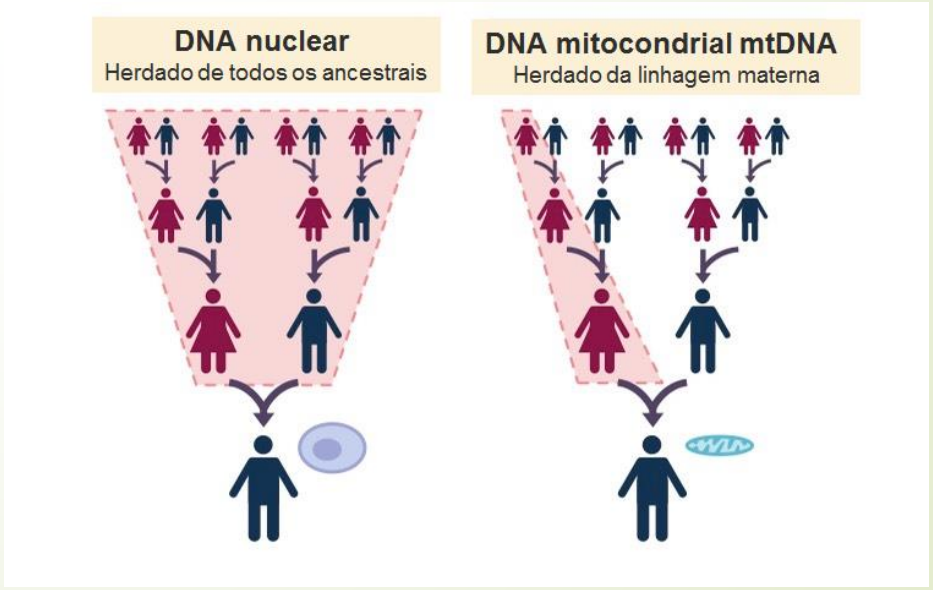
Programação Orientada a Objeto
POO – MVC

Herança e Polimorfismo

Herança : ação de herdar, de adquirir por sucessão.

- Iremos utilizar o mesmo projeto anterior para não fugir do escopo do projeto simples e já entendido por todos.
- Já criamos a abstração e creio que já entendido depois fomos para o encapsulamento como proposto e também vejo que isso já faz parte do nosso cotidiano da programação MVC daqui em diante vamos para a Herança como na definição acima diz. Vamos herdar.....



DevTeam

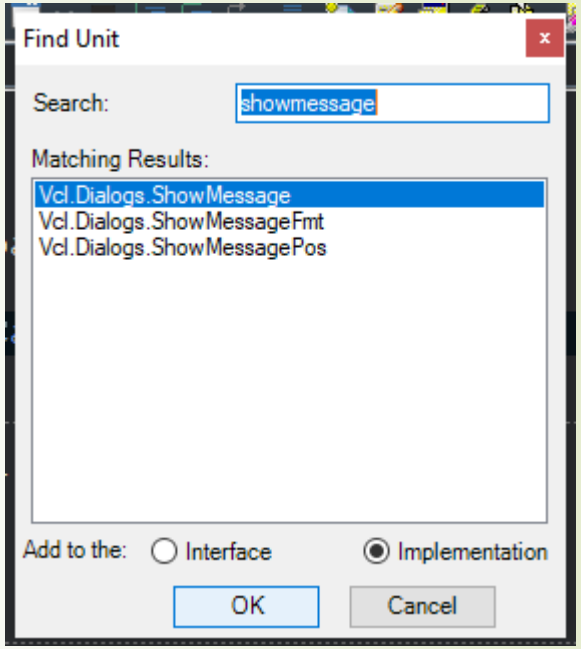
Iremos criar 4 métodos na classe Tpessoa

```
public

property nome : string read getNome write Setnome;
property endereco : string read Fendereco write Fendereco;
property cidade: string read Fcidade write Fcidade;

//Quatro médotos criados para nossa HERANÇA

procedure andar;
procedure correr;
function  falar:string;
function  comer:string;
```

Ctrl + Shift + C

Vamos implementar os métodos

```
procedure Tpessoa.andar;
begin
  showmessage('Caminhar...');
end;
function Tpessoa.comer: string;
begin
   result:='comer sanduiche';
end;
procedure Tpessoa.correr;
begin
    ShowMessage('correr 10 km');
end;
function Tpessoa.falar: string;
begin
     result:='falaremos até o infinito';
end;
```



Ctrl + Shift + A

Quando você colocar a função Message na sua classe irá reclamar uma uses de dialogs. Pressione as teclas Ele será encontrado.

DevTeam

Vamos para outra questão de herança.

- Para podemos herdar alguma coisa tem que ser de alguém e a nossa classe é **Tpessoa** é a classe **(Pai).**
- **Quem será que irá herda da classe pai – classe Filha (filha).**

Pai

HERDAR

Nome : string;
Endereco : string;
Correr : string;
Comer : string;
Falar : string;
Andar : string;

Filho

DevTeam

Vamos para outra questão de herança.

- O pai é um contador e é uma pessoa. O filho é médico também é uma pessoa ambas com as mesmas características pessoas, nome, endereço, em tem função e procedimentos como falar, correr, falar e etc.
- **Mais apenas o medico possui – CRM que não herda do pai. Correto!**

Pai

HERDAR

Nome : string;
Endereco : string;
Correr : string;
Comer : string;
Falar : string;
Andar : string;

Filho

DevTeam

Iremos para a classe pai (Tpessoa)

```
unit U_classepessoa;

interface
type
    Tpessoa = class    //classe Pai
      private

        Fendereco:string;
        Fcidade:string;
        Fnome : string;
        function getNome: string;
        procedure Setnome(const Value: string);
```

Sabendo que a classe pai é a classe principal.

Iremos criar uma outra classe. Essa será a classe:
Filha (Tmedico)onde a mesmo herdará do pai seus métodos

Criando a classe (Tmedico) simples.

```
end;
//Veja que criamos uma nova classe (Tmedico)

TMedico = class
private

public

constructor create;
destructor destroy;override;
end;
```

Ainda não é filha

Criamos a classe Tmedico mais ela não passa simplesmente de uma classe normal
Sem herda nada de ninguém, nem mesmo tem métodos ou procedures ou funções.
E para que isso aconteça é preciso dizer a ela que precisamos da classe (pai)

DevTeam

Criando a herança entre as classes(Tmedico)

```
end;
//Agora sim dissemos a ela para herdar da classe(TPessoa)

TMedico = class (Tpessoa) //Tpessoa - classe Pai
private

public

constructor create;
destructor destroy;override;
end;
```

Agora sou é filha

Agora veja bem o que significa essa herança no contexto de filha que herda de pai. Neste momento tudo mais tudo mesmo que tiver na classe pai será mostrada na Classe filha (Tmedico), vejamos na prática isso. Agora iremos para nosso formulário Principal e vamos instanciar a classe criada e mostrar sua herança. Mesmo sem existir uma Linha de código na classe filha(Tmedico).

DevTeam

Observe atentamente...

```
procedure TForm1.btn1Click(Sender: TObject);
var
pessoa:Tpessoa; // criando uma variável do Tipo Tpessoa (classe)
medico:TMedico; //criando a variavel do tipo Tmedico (classe filha)
```

Veja a variável medico.

```
begin
  pessoa:=Tpessoa.create; //estamos Instanciando (chamando-criando)

  try //tratativa --Estudaremos mais a diante.

      pessoa.nome:='Luis carlos';
      pessoa.endereco:='Rua São Francisco, 23';
      pessoa.cidade:='Caxias-MA';

      medico.

  finally
      FreeAndN
```

```
constructor  create;
destructor   Destroy;
property     nome: string;
property     endereco: string;
property     cidade: string;
procedure    andar;
procedure    correr;
```

TMedico Constructor - U_classepessoa.pas (37,17)

Declared in U_classepessoa.TMedico

41: 14 | Insert

Estamos herdando todos os
Métodos, function, procedures
Da classe Pai

DevTeam

Ainda não acabou!!!

```
procedure TForm1.btn1Click(Sender: TObject);
var
pessoa:Tpessoa; // criando uma variável do Tipo Tpessoa (classe)
medico:TMedico; //criando a variavel do tipo Tmedico (classe filha)

begin
  pessoa:=Tpessoa.create; //estamos Instanciando (chamando-criando)

  try //tratativa --Estudaremos mais a diante.

     pessoa.nome:='Luis carlos';
     pessoa.endereco:='Rua São Francisco, 23';
     pessoa.cidade:='Caxias-MA';

     //Veja que o médico tem outros dados (informações)
     medico.nome:='Dr. Maxiwel';
     medico.endereco:='codominio lages';
     medico.cidade:='Brasilia';
```

Era só isso????

Mais o medico tem suas funções e métodos e procedures

O médico tem as mesmas características de pessoas

Nome:

Endereço:

Telefone:

Cidade:

Mais vai existe características que só o medico tem

CRM

Especialização

.....

Ae é onde entra as próprias propriedade do médico.

Onde entra o que falamos o encapsulamento.

DevTeam

Estamos dando vida própria a classe Tmedico.

```
end;
//Agora sim dissemos a ela para herdar da classe(TPessoa)

TMedico = class (Tpessoa) //Tpessoa - classe Pai
private

FCRM:string; //campo do tipo String (CRM)

public
//crie os metodos GET e Set do CRM depois -Ctrl+Shift +C
property CRM: string read getCRM write SetCRM;

constructor create;
destructor destroy;override;
end;
```

Ctrl + Shift + C

Criamos uma variável CRM do tipo String ela será usada na classe Tmedico somente a ela será atribuída esse funcionalidade.

DevTeam

Feito isso, Iremos retornar ao Formulário.

```
begin
  pessoa:=Tpessoa.create; //estamos Instanciando (chamando-criando)
  medico:=TMedico.create; //Estamos instanciando classe medico

  try //tratativa --Estudaremos mais a diante.

      pessoa.nome:='Luis carlos';
      pessoa.
      pessoa.

      //Veja                          s dados (informações)
      medico.
      medico.                         ages';
      medico.
      medico.
```

property	**CRM**: string;
constructor	**create**;
destructor	**Destroy**;
property	**nome**: string;
property	**endereco**: string;
property	**cidade**: string;
procedure	**andar**;

CRM Property - U_classepessoa.pas (42,14)

CRM - System.string

Observe que todos as propriedade da classe pessoa e

DevTeam

Observe que a classe pessoa não...

```
begin
  pessoa:=Tpessoa.create; //estamos Instanciando (chamando-criando)
  medico:=TMedico.create; //Estamos instanciando classe medico

  try //tratativa --Estudaremos mais a diante.

      pessoa.nome:='Luis carlos';
      pessoa.endereco:='Rua São Francisco, 23';
      pessoa.cidade:='Caxias-MA';
      pessoa.cr

      //Veja
      medico.
      medico.
      medico.
      medico.
```

constructor **create;**

Encontra o CRM
que é
apenas da
classe Tmedico

Tpessoa Constructor - U_classepe 27,19)

Declared in U_classepessoa.Tpessoa

ages';

Agora podemos mostra o resultado.

```
pessoa:=Tpessoa.create; //estamos Instanciando (chamando-criando)
medico:=TMedico.create; //Estamos instanciando classe medico

try //tratativa --Estudaremos mais a diante.

    pessoa.nome:='Luis carlos';
    pessoa.endereco:='Rua São Francisco, 23';
    pessoa.cidade:='Caxias-MA';


    //Veja que o médico tem outros dados (informações)
    medico.nome:='Dr. Maxiwel';
    medico.endereco:='codominio lages';
    medico.cidade:='Brasilia';
    medico.CRM:='3056 MA';

finally
    FreeAndNil(medico);
    FreeAndNil(pessoa);
```

Não esqueça de Destruir

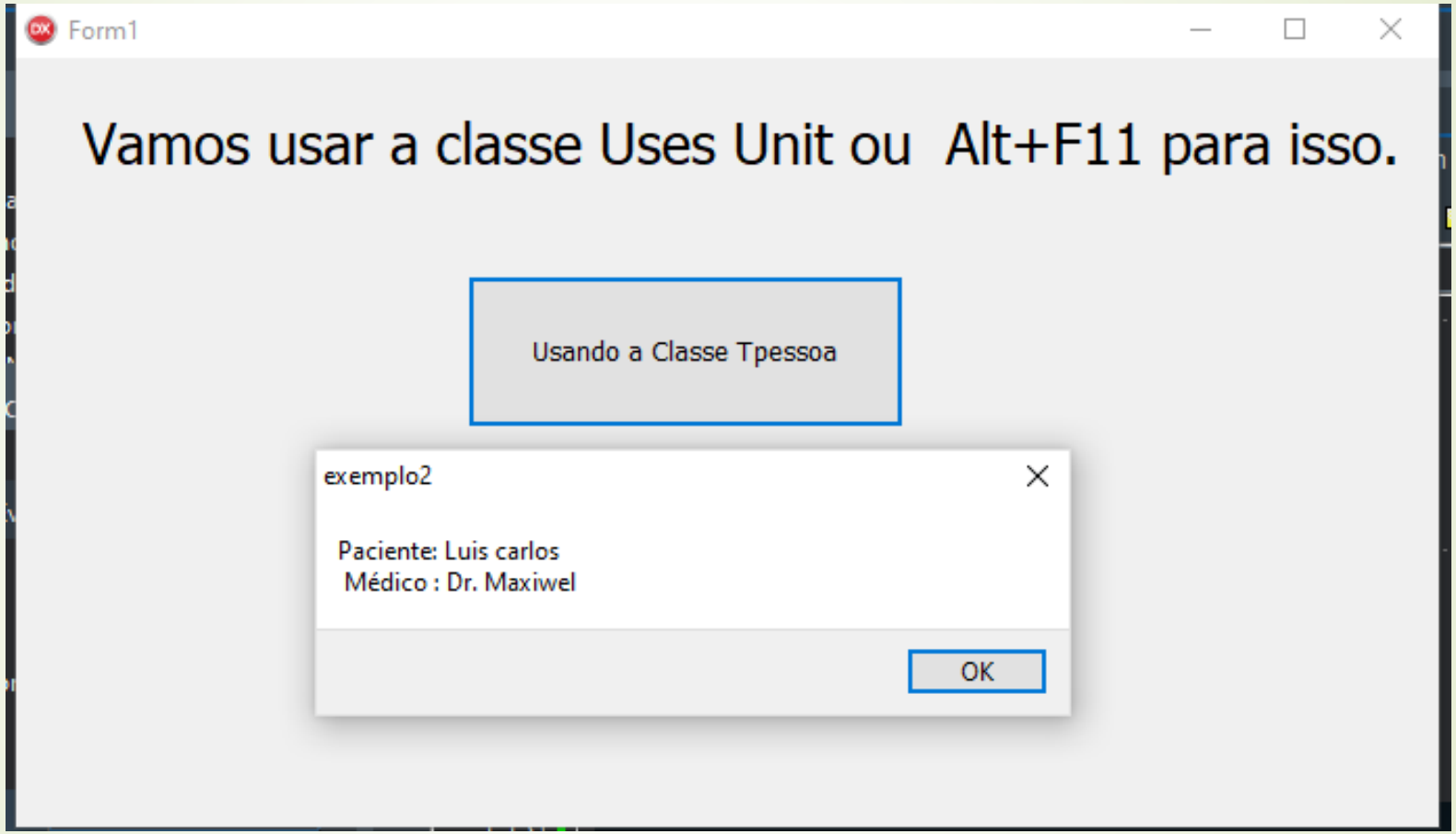
DevTeam

Agora pronto!

```
    pessoa.nome:='Luis carlos';
    pessoa.endereco:='Rua São Francisco, 23';
    pessoa.cidade:='Caxias-MA';

    ShowMessage('Paciente:' + pessoa.nome);

   //Veja que o médico tem outros dados (informações)
    medico.nome:='Dr. Maxiwel';
    medico.CRM:='3056 MA';

    ShowMessage('CRM' + medico.CRM);

    ShowMessage('Paciente: ' + pessoa.nome + #13 + ' Médico : '+ medico.nome);

finally
   FreeAndNil(medico);
   FreeAndNil(pessoa);
end;
```

DevTeam

Entendido sobre herança...simples assim.



POLIMORFISMO PROXIMO...

DevTeam