Disciplina: Qualidade de Software

2025.2 - BSI

Projeto de Testes de Software: Aplicando TDD na Camada de Negócio

1. Contexto do Problema

Uma empresa deseja criar um sistema simples de pontuação de clientes com base nas compras realizadas. O sistema precisa calcular e gerenciar os pontos de fidelidade dos clientes, seguindo regras de negócio que podem evoluir no futuro. O foco do projeto é permitir que os alunos pratiquem TDD (Test-Driven Development), garantindo qualidade na implementação de métodos da camada de negócio, sem necessidade de banco de dados ou interface gráfica.

2. Objetivo do Projeto

Os alunos deverão:
- Aplicar TDD, criando testes antes da implementação.
- Implementar testes unitários focados na lógica de negócio.
- Desenvolver métodos de domínio coerentes e testáveis.
- Manipular estruturas de dados em memória, como listas de clientes.
- Compreender a importância da manutenção da qualidade e refatoração segura baseada em testes automatizados.

3. Descrição do Sistema

O sistema deve gerenciar clientes e seus pontos de fidelidade. Cada cliente acumula pontos com base em compras, podendo receber bônus ou ter regras específicas.

Regras básicas:
1. Cada R$1 gasto gera 1 ponto para cliente padrão.
2. Clientes Premium recebem 1,5 ponto por real gasto.
3. Clientes VIP recebem 2 pontos por real gasto.
4. Funcionalidades do sistema:
- Registrar uma compra e atualizar pontos.
- Consultar total de pontos de um cliente.
- Resgatar pontos para desconto (1 ponto = R$0,05).
- Operar sobre listas de clientes: adicionar, filtrar, ordenar e remover.

4. Escopo Técnico

- Sem interface gráfica.
- Sem banco de dados; todos os dados ficam em memória.
- Linguagem livre (Python, Java, C#, etc.) com biblioteca de testes (pytest, JUnit, NUnit, Jest).
- Estruturas de dados: listas, objetos simples.

5. Etapas do Desenvolvimento (Aplicando TDD)

1. Escrever o teste primeiro (ex: calcular pontos de um cliente padrão).
2. Executar o teste (falha inicial).
3. Implementar o código mínimo para o teste passar.
4. Refatorar mantendo os testes verdes.
5. Repetir o ciclo para cada nova regra de negócio, incluindo operações sobre listas.

6. Exemplos de Testes a Criar

Nº	Nome do Teste (assinatura do método)	Objetivo / Cenário
1	test_calcular_pontos_compra_cliente_padrao()	Verificar se o cliente padrão recebe 1 ponto por real gasto.
2	test_calcular_pontos_cliente_premium()	Confirmar que clientes Premium recebem 1,5 ponto por real gasto.
3	test_calcular_pontos_cliente_vip()	Validar que clientes VIP recebem 2 pontos por real gasto.
4	test_acumular_pontos_varias_compras()	Testar o acúmulo de pontos em várias compras consecutivas.
5	test_consultar_pontos_cliente_existente()	Verificar se a consulta retorna o total correto de pontos.
6	test_resgatar_pontos_para_desconto()	Garantir que o resgate de pontos gere o desconto correto.
7	test_impedir_resgate_com_saldo_insuficiente()	Certificar que o cliente não possa resgatar mais pontos do que possui.
8	test_resgatar_todos_os_pontos_disponiveis()	Validar que o sistema permita resgatar todo o saldo disponível.
9	test_nao_gerar_pontos_para_valor_zero()	Assegurar que compras de valor zero não gerem pontos.
10	test_gerar_pontos_para_valores_decimais()	Confirmar que valores decimais geram pontos

		proporcionais.
11	test_nao_permitir_pontos_negativos()	Garantir que o saldo de pontos nunca seja negativo.
12	test_cliente_inexistente_lanca_excecao()	Verificar se o sistema lança erro ao consultar cliente inexistente.
13	test_registrar_novo_cliente_com_pontos_iniciais()	Validar o cadastro de um cliente com pontos de boas-vindas.
14	test_aplicar_bonus_promocional_em_compra()	Testar aplicação de bônus promocional sobre compras.
15	test_expirar_pontos_antigos_apos_periodo()	Simular expiração de pontos antigos após período determinado.
16	test_registrar_varios_clientes_em_lista()	Validar a inserção de múltiplos clientes em uma lista.
17	test_calcular_pontos_lista_clientes()	Calcular pontos para todos os clientes de uma lista.
18	test_filtrar_clientes_com_pontos_acima_de_limite()	Filtrar clientes cujo saldo de pontos é superior a determinado valor.
19	test_ordenar_clientes_por_pontos()	Ordenar clientes conforme o total de pontos acumulados.
20	test_remover_clientes_com_saldo_zero()	Remover da lista os clientes que possuem saldo de pontos igual a zero.
21	test_buscar_cliente_por_nome()	Pesquisar cliente pelo nome em uma lista de clientes.
22	test_somar_total_pontos_lista()	Calcular o total de pontos de todos os

		clientes da lista.
23	test_ranking_clientes_por_pontos()	Gerar ranking dos clientes ordenado por pontuação decrescente.

7. Entregáveis

1. Código-fonte completo da aplicação.
2. Conjunto de testes unitários (cobrindo regras de negócio e operações em listas).
3. Relatório breve (máx. 4 páginas) contendo:
- Estratégia de TDD utilizada.
- Casos de teste criados.
- Cobertura de código alcançada.

Obs: O trabalho, em dupla, tem valor 6,0 na Unidade I e a entrega deve ocorrer até 14 de dezembro, 23h59 via Classroom. A partir desta data, em cada semana posterior de atraso na entrega, há uma penalização de 2 pontos. Em 15 e 16 de dezembro ocorrerão as apresentações do projeto em sala para o professor e QA.

8. Critérios de Avaliação

Aplicação correta do TDD – 30%
Qualidade e clareza dos testes – 25%
Coerência da lógica de negócio – 25%
Estrutura e organização do código – 10%
Relatório e documentação – 10%

9. Extensões Opcionais (para alunos avançados)

- Adicionar novas categorias de clientes (Bronze, Prata, Ouro).
- Criar regras de expiração de pontos mais complexas.
- Desenvolver operações adicionais sobre listas (ranking dinâmico, busca por intervalo de pontos, merge de listas).

10. Linguagens/Bibliotecas Sugeridas

Python → pytest, unittest
Java → JUnit 5
C# → xUnit, NUnit
JavaScript → Jest