D1 CC - UDF

C# e programação orientada a **eventos**

Grupo Rosa Escuro

André - 30003032
Ewerton B.Ramos - 30189098
João Marcos - 30620406
Miguel - 30129061
Pedro Oliveira - 29988080
Juan- 30289955

D1 CC - UDF

Sumário

D1 CC - UDF

Sumário

D1 CC - UDF

Conceitos

- **Paradigma**

Etimologia: Do grego antigo παράδειγμα (parádeigma), que significa "padrão" ou "exemplo"

Definição: “Um paradigma de linguagem de programação é uma abordagem à programação baseada em teoria matemática ou um conjunto coerente de princípios.” (Roy, 2009, p. 10)

História: uso excessivo de GOTO e código spaghetti

D1 CC - UDF

Conceitos

- **Evento**

"Um evento é algo ao qual um programa em execução (um processo) precisa responder, mas que ocorre fora do programa, em um momento imprevisível. Os eventos mais comuns são as entradas em um sistema de interface gráfica do usuário (GUI): cliques na tecla, movimentos do mouse, cliques de botão. Eles também podem ser operações de rede ou outras atividades de I/O assíncronas: a chegada de uma mensagem, a conclusão de uma operação de disco solicitada anteriormente." (Scott, 2008, p. 434)

D1 CC - UDF

Conceitos

- **Event Listener**

É uma função ou procedure que espera por um evento e chama algum Event Handler.

Sebesta, Capítulo 14.6.2

D1 CC - UDF

Conceitos

- **Event Handler**

“Um event handler é um bloco de código que é executado em resposta a um evento. Event handlers possibilitam que o programa seja responsivo a ações do usuário” (Sebesta, 2012, p. 655)

D1 CC - UDF

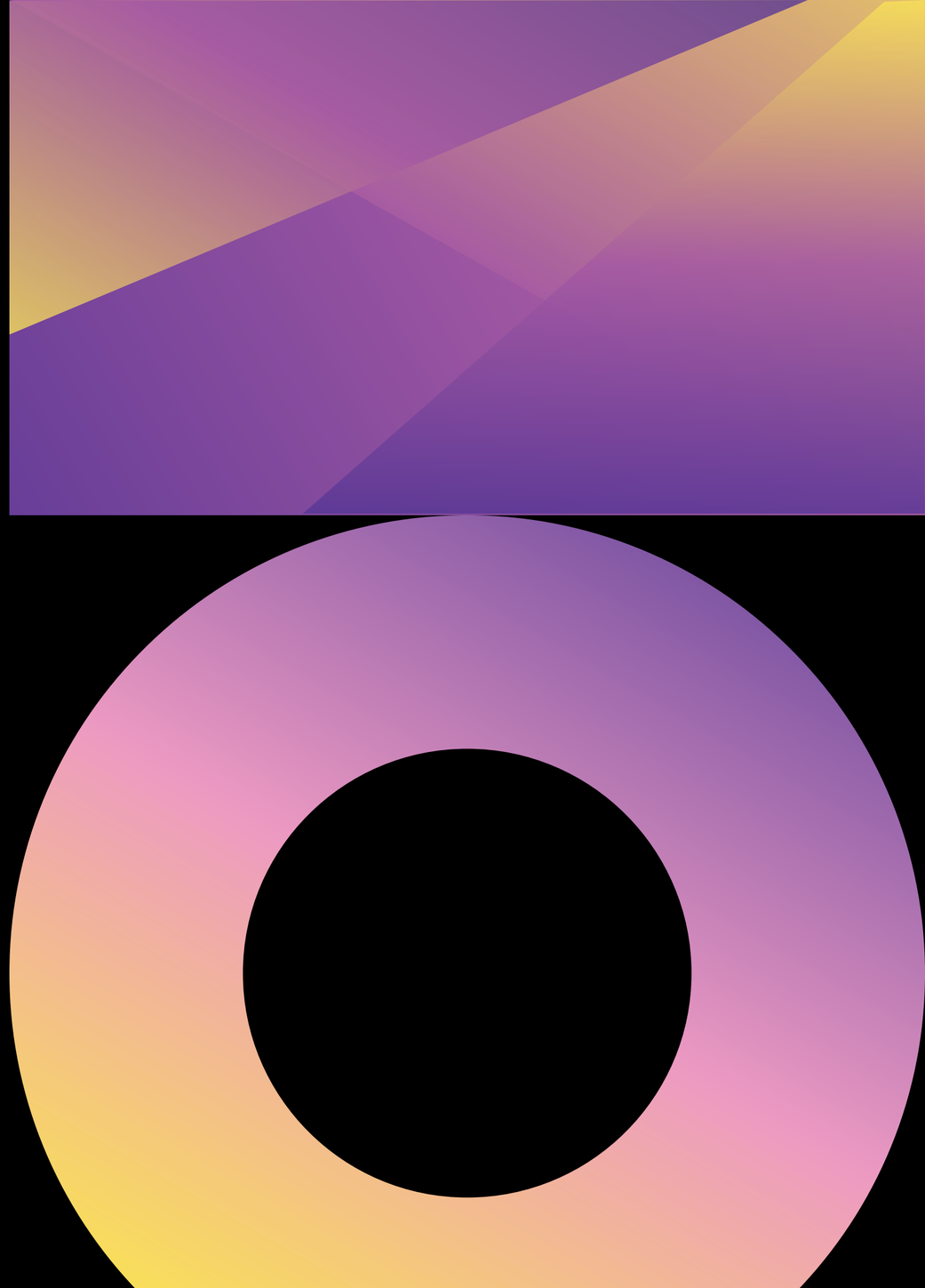
Vantagens - Paradigma

- Permite aplicações a responderem de forma efetiva aos inputs do usuário, resultando em uma experiência mais dinâmica e amigável.
- Possibilita o processamento assíncrono, otimizando a utilização de recursos e responsividade, crucial para aplicações em tempo real e interfaces de usuário.
- Proporciona a modularidade, simplificando a manutenção e escalabilidade separando responsabilidades e promovendo a reutilização do código.

D1 CC - UDF

Vantagens - Paradigma

- Pode ser aplicado em diversos domínios, incluindo aplicações web, interfaces gráficas, sistemas *server-side*, aplicações em tempo real, aplicações orientadas por dados, etc.
- A arquitetura assíncrona orientada a eventos permite que os aplicativos utilizem eficientemente os recursos do sistema, possibilitando a escala vertical e horizontal.



D1 CC - UDF

Desvantagens - Paradigma

- A natureza assíncrona de aplicações orientadas por eventos pode aumentar a complexidade do software. Garantir a sincronização correta, gerenciar as condições da corrida e abordar cenários de impasse pode exigir um esforço e precisão extensivos na codificação.
- Uma séries de eventos interconectados pode levar a efeitos em cascata, dificultando a previsão do resultado e o gerenciamento do estado do sistema.
- A escuta contínua por eventos pode levar a potenciais ineficiências na utilização de recursos e prejudicando a performance geral do sistema.

Casos de Uso

- Interfaces Gráficas
- Funções em Tempo Real
- Internet das Coisas
- Sistemas de bancos de dados
- Processamento de Eventos
- Desenvolvimento de Jogos

Evolução do Paradigma Orientado a Eventos

1950 - 1960
Whirlwind e SAGE

1980 - 1990
Smalltalk, Motif e Windows API

1990 - 2000
Web

1990 - HOJE
Javascript/AJAX

2009 - Atual
Node.js

Atual
Internet das Coisas e aplicações em tempo real

D1 CC - UDF

História

C# foi criada em 2000, integrada ao ecossistema .NET e feita como resposta da Microsoft à popularidade de Java e outras linguagens orientadas a objetos. A linguagem foi projetada para ser moderna, segura e simples, permitindo o desenvolvimento de aplicações robustas e escaláveis. Lançada junto com o .NET Framework, C# uniu conceitos de programação orientada a objetos com um forte gerenciamento de memória, oferecendo uma plataforma poderosa para o desenvolvimento de aplicações empresariais, web e jogos.

D1 CC - UDF

Anders Hejlsberg

Anders Hejlsberg foi o principal arquiteto da linguagem C# na Microsoft, liderando seu desenvolvimento a partir de 2000. Além de Hejlsberg, a criação do C# contou com contribuições importantes de Scott Wiltamuth e Peter Golde, que ajudaram no design e desenvolvimento da linguagem. Hejlsberg também é conhecido por seu trabalho anterior no Turbo Pascal, uma linguagem popular dos anos 80, e no Delphi, uma ferramenta para desenvolvimento rápido de software. Durante sua carreira na Microsoft, ele também desempenhou um papel fundamental no desenvolvimento do .NET Framework, que se tornou um dos principais pilares do ecossistema de desenvolvimento moderno.

No Mercado

- Desenvolvimento web com ASP.NET
- Aplicações para sistemas Windows NT
- Desenvolvimento de jogos com Unity e Unreal
- Cloud Computing com Azure
- Desenvolvimento mobile com Xamarin

Ano de pico

C# teve um pico de popularidade por volta de 2010-2015, especialmente com a expansão do .NET. Desde então, a linguagem se mantém estável entre as mais populares, especialmente em contextos empresariais e de desenvolvimento de jogos

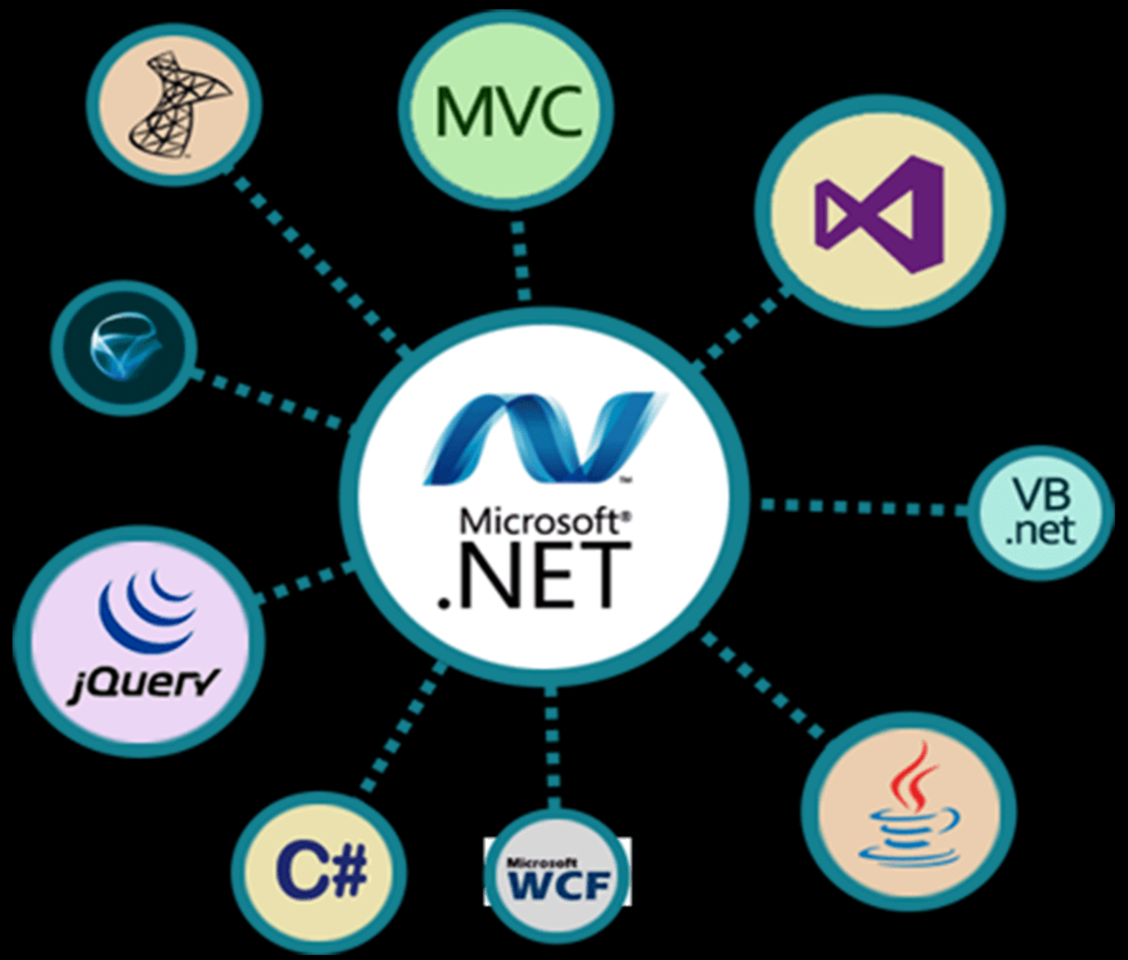
Logotipo

C# não possui um mascote oficial, mas seu logotipo é frequentemente associado ao hexágono do .NET, simbolizando robustez e versatilidade. O logotipo moderno é simples, refletindo a clareza e a eficiência da linguagem, alinhado com a identidade visual da Microsoft.

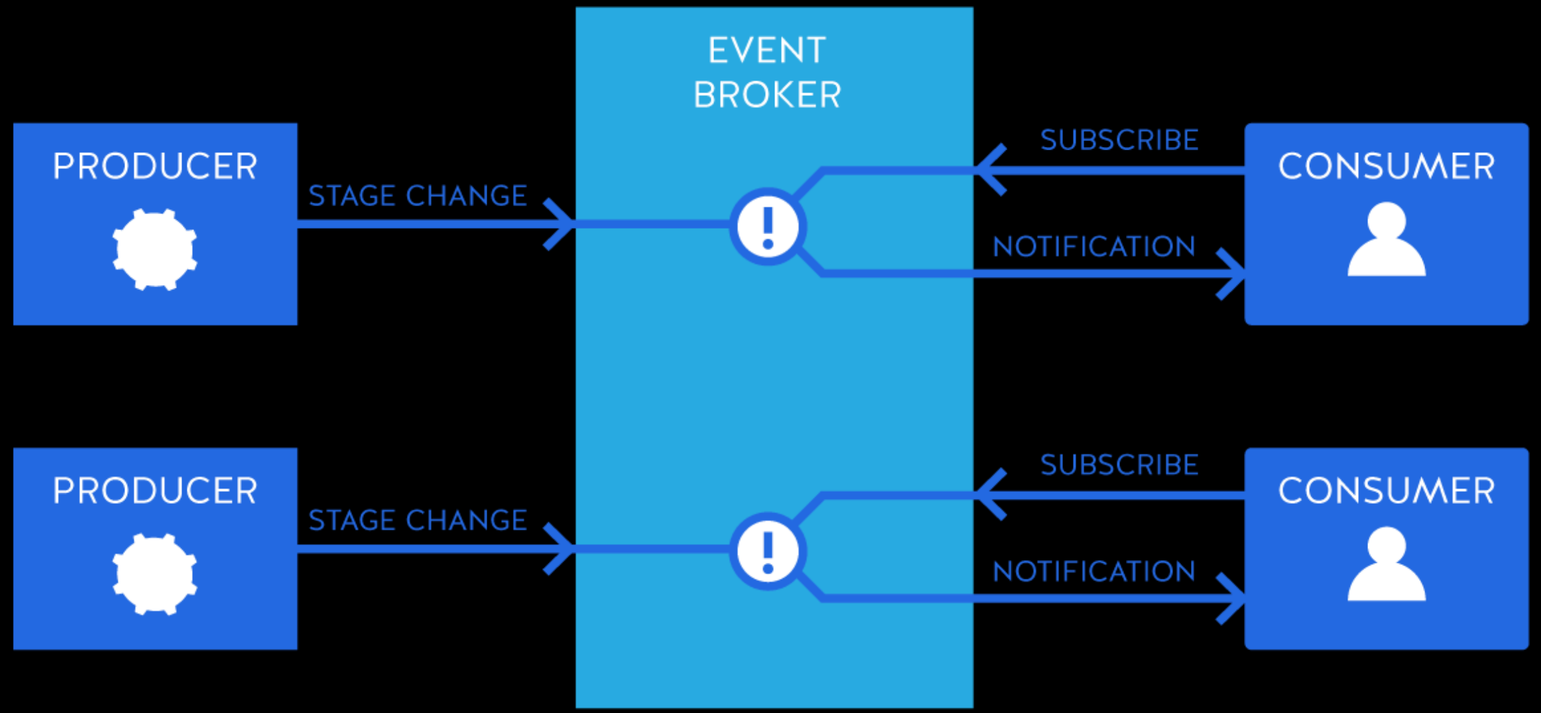
D1 CC - UDF

Ecossistema ASP.NET

- C#
- Visual Studio
- Visual Basic .NET
- MS SQL Server



O Paradigma


EVENT
BROKER
PRODUCER
STAGE CHANGE
SUBSCRIBE
NOTIFICATION
CONSUMER
PRODUCER
STAGE CHANGE
SUBSCRIBE
NOTIFICATION
CONSUMER

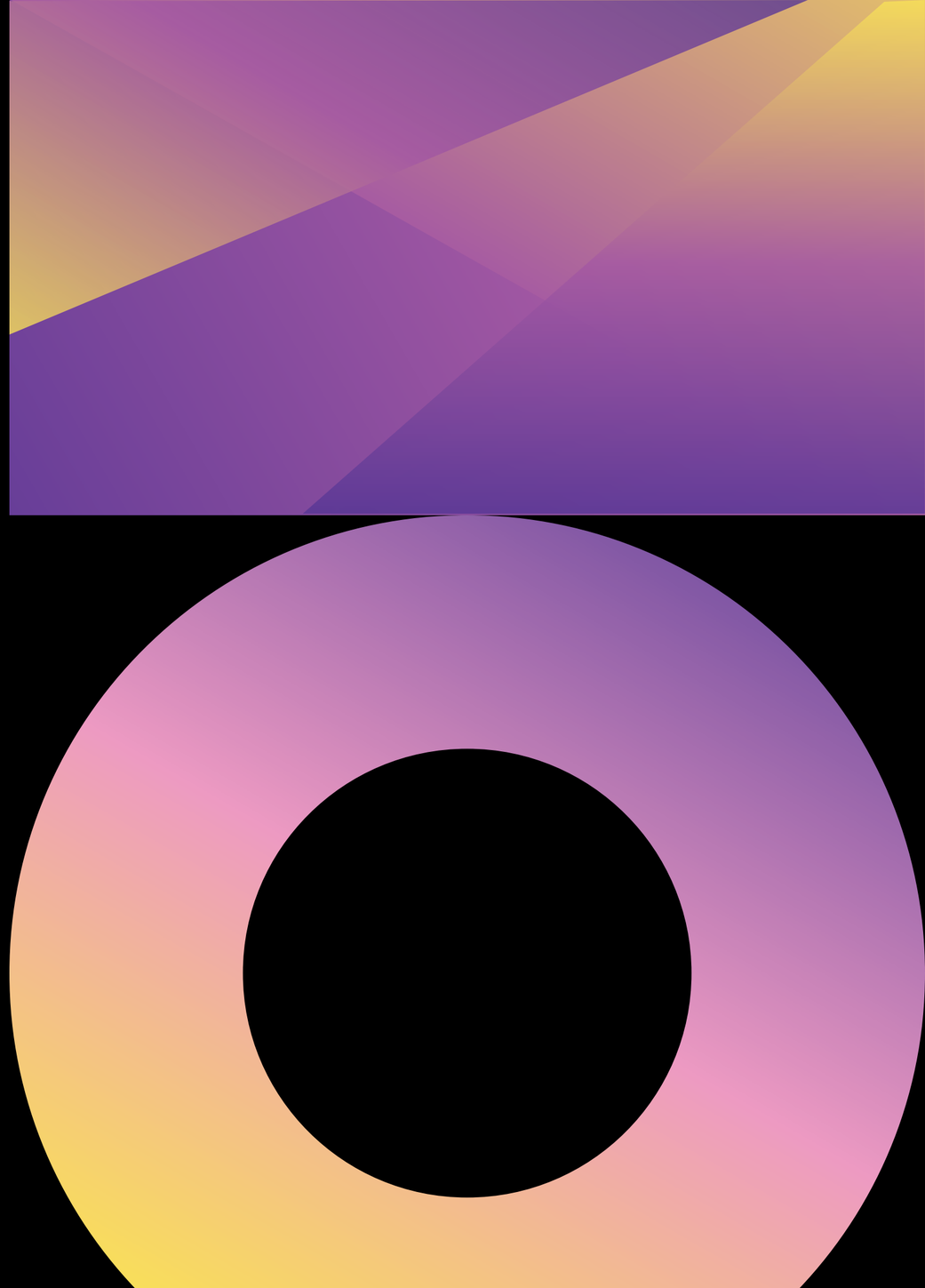
D1 CC - UDF

C# - Vantagens

- C# é profundamente integrada com o ecossistema .NET, oferecendo um vasto conjunto de bibliotecas e suporte para diferentes tipos de aplicações.

- Facilidade em reutilização de código através de conceitos de programação orientada a objetos.

- Possui um coletor de lixo embutido que ajuda a gerenciar a alocação e liberação de memória, simplificando o trabalho do desenvolvedor.



C# - Desvantagens

- Embora tenha se tornado mais multiplataforma, C# é fortemente ligada ao ecossistema Microsoft, o que pode ser um obstáculo em ambientes Linux.

- Pode ser mais complexa para novos desenvolvedores em comparação com linguagens como Python, devido à sua sintaxe e estrutura de tipos estáticos.

- Apesar de suportar o desenvolvimento móvel através do Xamarin, C# ainda enfrenta limitações em comparação com outras ferramentas.

Qual é a diferença?

D1 CC - UDF

O paradigma orientado a eventos é muito versátil e pode ser suportado por qualquer linguagem de programação.

Amplamente utilizado no desenvolvimento web na criação de *web pages* interativas através do modelo de eventos DOM.

Eventos de clique em botões, movimentação do mouse e carregamento de elementos.

Lua é comumente usada em ambientes que requerem gerenciamento de eventos, como em jogos ou sistemas embarcados.

Muitos motores de jogos usam Lua para lidar com eventos como movimentação de personagens, colisões, etc.

utiliza o conceito de goroutines, que permite a execução assíncrona de tarefas, sendo adequado para sistemas orientados a eventos.

Serviços em microarquitetura que respondem a eventos de diferentes sistemas, como chamadas a APIs ou respostas de bancos de dados.

Sintaxe C#

```
public class Program
{
    public static void Main(string[] args)
    {
        System.Console.WriteLine("Hello, World!");
    }
}
```

Namespace

```
namespace SampleNamespace
{
    class SampleClass
    {
        public void SampleMethod()
        {
            System.Console.WriteLine(
                "SampleMethod inside SampleNamespace");
        }
    }
}
```

Namespaces

Classes

```
public class Person
{
    public required string LastName { get; set; }
    public required string FirstName { get; set; }
}
```

```
Customer object1 = new Customer();
```

Structs

```
public struct Coords
{
    public Coords(double x, double y)
    {
        X = x;
        Y = y;
    }

    public double X { get; }
    public double Y { get; }

    public override string ToString() => $"({X}, {Y})";
}
```

Campos

Propriedades

Métodos

```
abstract class Motorcycle
{
    // Anyone can call this.
    public void StartEngine() {/* Method statements here */ }

    // Only derived classes can call this.
    protected void AddGas(int gallons) { /* Method statements here */ }

    // Derived classes can override the base class implementation.
    public virtual int Drive(int miles, int speed) { /* Method statements

    // Derived classes m
    public abstract doub
}
```

```
public class MinhaClasse
{
    private int meuCampo;
    public int MinhaPropriedade
    {
        get { return meuCampo; }
        set { meuCampo = value; }
    }
}
```

If/Else e Switch

```
void DisplayWeatherReport(double tempInCelsius)
{
    if (tempInCelsius < 20.0)
    {
        Console.WriteLine("Cold.");
    }
    else
    {
        Console.WriteLine("Perfect!");
    }
}
```

```
void DisplayMeasurement(double measurement)
{
    switch (measurement)
    {
        case < 0.0:
            Console.WriteLine($"Measured value is {measurement}; too low.");
            break;

        case > 15.0:
            Console.WriteLine($"Measured value is {measurement}; too high.");
            break;

        case double.NaN:
            Console.WriteLine("Failed measurement.");
            break;

        default:
            Console.WriteLine($"Measured value is {measurement}.");
            break;
    }
}
```

Fonte

For e Foreach

```
for (int i = 0; i < 3; i++)
{
    Console.Write(i);
}
```

```
Span<int> numbers = [3, 14, 15, 92, 6];
foreach (int number in numbers)
{
    Console.Write($"{number} ");
}
```

Fonte

While e Do while

```
int n = 0;
while (n < 5)
{
    Console.Write(n);
    n++;
}
```

```
int n = 0;
do
{
    Console.Write(n);
    n++;
} while (n < 5);
```

D1 CC - UDF

Try Catch Finally

```
public async Task HandleRequest(int itemId, CancellationToken ct)
{
    Busy = true;

    try
    {
        await ProcessAsync(itemId, ct);
    }
    finally
    {
        Busy = false;
    }
}
```

Throw

```
if (shapeAmount <= 0)
{
    throw new ArgumentOutOfRangeException(nameof(shapeAmount),
}
```

Fonte

LINQ

```
// Specify the data source.
int[] scores = [97, 92, 81, 60];

// Define the query expression.
IEnumerable<int> scoreQuery =
    from score in scores
    where score > 80
    select score;

// Execute the query.
foreach (var i in scoreQuery)
{
    Console.Write(i + " ");
}

// Output: 97 92 81
```

Expressões LAMBDA

```
Func<int, int> square = x => x * x;
Console.WriteLine(square(5));
// Output:
// 25
```

D1 CC - UDF

Delegados

```
delegate void Calculator(int x, int y);

class Program
{
    public static void Add(int a, int b)
    {
        Console.WriteLine(a + b);
    }
    public static void Mul(int a, int b)
    {
        Console.WriteLine(a * b);
    }
    static void Main(string[] args)
    {
        Calculator calc = new Calculator(Add);



        calc(20, 30);
    }

}
```

Eventos

```
public class TestingEvents : MonoBehaviour {

    public event EventHandler OnSpacePressed;

    private void Start() {
        OnSpacePressed += Testing_OnSpacePressed;
    }

    private void Testing_OnSpacePressed(object sender, EventArgs e) {
        Debug.Log("Space!");
    }

    private void Update() {
        if (Input.GetKeyDown(KeyCode.Space)) {
            // Space pressed!
            OnSpacePressed?.Invoke(this, EventArgs.Empty);
        }
    }
}
```

```
publisher.RaiseCustomEvent += HandleCustomEvent;
```

Fonte Eventos Delegados

Semântica C#

A sintaxe está correta,
mas a lógica está errada

```
using System;

namespace ErroSemantico
{
    class Program
    {
        static void Main(string[] args)
        {
            int numero = "dez"; // Atribuindo uma string a um inteiro
            Console.WriteLine(numero);
        }
    }
}
```

Classes

```csharp
public class Pessoa
{
    public required string Nome { get; set; }
    public int Idade { get; set; }
    public double Salario { get; set; }

    public void CalcularImposto()
    {
        // Erro semântico: Tentando calcular imposto com base na idade, que não é relevante
        double imposto = Idade * 0.1;
        Console.WriteLine($"Imposto a pagar: R${imposto}");
    }
}
```

Structs

```csharp
struct Ponto
{
    public int X { get; set; }
    public int Y { get; set; }

    public void Mover(int deltaX, int deltaY)
    {
        // Erro semântico: Modificando diretamente os campos de uma struct, o que viola a imutabilidade implícita
        X += deltaX;
        Y += deltaY;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Ponto p = new Ponto { X = 1, Y = 2 };
        p.Mover(3, 4);
        Console.WriteLine($"Novo ponto: ({p.X}, {p.Y})");
    }
}
```

If - Else

```
using System;

class Program
{
    static void Main()
    {
        int numero = 10;

        if (numero > 5)
        {
            Console.WriteLine("O número é maior que 5");
        }
        else if (numero < 5)
        {
            Console.WriteLine("O número é menor que 5");
        }
        else
        {
            Console.WriteLine("O número é maior que 5 e menor que 5"); // Erro de semântica: lógica impossível
        }
    }
}
```

For

```
using System;

class Program
{
    static void Main()
    {
        int[] numeros = {1, 2, 3, 4, 5};

        for (int i = 0; i <= numeros.Length; i++)
        {
            Console.WriteLine(numeros[i]); // Erro de semântica: o índice i pode ultrapassar o limite do array
        }

        Console.WriteLine("Loop encerrado");
    }
}
```

While

```
using System;

class Program
{
    static void Main()
    {
        int contador = 5;

        while (contador > 0)
        {
            Console.WriteLine(contador);
            contador++; // Erro de semântica: contador está sendo incrementado em vez de decrementado
        }

        Console.WriteLine("Loop encerrado");
    }
}
```

Evento

```
using System;
using System.Windows.Forms;

public class MeuFormulario : Form
{
    private Button meuBotao;

    public MeuFormulario()
    {
        meuBotao = new Button();
        meuBotao.Text = "Clique em mim";
        meuBotao.Click += new EventHandler(Botao_Click);
        Controls.Add(meuBotao);
    }

    private void Botao_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Botão clicado!");
    }

    private void AtualizarTextoDoBotao()
    {
        meuBotao.Text = "Atualizado!";
    }

    public static void Main()
    {
        MeuFormulario form = new MeuFormulario();
        form.AtualizarTextoDoBotao();//Erro de semântica: o botao está sendo atualizado antes de mostrar o formulário
        Application.Run(form);
    }
}
```

Referências

- BestProg: ["The history of the emergence of the C# programming language and Microsoft .NET technology"](https://www.bestprog.net/en/2022/05/22/c-the-history-of-the-emergence-of-the-c-programming-language-and-microsoft-net-technology/).
- DEV Community: ["C#'s History, .NET Relationship, and Advantages/Disadvantages"](https://dev.to/snelson723/cs-history-net-relationship-and-advantagesdisadvantages-102b).
- AltexSoft: ["C# Pros and Cons"](https://www.altexsoft.com/blog/c-sharp-pros-and-cons/).
- DEV Community: ["The History and Importance of C# in the Software Industry"](https://dev.to/dogaaydinn/the-history-and-importance-of-c-in-the-software-industry-if2).
- DotNetCurry: ["The Evolution of C#"](https://www.dotnetcurry.com/csharp/1465/csharp-evolution).
- CodeGuru: ["C# Advantages"](https://www.codeguru.com/csharp/c-sharp-advantages/).
- DevMedia: ["O que é C#?"](https://www.devmedia.com.br/introducao-ao-c/1696).
- Microsoft Docs: ["Visão geral do C#"](https://learn.microsoft.com/pt-br/dotnet/csharp/)

Referências

- GeeksforGeeks:[”Whats is the Event Driven Programming Paradigm”] (https://www.geeksforgeeks.org/what-is-the-event-driven-programming-paradigm/)
- StudySmarter:[”Event Driven Programming”] (https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/event-driven-programming/)
- https://aosabook.org/en/v2/twisted.html
- Concepts of Programming Languages - Robert W. Sebesta 2012
- Programming Language Pragmatics - Michael L. Scott 2008
- https://core.ac.uk/download/pdf/150140371.pdf
- https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/compiler-messages/feature-version-errors
- https://learn.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/
- https://learn.microsoft.com/en-us/training/modules/csharp-if-elseif-else/

Referências

- https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/statements/selection-statements
- https://learn.microsoft.com/pt-br/dotnet/csharp/programming-guide/events/
- https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-8.0
- https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements#the-for-statement
- https://en.wikipedia.org/wiki/Whirlwind_I
- https://en.wikipedia.org/wiki/Semi-Automatic_Ground_Environment
-

D1 CC - UDF

E agora para a **Demonstração!**