



Saber más.....

- Librerías de interés
 - Repaso clase String
 - Clase Math
 - Números aleatorios
 - Trabajo con fechas



Clase String

- El tipo cadena “**cadena de caracteres**” es una clase especial que permite instanciar objetos sin un constructor específico.
- Como objeto que es tiene métodos que nos facilitan su manejo y en ocasiones su tratamiento difiere del que podemos dar a un dato de tipo primitivo (como la comparación de cadenas)

Algunos métodos de la clase String

- **charAt(**n**)**

Nos permite obtener el carácter que ocupa la posición **n** de una cadena (empezando por la izquierda la primera posición es la 0)

Ej.: String cad;

cad= teclado.next();

car=cad.charAt(0); //primer carácter de cad

Algunos métodos de la clase String

- **equals(cadena)**

Compara la cadena con el parámetro que se le pasa

```
Ej.: String cad="Hola";  
      if(cad.equals("Hola"))  
          System.out.print("Son iguales");
```

- **equalsIgnoreCase(cadena)**

No distingue entre mayúsculas y minúsculas


equals

- ¿por qué no se utiliza == para comparar cadenas?
- Porque == compara OBJETOS y lo que nos dice es si 2 objetos son iguales, no si su “contenido” es igual.
- En ocasiones funciona porque depende de cómo se hayan instanciado los objetos cadena, el compilador decide que ya existe una instancia al declarar el segundo y simplemente la segunda apunta a la primera y se trata del mismo objeto (mismo espacio de memoria)



Clases envolventes de los tipos primitivos

- Cada tipo primitivo en java tiene su correspondiente clase envolvente cuyos objetos pueden almacenar un valor del tipo primitivo correspondiente.
- Estos objetos se suelen utilizar para utilizar sus métodos asociados que permiten la conversión de unos tipos primitivos a otros o desde una cadena de caracteres a un número y viceversa

- 
- Igualmente se utilizan en las colecciones de datos que sólo pueden contener objetos.
 - Las clases envolventes son Integer, Boolean, Character, Long, Float y Double.
 - Tienen el mismo nombre pero con la primera letra en mayúsculas.
 - Ej de método:
Float.parseFloat(cad) Convierte un String, cad, en un float

ENTRADA / SALIDA

- Lectura de 1 carácter:
 - Leemos una palabra y nos quedamos con el primer carácter (el que ocupa la posición 0)
 - `teclado.next().charAt(0);`
- Lectura de un real (float o double)
 - Al leer un real en lugar de aceptar el punto decima espera una coma, o da error.
 - Para evitarlo leemos una cadena y la convertimos en el tipo que nos interese
 - Ej.: `Double.parseDouble(teclado.next())`

Salida

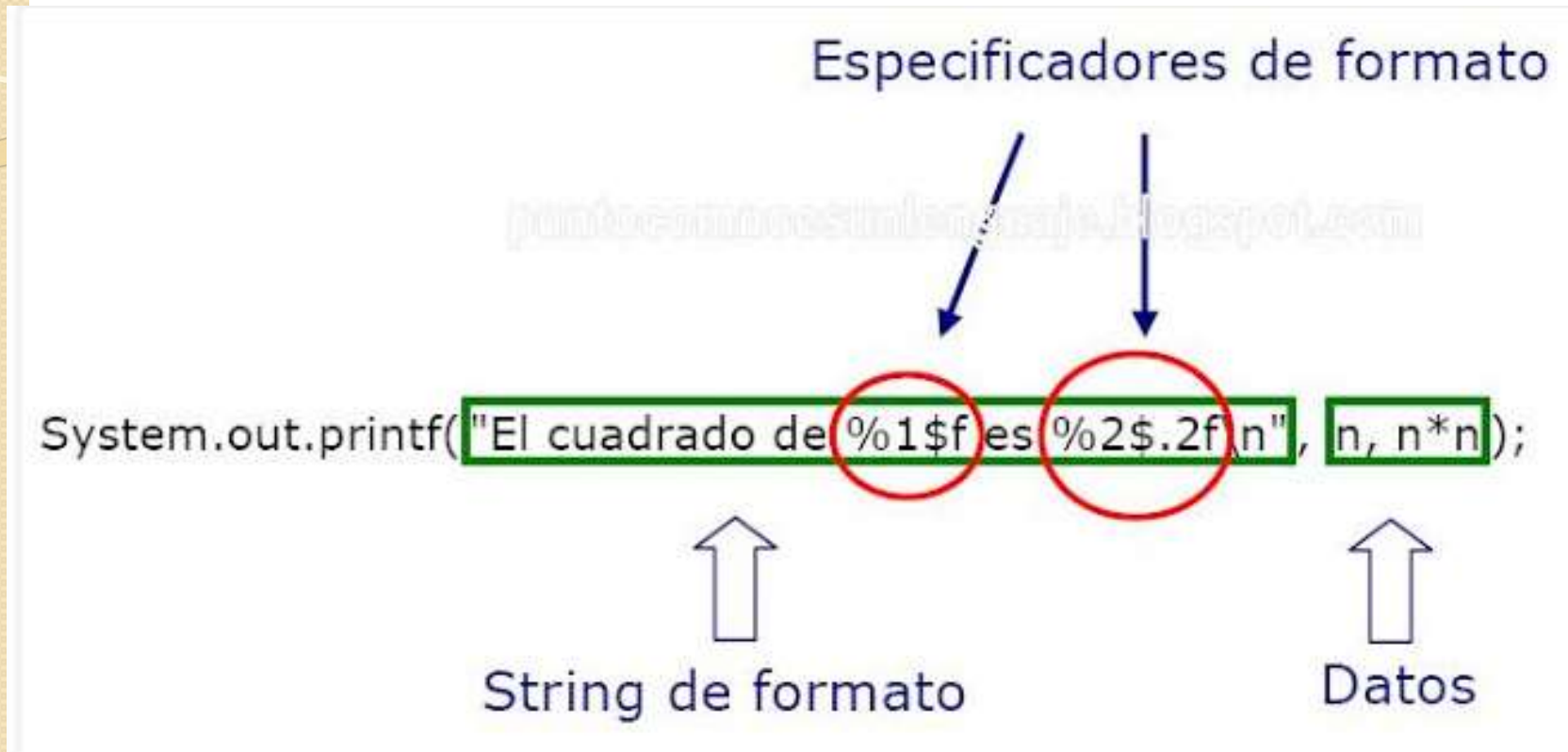
La sintaxis para los especificadores de formato de printf es:

`%[posición_dato$][indicador_de_formato][ancho][.precision]carácter_de_conversión`

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		

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

printf



`posicion$` indica el 'ordinal' del parámetro que viene a continuación.

Si no aparece, por defecto los formatos se corresponden con los parámetros en el mismo orden de aparición

Algunas funciones predefinidas. La clase Math

- Las constantes E y PI
Math.E=2.7182818284590452354
Math.PI=3.14159265358979323846
- Las funciones de redondeo, con x de tipo double:
 - `ceil(x)` : devuelve el número entero más pequeño que es mayor o igual a x
 - `floor(x)` : devuelve el número entero más grande que es menor o igual a x
 - `round(x)` : convierte el real x al entero más próximo
- Funciones trigonométricas
 - `sin(x)` : calcula el seno del ángulo (en radianes) x
 - `cos(x)` : calcula el coseno del ángulo (en radianes) x
 - `asin(x)` : calcula el arco seno del ángulo x (x entre -1 y 1)
 - `acos(x)` : calcula el arco coseno del ángulo x (x entre -1 y 1)
 - `atan(x)` : calcula el arco tangente del ángulo x

Algunas funciones predefinidas. La clase Math

- Otras funciones
 - `abs(x)` : calcula el valor absoluto de x (entero o real)
 - `exp(x)` : calcula e elevado a x (x es real)
 - `log(x)` : calcula el logaritmo natural de x (x real y no negativo)
 - `max(x,y)` : compara los números x e y (enteros o reales) y devuelve el mayor
 - `min(x,y)` : compara los números x e y (enteros o reales) y devuelve el menor
 - **`pow(x,y)`** : calcula x elevado a y. No está definida si x es negativo o 0 e y no es entero, ni tampoco si x=0 e y es negativo o 0
 - **`random()`** : genera un número (pseudo)aleatorio entre 0.0 y 1.0
 - **`sqrt(x)`** : calcula la raíz cuadrada de x (x no negativo)

Obtener números aleatorios.

- El método `Math.random()` devuelve un número double pseudoaleatorio entre 0 y 1
 - A partir de este método ajusto los límites y convertimos a entero
 - `(int) (Math.random() * valorLimite + valorInicial)`

Por ejemplo, para obtener un número del 1 al 6

```
dado = (int) ( Math.random( ) * 6 + 1 );
```

Obtener números aleatorios. Otra opción

- Para que Java genere números aleatorios podemos usar la clase Random

```
import java.util.Random;
...
Random rnd = new Random( );
//creamos un objeto Random

int numero =rnd.nextInt(1000);
//solicitamos un número entero entre 0 y 1000. Lo guardamos en la
variable numero

rnd.setSeed(new Date().getTime());
// establecemos la semilla a partir de la cual se generarán los
números pseudo-aleatorios a partir de la hora del sistema
//Esto solo lo haremos una vez
```




Trabajo con fechas

Aquí tenéis una muy buena explicación sobre el trabajo con fechas, con ejemplos detallados

java.time