

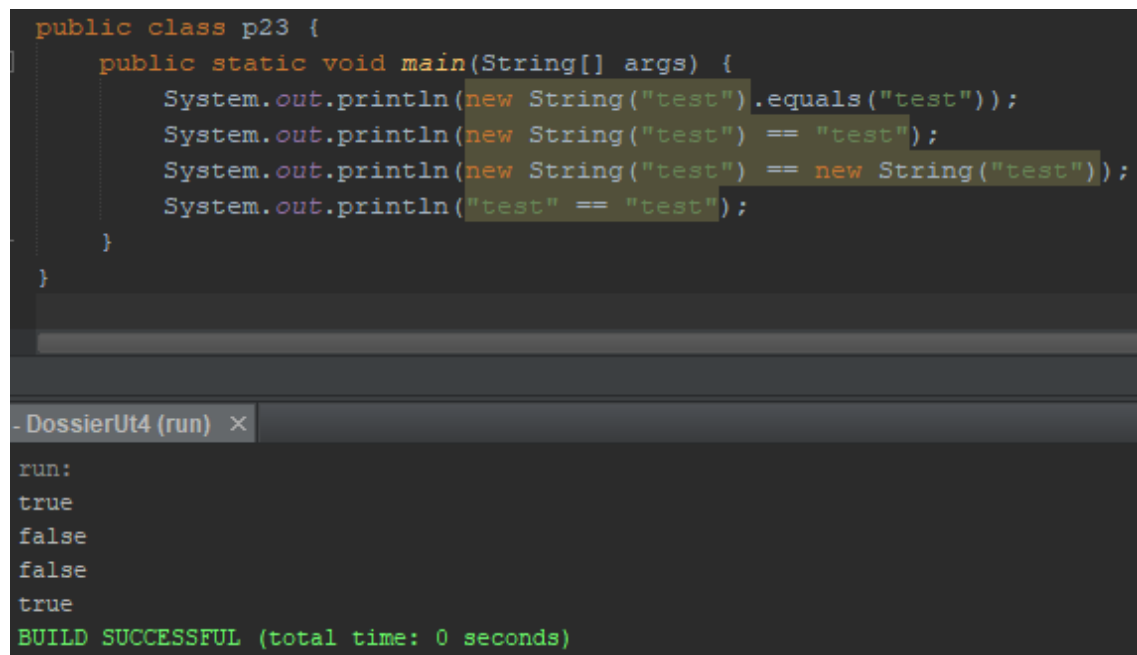
**Práctica 23:** Probar en el IDE las siguientes comparaciones. Obtener una salida en pantalla del valor booleano correspondiente y explicar por qué motivo la comparación nos sale true o false:

`new String("test").equals("test")`

`new String("test") == "test"`

`new String("test") == new String("test")`

`"test" == "test"`

A screenshot of an IDE window. The top part shows a Java class named 'p23' with a 'main' method. The method contains four lines of code: 'System.out.println(new String("test").equals("test"));', 'System.out.println(new String("test") == "test");', 'System.out.println(new String("test") == new String("test"));', and 'System.out.println("test" == "test");'. The bottom part of the screenshot shows the output of the program: 'run:', 'true', 'false', 'false', 'true', and 'BUILD SUCCESSFUL (total time: 0 seconds)'. The output corresponds to the four lines of code in the main method.

```
public class p23 {  
    public static void main(String[] args) {  
        System.out.println(new String("test").equals("test"));  
        System.out.println(new String("test") == "test");  
        System.out.println(new String("test") == new String("test"));  
        System.out.println("test" == "test");  
    }  
}
```

- DossierUt4 (run) ×

run:  
true  
false  
false  
true  
BUILD SUCCESSFUL (total time: 0 seconds)

En el **1er** caso, el resultado es true porque con el método equals se compara el contenido del objeto con el literal, y se encuentra que es igual.

En el **2do** caso, el resultado es false porque se está comparando la referencia del objeto con un literal, y son claramente distintas.

En el **3er** caso, el resultado es false, esto es porque se están creando y comparando dos referencias distintas para ambos objetos. Si en lugar de == usásemos equals sí que resultaría true.

En el **4to** caso, el resultado es true, ya que estamos comparando dos literales, y en este caso se compara el contenido, que es claramente idéntico.