Como tipos definidos podem nos ajudar a mandar uma sonda para Marte

Caliel Lima da Costa

Desenvolvedor .NET com mais de 10 anos de experiência.

Trabalhou em vários projetos, alguns desde a sua concepção.

Atualmente presta seus serviços na WAES.

The Mars Climate Orbiter

US$ 125 milhões

338 kg

Lançada em dezembro 1998

Sinal perdido na órbita de Marte setembro 1999

O que eu acho que aconteceu....

The Mars Climate Orbiter

Falha humana e de software

Operadores achavam	Enquanto era
Sistema imperial	Sistema métrico
polegadas e libras	metro e Newton

Foi só um equívoco

Como tipos definidos podem nos ajudar a mandar uma sonda para Marte

```
public readonly struct CPF
{
    public CPF(ulong codigo, byte digito)...

    public ulong Codigo { get; }
    public byte Digito { get; }

    public bool EhValido { get; }

    public override string ToString()...

    public static CPF Parse(string text)...
}
```

```
public readonly struct CNPJ
{
    public CNPJ(ulong codigo, ushort filial, byte digito)...

    public ulong Codigo { get; }
    public ushort Filial { get; }
    public byte Digito { get; }

    public bool EhValido { get; }

    public override string ToString()...

    public static CNPJ Parse(string text)...
}
```

```
public readonly struct TituloEleitor
{
    public TituloEleitor(ulong inscricao, byte digito)...

    public ulong Inscricao { get; }
    public byte Digito { get; }

    public bool EhValido { get; }

    public override string ToString()...

    public static TituloEleitor Parse(string text)...
}
```

Não é nenhuma ideia original

Precisamos ir mais longe

Sistema para industria de móveis

- Tampo da mesa
  - centímetros (cm)
- Espessura da madeira
  - milímetros (mm)

Prefix	Symbol	Multiples
deca	da	10
hecto	h	10^{2}
kilo	k	10^{3}
mega	M	10^{6}
giga	G	10^{9}
tera	T	10^{12}
peta	P	10^{15}
exa	E	10^{18}
zetta	Z	10^{21}
yotta	Y	10^{24}

Prefix	Symbol	Multiples
deci	de	10^{-1}
centi	c	10^{-2}
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
peco	p	10^{-12}
femto	f	10^{-15}
atto	a	10^{-18}
zepto	z	10^{-21}
yocto	y	10^{-24}

Dados Digitais

- Conexão discada
  - Kilobits por segundo (kbps)
- Espaço em disco
  - Gigabytes ou Gibibyte?

Sistema de medição elétrica

- Corrente elétrica
  - Ampere (A)
- Tensão elétrica
  - Volt (V)
- Potência elétrica
  - Watt (W)

Dinheiro

- Moedas diferentes
- Você trocaria € 300 por R$ 1000?
- E 100 pesos argentinos por R$ 10?

Ângulos

- No mundo real
  - Graus (°)
- Computador
  - Radianos (rad)

Que tal eu
demonstrar....

```
class Program
{
    0 references
    static void Main(string[] args)
    {

    }
}
```

“ Toda vez que preciso converter de Graus para Radianos eu preciso parar pra pensar

30° 45° 60°
sin 1/2 √2/2 √3/2
cos √3/2 √2/2 1/2
tan √3/3 1 √3
2x
60°
x
30°
x√3
45°
s√2
s
45°

```
static void Main(string[] args)
{
}

0 references
public static double CalculeIMC(double peso, double altura)
{
    return peso / (altura * altura);
}
```

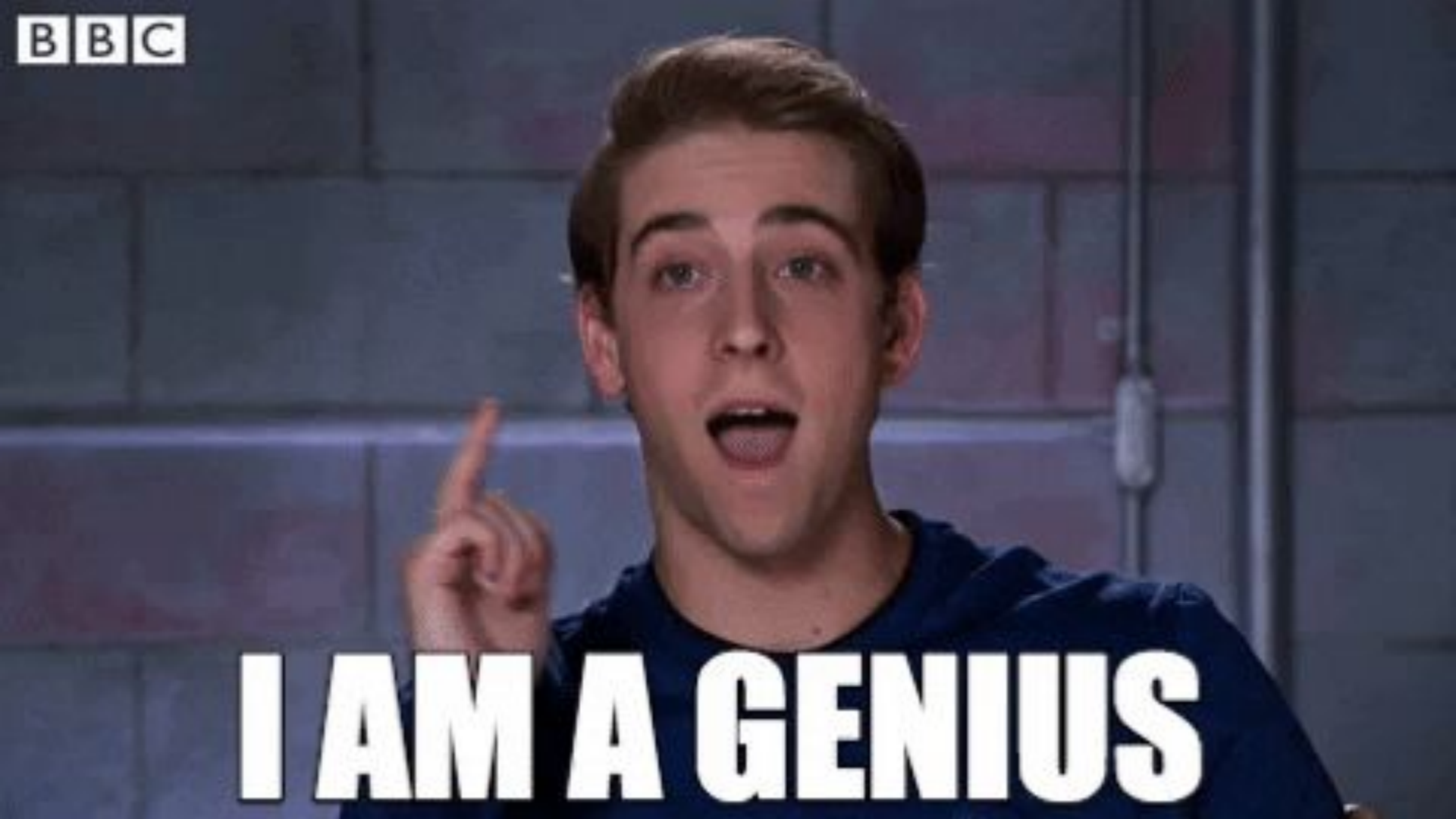
“Meu usuário sabe como fazer”

```
public static class SuperConversor
{
    0 references
    public static double RadianosParaGraus(double radianos)...
    0 references
    public static double GrausParaRadianos(double graus)...

    0 references
    public static double CentimetrosParaPolegadas(double centimetros)...
    0 references
    public static double PolegadasParaCentimetros(double polegadas)...

    0 references
    public static int IdadeEmAnos(DateTime dataDeNascimento)...
}
```

```
public class Servico
{
    private const double Milimetro = 1e-3;

    0 references
    public double EspessuraMinima => 2d * Milimetro;
    0 references
    public double EspessuraMaxima => 200d * Milimetro;
}
```

@LateNightSeth

“ E se eu criar uma classe para representar um ângulo em **graus** e outra classe para ângulo em **radianos**?

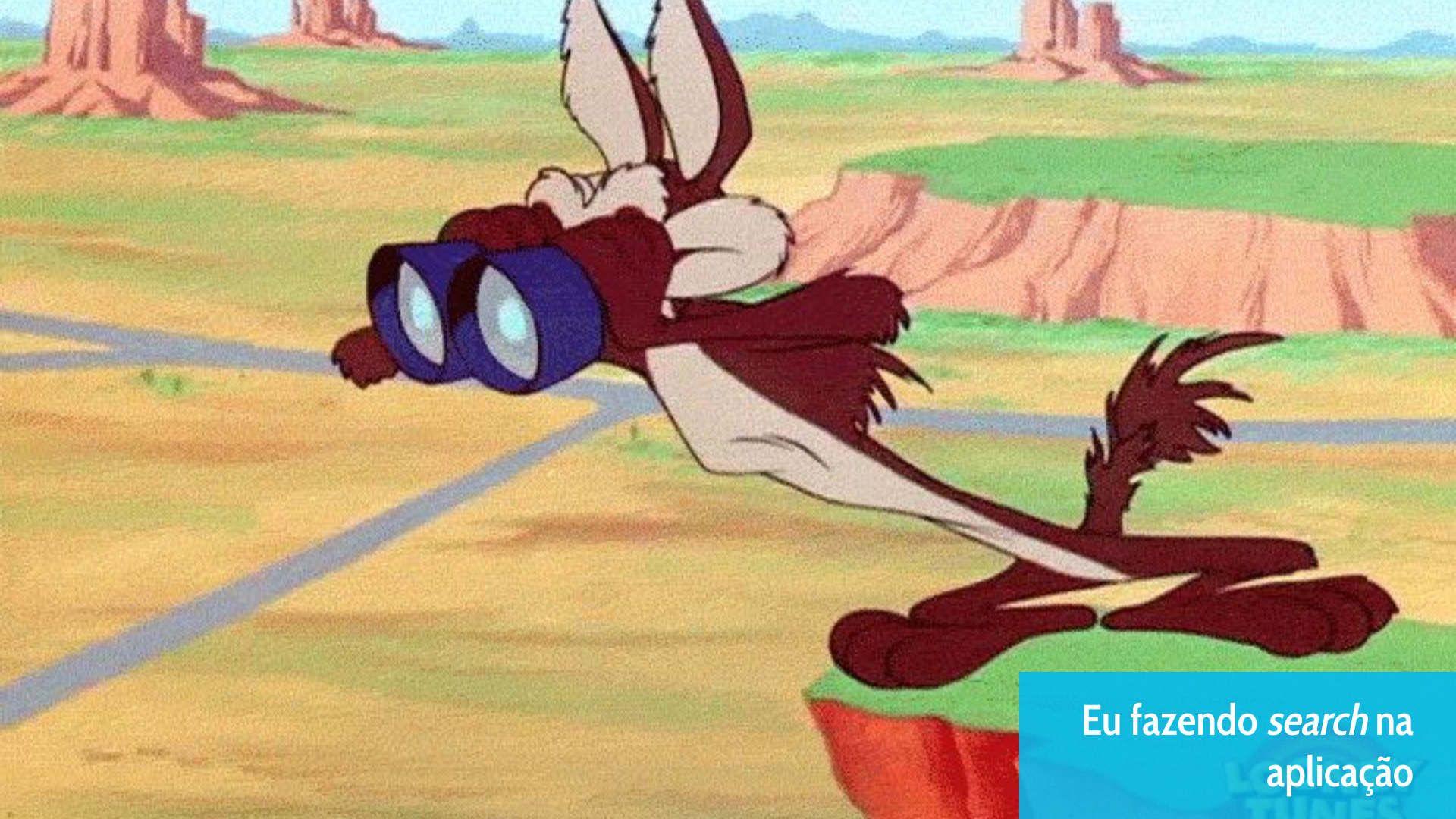

BBC
I AM A GENIUS

```
[DebuggerDisplay("{Value} rad")]
public readonly struct Radian
{
    public const string Symbol = "°";

    public Radian(double value)...

    public double Value { get; }

    public double Sin() => Math.Sin(Value);
    public double Cos() => Math.Cos(Value);

    public static explicit operator double(Radian self) =>
    public static explicit operator Radian(double self) =>
}
```

```
[DebuggerDisplay("{Value} °")]
public readonly struct Degree : IUnit, IUnitAngle
{
    private readonly double _radians;

    public const string Symbol = "°";

    public Degree(double value)...

    public double Value { get; }

    public double Sin() => Math.Sin(_radians);
    public double Cos() => Math.Cos(_radians);

    Radian IUnitAngle.AsRadian() => this;

    public static explicit operator double(Degree self) =>
    public static explicit operator Degree(double self) =>

    public static implicit operator Radian(Degree self) =>
    public static implicit operator Degree(Radian self) =>
}
```

Vamos testar isso

```
static void Main(string[] args)
{
    var grausUsandoConstrutor = new Degree(45);
    var grausUsandoCast = (Degree)275;

    var radianosUsandoCast = (Radian)Math.PI;

    var foo = Magica(grausUsandoCast);
}

public static double Magica(Radian angulo) => angulo.Value * 5;
```


Eu fazendo *search* na aplicação

```
[DebuggerDisplay("{Value} °")]
public partial struct Degree :
    IFormattable,
    IComparable,
    IEquatable<Degree>,
    IComparable<Degree>
{
```

```
public static bool operator ==(Degree self, Degree other) => self.
public static bool operator !=(Degree self, Degree other) => !self

public static bool operator <(Degree self, Degree other) => self.C
public static bool operator >(Degree self, Degree other) => self.C
public static bool operator <=(Degree self, Degree other) => self.
public static bool operator >=(Degree self, Degree other) => self.

public static Degree operator +(Degree self, Degree other) => new
public static Degree operator -(Degree self, Degree other) => new

public static Degree operator *(Degree self, double other) => new
public static Degree operator *(double self, Degree other) => new

public static Degree operator /(Degree self, double other) => new
```

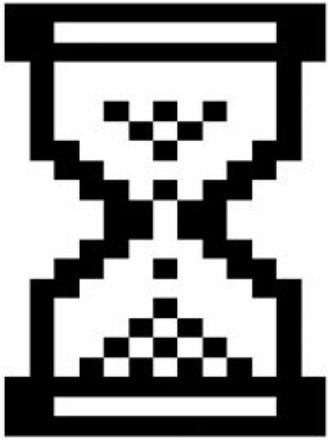
+

```
public interface IUnit
{
    double Value { get; }
}

internal interface IUnitAngle
{
    Radian AsRadian();
}
```

```
var angles = new IUnitAngle[]
{
    new Degree(45),
    new Radian(Math.PI),
    (Degree)270.55d,
};

var sum = (Radian)angles
    .Select(unit => unit.AsRadian())
    .Sum(x => x.Value);
```

Criei um Jaeger todo equipado

Depois de algum tempo



Em outro projeto

Sistema Internacional

- Metro (m)
- Centimetro (cm)

Sistema Imperial

- Pé (ft)
- Polegada (in)

```
/// <summary>
/// Represents a Centimetre (symbol cm).
/// </summary>
[Serializable]
[JsonConverter(typeof(UnitJsonConverter))]
[DebuggerDisplay("{Value} cm")]
public partial struct Centimetre :
    IUnit,
    IEquatable<Centimetre>,
    IComparable<Centimetre>,
    IXmlSerializable {
```

```
{
    var foo = new
    {
        MediaDistancia = (Metre)500,
        CurtaDistancia = (Centimetre)35,

        Polegadas = (Inch)5
    };
    var serialized = JsonConvert.SerializeObject(foo);
}
```

```
1  {
2      "MediaDistancia":500.0,
3      "CurtaDistancia":35.0,
4      "Polegadas":5.0
5  }
```

NEW
Não inventei a roda
SENORGIF.COM

Algumas vezes ficamos tendo trabalho a toa

“

Se você está escrevendo uma classe Util (Helper, Extensions), existe a chance de ter algo a melhorar com seu design

```
public static void ChameIMC()
{
    Console.WriteLine("Informe o Peso (EM QUILOS): ");
    var peso = Kilogram.Parse(Console.ReadLine());

    Console.WriteLine("Informe a Altura (EM CENTIMETROS, 100 Centimetros = 1 Metro):");
    var altura = Centimeter.Parse(Console.ReadLine());

    var imc = CalculeIMC(peso, altura);
    Console.WriteLine(imc);
}

/// <summary>
/// Calcula o IMC
/// </summary>
/// <param name="peso">informe o peso</param>
/// <param name="altura">informe a altura</param>
/// <returns></returns>
public static double CalculeIMC(Kilogram peso, Meter altura)
{
    return peso.Value / (altura.Value * altura.Value);
}
```

Aplicações

- Moedas
- TODAS as unidades de medidas
  - Peso
  - Energia
  - Volume
- Informação digital
- Tempo
  - Anos, meses, semanas

```
public class Funcionario
{
    public string Nome { get; }
    public BrazilianReal Salario { get; }
    public Years TempoDeCasa { get; }
    public Meter Altura { get; }
    public CPF CPF { get; }
    public TituloEleitor TituloEleitor { get; }
    public CodigoNumerico Setor { get; }
    public CodigoNumerico Departamento { get; }
    public CodigoNumerico Cracha { get; }
}
```

Ok
I'll give it a try

Como tipos definidos podem nos ajudar a mandar uma sonda para Marte

QUESTIONS?

Caliel Lima da Costa

https://www.linkedin.com/in/caliel/

https://github.com/calielc