Pruebas unitarias y TDD

Sesión Conceptual 02

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercicios
- Quiz

- Cierre

Inicio

{desafío}
latam_

Objetivo

- Comprender qué son los dobles en test para utilizarlos en Java.
- Usar Mocks utilizando Mockito para simular métodos.

Desarrollo

{desafío}
latam_

Mockito

Librería para trabajar con Mocks, muy popular en el mundo Java

Características

- Permite escribir pruebas expresivas.
- Ofrece una gama de servicios simples.
- Rodeado de una gran comunidad.

Mock de una Clase

```
package cl.desafiolatam.repositorios;

import static org.mockito.Mockito.mock;

public class RepositorioPersonaTest {

    private RepositorioPersona repositorioPersona = mock(RepositorioPersona.class);

}
```

Crea el objeto simulado de RepositorioPersona con el método estático mock. El cual crea un Mock dada una clase o una interfaz dada.

Uso Mockito

```
@Test
@DisplayName("given crearPersona mocked method when crearPersona invoked then mocked value
returned")
public void testCrearPersona() {
    Persona pepe = new Persona("1-2", "Pepe");
    when(repositorioPersona.crearPersona(pepe)).thenReturn("OK");
    String crearPersonaRes = repositorioPersona.crearPersona(pepe);
    assertEquals("OK", crearPersonaRes);
    verify(repositorioPersona).crearPersona(pepe);
}
```

Uso Mockito

- Se habilita la simulación de los métodos con el método estático **when**.
- Con el método **thenReturn**, se establece un valor de retorno que se devolverá cuando se llame al método.
- Para establecer las excepciones que se lanzan cuando se llama a un método se usa **thenThrow**.
- Se utiliza el método estático **verify**, para verificar que cierto comportamiento ha ocurrido una vez.

Test Driven Development

TDD

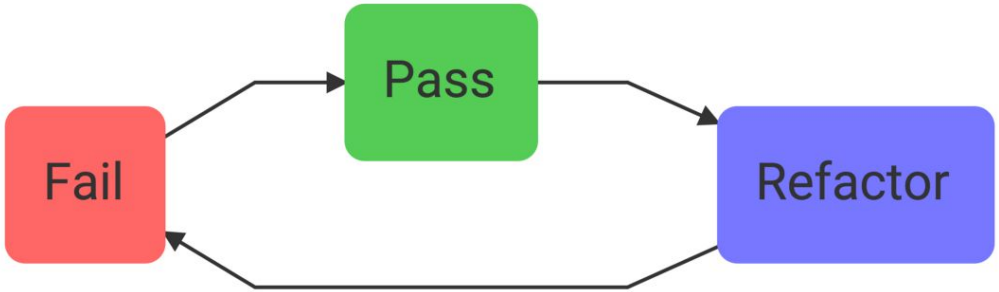
TDD es un procedimiento en donde se escriben

las pruebas antes de la implementación real.

Reglas

- Escribe sólo lo suficiente de una prueba unitaria para fallar.
- Escriba solo el código de producción suficiente para hacer que la prueba unitaria que falló pase.

Ciclo de TDD



Fase Roja

- Lo que se hace es escribir una prueba para que luego se pueda escribir el código de producción, la implementación vienen después.
- En esta fase, debe tomar decisiones sobre cómo se utilizará el código.
- Si se está pensando en cómo implementar el código o cómo va a escribir el código de producción se está haciendo mal.

Fase Verde

¿Por qué no se puede escribir todo el código que se tiene en mente?

- Una tarea sencilla es menos propensa a errores.
- En esta fase se minimizan los errores.

Se debe actuar como un desarrollador que tiene una tarea simple, escribir una solución sencilla que convierta la prueba fallida en una prueba exitosa.

Fase de Refactorización

- Se permite cambiar el código manteniendo las pruebas en verde.
- Es obligatorio eliminar la duplicación de código.

Cierre

{desafío}
latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales