

Comparación en Java.

En java para comparar datos básicos como enteros o carácter se usa `==`, pero no ocurre lo mismo con objetos.

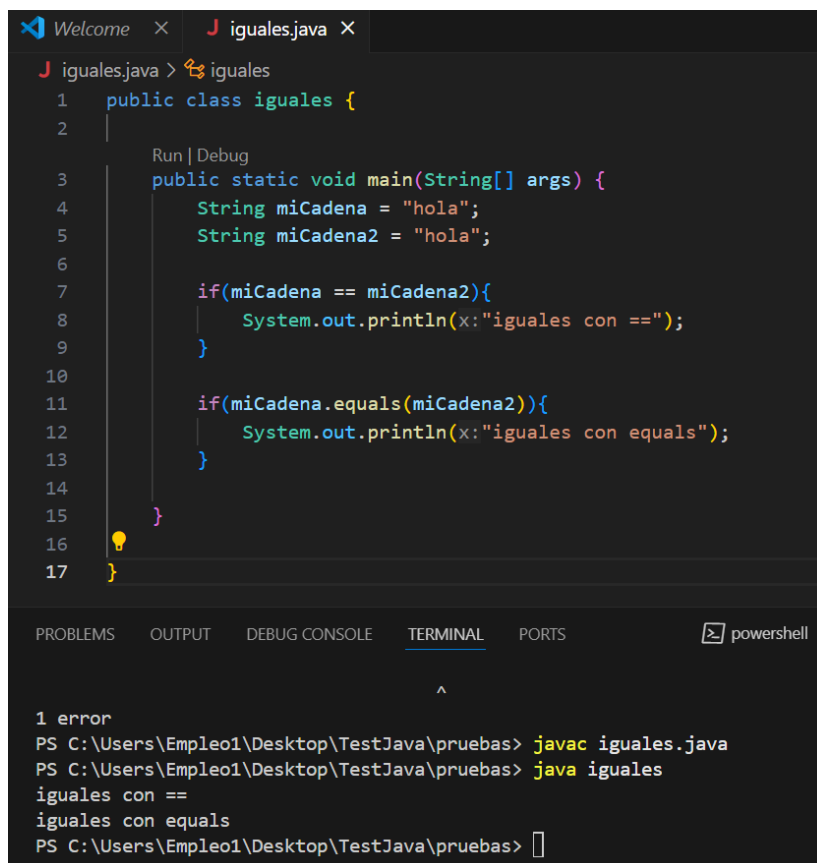
En Java, la comparación con `==` y `equals()` funciona de manera diferente cuando se trata de objetos:

1. **== (Doble Igual):** Compara las **referencias** en memoria. Es decir, verifica si ambas variables apuntan al mismo objeto.
2. **equals():** Compara el **contenido** de los objetos (si la clase lo ha sobrescrito adecuadamente).

Caso con String

Si `miCadena` y `miCadena2` contienen el mismo valor y fueron asignadas de la misma forma, el resultado varía:

Ejemplo 1: Usando literales de cadena



```
iguales.java
1 public class iguales {
2
3     public static void main(String[] args) {
4         String miCadena = "hola";
5         String miCadena2 = "hola";
6
7         if(miCadena == miCadena2){
8             System.out.println(x:"iguales con ==");
9         }
10
11         if(miCadena.equals(miCadena2)){
12             System.out.println(x:"iguales con equals");
13         }
14     }
15 }
16
17 }

1 error
PS C:\Users\Empleo1\Desktop\TestJava\pruebas> javac iguales.java
PS C:\Users\Empleo1\Desktop\TestJava\pruebas> java iguales
iguales con ==
iguales con equals
PS C:\Users\Empleo1\Desktop\TestJava\pruebas>
```

Explicación:

- En este caso, ambas variables apuntan al mismo objeto en el **pool de cadenas de Java**, por lo que `==` devuelve true.
- `equals()` compara los valores y también devuelve true.

Ejemplo 2: Usando `new String()`

```
J iguales.java > iguales > main(String[])
1  public class iguales {
3  public static void main(String[] args) {
14
15      String miCadena = new String(original:"Hola");
16      String miCadena2 = new String(original:"Hola");
17
18      if (miCadena == miCadena2) {
19          System.out.println(x:"iguales con ==");
20      }
21
22      if (miCadena.equals(miCadena2)) {
23          System.out.println(x:"iguales con equals");
24      }
25
26  }
27
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS power

```
PS C:\Users\Empleo1\Desktop\TestJava\pruebas> java iguales
iguales con equals
PS C:\Users\Empleo1\Desktop\TestJava\pruebas>
```

Explicación:

- `new String("Hola")` crea **dos objetos distintos en memoria**, por lo que `miCadena == miCadena2` es false.
- `equals()` compara los valores, que son iguales, por lo que devuelve true.

Conclusión

Si trabajas con **tipos primitivos**, `==` es seguro.

Para objetos, usa `equals()` para comparar contenido y `==` solo si quieres verificar si son exactamente la misma instancia.



Cofinanciado por
la Unión Europea



MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL



Fondos Europeos



Nota:

El **pool de cadenas de Java** es una sección especial de memoria donde Java guarda las cadenas (String) creadas como **literales** para **reutilizarlas** y ahorrar memoria.

Ejemplo:

```
String a = "Hola";  
String b = "Hola";  
  
System.out.println(a == b); // true (apuntan al mismo objeto en el pool)
```

Aquí, "Hola" se almacena una sola vez en el **pool**, y ambas variables apuntan a la misma referencia.

Pero si usamos `new String()`, se crea un objeto nuevo **fuera del pool**:

```
String c = new String("Hola");  
System.out.println(a == c); // false (son objetos diferentes en memoria)
```

conclusión:

Las cadenas literales se guardan en el **pool de cadenas** y se comparten, pero los objetos creados con `new String()` son independientes.



REGISTRO NACIONAL DE ASOCIACIONES N°611922
DECLARADA ENTIDAD DE UTILIDAD PÚBLICA ESTATAL
AGENCIA DE COLOCACIÓN: ID 0100000017

CENTRO EN MÁLAGA
C/DOS ACERAS 23, 29012
MÁLAGA | (+34) 952 300 500
ARRABAL@ARRABALEMPLEO.ORG

CENTRO EN CÁDIZ
TR.ª ALAMEDA DE SOLANO, 32, 11130
CHICLANA DE LA FRONTERA | (+34) 956 900 312
CHICLANA@ARRABALEMPLEO.ORG