DESENVOLVIMENTO ORIENTADO A TESTES E FERRAMENTAS CASE

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

TESTES DE UNIDADE

- Um teste é apenas uma classe Java simples.
- A classe testada geralmente aparece como uma variável de instância da classe Teste.
- Vários métodos de teste, anotados com @Test, usarão as funcionalidades da classe testada e então confrontarão os resultados esperados com os resultados obtidos.

Vamos ver como escrever um teste de unidade com o Junit?

Primeiro, precisamos de um exemplo de classe em teste, ou seja, uma classe implementando alguma funcionalidade que desejamos testar.

O exemplo a seguir é de uma classe calculadora, que retorna a soma entre dois valores inteiros:

Classe Calculator

```
class Calculator {
  int add(int a, int b){
 return a+b;
  }
}
```

Fonte: elaborado pelo autor.

A seguir a classe que testará a classe Calculator:

Classe que testará Calculator

```
class CalculatorTest {
  private Calculator calc = new Calculator();
  @Test
  void testAdd() {
 int x = 2;
 int y = 1;
 int result = calc.add(x, y);
 Assertions.assertEquals(3, result);
  }
}
```

Fonte: elaborado pelo autor.

O funcionamento dessa classe inclui a inicialização de duas variáveis de entrada (x e y) e da chamada do método add, que desejamos testar com a entrada fornecida. O resultado da operação é armazenado em uma variável e, por fim, o resultado esperado é confrontado com o obtido (o valor 3, neste caso).

ANOTAÇÕES JUNIT

Vamos ver agora algumas anotações do JUnit úteis para a organização e para a execução dos procedimentos de teste em classes Java.

@Test

Usada para sinalizar que o método anotado é um método de teste.

@AfterAll

Usada para sinalizar que o método anotado deve ser executado após todos os testes na classe de teste atual.

@AfterEach

para sinalizar que o método anotado deve ser executado após cada método @Test, @RepeatedTest, @ParameterizedTest, @TestFactory e @TestTemplate na classe de teste atual.

@BeforeAll

Usada para sinalizar que o método anotado deve ser executado antes de todos os testes na classe de teste atual.

@BeforeEach

Usada para sinalizar que o método anotado deve ser executado antes de cada método @Test, @RepeatedTest, @ParameterizedTest, @TestFactory e @TestTemplate na classe de teste atual.

@RepeatTest

Usada para sinalizar que o método anotado é um método de modelo de teste que deve ser repetido um número específico de vezes com um nome de exibição configurável.

Para saber mais sobre o conceito e a utilização de anotações em Java, leia o artigo *Entendendo anotações em Java*, de Carlos Araújo (ARAÚJO, 2012).

- ARAÚJO, C. Entendendo anotações em Java. **DevMedia**, [S.l.], 2012.