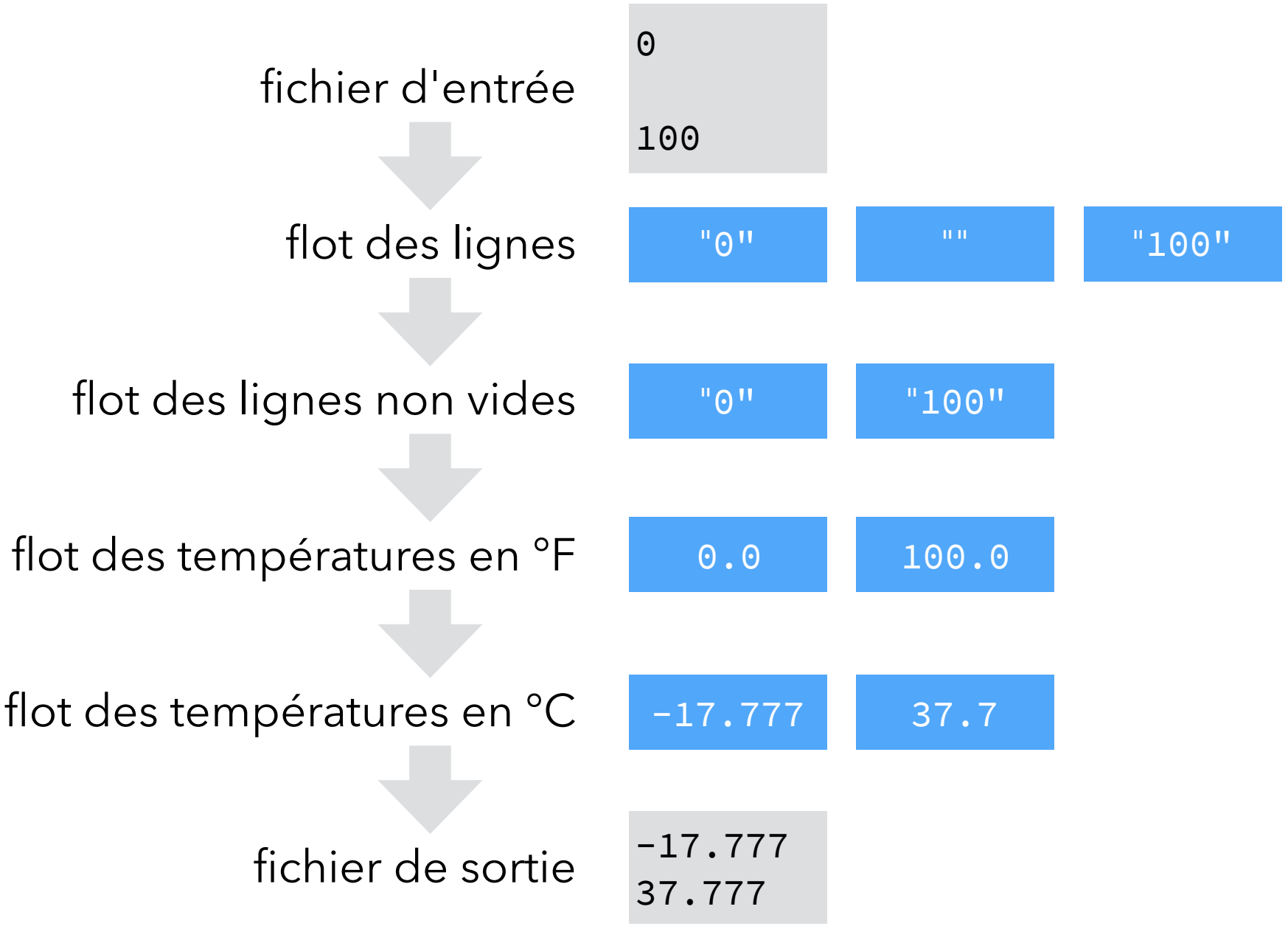
Programmation par flots

Pratique de la programmation orientée-objet

Michel Schinz – 2017-05-15

Conversion °F en °C



Interface Stream & autres

```
package java.util.stream;

public interface Stream<T> { … }

// ≅ Stream<Double>, mais plus rapide
public interface DoubleStream { … }
// ≅ Stream<Long>, mais plus rapide
public interface LongStream { … }
// ≅ Stream<Integer>, mais plus rapide
public interface IntStream { … }
```

Pipelines

Lorsqu'on « programme par flots », on construit des pipelines constitués de :

- *une* **méthode source**, qui fournit les valeurs du flot,
- *un nombre quelconque* de **méthodes intermédiaires**, qui transforment les valeurs du flot,
- *une* **méthode terminale**, qui consomme les valeurs du flot.

Méthodes sources

Les **méthodes sources** sont :

- des méthode statiques de `Stream` ou l'une de ses spécialisations (`DoubleStream`, etc.),
- des méthodes de pont (adaptateurs) d'entités pouvant être vues comme un flot : collections, lecteurs, etc.

Exemples :

- `Stream.of`, `Stream.generate`, `Stream.iterate`, `IntStream.range`, etc.
- `Collection.stream`, `Arrays.stream`, `BufferedReader.lines`, `Random.doubles`, etc.

Méthodes intermédiaires

Les **méthodes intermédiaires** (*intermediate methods* ou *intermediate operations*) sont des méthodes par défaut de `Stream` ou l'une de ses spécialisations et qui retournent un flot (décorateurs).

Exemples :

- `map`, `filter`,
- `limit`, `skip`,
- `sorted`, `distinct`,
- etc.

Méthodes terminales

Les **méthodes terminales** (*terminal methods* ou *terminal operations*) sont des méthodes par défaut de `Stream` ou l'une de ses spécialisations et qui retournent autre chose qu'un flot (adaptateur).

Exemples :

- `allMatch`, `anyMatch`, `noneMatch`,
- `count`, `min`, `max`, `reduce`,
- `forEach`, `forEachOrdered`,
- `toArray`, `collect`,
- etc.

Collecteurs

Un **collecteur** (*collector*) reçoit les valeurs d'un flot et les stocke dans une collection ou les réduit d'une manière ou d'une autre.

La classe `Collectors` fournit plusieurs collecteurs prédéfinis. Exemples :

- `toList`, `toSet`, `toMap`,
- `averagingDouble`, `averagingLong`, `averagingInt`, `summingDouble`, `summingLong`, `summingInt`,
- `minBy`, `maxBy`,
- `joining`,
- etc.