

Programación I

Variables

Variables

En el contexto de las funciones en Java, las variables locales y globales se refieren a cómo se almacenan y acceden a las variables dentro de una función y en el ámbito más amplio del programa.

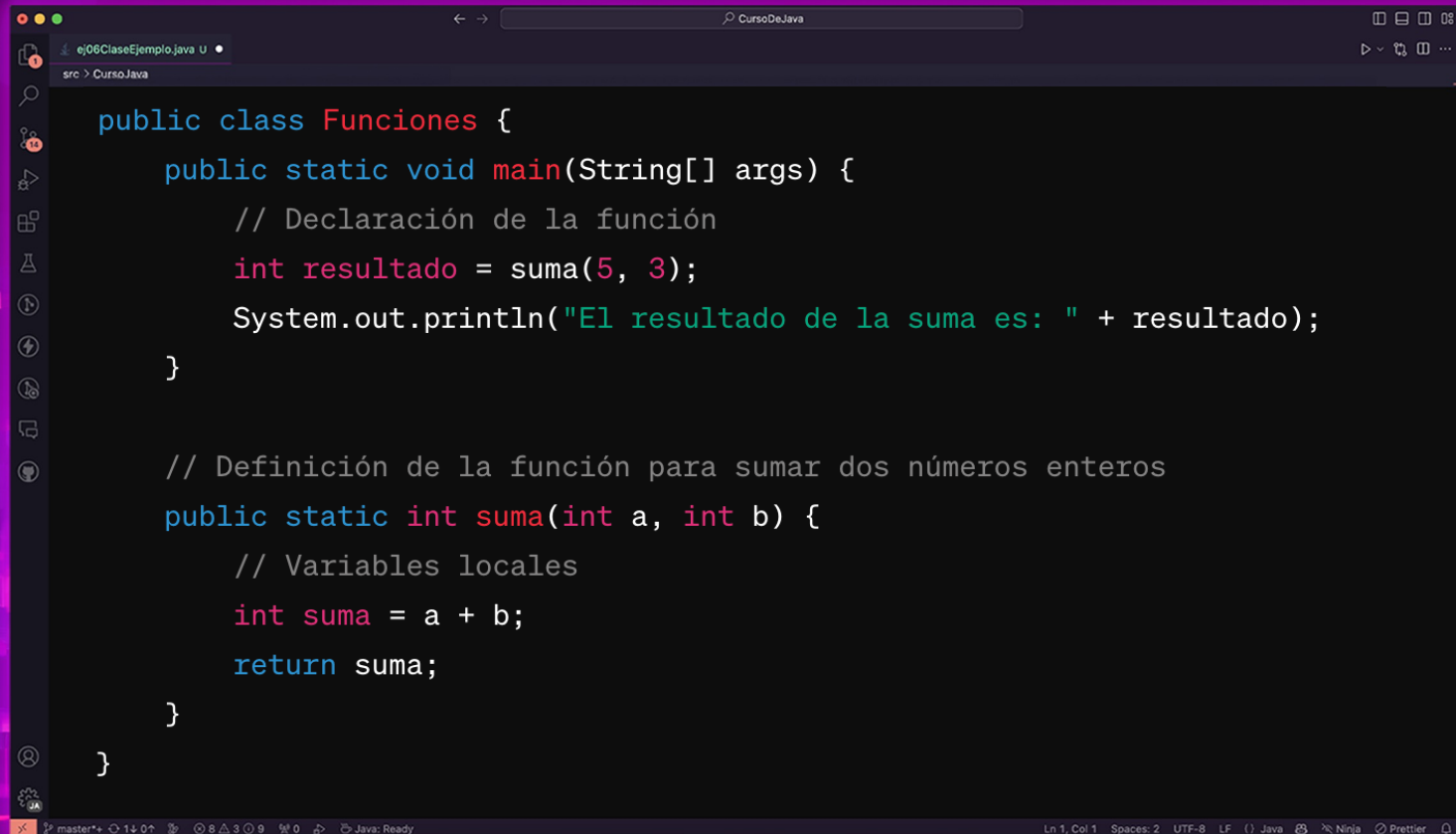
El uso excesivo de variables globales puede hacer que el código sea más difícil de entender y depurar.

Se recomienda utilizar variables locales siempre que sea posible, ya que tienen un ámbito más limitado y son más seguras en términos de encapsulamiento y mantenibilidad del código.

Variables locales

- Las variables locales son variables declaradas dentro de una función y solo son accesibles dentro de esa función.
- Estas variables tienen un ámbito limitado que se extiende desde el punto donde se declaran hasta el final de la función donde están definidas.
- Las variables locales se crean cuando la función se llama y se eliminan cuando la función termina su ejecución.
- No se puede acceder a las variables locales desde fuera de la función en la que se declaran.
- Las variables locales deben ser inicializadas antes de ser utilizadas en la función.

Ejemplo de variables locales

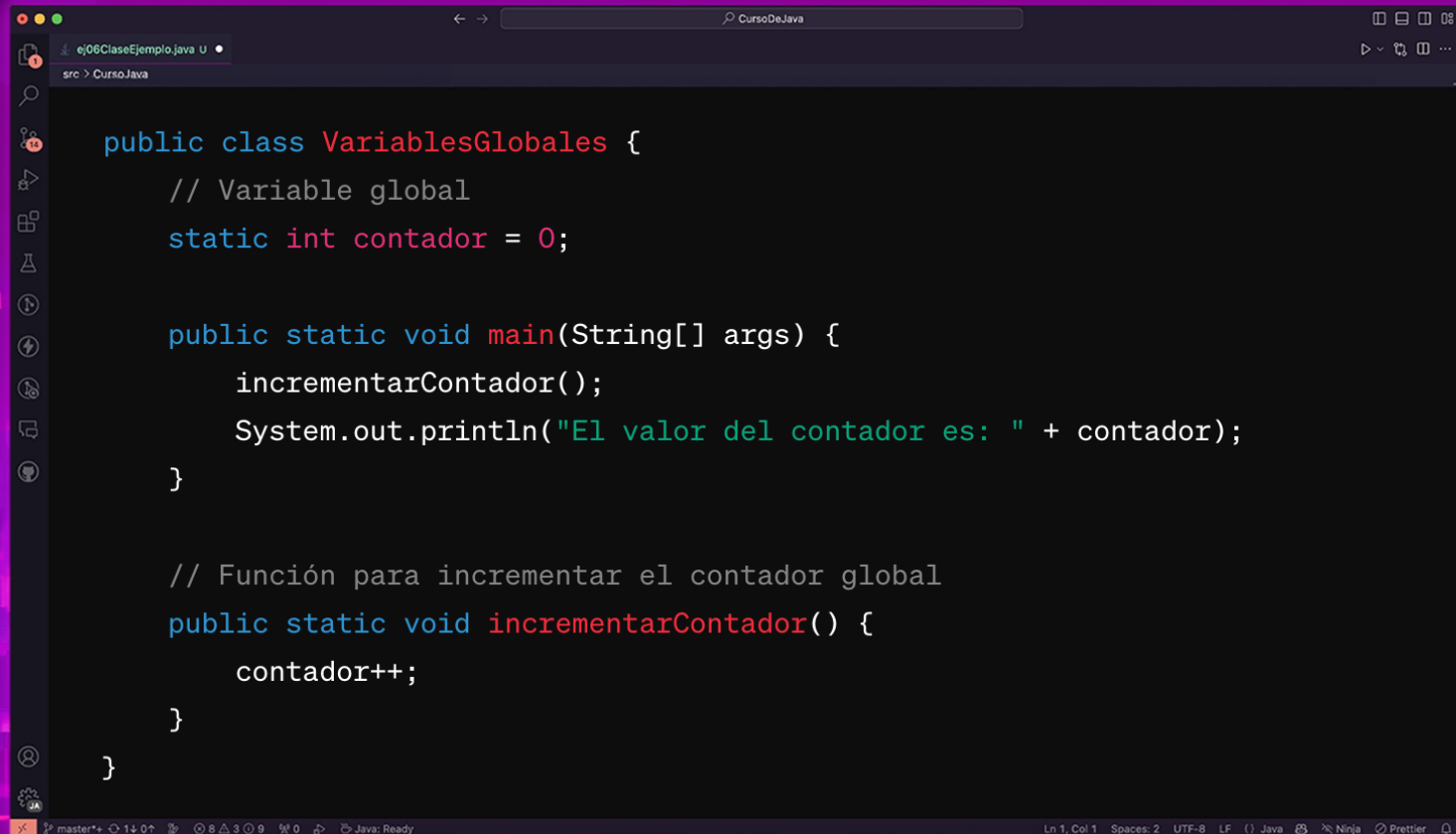
A screenshot of a code editor window titled 'CursoDeJava'. The editor shows a Java file named 'ej06ClaseEjemplo.java'. The code defines a class 'Funciones' with a 'main' method and a 'suma' method. The 'main' method calls 'suma(5, 3)' and prints the result. The 'suma' method uses local variables 'a' and 'b' to calculate the sum. The editor interface includes a sidebar with icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, Remote Explorer, Docker, and Accounts. The status bar at the bottom shows 'master*+', '14 0%', '8 3 0 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

```
public class Funciones {  
    public static void main(String[] args) {  
        // Declaración de la función  
        int resultado = suma(5, 3);  
        System.out.println("El resultado de la suma es: " + resultado);  
    }  
  
    // Definición de la función para sumar dos números enteros  
    public static int suma(int a, int b) {  
        // Variables locales  
        int suma = a + b;  
        return suma;  
    }  
}
```


Variables globales

- Las variables globales son variables declaradas fuera de cualquier función y son accesibles desde cualquier parte del programa.
- Estas variables tienen un ámbito que se extiende a lo largo de todo el programa.
- Las variables globales se crean cuando se inicia el programa y se destruyen cuando el programa termina su ejecución.
- Pueden ser accedidas y modificadas por cualquier función en el programa.

Ejemplo de variables Globales

A screenshot of a code editor window with a dark theme. The title bar shows 'CursoDeJava'. The file explorer on the left shows 'ej06ClaseEjemplo.java'. The code is a Java class named 'VariablesGlobales' with a static variable 'contador' and a static method 'main'. The status bar at the bottom shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

```
public class VariablesGlobales {  
    // Variable global  
    static int contador = 0;  
  
    public static void main(String[] args) {  
        incrementarContador();  
        System.out.println("El valor del contador es: " + contador);  
    }  
  
    // Función para incrementar el contador global  
    public static void incrementarContador() {  
        contador++;  
    }  
}
```

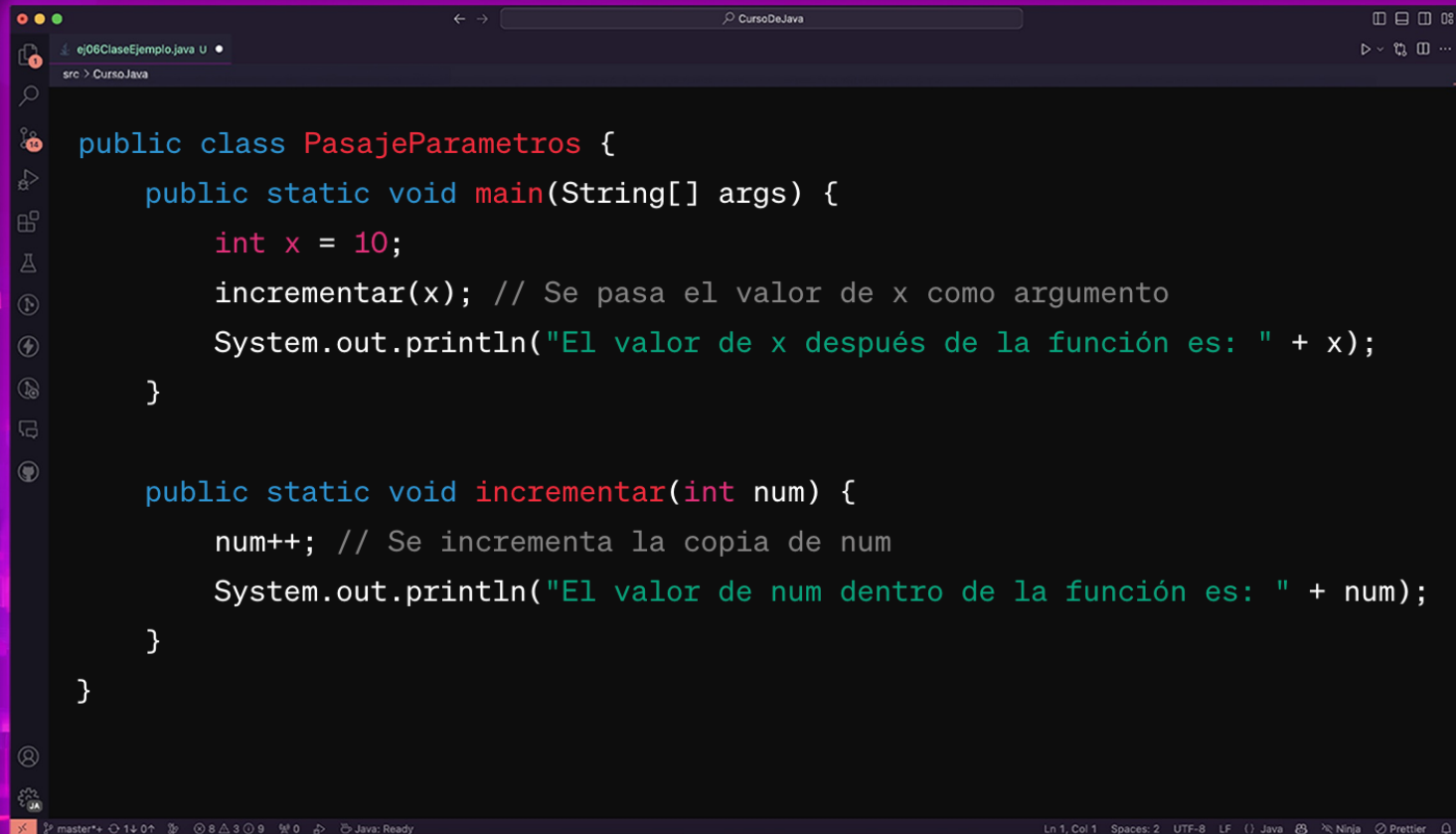
Pasaje de Parámetros

El pasaje de parámetros se refiere a cómo se transfieren los valores entre una función y la parte del programa que la llama. En Java, existen principalmente dos formas de pasar parámetros a una función: por valor y por referencia.

Pasaje de parámetros por valor

- En el pasaje por valor, se pasa una copia del valor de la variable como argumento a la función.
- La función trabaja con una copia de la variable original y cualquier modificación realizada en la copia no afecta a la variable original.
- Este es el método predeterminado para pasar parámetros en Java para tipos primitivos y objetos inmutables.

Ejemplo de variables Globales



The image shows a screenshot of a code editor window. The title bar at the top reads "CursoDeJava". The editor contains the following Java code:

```
public class PasajeParametros {  
    public static void main(String[] args) {  
        int x = 10;  
        incrementar(x); // Se pasa el valor de x como argumento  
        System.out.println("El valor de x después de la función es: " + x);  
    }  
  
    public static void incrementar(int num) {  
        num++; // Se incrementa la copia de num  
        System.out.println("El valor de num dentro de la función es: " + num);  
    }  
}
```

The code defines a class named `PasajeParametros` with two static methods. The `main` method declares a local variable `x` and calls the `incrementar` method, passing `x` as an argument. The `incrementar` method increments its parameter `num` and prints its value. The output of the program would be:

```
El valor de x después de la función es: 10  
El valor de num dentro de la función es: 11
```

The status bar at the bottom indicates the file is "master+1", the cursor is at "Ln 1, Col 1", and the encoding is "UTF-8". The editor is using the "Prettier" formatter.

Pasaje de parámetros por referencia

- En el pasaje por referencia, se pasa una referencia a la variable como argumento a la función.
- La función trabaja con la misma instancia de objeto que la variable original y cualquier modificación realizada en la referencia dentro de la función afecta a la variable original.
- Este método se utiliza para pasar parámetros en Java para objetos mutables.

Ejemplo de variables Globales

```
ej06ClaseEjemplo.java U
src > CursoJava

import java.util.ArrayList;

public class PasajeParametros {
    public static void main(String[] args) {
        ArrayList<String> lista = new ArrayList<>();
        lista.add("Hola");
        agregarElemento(lista); // Se pasa la referencia de lista como argumento
        System.out.println("Lista después de la función: " + lista);
    }

    public static void agregarElemento(ArrayList<String> lista) {
        lista.add("Mundo"); // Se agrega un elemento a la lista
    }
}
```

master*+ 140 8 3 0 0 Java: Ready Ln 1, Col 1 Spaces: 2 UTF-8 LF () Java Ninja Prettier

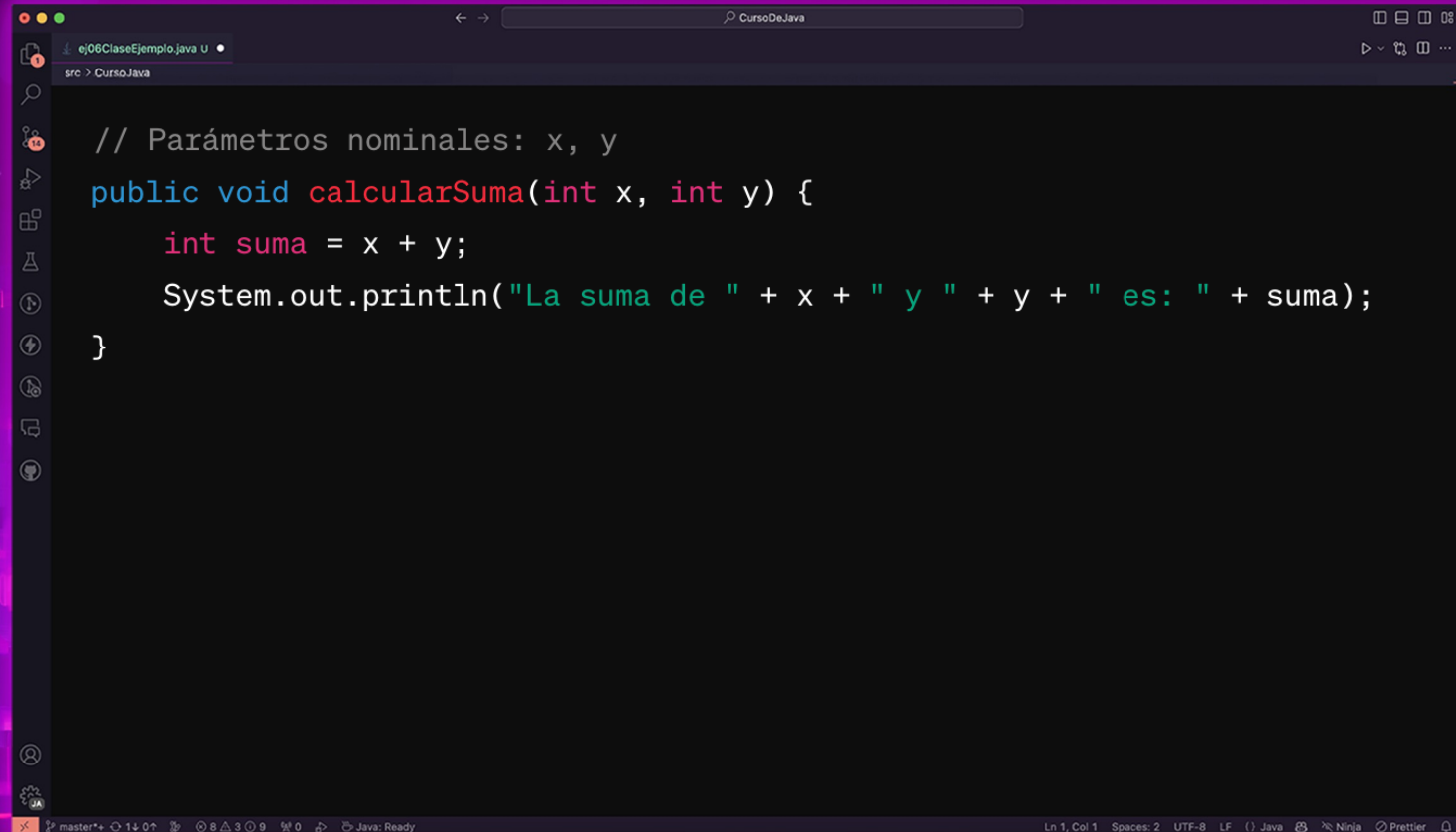
Parámetros nominales y efectivos

Los parámetros nominales y efectivos son conceptos relacionados con la forma en que se definen y utilizan los parámetros en una función o método

Parámetros nominales

- Los parámetros nominales son los nombres de los parámetros que se definen en la declaración de la función o método.
- Estos nombres son utilizados para identificar los valores que se pasan a la función cuando se llama.
- Son útiles para proporcionar significado y contexto a los valores que se pasan a la función.

Parámetros nominales



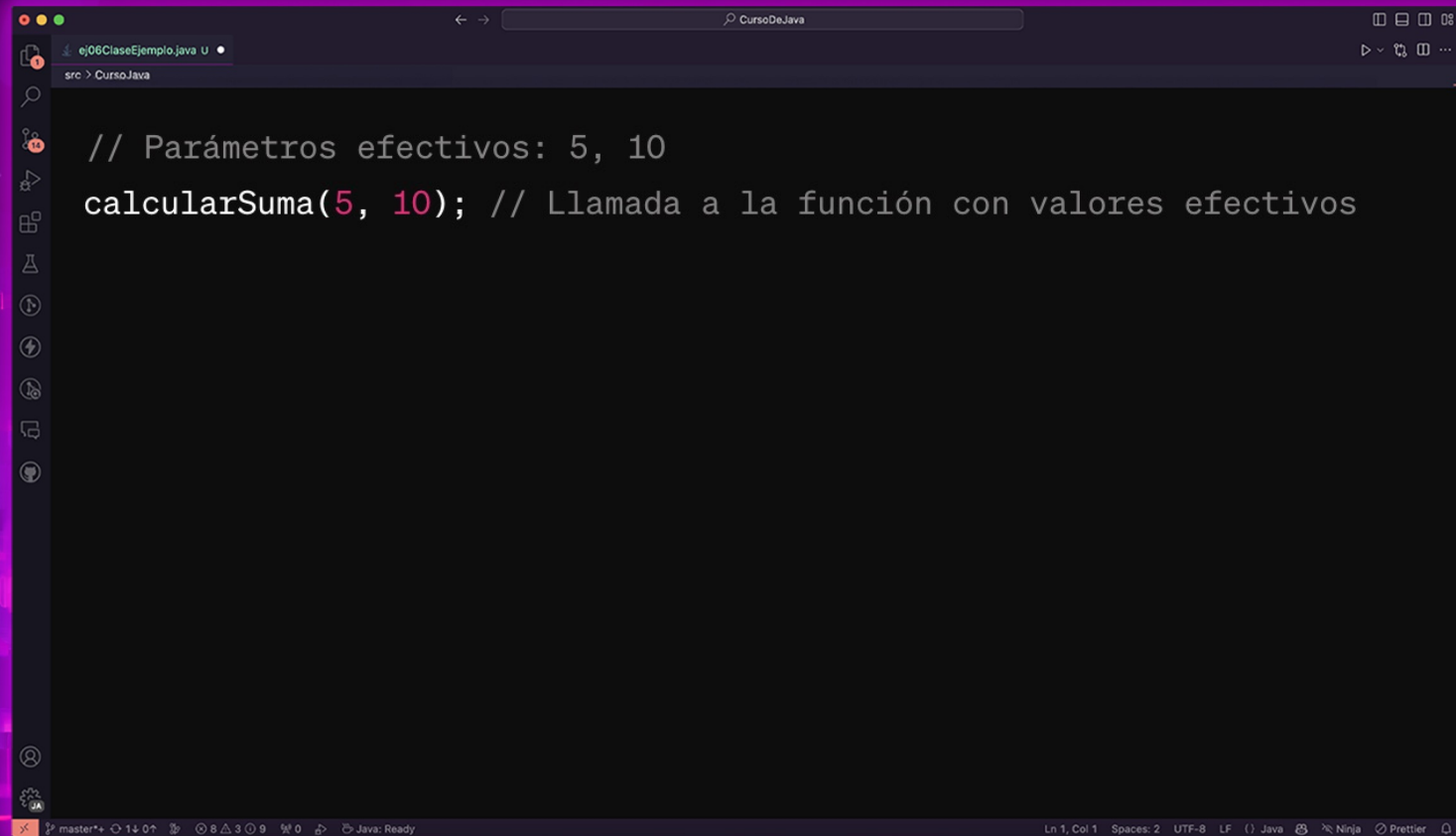
The image shows a code editor window with a dark theme. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The code is written in Java and demonstrates nominal parameters. It includes a comment, a method signature, and a method body with a calculation and a print statement. The editor's interface includes a sidebar with icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, Remote Explorer, Docker, and Settings. The status bar at the bottom shows 'master*+', '14 0 0', '8 3 0 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

```
// Parámetros nominales: x, y
public void calcularSuma(int x, int y) {
    int suma = x + y;
    System.out.println("La suma de " + x + " y " + y + " es: " + suma);
}
```

Parámetros efectivos

- Los parámetros efectivos son los valores reales que se pasan a la función cuando se llama.
- Estos valores pueden ser constantes, variables o expresiones que coinciden con los tipos de datos y el número de parámetros definidos en la función.
- Los parámetros efectivos son los valores que la función utiliza para llevar a cabo sus operaciones.

Parámetros nominales



The image shows a screenshot of a code editor window. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The editor contains two lines of Java code: a comment '// Parámetros efectivos: 5, 10' and a function call 'calcularSuma(5, 10); // Llamada a la función con valores efectivos'. The code is written in a dark theme with syntax highlighting. The background of the slide features a stylized, colorful cityscape at night.

```
// Parámetros efectivos: 5, 10
calcularSuma(5, 10); // Llamada a la función con valores efectivos
```