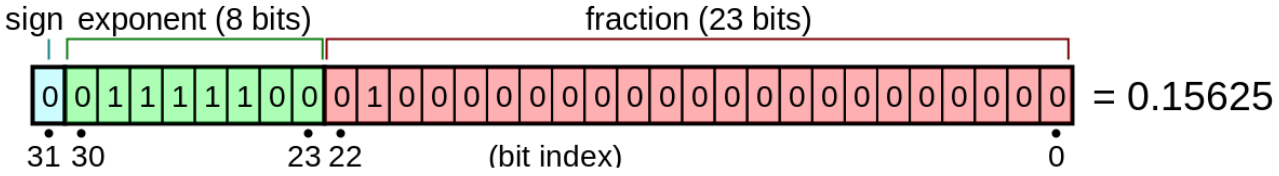
Annex II.- Sortida amb format

Fins ara hem treballat amb números reals però al imprimir-los en pantalla hem pogut observar que de vegades es mostren amb un nombre de decimals "inesperat". Això és per com es guarden els números reals a un dispositiu electrònic.

La forma més habitual és utilitzant l'estandar IEEE 754 format amb coma flotant que consta de:

- Bits de signe (s): 1 bit
- Exponent desplaçat (e): habitualment 8 bits.
- Mantissa (t): habitualment 23 bits.



El valor real final ve determinat per la següent fórmula:

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

on:

- b^0, b^1, …., b^{22} són bits de la mantissa.
- b^{31} és el bit que representa el signe, en aquest cas 0 (positiu)
- e és el valor del exponent, $2^{(e-127)} = 2^{(124-127)} = 2^{-3}$

per tant:

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

Per a poder donar-li format a la surtida de dades per pantalla tenim varies opcions, però ens centrarem principalment en dos:

- System.out.printf
- La classe DecimalFormat

System.out.printf

Desde la versió 1.5 de Java s'incorpora el mètode printf heretat del llenguatge C i que té el mateix comportament.

La sintaxis general de printf és:

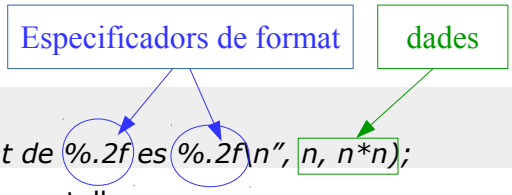
```
printf(String de format, Object … dades);
```

L'String de format és una cadena de caràcters que conté:

- Text fixe que serà mostrat tal qual.
- Especificadors de format que determinen la forma en la que es mostraran les dades.

L'Object dades representa informació que es va mostrar i sobre la que s'aplica el format. El nombre de dades que es poden mostrar és variable.

Per example:




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

mostraria el següent text per pantalla:

```
El quadrat de 45.26 es 2048.18
```

La sintaxi dels **especificadors de format** és la següent:

```
%[posició_dades$][indicador_de_format][amplària][.precisió]caràcter_de_conversió
```

- Els elements entre claudàtors `[]` són opcionals.
- `posició_dades$` indica la posició de la dada sobre la que va a aplicar-se el format. El primer per l'esquerra ocupa la posició 1.
- `indicador_de_format` és el conjunt de caracters que determina el format de sortida. Els indicadors de format de printf són:

Indicador	Significat	Indicador	Significat
-	Alineació a l'esquerre	+	Mostrar signe + en números positius
(	El números negatius es mostren entre parèntesi	0	Omplir amb zeros (s'ha de indicar l'amplària després).
,	Mostra el separador de milers		

- `amplària` indica el tamany mínim, mesurat en nombre de caracters, que ha d'ocupar la dada en pantalla.
- `precisió` indica el nombre de decimals que seran representats. Sols aplicable a dades de tipus float o double.
- caràcter_de_conversió indica com ha de ser formatada la dada. Els caràcters de conversió més comuns són:

Caràcter	Tipus	Caràcter	Tipus
d	Número enter en base decimal	X, x	Número enter en base hexadecimal
f	Número real amb punt fixe	s	String
E, e	Número real en notació científica	S	String en majúscules
g	Número real. Es representarà en notació científica si el número és molt gran o molt menut.	C, c	Caràcter Unicode. C: en majúscules

Exemple:

```
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);
```

Sortida per 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 classe DecimalFormat

La classe DecimalFormat ens permet mostrar els números en pantalla amb el format que desitgem, per example, amb dos decimals, amb una coma per separar els decimals, etc.

Presentació de decimals arrodonits

DecimalFormat permet presentar en pantalla el número que vulguem amb un nombre de decimals concret, però s'ha de tindre clar que només es tracta d'una representació visual, internament el float o double sobre el que s'aplique continuarà tenint el mateix valor.

Per a crear un objecte de tipus DecimalFormat necessitem importar la classe DecimalFormat del paquet java.text

```
import java.text.DecimalFormat;
```

Per a formatar la sortida per pantalla s'utilitzen diversos caràcters però ens centrarem en #, 0 i el punt (.).

- # representa una xifra

- 0 representa també una xifra però si falten números (per davant o per darrere) s'omplirà amb zeros.
- El punt (.) s'utilitza per representar la part decimal.

Exemples d'ús

```
//Crear l'objecte de la classe DecimalFormat
DecimalFormat formateador = new DecimalFormat("#.##");
System.out.println(formateador.format(3.43242383)); //imprimirà 3.43

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

Punts decimals i separador de milers: DecimalFormatSymbols

La classe DecimalFormat fa servir per defecte el format per al llenguatge que tinguem instal·lat a l'ordinador. És a dir, si el nostre sistema operatiu està en espanyol, es farà servir la coma per als decimals i el punt per als separadors de milers. Si estem en anglès, es farà servir el punt decimal.

Una opció per a canviar això, és crear una classe **DecimalFormatSymbols**, que vindrà amb la configuració de l'idioma per defecte, però podrem canviar-la per la que ens interesse. Per exemple, si estem en espanyol i volem utilitzar el punt decimal en lloc de la coma, podem fer:

```
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
...
DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
simbolos.setDecimalSeparator('.');
DecimalFormat formateador = new DecimalFormat("####.####",simbolos);
//Mostrarà amb el punt decimal, es a dir, 3.4324
System.out.println(formateador.format(3.43242383));
```

També és possible agafar el DecimalFormaSymbols d'alguna localització concreta que ens interesse i modificar o no el que necessitem. Per exemple, si ens interessa que la coma decimal siga un punt en comptes d'una coma, podríem agafar el DecimalFormatSymbols d'Anglaterra.

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

COMPTE! això canvia tot, també coses com la moneda (lliures esterlines o euros), etc.