même bloc. C'est ce bloc que l'on veut isoler dans une fonction. Mais pour tenir compte des différences entre ces deux copies, on doit introduire deux paramètres : l'un pour la direction et l'autre pour l'horaire du RER. Cela amène à définir une fonction `annoncerRer`

```
static void annoncerRer (final String d, final String h) {
 System.out.print("Le RER en direction de ");
 System.out.print(d);
 System.out.print(" partira à ");
 System.out.println(h);
 System.out.println();
 System.out.println();
 System.out.println();}
```

puis à l'utiliser dans le programme principal qui devient alors

```
annoncerRer("Saint-Rémy-lès-Chevreuse","8h50");
annoncerRer("Massy-Palaiseau","8h55");
```

Les variables `d` et `h` qui figurent comme arguments dans la définition de la fonction s'appellent les *arguments formels* de la fonction. Quand on appelle une fonction `annoncerRer("Massy-Palaiseau","8h55");` les expressions `"Massy-Palaiseau"` et `"8h55"` que l'on donne en arguments s'appellent les *arguments réels* de l'appel.

Un argument formel, comme toute variable, peut être déclaré final ou mutable. S'il est final, il ne peut pas être affecté dans le corps de la fonction.

Pour poursuivre la comparaison, le langage mathématique utilise aussi de telles définitions paramétrées : « Le groupe $\mathbb{Z}/n\mathbb{Z}$ est ... », « Un `K`-espace vectoriel est ... », ...

*En Caml, une définition de fonction s'écrit* `let f x y ... = t in p`.

```
let annoncerRer d h =
   print_string "Le RER en direction de ";
   print_string d;
   print_string " partira à ";
   print_string h;
   print_newline ();
   print_newline ();
   print_newline ()
in annoncerRer "Saint-Rémy-lès-Chevreuse" "8h50";
   annoncerRer "Massy-Palaiseau" "8h55"
```

*Les arguments formels sont toujours finaux. En revanche, si les arguments sont eux-mêmes des références, on peut les affecter comme n'importe quelle référence.*