**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.

**3.** **Utilizar el Scanner**

Para leer datos desde teclado con Scanner podemos usar los métodos nextXxx() donde Xxx indica el tipo de dato a leer:

* **nextByte()** para leer un dato de tipo byte.
* **nextShort()** para leer un dato de tipo short.
* **nextInt()** para leer un dato de tipo int.
* **nextLong()** para leer un dato de tipo long.
* **nextFloat()** para leer un dato de tipo float.
* **nextDouble()** para leer un dato de tipo double.
* **nextBoolean()** para leer un dato de tipo boolean.
* **nextLine()** para leer un String hasta encontrar un salto de línea.
* **next()** para leer un String hasta el primer delimitador, generalmente hasta un espacio en blanco o hasta un salto de línea.

**Ejemplos de lectura:**

**Ejemplo** de lectura por teclado de un número entero (tipo int):

int n;

System.out.print("Introduzca un número entero: ");

n = sc.nextInt(); //asigna a la variable n el número entero introducido por teclado

**Ejemplo** de lectura de un número de tipo double:

double x;

System.out.print("Introduzca número de tipo double: ");

x = sc.nextDouble(); //asigna a la variable x el número double introducido por teclado

**Ejemplo** de lectura de una cadena de caracteres (String):

String s;

System.out.print("Introduzca texto: ");

s = sc.nextLine(); //asigna a la variable s el String introducido por teclado

**Ejemplo** de lectura de un número de tipo long:

long ln;

System.out.print("Introduzca un número: ");

ln = sc.nextLong(); //asigna a la variable ln el número long introducido por teclado

Si el valor introducido por teclado no es del tipo esperado o de un tipo compatible al esperado, se produce un error. En este caso se lanza la excepción **InputMismatchException**

**Ejemplo completo de programa Java con lectura de datos con Scanner**:

El programa pide al usuario que introduzca por teclado su nombre y lo muestra por pantalla. A continuación pide que se introduzca por rteclado el valor del radio de una circunferencia de tipo double y muestra la longitud de la circunferencia. Además pide que se introduzca por teclado un número entero y muestra su cuadrado.

import java.util.Scanner; //import de la clase Scanner

public class Ejemplo1Scanner {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in); //Se crea 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 interno de la clase Java Scanner.**

Es importante conocer el funcionamiento interno de la clase Scanner para poder realizar las operaciones de lectura de forma correcta.

De forma resumida, el proceso de lectura por teclado en un programa Java es el siguiente:

* Los datos que se introducen desde teclado se almacenan en una zona de memoria que vamoso a llamar *buffer*.
* Mediante un **stream** estos datos pasan al programa.

Un **stream** o ***flujo de datos*** es un objeto que hace de intermediario entre el programa y el origen o el destino de los datos. El programa lee del stream o escribe en él, sin importarle de donde proceden los datos físicamente o hacia qué dispositivo se dirigen realmente.  
Un stream está formado por una secuencia de bytes utilizados para la entrada o salida de un programa.  
Java crea de forma automática los siguientes streams cuando se ejecuta un programa:

**System.in**: stream de entrada conectado al **teclado**

**System.out**: stream de salida conectado al **monitor**

**System.err**: stream de salida conectado al **monitor** para mensajes de error.

[Diagrama, Texto

Descripción generada automáticamente](https://1.bp.blogspot.com/-eTSasrZ64oo/YDVjgwTTFRI/AAAAAAAAAyk/KSO-QLrCT8Um7XInZWqXy64AhQJxibGUACLcBGAsYHQ/s684/Scanner%2Bstream%2Bsystem.in.png)

Además de estos *streams* estándar, java proporciona una gran cantidad de clases para streams que permiten leer y escribir en ficheros.

* Cuando en el programa aparece una instrucción para leer un dato por teclado, se accede al buffer de entrada en busca del dato. Si lo encuentra, lo extrae del buffer y lo incorpora al programa.
* Si el dato no ha sido encontrado en el buffer generalmente se deberá a que el buffer está vacío. Esta es la situación más habitual. En este caso el programa espera a que el usuario introduzca el dato por teclado. Cuando el usuario lo introduce y pulsa intro entonces se extrae el valor introducido del buffer y se incorpora al programa.
* Si el dato encontrado no es del tipo esperado o de un tipo compatible se produce un error. En este caso se lanza la excepción *InputMismatchException*. Esto se produce, por ejemplo, cuando se está ejecutando el método nextInt() para extraer un entero del buffer y se introduce por ejemplo un double o un char. El tipo de dato encontrado en el buffer no coincide con el tipo que se quiere leer.

SALIDAS DE DATOS EN JAVA

La salida de datos en Java se realiza a través de la clase System.out. y tiene varios métodos para imprimir diferentes tipos de datos, como enteros, cadenas, caracteres, etc.

Los métodos más comunes son:

* print(String s): imprime una cadena de texto en la salida estándar.
* println(String s): imprime una cadena de texto en la salida estándar y agrega un salto de línea al final.
* printf(String format, Object… args): imprime una cadena de texto formateada en la salida estándar.

## Método print en Java

El método **print()** es uno de los métodos principales de la clase **PrintStream** en Java, que se utiliza para imprimir una cadena de texto en la salida estándar (generalmente en la consola). Este método no agrega un salto de línea al final de la cadena de texto impresa, por lo que si se llama varias veces seguidas, todas las cadenas de texto se imprimirán en la misma línea.

Una de las ventajas de usar el método **print()** es que se pueden concatenar varias cadenas en una sola línea. También se pueden imprimir otros tipos de datos, como números o caracteres, utilizando los métodos **print()** correspondientes, como print(int), print(double), print(char), etc.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

## **Método println en Java**

El método **println()** en Java es muy similar al método **print()**, con la diferencia de que agrega automáticamente un salto de línea al final de lo que se está imprimiendo.

Esto significa que cada vez que se utiliza el método **println()**, el cursor se moverá a la siguiente línea después de imprimir el contenido especificado.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza mediaAquí te dejo algunos ejemplos de cómo utilizar el método**println()** en Java: