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 cómo funciona.
Para utilizar Scanner en un 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 tal y como se muestra en los ejemplos más abajo.

3. Cerrar el objeto Scanner

Al terminar de usar el objeto, cerramos el Scanner (solo una vez en el programa):

```
sc.close();
```

Ejemplos de lectura:

Para leer podemos usar el método nextXxx() donde Xxx indica el 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 (una línea hasta intro):

```
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/linea
			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));

			sc.close();
	}
}
```

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 (los guarda en un buffer interno).

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

Cuando 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 dato 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 Ejemplo {
	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.

Una solución es escribir la instrucción

```
sc.nextLine();
```

Después de la lectura del int y antes de leer el String:

```
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!!!

Hay una solución más elegante: leer siempre líneas con sc.nextLine(); y realizar la conversión al tipo correspondiente después:

```
import java.util.Scanner;
public class Scanner4 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		String nombre, radioS, nS;
		double radio;
		int n;

		System.out.print("Introduzca el radio de la circunferencia: ");

		radioS = sc.nextLine();
		radio = Double.parseDouble(radioS);

		System.out.println("Longitud de la circunferencia: " + 2*Math.PI*radio);
		System.out.print("Introduzca un numero entero: ");

		nS = sc.nextLine();
		n = Integer.parseInt(nS);

		System.out.println("El cuadrado es: " + Math.pow(n,2));

		System.out.print("Introduzca su nombre: ");
		nombre = sc.nextLine(); //leemos el String despues del double
		System.out.println("Hola " + nombre + "!!!");

		sc.close();
	}
}
```

Vamos a ver otro problema con Scanner. Lo siguiente **fallaría**:

```
//Get nombre
        System.out.println("Introduzca nombre del paciente");
        Scanner name_scanner = new Scanner (System.in);
        nombre=name_scanner.nextLine();
        name_scanner.close();

//Get apellido 1
        System.out.println("Introduzca primer apellido del paciente");
        Scanner apell1_scanner = new Scanner (System.in);
        apell1=apell1_scanner.nextLine();
        apell1_scanner.close();
```

Razón:

Cada vez que ejecutamos el `.close()` en un `Scanner`, esto también ejecuta `.close()` a `System.in`, lo que invalida todos los demás scanners que intentan leer del `System.in`.

No hay que crear una instancia diferente de `Scanner` cada vez que necesitemos interactuar con el usuario. Solo necesitamos una instancia de `Scanner` que podemos seguir usando a lo largo del programa (sin cerrarlo en ningún momento).