

Programación I

Estructuras de control de Flujo: Iteradores

Bucle while

El bucle while en Java se utiliza para repetir un bloque de código mientras una condición especificada sea verdadera.

El bloque de código se ejecutará repetidamente mientras la condición se evalúe como verdadera. Si la condición es falsa al principio, el bloque de código no se ejecutará en absoluto.

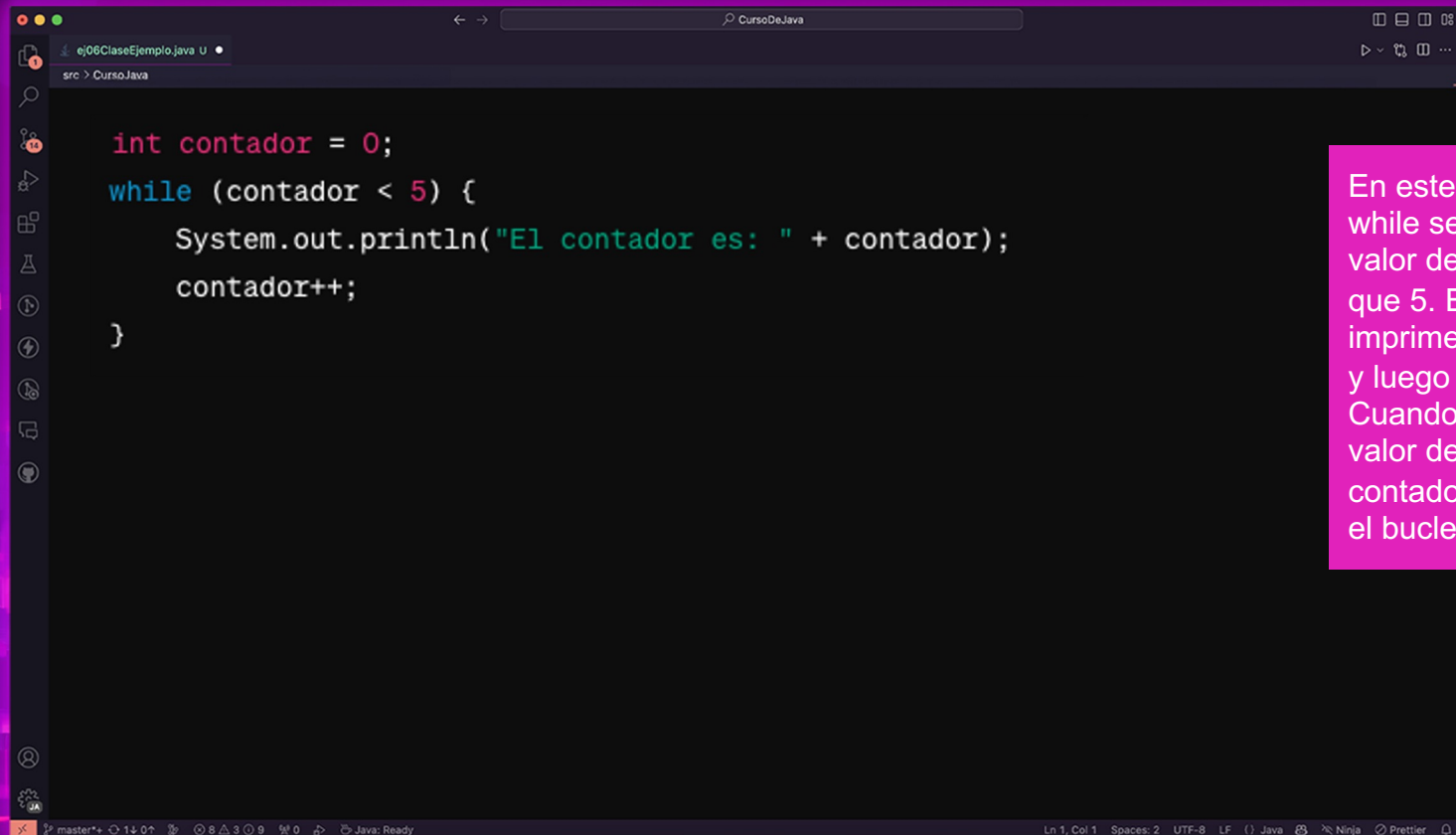
Bucle while

- La condición es una expresión booleana que se evalúa antes de cada iteración del bucle.
- Si la condición es verdadera, se ejecuta el bloque de código.
- Una vez que el bloque de código se ejecuta, la condición se evalúa nuevamente.
- El bucle continuará ejecutándose mientras la condición sea verdadera.
- Si la condición es falsa desde el principio, el bloque de código no se ejecutará.

Bucle while

```
while (condición) {  
    /** Ejecutar mientras  
        la condición sea  
        verdadera */  
}
```

Sintaxis del operador ternario



```
ej06ClaseEjemplo.java U
src > CursoJava

int contador = 0;
while (contador < 5) {
    System.out.println("El contador es: " + contador);
    contador++;
}
```

The screenshot shows a code editor window titled 'ej06ClaseEjemplo.java U' with a file explorer on the left showing 'src > CursoJava'. The code is a Java program using a while loop to print the value of a counter from 0 to 4. The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

En este ejemplo, el bucle while se ejecutará mientras el valor de contador sea menor que 5. En cada iteración, se imprime el valor del contador y luego se incrementa en 1. Cuando contador alcanza el valor de 5, la condición `contador < 5` se vuelve falsa y el bucle se detiene.

Bucle do-while

- Es una estructura de control que permite repetir un bloque de código mientras una condición especificada sea verdadera.
- El bloque de código dentro del bucle do-while se ejecutará al menos una vez, ya que la condición se evalúa después de la ejecución del bloque.
- Si la condición es verdadera después de la primera ejecución, el bloque de código se ejecutará nuevamente. Esto continuará hasta que la condición sea falsa.

Bucle do-while

```
do {  
    // Bloque de código  
} while (condición);
```

Sintaxis del operador ternario

```
Scanner scanner = new Scanner(System.in);
int numero;

do {
    System.out.print("Ingresa un número entre 1 y 10: ");
    numero = scanner.nextInt();
} while (numero < 1 || numero > 10);

System.out.println("Número válido ingresado: " + numero);
```

En este ejemplo, el programa solicita al usuario que ingrese un número entre 1 y 10 utilizando un bucle do-while. La condición del bucle do-while es `numero < 1 || numero > 10`, lo que significa que el bloque de código se ejecutará al menos una vez y se repetirá mientras el número ingresado no esté en el rango especificado.

Bucle For

Es un bucle determinado: Son aquellos en los que se sabe de antemano cuántas veces se va a ejecutar el código que hay en su interior.

El bucle for es una construcción general que está controlada por un contador que se actualiza después de cada iteración.

Sintaxis bucle for

```
for (inicialización; condición; iteración) {  
    // cuerpo del bucle  
}
```

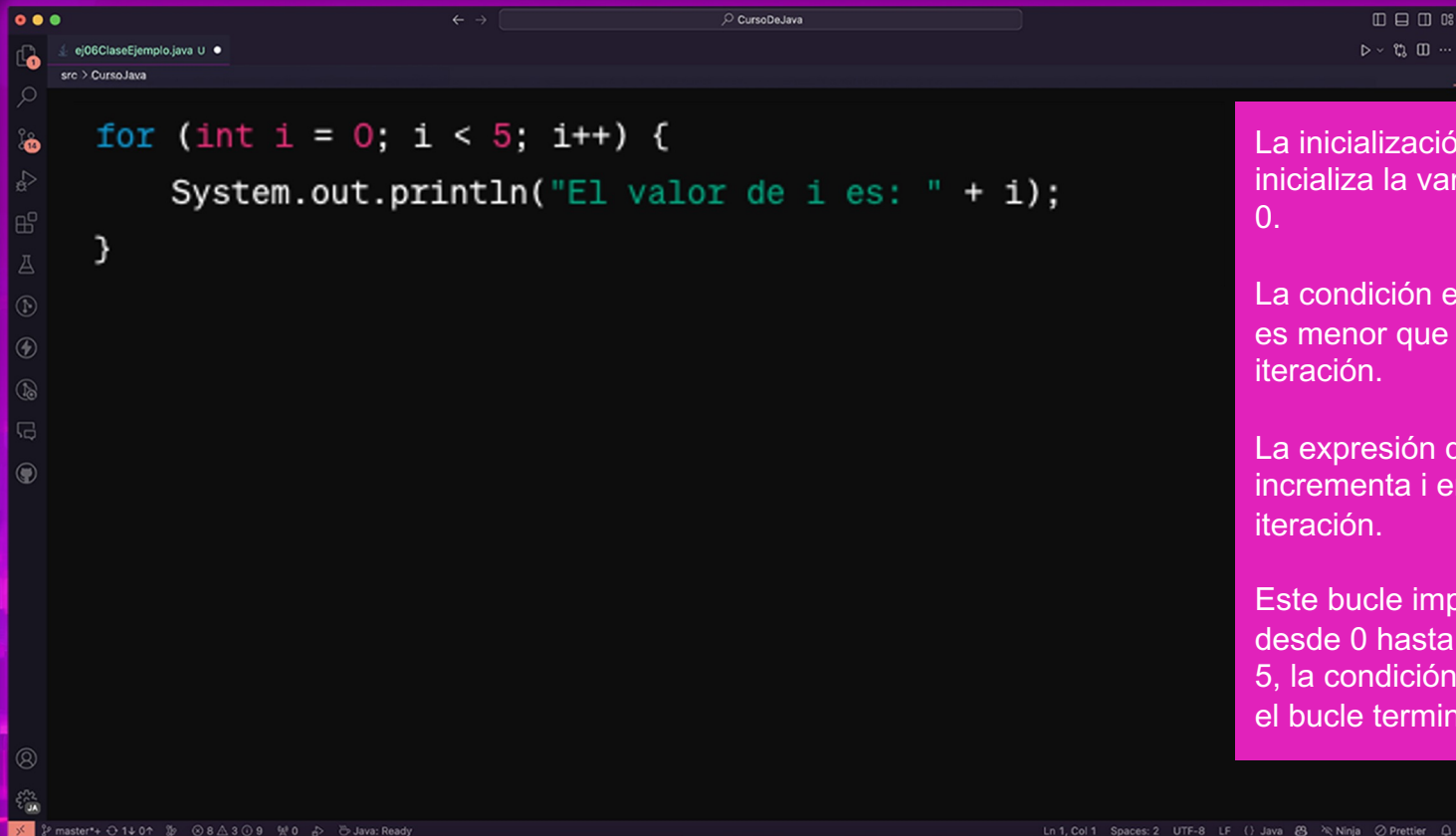
Bucle For

La inicialización se ejecuta una vez antes de que comience la iteración del bucle. Es donde generalmente se inicializan las variables de control del bucle.

La condición es una expresión booleana que se evalúa antes de cada iteración del bucle. Si la condición es verdadera, se ejecuta el cuerpo del bucle; si es falsa, el bucle termina y la ejecución continúa con la instrucción siguiente al bucle for.

La expresión de iteración se ejecuta al final de cada iteración del bucle, generalmente se utiliza para actualizar las variables de control del bucle.

Sintaxis bucle for



```
ej06ClaseEjemplo.java U
src > CursoJava

for (int i = 0; i < 5; i++) {
    System.out.println("El valor de i es: " + i);
}
```

The screenshot shows a code editor window titled 'ej06ClaseEjemplo.java U' with a file explorer on the left showing 'src > CursoJava'. The code is a Java for loop that prints the value of 'i' from 0 to 4. The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

La inicialización es `int i = 0;`, lo que inicializa la variable `i` con un valor de 0.

La condición es `i < 5;`, que verifica si `i` es menor que 5 antes de cada iteración.

La expresión de iteración es `i++`, que incrementa `i` en 1 después de cada iteración.

Este bucle imprimirá los valores de `i` desde 0 hasta 4. Una vez que `i` llega a 5, la condición `i < 5` se vuelve falsa y el bucle termina.

Bucle For-Each

El bucle for-each, también conocido como bucle mejorado, es una estructura de control en Java que permite recorrer fácilmente todos los elementos de una colección o un arreglo sin necesidad de utilizar un índice explícito. Esto hace que el código sea más limpio y fácil de entender.

es especialmente útil cuando necesitas recorrer todos los elementos de una colección o un arreglo sin preocuparte por el índice de cada elemento. Además, el código es más conciso y fácil de entender en comparación con el bucle for tradicional.

Sintaxis del bucle for-each

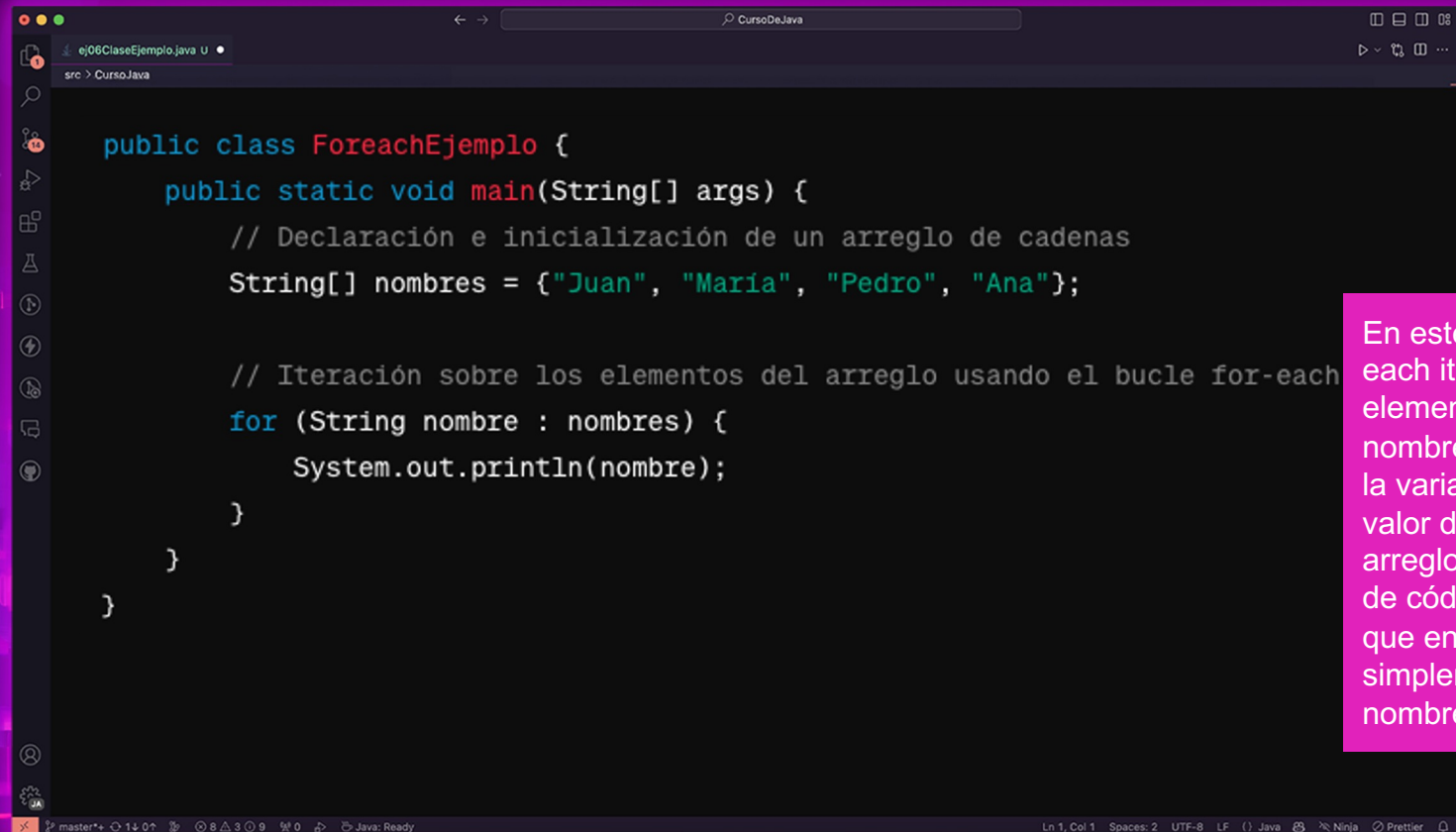
```
for (tipo elemento : colección) {  
    // Cuerpo del bucle  
}
```

tipo es el tipo de dato de los elementos de la colección.

elemento es una variable que representa cada elemento de la colección en cada iteración del bucle.

colección es la colección o arreglo a recorrer.

Uso del bucle for-each con un arreglo

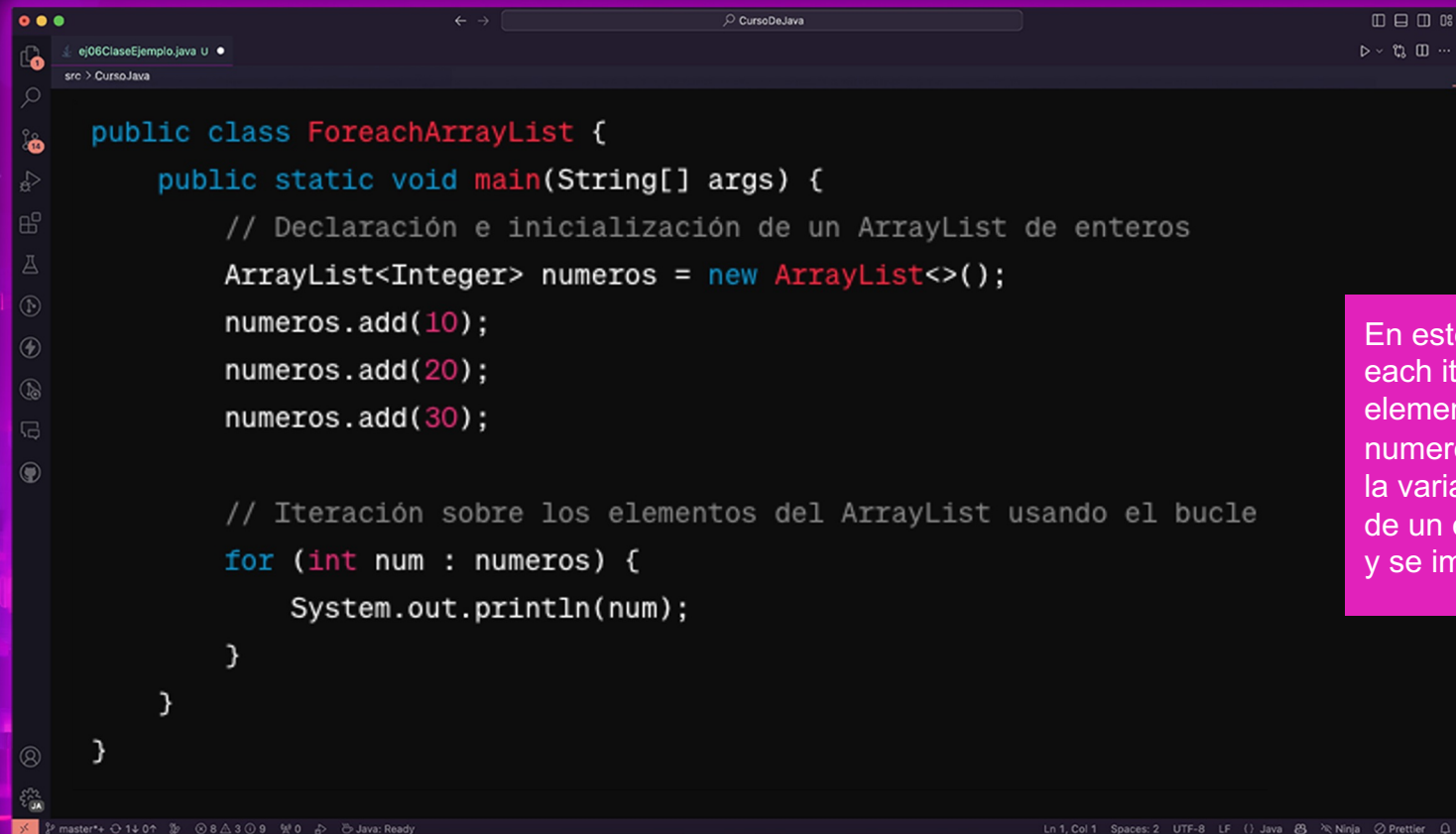
A screenshot of a code editor window. The title bar shows 'ej06ClaseEjemplo.java' and 'CursoDeJava'. The code is in Java and demonstrates a for-each loop. The code is as follows:

```
public class ForeachEjemplo {  
    public static void main(String[] args) {  
        // Declaración e inicialización de un arreglo de cadenas  
        String[] nombres = {"Juan", "María", "Pedro", "Ana"};  
  
        // Iteración sobre los elementos del arreglo usando el bucle for-each  
        for (String nombre : nombres) {  
            System.out.println(nombre);  
        }  
    }  
}
```

The editor has a dark theme. The status bar at the bottom shows 'master+ 140', '8 3 0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

En este ejemplo, el bucle for-each itera sobre todos los elementos del arreglo nombres. En cada iteración, la variable nombre toma el valor de un elemento del arreglo y se ejecuta el bloque de código dentro del bucle, que en este caso simplemente imprime el nombre en la consola.

Uso del bucle for-each con un arreglo

A screenshot of an IDE window titled 'CursoDeJava'. The code is in a file named 'ej06ClaseEjemplo.java'. It defines a public class 'ForeachArrayList' with a static 'main' method. Inside 'main', an 'ArrayList<Integer>' named 'numeros' is created and populated with the values 10, 20, and 30. A 'for' loop with the syntax 'for (int num : numeros)' iterates over each element, printing it to the console using 'System.out.println(num)'. The IDE interface includes a sidebar with icons for Explorer, Search, Run and Debug, Source, Test, Extensions, Output, and Settings. The status bar at the bottom shows 'master+ 140', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

En este ejemplo, el bucle for-each itera sobre todos los elementos del ArrayList `numeros`. En cada iteración, la variable `num` toma el valor de un elemento del ArrayList y se imprime en la consola.

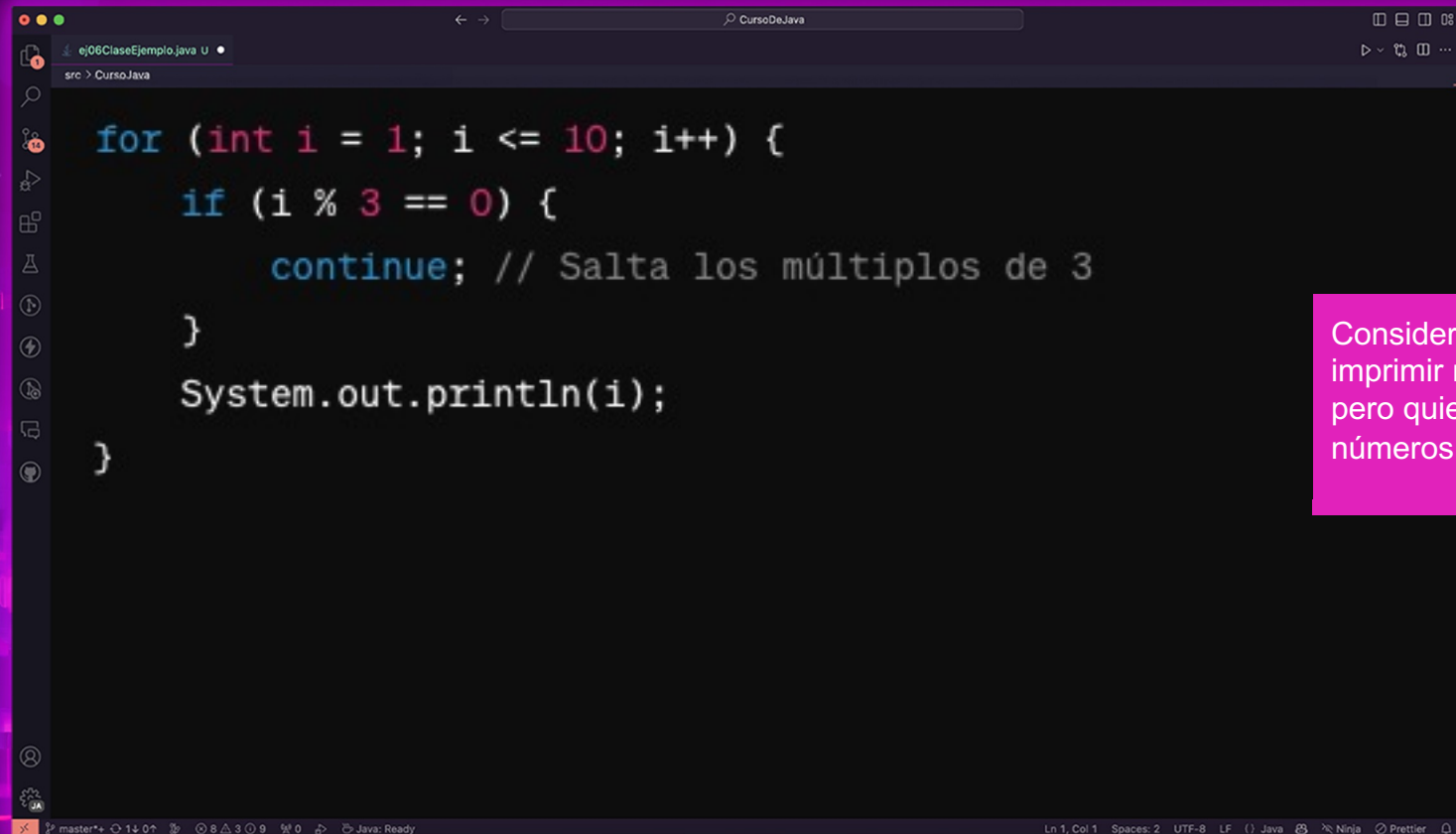
continue

Es utilizada para controlar el flujo de bucles, como for, while, y do-while. Su propósito es inmediatamente terminar la iteración actual del bucle en el que se encuentra, saltándose cualquier código que siga después de continue dentro de esa iteración, y procediendo directamente con la próxima iteración del bucle.

Funcionamiento del continue

Cuando el flujo del programa llega a una instrucción continue, el bucle omite el resto de su cuerpo para esa iteración particular y continúa con la siguiente iteración. Esto es especialmente útil cuando necesitas saltar sobre ciertas iteraciones basadas en condiciones específicas, sin salir completamente del bucle.

Uso del continue



```
for (int i = 1; i <= 10; i++) {  
    if (i % 3 == 0) {  
        continue; // Salta los múltiplos de 3  
    }  
    System.out.println(i);  
}
```

The screenshot shows a code editor window titled 'CursoDeJava' with a file named 'ej06ClaseEjemplo.java'. The code is a Java loop that prints numbers from 1 to 10, skipping multiples of 3. The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

Consideremos que deseas imprimir números del 1 al 10, pero quieres omitir todos los números que son múltiplos de 3

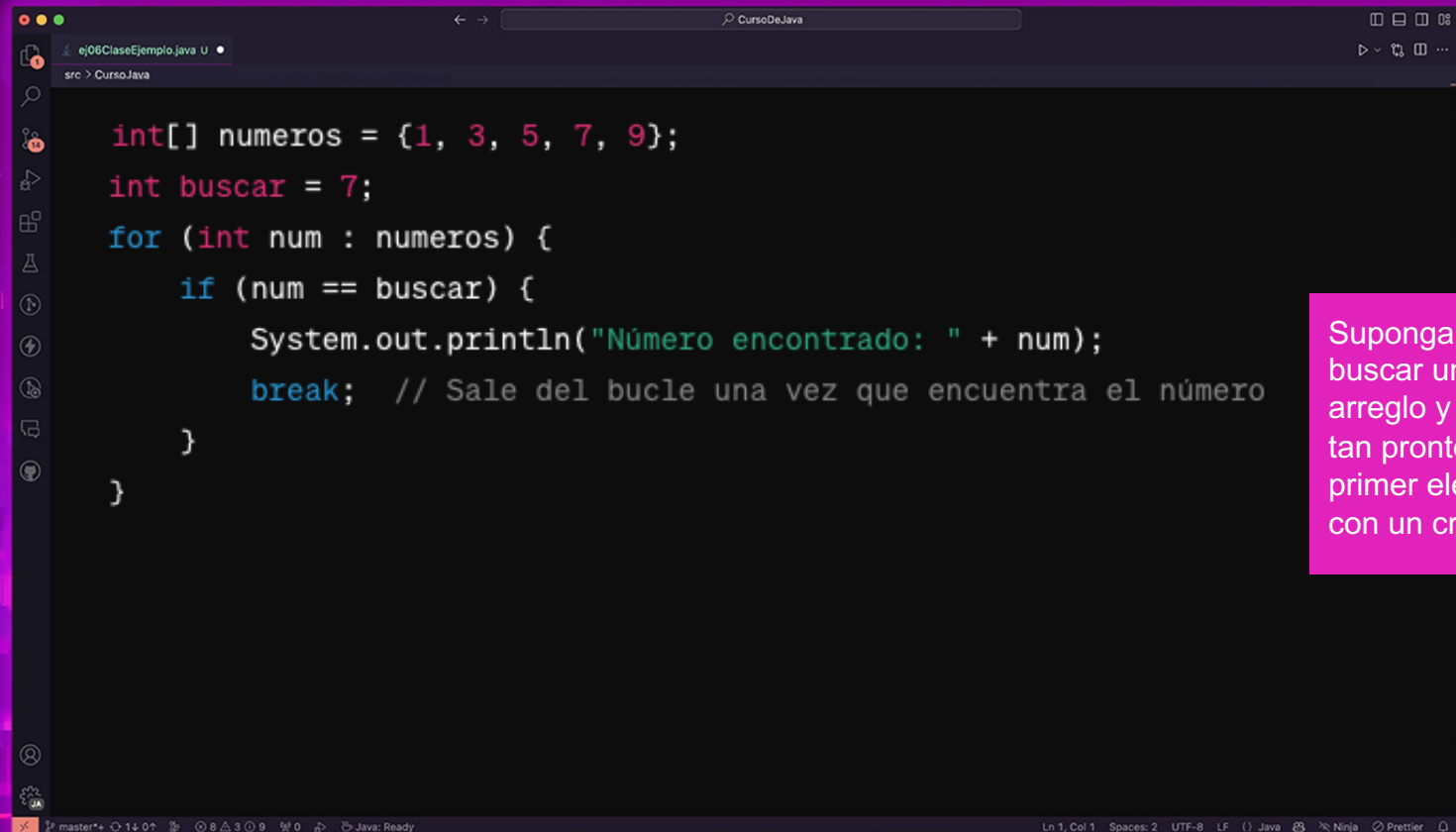
break

se utiliza para terminar de manera inmediata un bucle, independientemente del estado en que se encuentre su condición de continuación. Break puede ser usado en todos los tipos de bucles, incluyendo for, while y do-while, así como en estructuras de selección como switch.

Funcionamiento del break

Cuando el flujo del programa llega a una instrucción break, el bucle o la estructura switch en la que se encuentra se termina inmediatamente. El control del programa se transfiere al bloque de código que sigue inmediatamente después del bucle o switch.

Uso del break



```
int[] numeros = {1, 3, 5, 7, 9};
int buscar = 7;
for (int num : numeros) {
    if (num == buscar) {
        System.out.println("Número encontrado: " + num);
        break; // Sale del bucle una vez que encuentra el número
    }
}
```

The screenshot shows a code editor window titled 'CursoDeJava' with a file named 'ej06ClaseEjemplo.java'. The code is a Java program that searches for the number 7 in an array of integers {1, 3, 5, 7, 9}. It uses a for-each loop and a break statement to exit the loop as soon as the target number is found. The IDE interface includes a sidebar with icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, and Settings. The status bar at the bottom indicates 'master*+ 14 0', '8 3 0 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

Supongamos que necesitas buscar un elemento en un arreglo y detener la búsqueda tan pronto como encuentres el primer elemento que cumpla con un criterio específico