AfroDev

Java Básico - Aula 11 - Parte 2

JUNIT

- Framework para teste automatizado de unidade
- Criado por Kent Beck e Erich Gamma
- Teste quebrou ou não
- Aumenta a confiança no código produzido
- Diminui o custo com re-teste
- Garante a qualidade em *refactories*, correção de bugs e evoluções que geram impactos.
- Escrevendo testes: https://junit.org/junit5/docs/current/user-guide/#writing-tests
-

RED, GREEN, REFACTOR

- Essencialmente você deve ver o teste quebrar (red)
- Você altera o código minimamente para fazer o teste passar
- Você certifica que o teste passou (green)
- Você segue nesse ciclo “sem fim” (refactor)

TDD

- Test Driven Development
- O desenvolvimento de uma nova funcionalidade COMEÇA pelo teste
- Isso garante que somente o necessário será produzido

Vantagens gerais

- Confiança
- Código enxuto
- Melhora o design de APIs (se for difícil de testar, algo está errado com a modelagem ou ela pode ser melhorada)
- Melhora a qualidade geral da entrega, principalmente casos que necessitam de regressão
- “Qualquer um” pode “escrever” teste

Dicas

- Sempre teste algo público
- Sempre teste algo cenário de negócio
- Mock tudo que for externo a unidade em teste
- Ao final, observe a cobertura e avalie se ela é suficiente no seu caso
- Testes de integração e testes de sistemas são melhores executados com outros ferramentas

Observe

- Mockou demais, não vale mais a pena
- Se o teste for mais caro que o próprio código, pense melhor
- Não persiga a cobertura 100%

Demo

- Melhor ser no Eclipse...
- Calculadora (soma) e JUNIT -> live coding
- Calculado (outras operações) e TDD + Red-Green -> live coding

Discussão

- Você acha que 100% de cobertura é sinal de qualidade?
- Testes de stress devem ser feitos com JUNIT?
- Testes de timeout devem ser feitos com JUNIT?
- Códigos que funcionam, mas não tem cobertura de testes, podem ser alterados?
- O cliente pode escrever um JUNIT com você?
- Tenho certeza que o teste vai quebrar, preciso passar pela fase “red” do ciclo red-green-refactor?

Revisão

- Conhecemos o JUNIT, framework de teste de unidade automizado para JAVA
- Vimos que red-green-refactor é uma mecânica simples, mas que ainda gera resistência em sua adoção.
- TDD inverte a ordem “natural” do desenvolvimento colocando o teste a frente da produção de código