



Residência em Tecnologia da Informação e Comunicação JUNIT

Professor:
Alvaro Degas Coelho



INSTITUIÇÃO EXECUTORA



COORDENADORA



APOIO

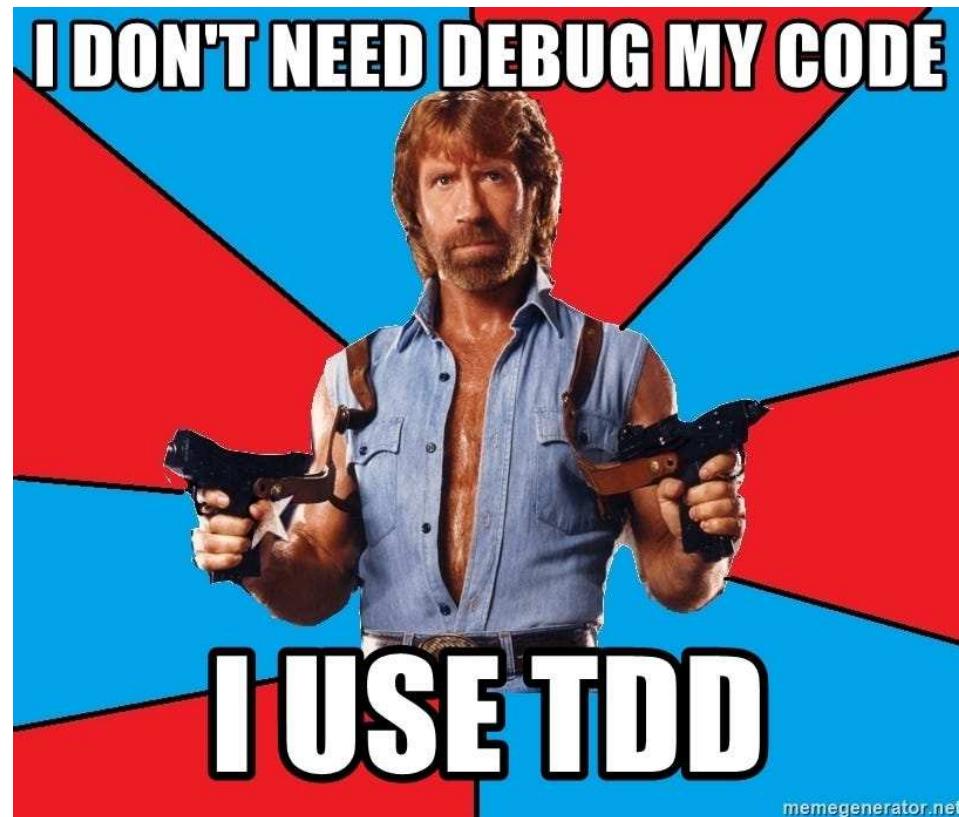




Residência
em Software

O que é JUNIT

- Framework para testes
 - E TDD
- Java
- Python?
 - PyUnit
- .NET
 - Nunit (“filho” do Junit)
- Ruby
 - RSpec



Mais que testes de Unidade

- Teste de Integração
 - Interações entre diferentes módulos. Fluxo de execução de uma funcionalidade específica
 - Exemplo: Fluxo de finalizar um pedido em um site de comércio eletrônico junto com o pagamento
- Teste funcional
 - Verifica se algum recurso funciona corretamente. Normalmente se associa a casos de uso.
 - Exemplo: a funcionalidade de o usuário logar, pesquisar produtos, adicionar ao carrinho e finalizar a compra (com pagamento)

Mais que testes de Unidade

- Teste de Aceitação
 - Verificar se um certo requisito está plenamente atendido junto ao cliente
 - Exemplo: o processo de venda de um produto deve registrar a venda, verificar o pagamento, alterar o estoque, contactar fornecedores (marketplace), confirmar venda. E TRATAR DE TODAS AS POSSÍVEIS EXCEÇÕES DO CASO
- Teste de Sistema
 - Teste completo de todas as funcionalidades (teste funcional), de todas as unidades (unitários) e da integração de todos os componentes (integração)

Mais que testes de Unidade

- Testes específicos
- Performance: avaliar se as respostas são chegadas com custo razoável de recursos
 - Teste de carga: aumentar a quantidade de usuários, de operações, de acesso, etc
 - Teste de estresse: verificar se o sistema consegue se recuperar de situações de estresse (normalmente ultrapassando a carga esperada)
- Testes regressivos
 - Para verificar a existência (ou localizar) de bugs após evolução



JUNIT: Testes de Unidade em Java

- Teste automático
 - Um programa testa um programa
- Testar uma unidade de código
- Aplicável a qualquer linguagem de programação
- Em linguagens OO
 - Teste de métodos. O mais comum. Testa a funcionalidade de um método a partir de contextos (parâmetros e ambiente) diversos
 - Teste de classe. Verifica se classes e subclasses funcionam adequadamente. Herança, interfaces
 - Teste de Instância (objeto). Verifica se as propriedades de um objeto atendem ao que é especificado para ele dado um contexto específico

Exemplos

- Teste de métodos.
 - Verificar se um método calcula corretamente o total de uma compra considerando os descontos dados ao cliente por conta de seu perfil
- Teste de classe.
 - Verificar se a implementação de um objeto de uma classe de colaborador atende a uma determinada interface (se efetivamente implementa os métodos)
- Teste de Instância (objeto).
 - Verificar se a criação de um Estudante estrangeiro com direito a auxílio permanência e documentação específica está sendo feita corretamente



Tipos de Teste JUNIT

- Tudo em Java é orientado a objeto. Logo, testes também são.
 - Classes implementam a funcionalidade de testar
- Basicamente há dois tipos de testes
- Test Case (Caso de teste)
 - Idealmente um para cada classe do sistema (mas não há nenhum impedimento em contrário)
 - Programador (ou ele e um Testador): projeta, escreve e executa
- Suíte de testes
 - Vários (todos?) casos de teste são executados
 - Validação final (quase que um teste de sistema)
 - Tipicamente: soma de todos os testes de unidade

O que se deve testar?

- Escolha difícil – envolve um exercício de antecipação
 - Principalmente se fica por conta do programador
- O que conta para uma boa escolha?
 - Experiência (20%): a experiência acumulada permite antecipar potenciais situações de erro na tarefa de codificar
 - Criatividade (20%): pensar em situações inesperadas, complexas ou até improváveis (fora da caixinha)
 - Pessimismo (80% - sim, passou de 100%): nunca subestime a lei de Murphy.
 - Se algo pode dar errado de algum jeito, vai dar (teste!)
 - Se algo pode dar errado de várias formas diferentes, vai dar errado da pior delas (teste!)

Dicas gerais

- Sem perder de vista o que está no slide anterior!
- Comece pelos testes mais simples e vá testando situações complexas no final
 - Primeiro teste a funcionalidade geral. Depois as funcionalidades específicas, os casos limites e as situações de exceção
- Tente cobrir todas as situações possíveis de erro mas não extrapole
 - Se 3 tipos de caso são suficientes, não precisa usar 10
- Não teste métodos triviais get e set
 - Exceto se eles envolverem mais que a mera atribuição/recuperação de atributos.
 - Por exemplo, um `getTodosOsItensJaComprados()` da classe `Cliente` deve ser testado. Um `setGalaxia()` de um sistema de astronomia também merece teste

Pense como testador

- Quanto mais se testa mais se aprende
- Um exercício
- Jogo “Quadrado Mágico”
 - Vinha num brinquedo antigo dos anos 80: Merlin!

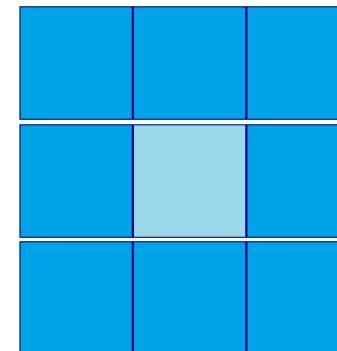
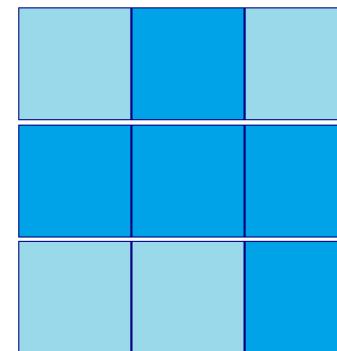




Residência
em Software

Ideia do Quadrado Mágico

- Formar uma seqüência com todas as células “ligadas”, exceto a do meio
- Iniciando a partir de uma distribuição qualquer
- Clicando em qualquer das células para mudar o estado





- Objetivo: produzir todo mundo X exceto o centro, que ficará com O
- Usando o teclado numérico para selecionar o “clique”
 - Cada número entre 1 e 9 corresponde a uma célula do jogo

Usando interface de texto e teclado numérico



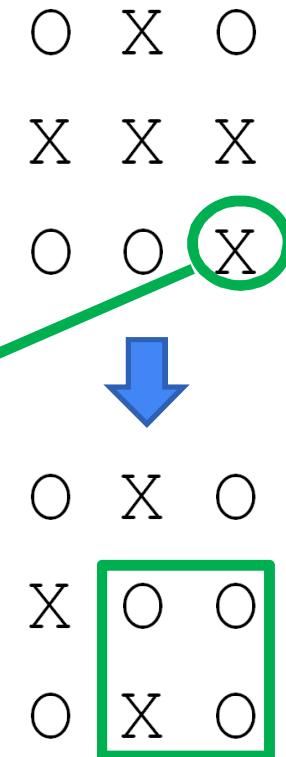
O	X	O
X	X	X
O	O	X
X	X	X
X	O	X
X	X	X





Regras

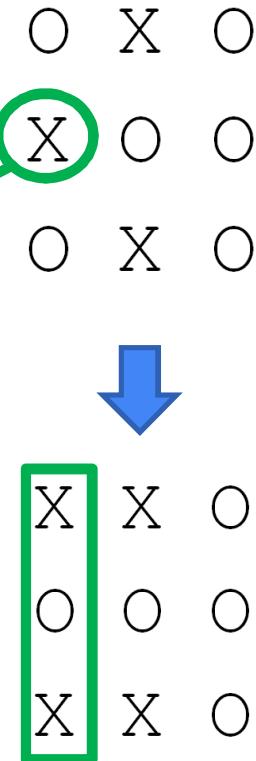
- “Clicar” numa célula de canto (números 1, 3, 7 ou 9)
 - Inverte a própria célula e todas as células adjacentes (horizontal, vertical e diagonal)
- Exemplo:
 - Clicar no 3 inverte as células correspondentes ao 3, 2, 5 e 6





Regras

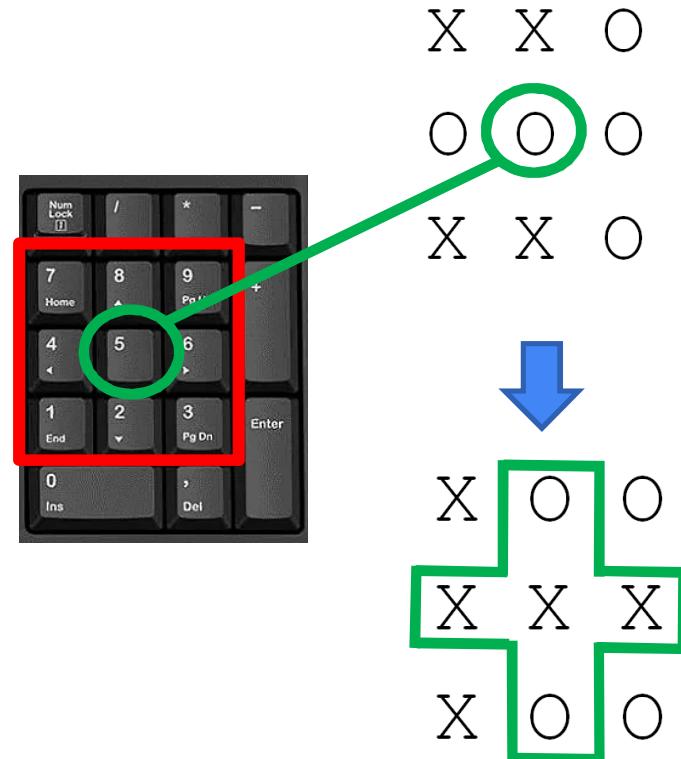
- “Clicar” numa célula de linha (números 2, 4, 6 ou 8)
 - Inverte a própria célula e todas as células adjacentes na mesma linha (horizontal ou vertical)
- Exemplo:
 - Clicar no 4 inverte as células correspondentes ao 1, 4, e 7





Regras

- “Clicar” na célula do meio (número 5)
 - Inverte a própria célula e todas as células adjacentes na mesma linha ou coluna (horizontal ou vertical): uma cruz
- Exemplo:
 - Clicar no 5 inverte as células correspondentes ao 2, 4, 6, 8 e 5





Residência
em Software

- Há uma classe Quadrado
 - Implementa o jogo
- Há um objeto quadrado do tipo Quadrado
 - Tem o estado momentâneo de todas as células (X ou O)
- Há um método clique(int k)
 - Recebe um número e muda o estado do jogo
- Há um método getEstado()
 - Retorna uma String de 9 caracteres (O ou X) descrevendo o estado do jogo
 - Já implementado!
- Há um método setEstado(String)
 - Atribui às células os valores descritos pelos caracteres (O ou X) da String que vai como parâmetro
 - Já Implementado e testado!

Seu trabalho: testador



O X O
X X X
O O X

índice	0	1	2	3	4	5	6	7	8
Caracter	O	O	X	X	X	X	O	X	O
Corresp.	1	2	3	4	5	6	7	8	9

Preparando o ambiente

- Projeto: QuadradoMagico
- Package: quadrado
- Classe: Quadrado



alamy

Image ID: 2jGD76T
www.alamy.com

Definindo casos de teste para o método clique(int k)

- Quais casos de teste você vê como relevantes?
 - Lembre de começar pelos mais triviais
- Como vai testar cada um deles?
 - Instâncias, objetos, etc.
- Quais situações limite você enxerga que são importantes de testar?
 - Não se preocupe se você ver algumas a mais ou a menos que o colega
- Construiremos uma classe de teste (o código atual da classe Quadrado segue ao lado)
- Vamos fazer o TDD

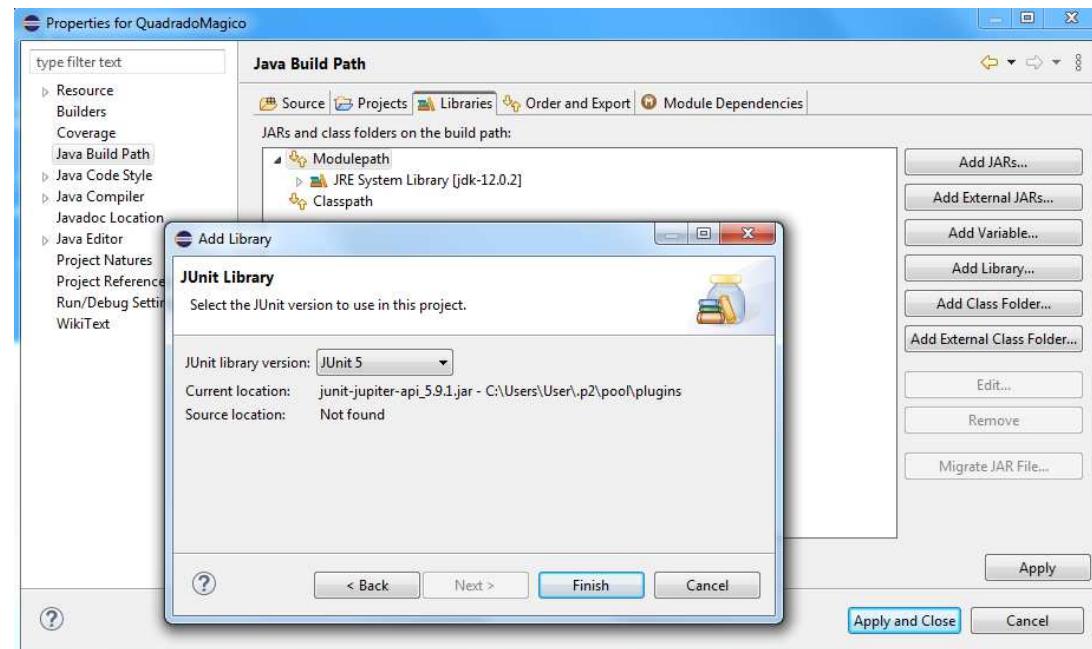
```
package quadrado;
public class Quadrado {
    private String estado;
    public String getEstado() {
        return estado;
    }
    public void setEstado(String estado) {
        this.estado = estado;
    }
    public Quadrado(String estado) {
        super();
        this.estado = estado;
    }
    public void clique(int k) {
        //falta implementar
    }
}
```



Residência
em Software

Criando a classe de teste

- Importar a library Junit
 - Projeto -> Properties -> Java Build Path
 - ModulePath -> Add Library -> Junit (next)
Junit 5 (finish)
 - Apply and Close



Crie sua classe de teste

- O que você precisa saber
 - O que será testado
 - Como
- O JUNIT dispõe de uma série de métodos para facilitar o trabalho de teste de unidade
- Os mais comumente usados são
 - assertEquals(), assertFalse(), assertNotNull(), assertNotSame(), assertNull(),
assertSame(), assertTrue(), fail()

assertEquals()

- Compara dois valores de igualdade. O teste passa (verde) se os valores são iguais.
- Valores, não endereços
- Padrão: o primeiro parâmetro é o valor esperado e o segundo parâmetro é o valor retornado
- Exemplo
 - assertEquals(3628800, calculadora.fatorial(10));
- assertEquals() é o teste inverso

assertTrue()

- Avalia uma expressão booleana. O teste passa (verde) se a expressão é verdadeira.
- Exemplo
 - `assertTrue(calculadora.EhPrimo(2305843009213693951));`
- `assertFalse()` é o teste inverso



assertNotNull()

- Compara uma referência de objeto nula. O teste passa (verde) se a referência não é nula.
- Exemplo
 - `assertNotNull(calculadora.getListaDivisores(2305843009213693951));`
- `assertNull()` é o teste inverso

assertSame()

- Compara o endereço de memória de duas referências de objeto usando o operador `==`. O teste passa se ambos se referem ao mesmo objeto.
 - Lembre-se que, em Java, dois objetos são iguais se eles efetivamente referem ao mesmo dado na memória
 - Exemplo: `assertSame(aluno, listaEstudantes.encontraPorMatricula("9011154-x"));`
- `assertNotSame()` é o teste inverso

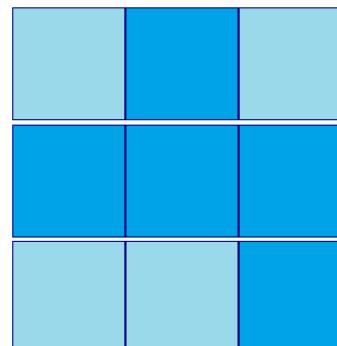
Fail()

- Faz com que o teste atual falhe. Isto é comumente usado com a manipulação de exceção.
 - Atualmente se propõe usar Assertions.assertThrows()
- Exemplo

```
try {  
    int x = calculadora.fatorial(-1);  
}  
catch (Exception e) {  
    if (e instanceof MinhaExcecao) {  
        assertEquals("Fatorial de negativo não existe", e.getMessage());  
    }  
    else  
        fail("Gerou exception errada!");  
}
```

Começando a classe de teste

- Supondo um estado inicial de um objeto Quadrado
 - Podemos usar o do exemplo



- Que estará representado internamente pela String “OOXXXXOXO”

Programando o teste

- Instancie um objeto **quadrado** do tipo **Quadrado** com o estado “OOXXXXOXO”
- Execute o método `clique()` passando o 1 como parâmetro.
 - Qual deverá ser o novo estado?
 - `assertEquals("...", quadrado.getEstado())`
 - Programe seu método para ele responder adequadmente
- Execute o método `clique()` com outras opções
 - Verifique sempre o novo estado

Casos limites

- Podemos pensar em casos limites
- Exemplo 1: um número fora da faixa de intervalo entre 1 e 9
 - Deverá gerar uma exception
- Exemplo 2: caso o jogo esteja terminado, não deve mais admitir nenhum novo clique
 - Deverá gerar uma exception

Continue o desenvolvimento (com TDD)

- Um método boolean venceu()
 - Retorna true se o jogador alcançou seu estado final
 - False caso contrário
- Um método String mostrarEstado()
 - Retorna uma String com O e X correspondentes
 - Pulando linha
 - Representa visualmente o estado do jogo
- Um método Quadrado iniciar() e Quadrado iniciar(String estado)
 - Instancia um objeto do tipo quadrado com um estado inicial aleatório ou um estado inicial definido



Residência
em Software

Crie seu jogo agora

- Com todos os métodos funcionando você pode criar uma fachada para iniciar, decorrer e encerrar o jogo do Quadrado Mágico



Residência
em Software

**Voltaremos para o projeto do jogo da
Senha**