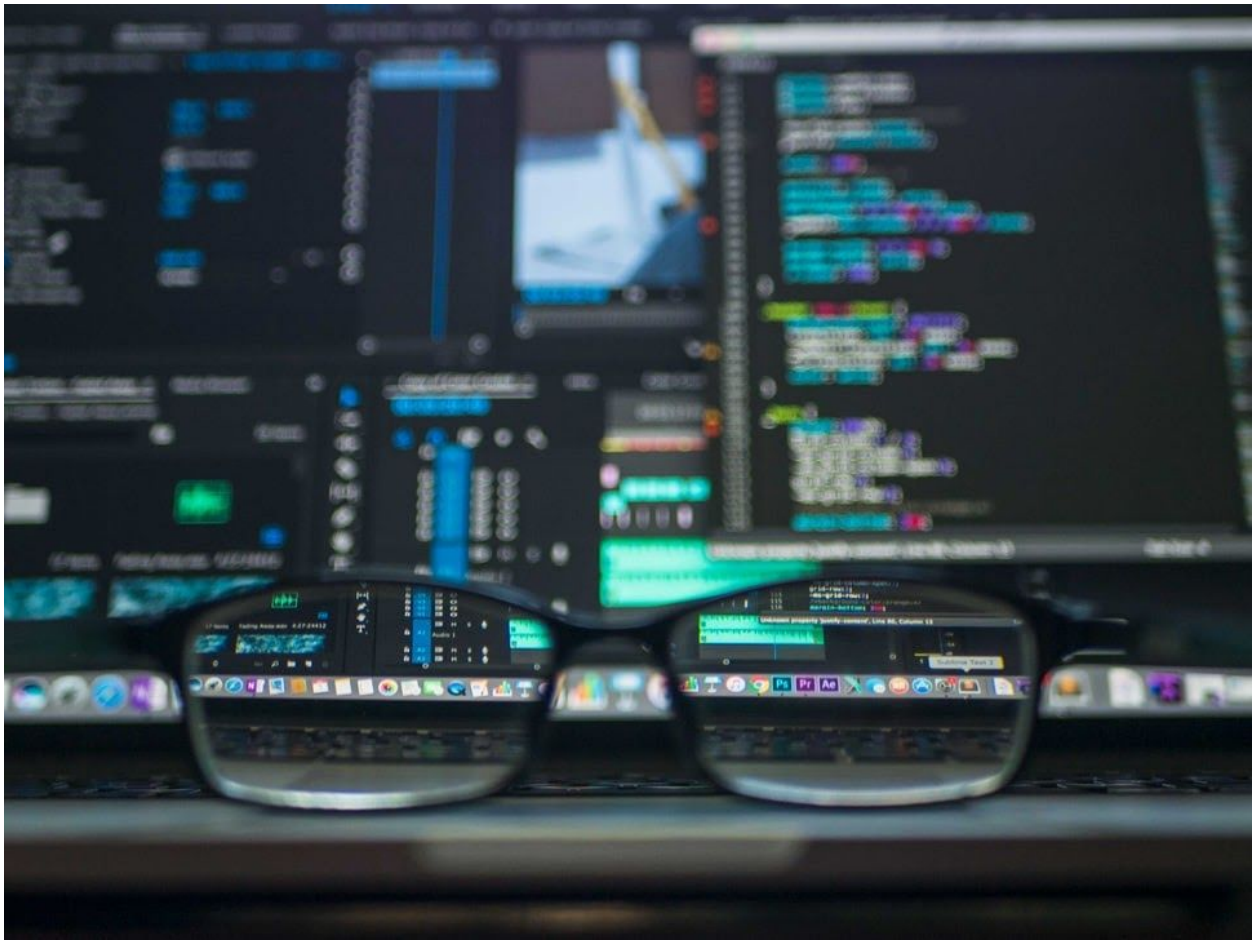
Memoria

Pruebas de Software



Miguel Alberto Lucea Duica

Asignatura “Ampliación de ingeniería del software”.
3.º Ingeniería informática

Pruebas unitarias.

En este apartado se pide crear pruebas unitarias para comprobar que la funcionalidad de los métodos getCellsIfWinner(); y checkDraw(); de la clase“Board” funcionan correctamente, el primero devuelve el número de celdas que contiene la línea en caso de que el jugador pasado como parámetro gane y si no devuelve null y el otro comprueba si existe empate y devuelve un boolean.

DATOS

Métodos	Funcion
InciacionEstadosPrueba()	Se ejecuta antes de cada test, en parte para evitar codigo innecesario, genera una simulación de la partida colocándola en un estado en el cual en pocos movimientos se puede ganar, perder o empatar.
primerJugadorGana()	Se ejecuta un movimiento que hace que el primer jugador gane, luego con los métodos assertThat se comprueba que el método getCellsIfWinner con la entrada J1(Que es el primer jugador en mover), devuelve una lista de longitud 3 que seria la “Linea” con la que el jugador 1 ha conseguido la victoria.Además se comprueba adicionalmente que el método checkDraw() funciona correctamente en los casos que producen una respuesta false.
primerJugadorPierde()	Se ejecutan varios movimientos que hacen que el primer jugador pierda, luego con los métodos assertThat se comprueba que el método getCellsIfWinner con la entrada J1(Que es el primer jugador en mover), devuelve null indicando la derrota del J1.Además se comprueba adicionalmente que el método checkDraw() funciona correctamente en los casos que producen una respuesta false.
empate()	Se genera una situación de empate y se comprueba que el método funciona correctamente para un caso positivo.Se comprueba si la salida es true.

Pruebas con dobles.

Se simulan conexiones a la partida para comprobar los eventos que envía la aplicación a los navegadores web son los esperados.Se sigue un esquema parecido al anterior.

DATOS

Métodos	Funcion
InciacionEstadosPrueba()	Se ejecuta antes de cada test, en parte para evitar codigo innecesario, genera una simulación de la partida colocándola en un estado en el cual en pocos movimientos se puede ganar, perder o empatar. Además se añaden a la partida las conexiones y los jugadores, y se verifica que el evento JOIN_GAME se reciba correctamente en ambos navegadores, uno por cada uno de los jugadores.
primerJugadorGana()	Se ejecuta un movimiento que hace que el primer jugador gane. Acto seguido se comprueba que la conexión 1 recibe el evento GAME_OVER y los valores que pasa son los correctos. En este caso el jugador1 victorioso y la fila de la victoria.
primerJugadorPierde()	Se ejecuta un movimiento que hace que el primer jugador gane. Acto seguido se comprueba que la conexión 2 recibe el evento GAME_OVER y los valores que pasa son los correctos. En este caso el jugador2 victorioso y la fila de la victoria.
empate()	Se genera una situación de empate y se comprueba que el juego envía el evento GAME OVER, pero en este caso no debe existir ningún jugador victorioso, por lo tanto el valor del evento es null.Ademas comprobamos que efectivamente la partida es un empate con el método checkDraw().

Pruebas de sistema de la aplicación.

Estas pruebas son implementadas con Selenium, una herramienta que permite simular en el navegador web deferentes casos de prueba

DATOS

Métodos	Funcion
InciacionEstadosPrueba()	Se ejecuta antes de cada test, en parte para evitar codigo innecesario, genera una simulación de la partida colocándola en un estado en el cual en pocos movimientos se puede ganar, perder o empatar. En este caso además se añaden dos instancias de los navegadores, y se identifican en ellos los elementos con los que se va a interactuar.
primerJugadorGana()	Se ejecuta un movimiento que hace que el primer jugador gane. Acto seguido se comprueba que el mensaje recibido por el navegador a través de un alert es el esperado, en este caso que el jugador1 gana. Se cierran los navegadores.
primerJugadorPierde()	Se ejecuta un movimiento que hace que el primer jugador gane. Acto seguido se comprueba que el mensaje recibido por el navegador a través de un alert es el esperado, en este caso que el jugador2 gana y el j1 pierde . Se cierran los navegadores.
empate()	Se genera una situación de empate y se comprueba que navegador recibe un mensaje de empate. Se cierran los navegadores.