Atividade Prática de Desenvolvimento Orientado a Testes (TDD) com Java *(em Dupla)*

Aplicar os conceitos de Desenvolvimento Orientado a Testes (TDD) para implementar um sistema de cadastro de produtos em Java, garantindo que todas as funcionalidades e validações especificadas estejam de acordo com os requisitos. Esta atividade tem como objetivo reforçar a importância do TDD na qualidade e confiabilidade de software, além de aprimorar as habilidades de programação orientada a objetos.

Contexto da Atividade: Vocês foram contratados para desenvolver um sistema de gerenciamento de produtos para uma loja online. Este sistema será responsável por cadastrar usuário e produtos com atributos essenciais e garantir que todos os dados sejam consistentes.

1. **Para o Cadastro de Usuário, o nome da classe pode ser Cadastro.java tal que:**
   - Possa adicionar, remover e atualizar(a senha) um usuário
     - Métodos: adicionar(Usuario), remover(Usuario), atualizar (String, String)
   - Esse usuário deve ter login e uma senha
     - Duas Strings
   - A senha não deve ter menos que 4 caracteres
   - A senha não pode ser igual ao login
   - A senha não pode ter espaços em branco no início nem no final.
   - Caso a senha fornecida não seja válida, lançar um IllegalArgumentException
   - Não posso ter usuários repedidos (com mesmo login)
   - Deve poder me retornar a lista/estrutura de todos os usuários inseridos
     - Método getAll() que retorna o vector, array ou lista

2. **Para o Cadastro de Produto, o nome da classe pode ser ProdutoCadastro.java tal que:**
   - Permitir a adição de novos produtos ao sistema através de um método adicionar(Produto produto).
   - Validações específicas devem ser realizadas no momento da adição para garantir a consistência dos dados.
   - Remover produtos existentes do sistema através de um método remover(Produto produto).
   - Permitir a atualização dos dados de um produto já existente (nome, descrição e preço) com um método atualizar(String codigo, String novoNome, String novaDescricao, double novoPreco).
   - Retornar todos os produtos cadastrados no sistema através do método getAll(), que deverá retornar uma lista de todos os produtos.
   - Ao adicionar ou atualizar produtos, o sistema deverá validar as seguintes regras:
     - **Código Único**: Cada produto deve ter um código único. Caso haja tentativa de adicionar um produto com código já existente, lançar uma IllegalArgumentException.
     - **Nome Obrigatório**: O nome do produto não pode ser vazio ou nulo. Caso o nome esteja em branco, lançar uma IllegalArgumentException.
     - **Descrição com Limite de Caracteres**: A descrição do produto não pode ultrapassar 200 caracteres. Caso a descrição exceda o limite, lançar uma IllegalArgumentException.

- **Preço Positivo**: O preço do produto deve ser maior que zero. Caso o preço seja igual ou menor que zero, lançar uma IllegalArgumentException.

Orientações para Desenvolvimento com TDD

1. **Planejamento dos Casos de Teste**
   Antes de implementar o código, defina os casos de teste para cada funcionalidade e validação. Para cada regra de validação, simule um cenário onde a regra é violada e verifique se a exceção correta é lançada.
2. **Implementação dos Testes**
   Implemente os testes usando JUnit, cobrindo os cenários de sucesso e de erro para cada funcionalidade e validação.
3. **Implementação do Código de Produção**
   Implemente a classe ProdutoCadastro e Produto, sempre seguindo o ciclo de TDD (escrever o teste, implementar, refatorar).
4. **Relatório de Testes**
   Prepare um relatório contendo a descrição dos casos de teste, o código de teste e o código da aplicação.

Estrutura do Relatório de Testes

O relatório deverá conter as seguintes seções:

1. Introdução

- Breve descrição do objetivo do sistema e do exercício.
- Importância do TDD no desenvolvimento de software e como ele foi aplicado nesta atividade.

2. Descrição dos Casos de Teste

Para cada funcionalidade, descreva os casos de teste detalhadamente, incluindo:

- **Nome do Caso de Teste**: Ex.: "Teste Adicionar Produto com Código Existente"
- **Objetivo**: Explicação do que o teste verifica, ex.: "Verificar se o sistema impede a adição de um produto com código duplicado".
- **Entrada**: Valores de entrada utilizados no teste.
- **Procedimento**: Passos executados no teste.
- **Resultado Esperado**: Descrição do resultado esperado (ex.: "O sistema deve lançar uma IllegalArgumentException com a mensagem 'Produto com este código já existe'").
- **Resultado Obtido**: Descrição do que ocorreu ao executar o teste (este campo será preenchido após a execução dos testes).

3. Código dos Testes Unitários

Inclua o código dos testes unitários em Java (pode usar o JUnit). Estruture os códigos de teste para cada caso descrito na seção anterior.

4. Código da Implementação

Inclua o código completo das classes com explicações resumidas sobre cada método, especialmente as validações e manipulações de dados.

5. Execução dos Testes e Resultados

- Descrição geral do processo de execução dos testes.
- Capture e insira no relatório os logs de execução dos testes (capturas de tela ou transcrição dos resultados no console).
- Resumo de quantos testes passaram e falharam. Se algum teste falhar, explique o motivo da falha e como foi corrigido (ou os ajustes planejados, se ainda não foram realizados).

6. Conclusão

- Resumo dos resultados obtidos e da qualidade do sistema implementado com base nos testes.
- Observações sobre a experiência em usar TDD, dificuldades encontradas e aprendizados adquiridos com a atividade.
- Considerações finais sobre a importância do TDD no desenvolvimento de software e como essa prática contribui para a confiabilidade e manutenção do sistema.

Entrega
O aluno deve entregar um arquivo .zip contendo:
- **Relatório de Testes** em formato .pdf ou .docx.
- **Código-fonte** das classes e dos testes unitários.

Critérios de Avaliação
A avaliação será baseada nos seguintes aspectos:
- **Completeness**: Implementação completa de todas as funcionalidades e validações requeridas.
- **Aderência ao TDD**: Evidências de que o desenvolvimento seguiu o ciclo de TDD (primeiro os testes, depois o código).
- **Clareza e Organização do Relatório**: Estrutura e detalhamento dos casos de teste, clareza das descrições e organização dos resultados.
- **Qualidade do Código e Testes**: Código bem estruturado, claro, com uso adequado de boas práticas de programação e cobertura adequada dos testes unitários.

O TRABALHO PODERÁ SER FEITO EM DUPLA