Curso

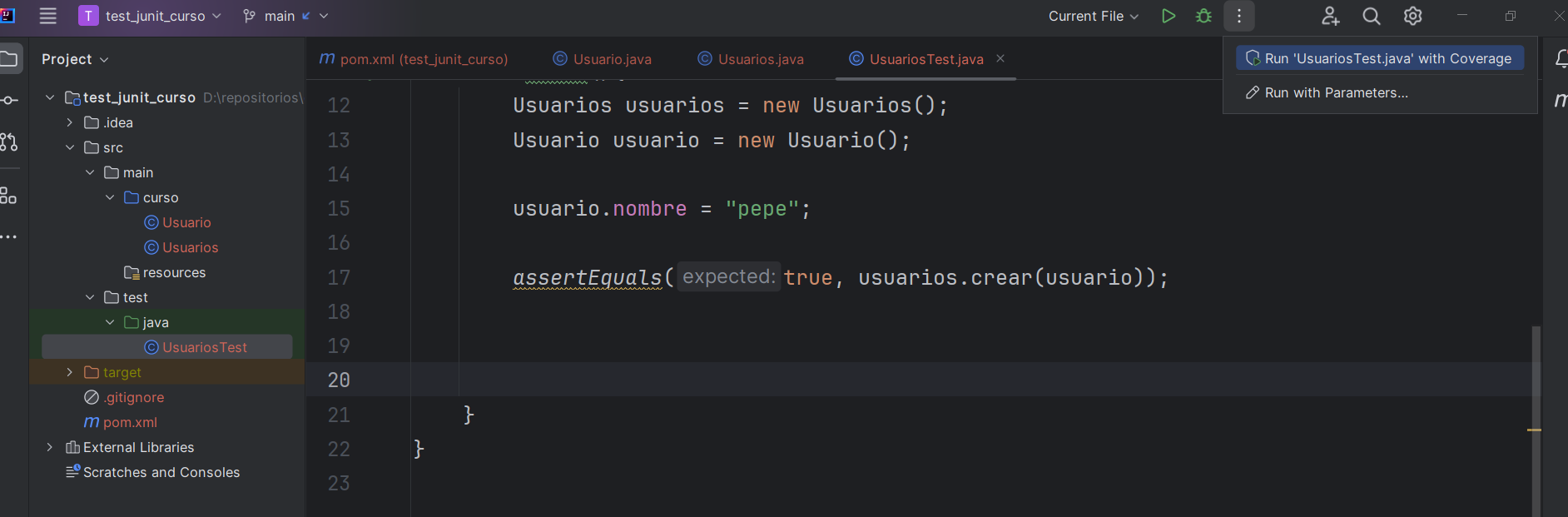
https://www.youtube.com/watch?v=4SD8q891ZQc&list=PLkVpKYNT\_U9fG0qEEXIvT9nJrMPZK6ORn&index=1

Github

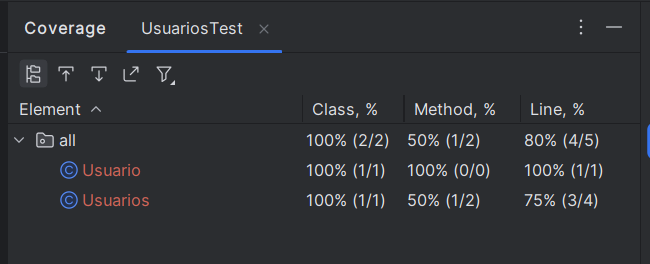
https://github.com/Open-Bootcamp/Testing-JUnit

MINUTO 🡪 video 4 (15”)

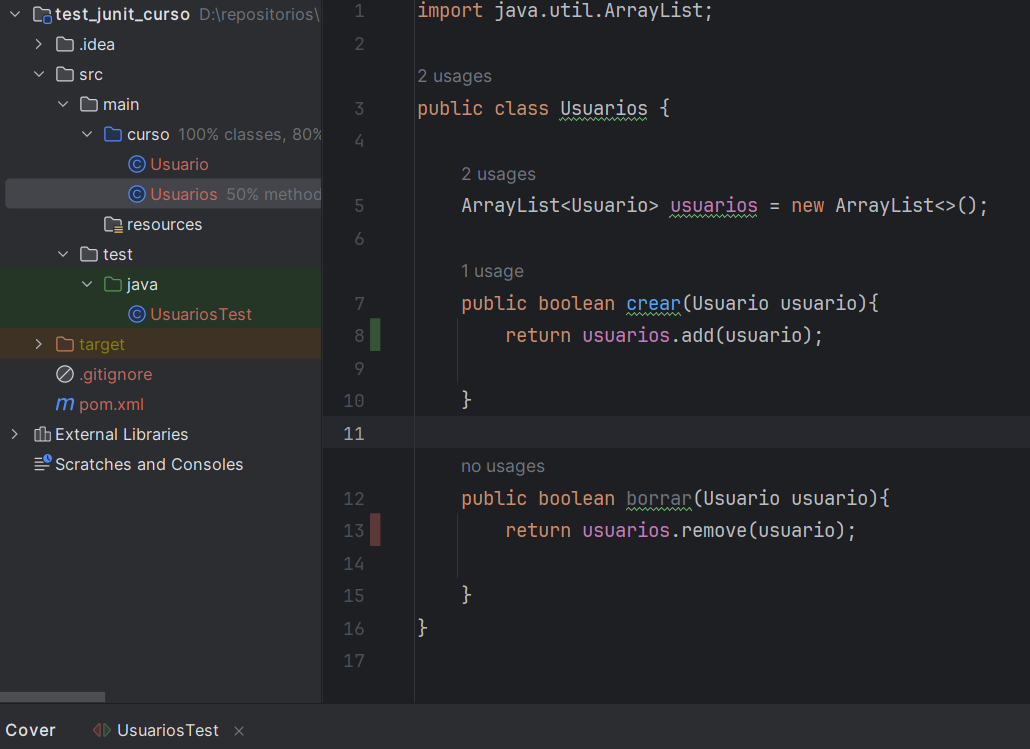
Para que este bien hecho los test tiene que haber leído mas del 95% del código para saber cuando a leído vamos a la carpeta test que nos ha creado al meter las dependencias de junit y no situamos en la clase UsuariosTest y arriba en los 3 puntos Run Usarios .... with coverage



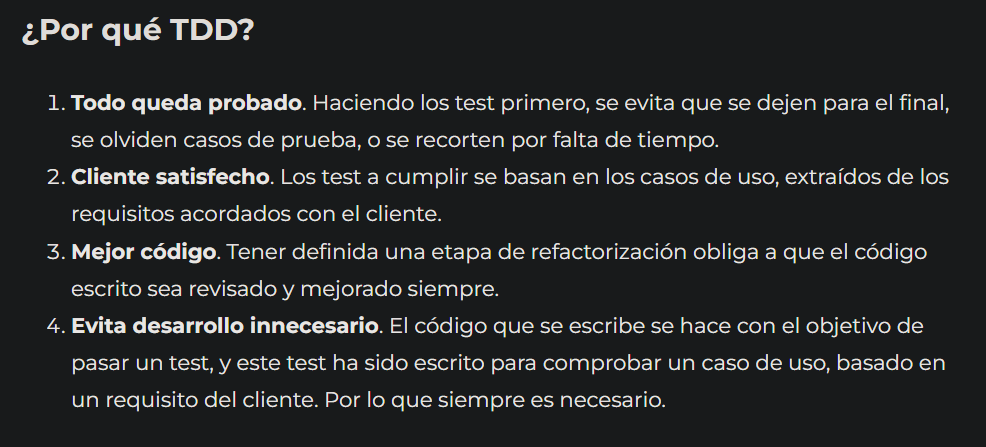
Y nos muestra el % de código que ha leído



Y si nos vamos a las clases nos indica con una barra verde que lo ha leído y una roja que no



Despues de la introduccion vamos a comenzar, hay una metodologia que se llama Test Driven Development (TDD) o “Desarrollo guiado por pruebas en la cual primero creamos los test y despues creamos la implementacion necesaria para que pasen los test



Fases

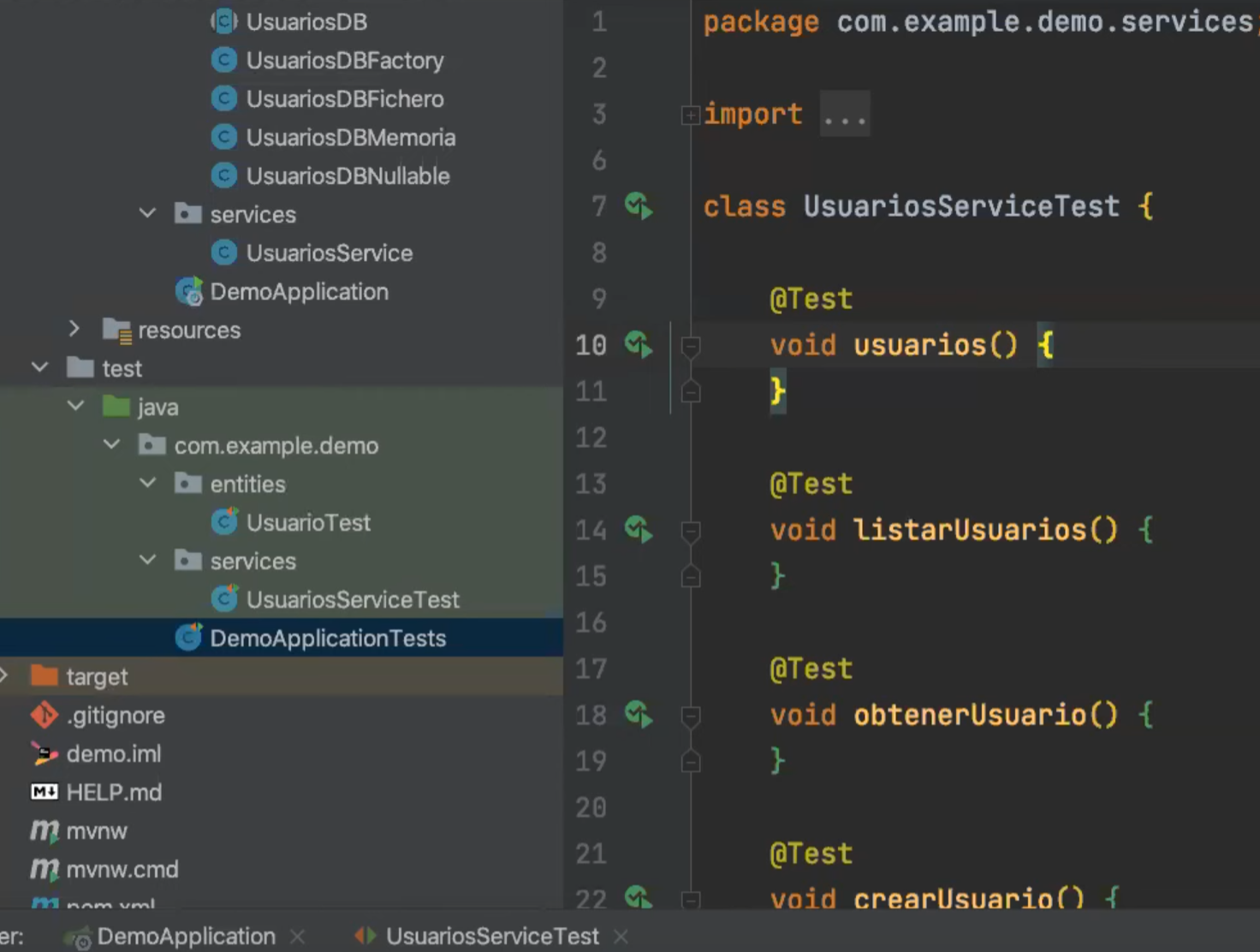


Las tres normas de TDD

Norma 1: No puede haber metodo o parte de código que no tenga test y hasta que no pase no podrá ir a producción, en resumen todo tiene que tener un test.

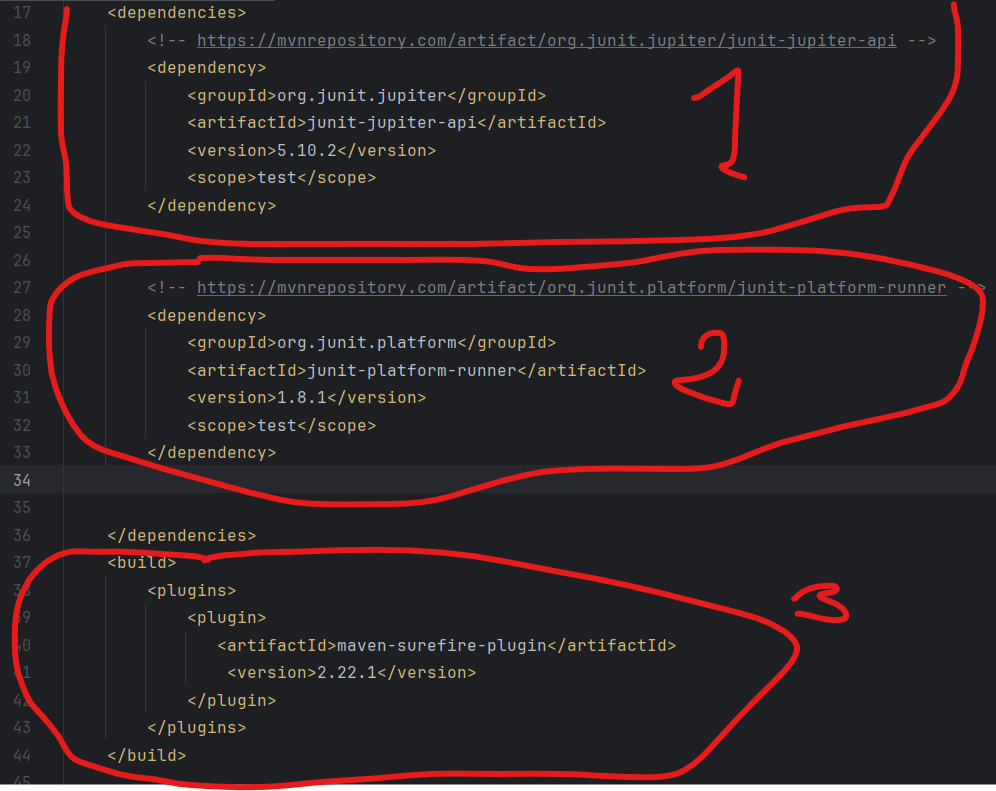
Norma 2: No escribir en la prueba unitaria mas código que el necesario para que ocurra el fallo, esto quier decir, que si estamos probando que un usuario es borrado de la bd no tienes que probar que antes se ha creado esto ya debería de haberse probado en otra prueba no las dos en la misma.

Norma 3: No escribir en la prueba unitaria mas código que el necesario para que pase la prueba, deberemos escribir el minimo codigo posible.

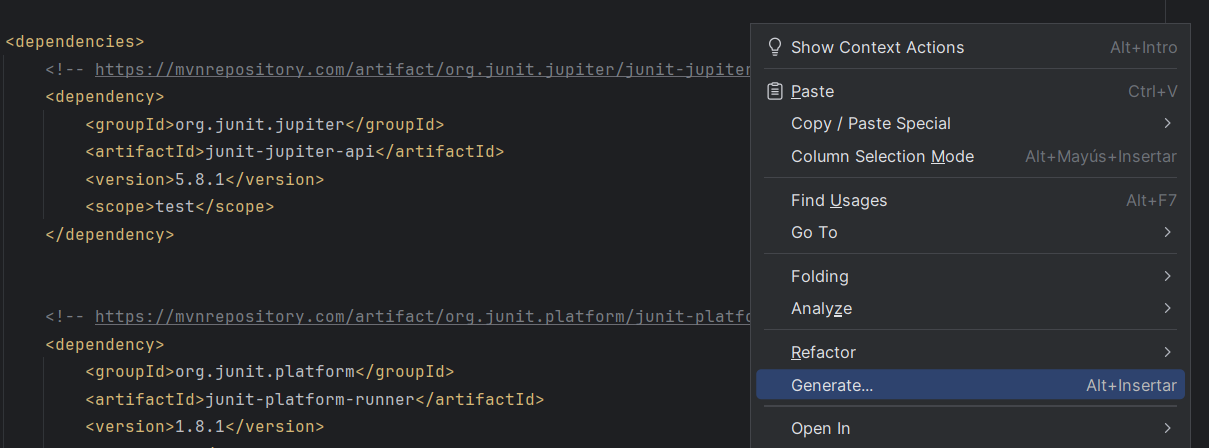


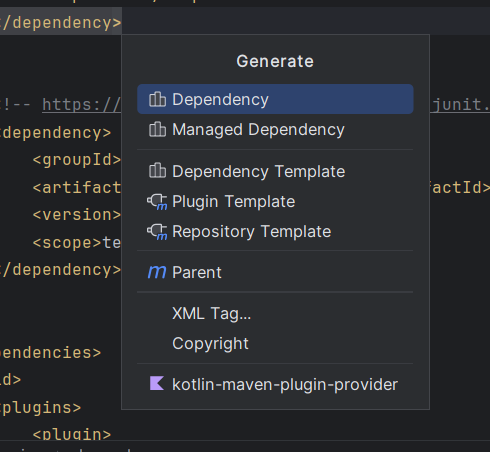
Configuración del POM en MAVEN

1. Dependencia del junit 5.x que es la actual ya no es 4
2. Dependencia para que lea versiones anteriores del junit
3. Dependencia para utilizar test de unit4 en entornos del 5

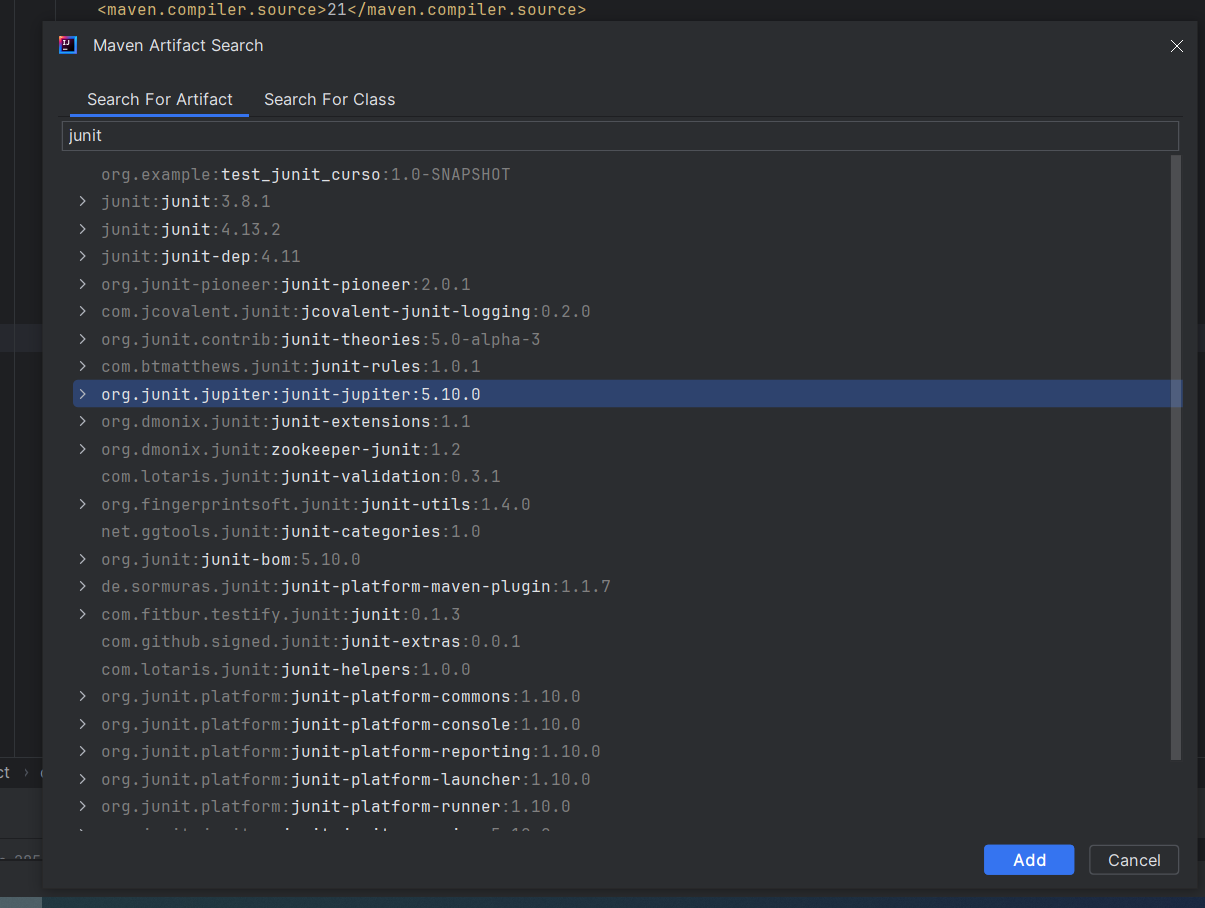


Tambien podemos incluir dependencias dentro del IDE para ello boton dch/generate/dependency y buscamos

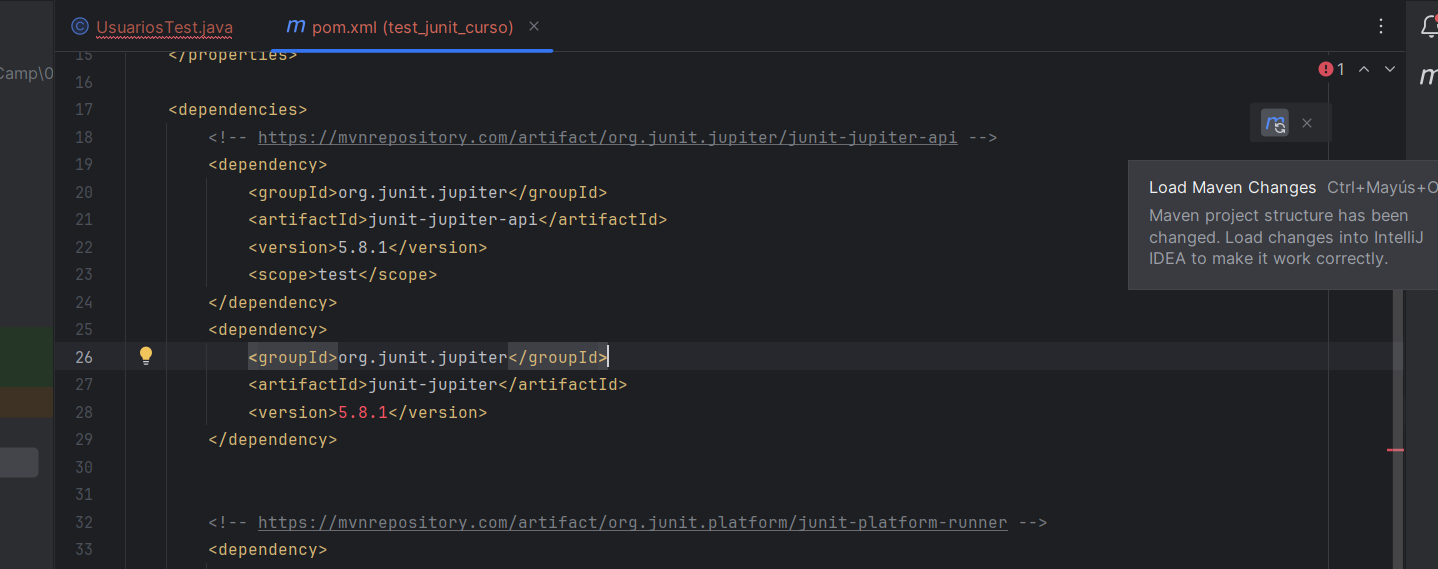




Seleccionamos y add



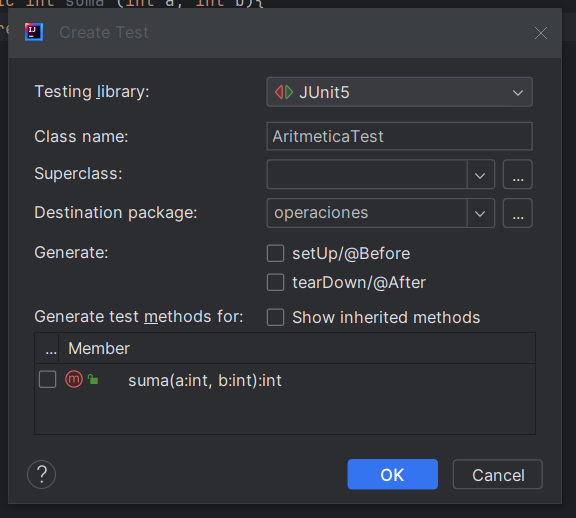
Y una vez con las dependencias debemos de recargar en el boton de la m (Load Maven Changes)

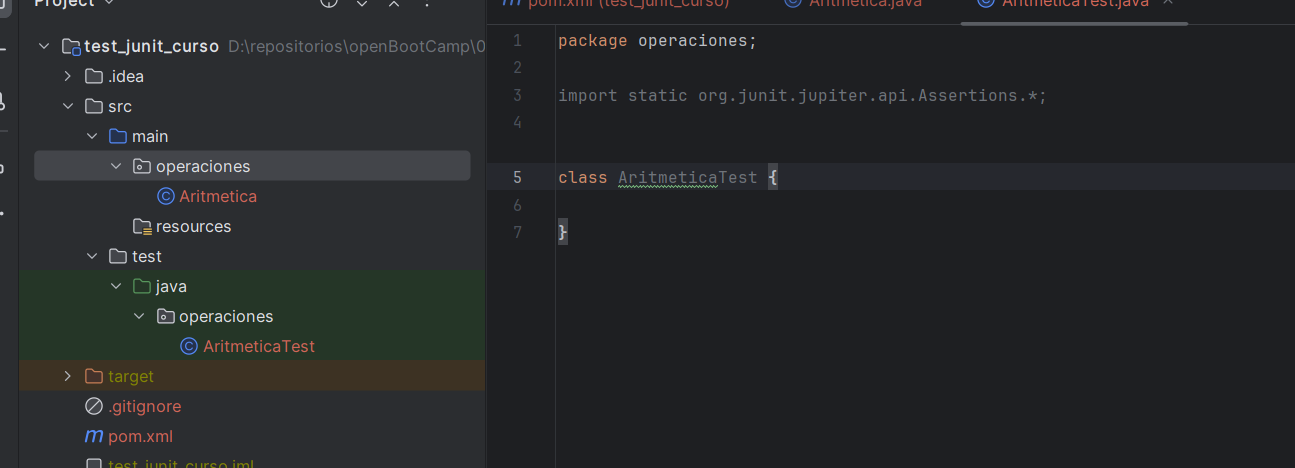


**Generar Test**

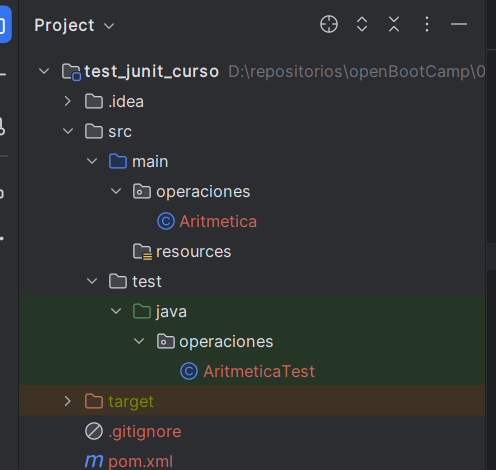
Para ello dentro del metodo boton dch/Genarete/Test/ok y nos generea dentro del paquete test una clase nueva







Se puede hacer a mano la carpeta de Test:Creamos un paquete igual que del main y dentro una clase que se llame igual a la clase+Test



Ahora comenzaremos en AritmeticaTest con la anotacion @Test para indicar que es un test.

@BeforeAll --> Se ejecuta 1 vez antes de todos los test

@BeforeEach --> Se ejecuta justo antes de cada test actual

@AfterAll --> Se ejecuta 1 vez despues de todos los test

@AfterEach --> Se ejecuta justo despues de cada test actual

@DisplayName --> Darle nombre (opcional), aunque el nombre del test deberia ser descriptivo por eso esta etiqueta no se suele usar