Seminário POO em C# — Aprofundamento

Propósito: aprofundar decisões de design orientado a objetos **sem repetir a aula** e **sem usar associações**. **Escopo técnico desta etapa:** implementar **uma única classe** por grupo; não utilizar relações entre classes, herança, interfaces ou coleções de outras classes. (Tipos primitivos, string, DateTime, enum e **arrays internos** são permitidos.)

Regras de escopo

Implementação Limitada

Implementar apenas 1 classe de domínio por grupo (ex.: ReservaSala, CartaoTransporte, Ingresso, Agendamento, ContaFinanceira).

Sem Associações

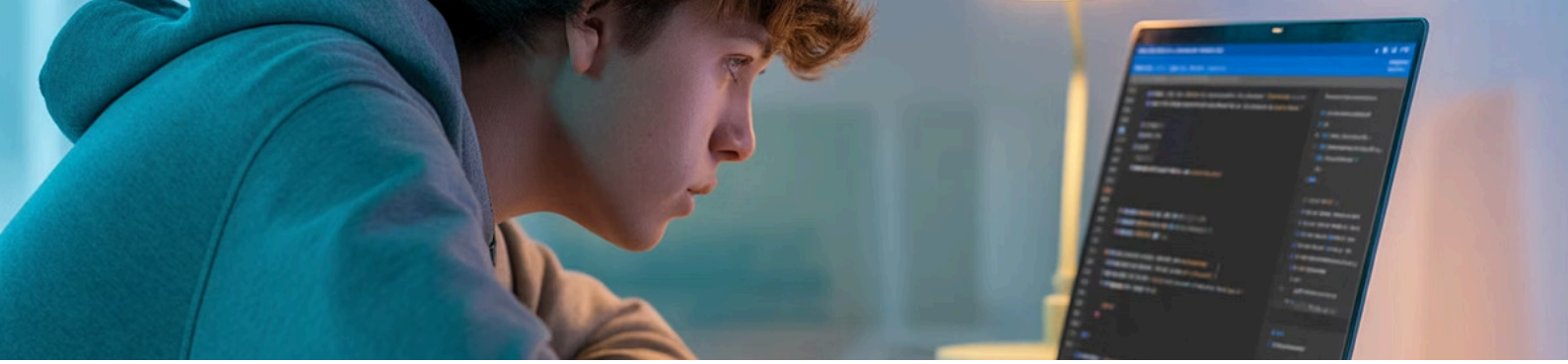
Não usar associações (sem referência a outras classes do domínio). Se precisar citar outro conceito, trate-o como **ID** ou **string** (ex.: SalaId, CodigoEvento) sem criar a classe correspondente.

Elementos Permitidos

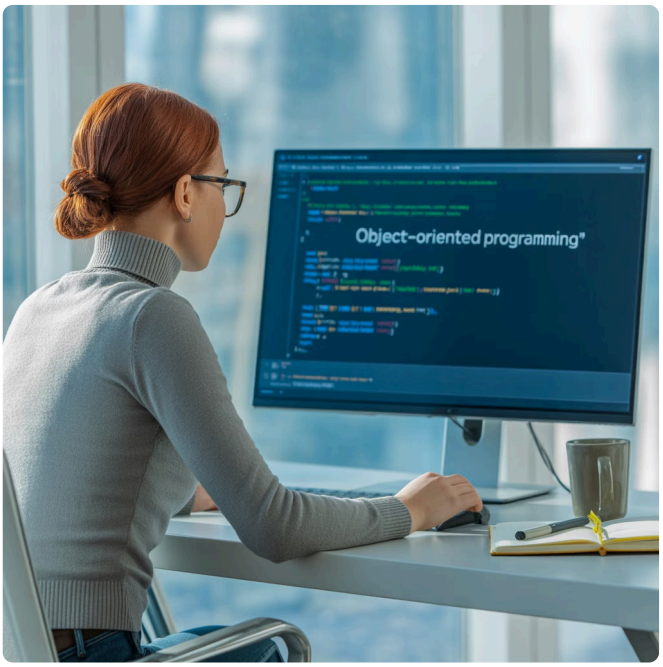
Propriedades com validação, **construtor principal + sobrecargas**, **métodos de negócio**, **exceções adequadas**, **arrays internos** de tipos simples (ex.: string[], decimal[], DateTime[]).

Elementos Proibidos

Proibidos nesta etapa: herança, interfaces, composição/agragação, coleções de objetos de outras classes, ORM, banco de dados, UI.



Objetivos de aprendizagem



- Justificar **encapsulamento**, **invariantes** e **exceções** em uma única classe de domínio.
- Compreender **valor × referência** e efeitos de **aliasing** na prática, usando a mesma classe.
- Decidir entre **auto-property** e **propriedade completa** com validação; organizar **construtores** (principal e sobrecargas) evitando duplicação.
- Explorar limites do uso de **arrays internos** para dados auxiliares da própria classe (ex.: janelas de horário, histórico curto de valores) e propor evolução futura — sem implementá-la.
- Comparar **solução estruturada × OO** para o **mesmo problema**, destacando clareza, proteção de regras e manutenção.

Temas/cenários sugeridos (ou escolha livre)

Escolha **um** cenário e **centralize tudo em uma única classe**. Exemplos de classe única por cenário (você pode renomear/ajustar):

Reserva de Salas

Classe: ReservaSala

Regra: horário válido, sem duração zero, status "solicitada/confirmada/cancelada" como enum.

Bilhetagem de Transporte

Classe: CartaoTransporte

Saldo não negativo, limites de recarga/uso por dia, histórico interno simples de recargas decimal[].

Eventos e Ingressos

Classe: Ingresso

Capacidade do lote local à classe via campo, check-in apenas uma vez, código de validação simples.

Agenda Médica

Classe: Agendamento

Data futura, tolerância de atraso, reagendamento dentro de janela; manter DateTime[] com lembretes internos.

Finanças Pessoais

Classe: ContaFinanceira

Saldo ≥ 0, Depositar/Sacar com validação; manter decimal[] curto para últimos lançamentos.

Biblioteca/Acervo

Classe: EmprestimoLivro

Datas consistentes, multa calculável, estados válidos com enum.

Livre: proponha outro tema. Justifique em 5–7 linhas por que **uma única classe** é suficiente para demonstrar as decisões da trilha escolhida nesta etapa.

Trilhas de pesquisa (escolha 1 por grupo)

Todas as trilhas devem manter o foco em **uma única classe**. Use a mesma classe para demonstrar regras, erros e comparações.

A. Encapsulamento + Invariantes + Exceções

1. Levante as **invariantes** da sua classe e **proponha 2 novas**.
2. Defina **onde validar** (construtor, set, método de negócio) e por quê.
3. Modele **exceções adequadas** e demonstre **2 violações**.
4. Compare com versão **estruturada**: quais bugs ficam mais prováveis?

B. Valor × Referência, Aliasing e Efeitos Colaterais

1. Mostre aliasing com a **mesma classe**. Explique o efeito.
2. Proponha **mitigações**: métodos que retornam cópias/snapshots, imutabilidade parcial, comandos ao invés de setters diretos.
3. Discuta quando **struct** (valor) *poderia* ser considerado para esta classe — sem implementar.

C. Propriedades (auto × completa) e Construtores (principal + sobrecargas)

1. Selecione 3 propriedades e **justifique**: auto-property ou propriedade completa?
2. Crie um **construtor principal** e **2 sobrecargas** que deleguem a ele (sem duplicar validação).
3. Compare com o equivalente **estruturado** (sem encapsular): clareza, erros evitados.

D. Arrays Internos: limites e evolução

1. Use **um array interno** de tipos simples na própria classe. Implementar **operar/adicionar/totalizar** conforme faça sentido.
2. Mostre **dores** (capacidade fixa, remoção/reordenação) e proponha **evolução futura** (conceitual, sem implementar coleções novas).
3. Compare com abordagem **estruturada** mantendo a mesma lógica.

E. Coesão, Acoplamento e Delegação dentro de 1 classe

1. Identifique riscos de "classe faz-tudo" e mova detalhes para **métodos privados** com nomes expressivos (delegação interna, não para outras classes).
2. Antes/depois com métricas simples (LOC, número de responsabilidades).
3. Compare com funções soltas (estruturado): leitura e manutenção.

F. Objetos Ricos × Objetos Anêmicos na mesma classe

1. Mostre a versão **anêmica** (só dados) vs versão **rica** (métodos que expressam regras) da **mesma classe**.
2. Delimite **partes imutáveis** (ex.: ID, data de criação) e mutáveis com validação.
3. Compare com estrutura de registros + funções (estruturado).

As trilhas G–J do plano original podem ser usadas opcionalmente no futuro — manteremos A–F para foco e profundidade com 1 classe.

Entregáveis por grupo

Documento único (mini-dossiê 3–5 páginas) contém **tudo**, inclusive a **Reflexão Individual** de cada integrante (ver §7):

1. Tema e **classe** escolhidos; justificativa.
2. Decisões de design (propriedades, encapsulamento, construtores, métodos) com trechos mínimos.
3. **Comparativo estruturada × OO** (tabela de trade-offs + riscos evitados).
4. **Matriz de invariantes** e **mapa de exceções** (se trilha A/D).
5. **Resultados da demo** (prints e síntese dos 2 casos "ok" + 2 "erro").
6. **§7. Reflexão Individual** anexada ao final do documento (1–2 páginas por estudante).
7. **Infográfico de 1 página**: decisões, invariantes, erros evitados, próximos passos.
8. **Demo de console**: reprodutível, com mensagens claras.
9. **3 perguntas à turma**: uma de **explicar**, uma de **aplicar**, uma de **avaliar**.

Roteiro de apresentação (12–15 min)

1. **Problema & dor (1 min)**
   Por que OO ajuda neste caso **com 1 classe**?
2. **Design da classe (5–6 min)**
   Propriedades (auto×completa), encapsulamento, invariantes, construtores, métodos de negócio; valor×referência quando fizer sentido.
3. **Comparativo estruturada × OO (2–3 min)**
   Ganhos concretos.
4. **Demo (2–3 min)**
   2 casos "ok" + 2 "erro"; mensagens legíveis.
5. **Destaques da Reflexão Individual (2 min)**
   Cada grupo apresenta **síntese** (1–2 insights) do §7 do documento antes do Q&A.
6. **Q&A + Parecer do Grupo-Curador (5–8 min combinados)**
   Perguntas da turma/curador e parecer de 1 minuto.

Observação: a **reflexão completa** de cada integrante **está no documento** (§7) e **deve ser entregue junto**. Na apresentação, traga apenas os **destaques**.

Reflexão Individual (no documento, 1–2 páginas por estudante)

Responda com exemplos **da sua classe**:

Conhecimento Prévio

O que eu já sabia de **programação estruturada** que me ajudou aqui?

Novos Aprendizados

O que a aula/conteúdo me trouxe para **pensar OO de forma diferente**?

Desafios

Dificuldades ao aplicar invariantes/encapsulamento/exceções; plano para superá-las.

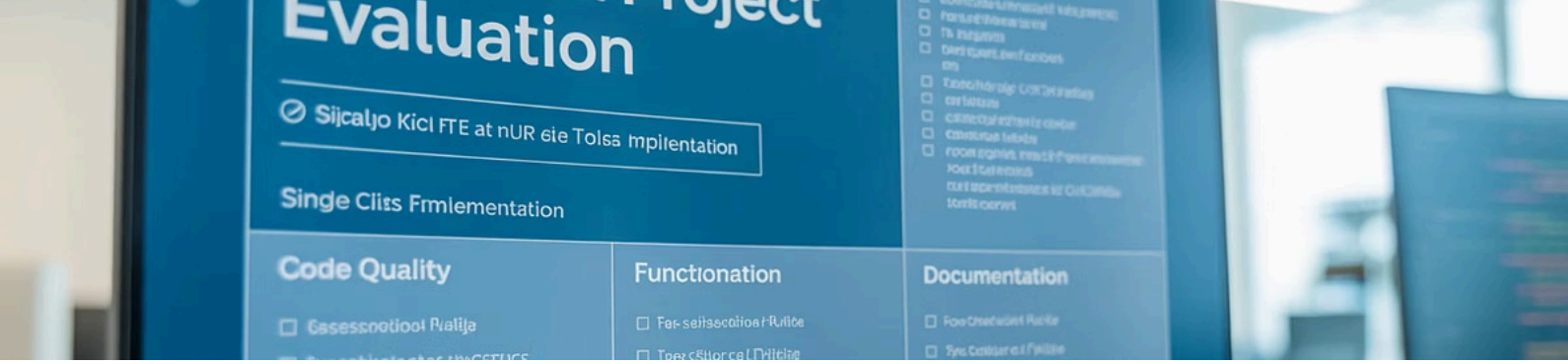
Próximos Passos

Liste 2 tópicos que pretendo estudar e por quê.

Aplicação Prática

Motivações para aplicar: onde vou usar já; cite **uma decisão de design** que vou adotar daqui para frente.

Entrega: anexar as reflexões individuais **dentro do mesmo PDF** do grupo, após a seção de resultados (cada aluno identifica seu nome no topo da sua reflexão).



Rubrica de avaliação e checklists

Rubrica de avaliação (100 pts)

- **Pesquisa e fundamentação (25)** decisões coerentes com a trilha e foco em 1 classe.
- **Qualidade técnica (30)** encapsulamento efetivo; validações e invariantes bem posicionadas; uso adequado de exceções; compreensão de valor×referência.
- **Demonstração (10)** reprodutibilidade; clareza nos casos "ok/erro".
- **Clareza e didática (15)** infográfico, comparação estruturada×OO, gestão do tempo.
- **Reflexão Individual integrada (10)** presença, profundidade, vínculo com a classe do trabalho.
- **Interação e curadoria (10)** respostas às perguntas e parecer do curador.

Checklist — Grupo Apresentador

- Apenas **1 classe** implementada (sem associações).
- Trilha selecionada e **comparação estruturada × OO** preparada.
- **Matriz de invariantes** e **mapa de exceções** (se aplicável).
- **Demo** com 2 casos "ok" + 2 "erro", mensagens claras.
- **Infográfico** de 1 página.
- §7 **Reflexões Individuais** anexadas ao documento.
- 3 perguntas (explicar/aplicar/avaliar).

⊗ **Penalidades:** violar o escopo (criar mais de uma classe, usar associação); repetir exemplos do material base; omitir casos de erro; não anexar as reflexões ao documento.