Format d'affichage – Fonction printf

Pour afficher un message à l'écran constitué de plusieurs informations, habituellement on concatène celles-ci avec l'opérateur +.

```
System.out.println("Entier =  " + nombreEntier + "\nReel =  " + nombreReel);
```

Java 5.0 a introduit une nouvelle possibilité qui ressemble à celle qui existe dans le langage C : la méthode ***printf*** (*print formatted*). L'instruction ci-dessus peut aussi s'écrire :

```
System.out.printf("Entier =  %d\nReel = %.2f\n", nombreEntier, nombreReel);
```

La fonction ou méthode *printf* est une méthode à nombre variable d'arguments (méthode *varargs*) :

- Le premier argument est la chaîne de formatage qui spécifie le texte à afficher et inclut un ou plusieurs **spécificateurs de format** sous la forme de séquences d'échappement commençant par le caractère **%**.

- Les autres arguments sont des valeurs à convertir en chaînes de caractères et à substituer aux spécificateurs de format situés dans la chaîne premier argument.

Par exemple, le spécificateur

%s	correspond à une chaîne de caractères
%d	correspond à un entier
%f	correspond à un nombre flottant
%tc	pour une date (classe prédéfinie *Date* ou *Calendar*)
%n	correspond à un retour à la ligne

a) Possibilités pour afficher les nombres entiers

Supposons que l'on souhaite afficher plusieurs résultats de type entier, chacun sur une ligne différente, et que l'on souhaite également que les entiers soient alignés en colonne à droite, comme dans l'exemple ci-dessous :

```
  123
   25
    8
 1256
```

Pour ce faire, on utilise le spécificateur de format `% un_nombre d.` Le nombre est un entier qui indique le nombre de positions qui seront dédiées à l'affichage de l'entier. Par exemple, si le nombre de positions est égal à 6, et si l'entier comporte un seul chiffre, 5 espaces seront d'abord affichés puis l'entier lui-même. Si par contre l'entier comporte 4 chiffres, seulement 2 espaces seront affichés avant le premier chiffre du nombre entier. Dans tous les cas, on a la garantie que la totalité des chiffres de l'entier sera affiché. Donc si l'entier comporte 10 chiffres, le format d'affichage spécifié sera ignoré.

On souhaite aussi parfois que des 0 préalables soient affichés avant le premier chiffre significatif du nombre entier. Pour ce faire, il faut utiliser le format `%0 un_nombre d.` Le fonctionnement est analogue au format précédent, à une différence près : des 0 sont affichés à la place des espaces.

Exemple

```
// TEST AFFICHAGE DES ENTIERS
System.out.printf("Entier = %6d \n", 5);            // Entier =      5
System.out.printf("Entier = %6d \n", 1234);         // Entier =   1234
System.out.printf("Entier = %6d \n", 123456);       // Entier = 123456
System.out.printf("Entier = %6d \n", 123456789);    // Entier = 123456789


// TEST AFFICHAGE DES ENTIERS avec 0 préalables
System.out.printf("Entier = %03d \n", 5);           // Entier = 005
System.out.printf("Entier = %03d \n", 12);          // Entier = 012
System.out.printf("Entier = %03d \n", 123);         // Entier = 123
System.out.printf("Entier = %03d \n", 123456789);   // Entier = 123456789
```

b) Possibilités pour afficher les nombres flottants

Supposons que l'on souhaite afficher un nombre flottant avec seulement 2 chiffres après le point. Le format d'affichage sera alors `%.2f`. Notons que la variable affichée pourra être aussi bien de type *double* que *float*.

Exemple

```
double reel;        // nombre saisi par l'utilisateur

// TESTS AFFICHAGE DES NOMBRES REELS
System.out.print("Donnez un reel : ");
reel = entree.nextDouble();
System.out.printf("reel = %.2f \n", reel);

/*
 *  si 17,17999 est saisi, affiche 17,18
 *  si 17,965 est saisi, affiche 17,97
 *  si 17,964 est saisi, affiche 17,96
 *  si 14,4 est saisi, affiche 14,40
 */
```

Comme pour l'affichage d'un entier, on peut mentionner que l'on souhaite que des espaces soient insérés avant le premier chiffre du nombre flottant, ceci dans le but par exemple d'aligner des résultats sur la virgule qui sépare la partie entière de la partie décimale. Par exemple, si l'on souhaite que la partie entière soit affichée sur 5 positions, et la partie décimale sur 2, on utilisera le format `%8.2f`. Notons que le nombre 8 indiqué dans le format correspond au nombre total de caractères du nombre à afficher.

Exemple

```
// TEST AFFICHAGE DES REELS
System.out.printf("Reel = %8.2f \n", 1.234);        // Reel =     1,23
System.out.printf("Reel = %8.2f \n", 12.345);       // Reel =    12,35
System.out.printf("Reel = %8.2f \n", 12345.678);    // Reel = 12345,68
System.out.printf("Reel = %8.2f \n", 123456789.55); // Reel = 123456789,55


// TEST AFFICHAGE DES REELS avec 0 préalables
System.out.printf("Reel = %08.2f \n", 1.234);       // Reel = 00001,23
System.out.printf("Reel = %08.2f \n", 12.345);      // Reel = 00012,35
System.out.printf("Reel = %08.2f \n", 12345.678);   // Reel = 12345,68
System.out.printf("Reel = %08.2f \n", 123456789.55);// Reel = 123456789,55
```