Universidade Luterana do Brasil – Ulbra
Campus Torres

Miguel da Silva Cardoso

Projeto Agência de Viagens – Pilares da Programação Orientada a Objetos em C#

Torres – Rio Grande do Sul

25/10/2024

INTRODUÇÃO

Nesta seção, apresentamos uma visão geral do projeto, destacando a importância da Programação Orientada a Objetos (POO) para a organização e reusabilidade do código.

O projeto consta em uma agência de viagens, onde é possivel fazer varias funções que se espera da mesma, como: fazer reservas, procurar destinos, procurar por preço, etc.

A (POO) é uma abordagem fundamental no desenvolvimento de software moderno por várias motivos. Iremos dissertar mais sobre eles nos próximos artigos.

PILARES DA PROGRAMAÇÃO ORIENTADA A OBJETOS

A seguir, detalharei os quatro pilares da POO aplicados no projeto.

Encapsulamento

O encapsulamento permite esconder detalhes internos das classes. No projeto, foi aplicado na classe destino como no exemplo abaixo:

O encapsulamento é essencial para proteger dados sensíveis e garantir a integridade. Neste exemplo, não quero que qualquer classe seja capaz de mexer nas propriedades do destino, assim, faço com elas só sejam acessadas a partir da própria classe.

```
11 references
public class Destino
{
    4 references
    private string nomeLocal;
    5 references
    private string pais;
    5 references
    private string codigo;
    4 references
    private string descricao;

```

Herança

Herança permite que uma classe compartilhe atributos e métodos de outra classe. No projeto, a classe `PacoteTuristico` herdou propriedades comuns de uma classe base, sendo essa a ServicoVIagem.

```
3 references
public abstract class ServicoViagem
{
    5 references
    public string Codigo { get; set; }
    7 references
    public string Descricao { get; set; }

    1 reference
    protected ServicoViagem(string codigo, string descrica
    {
        Codigo = codigo;
        Descricao = descricao;
    }

    2 references
    public abstract void Reservar();

    2 references
    public abstract void Cancelar();

    3 references
    public abstract void ExibirInformacoes();
}
```

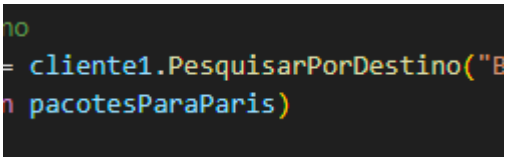
```
public override void Reservar()
{
    if(VagasDisponiveis > 0)
    {
        VagasDisponiveis--;
        Console.WriteLine("Reserva confirmada é us guri boa viagem");
    }
    else
    {
        Console.WriteLine($"cabo as vaga");
    }
}

2 references
public override void Cancelar()
{
    VagasDisponiveis++;
    Console.WriteLine("Reserva cancelada é us guri boa viagem");
}
```

Classe PacotesTuristicos herdando os metodos Reservar e Cancelar.

Polimorfismo

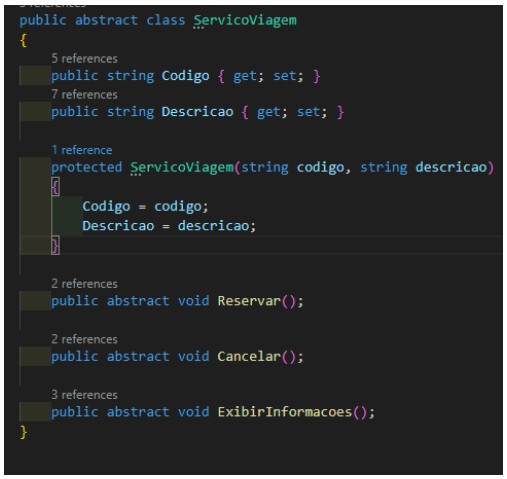
O polimorfismo permite a definição de métodos que podem ser usados de maneira geral, mesmo que implementados de formas distintas em diferentes classes. A interfaces IReservavel e IPesquisavel fazem ser possível diferentes classes utilizar o método PequisarPorCodigo, por exemplo. A classe PacoteTuristica, que herda da ServiçoViagem, usa a forma de sobreescrevevar os métodos Reservar e Cancelar.



Métodos na Interface IPesquisavel sendo utilizados pelo objeto cliente 1 da classe Cliente.

Abstração

A abstração foi utilizada para criar classes e métodos genéricos, simplificando a complexidade do sistema e permitindo foco nos comportamentos essenciais, além de garantir que as classes herdadas implementem os métodos.



Neste exemplo, as classes herdadas da ServicoViagem serão obrigadas a implementar os métodos. Isso é útil pois todos os serviços de viagem obrigatoriamente devem ter reservas e cancelamentos.

CONCLUSÃO

Desenvolver este projeto foi um grande desafio e sem dúvidas uma forma de práticar os pilares da POO. Atráves dessa prática, pude compreender muito melhor qual o motivo da criação das linguagens desse tipo, além de aprofundar minha experiência com um projeto com diversas classes e métodos.
No geral, recomendo a todos participarem de um projeto como esse, se desejam desenvolver sua habilidades ou até mesmo começar a entender a POO.

REFERÊNCIAS

https://www.w3schools.com/cs/index.php

https://learn.microsoft.com/en-us/dotnet/csharp/