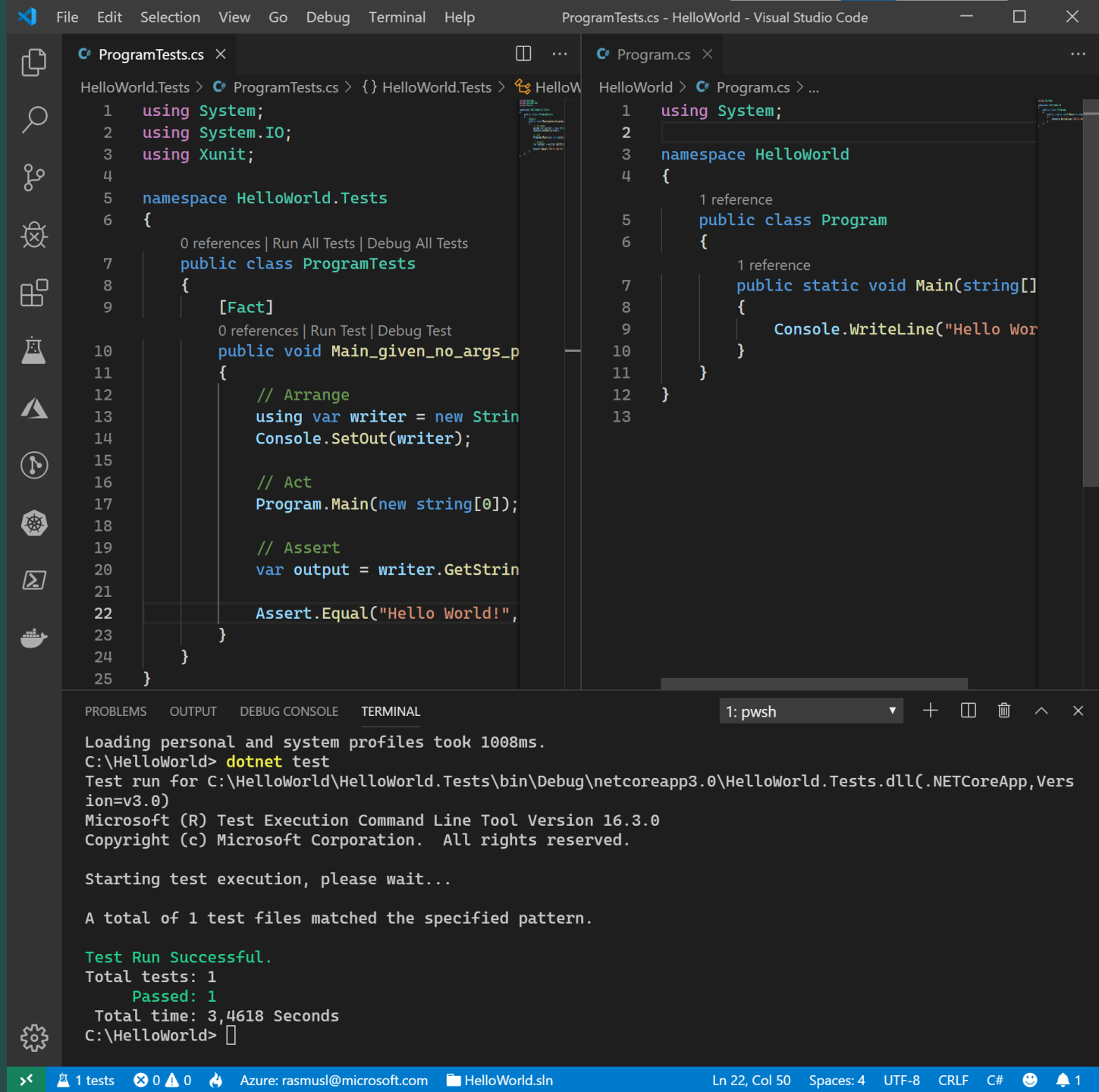
Test-Driven C♯

Rasmus Lystrøm
Associate Professor
ITU
rnie@itu.dk



2013-: Senior Consultant @ Microsoft
DevOps, Cloud, Security

2014-: Associate Professor @ ITU
Object-Oriented Programming, C♯, F♯, .NET Core

M.Sc. IT, ITU (2012)
Thesis: Forecalc – Developing a core spreadsheet implementation in F♯

1996-2008, 2013-: Captain @ Danish Army (Reserve)
Acting Battalion Chief of Staff,
Battalion Chief Operations Officer

Wife: Katrine
Children: Lærke (2), Laura (5), and Alma (12)

Origin: Aarhus
Current whereabouts: Vanløse, Copenhagen

Hobbies

SLAYER

MORBID ANGEL

OBITUARY

BRUTAL ASSAULT

CANNIBAL CORPSE

COPENHELL

W:O:A

HELLFEST

Agenda

Why C♯

Curriculum

Test-Driven Development

.NET (Core)

C♯

Visual Studio Code

Visual Studio 2019

Why C# - Popularity

C#: 31.4%

ASP.NET: 21.9%

ASP.NET Core: 19.1%

.NET: 35.1%

.NET Core: 24.5%

Microsoft SQL Server: 33.0%

Bash/Shell/PowerShell: 33.1%

GitHub: 82.8% (top 1)

Visual Studio Code: 50.7% (top 1) (2019)

Visual Studio: 31.5% (top 2) (2019)

Source: Stack Overflow Annual Developer Survey 2020 – 65,000 respondents
https://insights.stackoverflow.com/survey/2020

Why C♯ - Love

C♯: 59.7%

ASP.NET: 70.7% (top 1)

ASP.NET: 36.9%

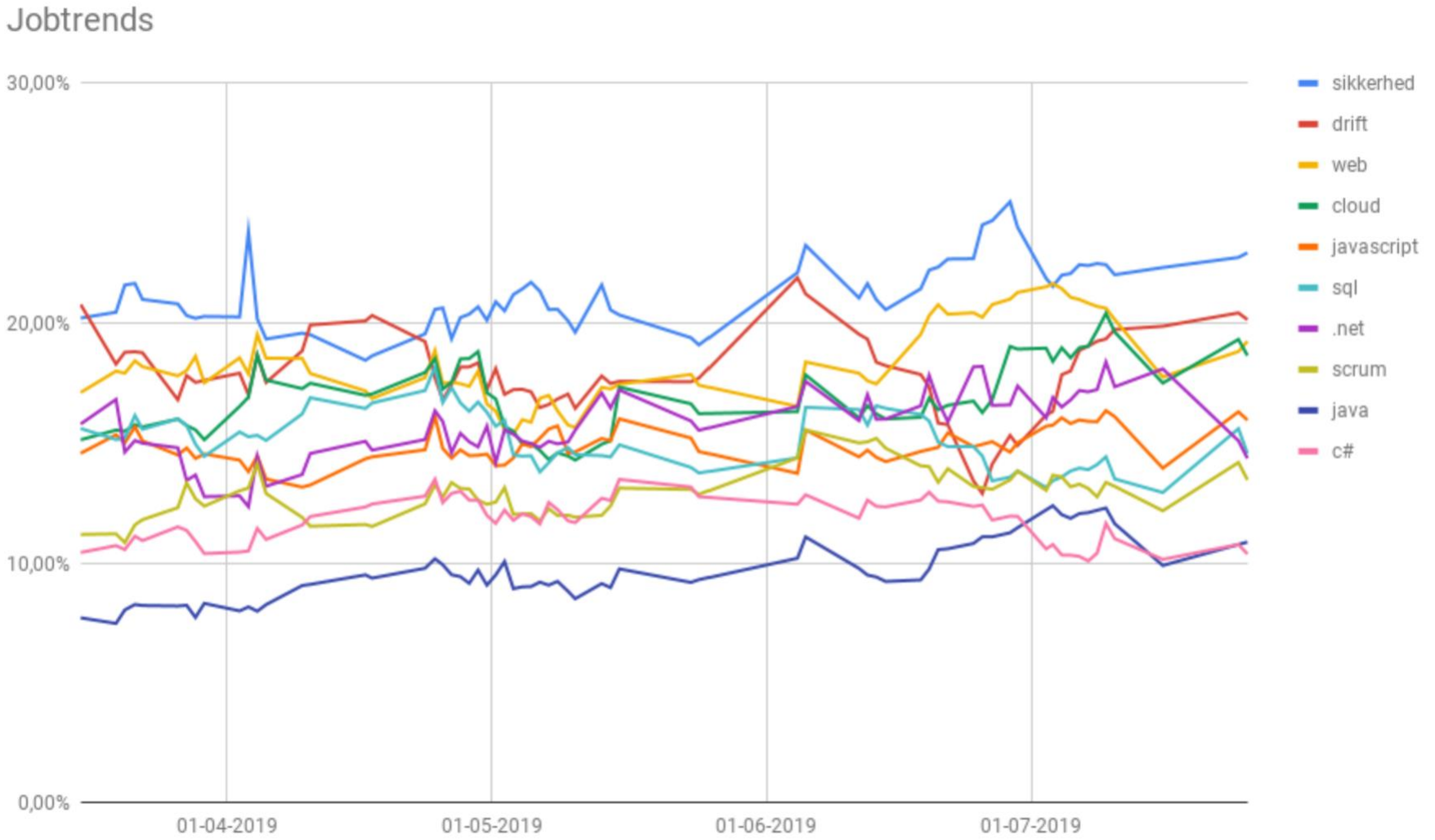
.NET: 47.5%

.NET Core: 71.5% (top 1)

Microsoft SQL Server: 50.9%

Source: Stack Overflow Annual Developer Survey 2020 – 65,000 respondents
https://insights.stackoverflow.com/survey/2020

Job trends



Hele top-ti ser sådan ud:

	Jobtrend 25-07-2019	Score
1	sikkerhed	22,9%
2	drift	20,1%
3	web	19,2%
4	cloud	18,6%
5	javascript	16,0%
6	sql	14,6%
7	.net	14,4%
8	scrum	13,5%
9	java	10,9%
10	c#	10,4%

Source: https://www.version2.dk/artikel/jobtrends-java-slaar-c-paa-maalfoto-1088564

Udvikler vild med C#: Ingen grund til at kode i Java nogensinde igen

(Illustration: Bigstock/Photosvit)

Seniorudvikler i konsulentfirma er krystalklar i mælet: C# og .Net er bare mindre bøvlet end Java. Og det er fordi, der kun er én leverandør bag platformen.

Tania Andersen @AndersenTania Fredag, 16. august 2019 - 5:11 17

FØLG

»Jeg synes, det er svært at finde ulemper ved C#. Hovedsageligt arbejder jeg med C#, men det sidste års tid har jeg arbejdet med Java. Det er besværligt. Der er ikke noget, der virker. Man skal trykke Java rigtigt på maven, før det gør det rigtige. I C# og .Net virker tingene bare. Der er også udfordringer, men det er så meget nemmere.«

Source: https://www.version2.dk/artikel/udvikler-vild-med-c-ingen-grund-at-kode-java-nogensinde-igen-1088651

Tentative Curriculum

Test-Driven C♯
Generics
Lambdas and Linq
Data access (SQL + Entity Framework)
Asynchronous and parallel processing
ASP.NET Core Web API
Design Patterns in Practice
Web apps with Blazor
Mobile apps with UWP and Xamarin.Forms
Security
Cloud

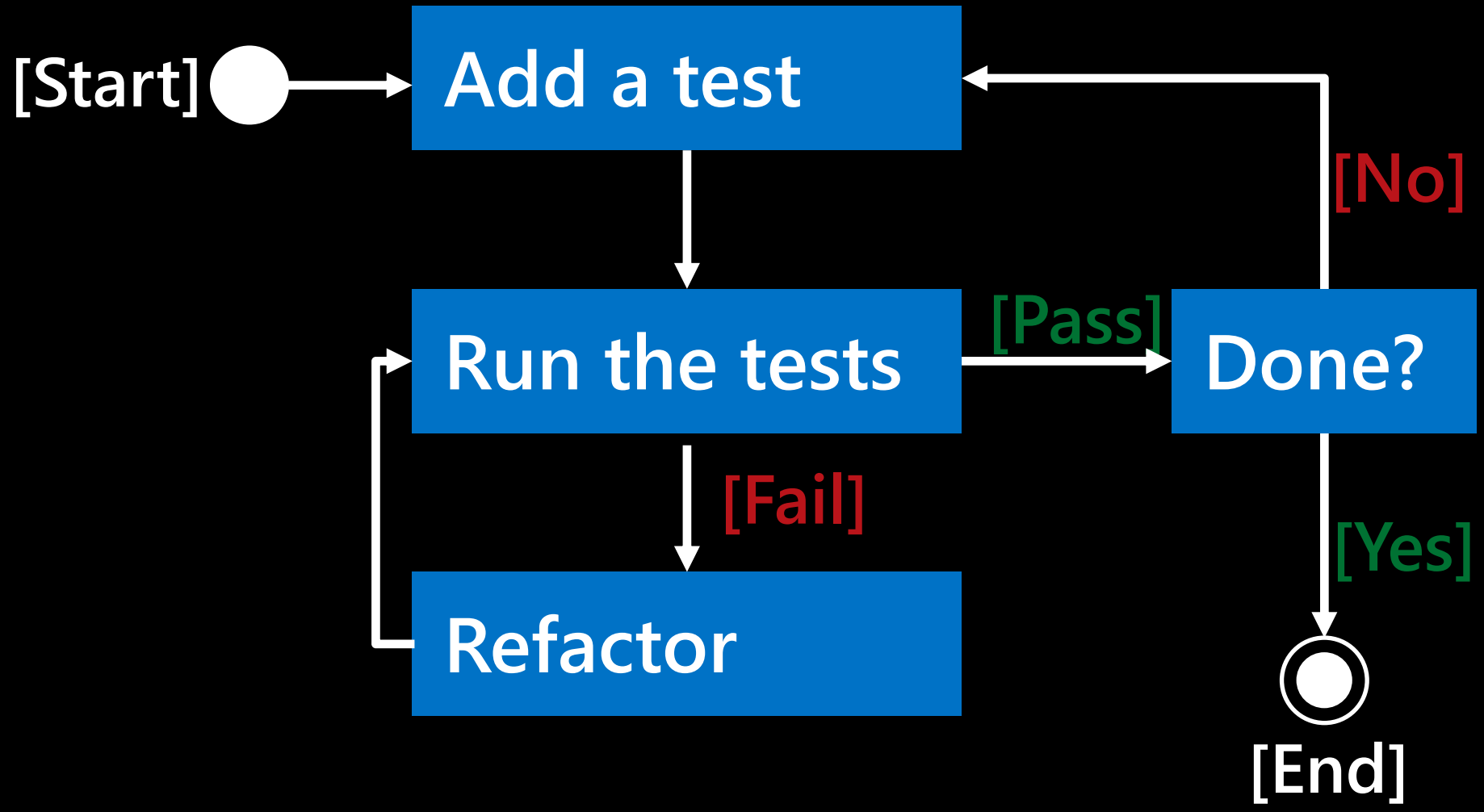
Test-Driven Development

WHAT?

WHY?

HOW?

Red – Green - Refactor

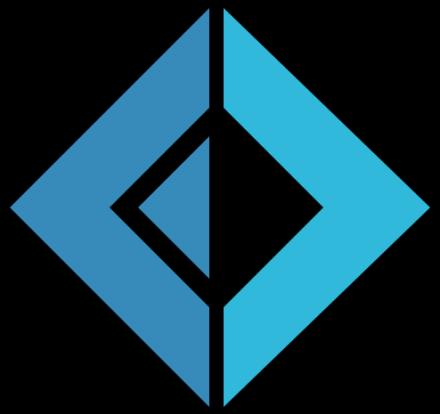


A brief introduction

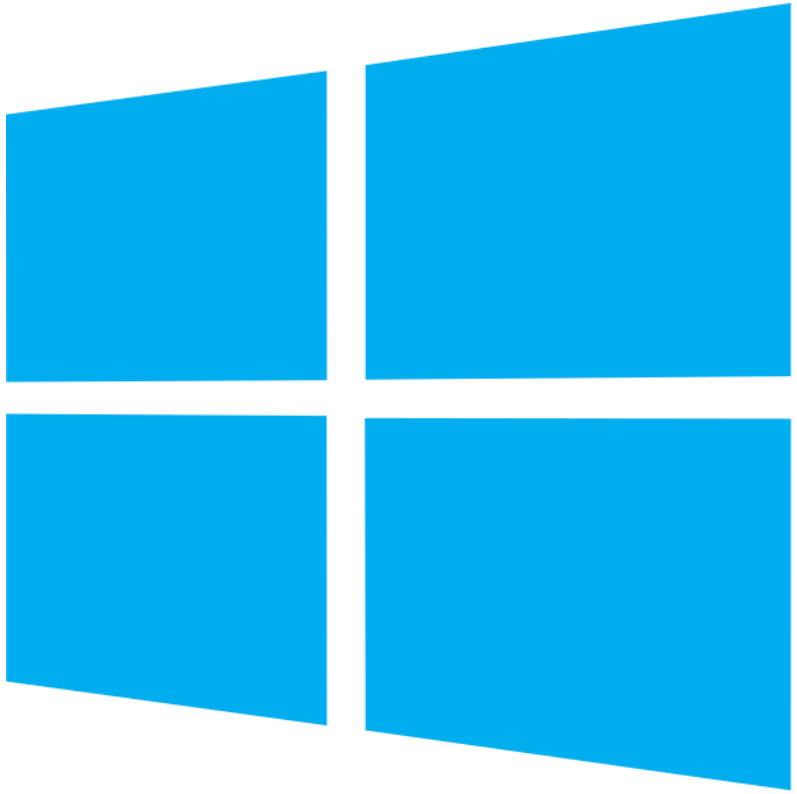
.NET

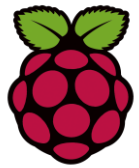
.NET is a free, cross-platform, open source developer platform for building many different types of applications.

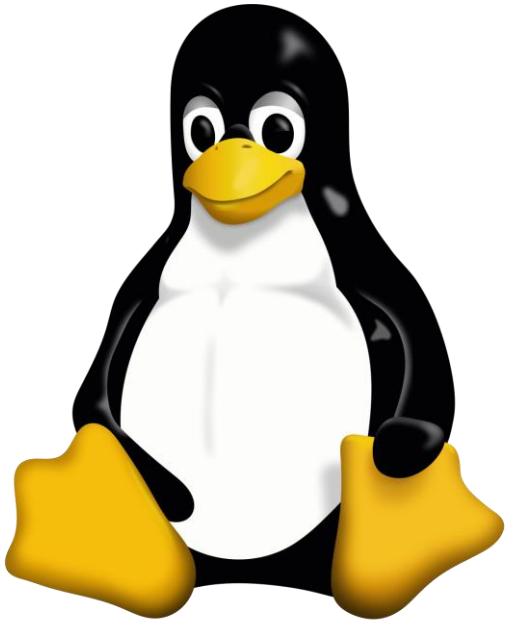
.NET Languages



.NET Platforms







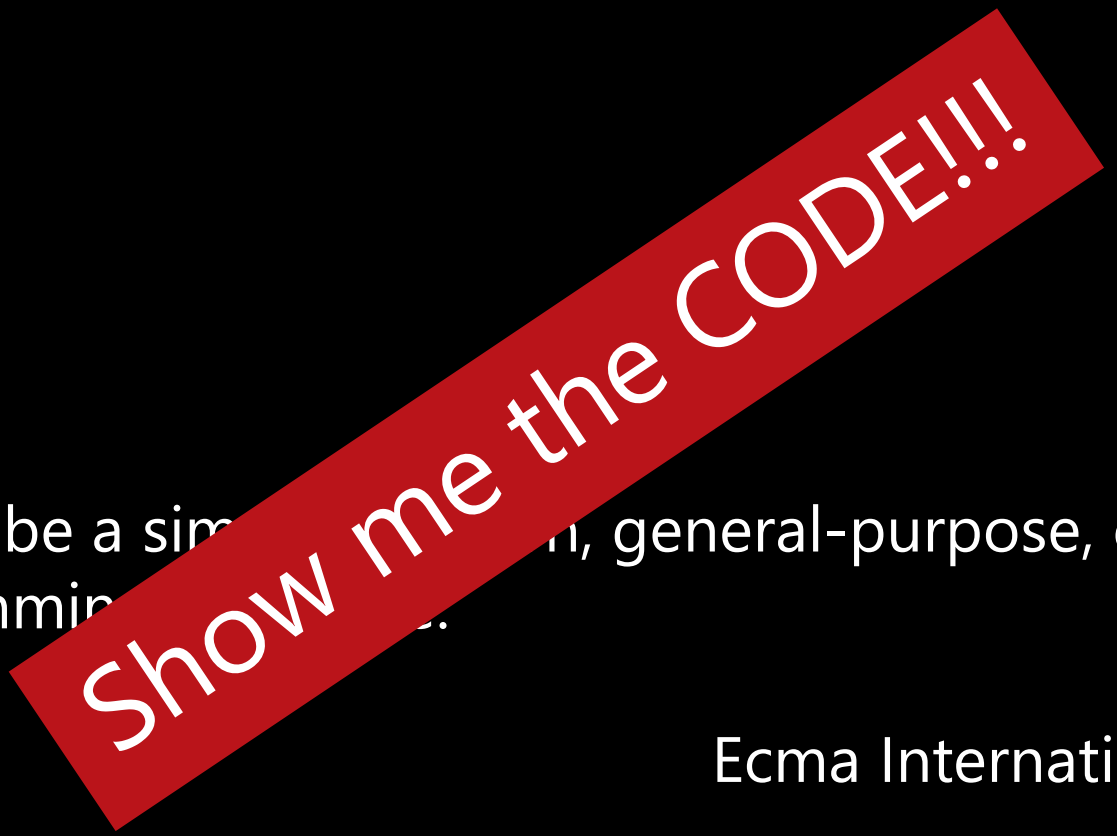
History of .NET and C#

	.NET	.NET Core	C#	F#	Visual Studio	Visual Studio Code
2002	1.0		1.0		.NET	
2005	2.0		2.0	1.0	2005	
2007			3.0			
2008	3.5				2008	
2010	4.0		4.0	2.0	2010	
2012	4.5			3.0	2012	
2013	4.5.1		5.0	3.1	2013	
2015	4.6		6.0	4.0	2015	
2016		1.0				1.0 – 1.8
2017	4.7	2.0	7.0		2017	1.9 – 1.19
2019	4.8	3.0	8.0	4.7	2019	1.20 – 130
2020	5.0		9.0			1.31 – 1.48…

C♯

C♯ is intended to be a sim[illegible]n, general-purpose, object-oriented programmin[illegible].

Ecma International (2006)



Coding Kata

String Calculator Kata

Create a method with the following signature:

```
int Add(string numbers)
```

The method can take up to two numbers, separated by commas, and will return their sum.

for example “” or “1” or “1,2” as inputs.

(for an empty string it will return 0)

…

Source: https://osherove.com/tdd-kata-1

C# basics

Create a C# console app with a test library

```
mkdir MyApp
cd MyApp

dotnet new console -o MyApp
dotnet new xunit -o MyApp.Tests

dotnet new sln
dotnet sln add MyApp
dotnet sln add MyApp.Tests
dotnet add MyApp.Tests reference MyApp
```

Build, test, and run your app

```
dotnet build
dotnet test
dotnet run --project MyApp
```

Naming conventions

Composed names

`currentLayout`, `CurrentLayout`

Variables and fields

`vehicle`, `leftElement`

Private fields

`_vehicle`, `_leftElement`

Methods

`CurrentVehicle()`, `Size()`

Properties

`Pi`, `Name`, `Size`

Classes

`MyClass`, `List<T>`

Interfaces

`IException`, `IObserver`

https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines

The C# class

```
using System;

namespace Namespace
{
    public class Class
    {
        private string _field;

        protected DateTime _inheritableField;

        public string Property { get => _field; } // Getter

        public int AutoProperty { get; set; }

        public Class() // Constructor
        {
        }
```

The C# class (methods)

```
public object InstanceMethod(int parameter)
{
    return null;
}

public virtual object OverridableInstanceMethod(bool parameter)
{
    return null;
}

public static void StaticMethod()
{
}

private void PrivateInstanceMethod()
{
}
```

The C# class (events and delegates)

```
        public event EventHandler Event;

        protected virtual void OnEvent(EventArgs e)
        {
            EventHandler handler = Event;
            handler?.Invoke(this, e);
        }

    }

    public delegate void MyEventHandler(object sender, EventArgs e);
}
```

Built-in types

```
bool boolean; // true || false
char character;  // 'a', 'b', 'c', '1', '2', '3'

// Floating point numeric types
decimal precisionFloatingPoint;
double floatingPoint64bit;
float floatingPoint32Bit;

// Integral numeric types
byte integer8bit;
int integer32bit;
long integer64bit;
short integer16bit;

sbyte signedByte;
uint unsignedInteger32bit;
ulong unsignedInteger64bit;
ushort unsignedInteger16bit;

// Reference types
object _object;
string _string;
dynamic dynamic;
```

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types

Basic Unit Test

```
public class Ticker
{
    public int Counter { get; private set; }
    public void Increment() => Counter++;
}

public class TickerTests
{
    [Fact]
    public void Increment_when_called_increases_Counter_by_1()
    {
        // Arrange
        var sut = new Ticker();

        // Act
        sut.Increment();

        // Assert
        Assert.Equal(1, sut.Counter);
    }
}
```