Surcharge

Alexis Nasr (d'après les slides de Arnaud Labourel)

Méthodes ayant le même nom

Dans une classe, plusieurs méthodes peuvent avoir le même nom.

C'est ce qu'on appelle la **surcharge** de méthode.

Les différentes méthodes partageant le même nom ne peuvent pas avoir la même séquence d'arguments.

La méthode est choisie par le compilateur de la façon suivante :

- Le nombre de paramètres doit correspondre
- Les affectations des paramètres doivent être valides
- Parmi ces méthodes, le compilateur choisit la plus spécialisée, c'est-à-dire celle entraînant le moins de changement de types (passage à une super-classe ou promotion pour les types primitifs)

Exemple de surcharge de méthode

```
public class DataArtist {

    public void draw(String s) {
        /* ... */
    }
    public void draw(int i) {
        /* ... */
    }
    public void draw(double f) {
        /* ... */
    }
    public void draw(int i, double f) {
        /* ... */
    }
}
```

Exemple de surcharge de méthode

```
class Adder {
  public static int add(int intVal1, int intVal2) {
    System.out.println("integer");
    return intVal1+intVal2;
  }

  public static double add(double doubleVal1,
                           double doubleVal2) {
    System.out.println("double");
    return doubleVal1+doubleVal2;
  }
}
```

Exemple de surcharge de méthode

```java
int intValue = 1;
double doubleValue = 2.2;
double result1 = Adder.add(doubleValue, doubleValue);
// → double

int result3 = Adder.add(intValue, intValue);
// → int
```

Promotion pour les types primitifs

Si les types des arguments ne correspondent pas aux types des paramètres, les types des arguments sont promus en suivant les flèches :

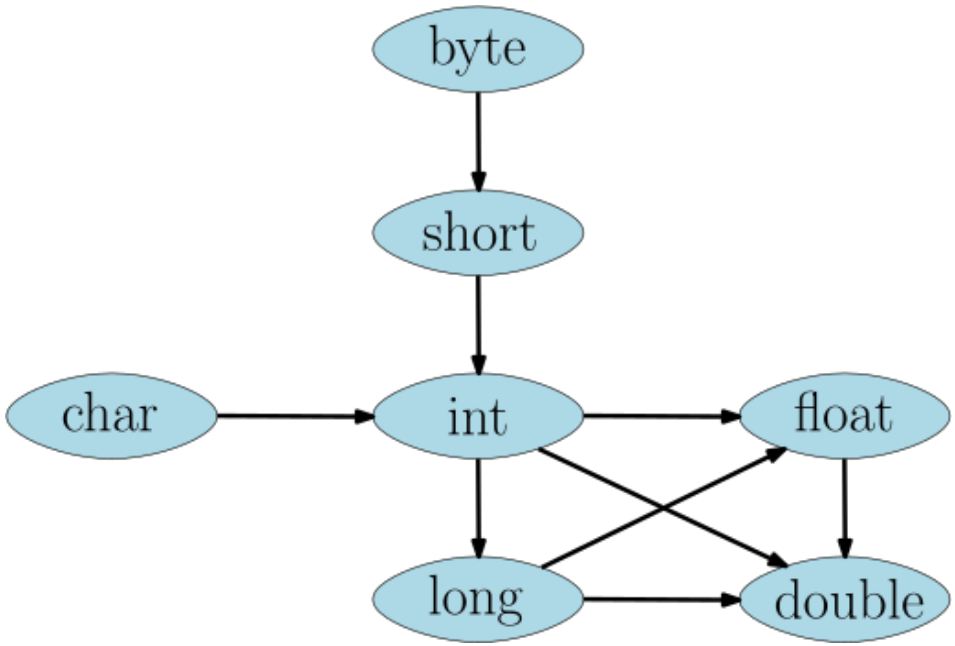

FIGURE 1 – promotion des types primitifs

```
double result2 = Adder.add(intValue, doubleValue);
// → double
```

Exemple de surcharge de méthode

Pour les instances de classe, c'est le type du conteneur à la compilation qui compte.

```
class Printer {
  static void print(Object object) {
    System.out.println("Object : "+object);
  }
  static void print(String string) {
    System.out.println("String : "+string);
  }
}
String string = "message";
Object object = string;
Printer.print(string); // → String : message
Printer.print(object); // → Object : message
```

Plusieurs constructeurs

C'est exactement les mêmes règles qui s'appliquent pour les constructeurs d'une classe.

```
public class Point{
    public int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
        public Point() {
        this(0, 0);
    }
    public Point(Point p) {
        this(p.x, p.y);
    }
}
```