SEGUNDO BIMESTRE

Pontos ?
Qtde de Exercícios ?

Exercício 1 - Desenvolva uma aplicação de console em C# para gerenciar alunos de uma instituição. A aplicação deve utilizar uma lista de alunos e permitir interações por meio de um menu.

Classe Aluno

A classe deve conter:

- `RA` (string): código único de identificação do aluno
- `Nome` (string): nome completo do aluno
- `Idade` (int): idade do aluno

Funcionalidades da aplicação

O programa deve permitir:

1. **Cadastrar um aluno** (com RA único)
2. **Listar todos os alunos cadastrados**
3. **Alterar dados de um aluno existente (localizado pelo RA)**
4. **Remover um aluno pelo RA**
5. **Encerrar o programa**

A aplicação deve executar continuamente até que o usuário opte por sair.

Exercício 2 - Cadastro e Gerenciamento de Produtos em C#

Desenvolva uma aplicação de console em C# que permita o cadastro e gerenciamento de produtos. O objetivo é praticar o uso de classes, listas e operações básicas com dados.

A aplicação deve conter:

- Uma classe `Produto` com as seguintes propriedades:
    - `Descricao` (texto do nome ou identificação do produto)
    - `Valor` (preço em decimal)

A partir dessa classe, crie uma lista de produtos (`List<Produto>`) que será manipulada por meio de um menu interativo com as seguintes funcionalidades:

1. **Cadastrar produto:** solicita ao usuário a descrição e o valor do produto, e adiciona à lista.
2. **Remover produto:** permite remover um produto informando a descrição.
3. **Pesquisar produto:** busca na lista um produto pela descrição e exibe seus dados, se encontrado.
4. **Mostrar produto com menor valor:** exibe o produto que possui o menor valor na lista.
5. **Sair:** finaliza o programa.

A aplicação deve permanecer em execução até que o usuário escolha sair. Utilize estruturas como `while` ou `do-while`, e aplique boas práticas como métodos separados para cada funcionalidade.

Exercício 3 - Pagamentos

Você está desenvolvendo um sistema de pagamentos para uma loja virtual. Crie uma interface chamada IPagamento com os seguintes membros:

```
public interface IPagamento
{
        void ProcessarPagamento(decimal valor);
}
```

Agora, implemente duas classes que usam essa interface:

1. PagamentoCartaoCredito
2. PagamentoBoleto

Ambas devem implementar o método ProcessarPagamento, com saída diferente no Console.WriteLine:

- Para PagamentoCartaoCredito, exiba:
  "Pagamento de R$[valor] processado no cartão de crédito."

- Para PagamentoBoleto, exiba:
  "Pagamento de R$[valor] processado via boleto bancário."

Depois, crie uma classe LojaVirtual com um método:

public void RealizarPagamento(IPagamento metodo, decimal valor)

Este método deve simplesmente chamar metodo.ProcessarPagamento(valor).

Objetivo

· - Criar e usar interfaces

· - Usar polimorfismo para processar diferentes formas de pagamento

· - Demonstrar como uma interface permite flexibilidade de implementação

Desafio extra (opcional)

Implemente uma terceira forma de pagamento: PagamentoPIX. E modifique o programa principal para aceitar escolha do usuário por Console.

Exercício 4 - Try e Catch - Exceção

Crie um programa em C# que solicite ao usuário dois números inteiros e realize a divisão do primeiro pelo segundo.

Implemente o tratamento de exceções com try/catch para lidar com os seguintes casos:

- O usuário digitar valores que não são números inteiros (FormatException)
- Tentativa de divisão por zero (DivideByZeroException)

O programa deve exibir mensagens apropriadas para cada tipo de erro e permitir que o usuário tente novamente.

Exemplo de saída esperada:

Digite o primeiro número: a
Erro: Valor inválido. Digite um número inteiro.

Digite o segundo número: 0
Erro: Não é possível dividir por zero.

Digite o primeiro número: 10
Digite o segundo número: 2
Resultado: 5

Objetivos

- - Utilizar a estrutura try/catch para capturar exceções
- - Tratar diferentes tipos de erros de entrada do usuário
- - Aplicar boas práticas de validação de dados

Desafio extra (opcional)

Permitir que o usuário continue tentando até digitar valores válidos e encerrar com uma opção de saída.

Exercício 4 - Competição e competidores

Crie um programa em C# que utilize duas classes: Competicao e Competidor.

- A classe Competidor deve conter:
  - Nome (string)
  - Idade (int)
  - Modalidade (string)

- A classe Competicao deve conter:
  - Nome (string)
  - Lista de Competidores (List<Competidor>)
  - Um método para adicionar competidores

Implemente também um menu no console que permita ao usuário:

1. Cadastrar uma nova competição (informar o nome)
2. Adicionar competidores à competição
3. Listar os competidores cadastrados
4. Sair

Utilize boas práticas como encapsulamento, uso de propriedades e construtores quando necessário.

Objetivos

· - Praticar o uso de classes e objetos

· - Trabalhar com listas de objetos

· - Implementar interações com o usuário via Console

· - Aplicar conceitos de encapsulamento e composição

Desafio extra (opcional)

Permitir editar ou remover um competidor já cadastrado.