



AD-2 ACTIVIDAD ENTORNOS DE DESARROLLO

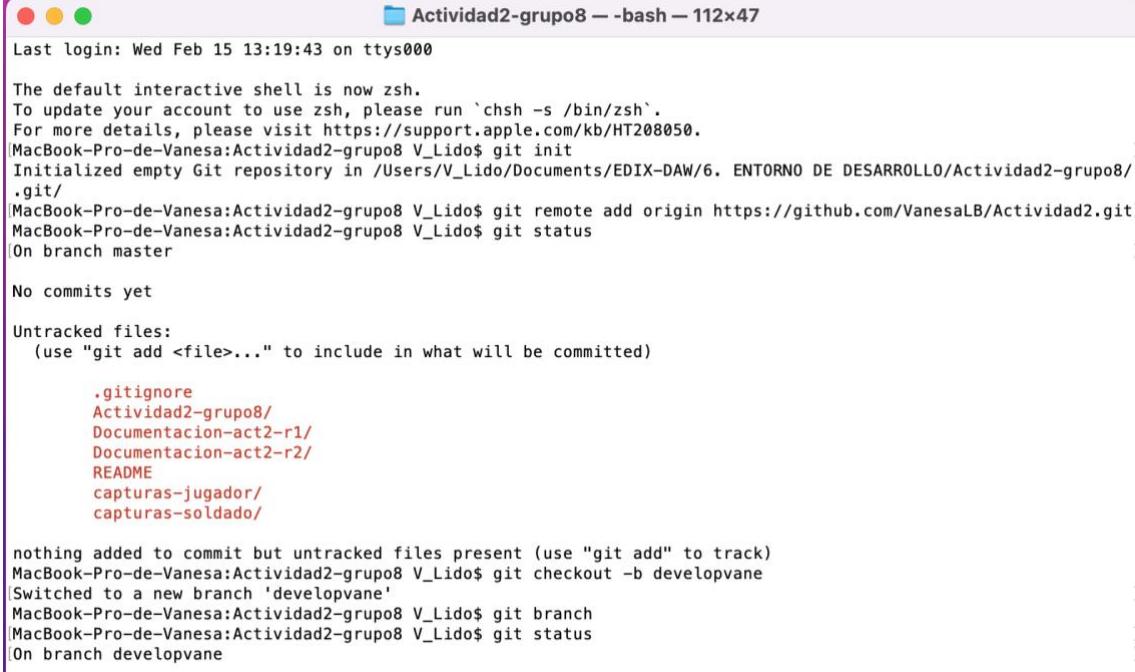


JavaDoc y JUnit

MARIA VANESA LIDÓ BOHIGAS
ALBERTO SÁNCHEZ GUTIÉRREZ
ESTEFANÍA BALLESTEROS TORRES

En esta actividad Vanesa creó el repositorio GitHub e hizo colaboradores a sus compañeros. Creó su rama developvane y subió la documentación JavaDoc realizada para el Requerimiento 1 y el Requerimiento2. Añadió también las capturas realizadas para el posterior documento pdf.

<https://github.com/VanesaLB/Actividad2>



```

Last login: Wed Feb 15 13:19:43 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git init
Initialized empty Git repository in /Users/V_Lido/Documents/EDIX-DAW/6. ENTORNO DE DESARROLLO/Actividad2-grupo8/.git/
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git remote add origin https://github.com/VanesaLB/Actividad2.git
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git status
On branch master

No commits yet

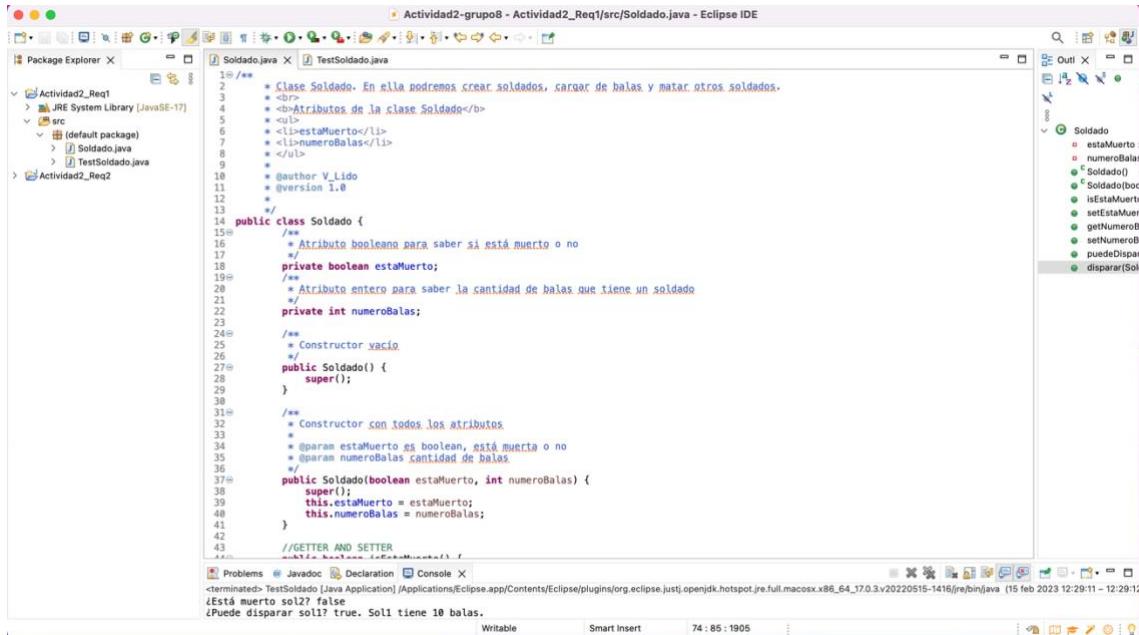
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  .gitignore
  Actividad2-grupo8/
  Documentacion-act2-r1/
  Documentacion-act2-r2/
  README
  capturas-jugador/
  capturas-soldado/

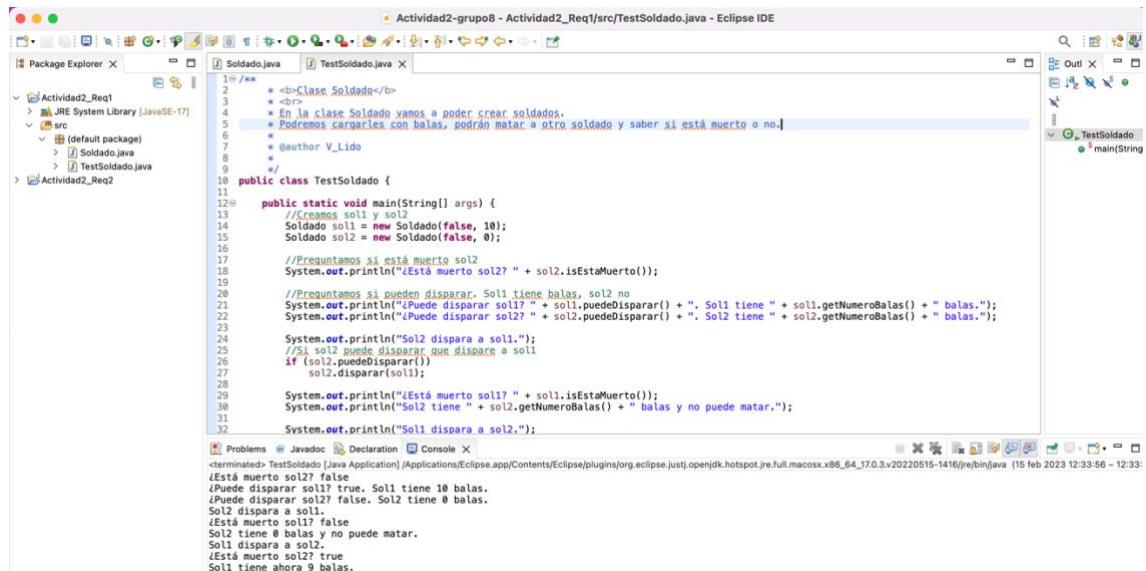
nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git checkout -b developvane
Switched to a new branch 'developvane'
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git branch
MacBook-Pro-de-Vanesa:Actividad2-grupo8 V_Lido$ git status
On branch developvane

```

Clase **Soldado** de Vanesa documentada para realizar JavaDoc



Clase **testSoldado** con test realizados como los hacemos en programación, para comprobar que funciona todo correctamente. Documentada para realizar **JavaDoc**.

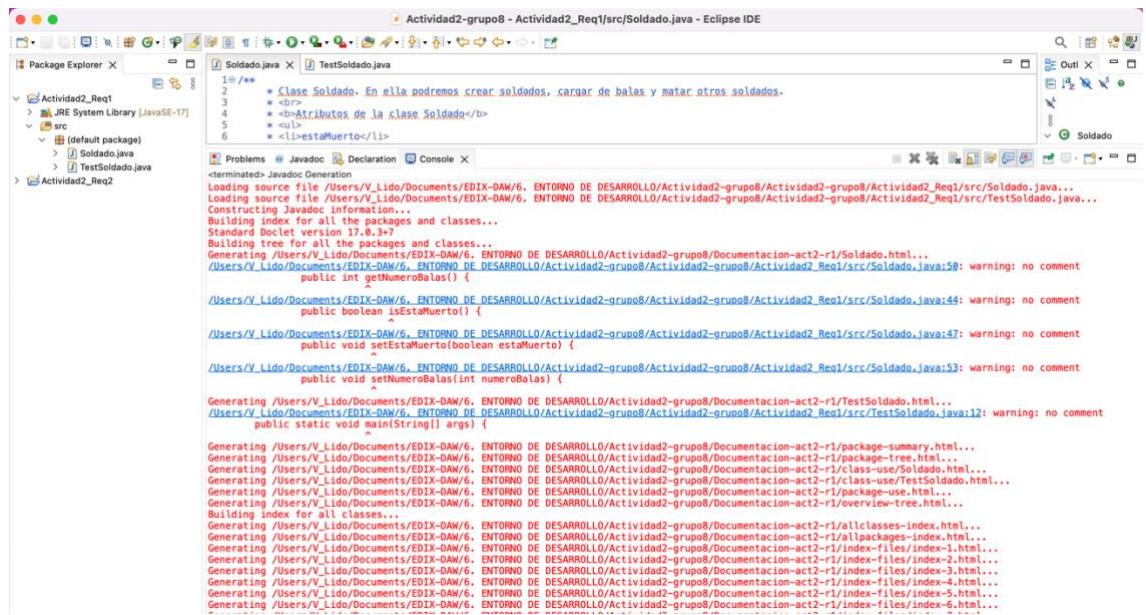


```

1 /**
2  * <br>
3  * <br>
4  * En la clase Soldado vamos a poder crear soldados.
5  * Podremos cargarles con balas, podrán matar a otro soldado y saber si está muerto o no.
6  *
7  * @author V_Lido
8  */
9
10 public class TestSoldado {
11
12     public static void main(String[] args) {
13         //Creamos sol1 y sol2
14         Soldado sol1 = new Soldado(false, 10);
15         Soldado sol2 = new Soldado(false, 0);
16
17         //Preguntamos si está muerto sol2
18         System.out.println("¿Está muerto sol2? " + sol2.isEstaMuerto());
19
20         //Preguntamos si pueden disparar. Sol1 tiene balas, sol2 no
21         System.out.println("¿Puede disparar sol1? " + sol1.puedeDisparar() + ". Sol1 tiene " + sol1.getNumeroBalas() + " balas.");
22         System.out.println("¿Puede disparar sol2? " + sol2.puedeDisparar() + ". Sol2 tiene " + sol2.getNumeroBalas() + " balas.");
23
24         System.out.println("Sol2 dispara a sol1.");
25         //Si sol2 puede disparar, querrá matar a sol1
26         if (sol2.puedeDisparar())
27             sol2.disparar(sol1);
28
29         System.out.println("¿Está muerto sol1? " + sol1.isEstaMuerto());
30         System.out.println("Sol1 tiene " + sol1.getNumeroBalas() + " balas y no puede matar.");
31
32         System.out.println("Sol1 dispara a sol2.");
33     }
34 }

```

JavaDoc de ambas clases. Los warning son de los *getter and setter* que no hace falta comentarlos. Posteriormente me he dado cuenta de que no he comentado el *main* y no tenía muy claro si también era necesario documentarla.



```

1 /**
2  * Clase Soldado. En ella podremos crear soldados, cargar de balas y matar otros soldados.
3  */
4
5  * <br>
6  * <br>
7  * Atributos de la clase Soldado</b>
8  */
9
10 public class Soldado {
11
12     /**
13      * <br>
14      * <br>
15      * <li>estaMuerto</li>
16     */
17
18     public int getNumeroBalas() {
19         return numeroBalas;
20     }
21
22     public void setEstaMuerto(boolean estaMuerto) {
23         estaMuerto = estaMuerto;
24     }
25
26     public void setNumeroBalas(int numeroBalas) {
27         numeroBalas = numeroBalas;
28     }
29
30     public String toString() {
31         return "Soldado{" +
32                 "estaMuerto=" + estaMuerto +
33                 ", numeroBalas=" + numeroBalas +
34                 '}';
35     }
36 }

```

JavaDoc de Vanesa de la clase **Soldado** y **TestSoldado** que podrás ver en el GitHub.

Class Soldado

```

java.lang.Object<@
Soldado

```

Clase Soldado. En ella podremos crear soldados, cargar de balas y matar otros soldados.
Atributos de la clase Soldado

- estaMuerto
- numeroBalas

Version:

1.0

Clase **Jugador** de Vanesa documentada para realizar **JavaDoc**

The screenshot shows the Eclipse IDE interface with the following details:

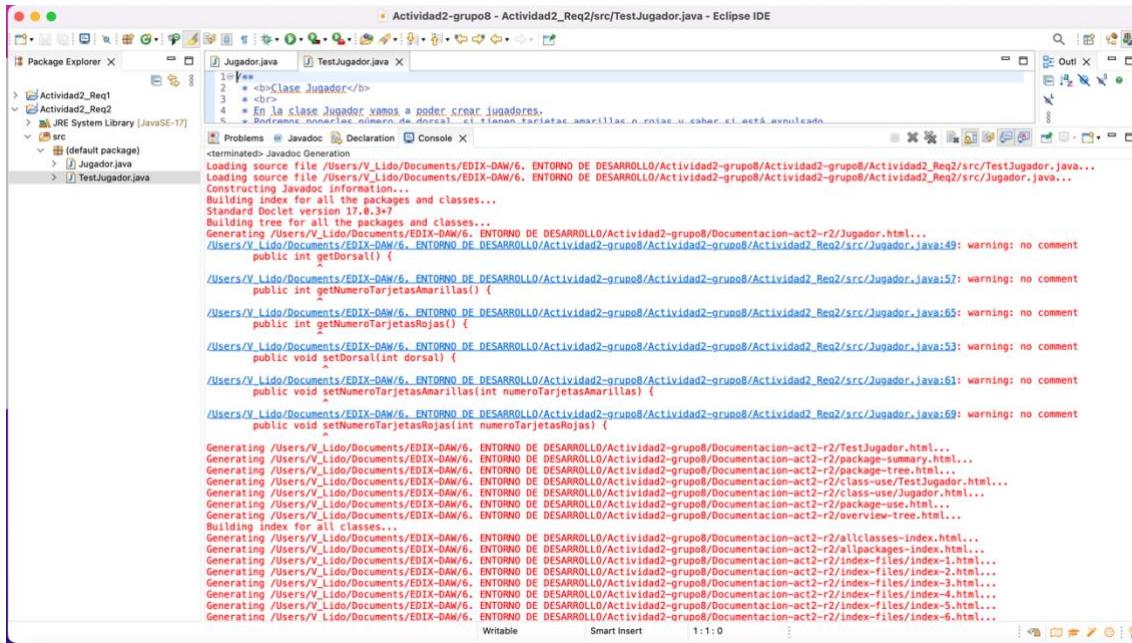
- Project Explorer:** Shows the project structure with packages `Actividad2_Req1`, `Actividad2_Req2`, and `JRE System Library [JavaSE-17]`. The `src` folder contains the `Jugador.java` and `TestJugador.java` files.
- Code Editor:** Displays the `Jugador.java` file. The code includes JavaDoc comments for the class and its constructor, as well as for the attributes `dorsal`, `numeroTarjetasAmarillas`, and `numeroTarjetasRojas`.
- Outline View:** On the right, it shows the generated JavaDoc for the `Jugador` class, listing the class itself and its three attributes.
- Console:** At the bottom, the console output shows the results of running the test code, indicating that players 1, 2, and 3 have dorsal numbers 10, 30, and 2 respectively.

Clase **testJugador** con test realizados como los hacemos en programación, para comprobar que funciona todo correctamente. Documentada para realizar **JavaDoc**.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages `Actividad2_Req1`, `Actividad2_Req2`, and `JRE System Library [JavaSE-17]`. The `src` folder contains the `Jugador.java` and `TestJugador.java` files.
- Code Editor:** Displays the `TestJugador.java` file. It contains a main method that creates three `Jugador` objects with dorsal numbers 10, 30, and 2 respectively. It then prints the dorsal numbers of these players.
- Outline View:** On the right, it shows the generated JavaDoc for the `TestJugador` class, which includes a single method `main`.
- Console:** At the bottom, the console output shows the expected output where players 1, 2, and 3 have dorsal numbers 10, 30, and 2 respectively.

JavaDoc de ambas clases. Los warning son de los *getter and setter* que no hace falta comentarlos. Al darme cuenta aquí sí he comentado el *main*.



JavaDoc de Vanesa de la clase **Jugador** y **TestJugador** que podrás ver en el GitHub.

Class Jugador

```
java.lang.Object
  Jugador
```

```
public class Jugador
  extends Object
```

Creamos una clase Jugador con la que podremos crear jugadores. Saber cantidad de tarjetas amarillas, tarjetas rojas y expulsarlos.
Atributos de la clase Jugador

- dorsal
- numeroTarjetasAmarillas
- numeroTarjetasRojas

Author:

V_Lido

Estefanía descarga la rama developvane y crea su rama developest añadiendo su parte del primer requerimiento con la documentación y las capturas realizadas. De esta manera podremos comparar nuestros trabajos.

CAPTURAS CÓDIGO Y JAVA DOC

ESTEFANIA BALLESTEROS TORRES

Capturas documentación

Test soldados código

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace_jre-eclipse-A02-JavaBox_BeatboxIt/sec/moduloJava/ejer/losSoldados.java - Help IDE
- Left Sidebar (Package Explorer):** Shows the project structure with packages like "01 excepciones", "02_estructuras_De_Control", "03_lectura_y_escritura_de_archivos", "04_Hacer_Mas_Opciones", "05_Sistema_Datos", "07_Arreglos", "08_Asignacion_if", "09_Relecion_Banco", "10_Ferendo_Animales", "11_Ferencia_Animales_Abstracto", "12_Colecciones", "13_intro_Orientacion_objetos", and "AD 2 JavaDoc JavaUnit". It also lists files such as "clase09JuegoAhorcado", "Cores_Ejercicios", "IIR_Actividad2", "EBT_Actividad4", "GRUPO14_actividad6", "main", and "Pruebe_fuciones".
- Central Area (Code Editor):** The code for "losSoldados.java" is displayed. The code initializes two Spanish soldiers and one English soldier, each with 10 and 15 bullets respectively. It then loops through the soldiers, printing their names and whether they can shoot. If they can, it asks if they want to shoot and handles the bullet count accordingly.
- Right Side (Console):** The output of the program is shown in the console window, detailing the initialization of soldiers and the loop where they are asked if they want to shoot and how many bullets remain.
- Bottom Bar:** Shows the Windows taskbar with various pinned icons and the system clock indicating 17:32 on 19/02/2023.

Test jugadores código

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure under the package `workspace_jar_A0-2_JavaDoc`. The `modelodeJuego` package contains files like `Soldado.java`, `Jugador.java`, `Tercugador.java`, and `TenSoldado.java`.
- Code Editor View:** Displays the `Tercugador.java` file. The code defines a class `Tercugador` with methods `cambiamos()` and `ponerDorsal()`. It uses `System.out.println` statements to output player information and their dorsal numbers.
- Console View:** Shows the output of the Java application. The console output includes:
 - Output from the `cambiamos()` method: "El jugador antonio tiene : 5" and "El jugador cesar tiene : 10".
 - Output from the `ponerDorsal()` method:
 - "El jugador antonio le damos el dorsal : 12"
 - "El jugador cesar le damos el dorsal : 10"
 - "El jugador andres ahora le damos el dorsal : -1"
 - "El jugador andres tiene : 2tarjetas amarillas y 1tarjetas rojas"
 - Final state of players:
 - "El jugador antonio tiene : 2tarjetas amarillas y 1tarjetas rojas"
 - "El jugador cesar tiene : 1tarjetas amarillas y 1tarjetas rojas"
 - "El jugador andres tiene : 2tarjetas amarillas y 1tarjetas rojas"
 - "El jugador antonio esta expulsado : true"
 - "El jugador jesus tiene : 1tarjetas amarillas y 1tarjetas rojas"
 - "El jugador jesus esta expulsado : true"

CAPTURAS CÓDIGO Y JAVA DOC

ESTEFANIA BALLESTEROS TORRES

JavaDoc soldado(4 warmings de los Gueter and setter)

JavaDoc test soldado

CAPTURAS CÓDIGO Y JAVA DOC

ESTEFANIA BALLESTEROS TORRES

JavaDoc jugador (6 warming Guetter and Setter)

JavaDoc test jugador

workspace_jet - AD-2_Jugadores_basico\src\main\java\beans\TestJugadores.java - Eclipse IDE

File Edit Source Refactor Navigator Search Project Run Window Help

Debug Explorer

① 01 excepciones

② 01_primeros_pasos.java

③ 02_instrucciones_De_Control

④ 03_métodos_main

⑤ 04_Hacer_Menú_Opciones

⑥ 05_Costa_o_String

⑦ 07_imprimir_consola

⑧ 08_ciclos_for

⑨ 14_Relaciones_Binarias

⑩ 15_Herencia_Animals

⑪ 16_Herencia_Animals_Abstracto

⑫ 17_Clases

⑬ 18_Estructura_Orientacion_objetos

⑭ Actividades5

⑮ AD_2_JavaDoc_JavaNet

⑯ IR_System_Library [JavaSE-17]

⑰ modelo_javabeans

⑱ I_Jugador.java

⑲ Soldado.java

⑳ I_losJugadores.java

⑳ I_losSoldados.java

⑳ I_dondeEstanLosJugadores.java

⑳ Cores_Futuros

⑳ I_10_Articulos2

⑳ EBT_Articulos4

⑳ GRUPO14_actividad6

⑳ main

⑳ Pruebas_funcionales

Soldado.java | Jugador.java | losJugadores.java | losSoldados.java | lanche.modelo.javabeans;

1 package modelo.javabeans;

2 *

3 * Clase test Jugador donde se crean nuevos jugadores y se les da o cambia un numero de dorsal así como las tarjetasAmarillas y las TarjetasRojas

4 *

5 *

6 * También se puede ver su numero de dorsal y cuantas tarjetas de ambos colores tienen

7 *

8 * Author fanyb

9 *

10 */

11 public class TestJugadores {

12 *///

13 * @metodomain para ejecutar los métodos

14 *

15 * @param args de metodo main

16 */

17

18 public static void main(String[] args) {

19 *///

20 * creamos los jugadores y les damos su numero de dorsal y de tarjetas amarillas y rojas

21 */

22 Jugador antonio=new Jugador(12, 2, 1);

23 Jugador david=new Jugador(5, 0, 1);

24 Jugador cesar=new Jugador(10, 0, 0);

25 Jugador andres=new Jugador(8, 2, 0);

26 Jugador jesus=new Jugador(4, 2, 0);

27 */

28 * le cambiamos el dorsal a andres

29 */

30 andres.ponerDorsal(0);

31 */

32 * vemos que dorsal tiene cada uno

33 */

34 System.out.println("el jugador David le damos el dorsal : "+david.getDorsal());

35 System.out.println("el jugador Antonio le damos el dorsal : "+antonio.getDorsal());

36 System.out.println("el jugador Cesar le damos el dorsal : "+cesar.getDorsal());

37

⑰ Declaration ⑰ Search X

No search results available. Start a search from the [search dialog](#).

⑰ Console x

terminated: javac: Generation

Loading source file: C:\Users\fanyb\OneDrive\Escritorio\AD-2_Jugadores_basico\src\main\java\beans\TestJugadores.java

Constructing Javadoc information...

Building index for all the packages and classes...

Standard docket version 17.0.3+7

Building tree for all the packages and classes...

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_Jugador.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_losJugadores.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_losSoldados.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_dondeEstanLosJugadores.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\Cores_Futuros.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_10_Articulos2.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\EBT_Articulos4.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\GRUPO14_actividad6.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\main.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\Pruebas_funcionales.java

Building index for all classes...

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_Jugador.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_losJugadores.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_losSoldados.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_dondeEstanLosJugadores.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\Cores_Futuros.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\I_10_Articulos2.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\EBT_Articulos4.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\GRUPO14_actividad6.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\main.java

Generating C:\Users\fanyb\OneDrive\Escritorio\AD-2_ESTRUCTURA\src\main\java\beans\Pruebas_funcionales.java

CAPTURAS DE DOCUMENTACION GENERADA

ESTEFANIA BALLESTEROS TORRES

The screenshot shows the JavaDoc interface for the `Jugador` class. The title bar says "Jugador". The menu bar includes "PACKAGE CLASS USE TREE INDEX HELP". The toolbar has "SEARCH: C:\\". The package is `modelo.javabeans`. The class is `Jugador`, which extends `Object`. It has two constructors: `Jugador()` (empty constructor) and `Jugador(int dorsal, int numeroTarjetasAmarillas, int numeroTarjetasRojas)` (constructor with parameters). The `estaExpulsado()` method returns a boolean indicating if the player is sent off. The `getDorsal()` and `getNumeroTarjetasAmarillas()` methods return the dorsal number and yellow card count respectively. The `getNumeroTarjetasRojas()` and `ponerDorsal(int dorsal)` methods return the red card count and set the dorsal number. The JavaDoc is generated from a file at `C:/Users/fanyb/OneDrive/Escritorio/AD-2%20ESTEFANIA/Actividad2/Actividad%20Estefania/requerimiento2javadoc/modelo/javabeans/Jugador.html`.

The screenshot shows the JavaDoc interface for the `TestJugados` class. The title bar says "TestJugados". The menu bar includes "PACKAGE CLASS USE TREE INDEX HELP". The toolbar has "SEARCH: C:\\". The package is `modelo.javabeans`. The class is `TestJugados`, which extends `Object`. It has one constructor: `TestJugados()`. The `main(String[] args)` method is a static method that executes the tests. The JavaDoc is generated from a file at `C:/Users/fanyb/OneDrive/Escritorio/AD-2%20ESTEFANIA/Actividad2/Actividad%20Estefania/requerimiento2javadoc/modelo/javabeans/TestJugados.html`.

CAPTURAS DE DOCUMENTACION GENERADA

ESTEFANIA BALLESTEROS TORRES

The screenshot shows the JavaDoc interface for the `Soldado` class. The title bar says "Soldado". The menu bar includes "PACKAGE", "CLASS", "USE", "TREE", "INDEX", "HELP". The search bar says "SEARCH: []". The package is `modelo.javabeans`. The class is `Soldado`. The class extends `java.lang.Object`. It has two constructors: `Soldado()` and `Soldado(boolean estaMuerto, int numeroBalas)`. The `estaMuerto` attribute is private and represents if the soldier is dead. The `numeroBalas` attribute is private and represents the number of bullets. The `disparar` method allows soldiers to shoot other soldiers and lose bullets. The `getNumeroBalas`, `isEstaMuerto`, `puedeDisparar`, `setEstaMuerto`, and `setNumeroBalas` methods are also listed.

The screenshot shows the JavaDoc interface for the `TestSoldado` class. The title bar says "TestSoldado". The menu bar includes "PACKAGE", "CLASS", "USE", "TREE", "INDEX", "HELP". The search bar says "SEARCH: []". The package is `modelo.javabeans`. The class is `TestSoldado`. The class extends `java.lang.Object`. It has one constructor: `TestSoldado()`. The `main` method creates soldiers, gives them bullets, and checks if they are dead or if they can shoot. It also lists the inherited methods from `java.lang.Object`.

Alberto crea su rama pero tiene problemas, lo que ha subido está en una carpeta azul con una flecha blanca, no lo podemos ver y no sabemos cómo solucionarlo. No hemos realizado captura de pantalla y al intentar borrar la rama ya no aparece pero la rama continúa ahí.

Mientras tanto Estefanía realiza su parte de JUnit y actualiza su rama con capturas sobre esto pero el archivo java lo ha puesto en un zip y no aparece en el GitHub.

Capturas de test pasados de Estefanía

Pasados test correctos de puedeDisparar y puedeDispararFalse.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace.java - AD-2_josephc_leonardo/test/junit/TestSoldados.java - Eclipse IDE
- File Menu:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** New Open Save Run Stop Run Window Help
- Package Explorer:** ✓ Albit x
- Console:** <terminated> testSoldados [JUnit] C:\Users\lany\OneDrive\Escritorio\PROGRAMACION\JAVA\Albit\src\main\java\com\javabeans\Soldado.java
- Left Panel:** Test Soldados (Runner) Run 5 [0.0%]
- Code Editor:** The code editor displays the following Java code with JUnit annotations:

```
package junit5;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.Assert.assertTrue;
import org.junit.jupiter.api.Test;
import modelo.javabeans.Soldado;

class TestSoldados {
    private boolean estaMuerto = false;
    public int numeroBalas;
    Soldado solespanol2=new Soldado(false, 10);
    Soldado soldingol1=new Soldado(false, 15);
    Soldado soldingol2=new Soldado(false, 6);

    @Test
    public void puedoDisparar() {
        int numeroBalas=9;
        //Soldado solespanol1=new Soldado();
        Solespanol solespanol2=new Soldado(false, 10);
        if (solespanol2.numeroBalas>0) {
            solespanol2.puedoDisparar();
        }
        assertTrue(solespanol2.puedoDisparar());
    }
}
```

- Bottom Status Bar:** Writable Smart Insert 29:43:702 19/02/2023
- System Tray:** 13°C Moymn: nublado

```
workspace.jie - AD-2_javabe_lawunit/test/unit/TestSoldados.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Package Explorer View Tools Help
Test Soldados [Unit] TestSoldados.java [TestSoldados.java]
private boolean estabulto=false;
public int numerosBalas;

private void soldado1=new Soldado(false, 10);
private void soldango1=new Soldado(false, 15);
private void soldinges2=new Soldado(false, 0);

@Text
public void puedesDisparar() {
    Soldado solespanol2=new Soldado(false, 10);
    if (solespanol2.numerosBalas>0) {
        solespanol2.puedesDisparar();
    }
    assertTrue(solespanol2.puedesDisparar());
}

@Text
public void puedesDispararFalse() {
    Soldado solespanol2=new Soldado(false, 0);
    if (solespanol2.numerosBalas>0) {
        solespanol2.puedesDisparar();
    }
    assertFalse(solespanol2.puedesDisparar());
}

@Text
public void disparar(Soldado soldado) {
    Soldado solespanol2=new Soldado(false, 15);
    Soldado soleinges1=new Soldado(false, 10);
}

Failure Trace
org.junit.jupiter.api.extension.ParameterResolver
at java.base/java.util.ArrayList.forEach(ArrayList.java:125)
at java.base/java.util.ArrayList.forEach(ArrayList.java:147)

No search results available. Start a search from the search dialog.
```

Test de disparar incorrecto , no pasa.

The screenshot shows an IDE interface with several windows open:

- Project Explorer**: Shows packages like `src.main.java`, `src.test.java`, and `src.test.junit`. A file named `SoldadoTest.java` is selected.
- Code Editor**: Displays the `SoldadoTest.java` file with JUnit test cases for the `Soldado` class. The code includes assertions for methods like `puedeDisparar` and `disparar`.
- Console**: Shows the output of the test run, indicating 3 passed tests and 0 failures.
- Failure Trace**: Shows the stack trace for a failed test, pointing to the `disparar` method in `SoldadoTest.java`.
- Search**: A search bar at the bottom of the IDE.

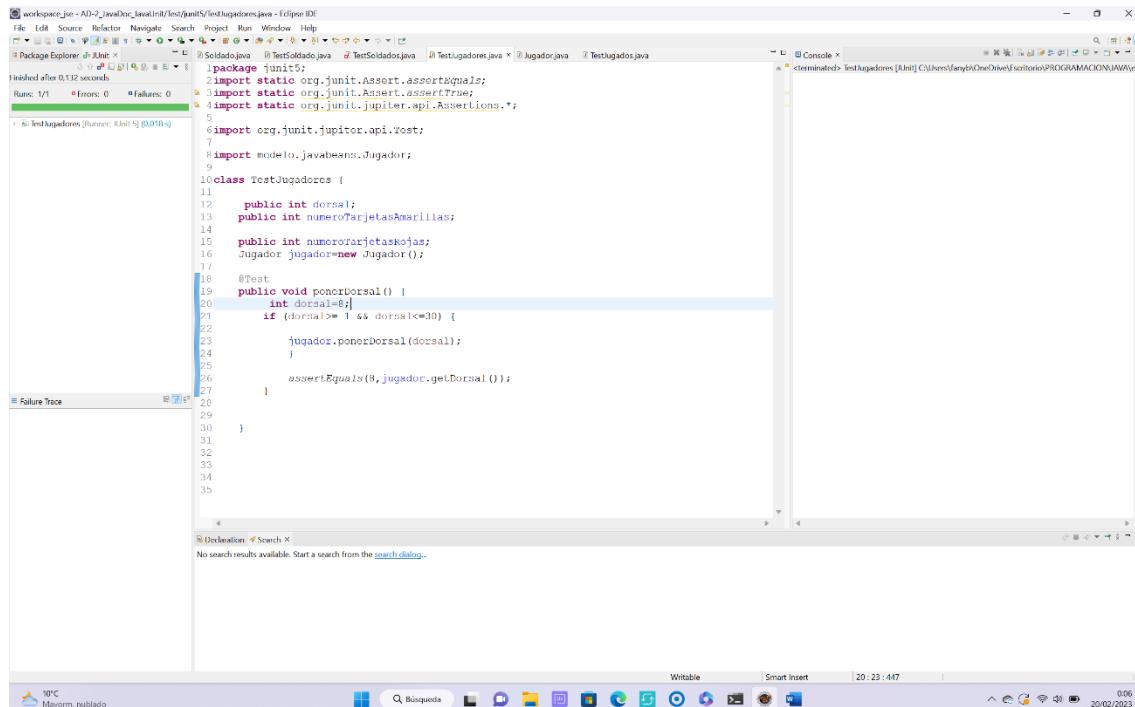
The status bar at the bottom shows system information: 13°C, Mayorn, nublado, 23:15, 19/02/2023, and a battery icon.

En este punto no ha sabido solucionarlo, puesto que no ha sido consciente de que podía hacer cambios en el método.

Ha continuado con el test para la clase Jugador.

Pruebas Unitarias Test jugadores

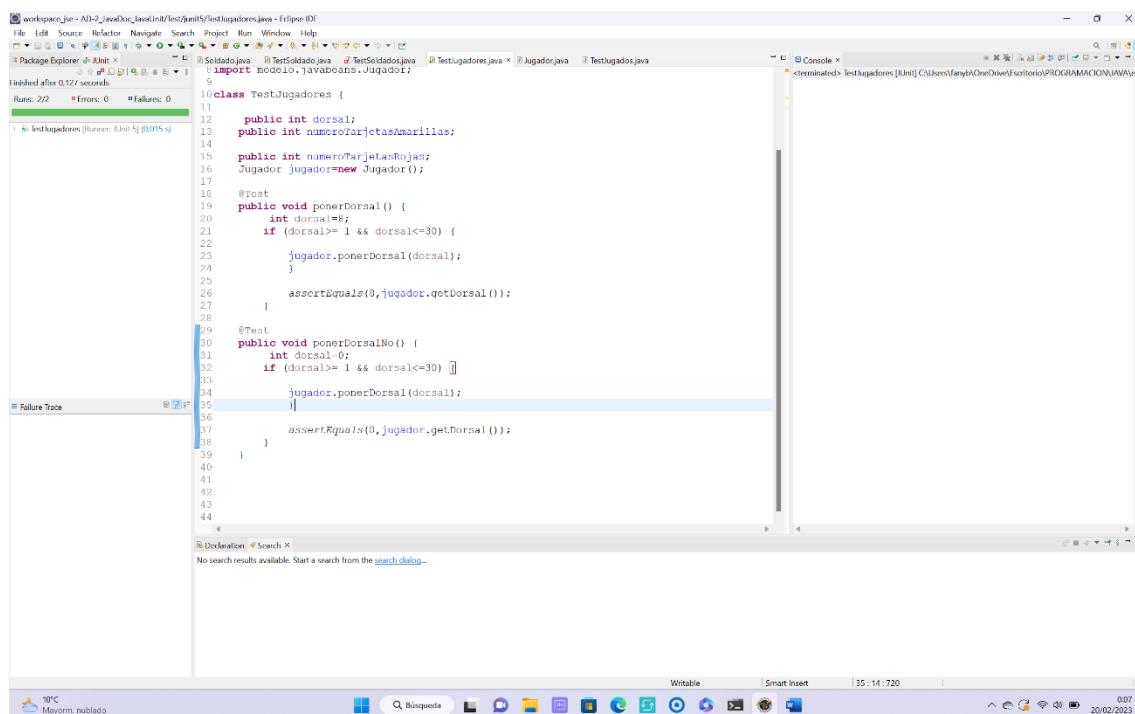
Test ponerDorsal (cumpliendo la condición)



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Soldado.java, TestSolidado.java, TestJugadores.java, Jugador.java, and TestJugadores.java.
- Console:** Displays the message <terminated> TestJugadores [Unit] C:\Users\lanyb\OneDrive\Escritorio\PROGRAMACION\JAVA\Adq\workspace_jar - AD-2_JavaDoc_levelUnit\test\junit\testJugadores.java - Eclipse IDE.
- Failure Trace:** Shows the code for the 'ponerDorsal' method, which includes an if-statement checking if 'dorsal' is between 1 and 30. It also includes an assertEquals call comparing the result with 8.
- Bottom Status Bar:** Shows the date and time as 20/02/2023 and 20:23:447.

Test ponerDorsal (no cumpliendo la condición)



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Soldado.java, TestSolidado.java, TestJugadores.java, Jugador.java, and TestJugadores.java.
- Console:** Displays the message <terminated> TestJugadores [Unit] C:\Users\lanyb\OneDrive\Escritorio\PROGRAMACION\JAVA\Adq\workspace_jar - AD-2_JavaDoc_levelUnit\test\junit\testJugadores.java - Eclipse IDE.
- Failure Trace:** Shows the code for the 'ponerDorsalNo' method, which includes an if-statement checking if 'dorsal' is between 1 and 30. It also includes an assertEquals call comparing the result with 0.
- Bottom Status Bar:** Shows the date and time as 20/02/2023 and 35:14:720.

Test estaExpulsado (con dos tarjetas amarillas)

The screenshot shows the Eclipse IDE interface with several open windows:

- Package Explorer**: Shows files like `Soldado.java`, `TestSoldado.java`, `TestSoldados.java`, `TestJugador.java`, and `TestJugados.java`.
- Console**: Displays the output of a terminated test run: `testJugadores.estabExpulsado2 [INFO] C:\Users\Anyh\OneDrive\Escritorio\PROGRAMAS`.
- Java Editor**: The main editor window contains the `Soldado.java` file with code for a soldier class and its tests.
- Failure Trace**: A small window showing the stack trace for a failed test case.

Soldado.java code (partial):

```
1 package javier_juanito.test;
2
3 public class Soldado {
4     ...
5     ...
6 }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31     int dorsal;
32
33     if (dorsal > 1 && dorsal < 30) {
34         jugad0r.ponerDorsal(dorsal);
35     }
36
37     assertEquals(0, jugad0r.getDorsal());
38 }
39
40
41     @Test
42     public void estabExpulsado() {
43         boolean expulsado=false;
44         if(numeroTarjetasAmarillas==2 || numeroTarjetasRojas==1) {
45             jugad0r.estabExpulsado();
46         }
47         jugad0r2.setNumeroTarjetasAmarillas(2);
48         jugad0r2.setNumeroTarjetasRojas(0);
49
50         assertTrue(jugad0r2.estabExpulsado());
51     }
52
53
54     @Test
55     public void estabExpulsado2() {
56         boolean expulsado=false;
57         if(numeroTarjetasAmarillas==2 || numeroTarjetasRojas==1) {
58             jugad0r.estabExpulsado();
59         }
60         jugad0r2.setNumeroTarjetasAmarillas(0);
61         jugad0r2.setNumeroTarjetasAmarillas(0);
62         jugad0r2.setNumeroTarjetasRojas(1);
63         jugad0r2.setNumeroTarjetasRojas(0);
64
65         assertTrue(jugad0r2.estabExpulsado2());
66     }
67 }
```

A tooltip is visible over the line `jugad0r2.setNumeroTarjetasAmarillas(0);` in the `estabExpulsado2()` method, showing the method signature: `void modelo.jugbeans.Jugador.setNumeroTarjetasAmarillas(int numeroTarjetasAmarillas)`.

Test estaExpulsado2 (con una tarjeta roja)

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace_jeo - Ab-2_juego_de_las_tarjetas.java [Unit] /src/test/java/testJugadores.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard toolbar with icons for New, Open, Save, Cut, Copy, Paste, Find, etc.
- Package Explorer:** Shows the project structure with files like Soldado.java, TestSoldado.java, TestSoldados.java, *TestJugadores.java, Jugador.java, and TestJugados.java.
- Code Editor:** Displays the Java code for `TestJugadores.java`. The code includes several test cases for methods like `estaExpulsado()` and `noEstaExpulsado()` using JUnit annotations (`@Test`).
- Console:** Shows the output of the test run, indicating it was terminated successfully.
- Failure Trace:** A stack trace for a `java.lang.AssertionError` pointing to the `TestJugadores` class.
- Search:** A search bar at the bottom with the placeholder "No search results available. Start a search from the search dialog."

Test NoEstaExpulsado (sin ninguna tarjeta)

Al principio no pasaba por que no había importado assertFalse. Después ya si .

The screenshot shows the Eclipse IDE interface. The left pane displays the code for `TestJugadores.java`. The right pane shows the terminal output of a test run.

```
workspace_jie - AD-2_javadoc_level/unit/test/quality/testJugadores.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer TestJugadores.java TestJugadores.java TestJugadores.java TestJugadores.java TestJugadores.java
Finished after 0.141 seconds
Run: 5/5 Errors: 0 Failures: 0
> de.TestJugadores [runner, KUnit] (0.074 s)
51     assertTrue(jugador2.estáExpulsado());
52 }
53
54     @Test
55     public void estáExpulsado2() {
56         boolean expulsado=false;
57         if(numeroTarjetasAmarillas==2 || numeroTarjetasRojas==1) {
58             jugador.estáExpulsado();
59         }
60         jugador2.setNúmeroTarjetasAmarillas(0);
61         jugador2.setNúmeroTarjetasRojas(1);
62
63         assertEquals(true,jugador2.estáExpulsado());
64     }
65
66     @Test
67     public void NoEstaExpulsado() {
68         boolean expulsado=false;
69         if(numeroTarjetasAmarillas==2 || numeroTarjetasRojas==1) {
70             jugador.estáExpulsado();
71         }
72         jugador2.setNúmeroTarjetasAmarillas(0);
73         jugador2.setNúmeroTarjetasRojas(0);
74
75         assertEquals(false,jugador2.estáExpulsado());
76     }
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99
```

The terminal output shows the test results:

```
<terminated> TestJugadores [JUnit] C:\Users\anyo\OneDrive\Escritorio\PROGRAMACION\Java\src\de\TestJugadores [runner, KUnit] (0.074 s)
5/5
```

Vanesa realiza también los test con **JUnit**.

Primero de la clase **Soldado**. Incluso he empezado probando con los *getter and setter* hasta que he arrancado para ver cómo hacerlo bien.

Todos los test salían bien hasta que he llegado al método **disparar()**

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer shows a single test run named 'SoldadoTest [Runner: JUnit 5] (0,023 s)' with 7 successful tests. The center shows the code for 'TestSoldado.java' and 'SoldadoTest.java'. The right side shows the 'Failure Trace' for a failed assertion error. The failure trace points to the 'testDisparaNoMata()' method in 'TestSoldado.java'.

```
82 * Para el método disparar(Soldado soldado) hay que probar:
83 * Cuando no tiene balas. No lo mata.
84 * Cuando tiene balas lo mata.
85 * Además podemos ver las balas que le quedan.
86 *
87 * He añadido en el método de dispara() una condición.
88 * Si tiene balas mata y resta y si no no hace nada.
89 *
90 */
91 @Test
92 public void testDisparaNoMata() {
93     Soldado soldadoA = new Soldado(false,0);
94     Soldado soldadoB = new Soldado(false,0);
95
96     soldadoA.disparar(soldadoB);
97     assertFalse(soldadoB.isEstaMuerto());
98
99     int resultadoEsperado = 0;
100    int resultado = soldadoA.getNumeroBalas();
101
102    assertEquals(resultadoEsperado, resultado);
103 }
104
105 @Test
106 public void testDisparaMata() {
107     Soldado soldadoA = new Soldado(false,0);
108     Soldado soldadoB = new Soldado(false,0);
109     soldadoB.setNumeroBalas(10);
110     soldadoB.disparar(soldadoA);
111
112     assertTrue(soldadoA.isEstaMuerto());
113
114     int resultadoEsperadoB = 9;
115     int resultadoB = soldadoB.getNumeroBalas();
116
117     assertEquals(resultadoEsperadoB, resultadoB);
118 }
119 }
```

Failure Trace:

```
java.lang.AssertionError
at junit5.SoldadoTest.testDisparaNoMata(SoldadoTest.java:102)
at java.base/java.util.ArrayList.forEach(ArrayList.java:154)
at java.base/java.util.ArrayList.forEach(ArrayList.java:134)
```

He revisado el método y el problema era que aunque no tuviera balas podía matar a otro soldado. De modo que lo he solucionado añadiéndole la condición de que si el número de balas es mayor de 0 entonces el otro soldado está muerto y además resta el número de balas.

```
71 /**
72  * Con este método podemos disparar a otro soldado. Cada vez que dispara resta balas,
73  * con lo cual cada vez que lo ejecutemos tendrá menos.
74  * <br>
75  * Además, si tiene balas matará a un soldado, si no tiene no podrá matarlo.
76  * @param sol en este parámetro le pasaremos un soldado al que va a matar o no.
77 */
78 public void disparar(Soldado sol) {
79     //He añadido esta condición para que si no tiene balas no mate al otro soldado
80     if(this.numeroBalas > 0) {
81         this.numeroBalas--;
82         sol.estaMuerto = true;
83     }
84 }
85
86 }
```

Ahora ya pasa todos los test realizados en **SoldadoTest**.

The screenshot shows the Eclipse IDE interface with the title bar "Actividad2-grupo8 - Actividad2_Req1/test/junit5/SoldadoTest.java - Eclipse". The left sidebar includes "Package Explorer", "JUnit", and "Failure Trace". The main editor area displays the code for `SoldadoTest.java`, which contains two test methods: `testDisparaNoMata()` and `testDisparaMata()`. The code uses JUnit annotations like `@Test` and `assertEquals` to verify the behavior of the `Soldado` class regarding bullet counts and death.

```
82 * Para el método disparar(Soldado soldado) hay que probar:  
83 * Cuando no tiene balas. No lo mata.  
84 * Cuando tiene balas lo mata.  
85 * Además podemos ver las balas que le quedan.  
86 *  
87 * He añadido en el método de dispara() una condición.  
88 * Si tiene balas mata y resta y si no no hace nada.  
89 *  
90 */  
91 @Test  
92 public void testDisparaNoMata() {  
93     Soldado soldadoA = new Soldado(false,0);  
94     Soldado soldadoB = new Soldado(false,0);  
95  
96     soldadoA.disparar(soldadoB);  
97     assertFalse(soldadoB.isEstaMuerto());  
98  
99     int resultadoEsperado = 0;  
100    int resultado = soldadoA.getNumeroBalas();  
101  
102    assertEquals(resultadoEsperado, resultado);  
103 }  
104  
105 @Test  
106 public void testDisparaMata() {  
107     Soldado soldadoA = new Soldado(false,0);  
108     Soldado soldadoB = new Soldado(false,0);  
109     soldadoB.setNumeroBalas(10);  
110     soldadoB.disparar(soldadoA);  
111  
112     assertTrue(soldadoA.isEstaMuerto());  
113  
114     int resultadoEsperadoB = 9;  
115     int resultadoB = soldadoB.getNumeroBalas();  
116  
117     assertEquals(resultadoEsperadoB, resultadoB);  
118 }  
119  
120 }
```

Los test de la clase **Jugador** han sido superados sin problema.

The screenshot shows the Eclipse IDE interface with the title bar "Actividad2-grupo8 - Actividad2_Req2/test/junit/JugadorTest.java - Eclipse IDE". The left sidebar includes "Package Explorer", "JUnit", and "Failure Trace". The main editor area displays the code for `JugadorTest.java`, which contains five test methods: `testAmarillasNoExpulsar()`, `testAmarillasExpulsar()`, `testRojasNoExpulsar()`, `testRojasExpulsar()`, and a constructor test. The code uses JUnit annotations to test the `Jugador` class's logic for yellow and red cards.

```
62     public void testAmarillasNoExpulsar() {  
63         Jugador jugador = new Jugador();  
64  
65         jugador.setNumeroTarjetasAmarillas(1);  
66  
67         assertFalse(jugador.estaExpulsado());  
68     }  
69  
70     @Test  
71     public void testAmarillasExpulsar() {  
72         Jugador jugador = new Jugador();  
73  
74         jugador.setNumeroTarjetasAmarillas(2);  
75  
76         assertTrue(jugador.estaExpulsado());  
77     }  
78  
79     @Test  
80     public void testRojasNoExpulsar() {  
81         Jugador jugador = new Jugador();  
82  
83         jugador.setNumeroTarjetasRojas(0);  
84  
85         assertFalse(jugador.estaExpulsado());  
86     }  
87  
88     @Test  
89     public void testRojasExpulsar() {  
90         Jugador jugador = new Jugador();  
91  
92         jugador.setNumeroTarjetasRojas(1);  
93  
94         assertTrue(jugador.estaExpulsado());  
95     }  
96  
97 }
```

ACTIVIDAD 2 ENTORNOS DESARROLLO. Parte de Alberto que no ha conseguido compartir finalmente su trabajo en el repositorio remoto que tenemos en común.

Requerimiento 1:

- En este primer paso he creado la clase soldado y la clase test para poder probar el correcto funcionamiento.
- Dentro del mismo podrá encontrar las anotaciones de Javadoc, haciendo especial hincapié en la cabecera del mismo explicando los private class y el creador del mismo ,más tarde explicando correctamente los métodos con los return, param,
- Por último utilice las técnicas de Junit para asegurarme del correcto funcionamiento del software. Creando una clase específica para su respectiva prueba.
- A Continuación se puede ver en las capturas todo el trabajo realizado.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "Actividad2Entornos [git5 developAlb2]". It includes a JRE System Library (JavaSE-17) and several source files under the src folder:
 - Junit1
 - SoldadoJUnitTest.java
 - Junit2
 - jugadorTestJunit.java
 - Requerimiento1
 - Soldado.java
 - TestSoldado.java
 - Requerimiento2
 - Jugador.java
 - Rrequerimiento2
 - Jugador.java
 - TestJugador.java
- Code Editor:** Displays the content of TestSoldado.java. The code creates two Soldado objects (soldado1 and soldado2) with 5 and 3 bullets respectively. It then checks if they can shoot and prints their status. Both are then shot at each other. The output shows that soldado1 is dead (isEstaMuerto() returns true) while soldado2 has 2 bullets left.

```
7 * Con esta clase crearemos soldados, los cuales podran tener balas, matar a otro soldado y ...
8 *
9 * @author Alb_Sánchez
10 */
11
12 public static void main(String[] args) {
13     //Creamos sold1 y sold2
14     Soldado soldado1 = new Soldado(false, 5);
15     Soldado soldado2 = new Soldado(false, 3);
16
17     //Preguntamos si pueden disparar. Sold1 tiene balas, sold2 no
18     System.out.println("¿Puede disparar el soldado1? " + soldado1.puedeDisparar() + " balas");
19     System.out.println("¿Puede disparar el soldado2? " + soldado2.puedeDisparar() + " balas");
20
21     System.out.println(" Soldado1 dispara a soldado2 ");
22     if (soldado1.puedeDisparar())
23         soldado1.disparar(soldado2);
24
25     System.out.println("¿ha muerto soldado2? " + Soldado.isEstaMuerto());
26
27     System.out.println("Soldado1 tiene " + soldado1.getNumeroBalas() + " balas, no puede disparar");
28
29     System.out.println(" Soldado2 dispara a soldado1.");
30     if (soldado2.puedeDisparar())
31         soldado2.disparar(soldado1);
32
33     System.out.println("¿ha muerto soldado1? " + Soldado.isEstaMuerto());
34
35     System.out.println("¿ha muerto soldado2? " + Soldado.isEstaMuerto());
36     System.out.println("Soldado2 vuelve a tener " + soldado2.getNumeroBalas() + " balas");
37 }
```
- Console:** Shows the terminal output of the application. It prints the interactions between the two soldados, including their bullet counts and whether they can shoot, followed by the final state where soldado2 has 2 bullets left.

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left displays a project structure under 'Actividad2Entornos [git5 developAlb2]'. The Java code editor in the center shows the file 'SoldadoJUnitTest.java' with JUnit test cases for the 'Soldado' class. The Console view at the bottom shows the output of running the application, displaying player statistics and expulsion status.

```

package Junit;
import Requerimiento1.Soldado;
public class SoldadoJUnitTest {
    public void testEstaMuerto() {
        assertFalse(Soldado.isEstaMuerto());
    }
    public void testEstaMuerto2() {
        assertFalse(Soldado.isEstaMuerto());
    }
    private void assertFalse(boolean estaMuerto) {
        // TODO Auto-generated method stub
    }
    public void testTieneBalas() {
        Soldado soldado = new Soldado(false,10);
        int resultadoEsperado = 10;
        int resultado = soldado.getNumeroBalas();
        assertEquals(resultadoEsperado, resultado);
    }
    private void assertEquals(int resultadoEsperado, int resultado) {
    }
}

```

```

terminated> TestJugador [Java Application] C:\Users\bereb\Downloads\eclipse\plugins\org.eclipse.jdt.core\org.eclipse.jdt.core\src\main\java\Requerimiento1\Soldado.java
jugador 1 tiene 2 tarjetas amarillas y 1 tarjetas rojas.
¿Está expulsado? true
jugador 2 tiene 4 tarjetas amarillas y 2 tarjetas rojas.
¿Está expulsado? false
jugador 3 tiene 1 tarjetas amarillas y 0 tarjetas rojas.
¿Está expulsado? false

```

Como se puede observar en las 2 imágenes, podemos ver como efectivamente he realizado los commit y add del trabajo realizado en git, quedando claro con la marca cilíndrica.

Requerimiento 2:

- En este segundo requerimiento realice las mismas anotaciones sobre las private class con sus respectivos parámetros e indicando correctamente en qué consistirá el funcionamiento de cada una. Más tarde explico con las mismas anotaciones y con el return la parte de los métodos, que aunque un poco más complejo que el de la clase soldado, no ha supuesto mayor dificultad.
- Para finalizar realice las pruebas de test con los Junit y pueda comprobar su perfecto funcionamiento.
- A continuación muestro con las capturas al igual que antes todo el proceso y como comenté anteriormente se puede observar mediante la marca cilíndrica que efectivamente puede realizar los commit y add en git para que quedara documentado.

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure for "Actividad2Entornos [git5 developAlb2]".
- Soldado.java:** Contains code for initializing three players (jugador1, jugador2, jugador3) and printing their dorsal numbers.
- TestSoldado.java:** Contains code for testing the dorsal numbers of the players.
- Console Output:**

```
<terminated> TestJugador [Java Application] C:\Users\bereb\Downloads\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20200511-1059
jugador 1 tiene 2 tarjetas amarillas y 1 tarjetas rojas.
¿Está expulsado? true
jugador 2 tiene 4 tarjetas amarillas y 2 tarjetas rojas.
¿Está expulsado? false
jugador 3 tiene 1 tarjetas amarillas y 0 tarjetas rojas.
¿Está expulsado? false
```

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure for "Actividad2Entornos [git5 developAlb2]".
- Soldado.java:** Contains code for initializing three players (jugador1, jugador2, jugador3) and printing their dorsal numbers.
- jugadorTestJunit.java:** Contains JUnit test cases for the Jugador class.
- Console Output:**

```
<terminated> TestJugador [Java Application] C:\Users\bereb\Downloads\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20200511-1059
jugador 1 tiene 2 tarjetas amarillas y 1 tarjetas rojas.
¿Está expulsado? true
jugador 2 tiene 4 tarjetas amarillas y 2 tarjetas rojas.
¿Está expulsado? false
jugador 3 tiene 1 tarjetas amarillas y 0 tarjetas rojas.
¿Está expulsado? false
```

Aclaración:

Como han comentado mis compañeras me ha sido imposible subir los cambios de la actividad al repositorio remoto, ya que he probado varias veces distintos tipos de push para comprobar si estaba

ahi el problema, incluso realice el trabajo completo por segunda vez y puse en practica todas las técnicas de git, no obstante no consigo que se realice el push al repositorio remoto.

```
.\Users\bereb\Desktop\git5>git add .
:.\Users\bereb\Desktop\git5>git status
on branch developAlb2
changes to be committed:
(use "git restore --staged <file>..." to unstage)
    modified:   Actividad2Entornos/src/Junit1/SoldadoJunitTest.java
    modified:   Actividad2Entornos/src/Requerimiento1/Soldado.java
    modified:   Actividad2Entornos/src/Requerimiento1/TestSoldado.java

:.\Users\bereb\Desktop\git5>git commit -m"todo completo"
[developAlb2 5e2fe3f] todo completo
 3 files changed, 8 insertions(+), 10 deletions(-)

:.\Users\bereb\Desktop\git5>git status
on branch developAlb2
nothing to commit, working tree clean

:.\Users\bereb\Desktop\git5>git push origin developAlb2
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

:.\Users\bereb\Desktop\git5>
```

Aqui dejo una captura de la rama realizada.

The screenshot shows a GitHub repository page for 'VanesaLB / Actividad2'. The 'Code' tab is selected. The 'developAlb' branch is currently selected. It shows 3 branches and 0 tags. A message indicates that this branch is 1 commit behind 'developvane'. The commit list shows several JavaDoc files and a README file, all committed by 'VanesaLB' 5 days ago.

File	Description	Commit
Actividad2-grupo8	Actividad 2 Vanesa, parte JavaDoc.	11e2ddc 5 days ago
Documentacion-act2-r1	Actividad 2 Vanesa, parte JavaDoc.	
Documentacion-act2-r2	Actividad 2 Vanesa, parte JavaDoc.	
capturas-jugador	Actividad 2 Vanesa, parte JavaDoc.	
capturas-soldado	Actividad 2 Vanesa, parte JavaDoc.	
.gitignore	Actividad 2 Vanesa, parte JavaDoc.	
README	Actividad 2 Vanesa, parte JavaDoc.	

Finalmente los trabajos que hemos podido comparar han sido los de Estefanía y Vanesa. No hemos tenido tiempo de ver el trabajo de Alberto que ha hecho lo posible por llegar a tiempo a la fecha de entrega.

Los JavaDoc eran muy similares. Respecto a los test realizados con JUnit hemos seleccionado el de Vanesa por la solución propuesta para que funcione el método dispara() de la clase Soldado.

Te recordamos el GitHub:

<https://github.com/VanesalB/Actividad2>