|  |
| --- |
|  |
| Ampliación de Ingeniería del Software |
| Memoria-Práctica 1 |
|  |
| **Paula Sestafe y Pablo Castaño** |
|  |

|  |
| --- |
| Doble grado en Ingeniería Informática y Administración y Dirección de Empresas |



ÍNDICE

[PRUEBAS UNITARIAS DE LA CLASE BOARD 3](#_Toc515200070)

[PRUEBAS CON DOBLES DE LA CLASE TICTACTOEGAME 4](#_Toc515200071)

[PRUEBAS DE SISTEMA DE LA APLICACIÓN 5](#_Toc515200072)

[INTEGRACIÓN CONTINUA 6](#_Toc515200073)

# PRUEBAS UNITARIAS DE LA CLASE BOARD

Este tipo de pruebas están implementadas con JUnit 4.

En primer lugar, se implementa un **setUp()** cuya función es inicializar los jugadores de la partida y crear el tablero en el que se va a jugar. Dicho método va precedido con la etiqueta @Before, para que sea ejecutado antes de ejecutar cualquier test.

Así pues, a continuación, únicamente creamos tres tests barajando las siguientes posibilidades:

1. El primer jugador en poner ficha, gana: Marcamos con las fichas correspondientes al jugador 1, que en este caso es la "X", las celdas {0,1,2}. De esta forma nos aseguramos que el jugador 1 gane sin necesidad de que el jugador 2 haga ningún movimiento.

Para comprobarlo únicamente comparamos el array {0,1,2}, con el array que tiene que devolver la función *getCellsIfWinner(pl1.getLabel())* en el caso de que sea el jugador 1 el ganador. Dicha comparación se hace con un **assertArrayEquals**.

1. El segundo jugador en poner ficha, gana: Marcamos con las fichas correspondientes al jugador 2, que en este caso es la "O", las celdas {3,4,5}. De esta forma nos aseguramos que el jugador 2 gane a pesar de que el jugador 1 haya iniciado los movimientos de la partida.

Para comprobarlo únicamente comparamos el array {3,4,5}, con el array que tiene que devolver la función *getCellsIfWinner(pl2.getLabel())* en el caso de que sea el jugador 2 el ganador. Dicha comparación se hace con un **assertArrayEquals**.

1. Ninguno de los dos jugadores gana, hay empate: Marcamos con las fichas correspondientes al jugador 1, que en este caso es la “X” las celdas {0,2,4,5,7}. En cuanto al jugador 2, marcamos con las fichas correspondientes, que en este caso es la “O” las celdas {1,3,6,8}, lo que da lugar a empate. Para comprobar este hecho utilizamos el método **assertNull** sobre las variables result1 y result2 que comprueban que ambas variables tiene valor nulo. Esto deber ser así, ya que, al no haber ganador, la función *getCellslfWinner* deberá devolver valor nulo. Además también utilizamos **assertTrue** sobre el método *checkDraw()*, para comprobar que el valor devuelto por dicho método es igual a true, lo que significa que el tablero está completo sin ninguna línea con fichas iguales.

# PRUEBAS CON DOBLES DE LA CLASE TICTACTOEGAME

Este tipo de pruebas están implementadas con JUnit5

En primer lugar, se definen los atributos de la clase, que en este caso son juego de tipo TicTacToeGame, pl1 y pl2 que serán los dos jugadores de tipo Player y c1 y c2 que serán los dobles de las dos conexiones de tipo Connection.

A continuación se implementa un **setUp()** cuya función es inicializar los jugadores de la partida, inicializar el juego, así como dar de alta los dos dobles de conexiones c1 y c2, llamando al método *addConnection*. Además, los jugadores de tipo Player, se añaden al juego llamando al método *addPlayer* por cada jugador. Cabe destacar que el método **setUp()** va precedido con la etiqueta @BeforeEach, para que sea ejecutado antes de ejecutar cualquier test.

A continuación creamos 3 test para cada una de las situaciones posibles:

1. El primer jugador en poner ficha, gana:

En primer lugar, comprobamos que se ha creado y se ha enviado un evento de tipo JOIN\_GAME con ambos jugadores a los navegadores web. Después de esto es el momento en el que se empieza a jugar y por tanto se empiezan a marcar las casillas correspondientes. Para ello se comprueba la creación y envío del evento SET\_TURN en cada turno y se marca la casilla elegida por el jugador.

Una vez terminado el juego, se comprueba que se ha enviado el evento GAME\_OVER, así como que el valor guardado en la variable **WinnerValue** corresponde al jugador 1. Esto se realiza mediante un **assertThat.**

1. El segundo jugador en poner ficha, gana:

Este caso es idéntico al anterior, cambiando únicamente la comprobación del valor del ganador final, que corresponderá al jugador 2.

En primer lugar, comprobamos que se ha creado y se ha enviado un evento de tipo JOIN\_GAME con ambos jugadores a los navegadores web. Después de esto es el momento en el que se empieza a jugar y por tanto se empiezan a marcar las casillas correspondientes. Para ello se comprueba la creación del evento SET\_TURN en cada turno y se marca la casilla elegida por el jugador.

Una vez terminado el juego, se comprueba que se ha creado el evento GAME\_OVER, así como que el valor guardado en la variable **WinnerValue** corresponde al jugador 2. Esto se realiza mediante un **assertThat.**

1. Ninguno de los jugadores gana, hay empate.

Este caso es parecido a los dos anteriores, cambiando la comprobación del valor del ganador final, que en este caso tendrá que ser Null, lo que comprobaremos con un **assertNull.**

# PRUEBAS DE SISTEMA DE LA APLICACIÓN

En este apartado hemos realizado pruebas con Selenium con el objetivo de simular una partida y comprobar el buen funcionamiento de la aplicación. El código utilizado se detalla a continuación:

En primer lugar, instanciaremos los dos objetos webdriver previamente creados, para después pasar a la localización de los elementos en la página, como los ‘nicknames’. Una vez hecho esto pasamos a la programación de los test según las diferentes opciones:

1. El primer jugador en poner ficha, gana:

En primer lugar, cada jugador va marcando las casillas correspondientes (en este caso el jugador1 marca la 0, la 4 y la 8, lo que le convierte en ganador final) y se envía el mensaje indicando el ganador. A través de un *thread.sleep(),* dejamos algo de tiempo para que se pueda captar el mensaje correctamente. Posteriormente mediante *switchTo().alert().getText()* captamos el mensaje y lo guardamos en una variable de tipo String. Por último, mediante un **assertEquals**, comprobamos que el mensaje esperado y el mensaje captado es el mismo (en este caso deber ser algo que nos indique que gana el jugador1).

1. El segundo jugador en poner ficha, gana:

Este test será similar al anterior, cambiando únicamente el ganador final, que será el jugador2.

En primer lugar, cada jugador va marcando las casillas correspondientes (en este caso el jugador2 marca la 1, la 4 y la 7, lo que le convierte en ganador final) y se envía el mensaje indicando el ganador. A través de un *thread.sleep(),* dejamos algo de tiempo para que se pueda captar el mensaje correctamente. Posteriormente mediante *switchTo().alert().getText()* captamos el mensaje y lo guardamos en una variable de tipo String. Por último, mediante un **assertEquals**, comprobamos que el mensaje esperado y el mensaje captado es el mismo (en este caso deber ser algo que nos indique que gana el jugador2).

1. Ninguno de los jugadores gana, hay empate:

Este test será similar a los dos anteriores cambiando en este caso que no habrá ganador final, por lo que se producirá un empate.

En primer lugar, cada jugador va marcando las casillas correspondientes (en este caso el jugador1 marca la 0, la 3, la 4, la 2 y la 7, por lo que el jugador 2 marcará la 1, la 6, la 5 y la 8, lo que dará como resultado final un empate) y se envía el mensaje indicando el ganador o el posible empate, como sucede en esta ocasión. A través de un *thread.sleep(),* dejamos algo de tiempo para que se pueda captar el mensaje correctamente. Posteriormente mediante *switchTo().alert().getText()* captamos el mensaje y lo guardamos en una variable de tipo String. Por último, mediante un **assertEquals**, comprobamos que el mensaje esperado y el mensaje captado es el mismo (en este caso deber ser algo que nos indique que se ha producido un empate).

# INTEGRACIÓN CONTINUA