Entregáveis e Fichas — Clínica de Bugs C# (Templates + Guia)

Este documento reúne **todos os artefatos** da atividade (modelos para copiar/colar ou imprimir). Ao final da correção (depois do prazo), compare com o **Gabarito — Clínica de Bugs C# (10 Problemas)**.

Checklist de Entrega (por equipe)

01

Código final corrigido para os 10 problemas (um projeto console por problema é aceitável).

02

Fichas de Erros (uma por erro identificado sintático, execução e lógico/semântico).

03

Tabela(s) de Testes com casos planejados e resultados observados.

04

Resumo de Aprendizagem por problema (5–8 linhas).

05

Registro de uso de IA (prompts feitos + síntese do que aprenderam, sem colar respostas).

06

Comentários no código seguindo **Clean Code** (explicar *porquês*, evitar redundâncias).

Ficha de Erros (modelo)

Preencher **uma ficha por erro**. Diferencie o tipo de erro e **transcreva mensagens do compilador** (CSxxxx) ou **exceções** quando ocorrer.

Identificação

Problema nº: *ex.: 3*

Arquivo/Trecho: *ex.: Program.cs, linhas 12–18*

Tipo de erro: ( ) Sintático ( ) Execução ( ) Lógico/Semântico

Sinais / Evidências

Mensagem do compilador/exceção: *ex.: CS0103: The name 'idadee' does not exist in the current context*

Linha(s) suspeitas: *ex.: 14*

Análise

Hipótese (por que ocorre?): *ex.: variável escrita com capitalização diferente*

Experimento (o que testei?): *ex.: renomear para 'idade' e recompilar*

Correção

Descrição da correção: *ex.: corrigido identificador; adicionado 'int.Parse' ao ler idade*

Teste de regressão (entradas/saídas esperadas):

- *Entrada:* 17 → *Saída esperada:* "Você é menor de idade."
- *Entrada:* 18 → *Saída esperada:* "Você é maior de idade."

Prevenção

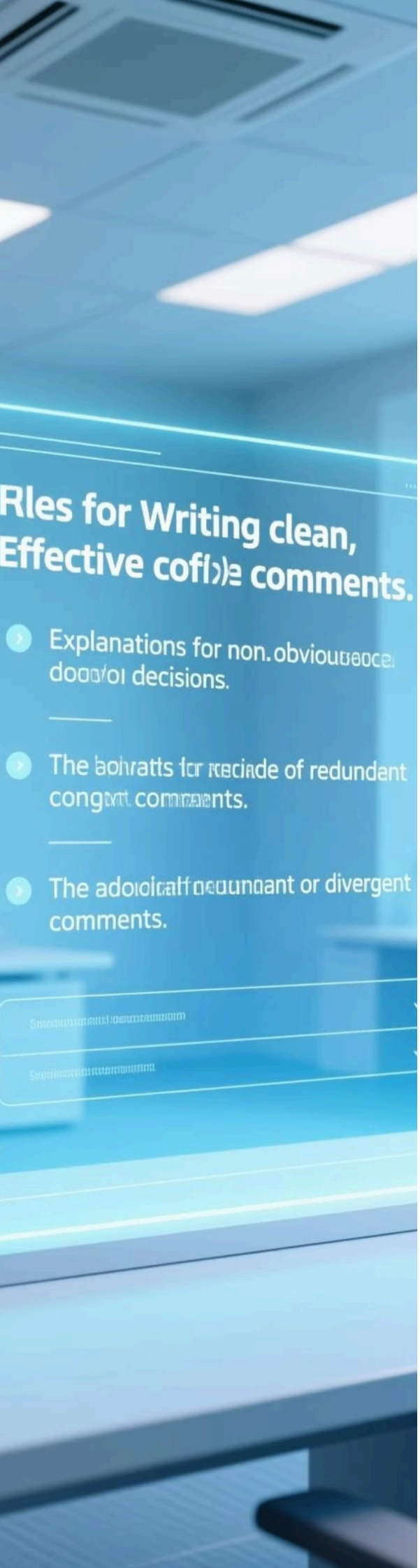
Regra/checklist: *ex.: evitar nomes parecidos; ativar warnings; usar IDE com análise estática*

Observações: ex.: aprendemos a diferença entre '=' e '==' em C#

Tabela de Testes (modelo)

Use uma por problema, acrescente linhas conforme necessário.

Caso	Entradas	Passos relevantes	Saída esperada	Saída observada	OK?
1	ex.: idade=18	Executar programa	"Você é maior de idade."		
2					
3					



Resumo de Aprendizagem (modelo por problema)

5–8 linhas; escrever com as próprias palavras.

O que aprendi sobre **C#** neste problema (tipos, literais, escopo, interpolação, etc.).

O **erro mais interessante** e como o diagnostiquei (mensagem, suspeita, teste).

Como garanti que **não volta** (teste de regressão/prevenção).

Dúvidas que ficaram para discutirmos.

Registro de uso de IA (modelo)

Se usar IA, **não** cole a resposta: documente **o processo**.

Quando usei

(momento e motivo)

Como apliquei

(o que aceitei/adaptei)

Pergunta(s) feita(s)

(prompts)

O que aprendi

(conceitos, mensagens CS, armadilhas)

Resposta(s) resumida(s)

(máx. 3 linhas cada)

Diretrizes de Comentários (Clean Code)

Faça

- Explique **por que** algo existe (decisão, suposição, risco, regra de negócio curta).
- Comente **coisas não óbvias** (ex.: cultura na formatação de números: pt-BR vs InvariantCulture).
- Cabeçalho curto por arquivo: objetivo, entradas/saídas esperadas, riscos conhecidos.

Evite

- Comentar o óbvio: i++ // incrementa i.
- Comentários que **divergem** do código (apodrecem com o tempo).
- Usar comentário para esconder má nomeação — renomeie variáveis/métodos.

Exemplo bom

```
// Usamos CultureInfo.InvariantCulture para garantir ponto decimal na leitura em qualquer SO.
```

Exemplo ruim

```
// Soma o total
soma = soma + x; // óbvio e redundante
```

Rubrica de Avaliação (0–10)

3	3	2	2
Identificação & classificação de erros	**Qualidade das justificativas e correções**	**Testes**	**Clareza & colaboração**
Reconhece sintaxe/execução/lógica e cita mensagens CS/exceções.	Hipótese → experimento → evidência → correção.	Casos relevantes, bordas, e teste de regressão.	Organização, papéis, comentários úteis, registro de IA.

Bônus (até +0,5): prevenção documentada em checklist reutilizável.

Mapa rápido de mensagens e exceções comuns

Compilador (CSxxxx)

- **CS1002**: ; esperado. → Erro sintático.
- **CS0103**: nome não existe no contexto atual. → Escopo/typo/using faltando.
- **CS0165**: uso de variável local não atribuída. → Fluxo de atribuição.
- **CS0029**: conversão implícita inválida de tipo A para B. → Tipagem/conversão.
- **CS1525/CS1003**: termo/token inválido. → Sintaxe geral.

Exceções comuns (execução)

- **IndexOutOfRangeException**: acesso fora dos limites do array.
- **DivideByZeroException**: divisão por zero.
- **FormatException**: conversão inválida (ex.: int.Parse("abc")).
- **OverflowException**: valor fora do intervalo do tipo.
- **NullReferenceException**: uso de referência nula.

Procedimento recomendado de depuração (passo a passo)

Compilar/Executar

O código fornecido *sem alterações* e **ler as mensagens** com calma.

Listar sinais

(CSxxxx, exceções, linhas suspeitas) e **classificar** o tipo de erro.

Gerar hipótese

De causa-raiz e planejar um **experimento mínimo** (alteração isolada).

Aplicar e testar

Registrar efeitos (antes/depois). **Apenas um erro por vez**.

Escrever prevenção

(regra/checklist) e criar **teste de regressão**.