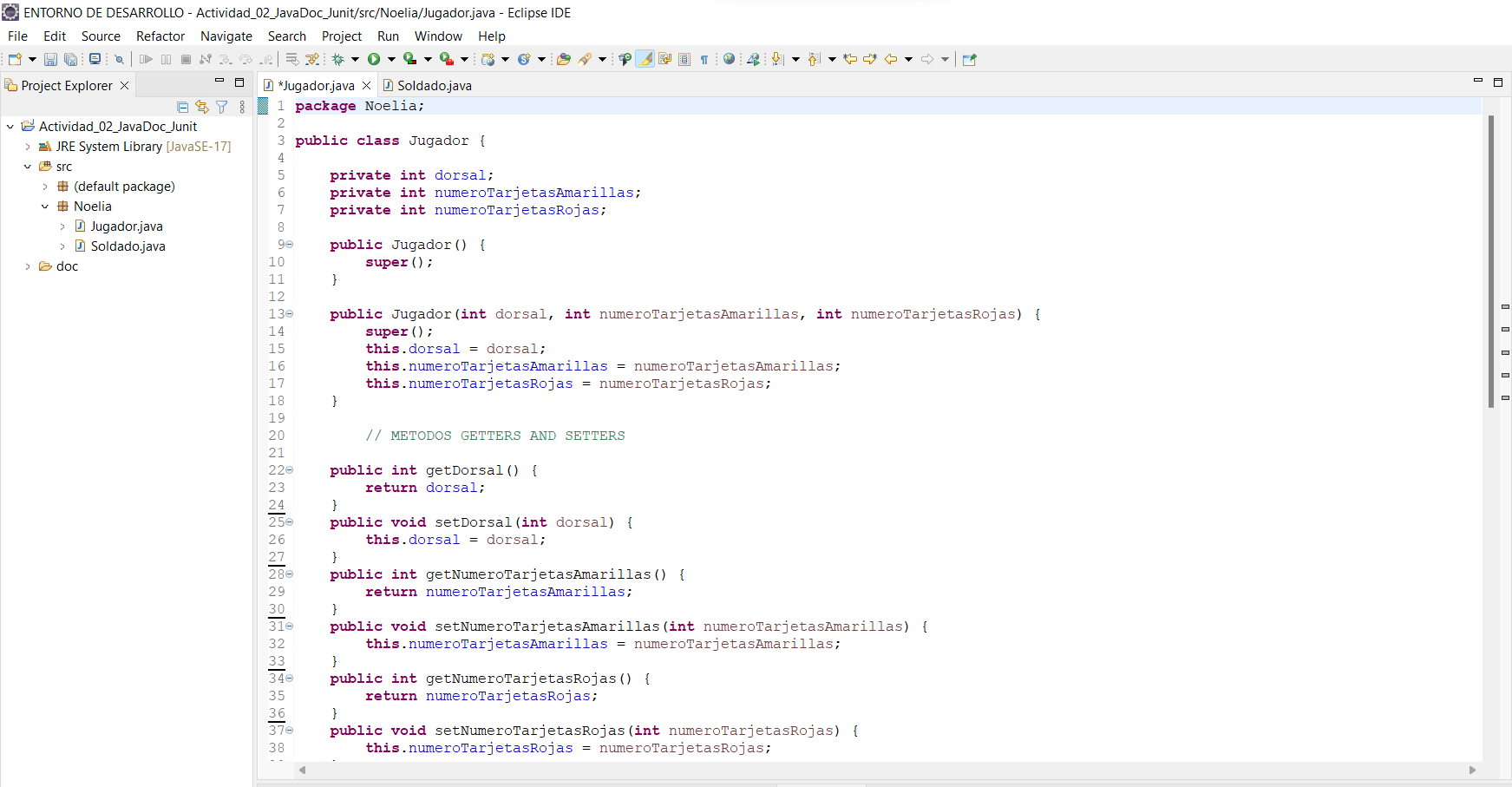
ENTORNOS DE DESARROLLO

ACTIVIDAD 2

Se inicia un proyecto en el ID y en él se genera en un paquete “Noelia” donde se añaden las clases pertinentes, en este caso Soldado y Jugador.

En cada clase se generan los atributos, constructores, métodos GET and SET, método TO STRING y los métodos propios para trabajar con la clase.

Ejemplo de la clase Jugador:



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ahora, tal como pide la actividad, se deben documentar ambas clases con JavaDoc.

El JavaDoc se crea dentro de las clases de java y se genera con:

/\*\*

\*

\*/

La documentación que se realice aparecerá de color azul, también se ha comentado ciertas cosas en las clases, pero sin generar documentación, esta información se diferenciará porque aparece de tal forma:

//

/\*

\*/

Y porque su aspecto es de color verde.

Una forma de generar JavaDoc desde eclipse es en: source Generate Element Comment.

Al comenzar la documentación de la clase se puede añadir anotaciones de tipo @autor el cual especifica quien realiza el código. Y @version para saber en qué versión nos encontramos.

Al documentar los métodos siempre debemos añadir también @param para describir los parámetros que va a necesitar dicho método, en caso de que el método devuelva algún resultado se le asigna la anotación @return, si el método es VOID es innecesario añadir dicha anotación.

Los GETER y los SETTER no los he comentado, solo se comentan en caso de que la funcionalidad de estos cambie o se modifiquen.

Eclipse → click sobre el proyecto donde queremos trabajar → Project → Generate JavaDoc → escoger el fichero java.exe → finish. Se observa que no salgan errores.

Ahora al meternos a la carpeta DOC que reside en la carpeta donde tenemos guardado el trabajo y pinchamos sobre el fichero INDEX, nos deben de salir las clases creadas donde podemos acceder a la información detallada dichas clases.

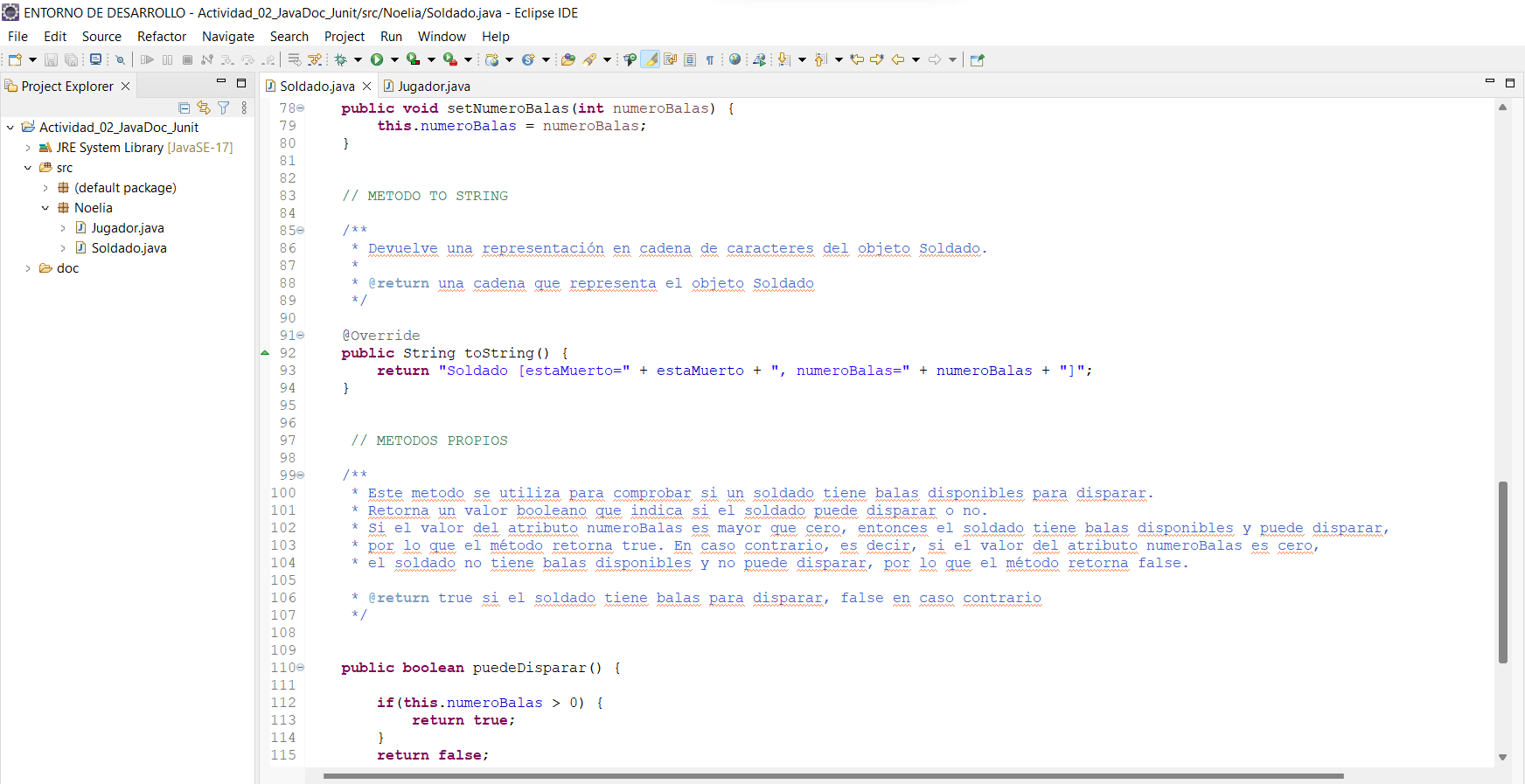
Ejemplo de la clase Soldado:

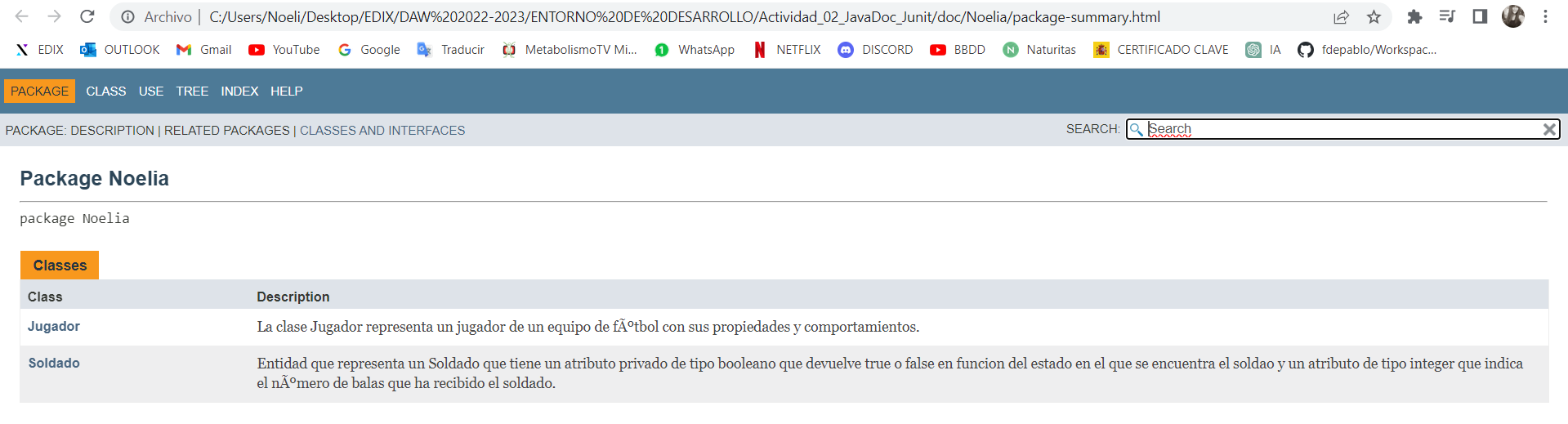
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente





Una vez que se ha realizado la documentación con JavaDoc en cada clase se debe generar el módulo de pruebas Junit. Para generarlo creamos primero una carpeta en nuestro proyecto a la que llamamos test la cual hay que incluirla en el “classpath” de la siguiente forma: pinchamos en la carpeta → Build path → configure build path.

Dentro de la configuración pulsamos en source → add folder (elegimos la carpeta test) → apply. Y seguidamente creamos el módulo de Junit: pinchamos en la carpeta test → new → other → Junit test case → next → valores por defecto (aun que es una buena practica no dejar el package default, por lo que le damos nombre: ejemplo Junit5). Por último le damos nombre: testSoldado / testJugador. → FINISH.

El siguiente paso es realizar las pruebas unitarias pertinentes en cada clase. Como ejemplo pondré los resultados de la clase SOLDADO.

En la clase Soldado existen dos métodos para probar sus diferentes posibilidades de resultados:

El primer método de la clase Soldado. solo tiene dos posibilidades:

Que pueda disparar si el soldado tiene balas y está vivo (retorna true) o que no pueda disparar porque el soldado está vivo, pero no tiene balas (retorna false).

**public** **boolean** puedeDisparar() {

**if** (**this**.numeroBalas > 0) {

**return** **true**;

}

**return** **false**;

primera prueba:

Se comprueba que el soldado está vivo y con balas (en este caso con 1) y puede disparar.

Introduzco los parámetros false (significa que no está muerto) y numero de balas (ejemplo: le queda una) para ver si estos datos esperados son iguales a los que se obtienen.

Segunda prueba:

Se comprueba que el soldado está vivo y sin balas (en este caso con 0) por lo tanto no puede disparar.

Introduzco los parámetros false (significa que no está muerto) y numero de balas (ejemplo: cero) para ver si estos datos esperados son iguales a los que se obtienen.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El segundo método de la clase Soldado, he realizado dos posibilidades:

Si el soldado tiene balas puede disparar al soldado que me pasan por parámetro y por tanto dicho soldado muere.

Si el soldado no tiene balas no puede disparar al soldado que me pasan por parámetro y por tanto el soldado que me pasan por parámetro está vivo.

**public** **void** disparar (Soldado sol) {

**this**.numeroBalas --;

sol.estaMuerto = **true**;

}

Primera prueba:

Si Soldado tiene balas soldado.setNumeroBalas(10);, Soldado puede disparar a Sol soldado.disparar(sol);y por tanto Sol muere devolviendo TRUE sol.isEstaMuerto(); es true

Segunda prueba:

Si soldado no tiene balas soldado.setNumeroBalas(0); entonces Soldado no puede disparar y por tanto Sol permanecerá vivo, por tanto no es posible que devuelva TRUE, es FALSE.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el caso de que alguna prueba diese fallo por un error al desarrollarlas, me saldría el test en ROJO y habría que revisar y modificar la prueba para corregir los fallos que se hayan cometido hasta que se ejecute como válida.

Las pruebas unitarias para la clase Jugador seria el mismo proceso que para la clase Soldado, es decir, ir viendo las posibilidades que tienen los métodos de la clase de resolverse y proceder a realizar las pruebas de esas posibilidades para ver que no se produce ningún error y así se consigan unas pruebas válidas.

RESULTADOS de las pruebas en la clase Jugador:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente