Ejercicio TDD

El desarrollo dirigido por pruebas (TDD o Test Driven Development) es una metodología de desarrollo de software mediante la cual se escriben todas las pruebas de un programa, con los resultados esperados, antes de crear el código propio del mismo. Este tipo de desarrollo supone una serie de ventajas en algunos aspectos frente a todo tipo de métodos de desarrollo y testing de programas. Algunas de estas ventajas podrían ser:

* Una mayor calidad del código, al promover un diseño más limpio y modular que puede ayudar a prevenir errores.
* Facilita el mantenimiento del código, puesto que al tener las pruebas realizadas desde el inicio, cualquier fallo posterior en el código se verá rápidamente en las pruebas. Además, la base de pruebas actuará como guía y documentación viva del programa.
* Incrementa la confianza al realizar cambios puesto que al tener una base sólida de pruebas no será un problema cometer errores y podrán revertirse rápidamente, haciendo así que el desarrollo del código sea más efectivo y rápido.
* Fomenta el enfoque en los objetivos por etapas, dividiendo el código en diferentes fases no muy complejas desde un inicio.

El ciclo de TDD implica 3 pasos: Hacer una o varias versiones de una misma prueba que fallen; después se realiza el código mínimo necesario para que la prueba pase y, por último, se refactoriza el código para mejorar su calidad en caso de ser necesario.

Con todo esto, el TDD fomenta la creación de código más limpio, confiable y orientado a los requisitos, al mismo tiempo que acelera la detección temprana de errores. Es una práctica clave en la programación ágil y la entrega continua.

# 

# Ejercicio 1:

Lo primero que debe hacerse en este ejercicio es crear un proyecto de Java y en este crear un método dentro de una clase al que se quieran realizar tests. Una vez creado dicho método, se deja completamente vacío, añadiendo la cantidad de código mínimo como para que el código se pueda ejecutar y los tests se puedan realizar. Para ello se añade un return “” en este caso, de forma que no devuelva ningún texto y posiblemente no coincida el resultado del método con el resultado que va a ofrecer el test, puesto que esa es la primera fase a la hora de hacer un TDD, a esta parte se le conoce como firma. Una vez hecho esto, se crea el test del método, dando clic derecho sobre el nombre del mismo, seleccionando “source action”, “generate tests” y finalmente habilitando los tests de Junit y seleccionando el test que se va a crear dentro de la clase.

Tras esto, se elabora el código del test, dividiendo su contenido en 3 partes: arrange, donde se establece el string al que se le va a dar la vuelta, así como el resultado que debería dar de darsele la vuelta, el resultado esperado. Después va el act, donde se llama al método original en base al valor establecido previamente, y finalmente el assert, donde se comprueba este resultado con el esperado.

Al realizar el test de esta manera y con el método vacío, el test fallará, siendo este el objetivo; aun así se repite el proceso al menos una vez más para asegurarse de que falla exitosamente, ya que en base al valor establecido en el método original, “”, en caso de que en el test se le estuviese dando la vuelta a eso mismo, el test funcionaría. En este caso no sería tan necesario, pero en otros sí.

Tras verificado esto, se procedería a realizar el código del método ahora si correctamente y nuevamente se volverían a hacer varias pruebas para ver si funciona como se espera y el resultado que ha dado el test no ha sido mera coincidencia.

no entra examen:

planificacion de las pruebas (pag 8)

**37 en adelante nada**

pruebas concolicas

herramientas para pruebas

estandares pruebas software

entra:

diferenciar entre no funcional y funcional en los requisitos.

saber diferencair entre verificacion y validacion.

tipos de errores de programacion y breve descripcion

casos de prueba y enfoques

tipos de pruebas (todas)