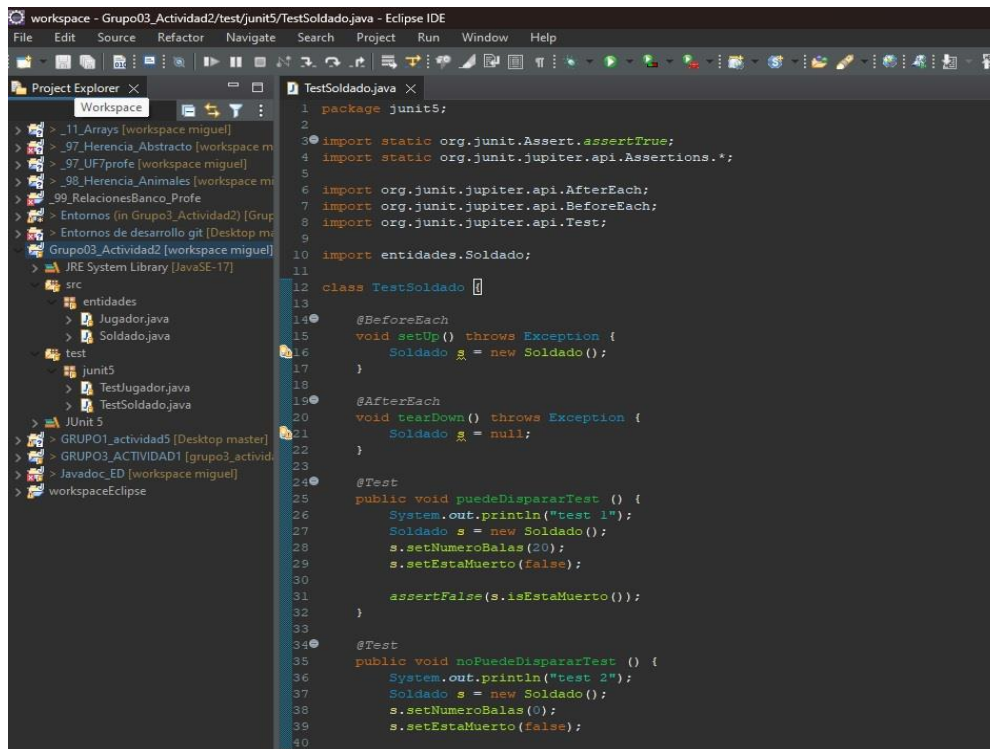


Actividad 2. Entornos de desarrollo. JavaDoc y JUnit.

Miguel García

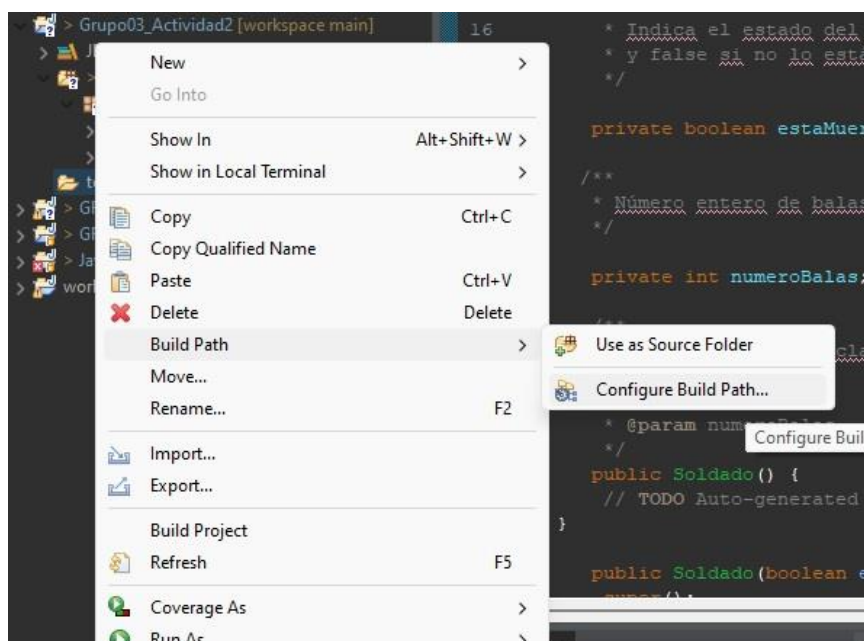
En primer lugar, he procedido a realizar el primer requerimiento de la actividad, en el cual se exponen unas líneas de código para crear dos clases: Jugador y Soldado, y crear los getter and setter en dichas clases. A continuación he realizado la documentación de java en cada clase con sus métodos y sus atributos. Todo esto esta empaquetado en un paquete dentro de la carpeta /src.

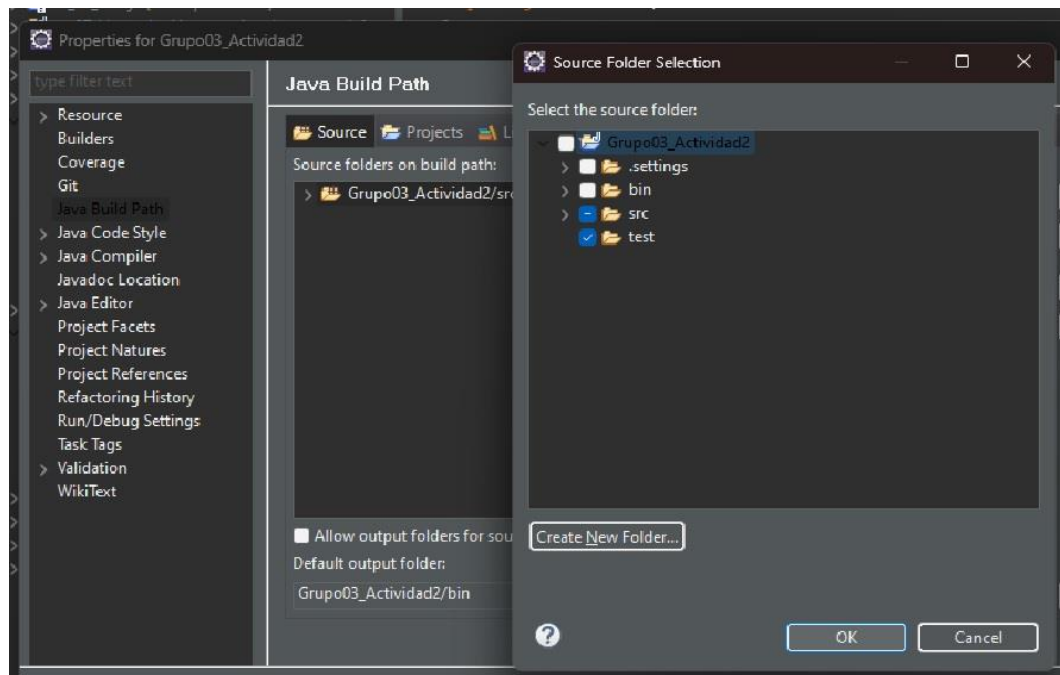


The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays a workspace named 'Grupo03_Actividad2'. Inside, there is a package 'entidades' containing 'Jugador.java' and 'Soldado.java', and a 'test' package containing 'TestJugador.java' and 'TestSoldado.java'. The main editor window shows the code for 'TestSoldado.java'. The code includes imports for JUnit 5, assertions, and the 'Soldado' class. It defines a 'TestSoldado' class with two test methods: 'puedeDispararTest' and 'noPuedeDispararTest'. The 'puedeDispararTest' method sets up a 'Soldado' object with 20 bullets and asserts that it is not dead. The 'noPuedeDispararTest' method sets up a 'Soldado' object with 0 bullets and asserts that it is not dead.

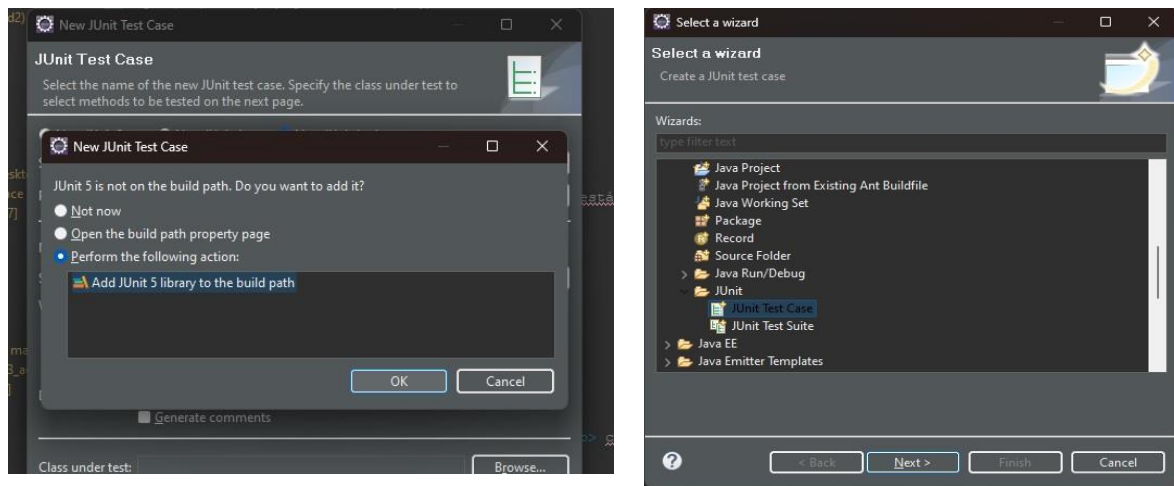
```
1 package junit5;
2
3 import static org.junit.Assert.assertTrue;
4 import static org.junit.jupiter.api.Assertions.*;
5
6 import org.junit.jupiter.api.AfterEach;
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.Test;
9
10 import entidades.Soldado;
11
12 class TestSoldado {
13
14     @BeforeEach
15     void setUp() throws Exception {
16         Soldado s = new Soldado();
17     }
18
19     @AfterEach
20     void tearDown() throws Exception {
21         Soldado s = null;
22     }
23
24     @Test
25     public void puedeDispararTest () {
26         System.out.println("test 1");
27         Soldado s = new Soldado();
28         s.setNumeroBalas(20);
29         s.setEstaMuerto(false);
30         assertFalse(s.isEstaMuerto());
31     }
32
33     @Test
34     public void noPuedeDispararTest () {
35         System.out.println("test 2");
36         Soldado s = new Soldado();
37         s.setNumeroBalas(0);
38         s.setEstaMuerto(false);
39     }
40 }
```

Tras crear la documentación de java de las dos clases, se crea otra carpeta dentro del proyecto java de nombre test, la cual se añade al build path.

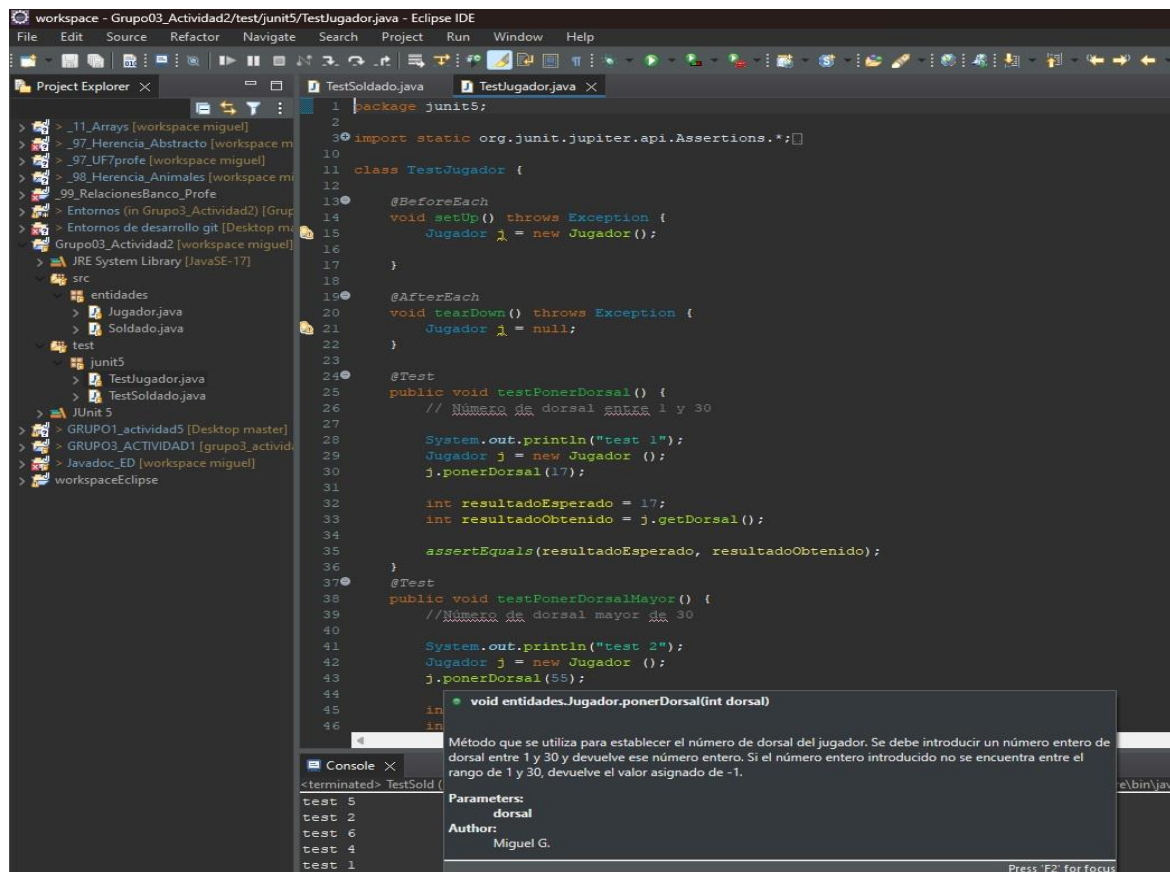




Ahora que tenemos la carpeta test añadida al build path podemos añadir un JUnit test Case para realizar las pruebas unitarias de cada clase y comprobar el correcto funcionamiento de ambas.



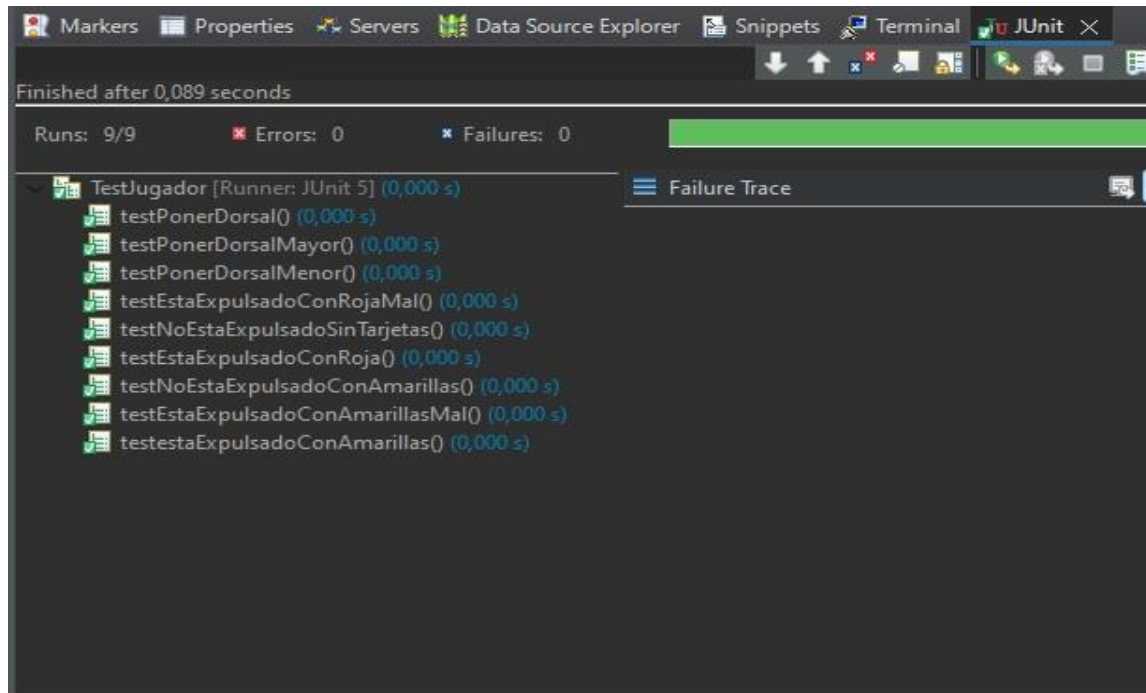
A continuación se realizan las pruebas unitarias de cada clase con JUnit 5.



En este punto, las pruebas unitarias que he realizado sobre la clase **Jugador** son las siguientes:

- ponerDorsal (): se elige un número de dorsal comprendido entre el 1 y el 30 y mediante una comprobación de resultados esperados y resultados obtenidos con j.getDorsal() pasa el test sin problemas puesto que el número introducido esta dentro del rango de resultado posible.
- ponerDorsalMayor (): En este caso se elige un número mayor de 30 para comprobar el correcto funcionamiento del método ponerDorsal (). Se introduce un número entero con valor 55 y mediante una comparación de resultado esperados y obtenidos, el resultado final del método tiene que ser -1.
- ponerDorsalMenor (): Este caso es igual que el anterior a diferencia que el número introducido es menor de 1, en este caso deberá devolver el mismo resultado de -1 obtenido en el anterior caso.
- estaExpulsadoConAmarillas (): Este test realiza la comprobación del método estaExpulsado() introduciendo un número de tarjetas amarillas = 2. El resultado obtenido es true mediante la aserción assertTrue (j.estaExpulsado()); En este caso, el método funciona correctamente.
- noEstaExpulsadoConAmarillas (): Este caso se comprueba que el jugador no esta expulsado pues el número de tarjetas amarillas no es igual a 2, pues se introduce el valor j.setNumeroTarjetasAmarillas(1);
- estaExpulsadoConAmarillasMal (): en este test se comprueba el funcionamiento del método estaExpulsado introduciendo un número de tarjetas amarillas superior a 2, con el que debe dar un resultado false.
- noEstaExpulsadoSinTarjetas (): Aquí se comprueba que el jugador no puede estar expulsado pues el número de tarjetas es = 0.

- estaExpulsadoConRoja (): En este caso, se introduce un numero de tarjetas rojas = 1, y haciendo uso del método estaExpulsado(), el resultado obtenido debe dar true, como se muestra en el test.
- estaExpulsadoConRojaMal (): este caso es igual que el anterior pero introduciendo un número de tarjetas rojas de 3, con lo cual el resultado se ve alterado y tiene que dar false.

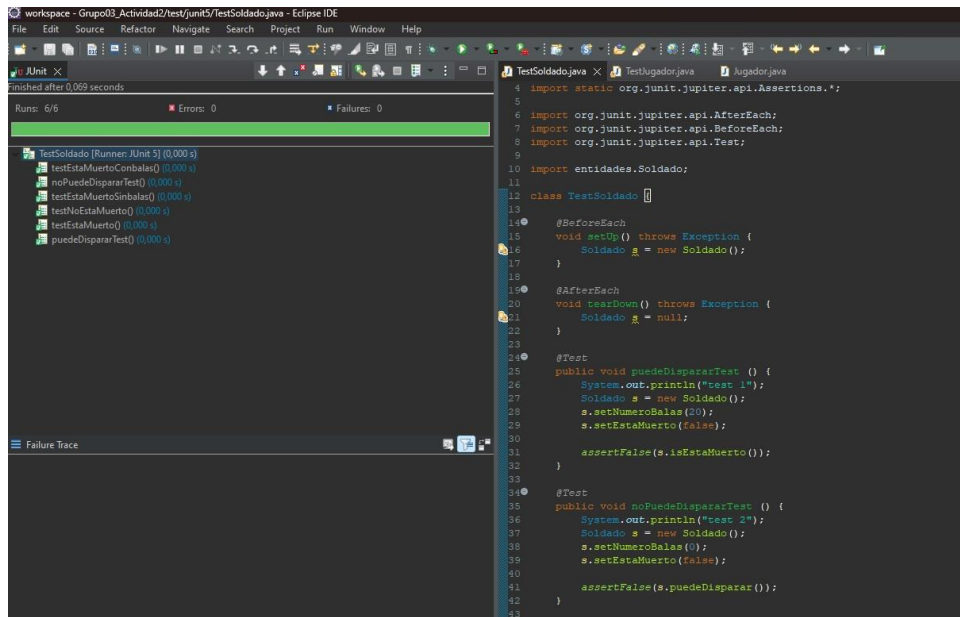


El resultado obtenido de las pruebas unitarias es positivo y todos los test funcionan correctamente.

En la clase **Soldado**, las pruebas unitarias serán las siguientes:

- puedeDisparar (): Se introduce un número positivo de balas y se prueba si el soldado puede disparar. El resultado esperado y el resultado obtenido es true.
- noPuedeDisparar (): Este caso es igual al anterior añadiéndole un 0 al numero de balas de las que dispone, con lo que sale un resultado false y el soldado no puede disparar.
- estaMuerto (): Se comprueba si el soldado está muerto o no.
- noEstaMuerto (): El método es similar al anterior pero en este caso para probar que el soldado no está muerto.
- estaMuertoConBalas (): Este método prueba si el soldado está muerto teniendo un número de balas mayor de 0. Se ejecuta el método disparar y se comprueba que el resultado obtenido es true, el soldado está muerto.
- estaMuertoSinbalas (): Similar al método anterior pero con un número de balas igual a 0. El resultado obtenido es true, el soldado está muerto sin balas.

Aquí se muestra el resultado obtenido de las pruebas unitarias de JUnit de la clase Soldado:



Tras la resolución del segundo requerimiento de la actividad, procedo a subir el proyecto java al repositorio creado por el compañero de grupo Nacho mediante la consola de comandos de Git.

