Universidade de Brasília Faculdade UnB Gama

FGA 0238 - Testes de Software – Turma:	<turma>	Semestre:	<2024.1>
Nome:	Marcos Castilhos	Matrícula:	<221008300>
Equipe:	Equipe 6 - TLE		

- Um resumo dos passos realizados no vídeo para a construção da calculadora de imposto de renda;
    - **JUnit**
    - **Criação de pacotes:**
        - **-> tst: onde ficam os testes**
            - Teste para cadastro de rendimentos
                - @Before -> prepara o objeto de teste
                    - -> setup -> irpf = new IRPF();
            - **@Test**
                - teste cadastrar salário 1
                    - irpf.cadastrarSalario(5000)
                    - assertEquals(5000, irpf.getTotalSalario(), 0)
                    - teste passa por técnica de falsificação
                - teste cadastrar salário 2
                    - irpf.cadastrarSalario(6000)
                    - assertEquals(6000, irpf.getTotalSalario(), 0)
                    - os dois passam com getTotalSalario retornando salário
                - teste cadastrar salário 3
                    - irpf.cadastrarSalario(5000)
                    - irpf.cadastrarSalario(6000)
                    - assertEquals(11000, irpf.getTotalSalario(), 0)
                        - não funcionou na primeira implementação pois retornou só o último salário cadastrado, não a soma dos dois

- alterou-se a função cadastrar salário para acumular os salários cadastrados -> os três testes passaram

- **-> app: aplicação**
  - Classe IRPF
    - CadastrarSalário()
    - float getTotalSalario()
      - retorna 5000 -> técnica de falsificação
      - retorna salário
- **backlog.txt**
  1. Cadastrar rendimentos
  2. Calcular total rendimentos
  3. Cadastrar deduções
     a. Previdência social
     b. Dependentes
     c. Pensão Alimentícia
     d. Outras
  4. Calcular base de cálculo
  5. Calcular imposto de renda por faixas
  6. calcular alíquota efetiva

- **Resumo**:
  - foi setado um valor hardcode para passar o teste por **falsificação**
  - foi **duplicado** o teste para usar novos parâmetros e foi observado que o segundo teste não passou
    - obteve-se a barra verde mexendo na unidade que estava sendo testada - getTotalSalario -> barra verde nos dois testes
  - **triangulação ->** se pelo menos três testes estão passando é possível ter um grau maior de certeza se a unidade está implementada corretamente.

- Um resumo da técnica TDD;

O Desenvolvimento Orientado a Testes (TDD) é uma abordagem de desenvolvimento de software que enfatiza escrever testes antes de implementar o código. O processo segue um ciclo de três etapas: vermelho (escrever um teste que falha), verde (fazer o teste passar escrevendo o código mínimo necessário) e refatorar (melhorar o código sem alterar seu comportamento). TDD ajuda os desenvolvedores a entender melhor os requisitos, a escrever código mais enxuto e a ter um controle maior sobre o ritmo de desenvolvimento.
Embora o TDD seja uma prática valiosa para garantir a qualidade do software e facilitar a manutenção no longo prazo, é importante lembrar que não é necessário utilizá-lo em 100% do tempo. Os desenvolvedores podem escolher quando aplicar o TDD com base na complexidade do problema e na necessidade de experimentação e aprendizado durante o desenvolvimento.

- Um resumo sobre os tipos de Mock (dublês de teste).

Existem diferentes tipos de dublês de teste, cada um com sua função específica para facilitar o processo de teste de software:

1. **Stubs (Atraentes):** Retornam valores fixos para simular o comportamento de dependências reais.

2. **Mocks (Simuladores):** Além de retornar valores predefinidos, registram as interações com o objeto sendo testado, permitindo verificar se métodos específicos foram chamados e quantas vezes.

3. **Spies (Espiões):** Observam o comportamento de uma dependência real, registrando todas as interações com o objeto espiado, sem simular o objeto em si.

4. **Fake Objects (Objetos Falsos):** São implementações simplificadas de componentes reais, usadas para simular funcionalidades complexas, como um banco de dados em memória.

5. **Dummy Objects (Objetos Falsos):** São passados para o componente em teste, mas não são utilizados durante a execução do teste, sendo úteis para preencher parâmetros sem relevância para o teste.