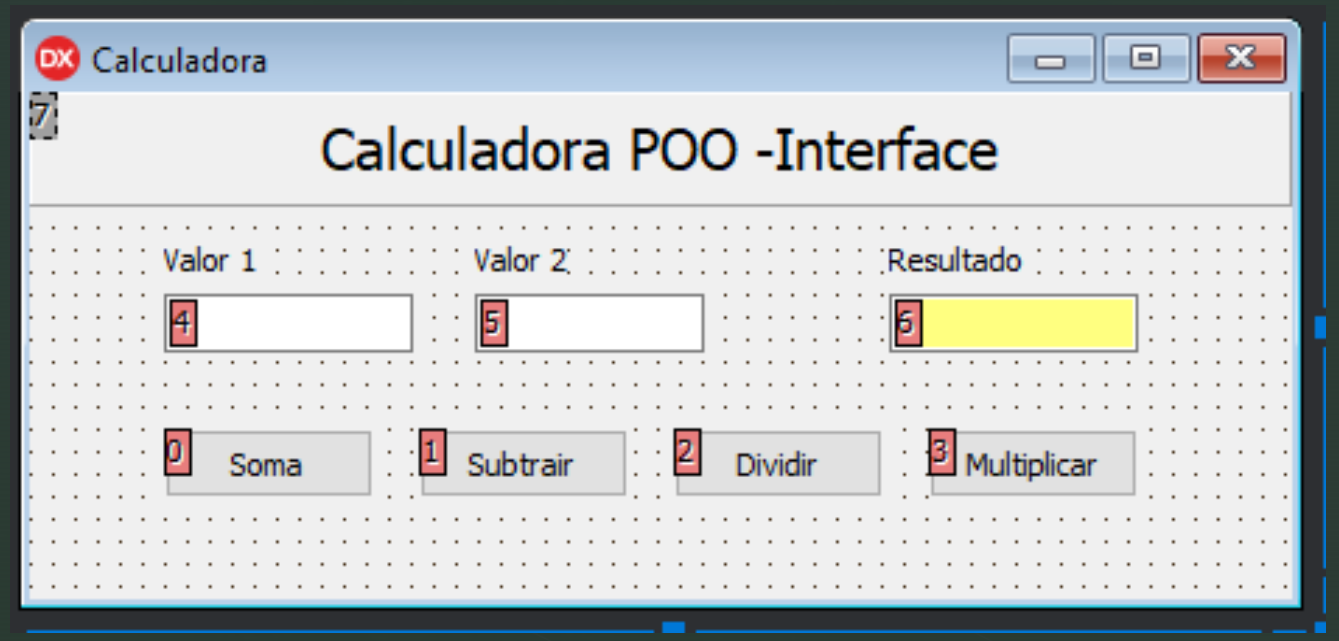
INTERFACE – MVC /POO

Alta Coesão e um Baixo Acoplamento/ Principio do Aberto e Fechado.

Criaremos um novo projeto simples.



3 – edits
4 – bottons
3 – label
1 – panel

Componentes

DevTeam

Agora criaremos nossa classe do tipo interface

```
unit Classe.calculadora;

interface
type

  icalculadora = interface //criado nossa classe interface...


  end;
```

Não é só isso!!!!

Quando trabalhamos com interface nos utilizamos o RAC – contador de automático de referencia
Precisamos criar uma assinatura para todas as interface que você criar.

```
icalculadora = interface //criado nossa classe interface...
['{2543D0A2-7BCC-4655-AE64-ADDD6754EEA9}']

end;
```

Ctrl + Shift + G

Ele cria uma assinatura automática

DevTeam

Criaremos uma função para nossa interface

```
interface
type

  icalculadora = interface //criado nossa classe interface...
  ['{2543D0A2-7BCC-4655-AE64-ADDD6754EEA9}'] //assinatura

  function Operacao(num1,num2 : double ) :Double ;

  end;

implementation
```

Função para calcular 2 números.

Para melhor entendimento é preciso conhecer as funcionalidade bem simples
Assim podemos crescer no andamento do processo. Vamos primeiro entender
Cada situação em sua visão própria. Depois vamos dar vida a nossa imaginação

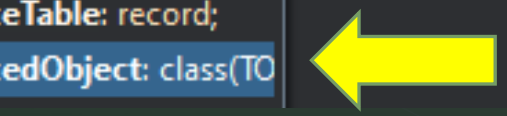
Observações importantes

DevTeam

Na classe Tsoma observe o que será inserido

```
interface
type

  icalculadora = interface //criado nossa classe interface...
  ['{2543D0A2-7BCC-4655-AE64-ADDD6754EEA9}'] //assinatura

  function Operacao(num1,num2 : double ) :Double ;

  end;
  Tsoma = class(Tinter

implementation
```

```
type    TInterfaceEntry: record;
type    TInterfaceTable: record;
type    TInterfacedObject: class(TO
```



Temos que herdar de TinterfacedObject

```
end;
Tsoma = class(TInterfacedObject, icalculadora)
```

Pegamos a função e adicionamos na nossa classe e já criamos o método.

```
function Operacao(num1,num2 : double ) :Double ;

end;
Tsoma = class(TInterfacedObject, icalculadora)
  function Operacao(num1,num2 : double ) :Double ;
end;
```

Ctrl + Shift + C

```
function Tsoma.Operacao(num1, num2: double): Double;
begin
  result:= num1 + num2;
end;
```

Daqui em diante é criamos as outras classes herdando de interfacedObject
E correr para o abraço.....não é apenas isso....

DevTeam

Vamos raciocinar juntos....

```
end;
Tsoma = class(TInterfacedObject, icalculadora)
  function Operacao(num1,num2 : double ) :Double ;
end;
Tdubtrair = class(TInterfacedObject, icalculadora)
  function Operacao(num1,num2 : double ) :Double ;
end;
Tdividir = class(TInterfacedObject, icalculadora)
  function Operacao(num1,num2 : double ) :Double ;
end;
Tmultiplicar = class(TInterfacedObject, icalculadora)
  function Operacao(num1,num2 : double ) :Double ;
end;
```

Vamos deixar como manda o processo.

```
{ Tsubtrair }
function Tsubtrair.Operacao(num1, num2: double): Double;
begin
    result:= num1 - num2;
end;

{ Tdividir }
function Tdividir.Operacao(num1, num2: double): Double;
begin
  result:= num1 / num2;
end;

{ Tmultiplicar }
function Tmultiplicar.Operacao(num1, num2: double): Double;
begin
    result:= num1 * num2;
end;
```

subtrair

dividir

multiplicar

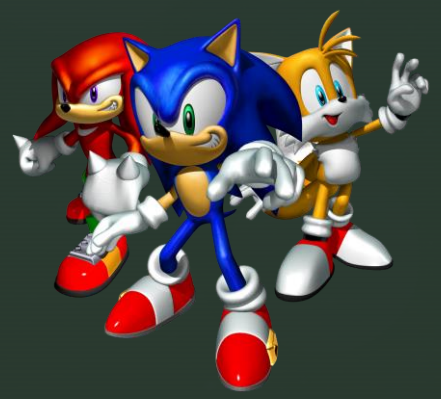
DevTeam

Iremos implementar a divisão que não pode ser pode 0

```
{ Tdividir }
function Tdividir.Operacao(num1, num2: double): Double;
begin
   if num2 <=0 then
   raise Exception.Create('O numero nao pode ser menor que zero');
  result:= num1 / num2;
end;
```

Quando o exception.create reclamar sua classe, pressione CTRL + SHIFT + A

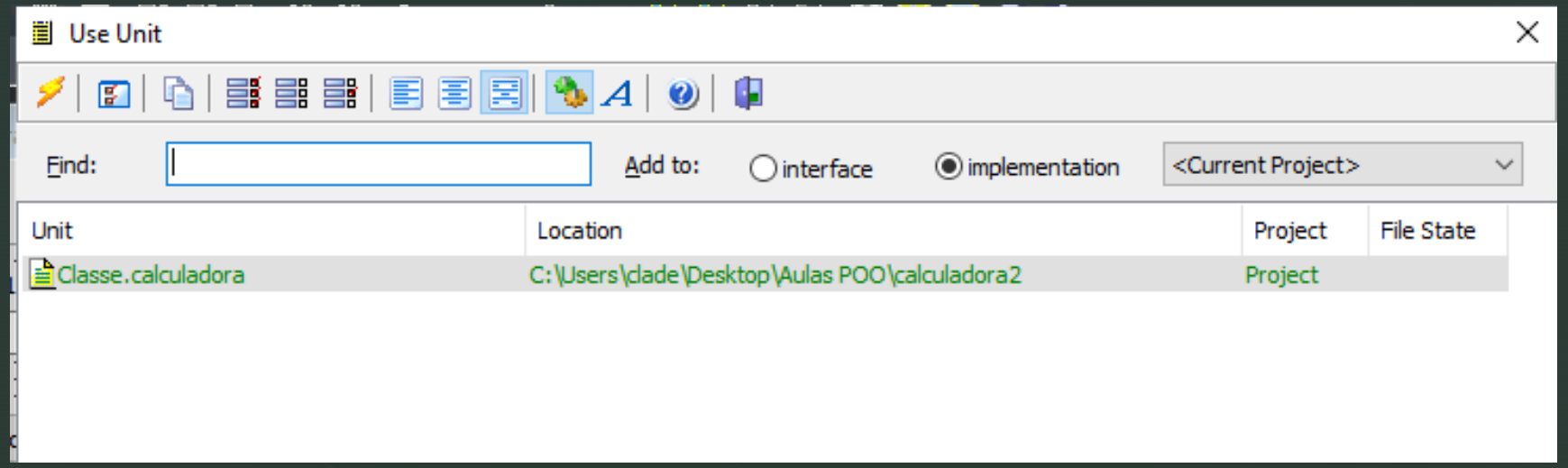
```
uses
System.SysUtils;
```



Agora vamos para nossa Camada de visão (view) Formulário

DevTeam

Acione o Alt + F11 ou File ->Use Unit



Na seção public criaremos uma property do tipo calculadora chamando a interface icalculadora.

```
private
  { Private declarations }
public

property calculadora: icalculadora;
  { Public declarations }
end;
```

Crtl+Shit+C

DevTeam

Declarou a property agora vamos...

```
private
  Fcalculadora: icalculadora;
  procedure Setcalculadora(const Value: icalculadora);
  { Private declarations }
public

property calculadora: icalculadora read Fcalculadora write Setcalculadora;
  { Public declarations }
end;
```

Crie um procedure na seção private

```
private
  Fcalculadora: icalculadora;
  procedure Setcalculadora(const Value: icalculadora);
  procedure Operacao; //esse nao vai dizer para que veio.
  //mais será a ação de todas as funcionalidades
  { Private declarations }
public
```

Ctrl + Shift + C

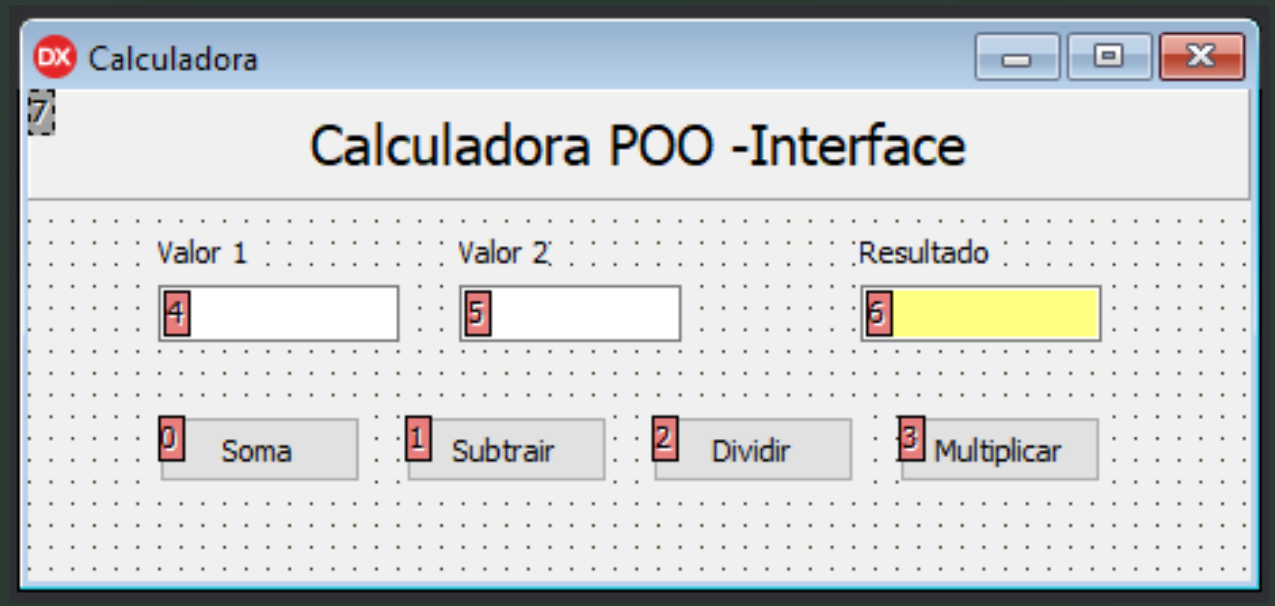
DevTeam

```
procedure TForm1.Operacao;
begin
   edt3.Text:=FloatToStr(calculadora.Operacao(StrToFloat(edt1.Text),StrToFloat(edt2.Text)));
end;
```

Veja que a procedure ficou Responsável pelo calculo

E os botões para que serve mesmo!!!!

Voltamos para nosso formulário



Agora cada botão ganhará Sua própria funcionalidade

DevTeam

Tsoma – lembra que você criou essa classe

```
procedure TForm1.btn1Click(Sender: TObject);
begin
    calculadora:=Tsoma.Create;
    Operacao;
end;
```

O operação é a procedure separada
Com sua própria responsabilidade
Única.

```
procedure TForm1.Operacao;
begin
   edt3.Text:=FloatToStr(calculadora.Operacao(StrToFloat(edt1.Text),StrToFloat(edt2.Text)));
end;
```

Agora é testar e tirar suas próprias conclusões.

```
procedure TForm1.btn2Click(Sender: TObject);
begin
    calculadora:=Tsubtrair.Create;
    Operacao;
end;

procedure TForm1.btn3Click(Sender: TObject);
begin
    calculadora:=Tdividir.Create;
    Operacao;
end;

procedure TForm1.btn4Click(Sender: TObject);
begin
    calculadora:=Tmultiplicar.Create;
    Operacao;
end;
```

DevTeam

Até a próxima.....

DevTeam