

Programación Java

Tutorial Java. Aprende a programar con Java desde cero.

Java Scanner para lectura de datos

Actualizado 2022

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 teclado 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 vamos 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.



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

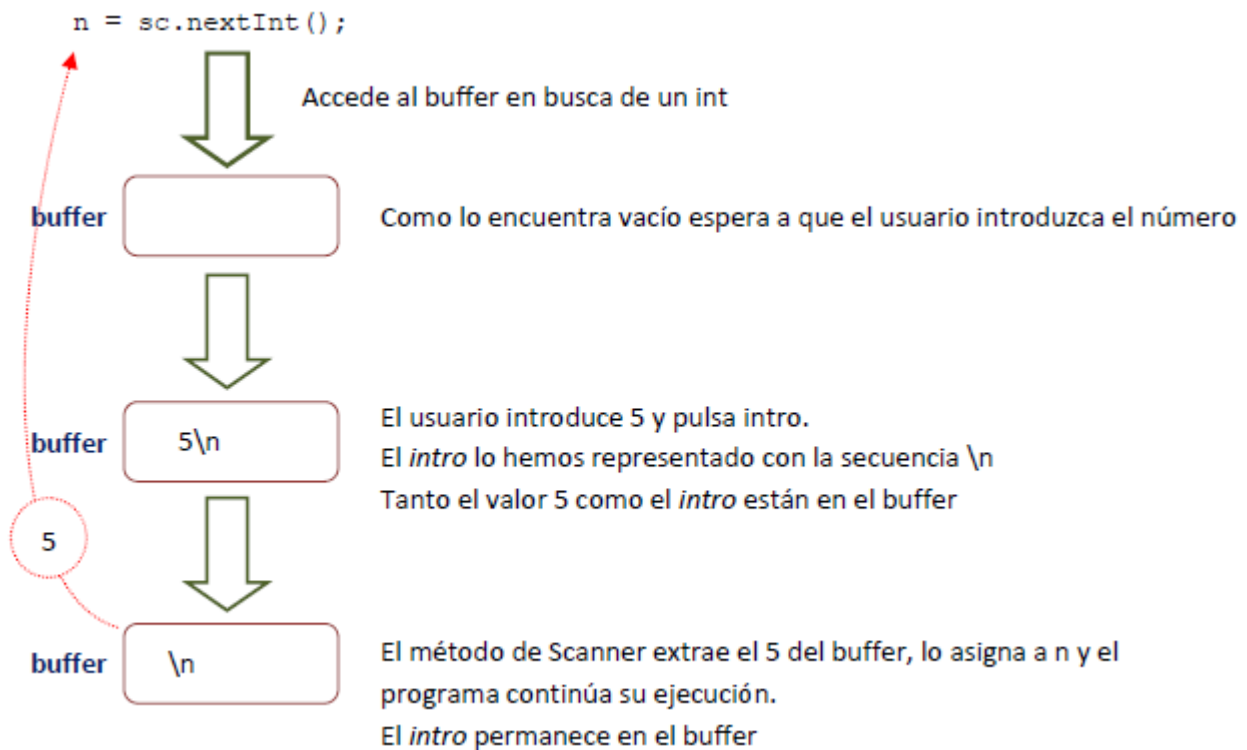
- Cuando en el programa aparece una instrucción para leer un dato por teclado, se accede al buffer de entrada en busca del dato. Si lo encuentra, lo extrae del buffer y lo incorpora al programa.
- Si el dato no ha sido encontrado en el buffer generalmente se deberá a que el buffer está vacío. Esta es 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.

Ejemplo: acciones internas que se realizan cuando se lee un número entero desde teclado.

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

Mediante la instrucción `sc.nextInt()` Scanner accede al buffer de entrada para obtener un dato de tipo `int` y asignarlo a la variable `n`.

Si en el buffer no hay nada (que será lo más habitual) el programa espera a que se introduzca un número. Cuando el usuario introduce el número y pulsa intro, se extrae ese número del buffer y se lo asigna a la variable `n`.



Importante: cuando se introducen datos desde teclado, se pueden introducir varios valores separados por espacios en blanco.

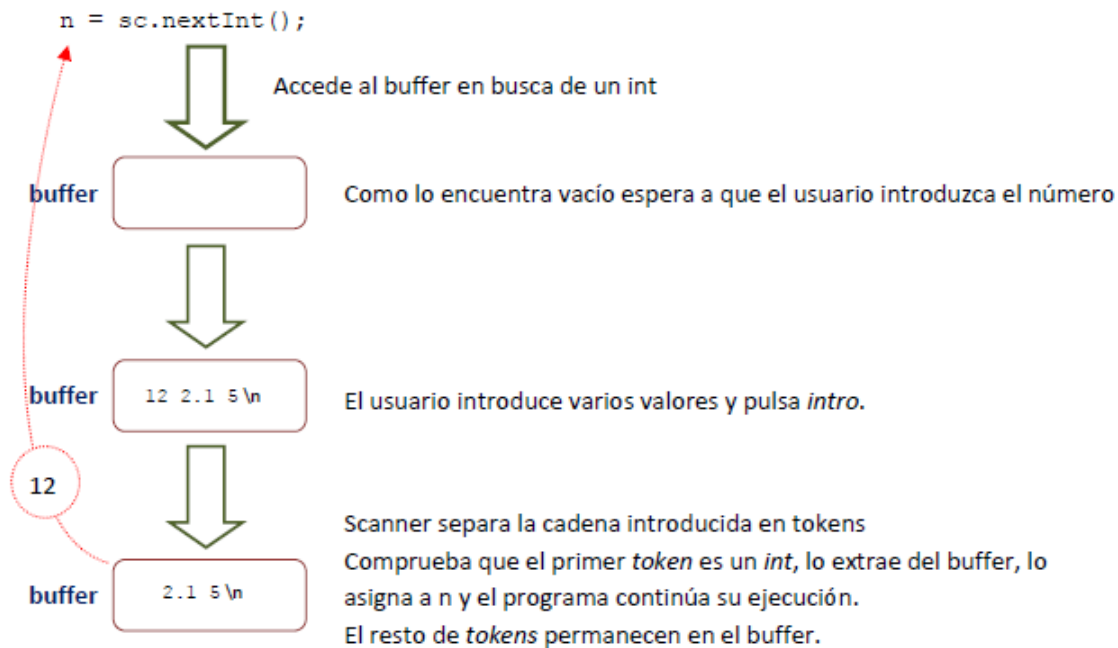
Por ejemplo, para el ejemplo anterior se podría haber introducido:

12 2.1 5

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

Ejemplo: acciones internas que se realizan cuando se quiere leer un número entero desde teclado pero se han introducido varios valores separados por espacios en blanco.

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

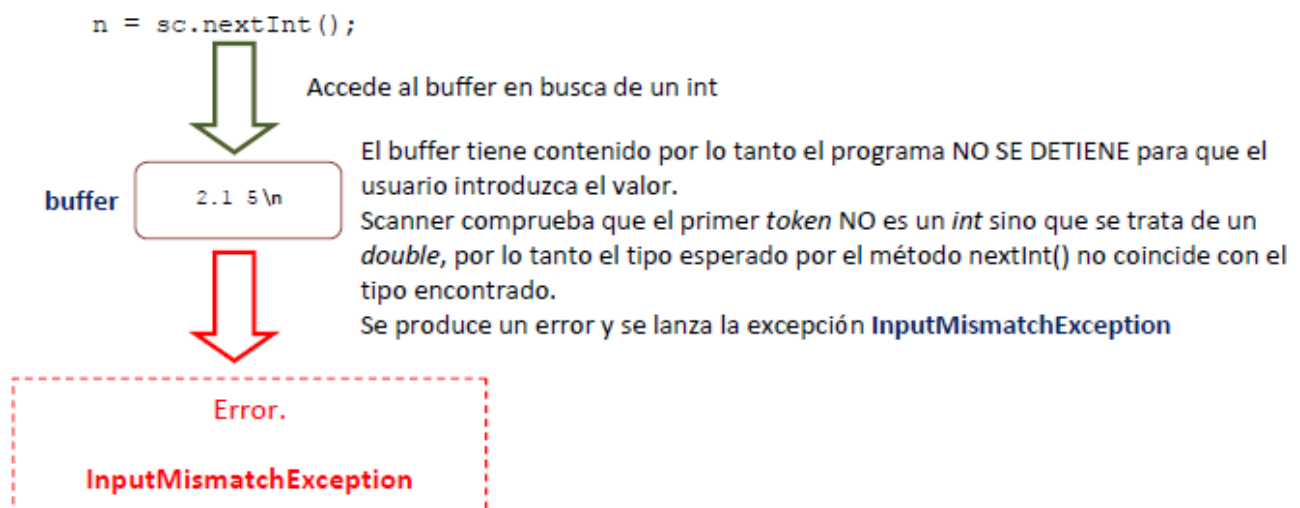


Debemos tener claro el funcionamiento de Scanner en estos casos para evitar errores durante la ejecución del programa.

Si a partir de esta situación del buffer se pide al usuario que introduzca otro número entero:

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

Ahora de forma interna se realizarían las siguientes acciones:



Para evitar estos errores la clase Scanner proporciona métodos para saber si hay tokens en el buffer y para saber el tipo del siguiente token a extraer:

- **hasNext():** Devuelve un boolean. Indica si existe o no un siguiente token para extraer.
- **hasNextXxx():** Devuelve un boolean. Indica si el siguiente token a extraer es del tipo especificado por Xxx, por ejemplo hasNextInt(), hasNextDouble(), etc.

Ejemplo: programa que lee por teclado un número entero positivo. Si el valor introducido no es un número entero positivo se muestra un mensaje y se vuelve a pedir.

```
import java.util.Scanner;
public class Ejemplo2Scanner {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N;
        do {
            System.out.print("Introduce un número entero positivo: ");
            while (!sc.hasNextInt()) {
                System.out.println("Valor no válido");
                sc.next();
                System.out.print("Introduce un número entero positivo: ");
            }
            N = sc.nextInt();
            if(N <= 0){
                System.out.println("El número debe ser positivo");
            }
        } while (N <= 0);
        System.out.println("Número introducido: " + N);
    }
}
```

En el programa se ha utilizado el método **hasNextInt()** para comprobar si el número que se introduce por teclado es de tipo int. El método hasNextInt() accede al buffer para comprobar si el siguiente token a extraer es un int. Si encuentra el buffer vacío espera a que el usuario introduzca un valor y una vez introducido comprueba su tipo. Si no es un int devuelve false. El while se está ejecutando mientras que el dato introducido por teclado no sea de tipo int.

Dentro del while aparece la instrucción:

```
sc.next();
```

Esta instrucción sirve para **extraer del buffer el número no válido introducido**. Es necesario escribirla ahí porque si no lo hacemos provocaremos que ese while se convierta en un bucle infinito. El dato no válido introducido seguirá estando en el buffer con lo que el método hasNextInt() comprobará de nuevo que no es válido y así seguirá indefinidamente.

Otro método que puede resultarnos útil cuando trabajamos con Scanner es el método **useDelimiter(String)** que sirve para establecer un nuevo delimitador de tokens.

Cómo limpiar el buffer de entrada en Java

En el ejemplo anterior hemos visto que en ocasiones es necesario extraer del buffer de entrada los datos no válidos introducidos.

En el ejemplo se ha eliminado el dato con el método next().

Podemos limpiar el buffer mediante dos métodos:

- **next()** extrae del buffer el siguiente token en forma de String. Debemos tener en cuenta que este método solo extrae un token, si hubiese más permanecerían en el buffer.

- **nextLine()** extrae del buffer un String con todo el contenido del buffer hasta encontrar un salto de línea. El salto de línea no se incluye en el String devuelto. El salto de línea se elimina del buffer.

Cuando se introducen datos de distinto tipo desde teclado habrá situaciones en la que será necesario eliminar totalmente el contenido del buffer de entrada. De otro modo la lectura de datos no se realizará de forma correcta, en particular cuando se intenten introducir datos de tipo String.

Ejemplo: Programa que lee por teclado el nombre, edad y dirección de una persona y lo muestra por pantalla.

```
import java.util.Scanner;
public class Ejemplo3Scanner {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nombre, direccion;
        int edad;
        System.out.print("Introduce tu nombre: ");
        nombre = sc.nextLine(); //leer el nombre
        System.out.print("Introduce tu edad: ");
        edad = sc.nextInt(); //leer la edad
        System.out.print("Introduce tu dirección: ");
        direccion = sc.nextLine(); //leer la dirección
        System.out.println("Datos introducidos");
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Dirección: " + direccion);
    }
}
```

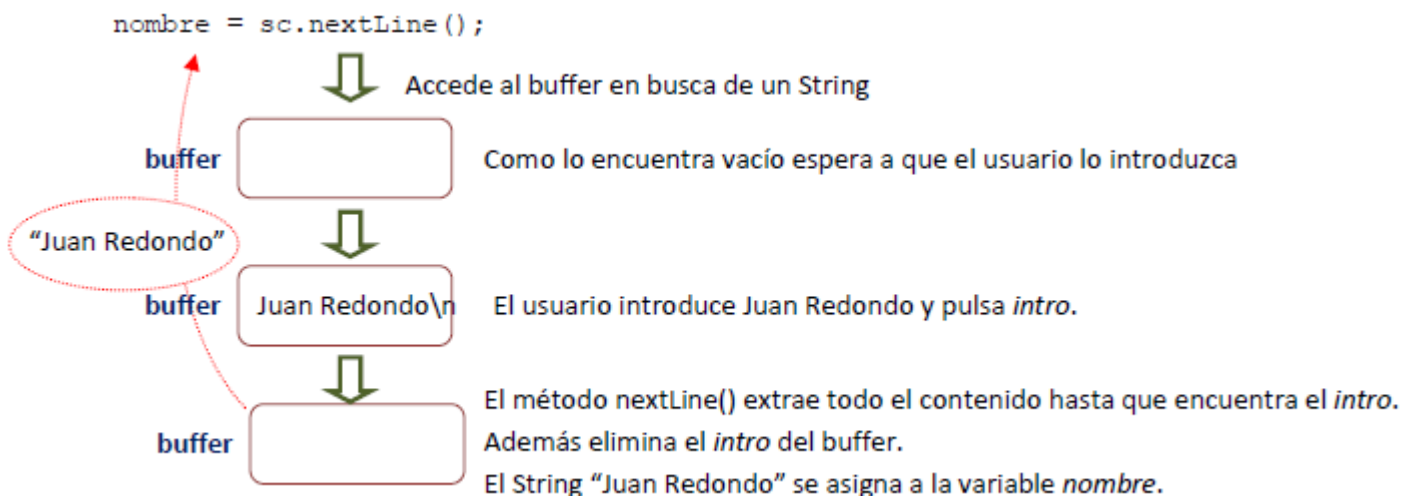
Si lo ejecutamos vemos que la lectura de datos no se realiza de forma correcta:

```
Introduce tu nombre: Juan Redondo
Introduce tu edad: 25
Introduce tu dirección: Datos introducidos
Nombre: Juan Redondo
Edad: 25
Dirección:
```

Se ha leído por teclado el nombre y la edad pero el programa no se ha detenido para que el usuario introduzca la dirección por lo que la variable dirección queda sin valor.

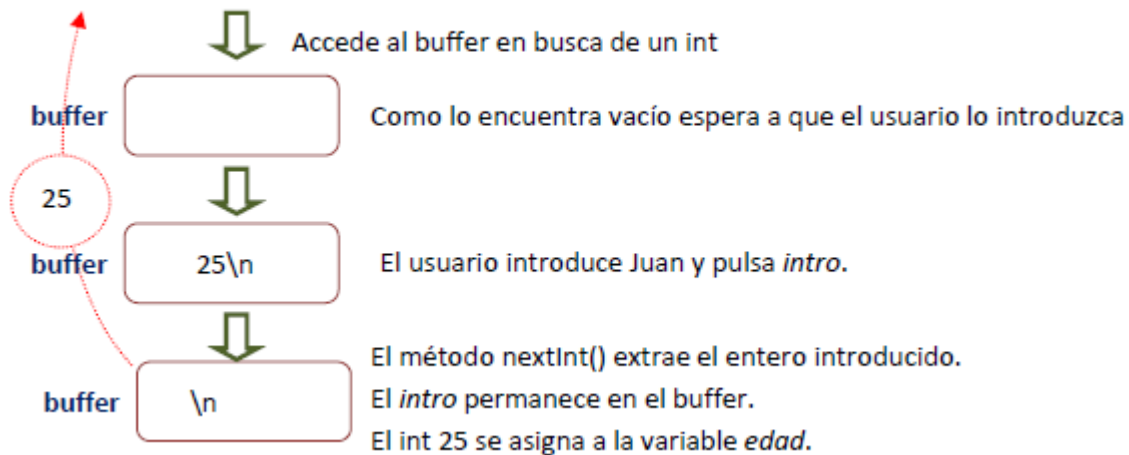
La explicación a lo que ha ocurrido podemos verla de forma gráfica:

Primero se introduce el nombre por teclado:



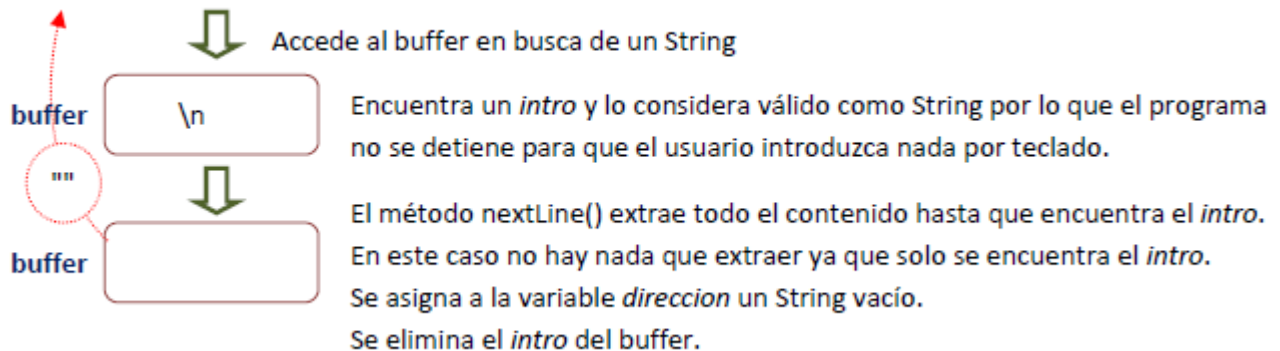
La siguiente operación de lectura por teclado es la edad:

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



La siguiente operación de lectura por teclado es la dirección:

```
direccion = sc.nextLine();
```



Después de leer la edad, el intro permanece en el buffer y si después intentamos leer un String, la lectura por teclado no se realizará.

La solución es sencilla:

Limpiar el buffer después de leer el número entero mediante el método nextLine();

El programa quedaría así:

```
import java.util.Scanner;
public class Ejemplo3Scanner {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nombre, direccion;
        int edad;
        System.out.print("Introduce tu nombre: ");
        nombre = sc.nextLine(); //leer el nombre
        System.out.print("Introduce tu edad: ");
        edad = sc.nextInt(); //leer la edad

        sc.nextLine(); //limpiar el buffer de entrada

        System.out.print("Introduce tu dirección: ");
        direccion = sc.nextLine(); //leer la dirección
        System.out.println("Datos introducidos");
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Dirección: " + direccion);
    }
}
```

```
}  
}
```

Ahora, después de introducir la edad, el método `nextLine()` elimina el intro que ha quedado en el buffer por lo que la lectura de la dirección se realizará de forma correcta.

```
Introduce tu nombre: Juan Redondo  
Introduce tu edad: 25  
Introduce tu dirección: C/Mayor 134 - Madrid.  
Datos introducidos  
Nombre: Juan Redondo  
Edad: 25  
Dirección: C/Mayor 134 - Madrid.
```

En general se debe **limpiar el buffer** de entrada de Scanner cuando tengamos que introducir un dato de tipo String después de haber introducido un dato de tipo numérico

Si quieres ver más ejemplos de entrada de datos en Java con Scanner los tienes en este enlace

[Ejercicios Básicos en Java](#)