**Índice**

[Enunciado 2](#_Toc370981405)

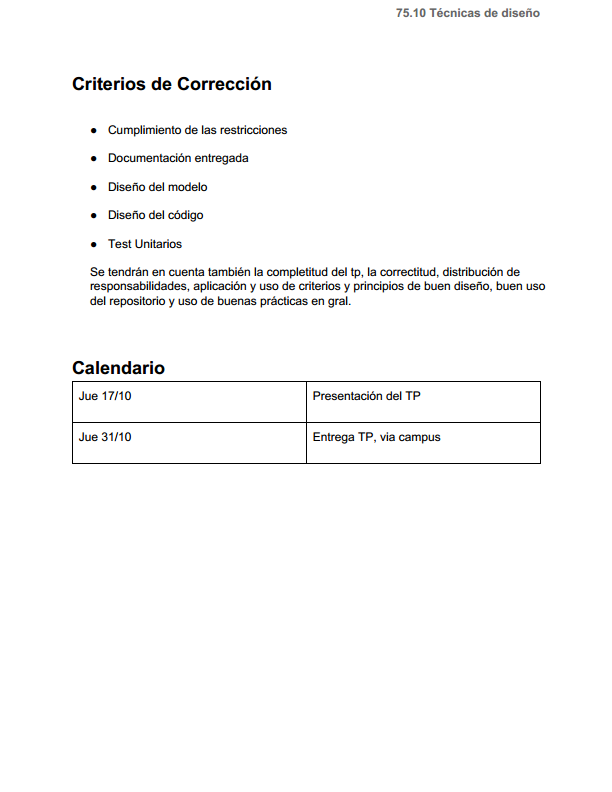
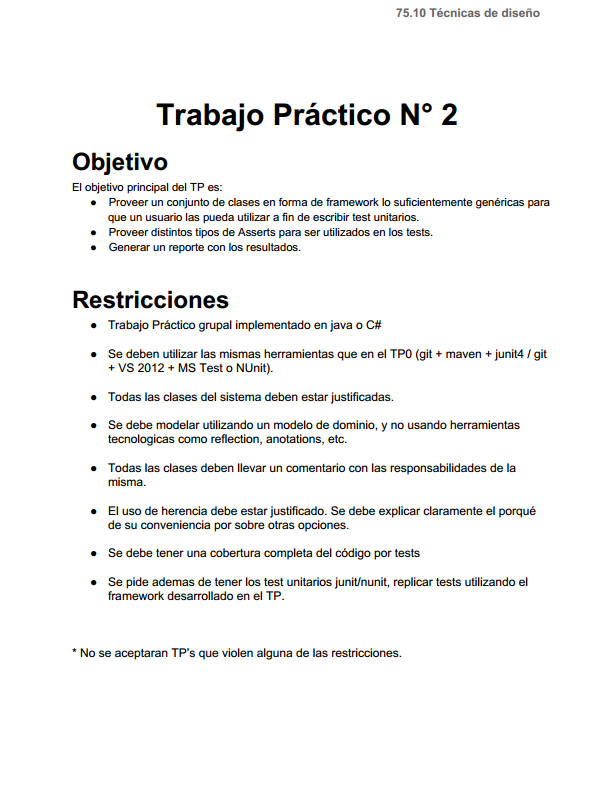
[Desarrollo 4](#_Toc370981406)

[Diagramas 5](#_Toc370981407)

[Diagrama de clases 5](#_Toc370981408)

[Subtítulo 2 6](#_Toc370981409)

Enunciado



Desarrollo

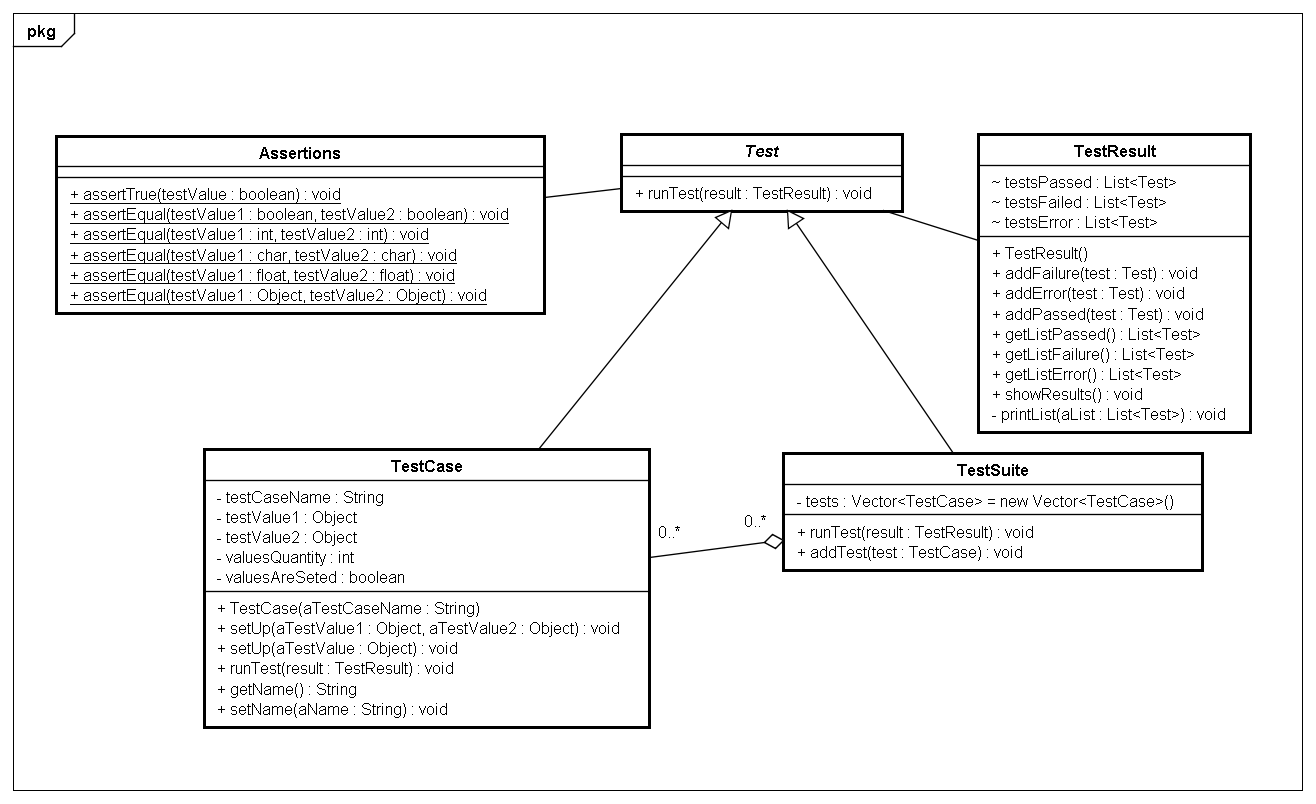
Para llegar a la solución se decidió usar el patrón *Composite*. Test es la clase de la cual heredan el componente y el contenedor. El componente es un test específico, mientras que el contenedor es un grupo de tests. Es el único momento en el que se usa la herencia. Si bien no se utiliza ahora, esto permitirá agrupar distintos tipos de Tests luego. Por ejemplo si quisieran agruparse tests para un producto, podrían agruparse dentro de un contenedor que tenga productos. Luego este formará parte de otro contenedor general que tendrá éste y otros Tests, ya sean simples o compuestos.

**Clases:**

* Test: fue creada para poder aplicar el patrón *Composite*.
* TestCase: cada instancia, cada prueba en particular. Cada Assert debe hacerse utilizando esta clase
* TestSuite: agrupa los distintos tests existentes
* Assertion: es la clase encargada de devolver el resultado de la comparación entre los 2 operandos recibidos. El resultado de la operación lo arroja mediante excepciones definidas por nosotros.
* TestResult: contenedor del resultado de las pruebas. Se pasará un TestResult por parámetro al TestSuite, quien lo irá pasando a todos los Tests que contiene para que lo completen. Posee 3 listas, en las cuales se guardarán los tests pasados, fallados y que dieron error, según corresponda.

*Nuevas Excepciones:*

* ErrorException: excepción que implica que hubo un error al correr el proceso. Actualmente solo se arroja si los parámetros no fueron inicializados.
* OkException: excepción que se utiliza para indicar que la comparación de los operandos da un resultado positivo.
* FailureExcepcion: excepción utilizada para indicar que la comparación de los operandos dio un resultado negativo.

Diagrama de clases