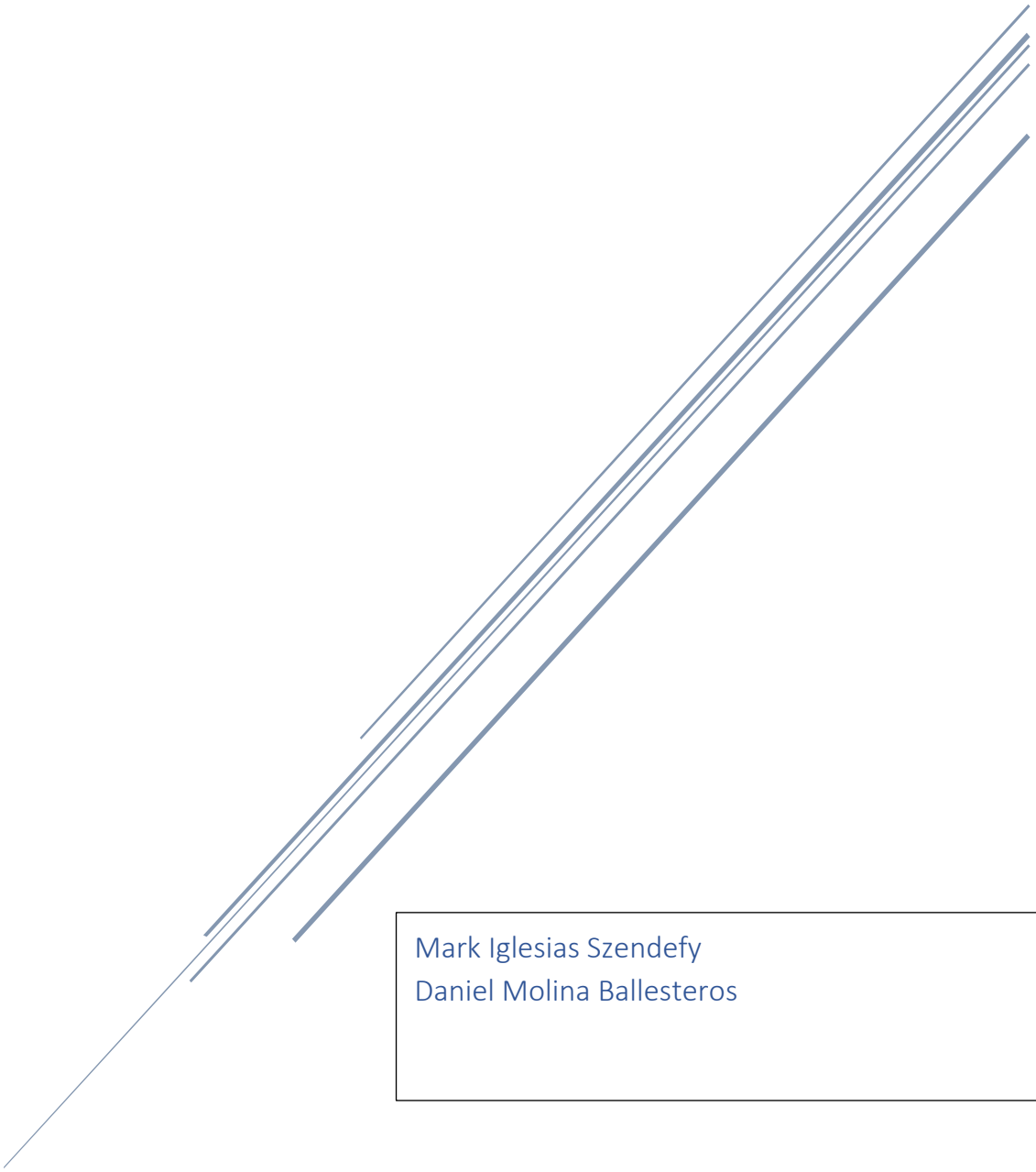
IMPLEMENTACIÓN DE PRUEBAS AUTOMÁTICAS

Práctica 1. Memoria.



Mark Iglesias Szendefy
Daniel Molina Ballesteros

INDICE

Pruebas unitarias de la clase Board

En este Test someteremos a pruebas unitarias para comprobar el correcto funcionamiento de la clase “Board”. Para ello inicializaremos en “setUp” una clase “Board” que en “tearDown” liberaremos. Para ejecutar los test nos ayudaremos de 2 funciones: “turnoJugador” que le pasaremos la etiqueta de ficha del jugador y la posición de la casilla y marcará la casilla y la activará simulando que la ha seleccionado un jugador; la otra función es “lineaJugador” que se le pasa la etiqueta del jugador y las casillas seleccionadas y dice que debería haber pasado al jugador con la etiqueta correspondiente, si ganar o no, en caso de ganar dice que casillas fueron las ganadoras. Implementamos 3 test simulando que dos jugadores juegan: el primero que el jugador con ficha ‘X’, el cual pone primero, gana y para ello comprobamos “getCellsIfwinner“ con ambos jugadores en el cual el primero debería devolver las casillas de la línea ganadora y para el segundo devolver ‘null’, también comprobamos que “checkDraw” da ‘False’; en el segundo test simulamos que gana el otro jugador y hacemos lo mismo, los resultados del “getCellsIfwinner“ deben cambiar y mantener el comportamiento de “checkDraw”; por último comprobamos si quedan empate que cuando llamamos “getCellsIfwinner“ ambos dan ‘null’ y que “checkDraw” es ‘True’.
Al hacer los ‘assert’ hemos incluido in String para saber cuál debería ser el resultado esperado.

Pruebas con dobles de la clase TicTacToeGame

En estos test comprobaremos el correcto funcionamiento de la clase “TicTacToeGame”. Para ello, simularemos con Mockito el comportamiento de las clases “Connection”.
Empezaremos creando en el método “setUp” la inicialización de los parámetros necesarios para la ejecución de los test. Inicializaremos los jugadores de la aplicación, las clases “Connection” con “Mockito” y añadiremos ambos a la clase “TicTacToeGame”. Los métodos “TearDown” será para cerrar las conexiones y liberar los recursos utilizados.
Probaremos las dos conexiones que se han realizado correctamente y pasan los dos jugadores cuando ya se han añadido los dos jugadores con dos test, uno para cada “connection”.
También probaremos en un que si un jugador marca una casilla que ya está marcada el tuno no cambia.
Por ultimo, en los siguientes test simularemos la partida mediante el método “mark”, que es el encargado de marcar las casillas y cambiar los turnos. Realizamos aserciones para comprobar que los cambios de turnos se realizan correctamente tras cada “mark”, para ello debería salir ‘False’ en el jugador que no tiene el turno y ‘True’ en el que le corresponde el turno.
Por último, comprobamos el resultado mirando si el valor devuelto por “sendEvent” es ‘null’ que significaría empate o ‘winner’ si es el ganador para ello ejecutamos 3 test combrobando que funciona correctamente cuando gana el primer jugador que pone ficha con un test, con otro si gana el segundo jugador que pone ficha y por último que no sale que ha ganado ninguno cuando empatan.

Pruebas de sistema de la aplicación

Para estos test usaremos Selenium con 'WebDrivers' que simularan las acciones que se le hacen a la aplicación Web. Para inicializar los 'WebDrivers' primero en "setUpClass" se configuran para el navegador que vayamos a usar, en nuestro caso de Chrome, y se inicializan en "setUp" junto con las demás estructuras y variables que vayamos a utilizar, que liberaremos en "tearDown". En "setUpClass" y "tearDownClass" iniciaremos "WebApp.start" y pararemos con "WebApp.stop" la aplicación para ejecutar los test.
En "setUp" inicializamos dos WebDrivers y buscamos la página: "http://localhost:8080/" con cada uno. En cada uno iniciaremos un jugador.
Haremos 3 test, en cada uno simularemos una partida, en el primero gana el primer jugador registrado, en el primer jugador pierde, es decir, gana el segundo jugador. Y en el último no ganará ninguno. Para comprobar el funcionamiento correcto al terminar la partida aparece un mensaje en forma de 'alert' que indica quien ha ganado y quien ha perdido o si ha habido empate, el mensaje es el mismo para ambos jugadores así que se recogerá el mensaje y comprobamos si es correcto.

Job de Jenkins

El job ejecuta un pipeline, utiliza Maven "M3", accede al repositorio de GitHub donde tenemos el proyecto. No genera un .jar debido a que en los test se puede ejecutar la aplicación web si lo necesita sin necesidad de este. Accede a donde se encuentran los test y los ejecuta. Después de ejecutarlos se recopilan los resultados de los test jUnit para que aparezcan en la interfaz de Jenkins.