JUnit

2024

JUnit

O JUnit é um dos mais populares frameworks de teste open source para Java, que permite a criação de testes unitários automatizados.

O JUnit 5 é a versão mais recente da ferramenta e trouxe muitas mudanças em relação às versões anteriores:

- Suporte para Java 8 e versões posteriores, com expressões lambda;
- Suporte para testes parametrizados;
- Nova arquitetura que permite extensões personalizadas e suporte a diferentes motores de execução de testes;
- Melhorias de desempenho;
- Suporte para anotações de teste personalizadas;
- Suporte a testes assíncronos.

Revisão – Testes Unitários

Testes unitários são uma prática de engenharia de software que consiste em escrever código automatizado para testar pequenas unidades de código, geralmente funções ou métodos individuais, para garantir que eles funcionem como esperado.
Testes unitários são geralmente escritos por desenvolvedores e executados durante o processo de desenvolvimento de software.

- Ajudam a detectar bugs o mais cedo possível no processo de desenvolvimento.
- Reduzem o tempo de depuração, pois tornam mais fácil identificar e corrigir problemas de código.
- Ajudam a documentar o código e as regras de negócio, pois descrevem explicitamente o comportamento esperado de cada unidade de código → **TDD**.
- Facilitam a manutenção do software, pois permitem alterações “sem medo”.

JUnit – Anatomia de um teste

Anotações (*Annotations*)
São usadas para definir e configurar os métodos de teste. Algumas das mais importantes são:
- @Test
- @BeforeEach / @BeforeAll
- @AfterEach / @AfterAll
- @ParameterizedTest
- @Disabled
- @Timeout

Classe de Teste
Sugere-se sempre usar o mesmo nome da Classe que está sendo testada adicionando o sufixo “Test”

```
public class CalculatorTest {

    @Test
    public void testAddition() {

        Calculator calculator = new Calculator();
        int result = calculator.add(2, 3);

        assertEquals(5, result);
    }
}
```

Método de Teste
Sugere-se usar um nome bastante descritivo sobre o cenário ou comportamento esperado a ser testado.

Lógica do teste
Implementações do código usado para execução do passo a passo do caso de teste.

Asserções (*Assertions*)
Verificam se o resultado da lógica implementada corresponde ao esperado. Algumas das mais importantes são:
- assertEquals() / assertNotEquals()
- assertTrue() / assertFalse()
- assertThrows() / assertDoesNotThrow()
- assertNull() / assertNotNull()

Atenção ⚠

CÓDIGO REPETITIVO

```java
public class CalculadoraTest {

    @Test
    void testeSomar() {

        Calculadora calc = new Calculadora();

        int a = 3;
        int b = 2;
        int resultado = calc.somar(a, b);

        assertEquals(a+b, resultado);
    }

    @Test
    void testeSubtrair() {

        Calculadora calc = new Calculadora();

        int a = 3;
        int b = 2;
        int resultado = calc.subtrair(a, b);

        assertEquals(a-b, resultado);
    }

    @Test
    void testeMultiplicar() {

        Calculadora calc = new Calculadora();

        int a = 3;
        int b = 2;
        int resultado = calc.multiplicar(a, b);

        assertEquals(a*b, resultado);
    }

    @Test
    void testeDividir() {

        Calculadora calc = new Calculadora();

        int a = 3;
        int b = 2;
        int resultado = calc.dividir(a, b);

        assertEquals(a/b, resultado);
    }
```

Anotações

`@BeforeEach`	Indica que o método deve ser executado antes de cada método de teste dentro da classe atual.
`@BeforeAll`	Indica que o método deve ser executado uma única vez antes de todos os métodos de teste dentro da classe atual.
`@AfterEach`	Indica que o método deve ser executado após cada método de teste dentro da classe atual.
`@AfterAll`	Indica que o método deve ser executado antes uma única vez após todos os métodos de teste dentro da classe atual.

Assertions

As Asserções (*Assertions*) são métodos utilitários do JUnit que verificam se uma dada condição ou comportamento do código está de acordo com o que era esperado.
Esses métodos são acessados pela classe `org.junit.jupiter.api.Assertions` no JUnit 5.

- Ordem dos parâmetros:
  <esperado>, <atual>

Assertions - Exemplos

`assertArrayEquals`	Verifica se as matrizes passadas nos parâmetros *expected* e *actual* são iguais.
`assertEquals` `assertNotEquals`	Verifica se os objetos passados nos parâmetros *expected* e *actual* são iguais ou diferentes.
`assertTrue` `assertFalse`	Verifica se dada condição retorna o booleanos Verdadeiro *True*) ou Falso (*False*).
`assertNull` `assertNotNull`	Verifica se um dado objeto é ou não nulo (*null).*
`assertThrows` `assertDoesNotThrow`	Permite verificar se um executável (*executable*) lança uma exceção do tipo especificado, ou não lança nenhuma exceção.
`assertAll`	Permite a criação de asserções agrupadas, onde todas são executadas e suas falhas reportadas em conjunto.
`fail`	Ao ser executado, atribui falha ao teste imediatamente, adicionando mensagem opcional de falha.