**Herança**

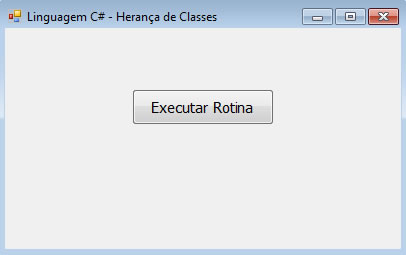
  
Crie uma aplicação “Windows Forms” e adicione um botão no formulário. A tela deverá ficar idêntica a Imagem 02.

Imagem 02. Formulário Padrão.

Para criação de classes seguimos as dicas contidas no mês anterior, clicando com o botão direito e escolhendo “Add/Add New Item.../Class”. Defina a classe como Pessoa e clique em Adicionar. Esta será a nossa “SuperClasse”. Para fins didáticos defino alguns atributos e métodos e o uso do operador “virtual” no Método “ExecutaMetodo”, o qual será sobrescrito posteriormente na “SubClasse”.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace heranca

{

public class Pessoa

{

private string nome;

private string cpf;

private string rg;

public string Nome

{

get { return nome; }

set { nome = value; }

}

public string Cpf

{

get { return cpf; }

set { cpf = value; }

}

public string Rg

{

get { return rg; }

set { rg = value; }

}

public void CadastrarPessoa()

{

MessageBox.Show("Cadastrando Pessoa!");

}

public virtual void ExecutarMétodo()

{

MessageBox.Show("Executando o método da Classe Pessoa!");

}

}

}

Para criar a subclasse utilize os mesmos passos descritos acima. A subclasse será denominada “Funcionario”. Herdaremos todas as funcionalidades da superclasse Pessoa, ou seja, os atributos Nome, CPF e RG e os métodos CadastrarPessoa() e ExecutarMetodo().

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace heranca

{

public class Funcionario : Pessoa

{

public Funcionario()

{

}

~Funcionario()

{ }

private int codigo;

private string tipo;

public int Codigo

{

get { return codigo; }

set { codigo = value; }

}

public string Tipo

{

get { return tipo; }

set { tipo = value; }

}

public void TarefasFuncionario()

{

MessageBox.Show("Executando Tarefas!");

}

public override void ExecutarMétodo()

{

base.ExecutarMétodo();

MessageBox.Show("Executando o Método da Classe Funcionário!");

}

}

}

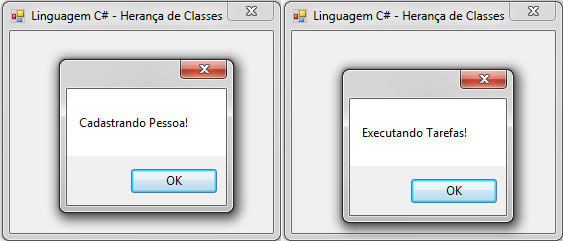
Foram adicionados alguns atributos como Nome e Código e um método “TarefasFuncionario”. Estamos sobrescrevendo o método ExecutarMétodo(), como foi explicado anteriormente. O operador “base” servirá pra reutilizar o método ExecutarMétodo() da classe Pessoa. Principal trecho do código abordado: public class Funcionario: Pessoa Para executar o exemplo utilize o código:

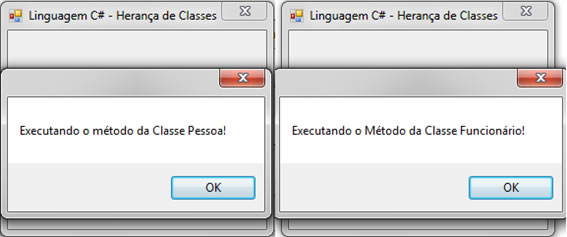
private void button1\_Click(object sender, EventArgs e)  
{  
      Funcionario func = new Funcionario();  
      func.Nome = "Thiago Montebugnoli";  
      func.Cpf = "998.899.999-899";  
      func.Rg = "99.888.777-6";

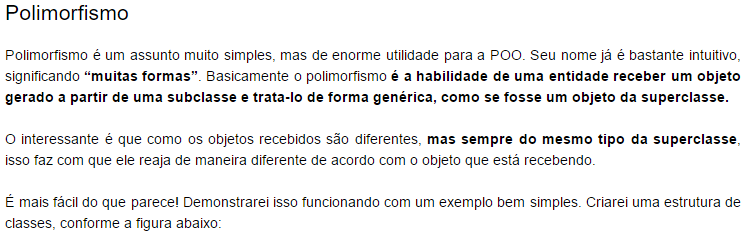
      func.Codigo = 1000;  
      func.Tipo = "Desenvolvedor";

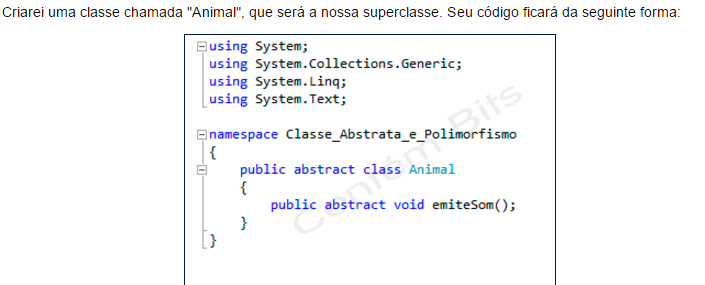
func.CadastrarPessoa();  
     func.TarefasFuncionario();  
     func.ExecutarMétodo();  
}

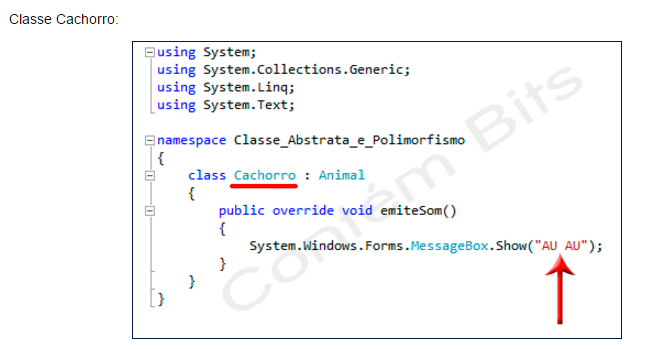
No código criado anteriormente primeiramente instancio a classe “Funcionario” e logo em seguida preenchemos os atributos Nome, Cpf e Rg. Percebam que estes atributos pertencem a classe “Pessoa”, a qual herdamos. Já os atributos Codigo e Tipo foram implementados na classe “Funcionario”. O método “CadastrarPessoa()” é originário da classe Base. O método “TarefasFuncionario()” foi implementado na própria classe “Funcionario”. Utilizamos o “Override” para o método ExecutarMetodo(). Ver Figura 03 e 04. 

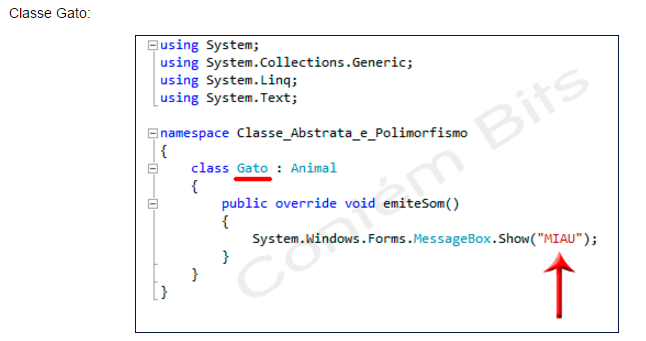
  
Imagem 03. Exemplo em “Run-Time”. (Método da classe Pessoa e da classe Funcionário)

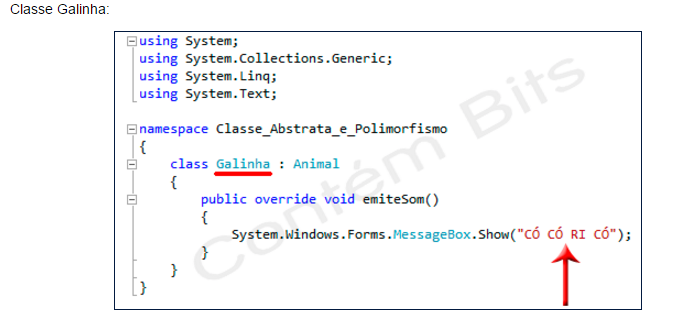
  
Imagem 04. Exemplo em “Run-time”. (Override de Método)

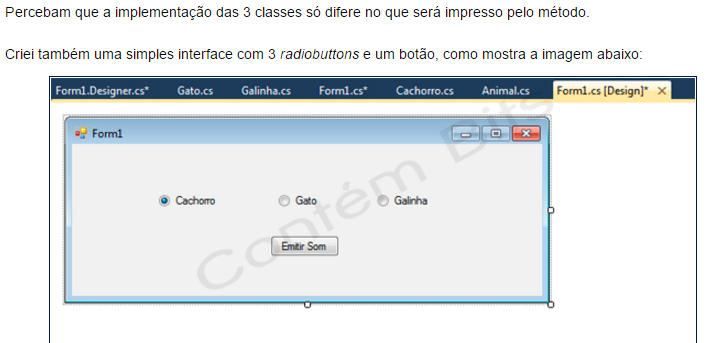


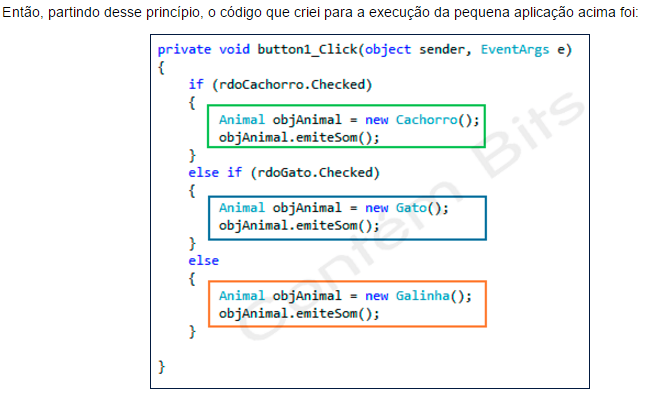




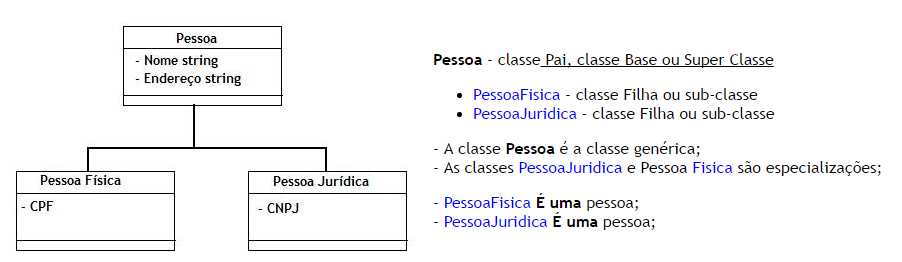








**EXERCICÍOS DE HERANÇA E POLIMORFISMO**

****

1. Crie a herança.
2. Crie um método para mostrar uma mensagem “Pessoa Física”, “Pessoa Jurídica”, utilizando os recursos do polimorfismo.