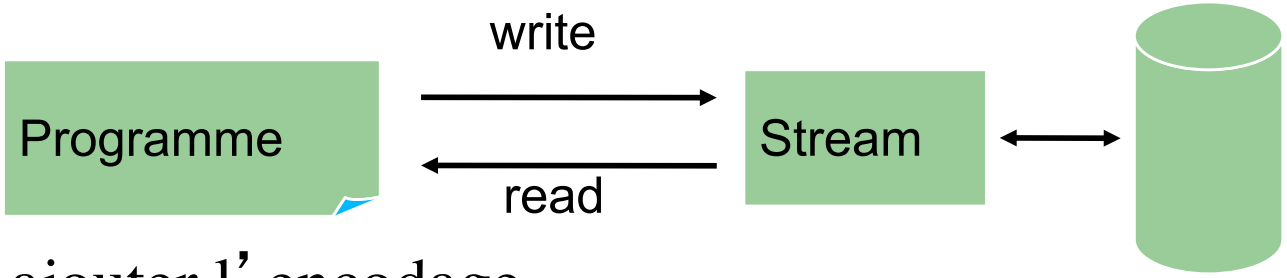
- Fichiers
  - File{In|Out}putStream
  - File{Reader|Writer}
- Tableaux
  - ByteArray{In|Out}putStream
  - CharArray{Reader|Writer}
- Chaînes de caractères
  - String{Reader|Writer}

Décoration

- De base, un flot ne sait que lire ou écrire
  - Supporté par la plupart des périphériques
  - Méthodes read et write
- Mais très vite limitatif
- Pleins de fonctionnalités possibles
  - Mise en mémoire tampon (Bufferisation)
  - Encodage/Décodage des données
  - Compression/Décompression
- Idéalement, indépendants de la lecture ou de l'écriture de base
- Comment ajouter des fonctionnalités sans toucher aux classes de base: la décoration

Décoration

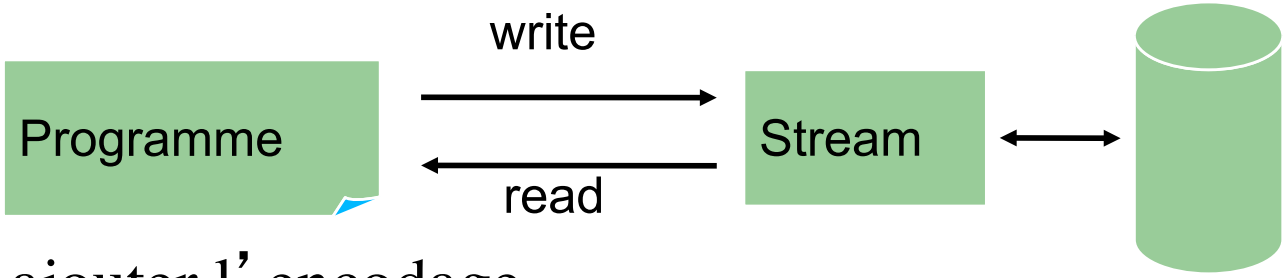
- Exemple: ajouter l'encodage de données



- Comment ajouter l'encodage
- 1ère solution: modification de la classe Stream
- Mais classes multiples (stream normal, stream avec encodage…)
- Nécessite de faire le même travail pour tous les périphériques

Décoration

- Exemple: ajouter l'encodage de données



- Comment ajouter l'encodage
- 1ère solution: modification de la classe Stream
- Mais classes multiples (stream normal, stream avec encodage…)
- Nécessite de faire le même travail pour tous les périphériques
- 2ème solution: on décore la classe Stream, on lui ajoute une fonctionnalité (par héritage ou wrapper)

Décorateurs

- Bufferisation des I/Os
  - Buffered{In|Out}putStream
  - Buffered{Reader|Writer}
- Lecture/écriture de types primitifs indépendamment de la plateforme
  - Data{In|Out}putStream
- Compteur de lignes
  - LineNumberReader
- Écriture de données sous forme de chaînes de caractères
  - PrintStream
  - PrintWriter
- Remettre un caractères dans un flot
  - PushbackInputStream
  - PushbackReader

1.1- Lecture de flots d’octets

Classe InputStream

- Classe abstraite
- Superclasse de toutes les classes permettant la lecture de flots d'octets
- Toutes les classes manipulant un flot d'octets seront vues comme un InputStream
  - Équivalent à une interface
- Possède un constructeur sans paramètre

Méthodes publiques

- int read()
  - Retourne le prochain octet dans le flot de données
  - Valeur de retour entre 0 et 255
  - Retourne -1 si fin de flot atteinte
  - Bloquante si aucun octet à lire
  - Abstraite

Méthodes publiques

- int read(byte[] b)
  - Essaie de lire assez d'octets pour remplir le tableau b
  - Retourne le nombre d'octets effectivement lus ou -1
  - Implémentée en utilisant la méthode read()
- Attention
  - Le tableau peut n'être rempli que partiellement
  - Retourne le nombre lu, pas la valeur!

Méthodes publiques

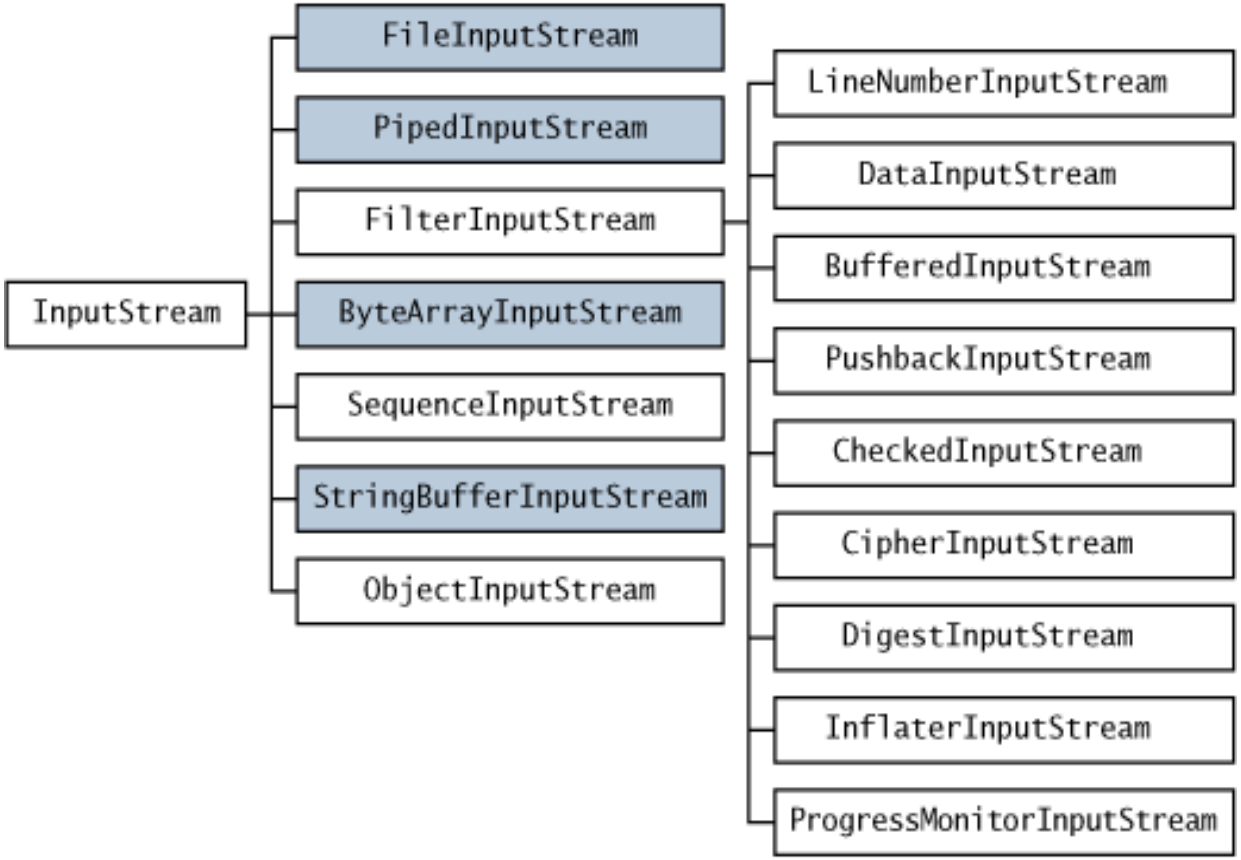
- int read(byte[] b,int off, int len)
  - Lis len octets et les place dans le tableau à partir de l'indice off.
  - Retourne le nombre d'octets effectivement lus ou -1
- long skip(long n)
  - Saute n octets dans le flot
  - Retourne le nombre d'octets sautés
- int available()
  - Renvoie le nombre d'octets pouvant être lus

Méthodes publiques

- boolean markSupported()
  - Test si le flot supporte la notion de marque et de retour en arrière
- void mark(int readlimit)
  - Marque la position courante
  - Readlimit : nombre d'octets lus avant oubli de la marque
- void reset()
  - Positionne le flot à la dernière marque

Sous classes de InputStream

- En gris sont indiquées des classes associées à des sources et destinations concrètes



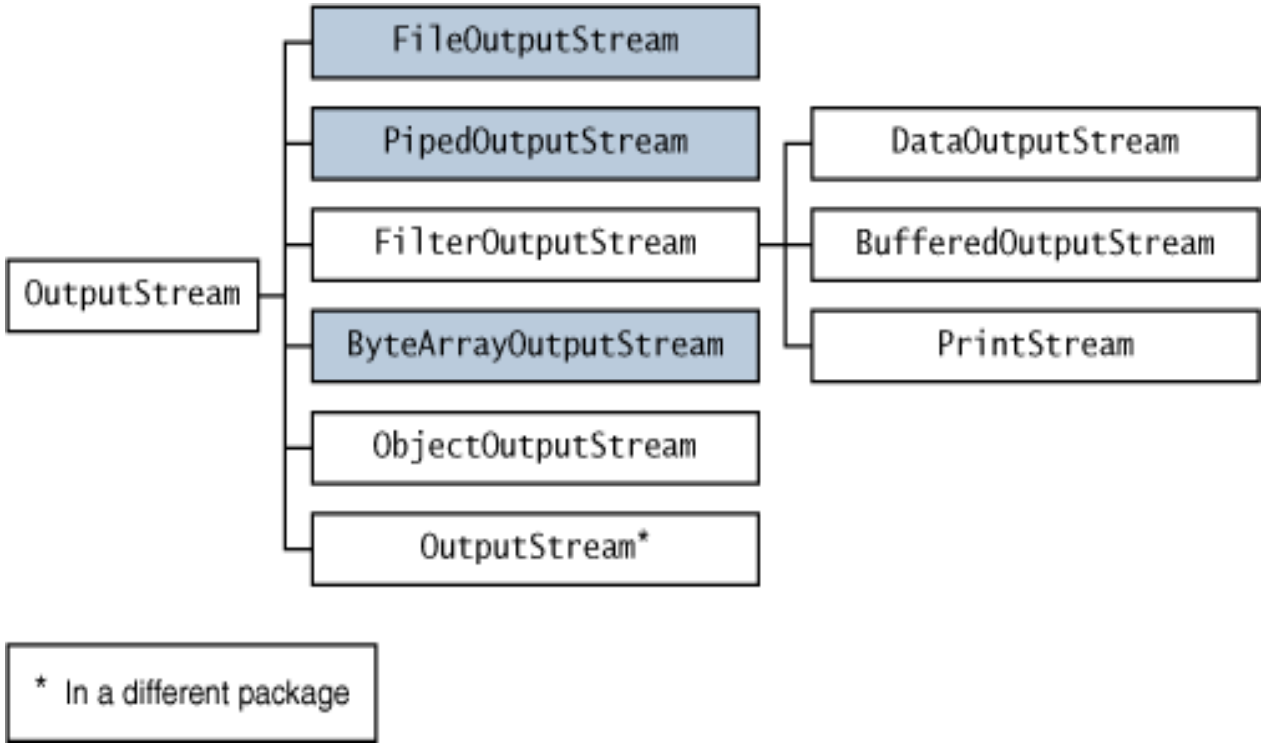
1.2- Écriture de flots d’octets

Classe OutputStream

- Classe abstraite
- Méthodes
  - int write(int c)
  - int write(byte cbuf[])
  - int write(byte cbuf[], int offset, int length)
- Dans le cas de write(int c) seuls les 8 bits de poids faible sont écrits

Sous classes de OutputStream

- En gris sont indiquées des classes associées à des sources et destinations concrètes



1.3- Décorateurs et Filtres

Principe

- Un objet décorateur ajoute une fonctionnalité à un objet décoré
- Le constructeur du décorateur reçoit l'objet qu'il décore
- Quand une méthode est appelée sur le décorateur
  - Il effectue un traitement
  - Utilise, si nécessaire, les méthodes de l'objet décoré
  - Retour un éventuel résultat

Principe

- Pour fonctionner, le décorateur doit être compatible avec le décoré
  - Utilisation d'interfaces ou héritage
- Principe récursif
  - Un décorateur peut être décoré
- En Java, les décorateurs sont sous classes d'InputStream et d'OutputStream

Exemple

- Décoration pour lire des flux depuis une Socket
- Obtenir le flot d'entrée de la socket
  - `maSocket.getInputStream()`
- Le décorer pour bufferiser les lectures
  - `BufferedInputStream bis = new BufferedInputStream(maSocket.getInputStream())`
- Ensuite, on peut lire des flots d'octets depuis bis
- Même fonctionnement pour l'ecriture
  - `BufferedOutputStream bos = new BufferedOutputStream( maSocket.getOutputStream());`

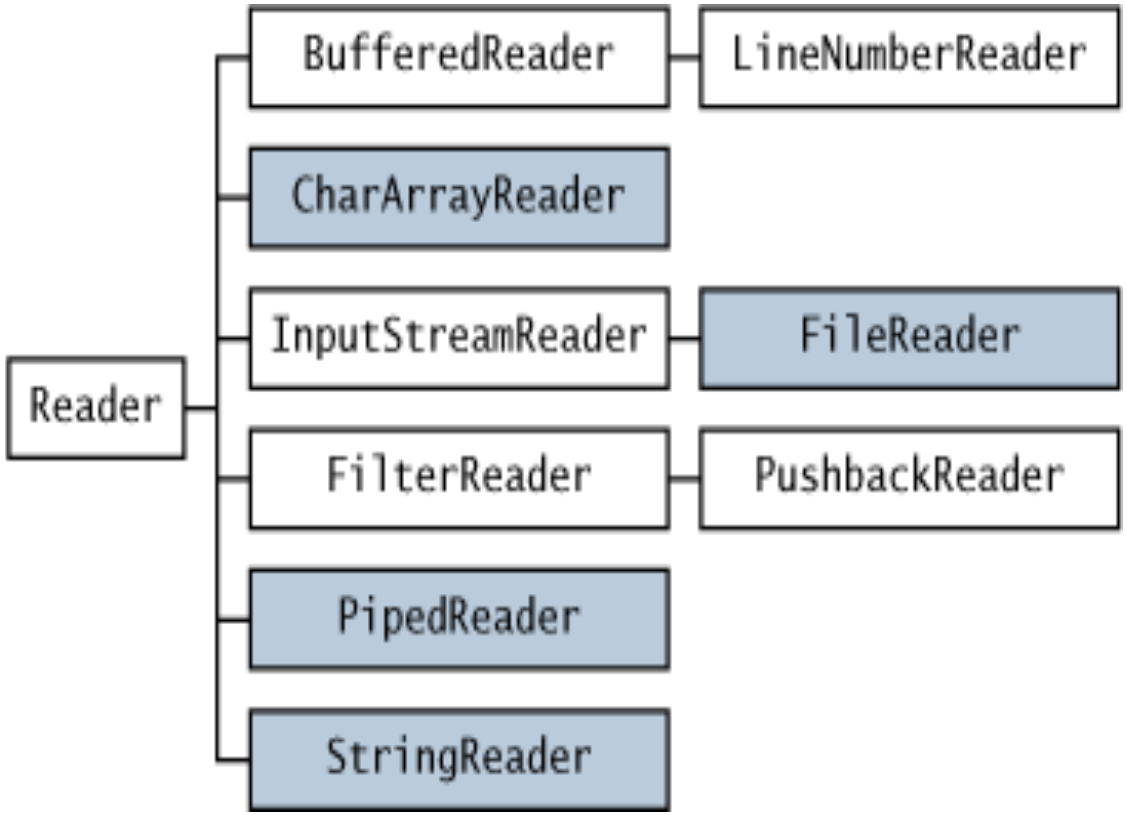
1.4- Lecture de flots de caractères

Classe Reader

- `int read()`
- `int read(char[] b)`
- `int read(char[] b, int off, int len)`
- `long skip(long n)`
- `boolean ready()`
- `abstract void close()`
- `void mark()`
- `void reset()`
- `boolean markSupported()`

Sous classes de Reader

- Pour lire des lignes de texte, on utilise `BufferedReader` et la méthode `readLine()`



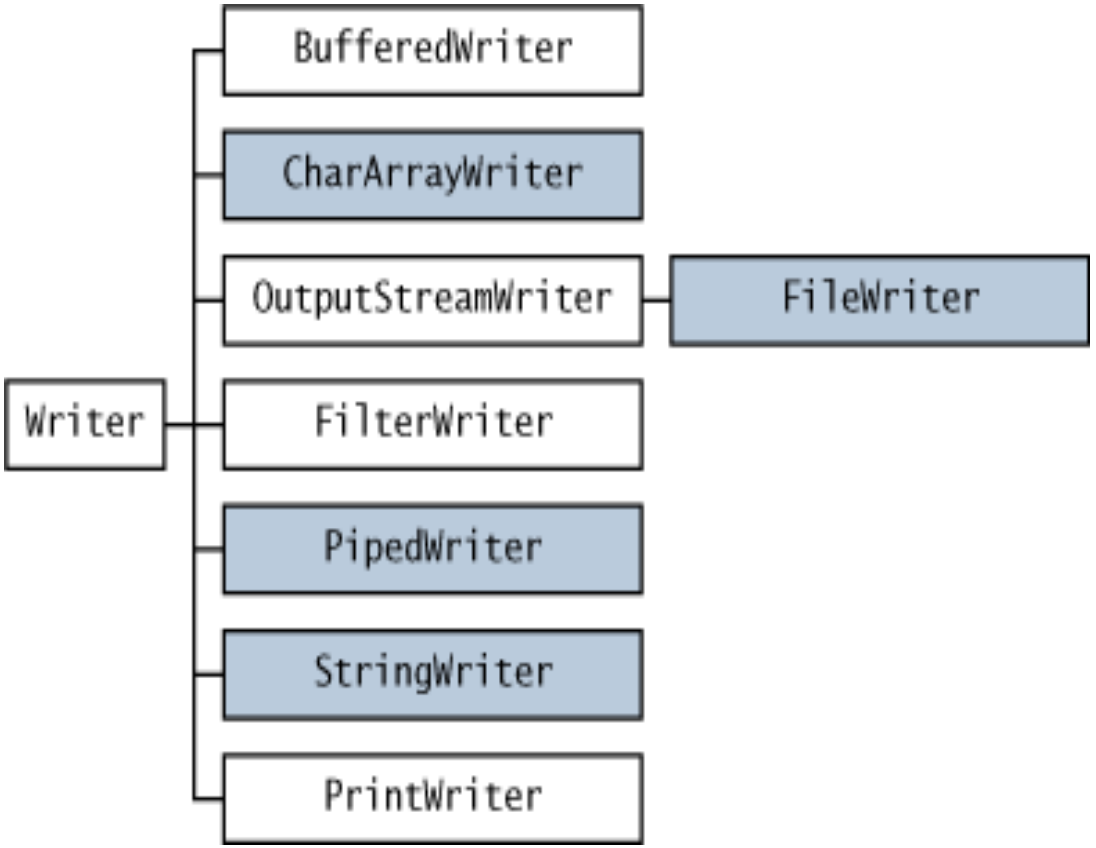
1.5- Écriture de flots de caractères

Classe Writer

- `int write(int c)`
- `void write(char[] b)`
- `void write(char[] b, int off, int len)`
- `void write(String s)`
- `void write(String s, int off, int len)`
- `abstract void flush()`
- `abstract void close()`

Sous classes de Writer

- Pour écrire des lignes de texte, on utilise `PrintWriter` et la méthode `println()`



Exemple

- Lecture de flux depuis une socket en mode caractère
  - `in = new BufferedReader(new InputStreamReader( maSocket.getInputStream()));`
- Même fonctionnement pour l'ecriture
  - `out = new PrintWriter(maSocket.getOutputStream(), true);`
- La transformation entre octets et caractères se fait avec les classes `InputStreamReader` et `PrintWriter`