Master Software Craftmanship

2018/2019

Memoria Práctica 2

Implementación de pruebas automáticas

Autores

* Javier Antonio Valdez Palomino
* Francisco Javier Jiménez Olivares

Contents

[Pruebas Automáticas 2](#_Toc7131819)

[Pruebas unitarias de la clase Board 2](#_Toc7131820)

[Objetivo 2](#_Toc7131821)

[Implementación 2](#_Toc7131822)

[Pruebas con dobles de la clase TicTacToeGame 3](#_Toc7131823)

[Objetivo 3](#_Toc7131824)

[Implementación 3](#_Toc7131825)

[Pruebas de sistema de la aplicación 4](#_Toc7131826)

[Objetivo 4](#_Toc7131827)

[Implementación 4](#_Toc7131828)

# Pruebas Automáticas

## Pruebas unitarias de la clase Board

### Objetivo

En estas pruebas se comprueba que la clase Board implementa correctamente la detección de que un jugador ha ganado, perdido o empatado.

Se implementan varias pruebas que cambian el estado de las celdas del tablero haciendo uso del método getCell() y se verifican los métodos getCellsIfWinner() y checkDraw().

### Implementación

Como se ha solicitado se ha implementado que el primer jugador que pone ficha gana, el primer jugador que pone ficha pierde y ninguno de los jugadores gana, hay un empate.

Para estas Pruebas Unitarias solo se ha necesitado la simulación de un solo jugador (jugador “X”) para implementar los tres test solicitados, en realidad solo hemos tenido que modificar el estado de las celdas como si hubiera sido realizado por el jugador.

En cada una las pruebas se verifican que:

* El procedimiento getCellsIfWinner() devuelve los valores correctos de acuerdo al jugador que se indica como parámetro de entrada. Esto es:
  + La lista de celdas que componen la jugada ganadora.
  + O Null en caso de que el jugador no se el ganador.
* El procedimiento checkDraw()detecta correctamente la situación de empate, devolviendo:
  + False si no hay empate.
  + True si hay empate.

## Pruebas con dobles de la clase TicTacToeGame

### Objetivo

En estas pruebas se comprueba que la clase TicTacToeGame implementa de forma adecuada el juego. Se llama a los distintos métodos de la clase simulando los mensajes que llegan de los navegadores durante el uso normal de la aplicación.

Además, se deberán simular las conexiones creando dobles de los objetos Connection (que representan una comunicación WebSockets). Estas conexiones simuladas se utilizarán para verificar si los eventos que envía TicTacToeGame a los navegadores web (representados por los dobles de Connection) son los esperados.

### Implementación

Como se ha solicitado se ha implementado que el primer jugador que pone ficha gana, el primer jugador que pone ficha pierde y ninguno de los jugadores gana, hay un empate.

Se hace uso del método TicTaeToeGame.mark() para realizar la jugada por cada jugador.

Se ha incluido un ArrayList (markList) para el facilitar el manejo correcto de los turnos de los jugadores en un ciclo controlado, cada elemento de la lista es una celda específica(jugada) y el índice de la lista % 2 determina que jugador hizo la jugada (el resultado índice de la lista % 2 si es 0 es jugador1 y si no es el jugador2).

En cada una las pruebas se verifican que:

* Que la función sendEvent de la clase Connection.class (gracias al uso del mock) se llame con los parámetros correctos.
  + Después de que el primer jugador entra al juego usando el método TicTaeToeGame.addPlayer() los parámetros esperados son:
    - EventType.JOIN\_GAME, y solamente el jugador que se añade.
  + Después de que el segundo jugador entra al juego usando el método TicTaeToeGame.addPlayer() los parámetros esperados son:
    - EventType.JOIN\_GAME, y la lista del numero de jugadores que se lleva añadido (en este caso2)
  + Después de cada jugada hecha usando el método TicTaeToeGame.mark() los parámetros esperados son:
    - EventType.SET\_TURN, y jugador que le toca su turno
  + Cuando algún jugador que gana después de ejecutar la lista de jugadas los parámetros esperados son:
    - EventType.GAME\_OVER, y el jugador ganador
  + Cuando hay un empate y ninguno de los 2 jugadores gana jugadas los parámetros esperados son:
    - EventType.GAME\_OVER, y un NULL

## Pruebas de sistema de la aplicación

### Objetivo

Se implementan pruebas de sistema con Selenium. Para simular una partida la prueba inicia dos navegadores web de forma simultánea e interactúa con ellos de forma alternativa. De esta forma, se simula una partida por turnos.

### Implementación

Como se ha solicitado se ha implementado que el primer jugador que pone ficha gana, el primer jugador que pone ficha pierde y ninguno de los jugadores gana, hay un empate.

Levantamos la WebApp proporcionada por el juego de forma local.

Nos conectamos con 2 WebDrivers uno para cada jugador.

Se ha incluido un ArrayList (markList) para el facilitar el manejo correcto de los turnos de los jugadores en un ciclo controlado, cada elemento de la lista es una celda específica(jugada) y el índice de la lista % 2 determina que jugador hizo la jugada (el resultado índice de la lista % 2 si es 0 es jugador1 y si no es el jugador2).

Se verifica que la ventana de notificación nos indica la información correcta dependiendo del resultado de la partido:

* Cuando el primer jugador que mueve gana.
  + Menaje esperado: “player1 wins! player2 looses.”
* Cuando el primer jugador que mueve pierde.
  + Menaje esperado: “player2 wins! Player1 looses.”
* Cuando ninguno gana, empate.
  + Menaje esperado: “Draw!”