

Introdução ao JUnit: Garantindo a Qualidade do Código em Java

JUnit é um framework de teste unitário amplamente utilizado no desenvolvimento Java. Ele permite validar o funcionamento correto de pequenos trechos de código, assegurando que o software mantenha alta qualidade e evite bugs em produção.

Faz parte da família xUnit, que conta com ferramentas similares para outras linguagens de programação como NUnit e pytest. Testar seu código ajuda a facilitar a manutenção, garantir a confiabilidade e prevenir falhas causadas por mudanças futuras.



por **Vitor Villa Nova De Oliveira**



Principais Características do JUnit

Anotações Essenciais

- `@Test` para indicar métodos de teste
- `@BeforeEach` e `@AfterEach` para setup e teardown
- `@BeforeAll` e `@AfterAll` para execuções globais
- `@Disabled` para ignorar testes temporariamente

Asserções para Validação

- `assertEquals` para comparação de valores
- `assertTrue/False` para condições booleanas
- `assertNull/NotNull` para objetos
- `assertThrows` para verificar exceções

```

14 gosai (ash: .audites(0000001,(a1/3)(nighent riate cooler)),      741), callertele);
17   ertel fter text: Calleceeravly stor(10);
45   >
16   calin ere commllou)
38   take callution conilogeates : grazly cate = 15);
29   deSteclsection:
22   eetal ine funirion))
35   eastcions (! for fangfarte, calllration (andt ereistly {
48       Tertin: Worlo 5:
33       !
97       colerto Justar 0E9I = 13))
28 | $ Cattercemont( :f Amplitejastz, Srlels, = Appletarn .autiogreth lestmng cornen fraglecthe Jo.12);
30 | <: jetyection:'Jone calucator to year 285));
21     ferpe(0ate);
25     Net linatre!15);
36     Net inte,"vuce seterffo today(lerayices,15);
26     Net inve that is (appfiert1);

```

1

Setup

Antes de cada teste, inicializamos uma instância da calculadora para garantir isolamento.

2

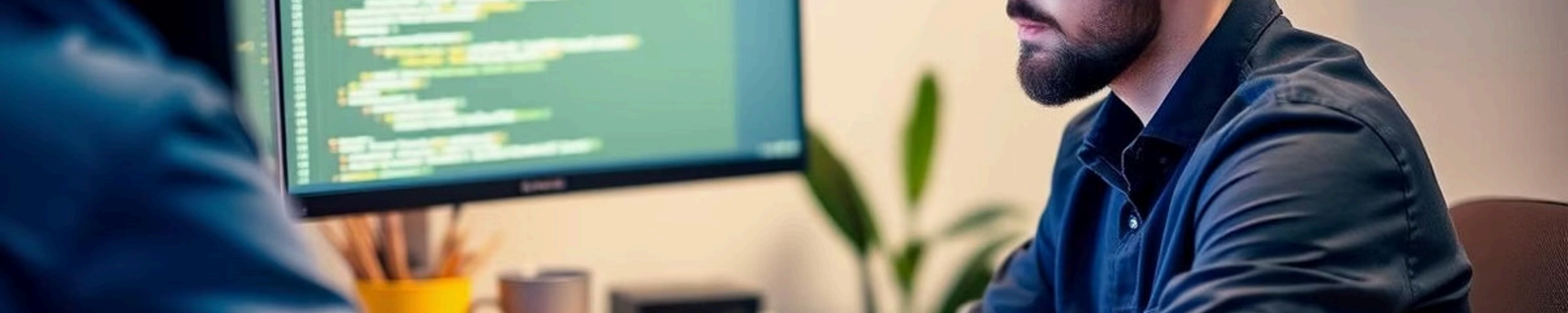
Testes para Operações

- Somar: valida se a soma está correta
- Subtrair: verifica o resultado da subtração
- Multiplicar: checa a multiplicação
- Dividir: comprova a divisão correta




3

Teste de Exceção

Confirma que a divisão por zero lança a exceção esperada `ArithmeticException`.



Por que e Como Escrever Boas Práticas em Testes

-  **Isolamento**
Cada teste deve rodar independente, sem interferir nos resultados de outros.
-  **Nomes Claros**
Utilize nomes descritivos como `testDividir_PorZero_LancaExcecao` para facilitar entendimento.
-  **Cobertura Ideal**
Busque cobertura entre 70%-80%, evitando testes desnecessariamente longos ou fracos.

Frameworks e Ferramentas Relacionadas ao JUnit

Mockito

Permite criar simulações (mocks) para dependências complexas como bancos de dados e APIs externas, facilitando o isolamento dos testes.

Jacoco

Ferramenta para medir a cobertura dos testes, mostrando quais partes do código estão sendo efetivamente testadas.

Diferenças Entre JUnit 4 e JUnit 5

JUnit 4

- Amplamente usado em projetos legados
- Menor suporte a expressões Lambda
- Anotações mais limitadas

JUnit 5

- Suporte avançado a lambdas e streams
- Anotações melhoradas para maior flexibilidade
- Arquitetura modular e extensível

Como Integrar JUnit em Seu Fluxo de Desenvolvimento

Escreva Testes Inicialmente

Comece criando casos de teste para os métodos mais críticos da aplicação.

Use Integração Contínua

Configure pipelines para rodar testes automaticamente a cada alteração do código.

Refatore com Segurança

Confie nos testes para modificar o código, evitando regressões e mantendo qualidade.



Conclusão

Garantir a qualidade do código é essencial para o sucesso de qualquer projeto de software. O JUnit oferece uma estrutura robusta para criar testes eficazes e confiáveis, promovendo boas práticas como isolamento, nomes claros e cobertura adequada. Integrar testes automatizados no fluxo de desenvolvimento melhora a manutenção, reduz erros e aumenta a confiança na entrega de software de qualidade.