FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico
Técnicas de Diseño 75.10

Entrega Final

Nombre y Apellido	Padron	Mail
Leandro Pellegrino	84441	lrpellegrino@gmail.com
Martin Andres Lucero	89630	luceromartinandres89630@gmail.com
Natalia Merino	85828	nati.merino@gmail.com

CONTENIDO

Enunciado

Trabajo Práctico N° 2.3

Objetivo

- Poder expresar un test de performance que falle si el test tardó más del tiempo especificado.
- Recordar corridas anteriores, para luego poder solo correr los últimos fails/errors + los nuevos (Se debe indicar a la hora de correr los test si es que se quiere correr bajo ese modo).
- Ofrecer al menos dos "stores" posibles para recordar las corridas. Se debe permitir al usuario elegir el que quiera. Plus: ofrecer una manera sencilla de agregar un nuevo

"store" por parte del usuario. (Es decir que sea "pluggable").

Casos de Prueba Ejemplo

Caso	GIVEN	WHEN	THEN
Test de performance fallido.	T1 (sin fallas o errores) que dura X + 1 con límite de X tiempo.	Correr T1	T1 debería ser fallido por tardar más tiempo.
Test de performance exitoso.	T1 (sin fallas o errores) que dura X con límite de X + 1 tiempo.	Correr T1	T1 debería ser exitoso.
Recordar Tests anteriores Con fallidos, errores, y nuevos.	Dado una corrida anterior donde: - T1 fue exitoso. - T2 dio error. - T3 fue fallido. Y: Un T4 nuevo . Todos pertenecientes a un TS.	Correr el TS indicando que solo se quieren correr los fallidos/errores/nuevos.	Se debería solo correr: T2, T3 y T4.
Recordar Tests anteriores Sin fallidos, ni errores, ni nuevos.	Dado una corrida anterior donde: - T1 fue exitoso. - T2 fue exitoso. - T3 fue exitoso. Todos pertenecientes a un TS.	Correr el TS indicando que solo se quieren correr los fallidos/errores/nuevos.	No se debería correr ningun test.

Restricciones

- Se debe trabajar sobre el trabajo práctico de otro grupo.
- Trabajo Práctico grupal implementado en java o C#
- Se deben utilizar las mismas herramientas que en el TPo (git + maven + junit4 / git + VS 2012 + MS Test o NUnit).
- Todas las clases del sistema deben estar justificadas.
- Se debe modelar utilizando un modelo de dominio, y no usando herramientas tecnológicas como reflection, annotations, etc.
- Todas las clases deben llevar un comentario con las responsabilidades de la misma.
- El uso de herencia debe estar justificado. Se debe explicar claramente el porqué de su conveniencia por sobre otras opciones.
- Se debe tener una cobertura completa del código por tests.
- Se debe realizar una crítica del Diseño, Código, Tests y herramientas utilizadas sobre el TP que les ha tocado.
- Se deberá hacer un commit del código recibido de otro grupo en un branch del repositorio propio, y de ahí en todos los cambios se realizarán sobre ese branch.

* No se aceptaran TP's que violen alguna de las restricciones.

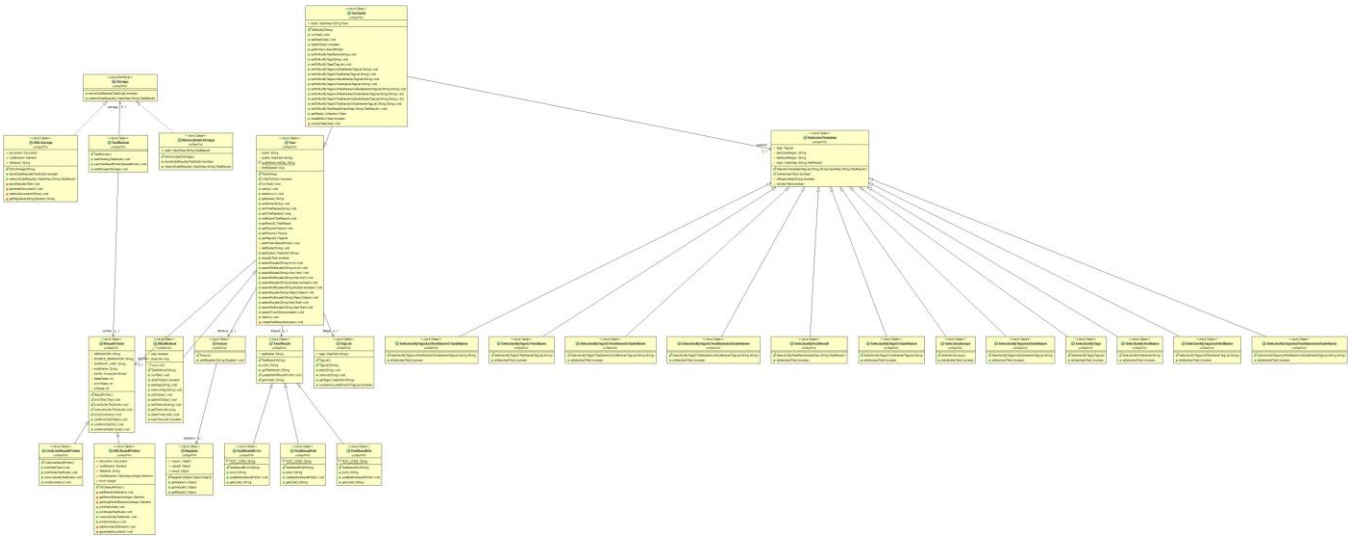
Criterios de Corrección

- Cumplimiento de las restricciones
- Documentación entregada
- Diseño del modelo
- Diseño del código
- Test Unitarios
- Crítica del Diseño, Código, Tests y utilización de herramientas por parte del otro grupo.

Se tendrán en cuenta también la completitud del tp, la correctitud, distribución de responsabilidades, aplicación y uso de criterios y principios de buen diseño, buen uso del repositorio y uso de buenas prácticas en gral.

Diagrama de Clases

Diseño Actual



Informe TP Grupo 13

Resumen

Si podemos ir de lo general a lo mas particular podemos encontrar los puntos fuertes y débiles del Trabajo Practico que mencionamos a continuación.

Si hablamos a nivel de GIT no tuvimos ningún inconveniente a la hora de realizar el branch y trabajar en el código, nos fue dado el acceso a su repositorio de manera inmediata y recibimos un mail en el cual se ofrecieron a responder cualquier inquietud.

A la hora de evaluar la estructura del proyecto notamos el poco uso de paquetes para que pueda ser el código más entendible. El cual debimos aplicar algunos refactors de paquetes para poder trabajar de una manera más cómoda.

Siguiendo con la misma línea cuando empezamos a analizar los nombres de las clases notamos una similitud de nombres de Clase a las de JUnit, lo cual al principio se nos fue un poco difícil distinguir entre ambas.

Ademas la clase Test pareciera ser una " God Class" ya que posee muchos métodos y muchas de las clases implementadas heredan de ella. Esto hace que muchas clases hereden métodos innecesarios para su uso.

Por último, se encontraron muchas clases del tipo "Selection" y pareciera ser que algo puede diseñarse para reducirlas.

Ahora bien, una particularidad no menor, fue encontrarnos que no se utilizo JUnit para probar el Framework, sino que utilizaron el propio Framework para testearlo. Esto puede ser peligroso ya que estamos confiándonos sobre el funcionamiento del mismo.

Cuando vamos a nivel código nos encontramos con un código documentado correctamente y prolijo, mas alla de la presencia de algún code smell en los nombres de atributos.

Con respecto al enunciado se creo una Interfaz Storage la cual se implemento con un HashMap en memoria y un XML en disco y con esto se le da la posibilidad al usuario de implementar su propio Storage.

Finalmente, creemos que el trabajo entregado es bueno, entendible y por sobre todo bien documentado.