PROGRAMAÇÃO ORIENTADA A OBJETOS EM C#

PROF. ME. HÉLIO ESPERIDIÃO

CLASSE (CACHORRO)

- Atributos(Características)
  - Nome, Idade, Tamanho, Cor, Peso, etc.
- Métodos
  - Latir, Correr em círculos, comer, etc.
- Objetos
  - As classes nada mais são do que moldes para criação de objetos.
- Instanciar
  - Cria objetos por meio de classes.

DIAGRAMA DE CLASSE - UML

Cachorro

+nome: string
+idade: int
+tamanho: int
+cor: string
+peso: int

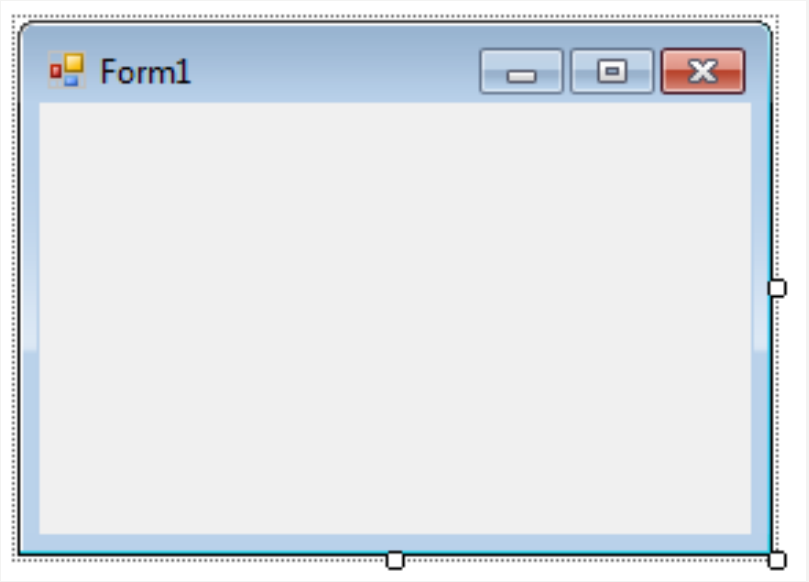
+Correr(velocidade: int): void
+Comer(): void

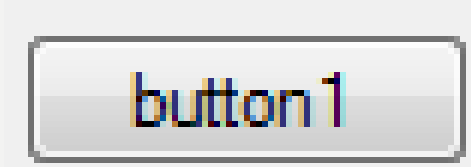
Nome da classe

Atributos

métodos

ANALOGIA – CLASSES E FORMULÁRIOS





Form1

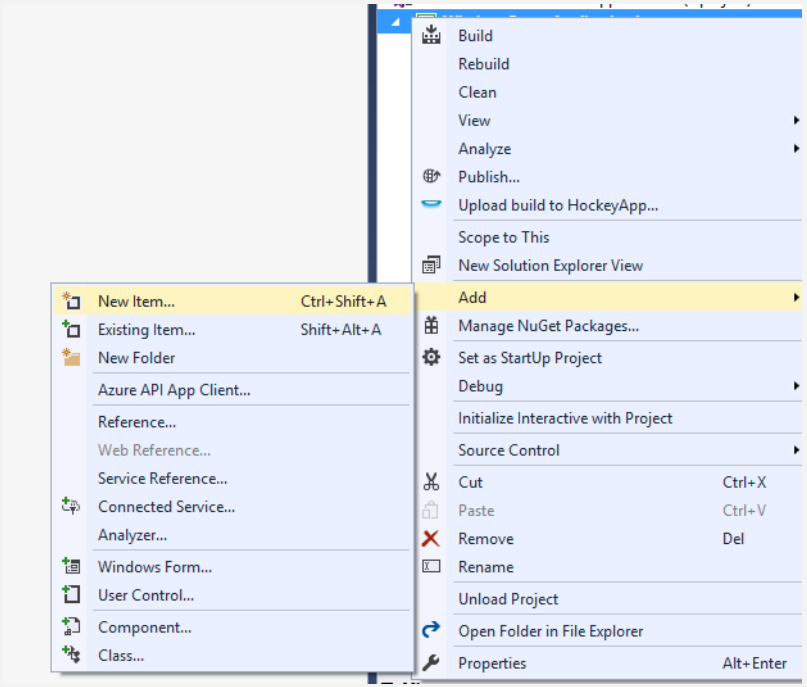
+titulo: string
+tamanho: int
+cor: string

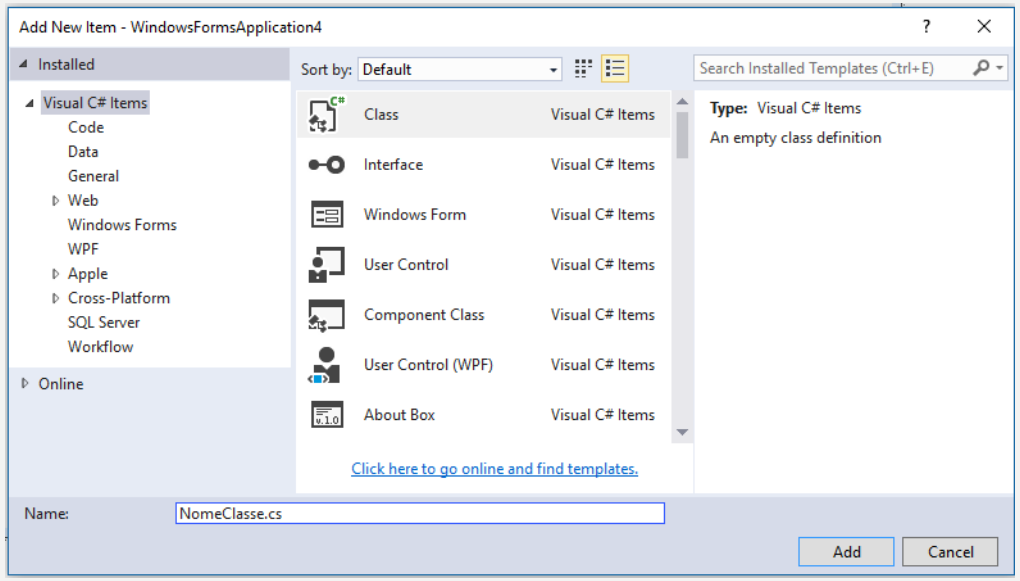
Botao

+titulo: string
+tamanho: int
+cor: string

CRIAR UMA CLASSE EM C#

- Para criar uma classe em C# é necessário adicionar um novo item.
- Clique com o botão direito sobre o nome do projeto e escolha a opção add > New Item
- Na janela abaixo escolha CLASS e determine seu





MINHAS CLASSES EM C#

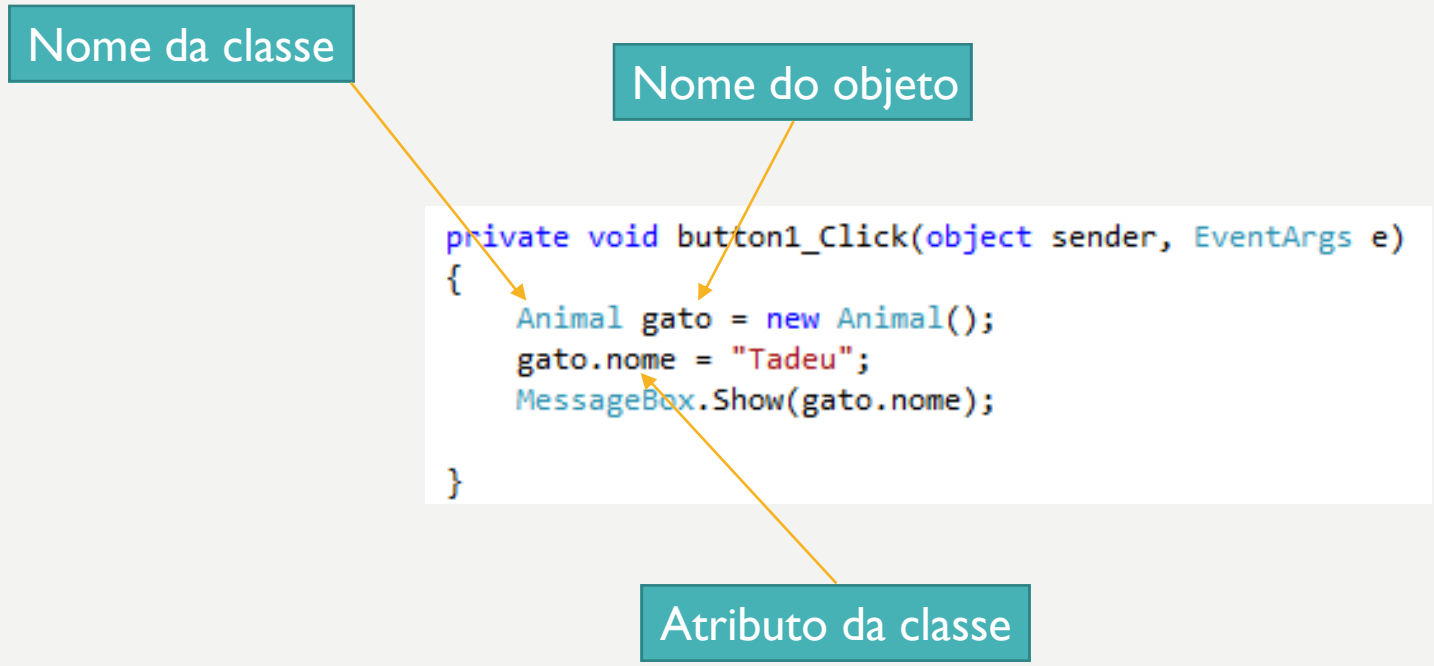
Animal

+nome: string
+idade: int
+tamanho: int
+cor: string
+peso: int

+Correr(velocidade: int): void
+Comer(): void

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication4
{
    class Animal
    {


        public string nome;
        public int idade;
        public int tamanho;
        public string cor;
        public int peso;
        public void Correr(int velocidade)
        {

        }
        public void comer()
        {

        }

    }
}
```

UTILIZANDO MINHAS CLASSES



CONSTRUTOR

Mediador de acesso público

Nome da classe

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApplication4
{
    class Animal{
        public string nome;
        public int idade;
        public int tamanho;
        public string cor;
        public int peso;
        public Animal(){
            this.nome = "Tadeu";
        }
        public void Correr(int velocidade){

        }
        public void comer(){

        }
    }
}
```

Qual valor será mostrado na mensagem?

```
private void button1_Click(object sender, EventArgs e)
{
    Animal gato = new Animal();
    MessageBox.Show(gato.nome);

}
```

SOBRECARGA DE MÉTODOS

- O método na classe pai deve ser marcado com virtual.
- O método na classe filha deve ser marcado com override.

Classe pai

```
class Conta
{
    private float saldo;
    //pode sobre sobrecarga
    public virtual void Debito(float valor) {
        Saldo += valor;
    }

    public float Saldo
    {
        get
        {
            return saldo;
        }

        set
        {
            saldo = value;
        }
    }
}
```

Método que pode sofrer sobrecarga

Classe filha

```
class Poupanca:Conta
{
    public override void Debito(float valor)
    {
        float taxa= 0.01f;
        base.Debito(valor - taxa);
    }
}
```

Sobrecarga do método Debito

MEDIADORES DE ACESSO

- Public
  - Todas as outras classes e funções tem acesso ao atributo na classe

- Private
  - Apenas a classe possui acesso a seus atributos
  - Usado como medida de segurança
  - Força o desenvolvedor a criar métodos de acesso aos atributos da classe.

GET E SET

- Os métodos GET e SET são utilizadas para gerenciamento do o acesso dos atributos de uma classe.
- Nesses métodos determinamos quando um determinado atributo poderá ser acessado.
- Permite um código limpo e padronizado.

OS MÉTODOS GET E SET

```
class Animal{
    private string nome;
    private int idade;
    private int tamanho;
    private string cor;
    private int peso;
    public string Nome
    {
        get
        {
            return nome;
        }

        set
        {
            value = value.ToUpper();
            nome = value;
        }
    }
   public Animal(){
         this.nome = "Tadeu";
    }
    public void Correr(int velocidade){

    }
    public void comer(){

    }
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    Animal gato = new Animal();
    gato.Nome = "Tadeu";
    MessageBox.Show(gato.Nome);

}
```

VETOR DE OBJETOS

```
private void button1_Click(object sender, EventArgs e)
{
    Animal[] gato = new Animal[100];
    gato[0] = new Animal();
    gato[0].Nome = "Tadeu";
    MessageBox.Show(gato[0].Nome);

}
```

HERANÇA

Classe Filha

```
class Cachorro : Animal
{
    private string raca;

    public string Raca
    {
        get
        {
            return raca;
        }

        set
        {
            raca = value;
        }
    }
}
```

Classe Pai

```
class Animal{
    private string nome;
    private int idade;
    private int tamanho;
    private string cor;
    private int peso;
    public string Nome
    {
        get
        {
            return nome;
        }

        set
        {
            value = value.ToUpper();
            nome = value;
        }
    }
    public Animal(){
        this.nome = "Tadeu";
    }
    public void Correr(int velocidade){

    }
    public void comer(){

    }
}
```

CONSTRUTORES DE CLASSES HERDADAS

- O construtor da classe **filha** sempre chama o construtor da classe **pai** e depois executa o seu próprio código;

```
class Animal{
    private string nome;
    private int idade;
    private int tamanho;
    private string cor;
    private int peso;
    public string Nome
    {
        get
        {
            return nome;
        }

        set
        {
            value = value.ToUpper();
            nome = value;
        }
    }
   public Animal(){
         this.nome = "Tadeu";
    }
    public void Correr(int velocidade){

    }
    public void comer(){

    }
}
```

```
class Cachorro : Animal
{
    private string raca;
    public Cachorro()
    {
        this.Raca = "Vira Lada";
    }
    public string Raca
    {
        get
        {
            return raca;
        }

        set
        {
            raca = value;
        }
    }
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    Cachorro[] dog = new Cachorro[100];
    dog[0] = new Cachorro();

    MessageBox.Show(dog[0].Nome + " - " + dog[0].Raca);

}
```

Qual valor será apresentado na Caixa de mensagem?

MEDIADORES DE ACESSO

- O modificador de acesso **protected** torna a variável de uma classe base somente acessível as **suas classes derivadas**; (outras classes não acessam a variável).

CRIANDO OBJETOS VISUAIS EM TEMPO DE EXECUÇÃO

```
private void button1_Click(object sender, EventArgs e)
{
    Button b = new Button();
    b.Width = 100;
    b.Height = 100;
    b.Text = "Eu sou um botão";
    b.Location = new System.Drawing.Point(0, 0);
    b.Click += new System.EventHandler(this.meuBotao_Click);
    this.Controls.Add(b);
}

private void meuBotao_Click(object sender, EventArgs e)
{
    MessageBox.Show("Ola mundo");
}
```