Java Scanner para lectura de datos

La clase Scanner está disponible a partir de Java 5 y facilita la lectura de datos en los programas Java.
Primero veremos varios **ejemplos de lectura de datos en Java con Scanner** y después explicaremos en detalle como funciona.
Para utilizar Scanner en el programa tendremos que hacer lo siguiente:

1. Escribir el import

La clase Scanner se encuentra en el paquete java.util por lo tanto se debe incluir al inicio del programa la instrucción:

```
import java.util.Scanner;
```

2. Crear un objeto Scanner

Tenemos que crear un objeto de la clase Scanner asociado al dispositivo de entrada.
Si el dispositivo de entrada es el teclado escribiremos:

```
Scanner sc = new Scanner(System.in);
```

Se ha creado el objeto *sc* asociado al teclado representado por *System.in*
Una vez hecho esto podemos leer datos por teclado.

Ejemplos de lectura:

Para leer podemos usar el método nextXxx() donde Xxx indica en tipo, por ejemplo nextInt() para leer un entero, nextDouble() para leer un double, etc.

Ejemplo de lectura por teclado de un número entero:

```
int n;
System.out.print("Introduzca un número entero: ");
n = sc.nextInt();
```

Ejemplo de lectura de un número de tipo double:

```
double x;
System.out.print("Introduzca número de tipo double: ");
x = sc.nextDouble();
```

Ejemplo de lectura de una cadena de caracteres:

```
String s;
System.out.print("Introduzca texto: ");
s = sc.nextLine();
```

Ejemplo de programa Java con lectura de datos con Scanner:
El programa pide que se introduzca el nombre de la persona y lo muestra por pantalla. A continuación lee por teclado el radio de una circunferencia de tipo double y muestra su longitud. Además lee un entero y muestra su cuadrado.

```
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);  //crear un objeto Scanner
        String nombre;
        double radio;
        int n;
        System.out.print("Introduzca su nombre: ");
        nombre = sc.nextLine();  //leer un String
        System.out.println("Hola " + nombre + "!!!");
        System.out.print("Introduzca el radio de la circunferencia: ");
        radio = sc.nextDouble(); //leer un double
        System.out.println("Longitud de la circunferencia: " + 2*Math.PI*radio);
        System.out.print("Introduzca un número entero: ");
        n = sc.nextInt(); //leer un entero
        System.out.println("El cuadrado es: " + Math.pow(n,2));
   }
}
```

Funcionamiento la clase Java Scanner.

De forma resumida podemos decir que cuando se introducen caracteres por teclado, el objeto Scanner toma toda la cadena introducida y la divide en elementos llamados **tokens**.
El carácter predeterminado que sirve de separador de tokens es el espacio en blanco.
Por ejemplo, si introducimos:
Esto es un ejemplo, lectura de datos.
Scanner divide la cadena en los siguientes tokens:
Esto
es
un
ejemplo,
lectura
de
datos.

Si introducimos la cadena:

12 20.001 Lucas w

Los tokens que se crean son:

12

20.001
Lucas
w

A continuación, utilizando los métodos que proporciona la clase Scanner se puede acceder a esos tokens y trabajar con ellos en el programa.

Ya hemos visto el método nextXxx(). Además la clase Scanner proporciona otros métodos, algunos de los métodos más usados son:

METODO	DESCRIPCIÓN
nextXxx()	Devuelve el siguiente token como un tipo básico. Xxx es el tipo. Por ejemplo, nextInt() para leer un entero, nextDouble para leer un double, etc.
next()	Devuelve el siguiente token como un String.
nextLine()	Devuelve la línea entera como un String. Elimina el final \n del buffer
hasNext()	Devuelve un boolean. Indica si existe o no un siguiente token para leer.
hasNextXxx()	Devuelve un boolean. Indica si existe o no un siguiente token del tipo especificado en Xxx, por ejemplo hasNextDouble()
useDelimiter(String)	Establece un nuevo delimitador de token.

Cómo limpiar el buffer de entrada en Java

Cuado en un programa se leen por teclado datos numéricos y datos de tipo carácter o String debemos tener en cuenta que al introducir los datos y pulsar intro estamos también introduciendo en el buffer de entrada el intro.

Es decir, cuando en un programa introducimos un datos y pulsamos el intro como final de entrada, el carácter intro también pasa al buffer de entrada.

Buffer de entrada si se introduce un 5: 5\n
En esta situación, la instrucción:

```
n = sc.nextInt();
```

Asigna a n el valor 5 pero el intro permanece en el buffer
Buffer de entrada después de leer el entero: \n

Si ahora se pide que se introduzca por teclado una cadena de caracteres:

```
System.out.print("Introduzca su nombre: ");
nombre = sc.nextLine();  //leer un String
```

El método nextLine() extrae del buffer de entrada todos los caracteres hasta llegar a un intro y elimina el intro del buffer.

En este caso asigna una cadena vacía a la variable nombre y limpia el intro. Esto provoca que el programa no funcione correctamente, ya que no se detiene para que se introduzca el nombre.

Solución:

Se debe **limpiar el buffer** de entrada si se van a leer datos de tipo carácter a continuación de la lectura de datos numéricos.

La forma más sencilla de limpiar el buffer de entrada en Java es ejecutar la instrucción:

sc.nextLine();

Lo podemos comprobar si cambiamos el orden de lectura del ejemplo y leemos el nombre al final:

```
import java.util.Scanner;
public class JavaApplication335 {
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nombre;
        double radio;
        int n;
        System.out.print("Introduzca el radio de la circunferencia: ");
        radio = sc.nextDouble();
        System.out.println("Longitud de la circunferencia: " + 2*Math.PI*radio);
        System.out.print("Introduzca un número entero: ");
        n = sc.nextInt();
        System.out.println("El cuadrado es: " + Math.pow(n,2));
        System.out.print("Introduzca su nombre: ");
        nombre = sc.nextLine();  //leemos el String después del double
        System.out.println("Hola " + nombre + "!!!");
    }
}
```

Si lo ejecutamos obtendremos:
Introduzca el radio de la circunferencia: 34
Longitud de la circunferencia: 213.62830044410595
Introduzca un número entero: 3
El cuadrado es: 9.0
Introduzca su nombre: Hola !!!

Comprobamos que no se detiene para pedir el nombre.
La solución es escribir la instrucción
sc.nextLine();
después de la lectura del int y antes de leer el String:

```
import java.util.Scanner;
public class JavaApplication335 {
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nombre;
        double radio;
        int n;
        System.out.print("Introduzca el radio de la circunferencia: ");
        radio = sc.nextDouble();
        System.out.println("Longitud de la circunferencia: " + 2*Math.PI*radio);
        System.out.print("Introduzca un número entero: ");
        n = sc.nextInt();
        System.out.println("El cuadrado es: " + Math.pow(n,2));
        sc.nextLine();
        System.out.print("Introduzca su nombre: ");
        nombre = sc.nextLine();
        System.out.println("Hola " + nombre + "!!!");
    }
}
```

Ahora la ejecución es correcta:
Introduzca el radio de la circunferencia: 23
Longitud de la circunferencia: 144.51326206513048
Introduzca un número entero: 5
El cuadrado es: 25.0
Introduzca su nombre: Lucas
Hola Lucas!!!

Java printf para dar formato a los datos de salida

Vamos a ver como 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 podeis ver todos los carateres 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 caracter 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 cual 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);

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 caracter 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:");

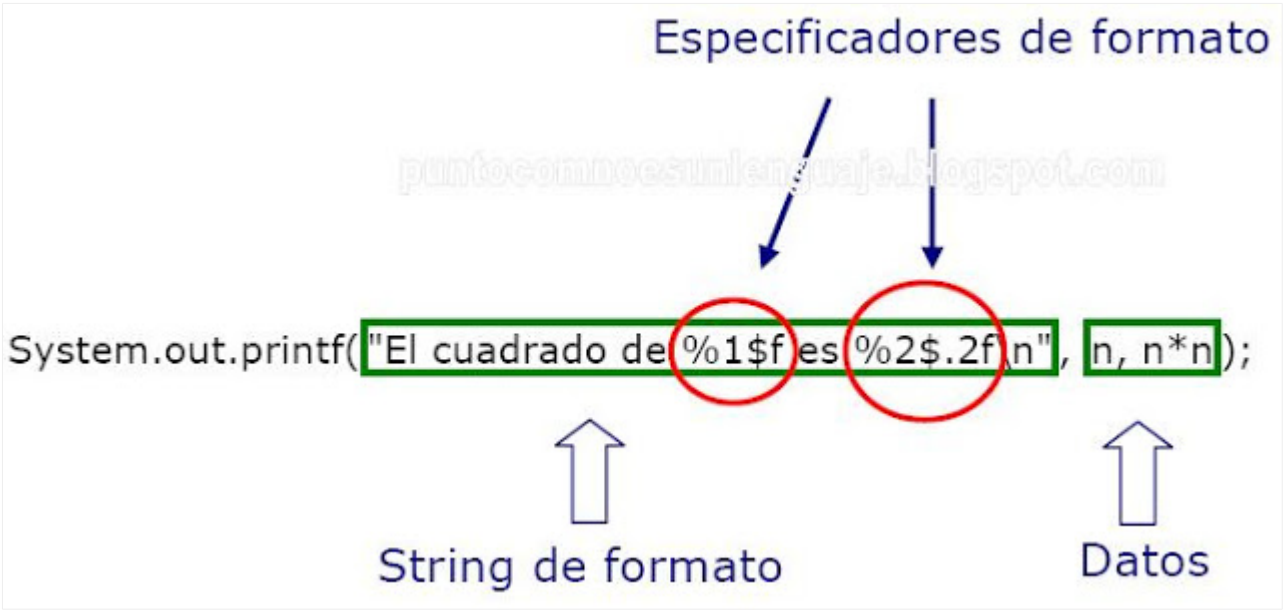
Salida:
*bbbb*Total:

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, 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 que se van mostrar los datos.
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.



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][.precision]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.

.precision: 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 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 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:

```
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
```