Java et OOP
Flux et fichiers

Fichier et répertoire

- *Fichier* : ensemble de données stockées dans la mémoire auxiliaire.
- *Répertoire* : conteneur de fichiers.
- Arborescence de fichiers.
- Nom de fichier/répertoire.
- Chemin d'accès – *absolu, relatif.*

La classe `File`

- Création/suppression de fichiers/répertoires :
  - `boolean createNewFile(), mkdir(), delete()`
- Test d'existance.
  - `boolean exists(), canWrite(), canRead()`
- Renommage :
  - `boolean renameTo(File dest)`
- Contenu des répertoires :
  - `String[] list()`
  - `File[] listFiles()`
- Modification d'attributs.
  - `boolean setLastModified(), setReadOnly()`
- Pas de lecture ou écriture !

Lecture et écriture dans fichiers

- Lecture/écriture séquentielle :
  - Chaque opération fait avancer un hypothétique pointeur de fichier du nb. de blocs lus/écrits.
  - Le pointeur de fichier ne peut que être incrémenté.
  - Le pointeur de fichier n'est pas accessible en tant que variable/attribut.
- Lecture/écriture en accès aléatoire :
  - Le pointeur de fichier est déplaçable dans les deux directions, sans faire des opérations de lecture/écriture..

Lecture/écriture séquentielle

- `FileInputStream/FileOutputStream` – flux d'octets, héritières de `InputStream`, resp. `OutputStream`.
  - Constructeurs : `FileInputStream(File nom)`, `FileInputStream(String nom)`.
  - Constructeurs : `FileOutputStream(File nom)`, `FileOutputStream(String nom)`, `FileInputStream(File nom, boolean append)`.
  - Exception `FileNotFoundException`.
  - Lecture : `int read()`, `int read(byte[] ouLire)`.
  - Écriture : `int write()`, `int read(byte[] dOuEcrire)`.
- `FileReader/Writer` – flux de caractères, héritières de `InputStreamReader`, resp. `OutputStreamWriter`.
  - Constructeur lecture : `FileReader(File nom)` ou `FileWriter(String nom)`.
  - Constructeur écriture : `FileReader(File nom)` ou `FileWriter(String nom)`, mais aussi `FileWriter(File nom, boolean append)`.
  - Exception `FileNotFoundError`.
  - Lecture : `int read()`, `int read(byte[] ouLire)`.
  - Écriture : `int write(String quoi)`, `int write(String[] quoi, int dOu, int jsqOu)`, `int write(char[] quoi, int dOu, int jsqOu)`.
  - Exception `FileNotFoundException` en lecture, `IOException` en écriture.

Exemple : concaténation des fichiers

- Classe `SequenceInputStream`
  - C'est une "concaténation logique" des fichiers passés en paramètre au constructeur.
  - Constructeur `SequenceInputStream(InputStream nom1, InputStream nom2)` pour concaténer deux fichiers.
  - Constructeur `SequenceInputStream(Enumeration liste)` pour concaténer plusieurs.
  - La lecture des données se fait dans l'ordre donnée par l'énumération `liste`.
  - Dans l'énumération `liste`, les types des objets au moment de l'exécution doivent être des héritiers de `InputStream` – sinon exception `IOException`.
- Pour concaténer une liste des fichiers et remettre le résultat dans un fichier il faut :
  - Écrire une classe `ListeFichiers` implémentant l'intérface `Enumeration`.
  - Écrire du code dans le main pour créer le fichier avec la concaténation – la `SequenceInputStream` n'assure pas cela !

Fichiers à accès aléatoire (random access)

- `RandomAccessFile`, héritière de `Object` !
- Elle ne fait pas partie de la hiérarchie des flux !
- Implémente les interfaces `DataInput, DataOutput`.
- Constructeurs : `RandomAccessFile(File nom, String mode)` ou `RandomAccessFile(String nom, String mode)`
  - Modes d'accès : `"r"` pour lecture seule, `"rw"` pour lecture et écriture.
- Recupérer la valeur du pointeur de fichier : `long getFilePointer()`.
- Déplacer le pointeur de fichier : `void seek(long ou)`.
- Récuperer la longueur du fichier (en octets) : `byte length()`
- Modifier la longueur du fichier : `void setLength(long nvLg)`.
- Lecture : `int read(), byte readByte(), float readFloat()` etc.
- Écriture : `int write(), void writeByte(byte quoi), void writeFloat(float quoi)` etc.
- Exceptions :
  - `EOFException` : si une lecture dépasse la fin du fichier avant de se terminer.
  - `FileNotFoundException, IllegalArgumentException` – dans les constructeurs.
  - `IOException` – d'autres problèmes d'entrée-sortie.