SCANNER EN JAVA

IMPORTAR SCANNER

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

CREACION DE OBJETO

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

NOTAS RAPIDAS

- System.in = flujo de entrada estándar (teclado).
- Si mezclas nextInt() con nextLine(), usa sc.nextLine() después para limpiar el salto de línea.

PROBLEMA CON SALTO DE LINEA

- Los métodos como nextInt() o nextDouble() solo consumen el dato numérico, pero no consumen el salto de línea (\n) que el usuario ingresa al pulsar Enter.
- Luego, si se usa nextLine(), este lee ese salto de línea restante como una línea vacía y el programa "se salta" esa entrada.

SOLUCIONES

Consumir la línea sobrante manualmente. Esta estrategia evita que nextLine() capture accidentalmente una línea vacía y consuma el \n sobrante

```
int num = sc.nextInt();
sc.nextLine();
String texto = sc.nextLine();
```

Leer todo como texto y convertir

```
String linea = sc.nextLine();
int num = Integer.parseInt(linea);
```

LECTURA

TIPO	EJEMPLO
ENTERO	int num = sc.nextInt();
DECIMAL	double decimal = sc.nextDouble()
STRING	String character = sc.nextLine()

QUE ES SYSTEM.IN?

El System.in es un único flujo de entrada de la consola. Solo existe uno.

- Stream (flujo) → es como una tubería por donde pasan datos en orden, uno tras otro.
- Ejemplo mental → los datos son como gotas de agua en una tubería: llegan de forma continua y se van leyendo en secuencia.

InputStream (entrada)	Si el agua fluye hacia adentro y además lee los datos
OutputStream (salida)	Envia los datos, como si el agua fluye hacia afuera

PROBLEMAS AL CREAR VARIOS SCANNER(SYSTEM.IN) EN UN PROYECTO

- Todos usan la misma tubería (System.in) → se pelean por los datos.
- Un Scanner puede robar entradas que iban para el otro.
- Si llamas scanner.close() → se cierra también System.in, osea todos los flujos y ya no sirve el teclado.
- Crear un nuevo Scanner(System.in) después de cerrar el primero → falla.

Error típico:

- NoSuchElementException: No line found → pasa porque System.in ya fue cerrado o consumido.

Consejo final: Usar un solo Scanner(System.in).

- Centralización → solo un Scanner, más ordenado.
- Reutilización → no repites validaciones ni código en todas partes.
- Menos errores → evitas peleas entre Scanners y cierres de System.in.