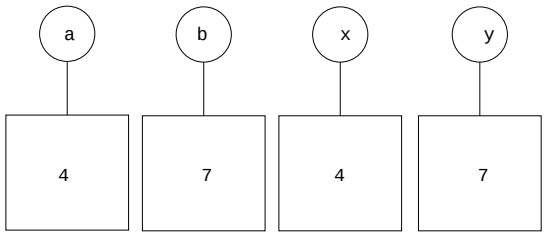
```
static public void main (String [] args) {
 a = 4;
 b = 7;
 swap(a,b);
 System.out.println(a);
 System.out.println(b);}}
```

De manière surprenante, alors que l'on s'attendrait naïvement à ce que les contenus de `a` et `b` aient été intervertis et que les nombres 7 puis 4 s'affichent, c'est le nombre 4 qui s'affiche en premier, suivi du nombre 7.

En fait, ce résultat est tout à fait cohérent avec la définition de la fonction Σ que nous avons donnée ci-dessus. On part avec un environnement `e = [a = r1, b = r2]` et une mémoire `m = [r1 = 4, r2 = 7]`. L'appel de la fonction `swap(a,b) ;` demande de calculer les valeurs des expressions `a` et `b` dans l'environnement `e` et la mémoire `m`. On obtient 4 et 7 respectivement. Puis on construit l'environnement `[a = r1, b = r2, x = r3, y = r4]` et la mémoire `[r1 = 4, r2 = 7, r3 = 4, r4 = 7]`



On intervertit ensuite le contenu des variables `x` et `y` ce qui donne la mémoire `[r1 = 4, r2 = 7, r3 = 7, r4 = 4, r5 = 4]` et on revient au programme principal. L'environnement est alors `e = [a = r1, b = r2]` et la mémoire `[r1 = 4, r2 = 7, r3 = 7, r4 = 4, r5 = 4]`. Le contenu des variables `a` et `b` n'a pas changé.

Autrement dit, la fonction `swap` ignore tout des variables `a` et `b`, elle ne peut utiliser que leur valeur au moment de l'appel et, en aucun cas, elle ne peut modifier leur contenu : exécuter l'instruction `swap(a,b) ;` donne le même résultat qu'exécuter l'instruction `swap(4,7) ;`.

Le mécanisme de passage des arguments, tel que nous l'avons décrit, et qui s'appelle le *passage par valeur*, ne permet donc pas d'écrire une fonction `swap` qui intervertit le contenu de deux variables. Cependant, la plupart des langages de programmation comportent une construction qui permet d'écrire une telle fonction. Mais, cette construction est un peu différente dans chaque langage. Avant d'étudier le cas de Java, Caml et C, il n'est pas inutile de décrire celui, beaucoup plus simple, de Pascal.