FIUBA - 7510
Técnicas de diseño

Trabajo práctico 2: Unit test
2do cuatrimestre, 2013

Alumnos:

Federico Piechotka, padrón 92126
Mail: fpiechotka@gmail.com

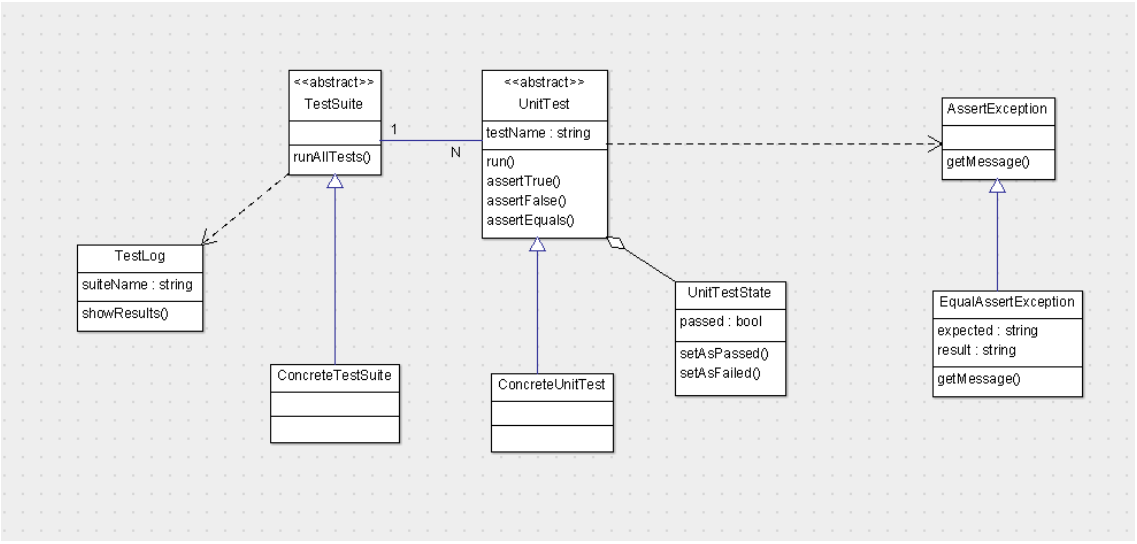
Marco Lotto, padrón 91967
Mail: marcol91@gmail.com

Tomás Reale, padrón 92255
Mail: tomasreale@gmail.com

Introducción

La idea de este trabajo práctico consistió en el diseño e implementación de un framework pensado para correr test unitarios y a partir del resultado de los mismos mostrar un reporte.

Solución planteada



Hacemos una breve descripción de la solución planteada:
Los test unitarios estarán representados por la clase abstract UnitTest. En el método run de esta clase será donde se ejecute cada test, provocando que dicho test pase al estado failed o al estado passed (queriendo decir que el test funcionó correctamente).

Además, los UnitTest proveen distintos asserts para que aquellas clases que hereden de esta clase abstracta puedan utilizar, al igual que en los distintos frameworks conocidos de unit testing.

En caso de que algún assert falle, se lanzará una AssertException o una EqualAssertException (para assertEquals), para poder detectar los fallos en las corridas.

En el TestSuite, es donde tendremos la colección de todos los tests que necesitamos correr, con el objetivo de que cuando se desee ejecutar la corrida se llame al método runAllTests del TestSuite, que a su vez ejecutará cada test que tenga asignado. Luego de eso, mostrará los resultados al usuario por medio del TestLog.

Mediante el TestLog, llevamos a cabo la transmisión de los resultados hacia el usuario que está corriendo los tests. Sirve para que una vez que se hayan ejecutado todos y cada uno de los tests, se le pueda mostrar al usuario los tests que fallaron, los que funcionaron correctamente, y la razón por la que fallaron aquellos que lo hicieron.

Las clases concretas que se ven en el diagrama son las que hereden de TestSuite y de UnitTest, que serán las que al fin y al cabo hagan uso efectivo de estas clases y permitan ejecutar tests concretos para luego conocer el resultado de su ejecución.

La razón por la que TestSuite es una clase abstracta de la cual extiende el usuario del framework, es porque seguramente se requiera agrupar diversos tests para correrlos todos juntos. Ahora muchas veces es conveniente no correr todos los tests juntos sino separarlos en diversas suites, es decir un usuario extiende TestSuite en tantas subclases como el crea conveniente, agrupando en cada uno los tests que el cree que deberían correr juntos.

Observar que cada UnitTest es un único test, a diferencia de por ejemplo jUnit que permite definir en cada TestCase mas de un test unitario.