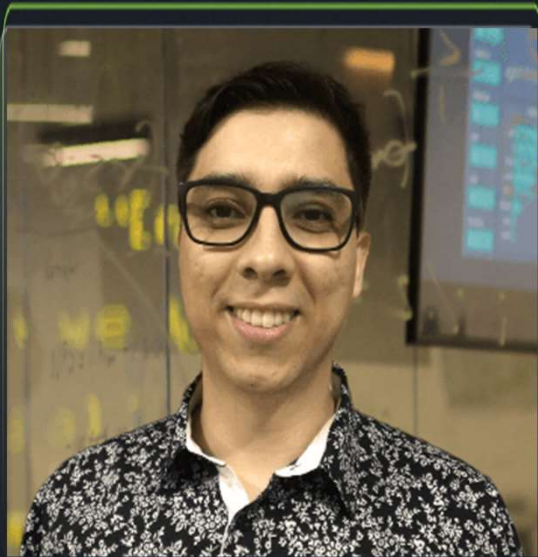


Programação Orientada a Objetos II (C#)

Singia - 956 | #BeTheNext - C#

Apresentação

Professores



Sobre o professor...

<https://www.linkedin.com/in/michael-tadeu>

Estrutura do Módulo e Metodologia de Avaliação

Estrutura do Módulo e Metodologia de Avaliação

-> Módulo

- > Interfaces

- > Abstração e Implementação

- > Generics

- > Segregação de Responsabilidades (SOLID)

- > Aberto e Fechado (SOLID)


-> Avaliação

- > Projeto Final

- > Exercícios em Aula e no Class

- > Participação em Aula

Combinados



Metodologia de Ensino do Módulo

Estrutura do Módulo e Metodologia de Avaliação

-> Aulas Expositivas

-> Live Coding

-> Exercícios/ Desafios

Ferramentas

Ferramentas

-> Repositório

-> GitHub

-> IDE

-> Visual Studio ou Visual Code

O que vamos aprender...

Programação Orientada a Objetos II

- > Revisão de POO
- > Interfaces
- > Abstração e Implementação
- > Generics
- > Segregação de Responsabilidades (SOLID)
- > Aberto e Fechado (SOLID)

Revisão de Programação Orientada a Objetos

Revisão de POO – Classes e Objetos

- > A classe é uma abstração do mundo real em programação
- > Como definir uma classe e seus atributos
- > Como criar uma instância (objeto) de uma classe

Revisão de POO – Modificadores de Acesso

-> São utilizados para definir níveis de acesso aos membros da classe

Declaração	Definição
public	Acesso ilimitado
private	Acesso limitado à classe e seus membros
internal	Acesso limitado ao programa (assembly)
protected	Acesso limitado à classe, seus membros e seus derivados

Revisão de POO – Métodos

-> Um método é um comando que representa uma ação

-> Utilizando sobrecarga de métodos

Revisão de POO – Construtores

-> Construtores são métodos especiais responsáveis pela implementação de ações necessárias para a existência de um objeto

-> Sobrecarga de Construtores

Revisão de POO – Herança

- > A herança está relacionada as hierarquias e as relações entre os objetos
- > É o mecanismo em que uma classe filha compartilha automaticamente todos os métodos e atributos de sua classe pai
- > A herança permite implementar classes descendentes implementando os métodos e atributos que se diferenciam da classe pai

Revisão de POO – Herança (Tipos de Herança)

-> **Simples**

Quando uma classe herda as propriedades de uma única classe pai

-> **Múltipla**

Ocorre quando uma classe tem mais de um pai

Revisão de POO – Polimorfismo

-> Polimorfismo significa: “Muitas Formas” e representa o fato de uma determinada característica ser diferente para cada filho

-> Partimos de um objeto mais simples e que vai evoluindo. Os conceitos do objeto pai continuam a existir, mesmo que tenham sofrido modificações ou assumido novas formas

Revisão de POO – Encapsulamento

-> Encapsulamento é o ato de esconder do usuário informações que não são de seu interesse

-> O objeto atua como uma caixa preta, que realiza determinadas operações mas o usuário não sabe e não precisa saber exatamente como

-> Basicamente o encapsulamento separa os elementos visíveis de um objeto dos invisíveis

Revisão de POO – Propriedades

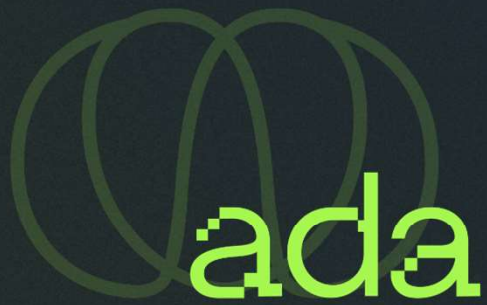
-> São métodos que protegem acesso aos membros da classe

Próxima Aula

POO – Abstração

-> Abstrair um objeto do mundo real para um contexto específico, considerando apenas os atributos importantes

Obrig.ada



Programação Orientada a Objetos II (C#)

Singia - 956 | #BeTheNext - C#

Projeto Final

Estrutura do Módulo e Metodologia de Avaliação

- > Projeto Final

 - > Apresentação

 - > Grupos

 - > Tema

O que vamos aprender...

Programação Orientada a Objetos II

-> Abstração e Implementação

-> Classes Abstratas

-> Interfaces



Abstração

POO – Abstração

- > Abstrair um objeto do mundo real para um contexto específico, considerando apenas os atributos importantes
- > NÃO forneça abstrações, a menos que sejam testadas desenvolvendo várias implementações concretas e APIs que consumam as abstrações
- > ESCOLHA cuidadosamente entre uma classe abstrata e uma interface ao criar uma abstração
- > CONSIDERE fornecer testes de referência para implementações concretas de abstrações. Esses testes devem permitir que os usuários testem se suas implementações implementam corretamente o contrato

Classes Abstratas

POO – Classes Abstratas

- > A finalidade de uma classe abstrata é fornecer uma definição comum de uma classe base que pode ser compartilhada por várias classes derivadas
- > Pode acontecer que ao escrever um método para uma classe você não saiba como ele vai ser implementado. Neste caso, a implementação será feita pela classe que herdar o método (a classe filha)
- > Pode acontecer também que você saiba que um determinado método será sobreposto com certeza na classe filha. Então, por que definir sua implementação se ela não será usada?

POO – Classes Abstratas

- > Uma classe abstrata é uma classe base genérica
 - > Contém métodos abstratos que devem ser implementados nas classes que derivam dela
- > Um método abstrato não apresenta implementação na classe base
- > Pode conter membros não-abstratos
- > Derivando a classe abstrata e implementando os membros abstratos

Interfaces

POO – Interface

-> Uma interface contém definições para um grupo de funcionalidades relacionadas que uma classe não abstrata deve implementar

-> Uma interface é parecida com uma classe abstrata, a diferença é que uma classe abstrata pode possuir métodos que não estejam implementados e pode possuir métodos que estejam implementados.

-> Uma interface possui somente métodos que não estão implementados e que devem ser implementados pela classe que usar a interface

POO – Interface

-> Como o C# não suporta herança múltipla as interfaces permitem que uma classe estenda múltiplas interfaces contornando o problema

-> Uma interface no C# não pode conter atributos, somente pode ter métodos, propriedades e eventos. Todos os membros de uma interface são públicos e não podem usar um modificador de acesso

POO – Interface

-> A classe que implementa a interface deve possuir a definição de todos métodos existentes na interface. Esta definição deve possuir o mesmo nome e a mesma assinatura, retorno e parâmetros, do método na interface.

-> O nome da classe e o nome da interface são separados por dois pontos(:)

POO – Interface

-> Pode tornar o comportamento de seus objetos semelhante ao comportamento dos objetos da .NET Framework

-> Exemplos:

ICollection

IComparer

IDictionary

Obrig.ada