Informatique 1

11b. Flux de données

Lire et écrire des fichiers

Dr Pierre-André Mudry, Dr Pierre Roduit
pierre-andre.mudry@hevs.ch

Rev 1.41

Objectifs de cette leçon

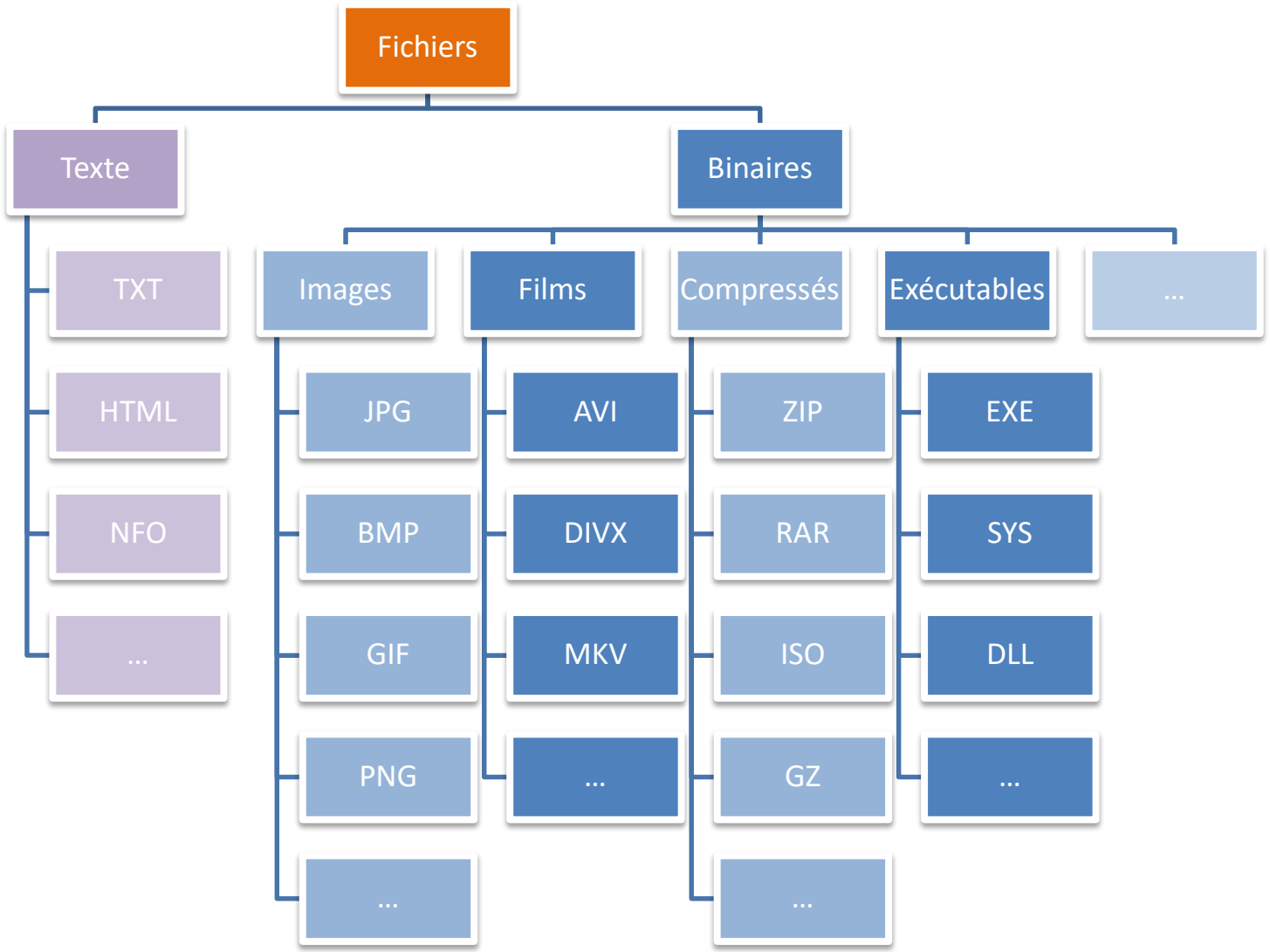
Pouvoir gérer des fichiers en Java

- Fichiers sur ordinateur
- Écrire un fichier
- Lire un fichier

Fichiers

- Stockés sur support
  - Disque, carte mémoire, stick USB
  - Toujours des 0 et des 1
- Types de fichiers ?
  - Certains lisibles par humain (texte simple, `.txt`)
  - Certains sont ***formatés*** et doivent être décodés par un algorithme (fichiers *Word*, images, films…)

Types de fichier

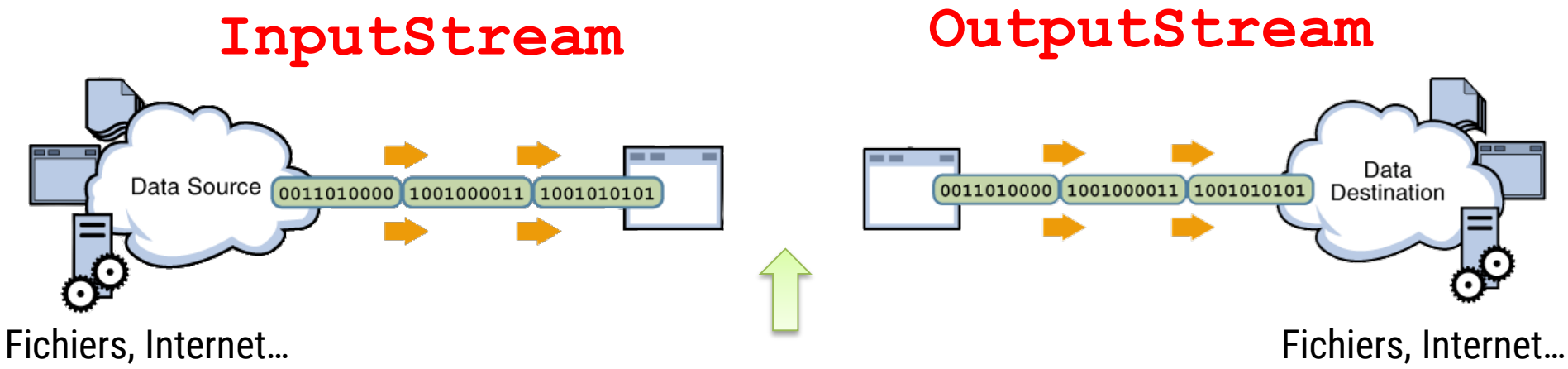


Créer ou lire des fichiers en Java ?

Les *streams* (flux) sont des objets Java pour transférer des données entre du code et des entrées/sorties.

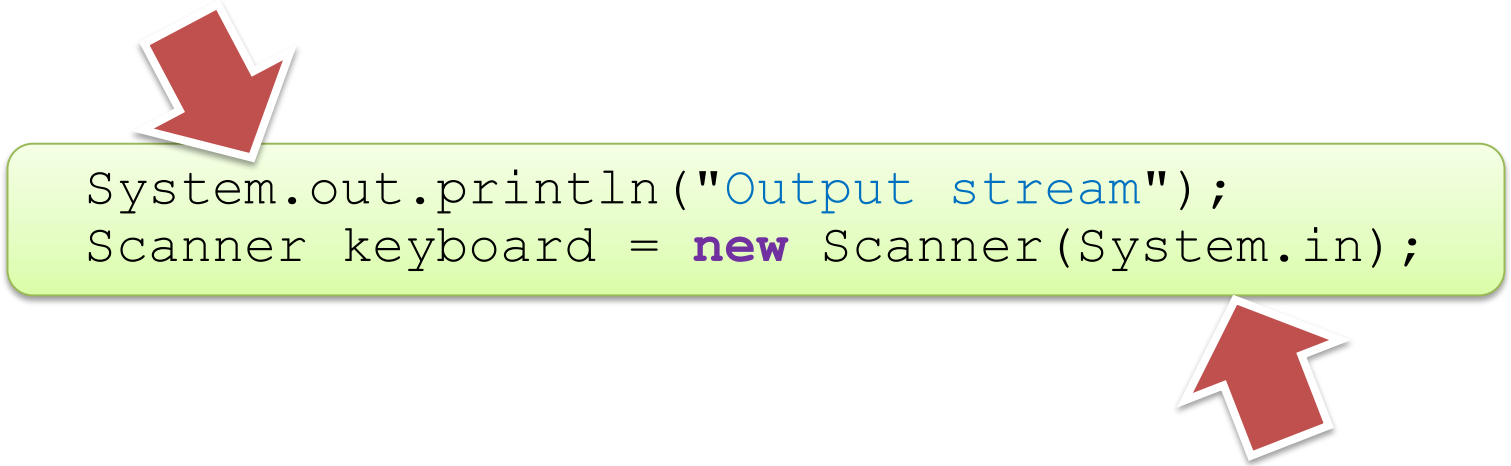
Stream

Entrées et sorties



Un flux connu : la console

- `System.out`
  - un *flux de sortie* connecté à la console
- `System.in`
  - un *flux d'entrée* connecté au clavier de la console



```
System.out.println("Output stream");
Scanner keyboard = new Scanner(System.in);
```

Objectifs

Gérer des fichiers en Java

- Fichiers sur ordinateur
- Écrire un fichier
- Lire un fichier

Écrire un fichier texte

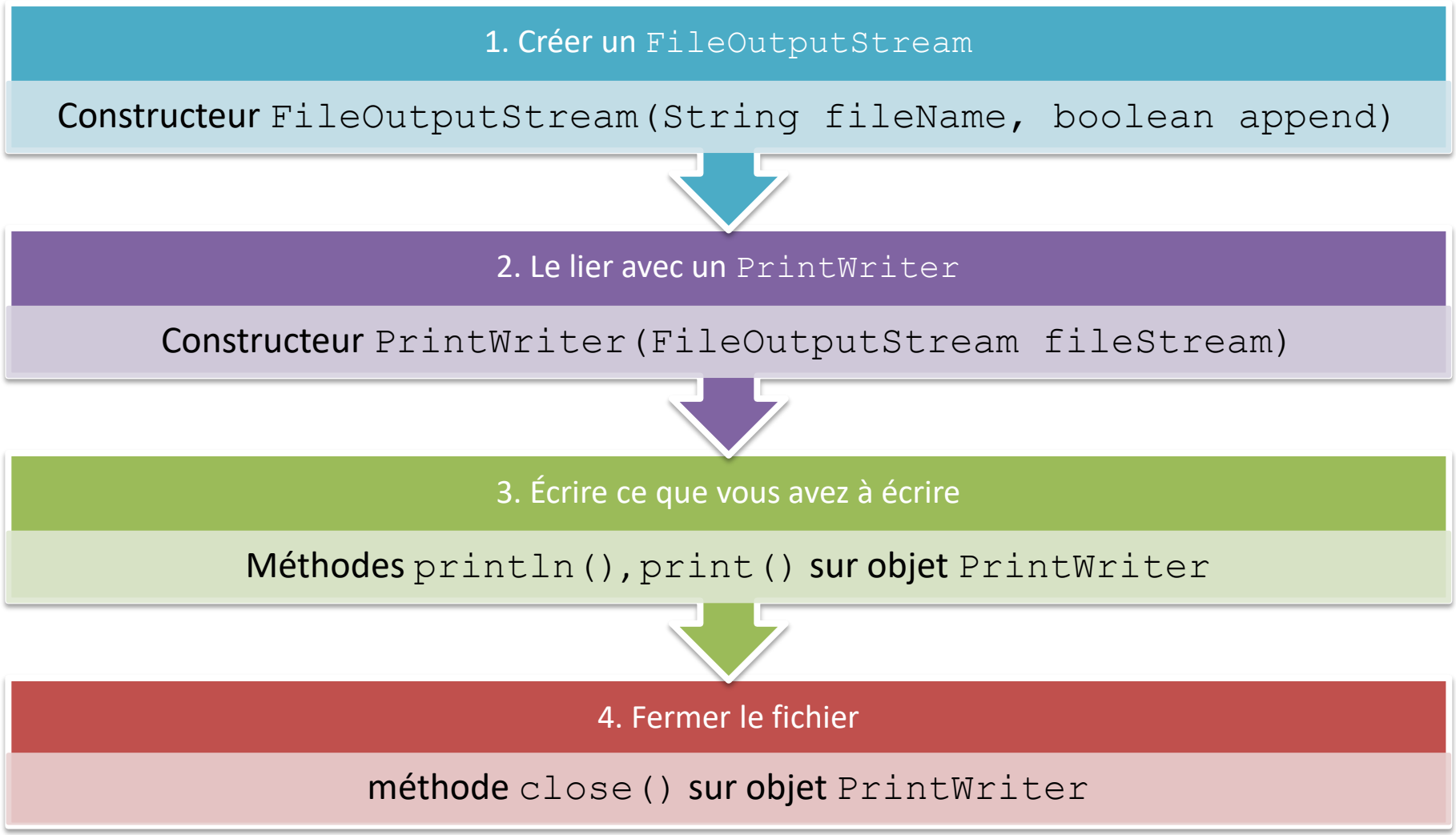
1. *Créer un flux de fichier*
   - `FileOutputStream`
     - lien entre code et système exploitation
     - dit où se trouve le fichier
2. *Pour écrire comme sur console*
   - `PrintWriter`

Imiter la console : `PrintWriter`

- Donner un flux dans constructeur
- Un objet `PrintWriter` possède les méthodes `print` et `println`
- Similaires aux méthodes de `System.out` mais sortie sur fichier, pas écran

StreamWriteDemo

Recette pour écrire fichier texte



Quelques méthodes de PrintWriter 1/3

Display 10.2 Some Methods of the Class `PrintWriter`

`PrintWriter` and `FileOutputStream` are in the `java.io` package.

```
public PrintWriter(OutputStream streamObject)
```

This is the only constructor you are likely to need. There is no constructor that accepts a file name as an argument. If you want to create a stream using a file name, you use

```
new PrintWriter(new FileOutputStream(File_Name))
```

When the constructor is used in this way, a blank file is created. If there already was a file named *File_Name*, then the old contents of the file are lost. If you want instead to append new text to the end of the old file contents, use

```
new PrintWriter(new FileOutputStream(File_Name, true))
```

(For an explanation of the argument `true`, read the subsection "Appending to a Text File.")

When used in either of these ways, the `FileOutputStream` constructor, and so the `PrintWriter` constructor invocation, can throw a `FileNotFoundException`, which is a kind of `IOException`.

If you want to create a stream using an object of the class `File`, you can use a `File` object in place of the *File_Name*. (The `File` class will be covered in Section 10.3. We discuss it here so that you will have a more complete reference in this display, but you can ignore the reference to the class `File` until after you've read that section.)

(continued)

W. Savitch, *Absolute Java*, 2nd edition, 2009

Quelques méthodes de PrintWriter 2/3

Display 10.2 Some Methods of the Class `PrintWriter`

```
public void println(Argument)
```

The *Argument* can be a string, character, integer, floating-point number, `boolean` value, or any combination of these, connected with + signs. The *Argument* can also be any object, although it will not work as desired unless the object has a properly defined `toString()` method. The *Argument* is output to the file connected to the stream. After the *Argument* has been output, the line ends, and so the next output is sent to the next line.

```
public void print(Argument)
```

This is the same as `println`, except that this method does not end the line, so the next output will be on the same line.

(continued)

W. Savitch, *Absolute Java*, 2nd edition, 2009

Quelques méthodes de PrintWriter 3/3

Display 10.2 Some Methods of the Class `PrintWriter`

`public PrintWriter printf(`*Arguments*`)`

This is the same as `System.out.printf`, except that this method sends output to a text file rather than to the screen. It returns the calling object. However, we have always used `printf` as a `void` method.

`public void close()`

Closes the stream's connection to a file. This method calls `flush` before closing the file.

`public void flush()`

Flushes the output stream. This forces an actual physical write to the file of any data that has been buffered and not yet physically written to the file. Normally, you should not need to invoke `flush`.

W. Savitch, *Absolute Java*, 2nd edition, 2009

Objectifs

Gérer des fichiers en Java

- Fichiers sur ordinateur
- Écrire un fichier
- Lire un fichier

Lire un fichier

- Démarche semblable à l'écriture
- `FileReader` fait le lien avec les fichiers physiques
- `BufferedReader`
  - Sert à manipuler le contenu du fichier
  - Méthodes `read()` et `readLine()`

StreamReadDemo