

# Pruebas unitarias con JUnit



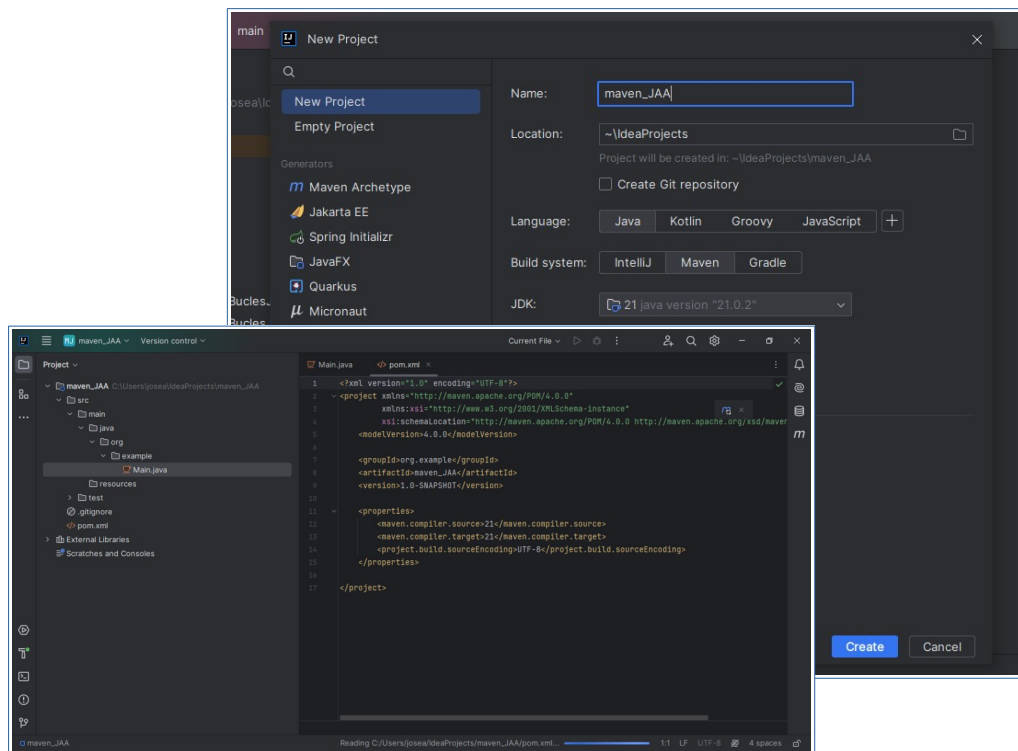
Índice

Crear proyecto con Maven e instalar JUnit.....3

Pillar librería JUnit.....3

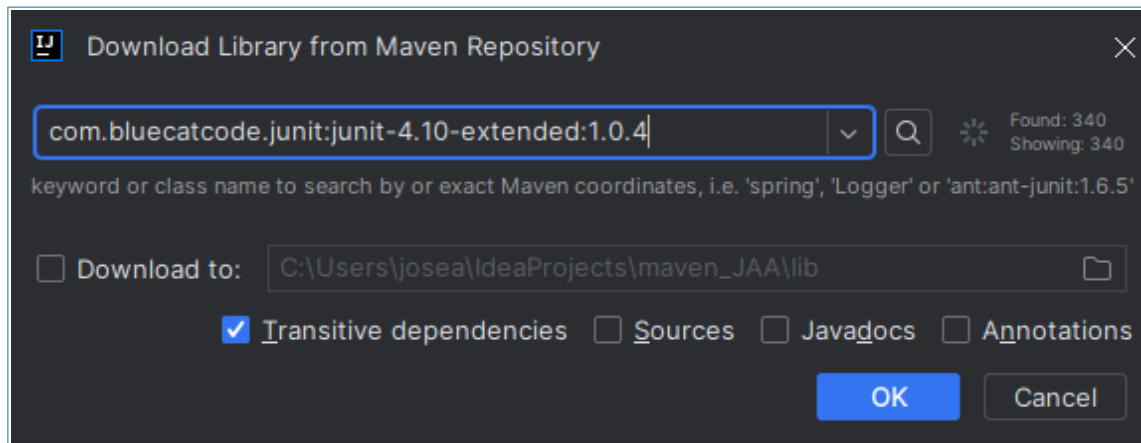
Hacer las pruebas.....6

# Crear proyecto con Maven e instalar JUnit

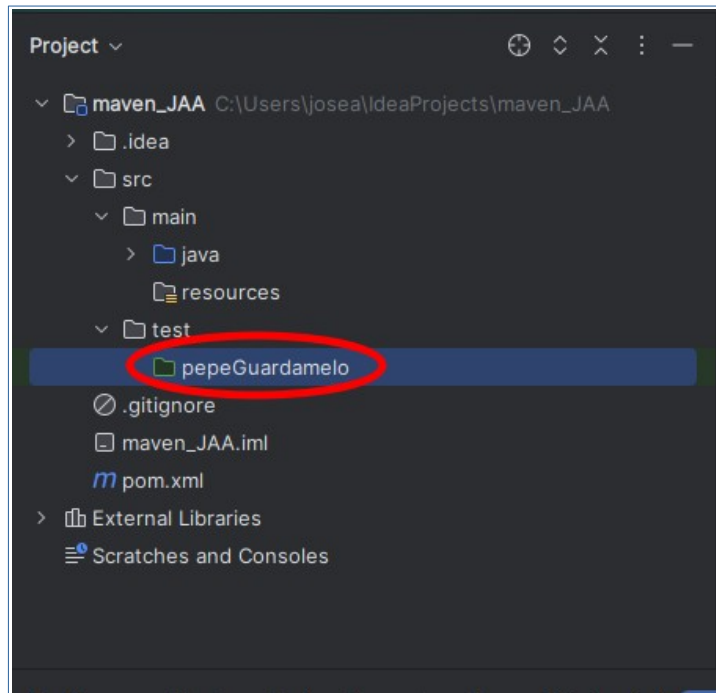


## Pillar librería JUnit

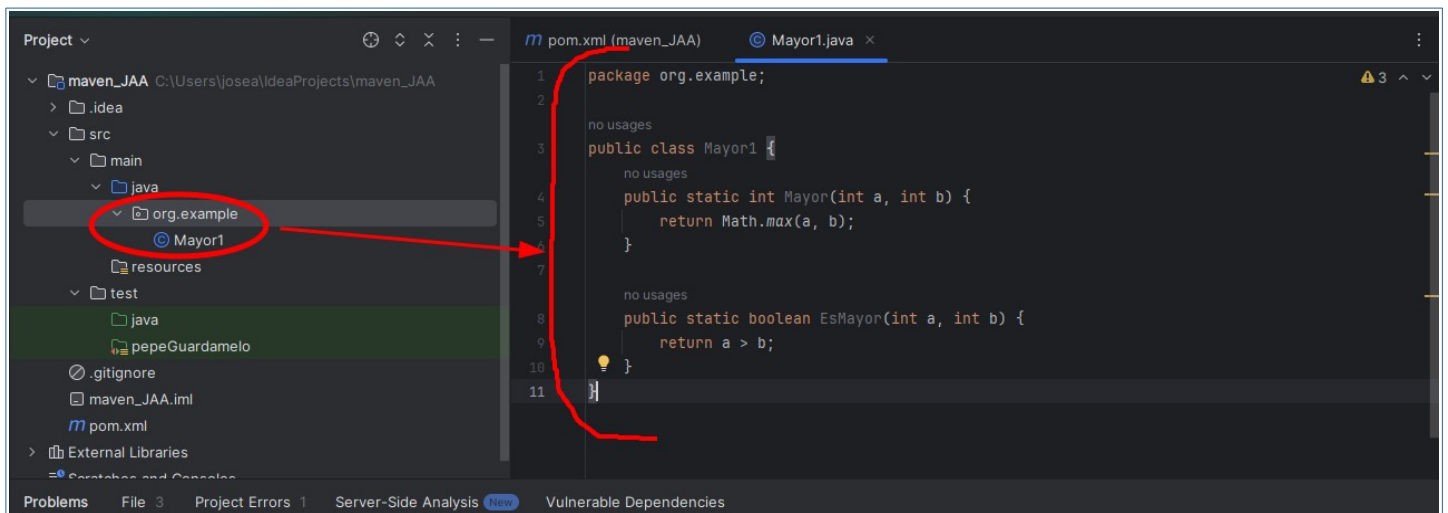
(Project structure < libraries < from maven y buscamos junit)



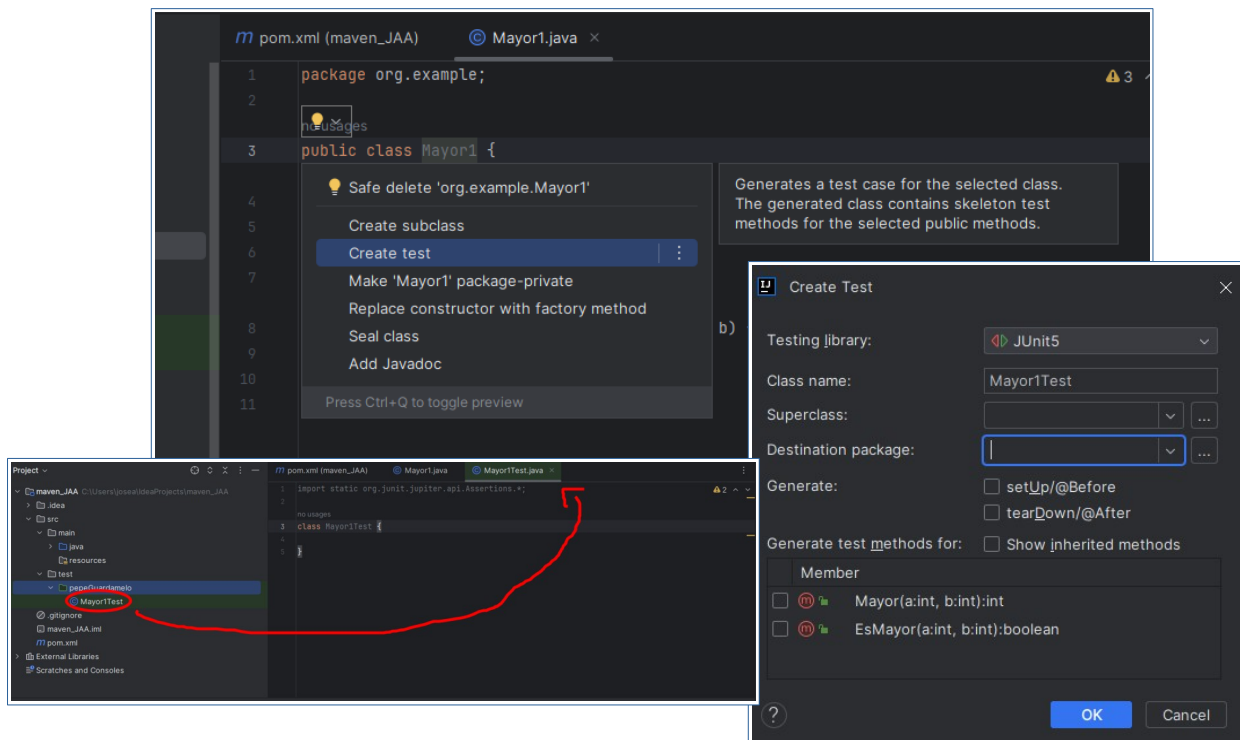
Creamos una carpeta en la carpeta test que se nos crea, y le ponemos Make directory as **Test Sources Root**, esto no es obligatorio, al hacer esto, cualquier test que hagamos en JUnit de default irán ahí



Creare un archivo con el que testearé



Pondré el cursor encima del public class y le daré click a la bombilla pa' crear el test



En JUnit hay diversos métodos de test para ver los resultados.

Estos se utilizan a través de una etiqueta llamada `@Test`.

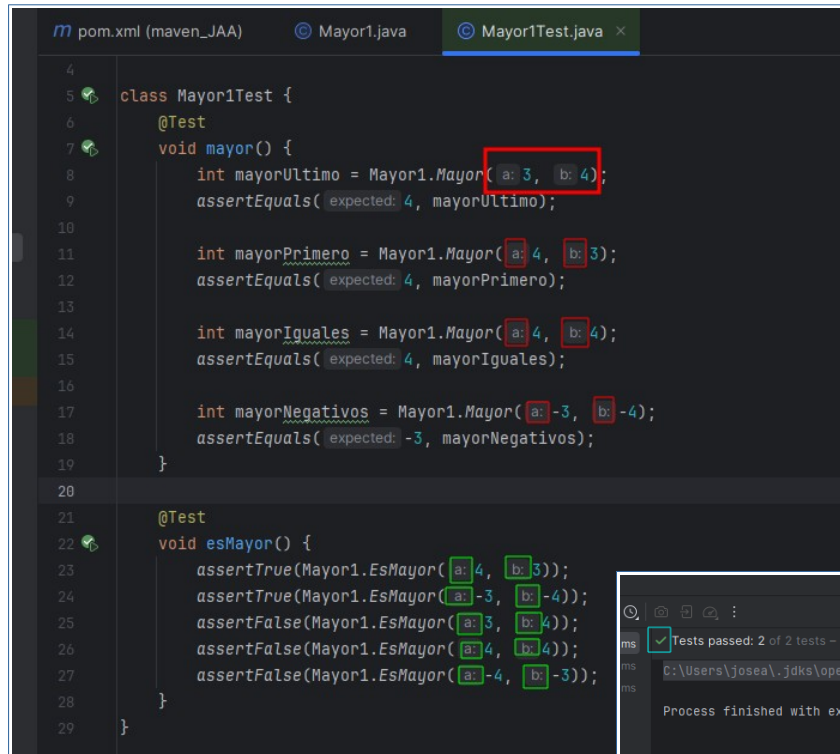
- **`assertEquals(valor esperado, valor devuelto)`:** El resultado del test será correcto si el valor esperado que le indicamos es igual al valor devuelto por nuestra función.
- **`assertTrue(resultado de la condición)`:** Si tenemos un método que devuelve un booleano y su resultado esperado es `True`, pasará el test.
- **`assertFalse(resultado de la condición)`:** Si tenemos un método que devuelve un booleano y el resultado esperado es `False`, pasará el test.

# Hacer las pruebas

Haré la prueba con un el test recientemente creado, si te fijas,

```
1 import static org.junit.jupiter.api.Assertions.*;
2 import org.example.Mayor1;
3 import org.junit.jupiter.api.Test;
```

(Importante, al tener los archivos en carpetas distintas, has de importar la ruta que lleva hacía el rachivo y tal vez el método de @Test)



The screenshot shows an IDE with two tabs: 'pom.xml (maven\_JAA)' and 'Mayor1Test.java'. The 'Mayor1Test.java' tab is active, displaying the following code:

```
4
5 class Mayor1Test {
6     @Test
7     void mayor() {
8         int mayorUltimo = Mayor1.Mayor(a: 3, b: 4);
9         assertEquals( expected: 4, mayorUltimo);
10
11         int mayorPrimero = Mayor1.Mayor(a: 4, b: 3);
12         assertEquals( expected: 4, mayorPrimero);
13
14         int mayorIguales = Mayor1.Mayor(a: 4, b: 4);
15         assertEquals( expected: 4, mayorIguales);
16
17         int mayorNegativos = Mayor1.Mayor(a: -3, b: -4);
18         assertEquals( expected: -3, mayorNegativos);
19     }
20
21     @Test
22     void esMayor() {
23         assertTrue(Mayor1.EsMayor(a: 4, b: 3));
24         assertTrue(Mayor1.EsMayor(a: -3, b: -4));
25         assertFalse(Mayor1.EsMayor(a: 3, b: 4));
26         assertFalse(Mayor1.EsMayor(a: 4, b: 4));
27         assertFalse(Mayor1.EsMayor(a: -4, b: -3));
28     }
29 }
```

Below the code editor, a console window shows the test results:

```
ms Tests passed: 2 of 2 tests - 45 ms
ms C:\Users\josea\.jdk\openjdk-21.0.2\bin\java.exe ...
ms Process finished with exit code 0
```

Si te fijas, entre paréntesis con los están las variables del archivo anterior.

Al ejecutar el archivo con los métodos de JUnit de antes, da todo bien, porque todo está puesto bien, Pondré ejemplos en los que da error al cambiar cosas:

