Aula 5

Escrever programas em C# que utilizem classes e objetos

Prof. Ricardo Frohlich da Silva

Classes

- São especificações para objetos;
  - Representam um conjunto de objetos que compartilham características e comportamentos comuns.
    - Todo carro tem em comum:
      - Característica
        - Cor
        - Pneu
        - Direção
      - Comportamento
        - Dirigir
        - Frear

Classes

- Classe Lampada
- Atributos
  - potencia, ligada
- Métodos
  - ligar, desligar, EstaLigada

Classes

- Nome da classe
  - Atributos
  - Métodos

+	**Pública** – O atributo ou método pode ser usado por qualquer objeto.
#	**Protegida** – O atributo ou método pode ser usado por qualquer objeto da classe e também por suas subclasses.
~	**Pacote** – O atributo ou método é visível por qualquer objeto dentro do pacote.
-	**Privada** – O atributo ou método é visível somente pela classe que o define.

Classes em C#

- Declaração de uma classe em C#

```
internal class Lampada
{
    public bool ligada;
    public double potencia;

    public void Ligar()
    {
        ligada = true;

    }
    public void Desligar()
    {
        ligada = false;
    }
    public bool EstaLigada()
    {
        return ligada;
    }
}
```

Classes em C#

- Testando a classe Lampada

```
static void Main(string[] args)
{
    Lampada l1 = new Lampada();
    l1.potencia = 200;
    l1.Ligar();
    l1.Ligar();
    Console.WriteLine("Status da lampada: " + l1.EstaLigada());
    l1.Desligar();
    Console.WriteLine("Status da lampada: " + l1.EstaLigada());
}
```

Resumo

- Objeto
  - Qualquer entidade que possui características e comportamento
- Classe
  - Descreve um tipo de objeto
  - Define atributos e métodos
- Atributo
  - Define características do objeto
- Método
  - Operações que o objeto pode realizar

Construtor

- Um método construtor pode ser utilizado para inicializar um objeto de uma classe.

- *Por padrão, o compilador fornece um construtor-padrão sem parâmetros em qualquer classe que não inclua explicitamente um construtor.*

Construtor

- A palavra-chave new chama o construtor da classe para realizar a inicialização

- Ou seja, quando uma classe é instanciada, o primeiro método a ser executado é o construtor daquela classe..

Construtor

- A chamada de um construtor é indicada pelo nome da classe seguido por parênteses.

- O nome do construtor deve ser igual ao nome da classe

Construtor

- Melhorando a classe Lampada

```
internal class Lampada
{
    public bool ligada;
    public double potencia;
    public string cor;

    public Lampada(double potencia, string cor)
    {
        potencia = potencia;
        cor = cor;
    }

    public void Ligar()
    {
        if (!EstaLigada())
        {
            ligada = true;
            Console.WriteLine("Ligando lâmpada!");
        }
        else
        {
            Console.WriteLine("A lâmpada já estava ligada!");
        }
    }
    public void Desligar()
    {
        if (EstaLigada())
        {
            ligada = false;
            Console.WriteLine("Desligando lâmpada!");
        }
        else
        {
            Console.WriteLine("A lâmpada já estava desligada!");
        }
    }
    public bool EstaLigada()
    {
        return ligada;
    }
}
```

```
internal class Lampada
{
    public bool ligada;
    public double potencia;
    public string cor;

    public Lampada(double potencia, string cor)
    {
        potencia = potencia;
        cor = cor;
    }

    public void Ligar()
    {
        if (!EstaLigada())
        {
            ligada = true;
            Console.WriteLine("Ligando lâmpada!");
        }
        else
        {
            Console.WriteLine("A lâmpada já estava ligada!");
        }
    }
    public void Desligar()
    {
        if (EstaLigada())
        {
            ligada = false;
            Console.WriteLine("Desligando lâmpada!");
        }
        else
        {
            Console.WriteLine("A lâmpada já estava desligada!");
        }
    }
    public bool EstaLigada()
    {
        return ligada;
    }
}
```

Exercício 1

- Crie uma classe chamada Pessoa com seus atributos conforme sua preferência (Nome, CPF, data de nascimento, RG etc).
    - Crie métodos para cadastrar e apresentar na tela os dados desta classe.

Exercício 2

- Escreva uma classe Aluno contendo todos os atributos de um aluno. Faça métodos para apresentar os dados.
    - Faça a leitura pelo teclado dos atributos e crie um construtor para fazer o instanciamento.

Exercício 3

- Crie uma classe Livro que represente os dados básicos de um livro, além destes, criar um atributo do tipo boolean chamado emprestado.
  - Crie métodos emprestar e devolver que altera o atributo emprestado
  - Crie um método construtor que receba todos os valores por parâmetros, exceto o atributo emprestado que obrigatoriamente deve ser inicializado como *false*
  - Faça a leitura pelo teclado dos atributos para instanciar dois livros

Exercício 4

- Crie uma classe chamada Personagem. Defina seus atributos, mas dentre eles deve conter: Nome, posição e itens coletados, no mínimo.
  - Crie construtor e faça a leitura dos atributos pelo teclado.
  - Crie um método chamado “atacar” que recebe por parâmetro uma variável do tipo double que indica o dano do ataque numa escala de 0 a 10. O método deve apresentar uma mensagem na tela com o dano.
  - Crie um método chamado “movimentar” que deve receber por parâmetro uma variável do tipo int que indica a direção que o personagem vai se mover (1 – frente, 2 – trás, 3 – direita e 4 – esquerda). O método deve apresentar uma mensagem na tela mostrando a direção que o personagem vai se mover.

Convenções de nomenclatura

- Em C#, é comum seguir convenções de nomenclatura para manter um código limpo, legível e consistente.
- Classes:
  - Os nomes das classes devem ser substantivos no singular e usar PascalCase (cada palavra começa com letra maiúscula e não há espaços). Por exemplo: MinhaClasse.
- Métodos:
  - Os nomes dos métodos devem ser verbos ou frases verbais e usar PascalCase. Por exemplo: CalcularTotal, ObterNomeCompleto.
  - Evite nomes de métodos muito longos e procure manter os nomes descritivos, mas concisos.

Convenções de nomenclatura

- Constantes:
  - Os nomes de constantes devem ser em letras maiúsculas com palavras separadas por sublinhados. Por exemplo: VALOR_PI, TAMANHO_MAXIMO.
- Parâmetros de métodos:
  - Os nomes de parâmetros devem ser descritivos e usar camelCase. Por exemplo: CalcularImposto(decimal valorTotal).

Convenções de nomenclatura

- Atributos (variáveis de instância):
  - Os nomes de atributos devem ser descritivos e usar camelCase (começar com letra minúscula e usar maiúsculas para palavras subsequentes). Por exemplo: nomeCompleto, quantidadeItens.
- Campos (atributos) *(Veremos mais sobre quando trabalharmos com encapsulamento)*
  - Os campos para utilização interna devem começar com letra minúscula, e para cada palavra, a primeira letra deve ser maiúscula. Como geralmente são utilizados no escopo da classe como protected ou private, devemos iniciar seu nome com underscore “”, e para cada palavra, a primeira letra deve ser maiúscula (padrão _lowerCamelCase). Exemplo: protected string _nome; private int _qtdProdutos;

Convenções de nomenclatura

- Propriedades *(Veremos mais sobre quando trabalharmos com encapsulamento)*

  - As propriedades devem seguir a mesma convenção de nomes dos atributos (camelCase). No entanto, elas devem ser declaradas usando PascalCase, com um get e set. Por exemplo: public string NomeCompleto { get; set; } public int QuantidadeItens { get; set; }

Convenções de nomenclatura

- Interfaces *(Veremos mais sobre quando trabalharmos com Herança e polimorfismo)*

  - As interfaces devem começar com a letra ‘I’ maiúscula e para cada palavra, a primeira letra também deve ser maiúscula (padrão PascalCase). Exemplo: public class IPedidoService { }