Análisis Dinámico De Pruebas Unitarias

POOB

Proyecto Inicial

Miguel Angel Vanegas Cardenas y Allan Steef Contreras

Ingeniería De Sistemas

2024-2



**Índice**

* **Análisis pruebas unitarias relacionadas con el ciclo 1 \_\_\_\_\_\_\_\_ pág 3**

**ADDTILE.**

**VALIDEOUTOFRANGE.**

**DELETETILE.**

**RELOCATETILE.**

**ADD GLUE.**

**DELETEGLUE.**

**TILT.**

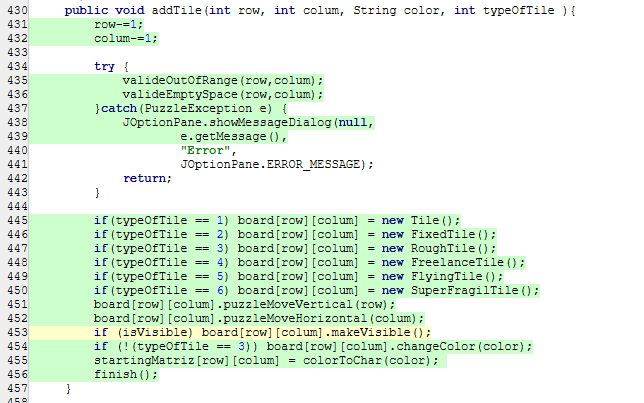
* **Análisis pruebas unitarias relacionadas con el ciclo 2**
* **Análisis pruebas unitarias relacionadas con el ciclo 3**
* **Análisis pruebas unitarias relacionadas con el ciclo 4**
* **Conclusión del cubrimiento total del codigo**

**CICLO 1**

**ANÁLISIS DEL CUBRIMIENTO DEL CÓDIGO:**

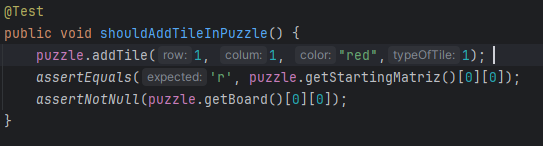
Para el análisis del cubrimiento del código con las pruebas unitarias, se mostrará la imagen del resultado del bloque de código que se deseaba probar, después de correr las pruebas unitarias *with coverage.*

**ADDTILE:**

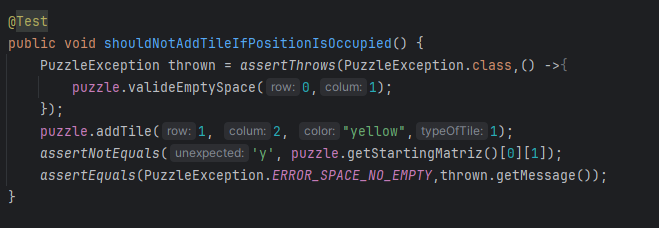
****

Como se puede observar en las pruebas unitarias hubo un buen cubrimiento de este método, aunque teniendo claro que no se está cubriendo todas las posibles ramas, teniendo en cuenta la línea 453, ya que las pruebas se hicieron con el tablero invisible en todo momento, Aun y con lo recién mencionado, se evidencia un cubrimiento de un muy alto porcentaje de este.

Los siguientes son los testeos relacionados con el método.:

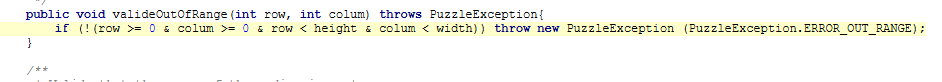
****

El primer test de este método fue diseñado con un tablero ya establecido, añadiendo una nueva baldosa y comprobar que el tablero si se halla actualizado. El diseño buscaba comprobar que siempre y cuando debiera añadirse la baldosa lo hiciera, ya que la posición en la que se desea agregar está dentro del puzzle y está vacía.

****

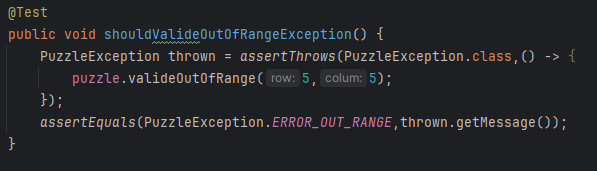
El segundo busca mostrar un caso de fallo, atrapando la excepción demostrando que la casilla ya está ocupada, comprobar que el mensaje que se está soltando es el esperado y por otro lado que la baldosa no se añada. El diseño buscaba demostrar que la excepción se está atrapando y manejando de la manera correcta.

**VALIDEOUTOFRANGE**



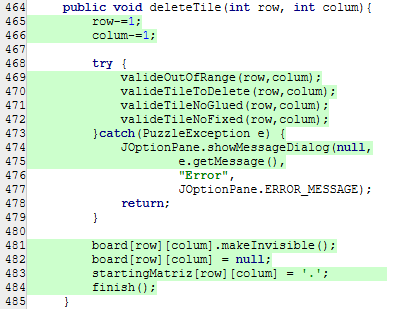
Este método por la prueba y su color, podemos deducir que el testeo no está cubriendo gran parte del código, debido a que está abarcando solo una de las posibles ramas, que es en la que se lanza el error y lo propaga.

El testeo relacionado es el siguiente:



En el test, se usa el método con valores que van a lanzar la excepción, se atrapa y se verifica que el mensaje que se está lanzando sea el esperado. El diseño fue hecho pensando en probar el caso más importante que es cuando el puzzle podría fallar.

**DELETETILE**

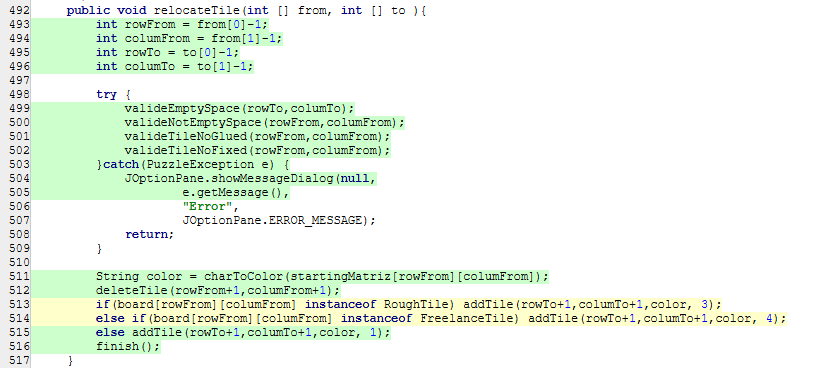
****

Se evidencia un buen cubrimiento del método con las pruebas ya que como se puede observar no hay ninguna línea con color rojo o amarrilo, por lo tanto se puede inferir que se están tratando todas posibles ramas del método.

En el primer test se elimina una de las baldosas, y se comprueba que se realice el cambio en el puzzle. El diseño fue pensando en un caso que no llegará a ninguna excepción y por lo tanto si se elimine la baldosa.

En el segundo test se valida que se esté lanzando una de las excepciones, ya que no se encontrara una baldosa que eliminar, aparte se comprobará que el mensaje sea el correcto. El diseño fue pensando en comprobar que se está atrapando y manejando de manera correcta la excepción.

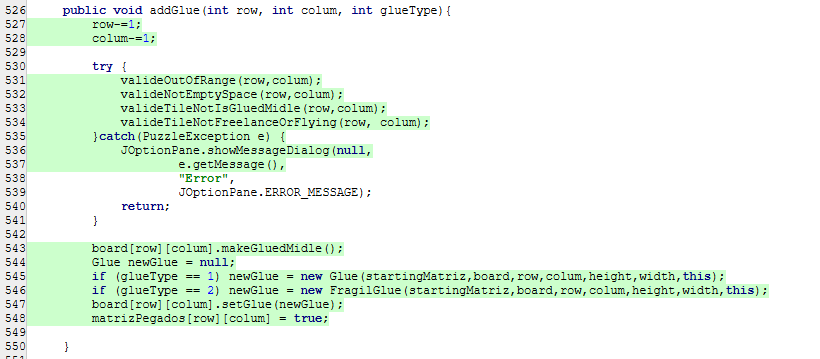
**RELOCATETILE.**

****En este caso las pruebas unitarias están cubriendo una buena proporción del método en cuestión, pero con algunos casos que se quedan fuera, ya que hay dos líneas en amarrillo lo que representa que existen ramas por probar.

El primer test es un caso normal en el que se debe relocalizar la baldosa sin problema alguno. Su diseño es básico, solo demuestra que si las condiciones se cumplen la baldosa se relocaliza sin problema alguno.

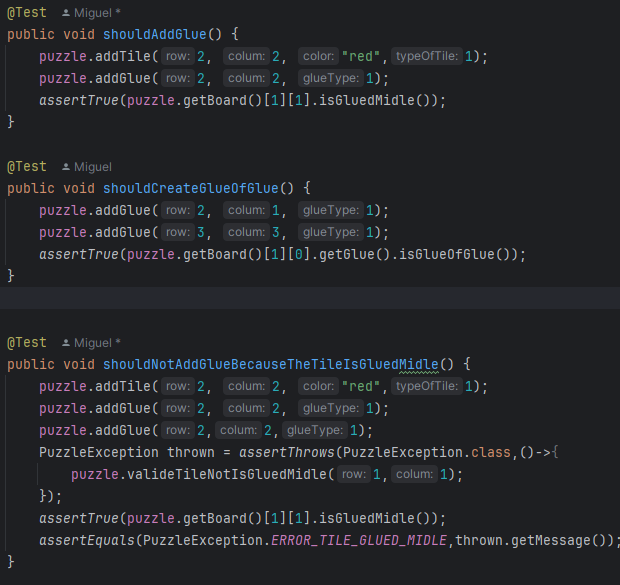
El segundo test comprueba que en caso de que la ficha esté pegada no se podrá relocalizar, se atrapa el error se verifica que es el correcto y se comprueba que no se realizó ningún cambio en el puzzle. El tercer test es bastante parecido pero con otro caso, comprueba el caso de que no existe ficha para relocalizar. El diseño de estos dos testeos se pensó para verificar que se estén trabajando de manera correcta dos de los casos bordes más importantes de este método.

**ADDGLUE**

****

Este método se desarrolló y se probó de una forma bastante correcta, debido a que como se puede observar se está teniendo en cuenta un muy gran porcentaje del mismo. No hay lineas de color rojo ni amarrillo por lo tanto no hay ramas por comprobar ni lineas que se hallan quedado fuera de las pruebas unitarias.

Las pruebas relacionadas son:



El primer test se añade un pegante con las condiciones correspondientes para que no tenga problema alguno. Se diseñó para probar que el caso en el que si se añade el pegante no falle.

El segundo test se comprueba que al añadir un pegante y añadir otro de manera que algunas de las baldosas que pertenezcan a uno de los pegantes, pertenezcan al otro, por lo tanto sería un pegante más grande. Se diseñó para demostrar que se puede añadir más de un pegante incluso si se intersectan.

El tercer test es un caso borde del ADDGLUE ya que no se le puede añadir un pegante a un baldosa que ya se le haya añadido pegante antes. Se diseñó en busca de comprobar que la excepción se esté manejando de la manera correcta.

**DELETEGLUE**