🧠 Exercício de Fixação 5 — Módulo 1 (POO com Classes Simples)

🎯 Objetivos

- Aprofundar conceitos avançados de Scrum e metodologias ágeis
- Explorar a estrutura de Classes em C# (POO básica, sem herança ou polimorfismo)
- Desenvolver lógica e design de software com desafios práticos envolvendo CRUD simples com classes
- Estimular argumentação técnica com questões dissertativas mais elaboradas

> 💡 **Nota mínima para aprovação: 8,5 pontos (85%)**

📘 PARTE 1 — Exercícios Teóricos

1.1 — Questões Objetivas: Scrum (0,25 ponto cada)

1. Em Scrum, como o Product Owner deve gerenciar o backlog para maximizar o valor entregue?

- [ ] A. Priorizando requisitos técnicos complexos
- [ ] B. Atualizando o backlog somente no início do projeto
- [ ] C. Refinando continuamente o backlog com feedback das partes interessadas
- [ ] D. Deixando o time de desenvolvimento decidir as prioridades

2. Qual o principal desafio ao aplicar Scrum em equipes distribuídas geograficamente?

- [ ] A. Falta de documentação
- [ ] B. Comunicação e sincronização entre os membros
- [ ] C. Falta de reuniões presenciais obrigatórias
- [ ] D. Estimativas incorretas

3. O que é um "Definition of Done" (DoD) em Scrum?

- [ ] A. Lista de requisitos do produto
- [ ] B. Critérios que definem quando uma tarefa está concluída
- [ ] C. Documento que o cliente assina

- [ ] D. Plano de teste automatizado

4. Qual das opções abaixo é um benefício direto do uso da retrospectiva de sprint?

- [ ] A. Definir o próximo backlog
- [ ] B. Identificar e implementar melhorias contínuas no processo
- [ ] C. Avaliar a performance individual dos membros
- [ ] D. Planejar a documentação final do projeto

5. Qual é a principal função do Scrum Master em um conflito de prioridades entre PO e Dev Team?

- [ ] A. Tomar decisão final sobre prioridades
- [ ] B. Mediar a discussão para facilitar entendimento e alinhamento
- [ ] C. Ignorar e deixar que o PO decida sozinho
- [ ] D. Delegar ao gerente de projeto

6. Em que momento o Scrum Master deve interferir durante o Daily Scrum?

- [ ] A. Para corrigir o trabalho dos desenvolvedores
- [ ] B. Para controlar o tempo da reunião
- [ ] C. Quando houver impedimentos que precisam ser removidos
- [ ] D. Para apresentar novos requisitos

1.2 — Questões Objetivas: C# e Classes (0,25 ponto cada)

7. Em C#, qual o modificador de acesso que permite que membros sejam acessíveis apenas dentro da mesma classe?

- [ ] A. `public`
- [ ] B. `private`
- [ ] C. `protected`
- [ ] D. `internal`

8. Qual é a finalidade principal de uma classe em C#?

- [ ] A. Armazenar dados temporários durante a execução do programa
- [ ] B. Definir um tipo personalizado que agrupa dados e comportamentos relacionados
- [ ] C. Executar operações matemáticas complexas
- [ ] D. Gerenciar conexões com banco de dados

9. Como declarar uma propriedade automática chamada `Nome` do tipo string dentro de uma classe?

- [ ] A. `public string Nome { get; set; }`
- [ ] B. `private string Nome;`
- [ ] C. `string Nome = "";`
- [ ] D. `public Nome string;`

10. Qual a sintaxe correta para instanciar um objeto da classe `Cliente`?

- [ ] A. `Cliente c = new Cliente();`
- [ ] B. `Cliente c = Cliente();`
- [ ] C. `Cliente c = Cliente.new();`
- [ ] D. `new Cliente c();`

11. Como acessar e alterar o valor da propriedade `Email` de um objeto `cliente`?

- [ ] A. `cliente.Email = "teste@exemplo.com";`
- [ ] B. `cliente->Email = "teste@exemplo.com";`
- [ ] C. `cliente.SetEmail("teste@exemplo.com");`
- [ ] D. `cliente[Email] = "teste@exemplo.com";`

12. Qual a finalidade de um método dentro de uma classe?

- [ ] A. Armazenar dados
- [ ] B. Executar ações ou comportamentos relacionados ao objeto
- [ ] C. Criar variáveis locais
- [ ] D. Definir um construtor

1.3 — Questões Dissertativas (1 ponto cada)

13. Explique o conceito de classe em programação orientada a objetos e como as propriedades e métodos estão relacionados a ela.

14. Explique a importância da transparência e da comunicação eficaz dentro de um time Scrum para o sucesso do projeto.

🛠 PARTE 2 — Desafios Práticos em C# (POO: Classes Simples + CRUD)

🔹 Exercício 1 — Classe Tarefa com Gerenciamento Básico (1,0 ponto)

Implemente uma classe `Tarefa` com as propriedades:

- `Id` (int)
- `Descricao` (string)
- `Concluida` (bool)

Dentro da classe `Tarefa`, crie uma lista estática para armazenar as tarefas e os métodos estáticos:

- `Adicionar(Tarefa tarefa)` - adiciona uma nova tarefa
- `MarcarConcluida(int id)` - marca a tarefa como concluída pelo id
- `ListarTodas()` - lista todas as tarefas

No método `Main`, demonstre:

- Adicionar 3 tarefas
- Marcar uma tarefa como concluída
- Listar todas as tarefas com indicação se estão concluídas ou não

---

🔹 Exercício 2 — Classe `Funcionario` com Métodos (1,0 ponto)

Crie uma classe `Funcionario` com:

- Propriedades públicas: `Nome` (string), `Salario` (double)
- Método público `AumentarSalario(double percentual)` que atualiza o salário com o percentual informado
- Método público `ExibirDados()` que imprime o nome e salário formatado

No `Main`, instancie um funcionário, aumente o salário em 15% e mostre os dados antes e depois.

🔹 Exercício 3 — Gerenciador de Produtos com Classes e Métodos Estáticos (1,0 ponto)

Crie uma classe `Produto` com propriedades:

- `Codigo` (int)
- `Nome` (string)
- `Preco` (double)

Implemente dentro da classe `Produto` os seguintes métodos **estáticos** para gerenciar uma lista interna de produtos:

- `Adicionar(Produto p)` — adiciona um produto à lista
- `Remover(int codigo)` — remove o produto pelo código
- `ListarTodos()` — retorna a lista completa de produtos
- `BuscarPorCodigo(int codigo)` — retorna o produto correspondente ao código

No método `Main`, implemente um menu que permita ao usuário:

- Adicionar produto
- Remover produto
- Listar produtos
- Buscar produto por código
- Sair

🔹 Exercício 4 — Classe `Pedido` e Itens (1,0 ponto)

Defina:

- Classe `ItemPedido` com propriedades `NomeProduto` (string), `Quantidade` (int), `PrecoUnitario` (double)
- Classe `Pedido` com uma lista de `ItemPedido` e método `CalcularTotal()` que retorna o valor total do pedido

No `Main`, crie um pedido, adicione 3 itens, e exiba o total formatado.

🔹 Exercício 5 — Classe `ContaBancaria` com Métodos Simples (1,0 ponto)

Implemente uma classe `ContaBancaria` com:

- Propriedades: `NumeroConta` (string), `Saldo` (double)
- Métodos:
  - `Depositar(double valor)` que adiciona saldo
  - `Sacar(double valor)` que subtrai saldo se houver saldo suficiente, senão exibe mensagem de erro
  - `ExibirSaldo()` que mostra o saldo atual formatado

No `Main`, crie uma conta, realize depósitos e saques, exibindo saldo a cada operação.

✅ Avaliação

Item	Pontuação
Teoria Scrum (6x0,25)	1,5 pts
Teoria C#/Classes (6x0,25)	1,5 pts
Dissertativas POO (2x1)	2,0 pts
Prática C# Classes + CRUD (5x1)	5,0 pts
Nota Máxima	**10,0 pts**
Nota Mínima p/ Aprovação (85%)	**8,5 pts**
