2.1.9 La surcharge

Il est impossible, en Java, de définir deux fonctions qui ont le même nom, par exemple

```
static int f (final int x) {
 return x;}

static int f (final int x) {
 return x + 1;}
```

sauf quand le nombre ou le type de leurs arguments sont différents. On peut, par exemple, définir les trois fonctions homonymes

```
static int f (final int x) {
 return x;}

static int f (final int x, final int y) {
 return x + 1;}

static int f (final boolean x) {
 return 7;}
```

Lors de l'évaluation d'une expression de la forme `f(`t_1`, ..., `t_n`)`, la fonction appelée est choisie en fonction de son nom et aussi du nombre et du type de ses arguments. Les expressions `f(4)`, `f(4,2)` et `f(true)` ont donc respectivement 4, 5 et 7 pour valeur. On dit, dans ce cas, que le nom `f` est *surchargé*.

Il n'y a pas de surcharge en Caml. Les programmes

```
let f x = x in let f x = x + 1 in print_int (f 4)
```

et

```
let f x = x in let f x y = x + 1 in print_int (f 4 2)
```

sont corrects, mais la première définition est simplement cachée par la seconde.

Enfin, il n'y a pas non plus de surcharge en C et le programme

```
int f (const int x) {return x;}
int f (const int x, const int y) {return x + 1;}
...
```

est incorrect.