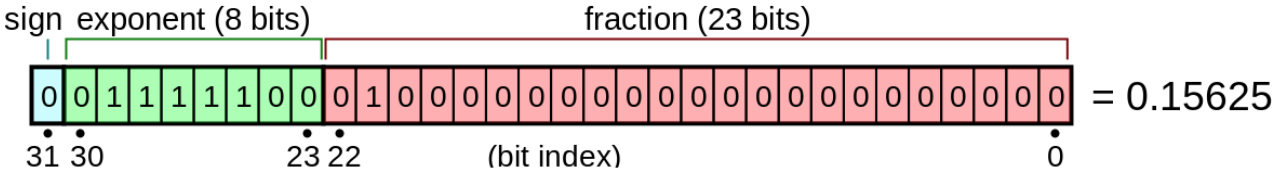
Anexo II .- Salida con formato

Hasta ahora hemos trabajado con números reales pero al imprimirlos en pantalla hemos podido observar que a veces se muestran con un número de decimales "inesperado". Eso se debe a cómo se guardan los números reales a un dispositivo electrónico.

La forma más habitual es utilizando la estándar IEEE 754 formato con coma flotante que consta de:

- Bits de signo (s): 1 bit
- Exponente desplazado (e): habitualmente 8 bits.
- Mantisa (t): habitualmente 23 bits.



El valor real final viene determinado por la siguiente fórmula:

$$v = (-1)^{b_{31}} \times (1, b_{22} b_{21} \ldots b_0)_2 \times 2^{e-127} = (-1)^{b_{31}} \times \left(1 + \sum_{i=0}^{22} b_{22-i} 2^{-i}\right) \times 2^{(e-127)}$$

dónde:

- b^0, b^1, …., b^{22} son bits de la mantisa.
- b^{31} es el bit que representa el signo, en este caso 0 (positivo)
- e es el valor del exponente, $2^{(e-127)} = 2^{(124-127)} = 2^{-3}$

por lo tanto:

- v = 1.25 x 2^{-3} = 0,15625

Para poder darle formado a la salida de datos por pantalla tenemos varías opciones, pero nos centraremos principalmente en dos:

- System.out.printf
- La clase DecimalFormat

System.out.printf

Desde la versión 1.5 de Java se incorpora el método printf heredado del lenguaje C y que tiene el mismo comportamiento.

La sintaxis general de printf es:

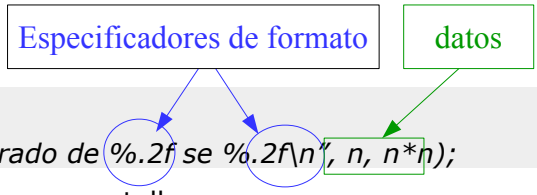
```
printf(String de formato, Object … datos);
```

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 la que se mostrarán los datos.

El Object datos representa la información que se va a mostrar y sobre la que se aplica el formato. El número de datos que se pueden mostrar es variable.

Por ejemplo:




```
float n = 45.2568f;
System.out.printf("El cuadrado de %.2f se %.2f\n", n, n*n);
```

mostraría el siguiente texto por pantalla:

```
El cuadrado de 45.26 se 2048.18
```

La sintaxis de los **especificadores de formato** es la siguiente:

```
%[posición_datos$][indicador_de_formato][anchura][.precisión]carácter_de_conversión
```

- Los elementos entre corchetes `[]` son opcionales.
- `posición_datos$` indica la posición del dato sobre la que va a aplicarse el formato. El primero por la izquierda ocupa la posición 1.
- `indicador_de_formato` es el conjunto de caracteres que determina el formado de salida. Los indicadores de formato de printf son:

Indicador	Significado	Indicador	Significado
-	Alineación a la izquierda	+	Mostrar signo + en números positivos
(	Los números negativos se muestran entre paréntesis	0	Rellenar con ceros (se tiene que indicar la anchura después).
,	Muestra el separador de miles		

- `anchura` indica el tamaño mínimo, medido en número de caracteres, que tiene que ocupar el dato en pantalla.
- `precisión` indica el número de decimales que serán representados. Solo aplicable a datos de tipos float o double.
- carácter_de_conversión indica como tiene que ser formateado el dato. Los caracteres de conversión más comunes son:

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 en notación científica	S	String en mayúsculas
g	Número real. Se representará en notación científica si el número es muy grande o muy pequeño.	C, c	Carácter Unicode. C: en mayúsculas

Ejemplo:

```
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 h%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("C = 2 * %1$5.5f * %2$4.1f, "+"A = %2$4.1f * %2$4.1f * %1$5.5f
%n",MATH.PI, r);
```

Salida por pantalla

```
1.0/3.0 = 0.333
1.0/3.0 = 0.33333
1.0/2.0 = 00000.500
1000/3.0 = 3.3e+02 h
3.0/4567.0 = 6.569e-04
-1.0/0.0 = -Infinity
0.0/0.0 =   NaN
pi = 3.142, e =     2.7183
C = 2 * 3.14159 *  1.1, A =  1.1 *  1.1 * 3.14159
```

La clase DecimalFormat

La clase DecimalFormat nos permite mostrar los números en pantalla con el formato que deseamos, por ejemplo, con dos decimales, con una coma para separar los decimales, etc.

Presentación de decimales redondeados

DecimalFormat permite presentar en pantalla el número que queramos con un número de decimales concreto, pero se tiene que tener claro que solo se trata de una representación visual, internamente el float o double sobre el que se aplique continuará teniendo el mismo valor.

Para crear un objeto de tipo DecimalFormat necesitamos importar la clase DecimalFormat del paquete java.texto

```
importe java.texto.DecimalFormat;
```

Para formatear la salida por pantalla se utilizan varios caracteres pero nos centraremos en #, 0 y el punto (.).

- # representa una cifra

- 0 representa también una cifra pero si faltan números (por delante o por detrás) se rellenará con ceros.
- El punto (.) se utiliza para representar la parte decimal.

Ejemplos de uso

```
//Crear el objeto de la clase DecimalFormat
DecimalFormat formateador = new DecimalFormat("#.##");
System.out.println(formateador.formato(3.43242383)); //imprimirá 3.43

DecimalFormat formateador = new DecimalFormat("0000.00");
System.out.println(formateador.formato(3.4)); //imprimirá 0003.4
```

Puntos decimales y separador de miles: DecimalFormatSymbols

La clase DecimalFormat usa por defecto el formato para el lenguaje que tengamos instalado al 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 crear una clase **DecimalFormatSymbols**, que vendrá con la configuración de el idioma por defecto, pero podremos cambiarla por la que nos interese. Por ejemplo, si estamos en español y queremos utilizar el punto decimal en lugar de la coma, podemos hacer:

```
importe java.texto.DecimalFormat;
importe java.texto.DecimalFormatSymbols;
...
DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
simbolos.setDecimalSeparator('.');
DecimalFormat formateador = new DecimalFormat("####.####",simbolos);
//Se mostrará con el punto decimal, es decir, 3.4324
System.out.println(formateador.formato(3.43242383));
```

También es posible coger el DecimalFormaSymbols de alguna localización concreta que nos interese y modificar lo que necesitamos. Por ejemplo, si nos interesa que la coma decimal sea un punto en lugar de una coma, podríamos coger el DecimalFormatSymbols de Inglaterra.

```
DecimalFormatSymbols simbolos = DecimalFormatSymbols.getInstance(Locale.ENGLISH);
DecimalFormat formateador = new DecimalFormat("####.####",simbolos);
```

CUIDADO! esto cambia todo, también cosas como la moneda (libras esterlinas o euros), etc.