

# Módulo 5

# Variables, Operadores e Instrucciones Básicas



# Identificadores

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)
- Guión bajo (\_)

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)
- Guión bajo (\_)

Sin embargo, no pueden:

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)
- Guión bajo (\_)

Sin embargo, no pueden:

- El primer caracter ser un número



# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)
- Guión bajo (\_)

Sin embargo, no pueden:

- El primer caracter ser un número
- Contener puntos (.), asteriscos (\*), u otros caracteres especiales (@!ñ%&).

# Identificadores

Un identificador es el nombre que se utiliza para referirnos una variable o método. Las variables pueden contener:

- Letras (sin caracteres especiales)
- Números (0123456789)
- Guión bajo (\_)

Sin embargo, no pueden:

- El primer caracter ser un número
- Contener puntos (.), asteriscos (\*), u otros caracteres especiales (@!ñ%&).
- Ser una palabra clave o palabra reservada. Ver lista

```
01  contador
02  Acum
03  dolares_01
04  arroba
05  happyFace
06  contador inicial
07  1
08  a
09  _
10  01_dolares
11  char
12  TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

# 01 contador

```
02  Acum
03  dolares_01
04  arroba
05  happyFace
06  contador inicial
07  1
08  a
09  _
10  01_dolares
11  char
12  TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✓ Válido

01 contador

## 02 Acum

03 dolares\_01

04 arroba

05 happyFace

06 contador inicial

07 1

08 a

09 \_

10 01\_dolares

11 char

12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance

✓ Válido, combinar mayúsculas y minúsculas está permitido.

01 contador

02 Acum

**03 dolares\_01**

04 arroba

05 happyFace

06 contador inicial

07 1

08 a

09 \_

10 01\_dolares

11 char

12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance

✓ Válido, combinar caracteres y números está permitido.

01 contador

02 Acum

03 dolares\_01

**04 arroba**

05 happyFace

06 contador inicial

07 1

08 a

09 \_

10 01\_dolares

11 char

12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance

✓ Válido

```
01 contador
02 Acum
03 dolares_01
04 arroba
```

## 05 happyFace

```
06 contador inicial
07 1
08 a
09 _
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✓ Válido



```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
```

## 06 contador inicial

```
07 1
08 a
09 _
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✖ Inválido! No podemos combinar letras y espacios.

```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
06 contador inicial
```

```
07 1
```

```
08 a
09 _
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

❌ Inválido! Los identificadores no pueden comenzar con un número.

```
01  contador
02  Acum
03  dolares_01
04  arroba
05  happyFace
06  contador inicial
07  1
08  a
09  _
10  01_dolares
11  char
12  TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✓ Válido

```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
06 contador inicial
07 1
08 a
```

09

```
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

❌ Inválido! Los identificadores no pueden comenzar con caracteres especiales.

```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
06 contador inicial
07 1
08 a
09 _
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✖ Inválido! Los identificadores no pueden comenzar con un número.

```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
06 contador inicial
07 1
08 a
09 _
10 01_dolares
```

## 11 char

```
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

❌ Inválido! Existen palabras reservadas que NO pueden ser utilizadas como nombres de variables.

```
01 contador
02 Acum
03 dolares_01
04 arroba
05 happyFace
06 contador inicial
07 1
08 a
09 _
10 01_dolares
11 char
12 TodayIsAVeryHappyDayInMyLifeAllIWantToDoIsDance
```

✓ Válido, es posible combinar mayúsculas y minúsculas. Los identificadores pueden ser arbitrariamente largas.

# Variables



# Variables

Una *variable* es la unión de un **tipo de dato** y un **identificador** que nos permiten almacenar información. En Java, las variables son una ubicación de **memoria RAM**.

Cuando la variable almacena cierta información, el dato se codifica en 1's y 0's y se almacena en memoria RAM.



# Variables

Una variable pasa por dos fases:

# Variables

Una variable pasa por dos fases:

1. **Declaración:** Especificamos un tipo de dato [int, char, String]

# Variables

Una variable pasa por dos fases:

1. **Declaración:** Especificamos un tipo de dato [int, char, String]
2. **Asignación:** Utilizamos el símbolo = para especificar un valor.

# Variables

Una variable pasa por dos fases:

1. **Declaración:** Especificamos un tipo de dato [int, char, String]
2. **Asignación:** Utilizamos el símbolo = para especificar un valor.

Una variable SIEMPRE debe declararse antes de poderse usar.

# Tipos de datos primitivos

Type Name	Kind of Value	Memory Used	Range of Values
byte	Integer	1 byte	−128 to 127
short	Integer	2 bytes	−32,768 to 32,767
int	Integer	4 bytes	−2,147,483,648 to 2,147,483,647
long	Integer	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
double	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
char	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
boolean		1 bit	True or false

# Variables primitivas

Las variables primitivas son los tipos de datos más básicos existentes en Java. Al combinar estos tipos de datos podemos representar cualquier estructura, juego, video, u objeto que nos imaginemos.



# Variables primitivas

Las variables primitivas son los tipos de datos más básicos existentes en Java. Al combinar estos tipos de datos podemos representar cualquier estructura, juego, video, u objeto que nos imaginemos.



```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

Declaración de la variable "money\_in\_bank" de tipo "double".

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

Almacenamos el valor "100.50" en la variable "money\_in\_bank".

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

Declaración de la variable "myName" de tipo "String".

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

Guardamos el texto "Arthur" en la variable "myName".

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

Reemplazamos el contenido de "myName" con "Arthur Fleck".

```
01 //Declaration
02 double money_in_bank;
03 //Assignment
04 money_in_bank = 100.50;
05
06 //Declaracion
07 String myName;
08 // First assignment
09 myName = "Arthur";
10 //Overwriting with second assignment
11 myName = "Arthur Fleck";
12
13 //Declaration and Assignment
14 int daysOfWeek = 7;
```

También es posible hacer una declaración y asignación en una misma instrucción.



# Operadores

# Operadores Aritméticos

Podemos formar expresiones aritméticas mediante los operadores:

- + Suma
- - Resta
- \* Multiplicación
- / División
- % Módulo o residuo

```
01 //Declaration
02 double pay;
03 int hoursWorked = 40;
04 double payRate = 8.25;
05
06 //Assignment
07 pay = hoursWorked * payRate;
08 System.out.println(pay);
```

```
01 //Declaration
02 double pay;
03 int hoursWorked = 40;
04 double payRate = 8.25;
05
06 //Assignment
07 pay = hoursWorked * payRate;
08 System.out.println(pay);
```

Declaraciones iniciales

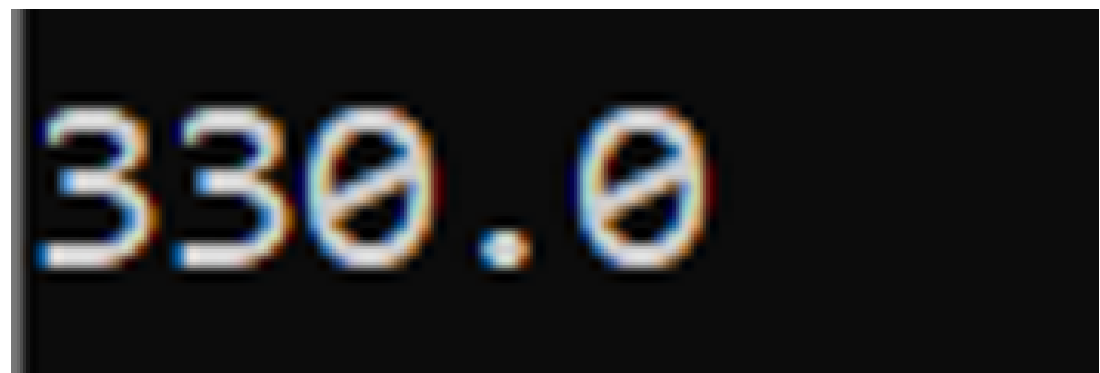
```
01 //Declaration
02 double pay;
03 int hoursWorked = 40;
04 double payRate = 8.25;
05
06 //Assignment
07 pay = hoursWorked * payRate;
08 System.out.println(pay);
```

Multiplicamos "hoursWorked" y "payRate", almacenamos el resultado en "pay".

```
01 //Declaration
02 double pay;
03 int hoursWorked = 40;
04 double payRate = 8.25;
05
06 //Assignment
07 pay = hoursWorked * payRate;
08 System.out.println(pay);
```

```
01 //Declaration
02 double pay;
03 int hoursWorked = 40;
04 double payRate = 8.25;
05
06 //Assignment
07 pay = hoursWorked * payRate;
08 System.out.println(pay);
```

# Output!



330.0

# División

## IMPORTANTE!!!

Cuando ambos operandos son de tipo `int`, el símbolo (`/`) indica una división entera. El resultado se trunca:

$$9 / 2 = 4$$

$$100 / 99 = 1$$



# División

## IMPORTANTE!!!

Cuando ambos operandos son de tipo `int`, el símbolo (`/`) indica una división entera. El resultado se trunca:

$$9 / 2 = 4$$

$$100 / 99 = 1$$

Cuando por lo menos alguno de los operandos de la división es de tipo `double` o `float`, se hace división decimal. El resultado contará con una parte decimal:

$$9 / 2.0 = 4.5$$

$$100.0 / 99 = 1.0101010101010102$$

# Módulo

El operador de **módulo** (%) calcula el residuo después de una división entera.

$$9 \% 2 = 1$$

$$17 \% 3 = 2$$

Este operador tiene muchos usos, por ejemplo **verificar si un número es par o impar**.

# Operadores Incrementales y Decrementales

Hay instrucciones especiales en Java que nos permiten incrementar el valor de una variable en 1. Estos se identifican mediante `++` y `--`. Por ejemplo, asumiento que `count` es una variable numérica:

```
count++
```

```
++count
```

```
count--
```

```
--count
```

# Operadores Incrementales y Decrementales

Hay instrucciones especiales en Java que nos permiten incrementar el valor de una variable en 1. Estos se identifican mediante **++** y **--**. Por ejemplo, asumiento que count es una variable numérica:

```
count++  
++count  
count--  
--count
```

Cuando el operador está antes de la variable, se actualiza el contenido de la variable antes de evaluar la expresión.

Cuando el operador está después, se evalúa la expresión y después se actualiza la variable.

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

"m" tiene un valor de 4.

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

Al evaluar la asignación, primero se actualiza el valor de "m", y después se evalúa la multiplicación.  $\text{result} = 3 \times 5$

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```



```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

n tiene un valor inicial de 4

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

```
01  int m = 4;
02  int result = 3 * (++m);
03
04  //After executing:
05  // - result has a value of 15
06  // - m has a value of 5
07
08
09  int n = 4;
10  int result = 3 * (n++);
11  //After executing:
12  // - result has a value of 12
13  // - n has a value of 5
```

Al evaluar la asignación, primero se evalúa la multiplicación 3 x 4, y luego se incrementa el valor de "n"

# Precedencia de Operaciones

## *Highest Precedence*

First: the unary operators  $+$ ,  $-$ ,  $!$ ,  $++$ , and  $--$

Second: the binary arithmetic operators  $*$ ,  $/$ , and  $\%$

Third: the binary arithmetic operators  $+$  and  $-$

## *Lowest Precedence*

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**a = 7**

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**b = 4**

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**c = 7 + 5**



```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**d = 7 + 4 + 12**

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**e = 7 + 4 + -11 + 23**

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

**sum = 7 + 4 + 11 + 22 + 23**

```
01 class Operators1 {  
02     public static void main(String[] args) {  
03         int a = 7;  
04         int b = 4;  
05         int c = a++ + ++b;  
06         int d = --a + --b + c--;  
07         int e = a + +b + -c + d--;  
08         int sum = a + b + c + d + e;  
09         System.out.println("Suma: " + sum);  
10     }  
11 }
```

"Suma: 67"

```
01 class Operators1 {
02     public static void main(String[] args) {
03         int a = 7;
04         int b = 4;
05         int c = a++ + ++b;
06         int d = --a + --b + c--;
07         int e = a + +b + -c + d--;
08         int sum = a + b + c + d + e;
09         System.out.println("Suma: " + sum);
10     }
11 }
```

# Bloques

# Bloques

Un bloque es una sección de código encapsulada por llaves ( `{ }` ).

Un bloque no indica alguna instrucción, sino sirve para indicar el inicio o fin de algun grupo de instrucciones.

Cada llave de apertura debe ser posteriormente cerrada por una llave opuesta:

```
01 public class Test {  
02     public static void main(String[] args) {  
03         System.out.println("Hello, World!");  
04     }  
05 }
```

Las llaves se pueden acomodar en una misma línea.  
(Esto es lo más común!)



```
01 public class Test
02 {
03     public static void main(String[] args)
04     {
05         System.out.println("Hello, World!");
06     }
07 }
```

¡O en líneas distintas!

# Clase Scanner

Lectura del teclado

# Clase Scanner

La clase Scanner es una librería de instrucciones que permiten al programador **leer información del teclado a través de la consola**. Cuando queremos incorporar librerías externas a alguna clase, agregamos referencia al programa utilizando la instrucción import.

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

De esta forma, toda la funcionalidad de la clase Scanner se incluye al programa en el que estamos trabajando.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!")
12         keyboard.close();
13     }
14
15 }
```

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!"
12         keyboard.close();
13     }
14
15 }
```



Agregamos el include al comenzar el archivo. Aquí podemos incluir todas las librerías que vayamos a importar.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!");
12         keyboard.close();
13     }
14
15 }
```

Declaramos la clase y el método main

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!")
12         keyboard.close();
13     }
14
15 }
```



Declaramos la variable "keyboard" de la clase Scanner.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!");
12         keyboard.close();
13     }
14
15 }
```



Inicializamos la variable "keyboard" para hacer referencia al teclado.



```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!");
12         keyboard.close();
13     }
14
15 }
```



Imprimimos un mensaje en consola.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!");
12         keyboard.close();
13     }
14
15 }
```



El programa se detendrá hasta que el usuario escriba algo en consola, y presione la tecla Enter. El mensaje escrito en consola se guardará en la variable "n"

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!")
12         keyboard.close();
13     }
14
15 }
```



Concatenamos los mensajes, y les damos salida en la consola.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!");
12         keyboard.close();
13     }
14
15 }
```



Cerramos la conexión al teclado. A partir de este momento, la variable "keyboard" no podrá ser utilizada para leer información del teclado.

```
01 import java.util.Scanner;
02
03 public class ScannerTest{
04     public static void main(String[] args){
05
06         //Configurar la lectura del teclado
07         Scanner keyboard;
08         keyboard = new Scanner(System.in);
09         System.out.print("Escribe tu nombre: ");
10         String n = s1.nextLine();
11         System.out.println("¡Wow! Tu nombre " + n + " es muy bonito!")
12         keyboard.close();
13     }
14
15 }
```

```
01 Scanner keyboard = new Scanner(System.in);
02
03 String s1 = keyboard.nextLine();
04
05 float f1 = keyboard.nextFloat();
06 double d1 = keyboard.nextDouble();
07
08 byte b1 = keyboard.nextByte();
09 short sh1 = keyboard.nextShort();
10 int i = keyboard.nextInt();
11 long l1 = keyboard.nextLong();
12
13 boolean b1 = keyboard.nextBoolean();
```

Para leer otros tipos de datos primitivos, podemos utilizar la nomenclatura:

`keyboard.next + data type`

# Clase String

## Manejo de cadenas