Novo Exercício: Sistema de Gerenciamento de Tarefas com Modificadores de Escopo

Objetivo: Desenvolver um sistema para gerenciar tarefas de um projeto, utilizando modificadores de escopo para controlar a visibilidade e acessibilidade dos membros da classe.

Instruções

1. **Crie a Classe Base Tarefa**
    - Atributos:
        - `Titulo` (deve ser acessível de qualquer lugar)
        - `Descricao` (deve ser acessível apenas dentro da classe `Tarefa` e suas classes derivadas)
        - `DataCriacao` (deve ser acessível dentro do mesmo assembly)
        - `Prioridade` (deve ser acessível dentro do mesmo assembly e suas classes derivadas)
    - Método:
        - `MostrarDetalhes()` (deve ser acessível de qualquer lugar)
2. **Crie a Classe Derivada TarefaDesenvolvimento**
    - Atributos:
        - `LinguagemDeProgramacao` (deve ser acessível de qualquer lugar)
    - Método:
        - `MostrarDetalhes()` (deve ser acessível de qualquer lugar e sobrescrever o método da classe base)
3. **Crie a Classe Derivada TarefaDesign**
    - Atributos:
        - `FerramentaDesign` (deve ser acessível de qualquer lugar)
    - Método:
        - `MostrarDetalhes()` (deve ser acessível de qualquer lugar e sobrescrever o método da classe base)
4. **Defina as Propriedades e Seus Modificadores de Escopo**
    - Para cada propriedade, forneça uma breve descrição e peça aos alunos para determinar o modificador de escopo apropriado.

Código Exemplo

```
using System;

namespace GerenciamentoDeTarefas
{
    // Classe base para tarefas
    public abstract class Tarefa
    {
        // Propriedade para o título da tarefa, deve ser acessível de qualquer lugar
        public string Titulo { get; set; }
```

```
        // Propriedade para a descrição da tarefa, deve ser acessível apenas dentro da classe e suas derivadas
        protected string Descricao { get; set; }

        // Propriedade para a data de criação da tarefa, deve ser acessível dentro do mesmo assembly
        internal DateTime DataCriacao { get; set; }

        // Propriedade para a prioridade da tarefa, deve ser acessível dentro do mesmo assembly e suas derivadas
        protected internal int Prioridade { get; set; }

        public Tarefa(string titulo, string descricao, DateTime dataCriacao, int prioridade)
        {
            Titulo = titulo;
            Descricao = descricao;
            DataCriacao = dataCriacao;
            Prioridade = prioridade;
        }

        // Método para mostrar detalhes da tarefa, deve ser acessível de qualquer lugar
        public virtual void MostrarDetalhes()
        {
            Console.WriteLine($"Título: {Titulo}, Descrição: {Descricao}, Data de Criação: {DataCriacao}, Prioridade: {Prioridade}");
        }
    }

    // Classe derivada para tarefas de desenvolvimento
    public class TarefaDesenvolvimento : Tarefa
    {
        // Propriedade para a linguagem de programação, deve ser acessível de qualquer lugar
        public string LinguagemDeProgramacao { get; set; }

        public TarefaDesenvolvimento(string titulo, string descricao, DateTime dataCriacao, int prioridade, string linguagemDeProgramacao)
            : base(titulo, descricao, dataCriacao, prioridade)
        {
            LinguagemDeProgramacao = linguagemDeProgramacao;
        }

        // Método para mostrar detalhes da tarefa de desenvolvimento, deve ser acessível de qualquer lugar
        public override void MostrarDetalhes()
        {
            base.MostrarDetalhes();
            Console.WriteLine($"Linguagem de Programação: {LinguagemDeProgramacao}");
        }
    }
```

```
    // Classe derivada para tarefas de design
    public class TarefaDesign : Tarefa
    {
        // Propriedade para a ferramenta de design, deve ser acessível de qualquer
lugar
        public string FerramentaDesign { get; set; }

        public TarefaDesign(string titulo, string descricao, DateTime dataCriacao, int
prioridade, string ferramentaDesign)
            : base(titulo, descricao, dataCriacao, prioridade)
        {
            FerramentaDesign = ferramentaDesign;
        }

        // Método para mostrar detalhes da tarefa de design, deve ser acessível de
qualquer lugar
        public override void MostrarDetalhes()
        {
            base.MostrarDetalhes();
            Console.WriteLine($"Ferramenta de Design: {FerramentaDesign}");
        }
    }

    public class Program
    {
        public static void Main()
        {
            TarefaDesenvolvimento tarefaDev = new TarefaDesenvolvimento("Desenvolver
API", "Criar endpoints RESTful", DateTime.Now, 1, "C#");
            TarefaDesign tarefaDesign = new TarefaDesign("Criar Wireframe", "Design do
layout da página inicial", DateTime.Now, 2, "Figma");

            tarefaDev.MostrarDetalhes();
            tarefaDesign.MostrarDetalhes();
        }
    }
}
```

Exercícios Propostos

1. **Defina os Modificadores de Escopo**

    - Atribua os modificadores de escopo apropriados para as seguintes propriedades com base nas descrições fornecidas:
        - `Titulo`: Deve ser acessível de qualquer lugar.
        - `Descricao`: Deve ser acessível apenas dentro da classe `Tarefa` e suas classes derivadas.
        - `DataCriacao`: Deve ser acessível dentro do mesmo assembly.
        - `Prioridade`: Deve ser acessível dentro do mesmo assembly e suas classes derivadas.

2. **Implemente uma Nova Classe Derivada**

- Crie uma nova classe derivada chamada `TarefaTeste` que herde da classe `Tarefa`.
- Adicione uma propriedade chamada `AmbienteTeste` que deve ser acessível apenas dentro da classe `TarefaTeste` e suas derivadas.
- Implemente o método `MostrarDetalhes()` para exibir os detalhes da tarefa de teste.

3. **Teste a Implementação**

   - Crie instâncias das classes `TarefaDesenvolvimento`, `TarefaDesign` e `TarefaTeste` e utilize seus métodos, verificando a visibilidade dos membros de acordo com os modificadores de escopo.

Esses exercícios irão ajudar a consolidar o entendimento sobre o uso e a importância dos modificadores de escopo em C#.