



FGA0314 - TESTES DE SOFTWARE (2025.2)

Atividade 4 – TDD

Validador de Expressões Matemáticas

Objetivo:

Aplicar a prática de Test-Driven Development (TDD) desenvolvendo um algoritmo que valida expressões matemáticas simples, aprendendo a evoluir a solução incrementalmente a partir dos testes.

Atividade:

Implemente o método ehExpressaoValida (String expressao) que retorna true se a expressão matemática for válida e false caso contrário.

A expressão pode conter:

- Números inteiros positivos
- Operadores +, -, *, /
- Parênteses (e)

Regras:

- 1. Os parênteses devem estar **corretamente balanceados**.
- 2. Não pode haver dois ou mais operadores seguidos (++, --, +*/, etc.).
- 3. A expressão não pode começar ou terminar com um operador.
- 4. Após um número, só pode vir:
 - o um operador, ou
 - o um parêntese de fechamento)
- 5. Após um operador, só pode vir:
 - o um número, ou
 - o um parêntese de abertura (.
- 6. Espaços devem ser ignorados.

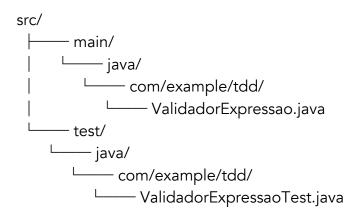
Exemplos de casos de teste:

Entrada	Saída Esperada	Justificativa
"(1 + 2)* 3"	True	Válida, espaços ignorados
"(1+2)*3/"	False	Termina com operador





Estrutura do código sugerida (exemplo Java + JUnit 5):



Instruções adicionais:

- Prática em formato Coding Dojo:
 - Durante a implementação do Gerador de Senhas Fortes, a atividade deverá ser realizada em formato de Coding Dojo, para a prática disciplinada do TDD: apenas uma pessoa por vez estará codificando, enquanto os demais observam, discutem e ajudam nas decisões. Cada membro da equipe codifica um ciclo do TDD.
- O foco é seguir o ciclo do TDD:
 - o Escrever um teste \rightarrow Ver falhar \rightarrow Escrever o código mínimo para passar \rightarrow Refatorar.
 - o Adicionar casos de teste progressivamente, a cada ciclo.
- Dicas para o grupo:
 - Comecem pequeno: testem e implementem regras simples primeiro (ex: deveAceitarExpressaoSimplesValida).
 - o Discutam cada teste antes de escrever: "O que esse teste quer validar?"
 - o Refatore sempre que repetir lógica.

Entrega: a atividade deve ser realizada em equipe e entregue até o final da aula, contendo:

- 1. Relatório descrevendo o desenvolvimento da seguinte forma:
 - 1.1 Linguagem e framework de teste unitário utilizados.
 - 1.2 Para cada ciclo (Red-Green-Refactor) apresentar:
 - Número do ciclo.
 - Red: captura de tela do código de teste implementado no ciclo, com o teste falhado.
 - Green: captura de tela do código da funcionalidade implementada no ciclo, com o teste bem-sucedido.





• Refactor: captura de tela do código refatorado no ciclo, quando for o caso, com o teste bem-sucedido. Pelo menos um dos ciclos deve apresentar refatoração.

1.3 Ao final do relatório, apresentar:

- Código de produção completo.
- Código da suíte de testes completa.
- Captura de tela da IDE com todos os testes da suíte executados.
- Reflexão sobre a prática realizada (o que funcionou bem? o que foi mais difícil?)
- 2. Arquivo da classe implementada.
- 3. Arquivo da classe de teste.