Java printf para dar formato a los datos de salida

Vamos a ver cómo utilizar printf para dar formato a los datos que se imprimen por pantalla en Java. Este problema se nos plantea por ejemplo cuando queremos mostrar un número de tipo float o double con un número determinado de decimales y no con los que por defecto muestra Java.

A partir de la versión Java 5 se incorporan los métodos **format** y **printf** que permiten aplicar un formato a la salida de datos por pantalla. Ambos realizan la misma función, tienen exactamente el mismo formato y emulan la impresión con formato *printf() de C*.

Veamos primero varios ejemplos de printf en Java y después explicaremos en detalle la sintaxis de printf. Si queremos mostrar el número 12.3698 de tipo double con dos decimales:

System.out.printf("%.2f %n", 12.3698);

El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas, .2 indica el número de decimales. La f indica que el número es de tipo **float** o **double**. En la tabla que aparece más adelante podéis ver todos los caracteres de conversión para todos los tipos de datos. %n indica un salto de línea, equivale a \n. Con printf podemos usar ambos para hacer un salto de línea.

La salida por pantalla es: 12,37

Comprobamos que printf realiza un redondeo para mostrar los decimales indicados.
Lo más común será que tengamos el valor en una variable, en ese caso si queremos escribir el valor de n con tres decimales:

```
double n = 1.25036;
System.out.printf("%.3f %n", n);
```

Salida:
1,250

Para mostrar el signo + en un número positivo:

```
double n = 1.25036;
System.out.printf("%+.3f %n", n);
```
Salida:
+1.250

Si el número a mostrar es un **entero** se utiliza el carácter d:

```
int x = 10;
System.out.printf("%d %n", x);
```
Salida:

10
Para mostrarlo con signo:

```
int x = 10;
System.out.printf("%+d %n", x);
```
Salida:
+10

Para mostrar varias variables pondremos **tantos % como valores vamos a mostrar**. Las variables se escriben a continuación de las comillas separadas por comas:

```
double n = 1.25036;
int x = 10;
System.out.printf("n = %.2f x = %d %n", n, x);
```

Salida:
n = 1,25 x = 10

Cuando hay varias variables podemos indicar de cuál de ellas es el valor a mostrar escribiendo 1$, 2$, 3$, ... indicando que el valor a mostrar es el de la primera variable que aparece a continuación de las comillas, de la segunda, etc.

La instrucción anterior la podemos escribir así:

```
System.out.printf("n = %1$.2f x = %2$d %n", n, x);
```
O darle la vuelta así:

```
System.out.printf("x = %2$d n = %1$.2f %n", n, x);
```

Este número es opcional, si no aparece se entenderá que el primer valor proviene de la primera variable, el segundo de la segunda, etc.

Si queremos mostrar el número 123.4567 y su cuadrado ambos con dos decimales debemos escribir:

```
double n = 123.4567;
System.out.printf("El cuadrado de %.2f es %.2f\n", n, n*n);
```

Salida:

```
El cuadrado de 123,46 es 15241,56
```

printf permite mostrar valores con un ancho de campo determinado. Por ejemplo, si queremos mostrar el contenido de n en un ancho de campo de 10 caracteres escribimos:

```
double n = 1.25036;
System.out.printf("%+10.2f %n", n);
```
Salida:
bbbbb+1.25

Donde cada *b* indica un espacio en blanco.

El 10 indica el tamaño en caracteres que ocupará el número en pantalla. Se cuentan además de las cifras del número el punto decimal y el signo si lo lleva. En este caso el número ocupa un espacio de 5 caracteres (3 cifras, un punto y el signo) por lo tanto se añaden 5 espacios en blanco al principio para completar el tamaño de 10.

Si queremos que en lugar de espacios en blancos nos muestre el número completando el ancho con ceros escribimos:

System.out.printf("%+010.2f %n", n);

Salida:
+000001.25

Más ejemplos de printf:

Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.

double precio = 1.22;
System.out.printf("%10.2f", precio);

Salida:
*bbbbbb*1.22
(el carácter b indica un espacio en blanco)

El número ocupa un espacio total de 10 caracteres incluyendo el punto y los dos decimales.

Mostrar la cadena "Total:" con un ancho de 10 caracteres y alineada a la izquierda:

System.out.printf("%-10s", "Total:");

Salida:
Total:*bbbb*

El carácter s indica que se va a mostrar una cadena de caracteres.

El signo - indica alineación a la izquierda.

Mostrar la cadena "Total:" con un ancho de 10 caracteres y alineada a la derecha:

System.out.printf("%10s", "Total:");

Salida:

*bbbb*Total:

Al final puedes ver un ejemplo completo con distintos usos de printf.

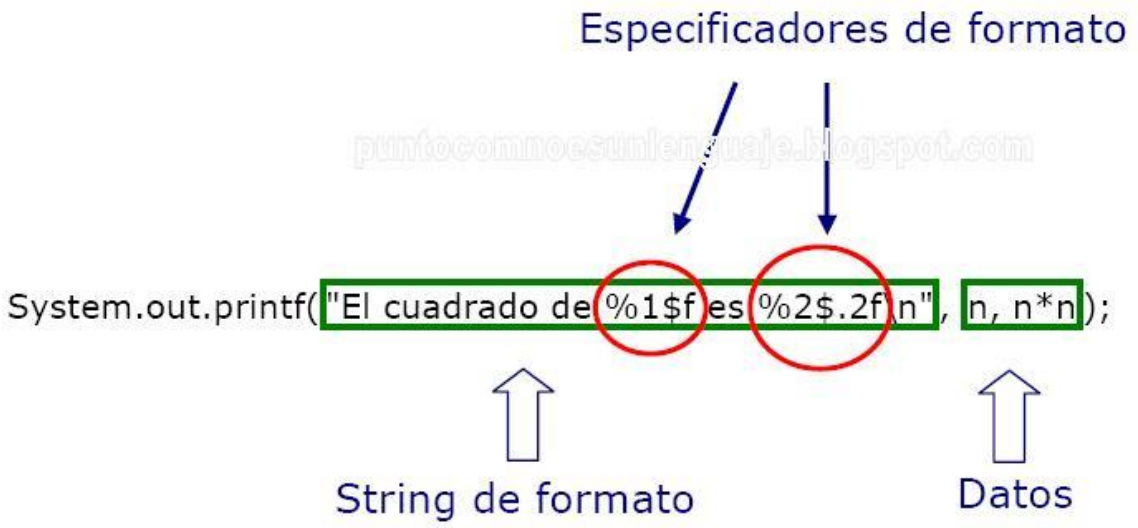
Veamos ahora detenidamente la sintaxis de printf:

La sintaxis general de printf es:

printf (String de formato, dato1,… daton);

El *String de formato* es una cadena de caracteres que contiene:
- **texto fijo** que será mostrado tal cual.
- **especificadores de formato** que determinan la forma en que se van a mostrar los datos.

Los *datos* representan la información que se va a mostrar y sobre la que se aplica el formato anterior. El número de datos que se pueden mostrar es variable.



Explicación de cada una de las partes que aparecen en la instrucción printf:

Especificadores de formato:

La sintaxis para los especificadores de formato de printf es:

%[posición_dato$][indicador_de_formato][ancho][.precisión]carácter_de_conversión

Los elementos entre corchetes son opcionales.

posición_dato: indica la posición del dato sobre el que se va aplicar el formato. El primero por la izquierda ocupa la posición 1.

indicador_de_formato: es el conjunto de caracteres que determina el formato de salida. Los indicadores de formato de printf en Java son:

INDICADORES DE FORMATO			
Indicador	**Significado**	**Indicador**	**Significado**
-	Alineación a la izquierda	+	Mostrar signo + en números positivos
(	Los números negativos se muestran entre paréntesis	0	Rellenar con ceros
,	Muestra el separador decimal		

ancho: Indica el tamaño mínimo, medido en número de caracteres, que debe ocupar el dato en pantalla.

.precisión: Indica el número de decimales que serán representados. Solo aplicable a datos de tipo float o double.

carácter_de_conversión: Carácter que indica cómo tiene que ser formateado el dato. Los más utilizados se muestran en la tabla.

CARACTERES DE CONVERSIÓN			
Carácter	**Tipo**	**Carácter**	**Tipo**
d	Número entero en base decimal	X, x	Número entero en base hexadecimal
f	Número real con punto fijo	s	String
E, e	Número real notación científica	S	String en mayúsculas
g	Número real. Se representará con notación científica si el número es muy grande o muy pequeño	C, c	Carácter Unicode. C: en mayúsculas

Ejemplo completo con distintos usos de printf en Java:

```
public class Printf2 {
        public static void main(String[] args) {

                double q = 1.0/3.0;

                System.out.printf("1.0/3.0 = %5.3f %n", q);
                System.out.printf("1.0/3.0 = %7.5f %n", q);

                q = 1.0/2.0;
```

```
                System.out.printf("1.0/2.0 = %09.3f %n", q);

                q = 1000.0/3.0;
                System.out.printf("1000/3.0 = %7.1e %n", q);

                q = 3.0/4567.0;
                System.out.printf("3.0/4567.0 = %7.3e %n", q);

                q = -1.0/0.0;
                System.out.printf("-1.0/0.0 = %7.2e %n", q);

                q = 0.0/0.0;
                System.out.printf("0.0/0.0 = %5.2e %n", q);

                System.out.printf("pi = %5.3f; e = %10.4f %n", Math.PI, Math.E);

                double r = 1.1;
                System.out.printf("%1$.5f *%1$.5f %n", Math.PI, r);
                System.out.printf("%1$6.6f *%1$6.6f %n", Math.PI, r);

                System.out.printf("C = 2 * %1$5.5f * %2$4.1f, "+"A = %2$4.1f * %2$4.1f *
%1$5.5f %n", Math.PI, r);
        }
}
```

Salida:

```
1.0/3.0 = 0,333
1.0/3.0 = 0,33333
1.0/2.0 = 00000,500
1000/3.0 = 3,3e+02
3.0/4567.0 = 6,569e-04
-1.0/0.0 = -Infinity
0.0/0.0 =   NaN
pi = 3,142; e =     2,7183
3,14159 *3,14159
3,141593 *3,141593
C = 2 * 3,14159 * 1,1, A = 1,1 * 1,1 * 3,14159
```

Java String format()

Esta clase permite generar String a partir de datos usando una especificación de formato.

La funcionalidad se presta a diferentes clases para un uso cómodo:

```
String s = String.format(formato, valores ...);
System.out.format(formato, valores ...);
```

Es muy similar a printf:

```
System.out.printf(formato, valores ...);
```

Se puede añadir un primer parámetro: Locale, que si no se usa es Locale.getDefault (). Se refiere al formato por defecto local.

Formatter (clase) java.util.Formatter

```
String saludo =
 String.format("Hola amigos, bienvenidos a %s !", "Madrid");
                System.out.println(saludo);

                saludo = String.format(
                                "Hola %2$s, bienvenidos  %1$s !",
                                "Madrid",
                                "chicos");
                System.out.println(saludo);

                int numero = 425;

                Formatter fmtr = new Formatter();

                fmtr.format("%08d", numero);
                System.out.println("El numero formateado " + fmtr);

                Object result = fmtr.format("%1$4d - el año %2$4.2f", 1951, Math.PI);
                System.out.println(result);

                // Otra forma de hacerlo
                System.out.format("%1$04d - el año %2$4.2f%n", 1951, Math.PI);

                fmtr.close();

                // Parecido al printf que ya vimos:
                System.out.printf("%1$04d - el año %2$4.2f%n", 1951, Math.PI);
                System.out.printf("PI es mas o menos %1$4.2f%n", Math.PI);
```

```
System.out.printf("Tu nombre es %1$s y el mio es %2$s%n", "Juan", "Pepi");
```

DecimalFormat: Formateador decimal en Java

La clase **DecimalFormat**: Es una **clase de Java** que nos permite mostrar los números con un formato deseado, puede ser limitando los dígitos decimales, si usaremos punto o coma, etc, incluso podemos tomar los valores desde un **JTextField** y reconstruir el número, ejemplo:

```
//import requerido para el Formateador
import java.text.DecimalFormat;
DecimalFormat formateador = new DecimalFormat("####.####");
// Imprime esto con cuatro decimales, es decir: 7,1234
System.out.println (formateador.format (7.12342383));
```

Si utilizamos ceros en lugar de los #, los dígitos no existentes se rellenarán con ceros, ejemplo:

```
DecimalFormat formateador = new DecimalFormat("0000.0000");
// Imprime con 4 cifras enteras y 4 decimales: 0001,8200
System.out.println (formateador.format (1.82));
```

También podemos utilizar el signo de porcentaje (%) en la máscara y así el número se multiplicará automáticamente por 100 al momento de Imprimir.

```
DecimalFormat formateador = new DecimalFormat("###.##%");
// Imprime: 68,44%
System.out.println (formateador.format(0.6844));
```

La clase **DecimalFormat** usa por defecto el formato para el lenguaje que tengamos instalado en el ordenador. Es decir, si nuestro sistema operativo está en español, se usará la coma para los decimales y el punto para los separadores de miles. Si estamos en inglés, se usará el punto decimal.

Una opción para cambiar esto, es utilizar la clase **DecimalFormatSymbols**, que vendrá con el idioma por defecto, y podremos cambiar en ella el símbolo que nos interese. Por ejemplo, si estamos en español y queremos usar el punto decimal en vez de la coma, podemos hacer lo siguiente:

```
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
simbolos.setDecimalSeparator('.');
```

```
DecimalFormat formateador = new DecimalFormat("####.####",simbolos);
// Imprime: 3.4324
System.out.println(formateador.format (3.43242383));
```

Reconstruyendo un Número:

Supongamos que leemos algún número desde consola o un **JTextField** y deseamos reconstruirlo con **DecimalFormat**, se hace de la siguiente forma:

```
JTextField textField = new JTextField();
DecimalFormat formateador = new DecimalFormat("####.####");
String texto = textField.getText();
try
{
  // parse() lanza una ParseException en caso de fallo que hay que capturar.
  double valor = formateador.parse(texto).doubleValue();
}
catch (ParseException e)
{
// Error. El usuario ha escrito algo que no se puede convertir a número.
}
```