Aluno: Igor Profeta Silveira Matrícula: 122215665

Descrição do Caso de Teste - Projeto em Java com JUnit

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

Projeto em **Java** utilizando a biblioteca **JUnit** para realizar testes unitários em uma classe chamada **Calculadora**. Podemos ver a estrutura do projeto contendo os arquivos **Calculadora.java** e **CalculadoraTest.java**, onde o último é responsável pelos testes unitários.

Estrutura do Código:

- **Classe Calculadora (Calculadora.java):**

Esta classe possui quatro métodos básicos para operações matemáticas:

- **soma(int a, int b):** Retorna a soma de dois números inteiros.

- **subtracao(int a, int b):** Retorna a subtração de dois números inteiros.

- **multiplicacao(int a, int b):** Retorna a multiplicação de dois números inteiros.

- **divisao(int a, int b)** e **divisao(double a, double b):** Retornam a divisão de dois números, tanto para inteiros quanto para double.

- **Classe de Testes (CalculadoraTest.java)**:

A classe de testes **CalculadoraTest** utiliza a anotação **” @Test”** do **JUnit** para testar os métodos da classe **Calculadora**. Cada teste verifica se os métodos retornam os valores esperados utilizando o método**” Assertions.assertEquals”**.

Testes Aplicados:

1. somaDoisNumeros():

- Testa o método soma().

- Soma "30" e "40" e espera o resultado "70".

- O teste foi executado com sucesso.

2. subtraiDoisNumeros():

- Testa o método subtracao() da classe **Calculadora**.

- Está subtraindo "40" de "50" e espera o resultado "10".

- O teste foi executado com sucesso, conforme indicado pelo ícone verde.

3. multiplicaDoisNumeros():

- Testa o método multiplicacao().

- Multiplica "10" por "40" e espera o resultado "400".

- O teste foi executado com sucesso.

4. divideDoisNumeros():

- Testa o método divisao() para números inteiros.

- Divide "80" por "40" e espera o resultado "2".

- O teste foi executado com sucesso.

5. divideDoisNumerosD():

- Testa o método divisao() para números do tipo double.

- Divide "20.0" por "40.0" e espera o resultado "0.5".

- O teste foi executado com sucesso.

Resultados dos Testes:

É mostrado que todos os 5 testes foram executados com sucesso e o processo foi finalizado sem erros (indicando **“exit code 0”**), o que confirma que todos os métodos da calculadora estão funcionando corretamente e retornando os valores esperados.

Conclusão:

Este caso de teste demonstra um exemplo clássico de teste unitário onde cada função de uma classe é testada individualmente para garantir que os resultados esperados sejam retornados. Esse processo aumenta a confiabilidade do código, uma vez que garante que cada método da classe ” **Calculadora**” está operando conforme esperado.

Tela de computador

Descrição gerada automaticamente

A classe **Calculadora** foi expandida com um novo método:

**ePar(int x):** Verifica se um número é par. O método retorna **true** se o número for divisível por 2 sem resto (x % 2 == 0).

Além deste método, a classe ainda contém os métodos já mencionados anteriormente para soma, subtração, multiplicação e divisão.

A classe de testes **CalculadoraTest** contém os mesmos testes anteriores e adiciona um novo teste para o método **ePar()**, chamado **verificaPar()**.

**Testes Aplicados:**

1. **somaDoisNumeros()**, **subtraiDoisNumeros()**, **multiplicaDoisNumeros()**, **divideDoisNumeros() e verificaPar()**:

Estes testes continuam sendo executados com sucesso, como indicado pelo ícone verde.

1. **divideDoisNumerosD()**:

Este teste apresenta uma falha, conforme mostrado no painel inferior. O teste está verificando a divisão de dois números do tipo double, onde o resultado esperado era 1.0, mas o resultado atual foi 0.5.

O erro ocorreu ao dividir 20.0 por 40.0. Parece que o valor esperado no teste está incorreto, já que a divisão correta de 20.0 / 40.0 é 0.5, e não 1.0.

1. **verificaPar()**:

Este é o novo teste, que verifica se um número é par utilizando o método ePar(). O teste está verificando o valor 6, e espera que o retorno seja true, o que é correto.

**Erros e Falhas:**

* **divideDoisNumerosD()**: O teste falhou porque o valor esperado (1.0) não corresponde ao valor real (0.5). Isso pode ser corrigido alterando o valor esperado para 0.5 no teste.

**Conclusão:**

* A falha em **divideDoisNumerosD()** pode ser resolvida corrigindo o valor esperado no teste.