JUNIT

1. Propósito

JUnit es el entorno que hemos utilizado para crear pruebas unitarias que verifiquen de forma automática el programa desarrollado. Estas pruebas están basadas en asertos y han sido realizadas con el fin de obtener una versión lo más correcta posible del juego Scrabble.

La versión de JUnit usada es la correspondiente a JUnit 4 por el simple hecho de tener acceso a más información sobre versiones menos recientes del mismo.

2. Pruebas unitarias

Las pruebas unitarias pueden clasificarse en:

- Pruebas de unidad
- Pruebas de integración
- Pruebas de sistema

Las primeras se centran en probar las operaciones de cada una de las clases. Posteriormente, en las pruebas de integración se comienza la construcción del sistema, probando primero aquellas clases más independientes y luego las restantes. Por último, en las pruebas de sistema se busca validar el comportamiento, es decir, las acciones visibles al usuario y salidas del sistema, en vez de la organización interna del mismo, la cual ya ha sido probada en las pruebas de inferior nivel.

En nuestro proyecto hemos realizado mayoritariamente pruebas de sistema debido a la falta de conocimientos sobre JUnit en fases tempranas del desarrollo de la aplicación, con lo que muchas operaciones simples se han quedado sin probar aunque con las pruebas de sistema nos aseguramos que dichas operaciones son correctas ya que las últimas dependen de las primeras.

Para correr el conjunto de todas las pruebas unitarias ejecutar la clase ***AllTestsSuite.java*** ubicada en /src/Test/AllTestsSuite.java.

Las pruebas unitarias hechas sobre nuestro proyecto son las siguientes:

- Tablero Test

  En la clase *TableroTest.java* se han hecho pruebas sobre la clase *Tablero.java* para verificar la funcionalidad básica del tablero: colocar una ficha, quitar una ficha... También se ha comprobado que las casillas del tablero se encuentran correctamente inicializadas.

- Casilla Test

En la clase *CasillaTest.java* se han hecho pruebas de unidad sobre la clase *Casilla.java* y también se ha verificado que las puntuaciones de las palabras se generan correctamente tanto para las palabras formadas en horizontal como en vertical, teniendo en cuenta las casillas sobre las que están colocadas las letras. Para ver más información sobre la puntuación de las palabras consultar documento con las reglas del juego en la carpeta de Documentos Específicos de la Entrega final.

- Lobby Test

En la clase *LobbyTest.java* se han hecho pruebas sobre la clase *Lobby.java* para verificar el correcto funcionamiento de la clase en la que se registran los jugadores antes de iniciar una partida.

- Generadores Tests

Estas pruebas están destinadas a verificar el funcionamiento de los generadores del mazo que contiene las fichas y del diccionario del juego, el cual es tomado como referencia para verificar las palabras formadas en el tablero. Dichas pruebas se encuentran en *GeneradorMazoTest.java* y *GeneradorDiccionarioTest.java*.

  - Para generar el mazo se debe verificar que se crean y se mezclan las fichas correctamente.
  - Para generar el diccionario se debe verificar que se carga correctamente el diccionario y también se ha probado que las palabras se verifican correctamente a partir de este diccionario.

- Command Tests

Estas pruebas se centran en verificar el correcto funcionamiento de los comandos. En el proyecto hemos implementado los comandos con el patrón Command, que permite encapsular las operaciones concretas de cada uno de los comandos. Por tanto, las pruebas se han hecho generando el comando e invocando a su método *perform().*

Debido a que muchos de los comandos dependen de parámetros externos para poder ejecutar su *perform()* en algunas pruebas se han hecho operaciones en clases externas antes de ejecutarlos.

Por ejemplo, para comprar una ficha comodín se necesitan 5 monedas y para ello es necesario modificar al jugador proporcionándole las monedas antes de ejecutar el ComandoComprarComodin.

Otro caso de dependencia a destacar es el del ComandoQuitarFicha que depende del ComandoColocarFicha. Esto significa que en caso de fallar el segundo, también lo hará el primero y ambas pruebas unitarias tendrán errores.

En este caso, para detectar el verdadero origen del posible fallo se ha probado tanto el *execute()* como el *perform()* del ComandoQuitarFicha por separado.

Los comandos que se han probado son:

- *ComandoColocarFichaTest.class*
- *ComandoQuitarFichaTest.class*
- *ComandoPasarTurnoTest.class*
- *ComandoCambiarFichaTest.class*
- *ComandoSaltarJugadorTest.class*
- *ComandoComprarComodinTest.class*
- *ComandoInvertirSentidoTest.class*

- Turnos Test

En la clase *TurnosTest.java* se han hecho pruebas sobre los turnos debido a unos errores que aparecieron y mediante estas pruebas pudimos corregirlos.

- Fin de Juego Test

Por último cabe mencionar que inicialmente teníamos pensado realizar pruebas para verificar que se cumpliesen todas las reglas del juego, pero finalmente estas pruebas no se han implementado debido a que, a pesar de que el final del juego está claro en las Reglas del juego (mirar documento con las reglas del juego en la carpeta de documentos específicos), en la práctica no hemos implementado todos los casos. Debido a esto se eliminó la clase *FinJuegoTest.java* en la que teníamos pensado llevar a cabo las pruebas. La razón por la cual el final del juego no está bien definido en el código es porque en los últimos sprints antes de finalizar el proyecto tuvimos que realizar numerosas refactorizaciones.