

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:

- 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:

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	−9,223,372,036,8547,75,808 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.



```
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

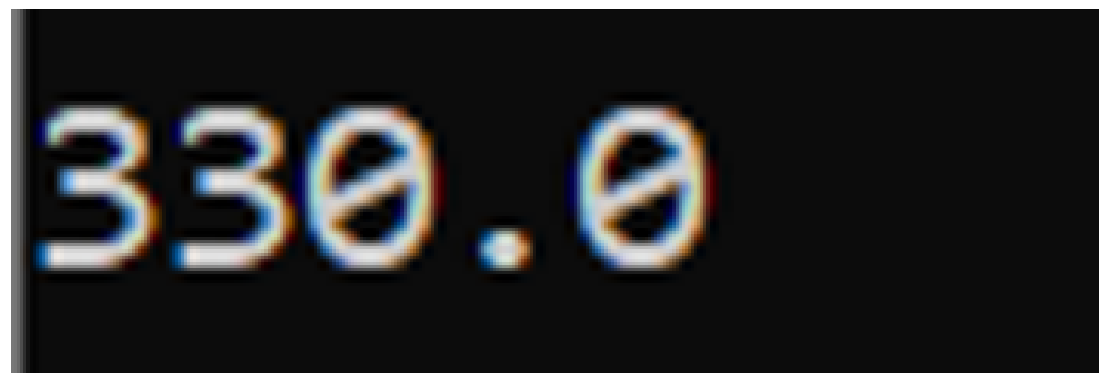
Operadores Aritméticos

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);
```

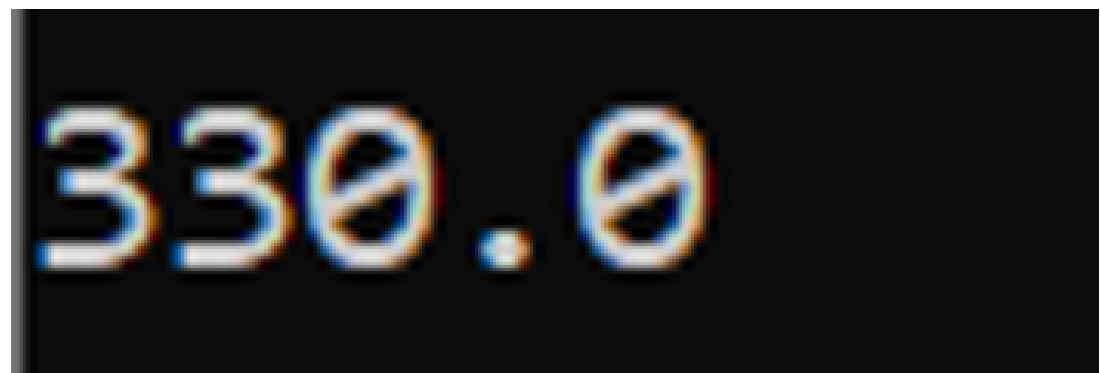
Output!



330.0


```
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!



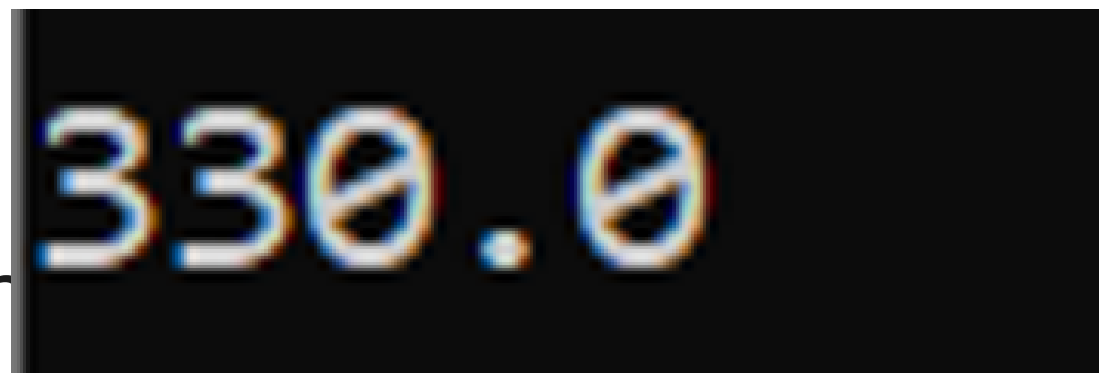
330.0

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);
```

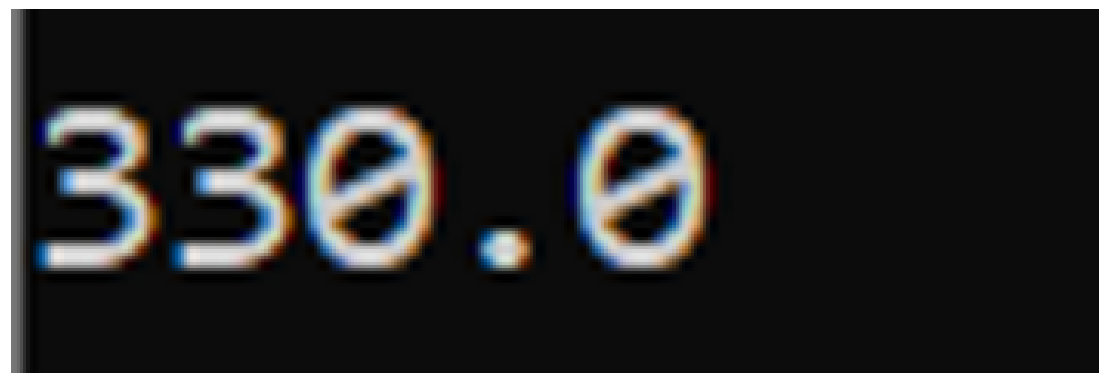
Output!

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

A screenshot of a terminal window with a black background. The text "330.0" is displayed in a light blue, monospaced font, representing the output of the Java program.

```
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:

$$9 / 2 = 4$$

$$100 / 99 = 1$$

Cuando por lo menos algun operando 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
```

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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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 = keyboard.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


Strings

Un `String` es una cadena o conjunto de caracteres. No es una variable primitiva, porque se puede subdividir en caracteres individuales. La Clase `String` cuenta con métodos para facilitar el manejo y procesamiento de cadenas.



String in Java

Index	0	1	2	3	4	5	6	7	8	9
Name	D	A	T	A	F	L	A	I	R	\0
Address	101	102	103	104	105	106	107	108	109	110



Un `String` se puede declarar de tres formas distintas:

```
01 String g1;  
02 g1 = "Hello!";  
03  
04 String g2 = "Hello!";  
05  
06 String g3 = new String("Hello!");
```

Las variables de tipo `String` pueden almacenar cualquier caracter de Unicode utilizando **representación UTF-16 [16 bits]**.

Concatenar Strings

Dos strings pueden concatenarse utilizando el operador `+`.

Podemos concatenar valores y variables de diferentes tipos, como `Strings`, `int`, `double`, `char`, `boolean`, etc.

```
01 String greeting;  
02 greeting = "Hello ";  
03  
04 String sentence;  
05 sentence = greeting + "officer";  
06 System.out.println(sentence); //Prints "Hello officer"
```

Inicializar un String

Hay veces que es necesario inicializar un String con un valor vacío. Esto se puede hacer de la siguiente forma:

```
01 String s1 = "";
```

String length()

El método `length()` nos permite calcular el tamaño de un String. Este método devolverá un entero con la cantidad de caracteres que un String contenga.

```
01 String s1;  
02 int n;  
03  
04 s1 = "12345";  
05 n = s1.length();  
06 System.out.println(n); //Prints 5  
07  
08 s1 = "aabc12345";  
09 n = s1.length();  
10 System.out.println(n); //Prints 9  
11  
12 s1 = "";  
13 n = s1.length();  
14 System.out.println(n); //Prints 0
```

String toLowerCase() / toUpperCase()

Los métodos `toLowerCase()` y `toUpperCase()` nos permiten procesar un texto para convertir una cadena a minúscula o mayúscula respectivamente. Estos métodos generan cadenas nuevas, por lo que debemos asignar el valor resultante a una variable de tipo `String`.

```
01 String s1 = "abCD";  
02 String s2 = "abCD";  
03  
04 String lowerCase = s1.toLowerCase();  
05 String upperCase = s2.toUpperCase();  
06  
07 System.out.println(lowerCase); //Prints "abcd"  
08 System.out.println(upperCase); //Prints "ABCD"
```

String replace()

El método `replace(char oldChar, char newChar)` sirve para reemplazar un caracter dentro del texto especificado por otro.

```
01 String s1 = "Bienvenido a la ciudad!";
02 String s2 = s1.replace('e','x');
03 System.out.println(s2); //Prints "Bixnvxnido a la ciudad!"
04
05 s2 = s2.replace('a','x');
06 System.out.println(s2); //Prints "Bixnvxnido x lx ciudxd!"
```


String substring()

El método `substring(int beginIndex, int endIndex)` regresa un recorte del String original. El substring comienza en el índice especificado como `beginIndex` y se extiende hasta `endIndex - 1`. El tamaño del String resultante será: `endIndex - beginIndex`.

NOTA: El conteo de posiciones comienza desde 0.

Ejemplo:

```
01 String s1 = "Monterrey, Nuevo León";
02 String ciudad = s1.substring(0,9);
03 String estado = s1.substring(11,21);
04
05 System.out.println(ciudad); //Prints "Monterrey"
06 System.out.println(estado); //Prints "Nuevo León"
07
08 String s2 = "smiles".substring(1, 5);
09 System.out.println(s2); //prints "mile"
```

String trim()

El método `trim()` crea un nuevo string eliminado los espacios vacíos al inicio y al final.

```
01 String s1 = "    :)    ";  
02 String s2 = s1.trim();  
03 System.out.println(s2); //prints ":)"
```

String charAt()

El método `charAt(int index)` permite recuperar el caracter que se encuentra en la posición especificada, siendo 0 el índice del primer caracter, y `length()-1` índice del último.

```
01 String s1 = "The Jungle Book";
02 char c1 = s1.charAt(1);
03 System.out.println(c1); //prints 'h'
04
05 char c2 = s1.charAt(4);
06 System.out.println(c2); //prints 'J'
07
08 char c3 = s1.charAt(s1.length()-1);
09 System.out.println(c3); //prints 'k'
```

Comentarios

Comentarios

Los comentarios son mensajes en el código ignorados por el compilador que le permiten un programador **explicar el funcionamiento de un programa** a otros programadores (o él mismo en el futuro!)

90% of all code comments:



// Comentarios individuales

Podemos agregar un comentario en el código utilizando los símbolos `//`.

Podemos agregarlo al comenzar o finalizar una oración. Todo lo que le sigue posterior será considerado un comentario.

```
01 //This is a comment!!  
02 double radius; //in centimeters
```

`/* ... */` Comentarios multilinea

Cuando queremos agregar comentarios de varias líneas, podemos comenzar los comentarios con: `/*` y terminar `*/`.

```
01  /*  
02  This program will only work when processing  
03  lengths in centimeters. If you need to use  
04  other length units you must modify it.  
05  */
```


FIXME y TODO

Hay ciertos identificadores especiales en los comentarios que un IDE (como Visual Studio Code, Eclipse, Netbeans, etc) reconoce como notación especial. Algunos ejemplos son: **FIXME** y **TODO**. Algunos ambientes de programación resaltan estos mensajes, pues indican que un programa aún no está terminado o tiene errores.

Ejemplo #1

```
01  /*  
02  FIXME: Logic needs to be updated.  
03  */
```

Ejemplo #2

```
01  /*  
02  TODO: Add functionality to include calculation  
03  */
```

Condiciones IF

IF

Existen instrucciones de control de flujo que nos permiten ejecutar selectivamente ciertas instrucciones. La primera instrucción instrucción es el IF.

```
01 int result;  
02 if (denominator != 0) {  
03     result = numerator / denominator;  
04 }
```

Cuando la condición dentro de los paréntesis `denominator != 0` se cumple, entonces las instrucciones dentro del bloque `{ }` se ejecutarán.

IF-ELSE

Hay un segundo bloque, el ELSE que sólo se ejecuta cuando la condición NO se cumple.

```
01  int result;
02
03  if (denominator == 0) {
04      System.out.println("Error! You cannot divide by zero.");
05  } else {
06      result = numerator / denominator;
07  }
```

IF-ELSE

Hay un segundo bloque, el ELSE que sólo se ejecuta cuando la condición NO se cumple.

```
01 int result;  
02  
03 if (denominator == 0) {  
04     System.out.println("Error! You cannot divide by zero.");  
05 } else {  
06     result = numerator / denominator;  
07 }
```

Cuando denominador es diferente a 0, se ejecuta esta instrucción.

Expresiones booleanas

Una expresión booleana es una operación cuyo valor puede ser **TRUE** o **FALSE**. Hacemos uso de los operadores booleanos para realizar las comparaciones.

```
01 time < limit
02 count > 0
03 letter == 'c'
```

Operadores booleanos

Math Notation	Name	Java Notation	Java Examples
=	Equal to	==	<code>balance == 0</code> <code>answer == 'y'</code>
≠	Not equal to	!=	<code>income != tax</code> <code>answer != 'y'</code>
>	Greater than	>	<code>expenses > income</code>
≥	Greater than or equal to	>=	<code>points >= 60</code>
<	Less than	<	<code>pressure < max</code>
≤	Less than or equal to	<=	<code>expenses <= income</code>

Expresiones booleanas compuestas

Es posible combinar múltiples expresiones booleanas mediante los operadores AND (&&) y OR (||). La sintaxis es la siguiente:

(Expresion_1) && (Expresion_2)

Los paréntesis son opcionales, pero facilitan la lectura de las expresiones.

```
01 //store is open between 8am and 10pm
02 ((hour >= 8) && (hour <= 22))
03
04 //store is open between 8am and 1pm, then from 3pm to 10pm
05 ((hour >= 8) && (hour <= 13) || (hour >= 15 && hour <= 22))
```

Name	Java Notation	Java Examples
Logical <i>and</i>	&&	(sum > min) && (sum < max)
Logical <i>or</i>		(answer == 'y') (answer == 'Y')
Logical <i>not</i>	!	!(number < 0)

Value of <i>A</i>	Value of <i>B</i>	Value of <i>A</i> && <i>B</i>	Value of <i>A</i> <i>B</i>	Value of ! (<i>A</i>)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Comparación de variables primitivas

Para comparar la **igualdad** de dos valores primitivos, podemos utilizar el operador de igualdad (**==**).

Por ejemplo:

```
01  if (a == 3) {  
02      System.out.println("a equals 3");  
03  }
```

OJO! Recordemos que los valores de punto flotante pueden perder precisión, por lo que la mejor forma de comparar dos valores double es mediante los operadores menor o igual que (\leq) y mayor o igual que (\geq).

```
01 double total = 1.0 / 10.0;  
02 total = total + 1.0 / 10.0;  
03 total = total + 1.0 / 10.0;  
04 total = total + 1.0 / 10.0;  
05 total = total + 1.0 / 10.0;  
06 total = total + 1.0 / 10.0;  
07 total = total + 1.0 / 10.0;  
08 total = total + 1.0 / 10.0;  
09 total = total + 1.0 / 10.0;  
10 total = total + 1.0 / 10.0;  
11 System.out.println(total); //prints 0.9999999999999999
```

```
01 double total = 1.0 / 10.0;  
02 total = total + 1.0 / 10.0;  
03 total = total + 1.0 / 10.0;  
04 total = total + 1.0 / 10.0;  
05 total = total + 1.0 / 10.0;  
06 total = total + 1.0 / 10.0;  
07 total = total + 1.0 / 10.0;  
08 total = total + 1.0 / 10.0;  
09 total = total + 1.0 / 10.0;  
10 total = total + 1.0 / 10.0;  
11 System.out.println(total); //prints 0.9999999999999999
```

Instanciamos una variable double de punto flotante.


```
01 double total = 1.0 / 10.0;
02 total = total + 1.0 / 10.0;
03 total = total + 1.0 / 10.0;
04 total = total + 1.0 / 10.0;
05 total = total + 1.0 / 10.0;
06 total = total + 1.0 / 10.0;
07 total = total + 1.0 / 10.0;
08 total = total + 1.0 / 10.0;
09 total = total + 1.0 / 10.0;
10 total = total + 1.0 / 10.0;
11 System.out.println(total); //prints 0.9999999999999999
```

```
01 double total = 1.0 / 10.0;  
02 total = total + 1.0 / 10.0;  
03 total = total + 1.0 / 10.0;  
04 total = total + 1.0 / 10.0;  
05 total = total + 1.0 / 10.0;  
06 total = total + 1.0 / 10.0;  
07 total = total + 1.0 / 10.0;  
08 total = total + 1.0 / 10.0;  
09 total = total + 1.0 / 10.0;  
10 total = total + 1.0 / 10.0;  
11 System.out.println(total); //prints 0.9999999999999999
```

Se pierde precisión por el contenido de las variables!

```
01 double a = 1/3.0;
02 double b = 0.3333;
03 double tolerance = 0.0001;
04
05 //Check if numbers are equal considering a tolerance
06 if (Math.abs(a - b) <= tolerance){
07     System.out.println("They are equal!");
08 } else {
09     System.out.println("They are different!");
10 }
```

```
01 double a = 1/3.0;
02 double b = 0.3333;
03 double tolerance = 0.0001;
04
05 //Check if numbers are equal considering a tolerance
06 if (Math.abs(a - b) <= tolerance){
07     System.out.println("They are equal!");
08 } else {
09     System.out.println("They are different!");
10 }
```

Declaramos variables

```
01 double a = 1/3.0;
02 double b = 0.3333;
03 double tolerance = 0.0001;
04
05 //Check if numbers are equal considering a tolerance
06 if (Math.abs(a - b) <= tolerance){
07     System.out.println("They are equal!");
08 } else {
09     System.out.println("They are different!");
10 }
```

Calculamos la diferencia con un valor absoluto

```
01 double a = 1/3.0;
02 double b = 0.3333;
03 double tolerance = 0.0001;
04
05 //Check if numbers are equal considering a tolerance
06 if (Math.abs(a - b) <= tolerance){
07     System.out.println("They are equal!");
08 } else {
09     System.out.println("They are different!");
10 }
```

Descartaremos diferencias pequeñas [con una tolerancia de 0.0001] por diferencias de precisión en variables de punto flotante.

Comparación de Strings

Para comparar dos variables de tipo String, NO podemos utilizar el operador (`==`), pues **no son variables primitivas**.

Para esto, debemos utilizar el método `equals()`.

```
01 String s1 = "hola";
02 String s2 = "adios";
03
04 if (s1.equals(s2)){
05     System.out.println("Iguales!");
06 } else {
07     System.out.println("Diferentes!");
08 }
```

Comparación de Strings

También podemos utilizar el método `equalsIgnoreCase()` si queremos verificar comparar sin considerar mayúsculas o minúsculas.

```
01 String s1 = "hola";
02 String s2 = "HOLA";
03
04 if (s1.equalsIgnoreCase(s2)) {
05     System.out.println("Iguales!");
06 } else {
07     System.out.println("Diferentes!");
08 }
```


Ejemplo

Monkey Trouble!

Tenemos dos changos:

- Si los dos changos están sonriendo, estamos en problemas
- Si los dos changos están serios, estamos en problemas.

Diseña un programa de Java que permita modelar este ejercicio.



```
01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }
```

```
01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }
```

Imports, definición de la clase y main

```

01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }

```

Preguntar y capturar el estado de los dos monos. Al ser variables boolean, hay que capturar TRUE / FALSE

```

01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }

```

Si ambos monos, o ninguno, están sonriendo, entonces imprimimos en pantalla un mensaje de alerta.

```
01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }
```

De lo contrario, imprimimos que todo está OK

```
01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }
```

Se cierra el objeto teclado.

```
01 import java.util.Scanner;
02
03 public class MonkeyTrouble {
04     public static void main(String[] args) {
05         Scanner teclado = new Scanner(System.in);
06
07         System.out.print("Is the first monkey smiling? (true/false): ");
08         boolean monkey1 = teclado.nextBoolean();
09
10         System.out.print("Is the second monkey smiling? (true/false): ");
11         boolean monkey2 = teclado.nextBoolean();
12
13         if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
14             System.out.println("Look out! The monkeys are planning something!");
15         }
16         else {
17             System.out.println("Don't worry, everything is OK");
18         }
19         teclado.close();
20     }
21 }
```


Como la variable monkey es una variable de tipo boolean que ya almacena un valor TRUE/FALSE, podemos ahorrarnos la comparación.

```
01 if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {  
02     System.out.println("Look out! The monkeys are planning something!");  
03 }  
04 else {  
05     System.out.println("Don't worry, everything is OK");  
06 }
```

Y utilizar la siguiente expresión:

```
01 if ((monkey1 && monkey2) || (!monkey1 && !monkey2)) {  
02     System.out.println("Look out! The monkeys are planning something!");  
03 }  
04 else {  
05     System.out.println("Don't worry, everything is OK");  
06 }
```

Ambas expresiones son equivalentes.

Switch

Switch

Un switch es una instrucción condicional que nos permite evaluar una expresión integral. Es decir, podemos utilizar una variable numérica o de texto para generar los caminos.

Un switch está compuesto de 3 elementos:

- Switch
- Case
- Default

Sintaxis

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': //green light
12         System.out.println("Go!");
13         break;
14 }
```

Sintaxis

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': //green light
12         System.out.println("Go!");
13         break;
14 }
```

Leemos un caracter del teclado.

Sintaxis

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': //green light
12         System.out.println("Go!");
13         break;
14 }
```

Evaluamos la variable traffic_light

Sintaxis

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': //green light
12         System.out.println("Go!");
13         break;
14 }
```

Dependiendo del contenido de la variable `traffic_light`, se ejecutarían distintas secciones.

Sintaxis

Podemos incluir un caso que atrape el resto de las condiciones utilizando el keyword `default`.

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R':
06         System.out.println("Stop!");
07         break;
08     case 'Y':
09         System.out.println("Slow down!");
10         break;
11     case 'G':
12         System.out.println("Go!");
13         break;
14     default:
15         System.out.println("Wrong color!");
16         break;
17 }
```



```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

traffic_light tiene un valor 'G'

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

Se evalúa el contenido de traffic_light en el switch

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

traffic_light == 'R'? false, por lo que continuamos a la siguiente expresión

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

traffic_light == 'Y'? false, por lo que continuamos a la siguiente expresión

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

traffic_light == 'G'? true!, por lo que entramos a la sección

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

Se imprime en pantalla "Go"

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

El break indica que debemos salir del bloque {}


```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

Salimos del bloque

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06         break;
07     case 'Y': //yellow light
08         System.out.println("Slow down!");
09         break;
10     case 'G': //green light
11         System.out.println("Go!");
12         break;
13     default:
14         System.out.println("Wrong color!");
15         break;
16 }
```

Es posible hacer que dos valores sigan el mismo camino:

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': case 'r': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': case 'y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': case 'g': //green light
12         System.out.println("Go!");
13         break;
14     default:
15         System.out.println("Wrong color!");
16         break;
17 }
```

Es posible hacer que dos valores sigan el mismo camino:

```
01 String input = keyboard.nextLine();
02 char traffic_light = input.charAt(0);
03
04 switch(traffic_light) {
05     case 'R': case 'r': //red light
06         System.out.println("Stop!");
07         break;
08     case 'Y': case 'y': //yellow light
09         System.out.println("Slow down!");
10         break;
11     case 'G': case 'g': //green light
12         System.out.println("Go!");
13         break;
14     default:
15         System.out.println("Wrong color!");
16         break;
17 }
```

Errores comunes

1. Olvidar el break: Cuando omitimos los breaks, todas las siguientes opciones se ejecutan.
2. Repetir opciones: dos casos distintos NO PUEDEN TENER un mismo valor.

```
01 char traffic_light = 'G';
02
03 switch(traffic_light) {
04     case 'R': //red light
05         System.out.println("Stop!");
06     case 'Y': //yellow light
07         System.out.println("Slow down!");
08     case 'G': //green light
09         System.out.println("Go!");
10     default:
11         System.out.println("Wrong color!");
12 }
```

❗ Esto imprimiría:

```
> Stop!
> Slow down!
> Go!
> Wrong color!
```