**Índice**

[Enunciado 2](#_Toc372231822)

[Desarrollo 7](#_Toc372231823)

[Hipótesis 7](#_Toc372231824)

[Clases 8](#_Toc372231825)

[Test 8](#_Toc372231826)

[TestCase 8](#_Toc372231827)

[TestSuite 8](#_Toc372231828)

[Assertion 8](#_Toc372231829)

[TestResult 8](#_Toc372231830)

[TestAssertResult 9](#_Toc372231831)

[FailureExcepcion 9](#_Toc372231832)

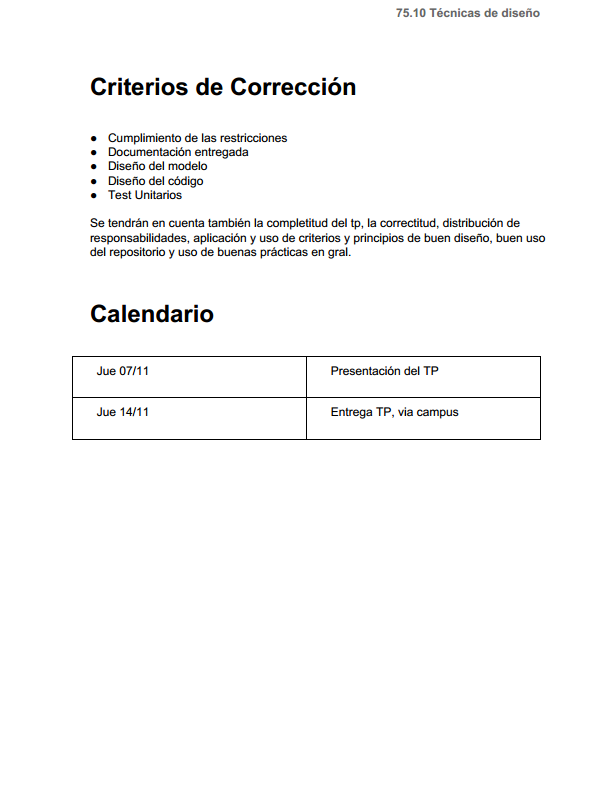
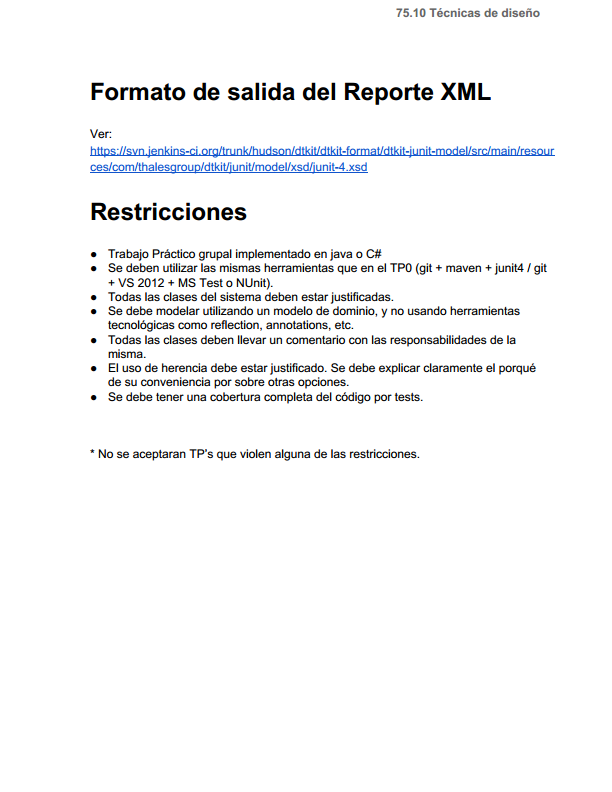
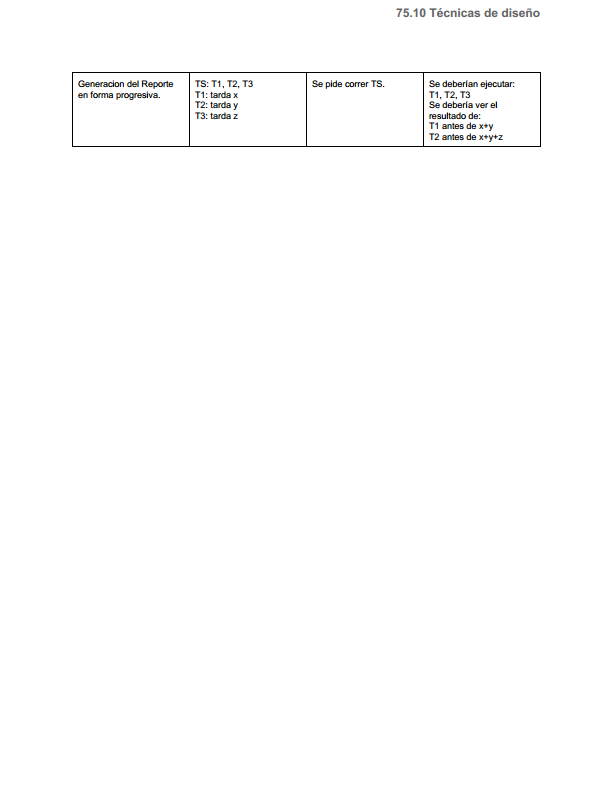
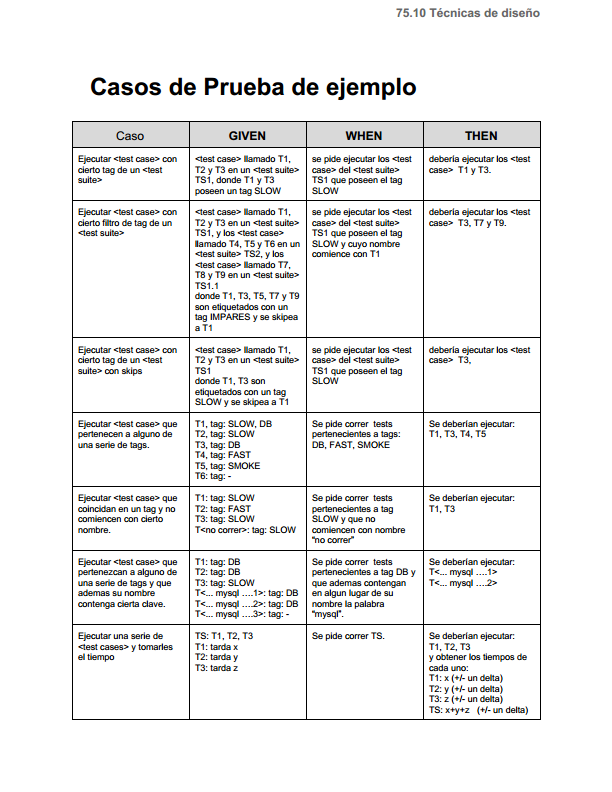
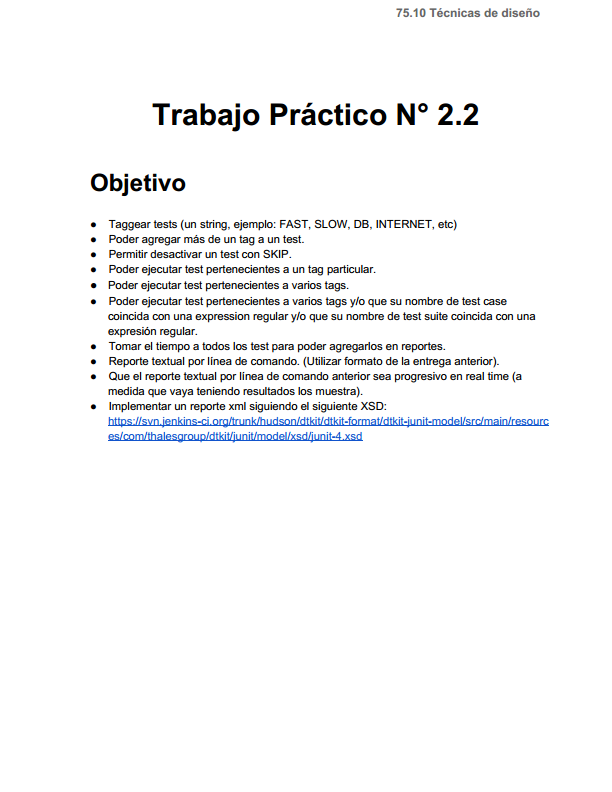
[TestConditions 9](#_Toc372231833)

[TestConditionsBuilder 9](#_Toc372231834)

[Timer 9](#_Toc372231835)

[Diagrama de clases 10](#_Toc372231836)

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.

Hipótesis

* Sólo *TestCase* tiene tags, no *TestSuite*
* *TestCase* y *TestSuite* ambos tienen el booleano “hasToBeSkipped”
* *TestCase* y *TestSuite* ambos tienen “elapsedTime”

Clases

Test

Esta clase abstracta es la clase padre de TestCase y TestSuite. La misma fue creada para poder aplicar el patrón Composite.

TestCase

Clase que representa cada “test individual”, o sea cada prueba en particular.

Para poder ejecutar correctamente "runTest", primero hay que setear los valores a comparar, con "setAssertValue".

TestSuite

Clase que contiene a los tests existentes. Puede contener tanto TestCases como otros TestSuites (grupos de tests).

Posee una familia de métodos *runTest* (con regex, sin regex, se le pasa un TestResult, no se le pasa un TestResult, etc.) para correr todos los TestCase contenidos en ella.

Assertion

Es la clase encargada de devolver el resultado de la comparación entre el/los operando/s recibido/s. El resultado de la operación lo arroja mediante excepciones definidas por nosotros.

Si el *assert* falla, retorna un FailureException. En el caso de que se produzca un error, se lanzara un error acorde a eso (propio de java).

TestResult

Clase que guarda el resultado de todos los TestCase corridos dentro de un TestSuite y otros TestResults de los TestSuite que contiene. Para obtener los valores, se la debe recorrer recursivamente.

Después de ordenar los resultados de los tests, puede mostrar los resultados tanto por consola como a través de un archivo de texto

TestAssertResult

Clase que se utiliza para guardar el resultado de correr un test. Los posibles valores que va a recibir son "Ok", "Fail" o "Error".

FailureExcepcion

Excepción utilizada para indicar que la comparación de los operandos dio un resultado negativo, o sea cuando el *assert* no es satisfactorio.

TestConditions

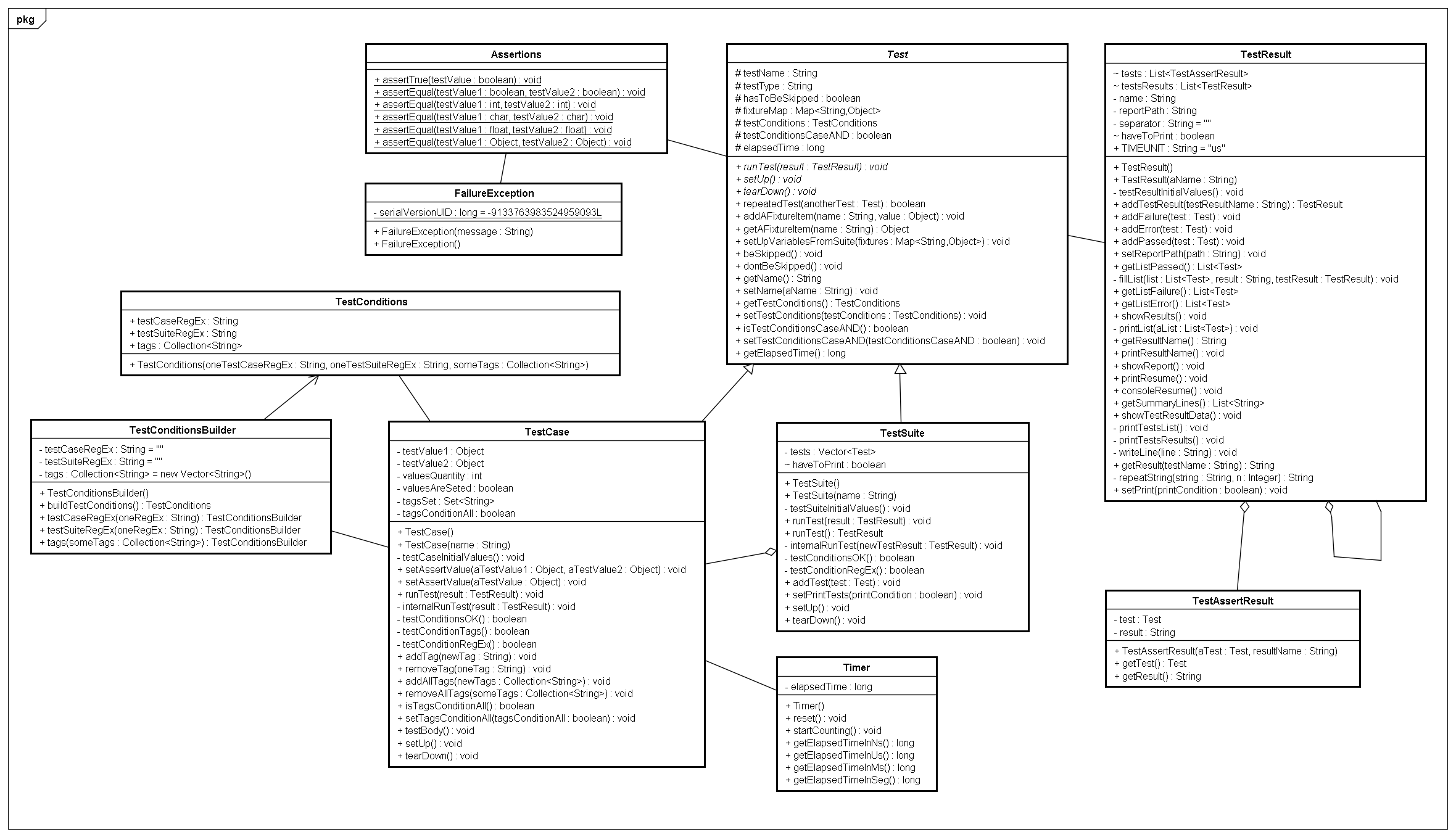
Clase que contiene todas las restricciones a la hora de correr un test (tanto para TestCase como TestSuite), como por ejemplo un RegEx para el nombre de TestCase ó TestSuite, como una serie de tags.

TestConditionsBuilder

Clase que se encarga de construir un *TestConditions*. Tiene valores por defecto de todos los atributos de *TestConditions*, entonces si el usuario no le pasa alguno, no hay problema.

Timer

Clase que se encarga de la lógica de tiempos. TestCase lo usa para contabilizar el tiempo que tarda en ejecutar su función principal (la que checkea el test).

Diagrama de clases