POLIMORFISMO MVC / POO

29/10/2021

- Override - Sobrepor
- Overload - Sobrecarga
- Virtual - Virtual
- Abstratc – Resumo
- Inherited - Herdada

IREMOS CONHECER ANTES DE USAR POLIMORFISMO AS DIRETIVAS.

```
Tpessoa = class   //classe Pai
      public
      function trabalhar : string;
      constructor create;
      destructor destroy;override;

    end;
    TFilho = class
      public
      function trabalhar :string;

      constructor create;
      destructor Destroy; override;
    end;
    Tsobrinho = class
    public
    function trabalhar :string;
    constructor create;
    destructor destroy; override;
    end;
```

Criamos a classe TFilho ainda simples que não herda nada
Criamos a classe TSobrinho também simples não herda.

Classe: TFilho

As duas classes Receberam
a function trabalhar do tipo string
A mesmo usada na classe pai (Tpessoa).
Que herdaram da mesma.

Classe: TSobrinho

VAMOS CRIAR 2 NOVAS CLASSES PARA EXEMPLIFICAR O POLIMORFISMO

DevTeam

```
Tpessoa = class   //classe Pai
      public
      function trabalhar : string; virtual;
      constructor create;
      destructor destroy;override;

    end;
    TFilho = class (Tpessoa)
      public
      function trabalhar :string; Overload;

      constructor create;
      destructor Destroy; override;
    end;
    Tsobrinho = class (Tpessoa)
    public
    function trabalhar :string; Overload;
    constructor create;
    destructor destroy; override;
    end;
```

Virtual – ele aceita ser sobrecarregado

Sobrecarga – vai ser sobrecarregado

Sobrecarga – vai ser sobrecarregado

VAMOS CRIAR 2 NOVAS CLASSES PARA EXEMPLIFICAR O POLIMORFISMO

DevTeam

```
Tpessoa = class   //classe Pai
    public
      Nome:String;
      Trabalho:String;
      Salario: Double;
    function trabalhar : string; virtual;
    constructor create;
    destructor destroy;override;

  end;
  TFilho = class (Tpessoa)
    public
    function trabalhar :string; Overload;

    constructor create;
    destructor Destroy; override;
  end;
  Tsobrinho = class (Tpessoa)
  public
  function trabalhar :string; Overload;
  constructor create;
  destructor destroy; override;
  end;
```

Observe os atributos da classe Tpessoa (Pai)

Todas as classes tem o mesmo método Trabalhar.

Agora estão herdado de Tpessoa

Todas as classes tem o mesmo método Trabalhar.

Agora estão herdado de Tpessoa

Todas as classes tem o mesmo método Trabalhar.

Iremos criar os métodos para todos Ctrl + Shift + C

DevTeam

```
constructor Tpessoa.create;
begin
   inherited create;
end;

destructor Tpessoa.destroy;
begin
    inherited;
end;

function Tpessoa.trabalhar: string;
begin
  result:= 'sou funcionario publico'+trabalho;
end;

{ TFilho }

constructor TFilho.create;
begin
  inherited create
end;

destructor TFilho.Destroy;
begin
  inherited;
end;
```

```
function TFilho.trabalhar: string;
begin
   result:='Sou funcionario federal'+trabalho;
end;

{ Tsobrinho }

constructor Tsobrinho.create;
begin
  inherited create;
end;

destructor Tsobrinho.destroy;
begin
   inherited;
end;

function Tsobrinho.trabalhar: string;
begin
  result:=' Sou funcionario Estadual'+trabalho;
end;
```

Veja que cada Result recebe uma mensagem diferente da outra.

```
procedure TForm1.btn1Click(Sender: TObject);
var
pessoa:Tpessoa;

begin
  pessoa:=Tpessoa.create;

    try
    pessoa.nome:='Luis Carlos';
    pessoa.Salario:=1200;
    pessoa.trabalhar;

  finally

   FreeAndNil(pessoa);
  end;
```

Variável pessoa: da classe pessoa

Instância da classe pessoa

Os atributos da classe pessoa

Destruindo a classe pessoa

IREMOS VOLTAR PARA NOSSA FORMULÁRIO PRINCIPAL E EXEMPLIFICAR NA PRÁTICA O POLIMORFISMO.

DevTeam

```
procedure TForm1.btn1Click(Sender: TObject);
var
pessoa:Tpessoa;
filho:TFilho;
sobrinho:Tsobrinho;

begin
  pessoa:=Tpessoa.create;
  filho:=TFilho.create;
  sobrinho:=Tsobrinho.create;
 try

 finally
   FreeAndNil(pessoa);
   FreeAndNil(sobrinho);
   FreeAndNil(filho);

 end;
```

Criamos as variáveis

Instanciamos as variáveis

destruímos as variáveis

```
begin
  pessoa:=Tpessoa.create;
  filho:=TFilho.create;
  sobrinho:=Tsobrinho.create;
 try
    pessoa.nome:='Luis Carlos';
    pessoa.Salario:=800;


    filho.nome:='Genival Pereira';
    filho.Salario:=15000;

    sobrinho.nome:='Fabio Pessoa';
    sobrinho.Salario:=6000;

 finally
   FreeAndNil(pessoa);
   FreeAndNil(sobrinho);
   FreeAndNil(filho);
```

Atribuição

Acrescentamos informações ao atributo (pessoa)

Acrescentamos informações ao atributo (filho)

Acrescentamos informações ao atributo (sobrinho)

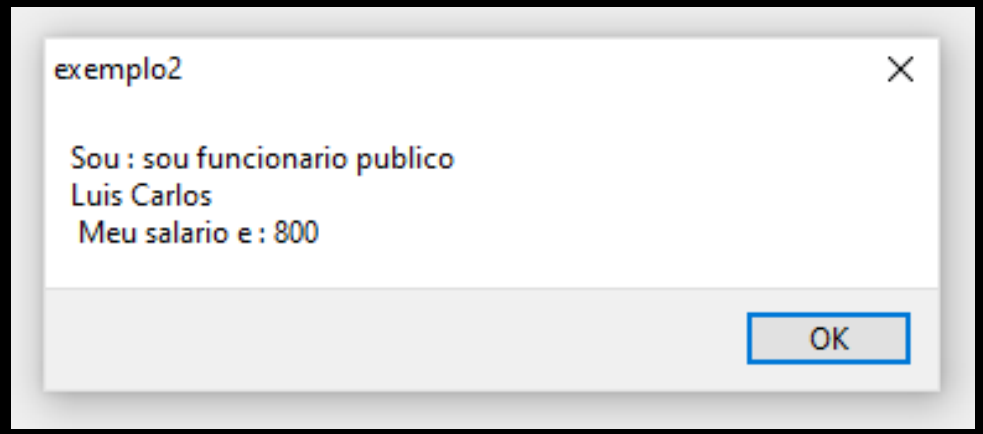
DevTeam

Importante observar que cada function tem o mesmo método trabalhar, quem faz essa chamada
É o overload, porque a classe principal tem o método virtual dando essa permissão.

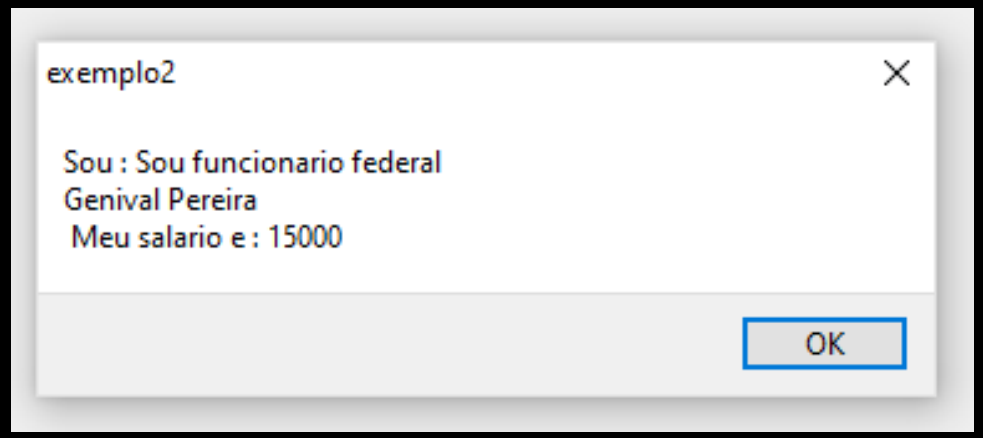
```
  //classe pai Pessoa - com a função trabalhar
 ShowMessage('Sou : '+pessoa.trabalhar +#13
                     +pessoa.nome +#13+ ' Meu salario e : '
                     +(FloatToStr(pessoa.Salario )));

 //classe filha herda- filho - com a mesma função trabalhar
ShowMessage('Sou : '+filho.trabalhar  +#13
                    +filho.nome +#13+ ' Meu salario e : '
                    +(FloatToStr(filho.Salario)));

//classe filha herda- sobrinho - com a mesma função trabalhar
   ShowMessage('Sou : '+sobrinho.trabalhar  +#13
                       +sobrinho.nome +#13+ ' Meu salario e : '
                       +(FloatToStr(sobrinho.Salario)));
```

Veja a function trabalhar sendo utilizada Em todos os showmessage

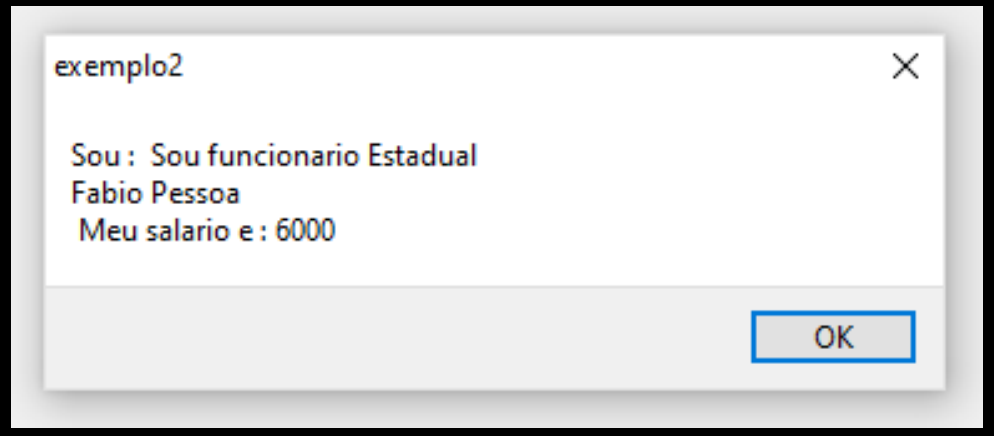
IREMOS AGORA CHAMAR AS FUNCIONALIDADE UTILIZANDO O MESSAGE



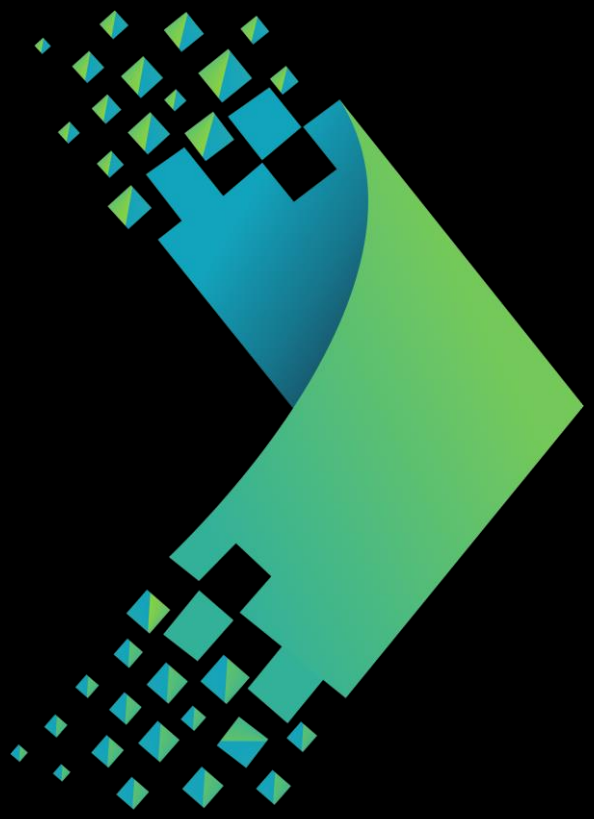
Classe Pessoa



Classe Filho



Classe Sobrinho



ATÉ A PRÓXIMA.....