

Haladó fejlesztési technikák

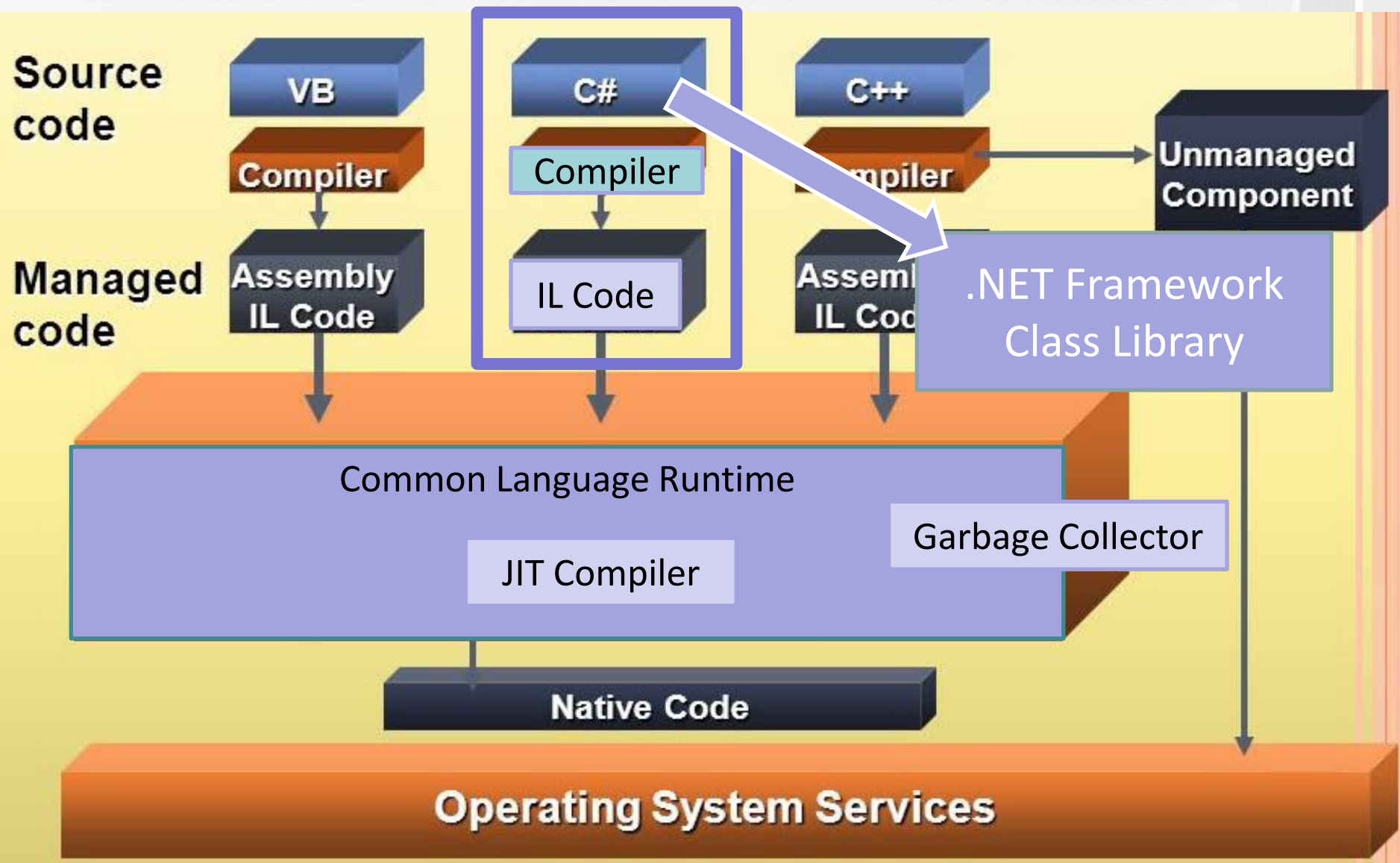
.NET Framework: CLR, MSIL, JIT, AOT

.NET memóriamenedzsment: GC

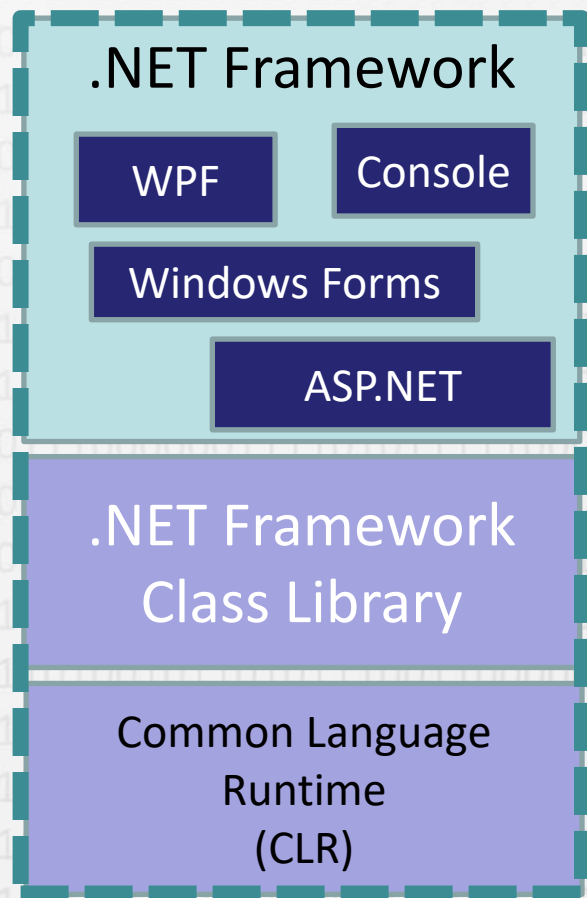
.NET változatok: .NET Framework, .NET Core, Mono, .NET Standard

.NET vNext

.NET Architecture

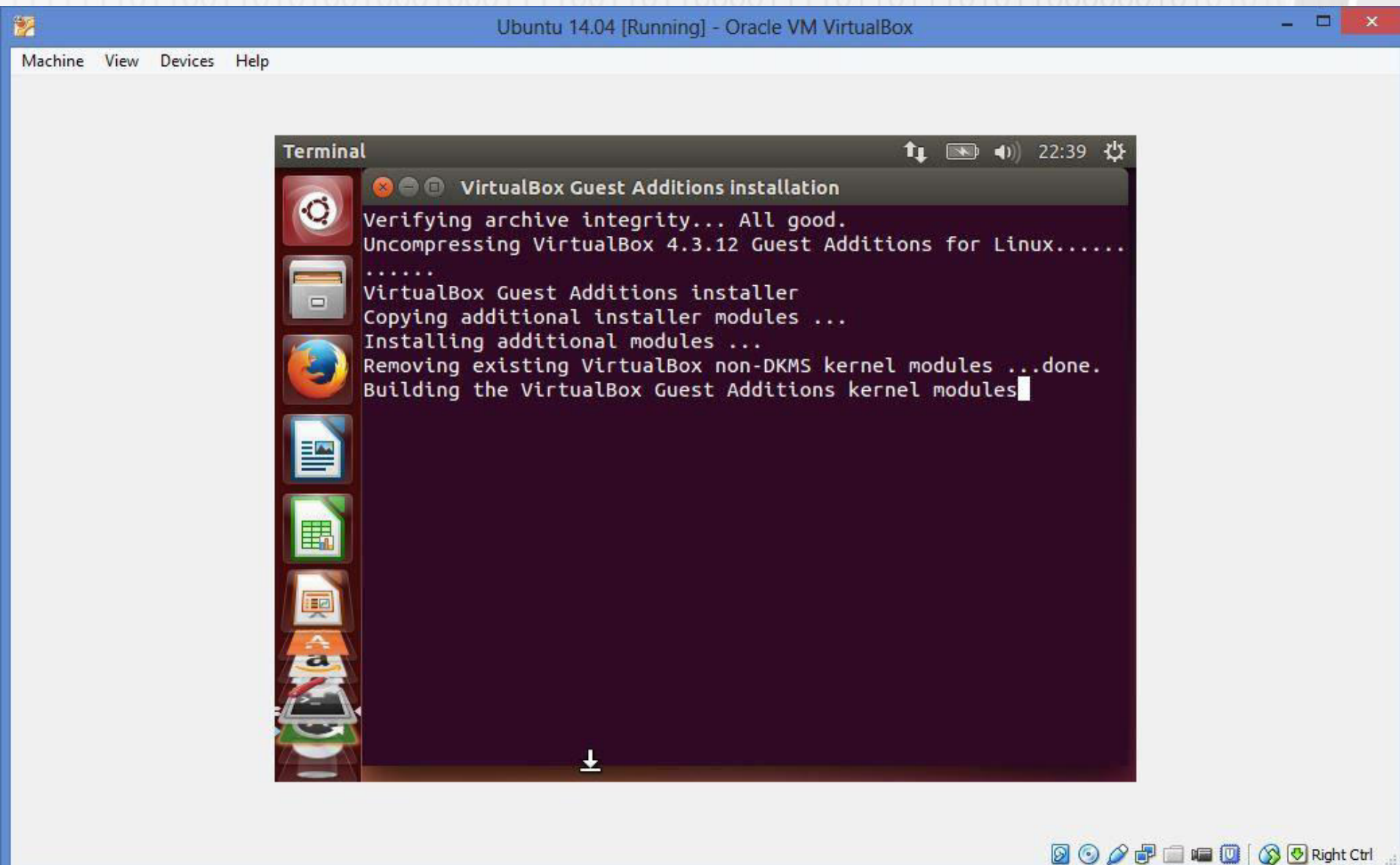


A mai nap anyaga...



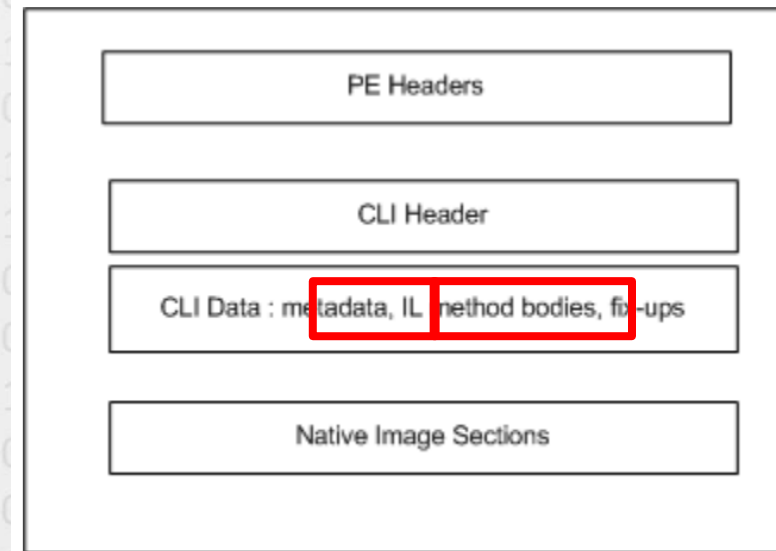
- **.NET Framework: „menedzselte futtatókörnyezet”**
- **Standardok által definiált módon működik (ECMA 334, 335)**
- **Tartalma:**
 - Virtuális gép
 - + futásidejű szolgáltatások
 - Osztálykönyvtárak
- **„.NET Framework Dev Pack”**
 - Kb. az SDK
 - „Workload”-okat támogat 

Virtuális gép???

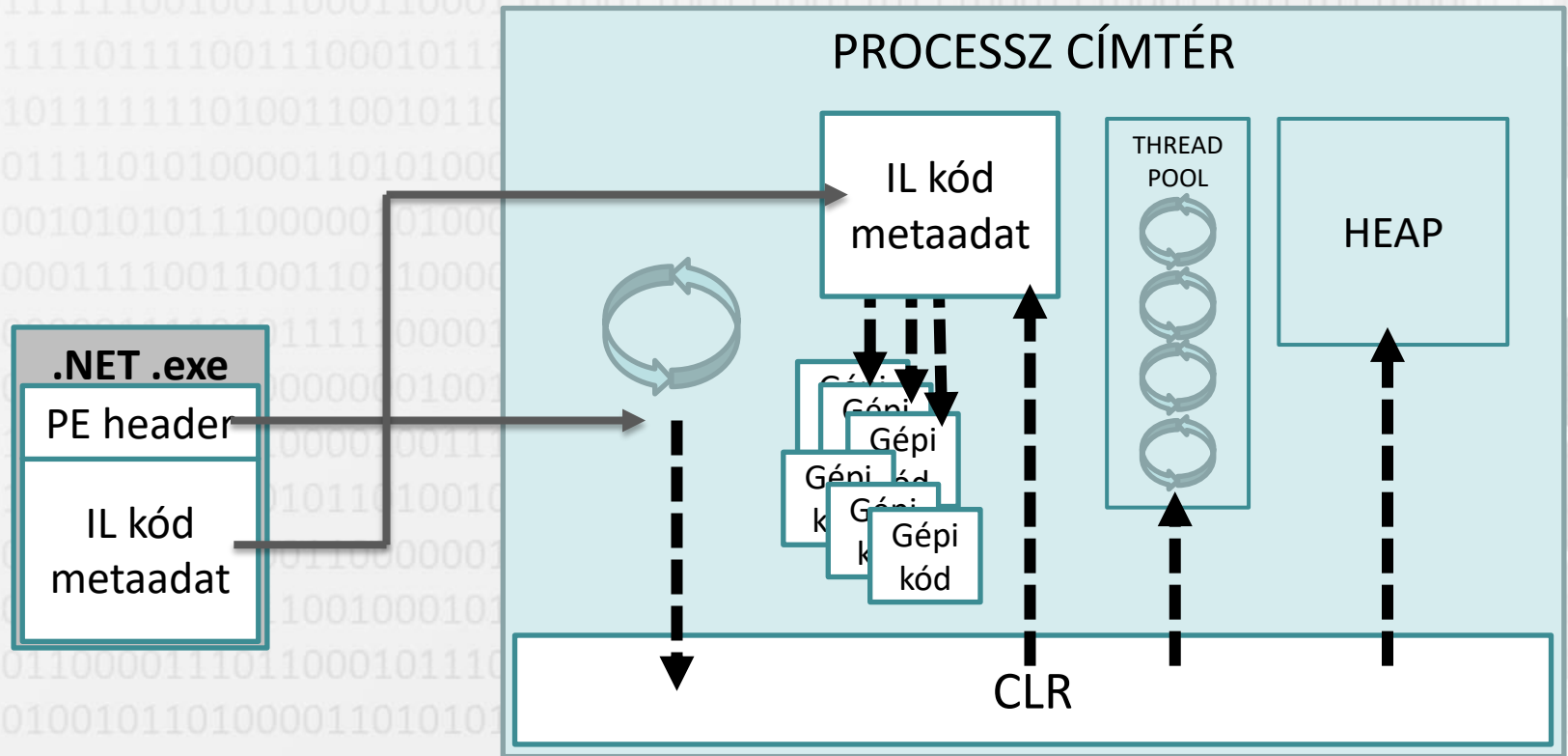


Common Language Runtime

- **Processz szintű virtualizációt támogat**
- **Bytekódot futtat, stack alapú végrehajtással**
- **Intermediate Language, IL / CIL, MSIL**
 - C# / VB / F# / Visual C++ / egyéb fordító állítja elő (.NET Framework Dev Pack)
 - Önmagában nem végrehajtható, .NET CLR értelmezi, fordítja gépi nyelvre és hajtja végre
- **Formátum: PE (Portable Executable), mint minden más Windowsos .exe és .dll!**
 - ... A .NET 1 nem volt az OS része, a Windows nem ismerte a .NET fogalmát...
 - ... Így natív kód tölti be a CLR-t, amely elolvassa és végrehajtja a fájl többi részét
 - A PE struktúra nagy része üres / előre definiált adatot tartalmaz



Common Language Runtime



- **JIT: on-demand, a metódus első futtatásakor áll elő a gépi kód**
- **... a JIT Compiler (CLR futásidejű szolgáltatás) által**
 - Optimalizáció: NGEN

(... vs Java JAR)

- **OS-függetlenség volt előtérben, nem az integráció: saját, könnyen olvasható formátum**
- **Java .class fájlok + metaadat, ZIP tömörítéssel**
 - Kicsomagolható: `jar -xf foo.jar`
 - Metaadat: `META-INF/MANIFEST.MF`
- **Java .class: 1 Java osztály adatai változó hosszúságú mezőkkel + bytekód**
- **.java → build → .class → JIT → JVM → OS**
- **Android DEX: azonos megközelítés, JAVA syntax, de más bytekód a végeredmény (Nincs JRE!)**
- **Java/Dart code (esetleg előfeldolgozóval: Kotlin/Flutter) → build → DEX/APK → JIT/AOT → kezdetben DVM (Dalvik VM), később ART (Android RunTime)**

.NET IL kód

```
public class Program
{
    static void Main()
    {
        Console.WriteLine("hello world");
    }
}
```

```
.class public auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
{
    // Methods
    .method public hidebysig static
        void Main () cil managed
    {
        // Method begins at RVA 0x2050
        // Code size 13 (0xd)
        .maxstack 8

        IL_0000: nop
        IL_0001: ldstr "hello world"
        IL_0006: call void [mscorlib]System.Console::WriteLine(string)
        IL_000b: nop
        IL_000c: ret
    } // end of method Program::Main
```

- Objektorientált assembly-szerű nyelv
- A .NET fordító generálja (C# / VB / F# / Visual C++ / etc.)
- JIT (vagy NGEN) fordítja a processzor által értelmezhető gépi kóddá
- Platformfüggetlen: verem (stack) műveletek

C# → IL → gépi kód

- **IL kód:**

- ildasm.exe (kész assemblyből)
- ILSpy / Reflector / DotPeek ... (kész assemblyből)
- <http://www.sharplab.io> (gévelt kód)

- **JIT-elt gépi kód:**

- Visual Studio breakpointon állva: Debug / Windows / Disassembly (futáskor)
- <http://www.sharplab.io> (gévelt kód)

The screenshot shows the SharpLab website interface. The left pane displays the C# source code for a simple program. The right pane shows the resulting IL code, which is a low-level representation of the C# code. The interface includes a browser window at the top, a code editor with tabs for 'Code' and 'Results', and a 'Debug' button.

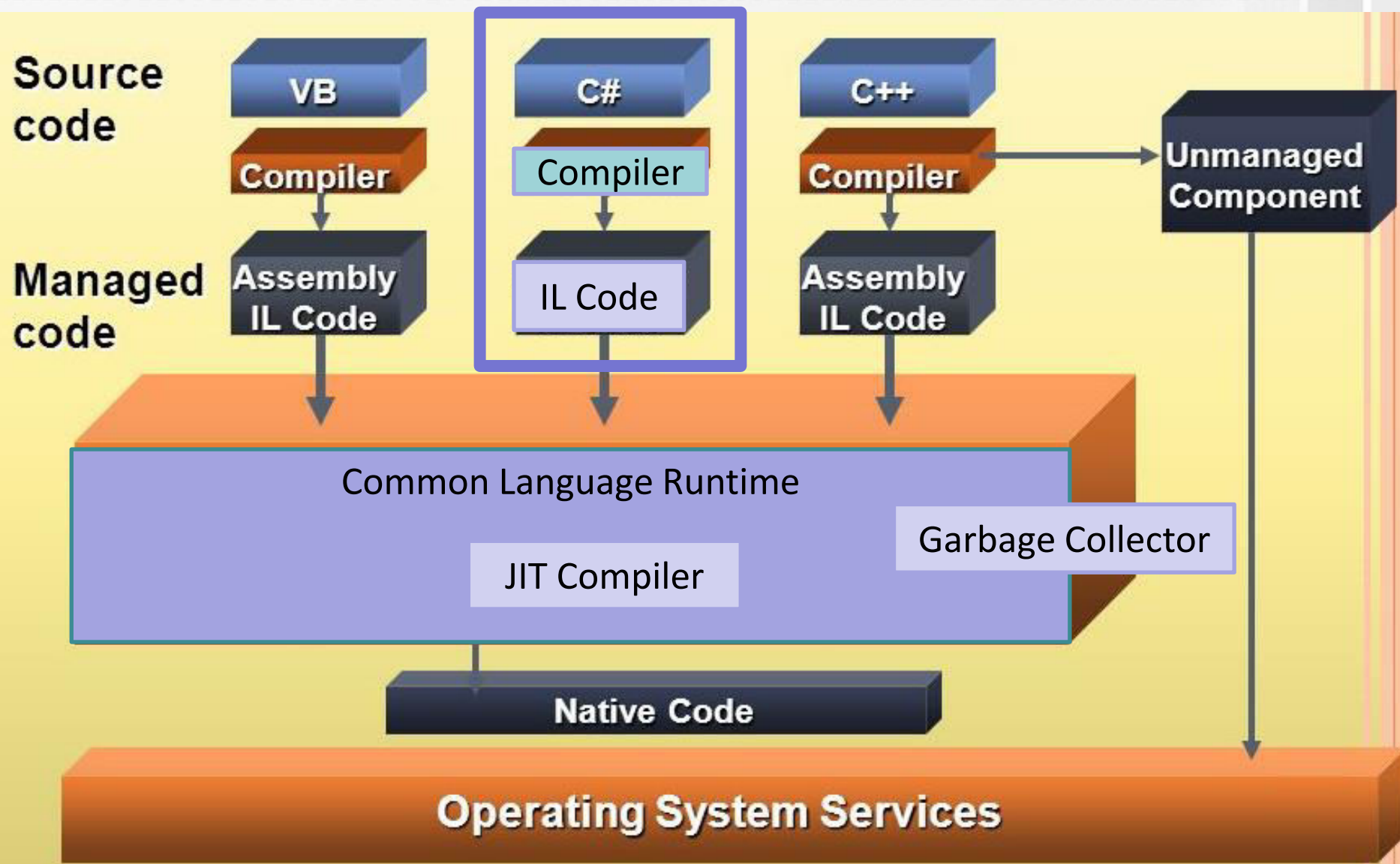
```
using System;
public class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello world!");
    }
}
```

```
.class private auto ansi '<Module>'
{
} // end of class <Module>

.class public auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
{
    // Methods
    .method public hidebysig static
        void Main () cil managed
    {
        // Method begins at RVA 0x2050
        // Code size 13 (0xd)
        .maxstack 8

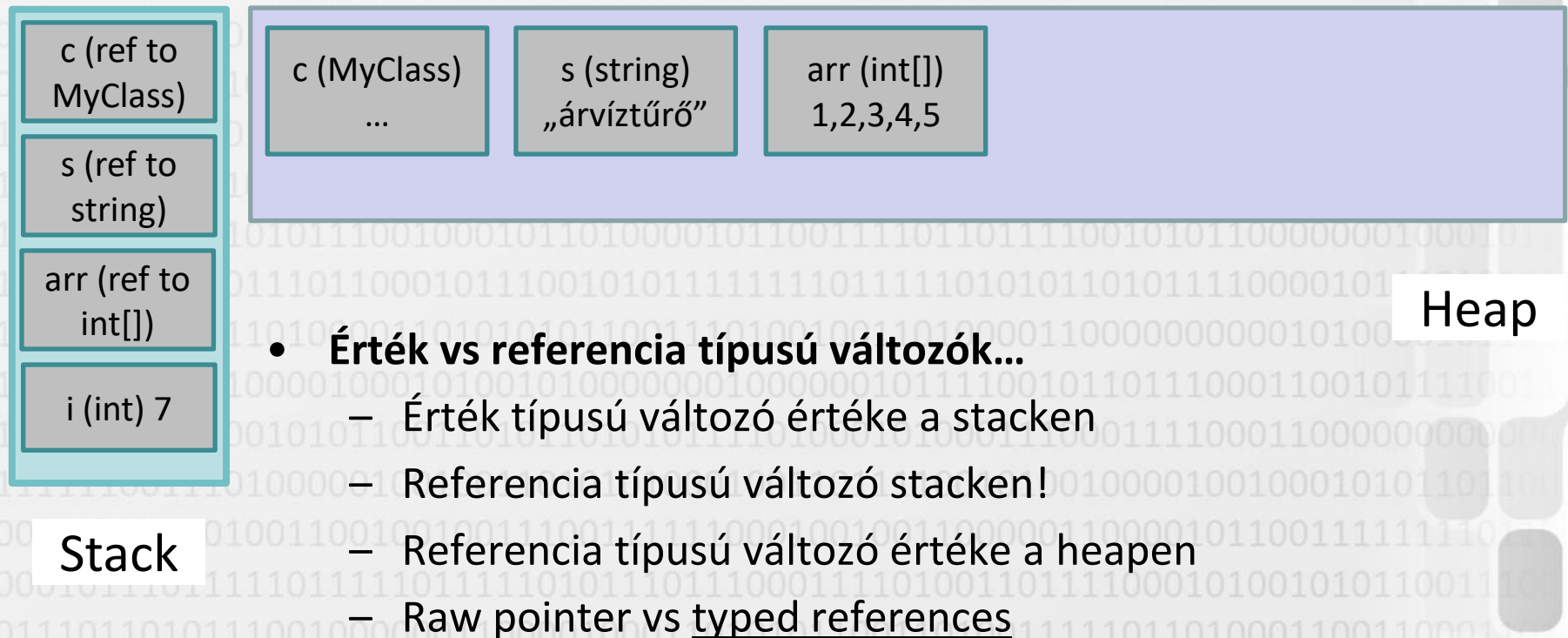
        IL_0000: nop
        IL_0001: ldstr "Hello world!"
        IL_0006: call void [mscorlib]System.Console::WriteLine(string)
        IL_000b: nop
        IL_000c: ret
    } // end of method Program::Main
```

.NET Architecture



.NET memóriakezelés – ismétlés...

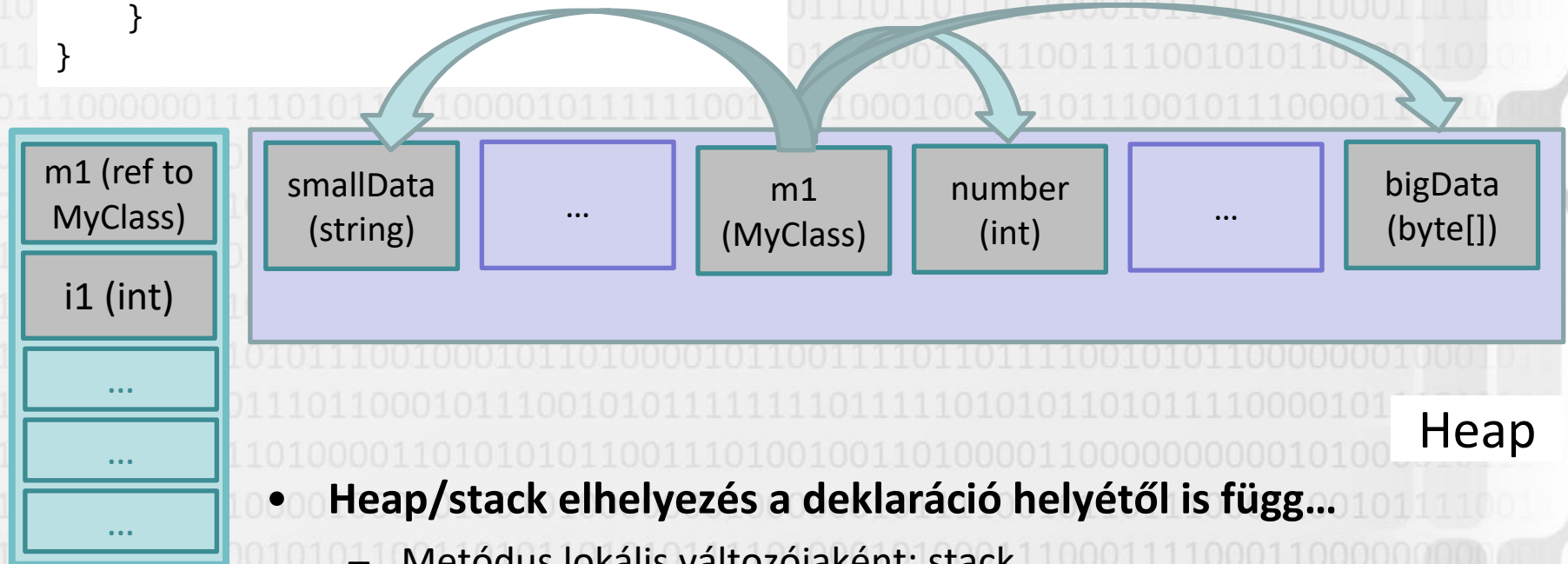
```
class Program
{
    static void Main(string[] args)
    {
        int i = 7;
        string s = "árvíztűrő";
        int[] arr = { 1, 2, 3, 4, 5 };
        MyClass c = new MyClass();
    }
}
```



.NET memóriakezelés – pontosítás...

```
class Program
{
    // ...
    static void DoSomething()
    {
        int i1 = 12;
        MyClass m1 = new MyClass();
        // do something...
    }
}
```

```
class MyClass
{
    public int number = 4;
    public string smallData = "0123456789";
    public byte[] bigData = new byte[86000];
}
```



Stack

Heap

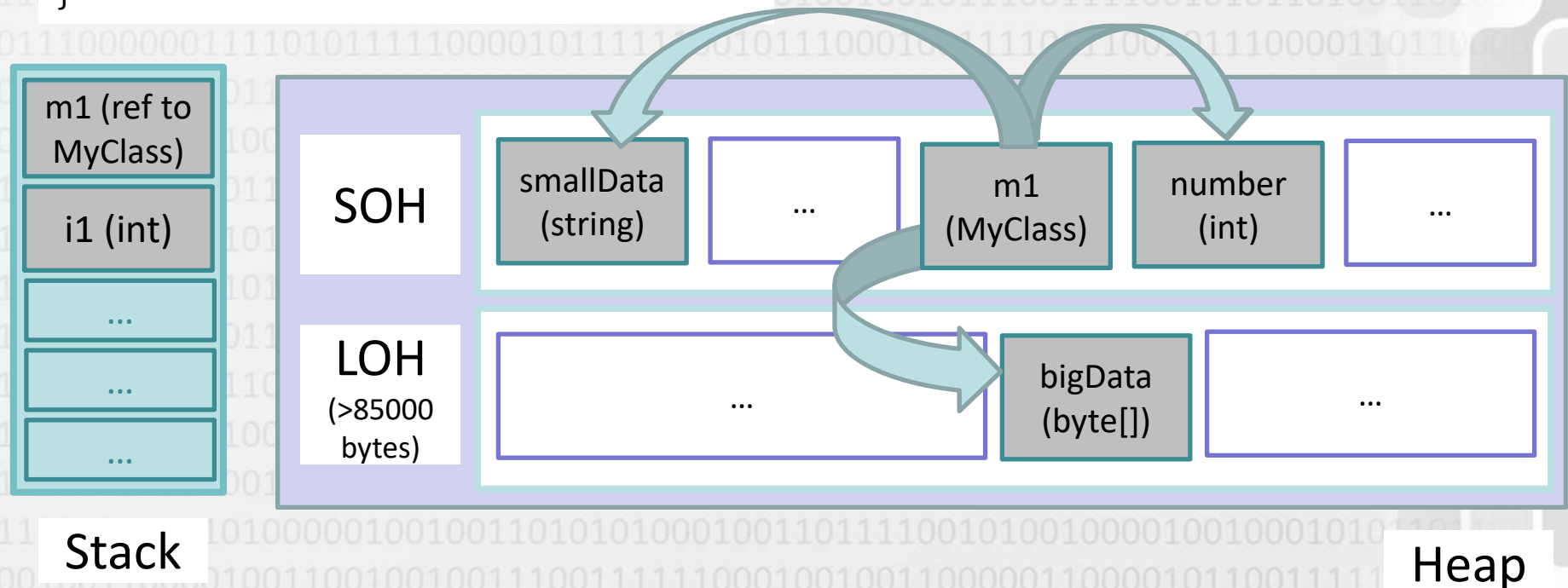
- **Heap/stack elhelyezés a deklaráció helyétől is függ...**

- Metódus lokális változójaként: stack
- Értéktípus **tag**jaként (pl. Point.X): stack (nincs példa a dián!!!)
- Ref. típus **tag**jaként: heap

.NET memóriakezelés – pontosítás...

```
class Program
{
    // ...
    static void DoSomething()
    {
        int i1 = 12;
        MyClass m1 = new MyClass();
        // do something...
    }
}
```

```
class MyClass
{
    public int number = 4;
    public string smallData = "0123456789";
    public byte[] bigData = new byte[86000];
}
```



.NET memóriakezelés



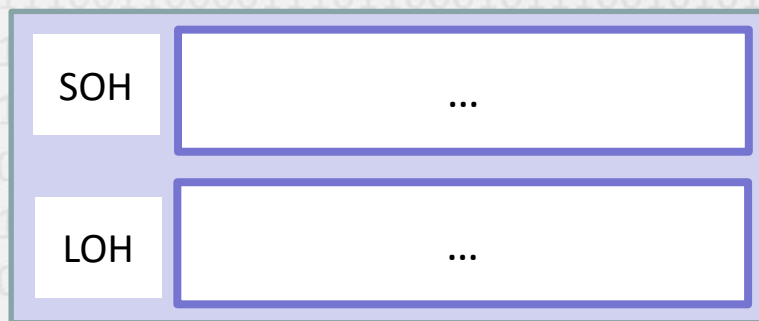
Stack

- **A stack „önmenedzselő”**

- A program futása során kerülnek rá és kerülnek le róla az elemek
- 1stack/szál

- **A heap menedzseléséért a Garbage Collector (GC) felel**

- Memóriát kér az OS-tól, illetve ami már nincs használatban, azt visszaadja
- Memóriát biztosít a program számára
- Eldönti, hogy a memória mely része van még használatban
- A használatlan részt kitakarítja (újrafelhasználás miatt)



Heap

.NET memóriakezelés

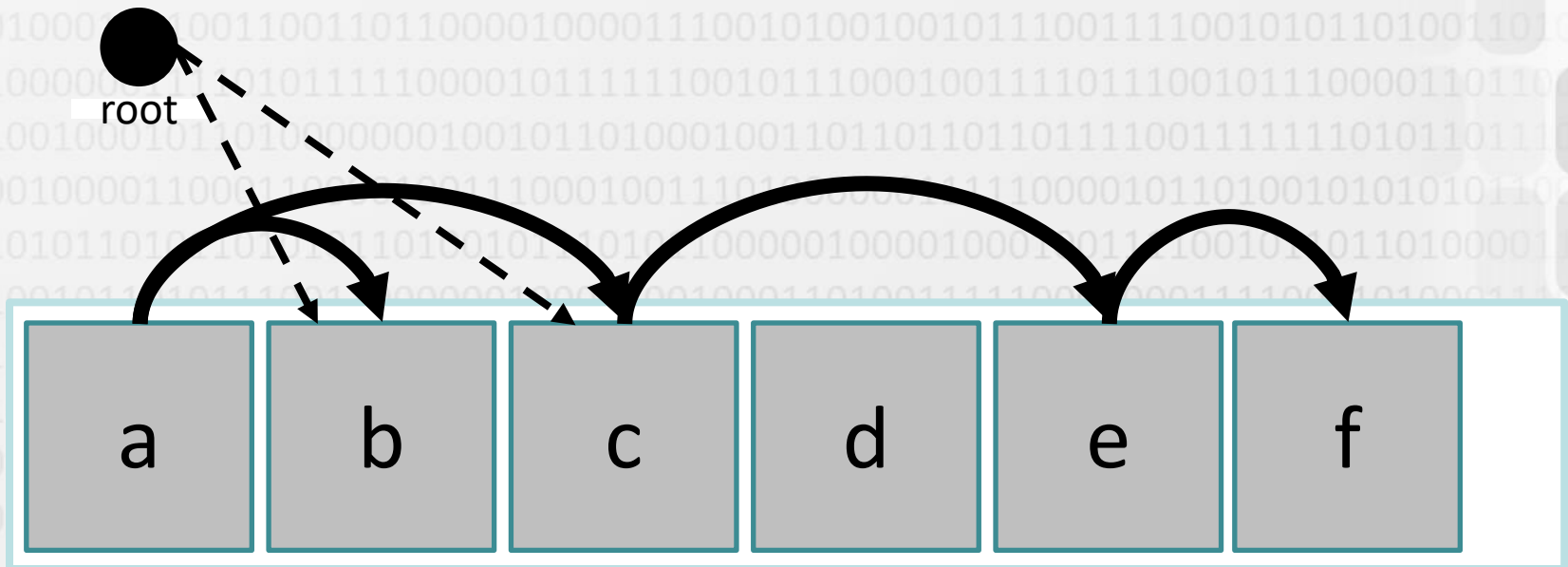
- Kitakarítja... Mikor?
- *Ismertetlen időben*
- „Not part of the documented, guaranteed behavior”
- Kb:
 - Ha kevés a fizikai memória (OS jelzés)
 - Ha a heapen található objektumok által használt memóriamennyiség meghalad egy küszöböt
 - A program futása során változik
 - A GC.Collect() metódus hívásakor
- Nagyon jó mechanizmus, elvileg nem kell fejlesztés közben foglalkozni vele → IDisposable interface???

Automatikus szemétgyűjtés menete

- A GC feltételezi, hogy a memóriában minden szemét
- Referenciakövetéssel győződik meg, mely objektumok vannak még használatban
 - A követés az ún. root objektumokból indul:
 - Stack referenciák
 - Statikus objektumreferenciák
 - „Finalization queue”-n lévő objektumok referenciái
 - CPU regiszterek
 - Interop objektumok referenciái
- **Menedzselt nyelvekben is lehetséges memory leak, ha váratlan helyeken megmaradnak referenciák**
 - .NET event/delegate – ha elfelejtünk leiratkozni...
 - Statikus referenciák (pl. singleton által!)
 - Fordító által generált konstrukciók miatt: pl. lambda closures, outer variable trap

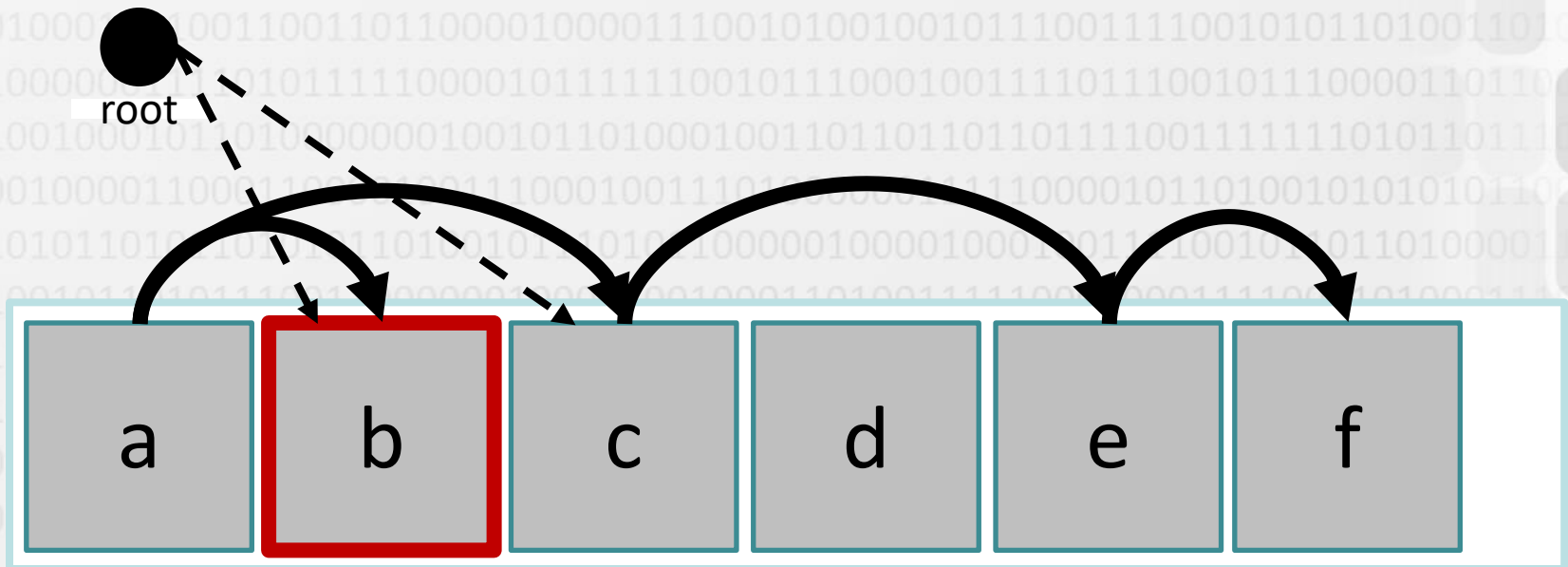
Referenciák bejárása

- „Graph of reachable objects”



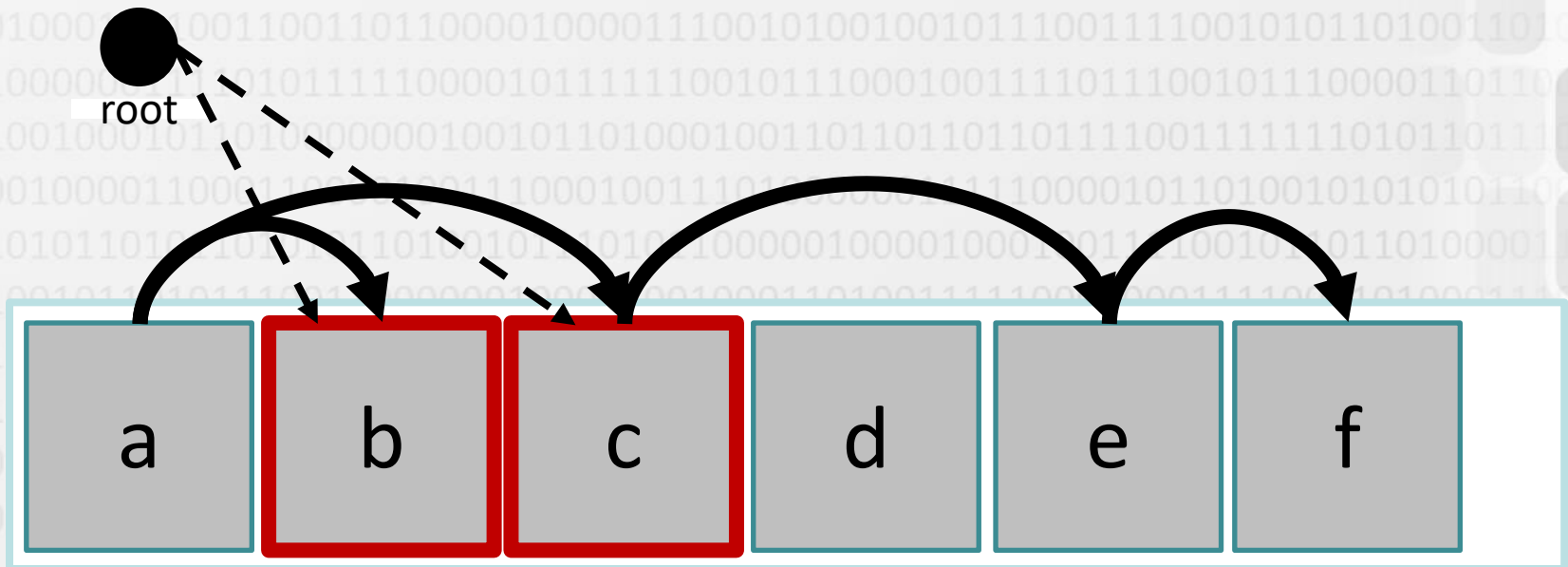
Referenciák bejárása

- „Graph of reachable objects”



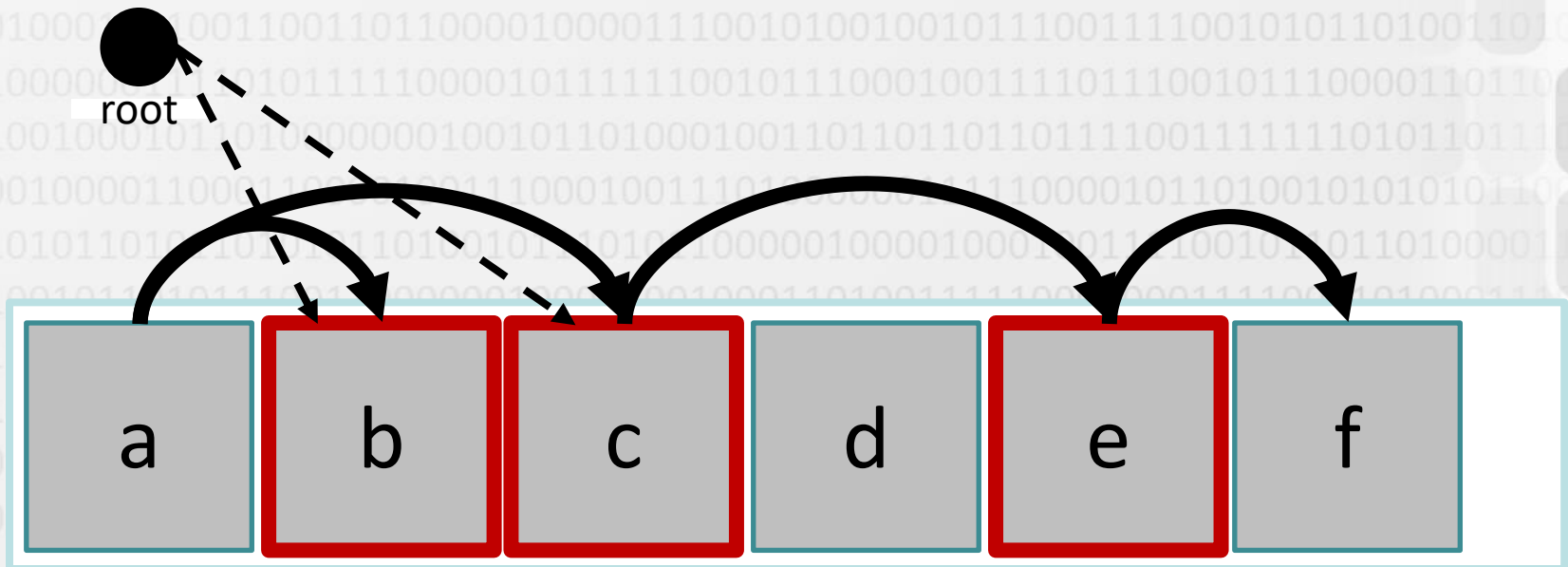
Referenciák bejárása

- „Graph of reachable objects”



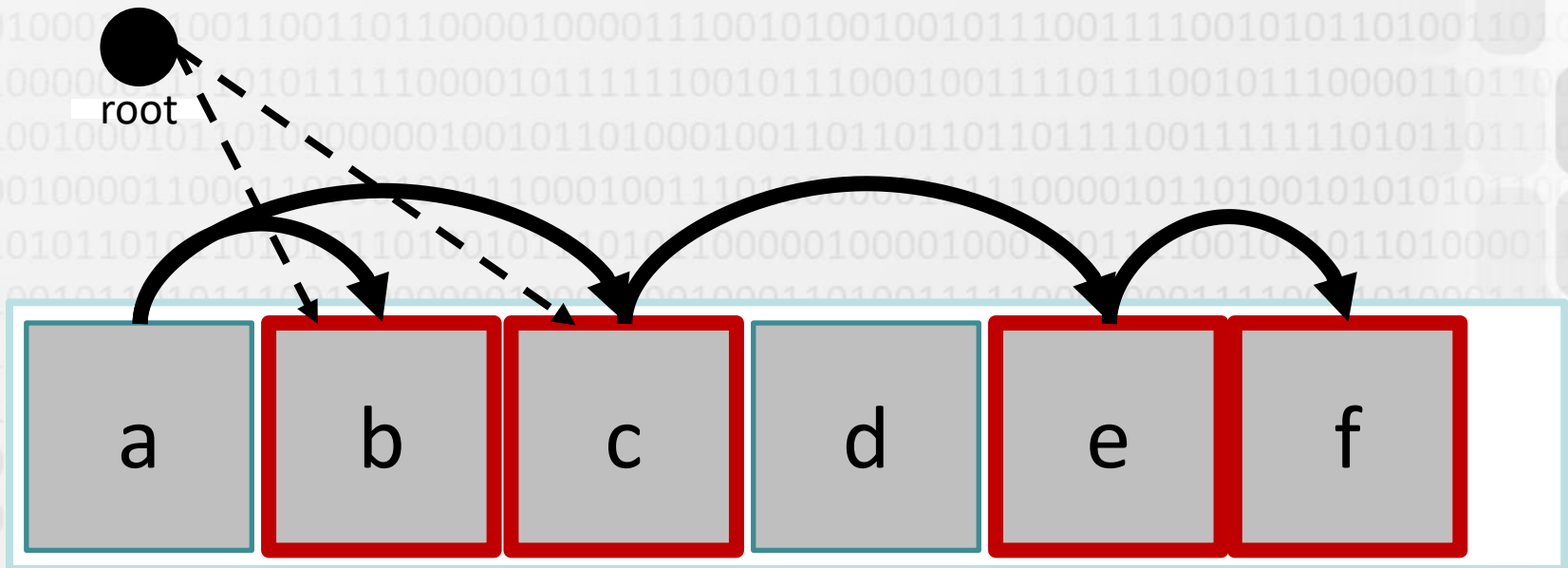
Referenciák bejárása

- „Graph of reachable objects”

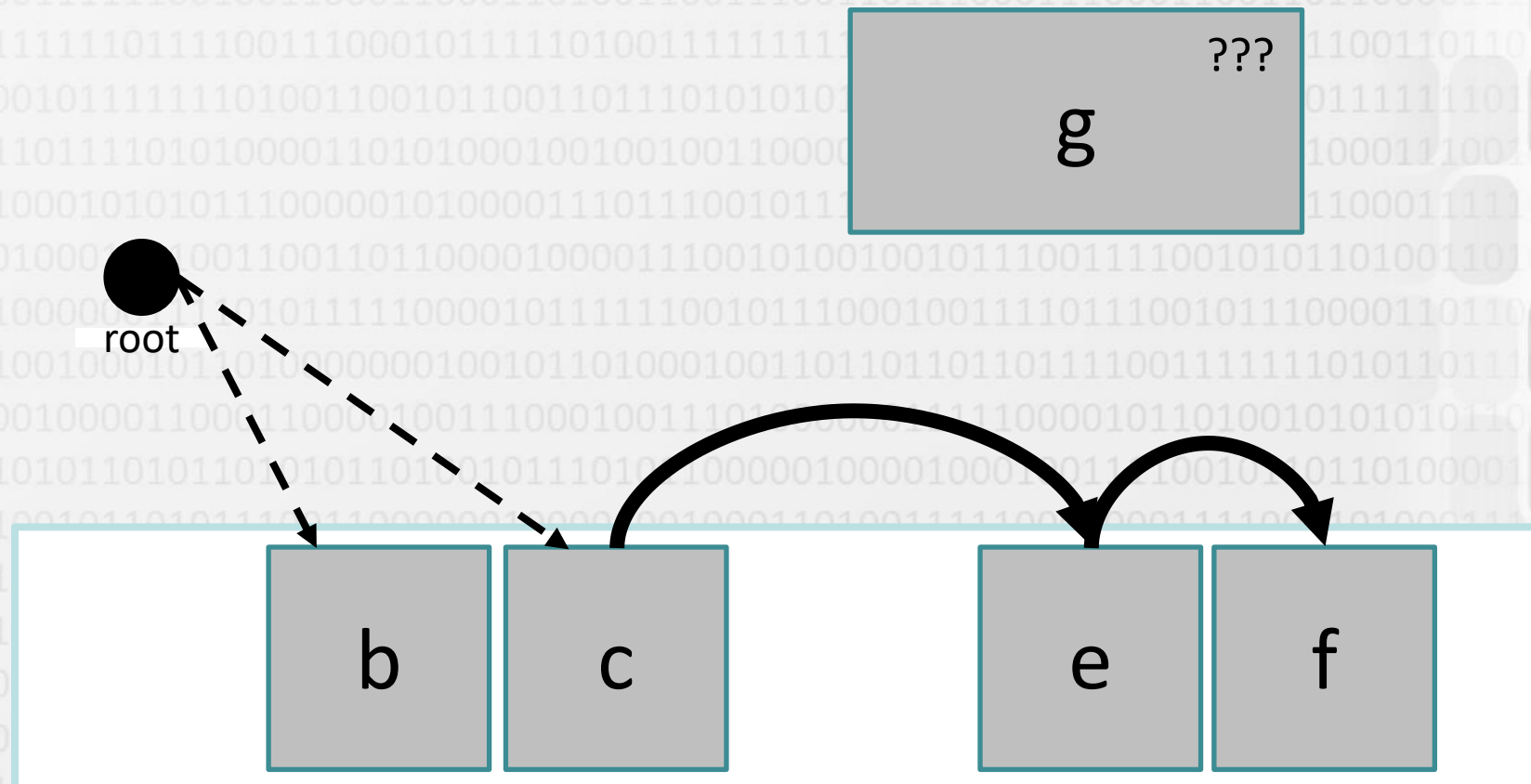


Referenciák bejárása

- „Graph of reachable objects”

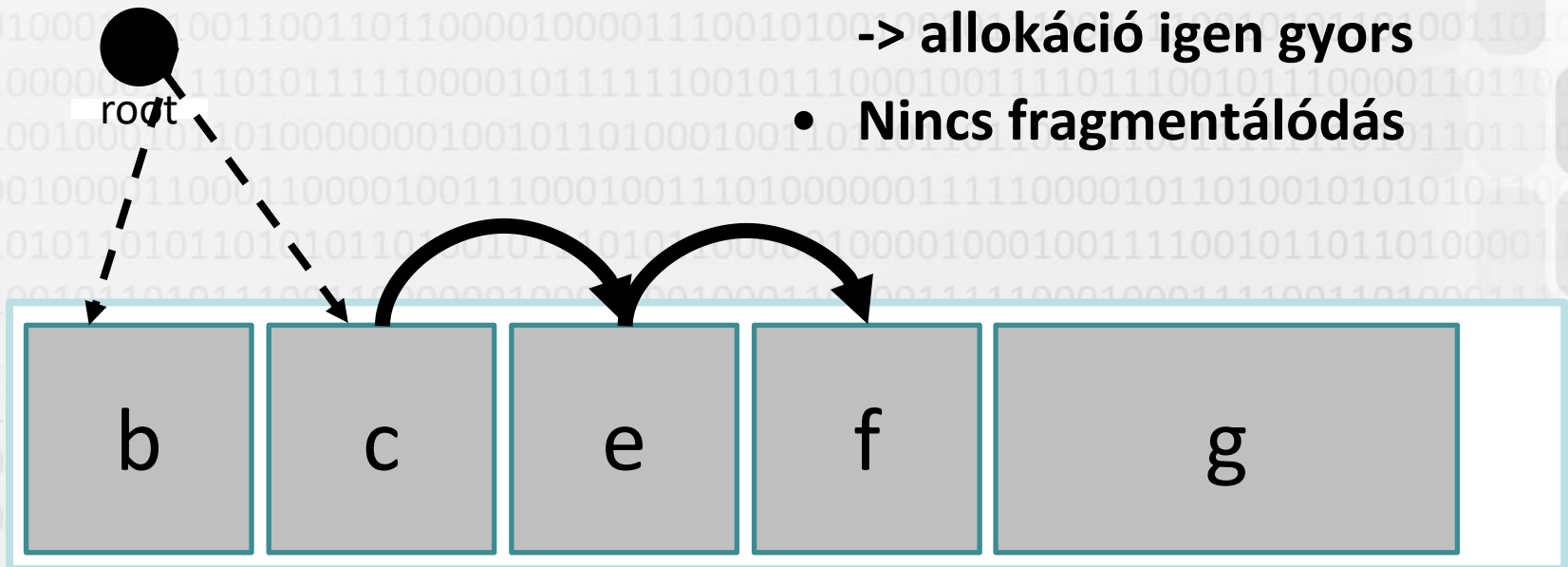


Szemétygyűjtés



Tömörítés

- Kivéve: LOH (<.NET 4.5.1)
 - Túl nagy objektumok...
- Kivéve: „pinnelt” objektumok
- Folytatólagosan tartja a memóriában az objektumokat
-> allokáció igen gyors
- Nincs fragmentálódás



.NET memory leak + event

```
public class MyClassWithEvent
{
    public event EventHandler MyEvent;
}

public class MySubscriberClass
{
    public void OnMyEvent(object sender, EventArgs eventArgs)
    {
        Console.WriteLine($"{sender} fired MyEvent!");
    }
}

static MyClassWithEvent myClassWithEvent = new MyClassWithEvent();

public static void Main()
{
    MySubscriberClass subscriber = new MySubscriberClass();
    myClassWithEvent.MyEvent += subscriber.OnMyEvent;
    // myClassWithEvent hívkozik a subscriber példányra, amíg
    // myClassWithEvent.MyEvent -= subscriber.OnMyEvent;
    // ... Itt nem gond, minden felszabadul a Main() után ...
}
```

.NET memory leak + event

```
static MyClassWithEvent myClassWithEvent = new MyClassWithEvent();

public static void OtherMethod()
{
    MySubscriberClass subscriber = new MySubscriberClass();
    myClassWithEvent.MyEvent += subscriber.OnMyEvent;
    // ...
}

// A subscriber példány a metódus után elvileg nem elérhető → ???
```

- **Kötelező mindig**

- 1000% biztosnak lenni abban, hogy a feliratkozó és az esemény forrás ugyanakkor felszabadítható (a kettőnek azonos az élettartama)
- Leiratkozni az eseményről saját művelettel (a -= operátorral)
- Leiratkozni az eseményről megfelelően implementált IDisposable interfésszel (full IDisposable pattern, ld. később)
- Hagyni a Garbage Collectort, hogy felszabadítsa a subscriber példányt Weak Delegates Design Pattern segítségével (prog4), és utána rendesen le tudunk iratkozni az eseményről, ha a subscribert pont felszabadította a GC

Generációk

- **Probléma: a GC lassúúúúúú...**
- **Feltételezés: a program futása során a legtöbb objektum rövid életű**
- **Az objektumok korát a GC „generációkkal” követi**
 - gen0: rövid életű → gen2: hosszúéletű objektumok
- **A szemétgyűjtés mindig generáció(k)ra érvényes**
 - gen0 GC: gen0 objektumokat takarítja
 - gen1 GC: gen0 + gen1 objektumokat takarítja
 - gen2 GC: gen0 + gen1 + gen2 objektumokat takarítja
- **Ha az objektum túléli a szemétgyűjtést, a következő generációba lép elő**

GC.Collect()

- Csak különleges szituációkban (= soha ☺)!

```
class MyGCCollectClass
{
    private const long maxGarbage = 1000;
    void MakeSomeGarbage()
    {
        Version vt;
        for (int i = 0; i < maxGarbage; i++)
        {
            vt = new Version();
        }
    }

    static void Main()
    {
        MyGCCollectClass myGCCol = new MyGCCollectClass();
        myGCCol.MakeSomeGarbage();
        GC.Collect();           // trigger GC manually
        GC.KeepAlive(myGCCol); // do not clean up myGCCol
    }
}
```

Teljesítménykérdések – GC beállítások

- A GC szemétygyűjtéskor ideiglenesen leállítja az összes threadet
 - Pl. adatfeldolgozáskor, megjelenítéskor is...

```
GCSettings.LatencyMode = GCLatencyMode.SustainedLowLatency;
```

| | |
|---------------------|--|
| Batch | Gen2 szemétygyűjtés előtérshálón |
| Interactive | Gen2 szemétygyűjtés háttérshálón |
| LowLatency | Letiltja a gen2 szemétygyűjtést (de ha mégis, előtérshálón történik) |
| SustainedLowLatency | Letiltja a gen2 szemétygyűjtést (de ha mégis, háttérshálón történik) |
| NoGCRegion | Ideiglenesen nincs szemétygyűjtés |

- LOH tömörítés (>.NET Framework 4.5.1)

```
GCSettings.LargeObjectHeapCompactionMode =  
    GCLargeObjectHeapCompactionMode.CompactOnce;  
GC.Collect();  
  
Console.WriteLine(GCSettings.LargeObjectHeapCompactionMode);
```

Default

Teljesítménykérdések – GC beállítások

```
static void Main()
{
    // NoGCRegion
    if (GC.TryStartNoGCRegion(maxGarbage*16))
    {
        try
        {
            // ... do extremely performance critical allocations...
        }
        finally
        {
            if (GCSettings.LatencyMode == GCLatencyMode.NoGCRegion)
            {
                GC.EndNoGCRegion();
            }
        }
    }
}
```

Teljesítménykérdések – destruktor

- **Menedzseletlen erőforrás == amiről a GC nem tud**
 - Pl.: `FileStream -> Win32.CreateFile() -> IntPtr...`
 - vagy natív kódhívásnál (pl: dll-ek!)
 - Normál: `FileStream.Dispose()`
 - Fallback: `~FileStream()`
- **Speciális GC algoritmus...**
 - Minden destruktorral rendelkező objektum +1 helyről hivatkozva van: a „finalization queue”-ról
 - GC-kor, ha már nincs rá hivatkozás, átkerül a „fReachable queue”-ra
 - „Finalization thread” futtatja a destruktorokat...
- **... Csak a következő GC takarítja ki**
- **Következmények:**
 - Ideiglenes root objektum (mert az összes hivatkozott objektum is meg kell maradjon a destruktor futásáig)
 - 2 GC kell a kitakarításához
 - Az első GC után elő is lép!
 - Gen2 objektumoknál eleve ritka a GC...
 - Ismeretlen időben meghívott (a GC által)

Dispose

- Ne csináljunk destruktort, ha nem nagyon muszáj!
- Egyszerű Dispose pattern (csak menedzselt erőforrás)

```
public class ClassWithBasicDispose : IDisposable
{
    public void Dispose()
    {
        Dispose(true); // usually not needed - put code into Dispose()
        GC.SuppressFinalize(this); // take object off the finalization q
    }

    protected virtual void Dispose(bool disposing)
    {
        if (disposing)
        {
            // Dispose() all managed resources
        }
    }
}
```

Dispose

→ Full Dispose pattern (ha van unmanaged erőforrás is)

```
public class ClassWithFullDispose : IDisposable
{
    private bool disposed = false;

    protected virtual void Dispose(bool disposing)
    {
        if (this.disposed) return;
        if (!this.disposed)
        {
            if (disposing)
            {
                // Dispose() all managed resources
            }

            // Cleanup all unmanaged resources - e.g. close handles,
            // free pointers, set large fields to null
            disposed = true;
        }
    }

    // ... public void Dispose()
    //     ~ClassWithFullDispose()
```


Dispose

→ Full Dispose pattern (ha van unmanaged erőforrás is)

```
public class ClassWithFullDispose : IDisposable
{
    // ... protected virtual void Dispose(bool disposing)

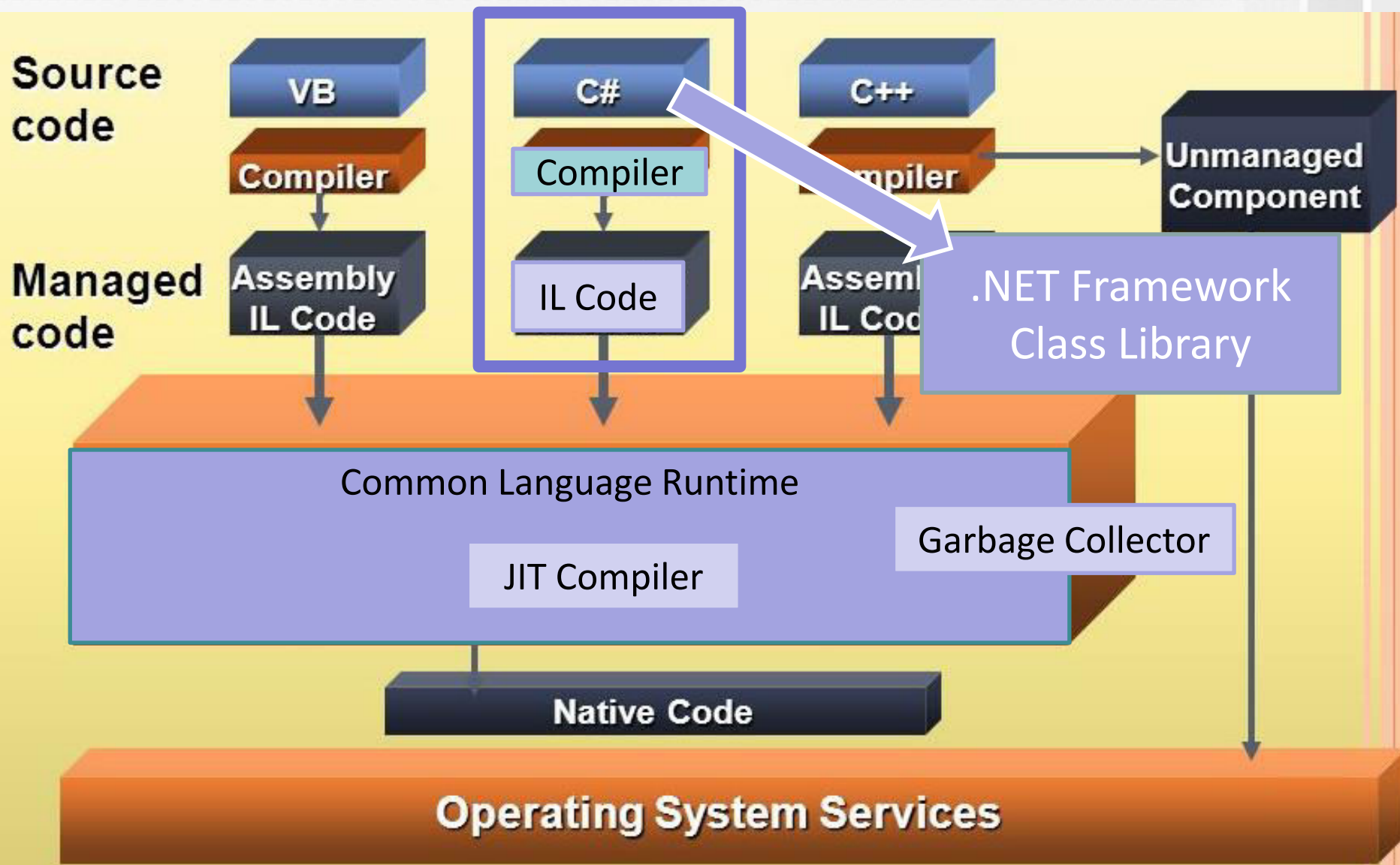
    public void Dispose()
    {
        Dispose(true);
    }
}

using (StreamReader sr = new StreamReader("TestFile.txt"))
{
    string line;
    // Read and display lines from the file until the end of
    // the file is reached.
    while ((line = sr.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

A GC teljesítményét rontó code smell-ek

- **Felesleges destruktorkok**
- **Túl sok allokáció**
 - Pl. sok `string.Split()[ind]` vs sok `Substring()` ... -> gyakoribb GC
- **Túl nagy allokációk**
 - Pl. a kelleténél nagyobb tömbök -> hosszabb GC
- **Egymásra össze-vissza referáló objektumok**
 - Különösen, ha rövid életűek -> GC végigköveti, gen0 objektumokat ráadásul gyakran
- **„Root objektum szaporítás”**
 - Pl. nagyon mély rekurzió/függvényhívási lánc, sok referencia lokális változóval
- **„Közepes élettartamú” objektumok**
 - gen2-ről már nehezen „szabadulnak ki”

.NET Architecture



.NET Framework Class Library

- Framework által biztosított típusok (osztályok, interfészek, delegáltak, érték típusok)
- Legtöbbjüket egy speciális, CLS-nek nevezett szabványt követve programozták
 - CLS: Common Language Specification (nyelvek közötti interoperáció miatt)

```
public class Data
{
    public void Read()
    {
        using (StreamReader stream = new StreamReader(@"C:\programs\file.txt"))
        {
            var valid = true;
            while (valid)
            {
                var line = stream.ReadLine();
                if (line == null)
                {
                    valid = false;
                }
            }
        }
    }
}
```

```
type Data() =
    member x.Read() =
        // Read in a file with StreamReader.
        use stream = new StreamReader @"C:\pr

        // Continue reading while valid lines
        let mutable valid = true
        while (valid) do
            let line = stream.ReadLine()
            if (line = null) then
                valid <- false
            else
```

.NET Framework Class Library

The screenshot shows a Windows File Explorer window titled "v4.0.30319". The address bar displays the path: "Windows > Microsoft.NET > Framework > v4.0.30319". The left sidebar shows a list of folders named "ConsoleApp4" through "ConsoleApp21". The main pane displays a table of files in the .NET Framework Class Library.

| Name | Date modified | Type | Size |
|---|------------------|-----------------------|-------|
| System.Data.Services.Client.dll | 12/04/2018 01:35 | Application extens... | 435 |
| System.Data.Services.Design.dll | 12/04/2018 01:35 | Application extens... | 171 |
| System.Data.Services.dll | 12/04/2018 01:35 | Application extens... | 658 |
| System.Data.SqlXml.dll | 12/04/2018 01:35 | Application extens... | 718 |
| System.Deployment.dll | 12/04/2018 01:35 | Application extens... | 859 |
| System.Design.dll | 12/04/2018 01:35 | Application extens... | 4,904 |
| System.Device.dll | 12/04/2018 01:35 | Application extens... | 63 |
| System.Diagnostics.Contracts.dll | 12/04/2018 01:35 | Application extens... | 30 |
| System.Diagnostics.Debug.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.FileVersionInfo.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.Process.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.StackTrace.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.TextWriterTraceListen... | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.Tools.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.TraceSource.dll | 12/04/2018 01:35 | Application extens... | 29 |
| System.Diagnostics.Tracing.dll | 12/04/2018 01:35 | Application extens... | 40 |
| System.DirectoryServices.AccountManag... | 12/04/2018 01:35 | Application extens... | 290 |
| System.DirectoryServices.dll | 12/04/2018 01:35 | Application extens... | 414 |
| System.DirectoryServices.Protocols.dll | 12/04/2018 01:35 | Application extens... | 197 |

.NET ökoszisztéma

- .NET 1.0: 2002 február 13. (.NET Framework + Visual Studio)
- → .NET Framework 4.8: 2019. APR. 18. („the last one”)
- Mostanában: ... Web, mobil, felhő ...
 - .NET Framework
 - Framework Class Library
 - Portable Class Libraries
 - .NET Core
 - Roslyn Compiler
 - .NET Standard
 - ...

???

New Project

Recent

Installed

Visual C#

Windows Universal

Windows Classic Desktop

Web

.NET Core

.NET Standard

Android

Cloud

Cross-Platform

Extensibility

iOS

Test

tvOS

WCF

Other Languages

Other Project Types

Online

Not finding what you are looking for?

Open Visual Studio Installer

.NET Framework 4.6.1

Sort by: Default

C#

Cross-Platform App (Xamarin.Forms)

Visual C#

C#

WPF App (.NET Framework)

Visual C#

C#

Windows Forms App (.NET Framework)

Visual C#

C#

Console App (.NET Core)

Visual C#

C#

Console App (.NET Framework)

Visual C#

C#

Class Library (.NET Standard)

Visual C#

C#

Class Library (.NET Framework)

Visual C#

ASP.NET Core Web Application

Visual C#

ASP.NET Web Application (.NET Framework)

Visual C#

C#

Shared Project

Visual C#

C#

Class Library (Legacy Portable)

Visual C#

C#

WCF Service Application

Visual C#

Search (Ctrl+E)

Type: Visual C#

A project for creating a C# class library (.dll)

Name:

ClassLibrary5

Location:

Solution name:

ClassLibrary5

Browse...

Create directory for solution

Create new Git repository

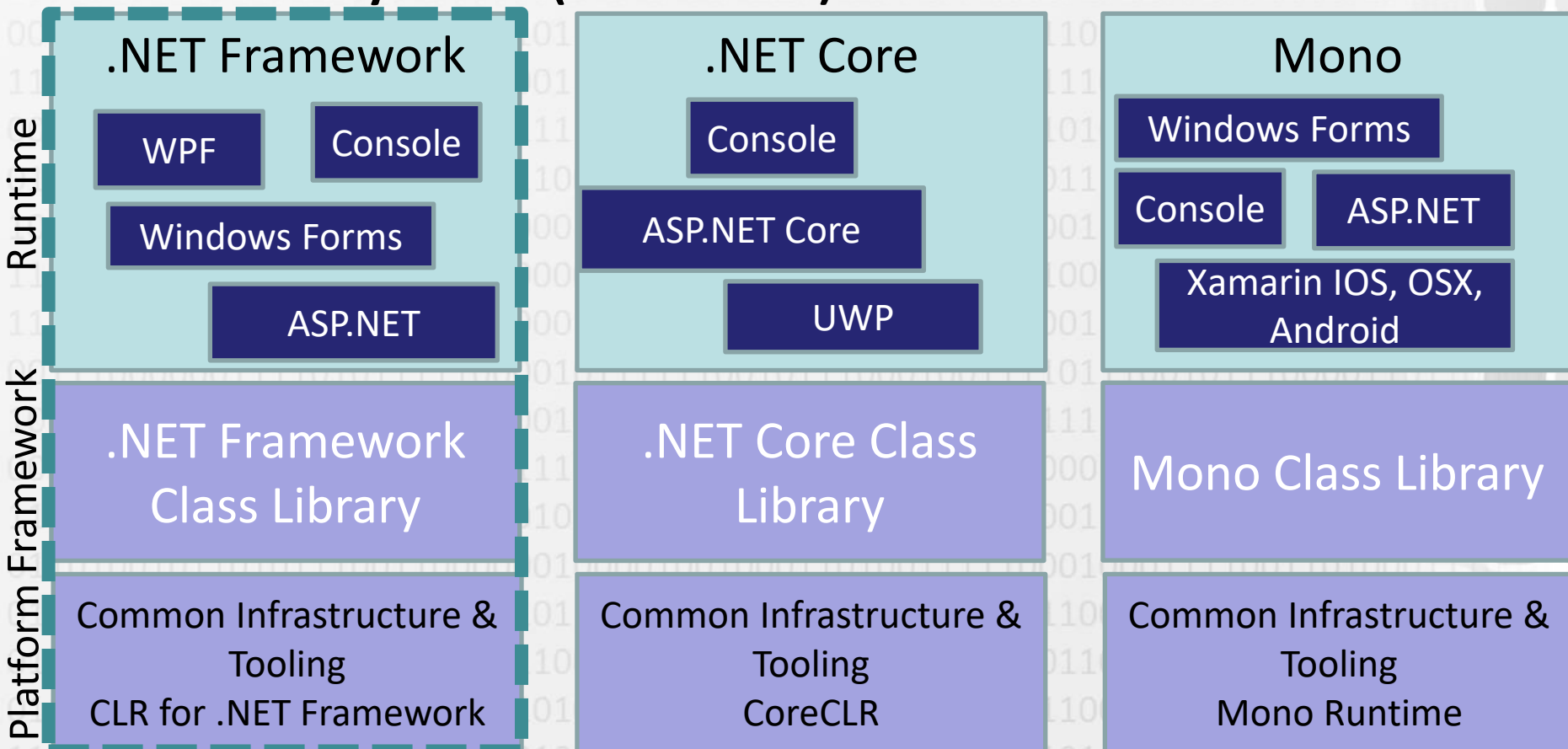
OK

Cancel

1011101101011100100000011000010001101010110011010011111011010001100110001

.NET ökoszisztéma

- **Futtatókörnyezetek (runtime-ok)**



- **Windows-centrikus**

- **Windows-függő API-k**

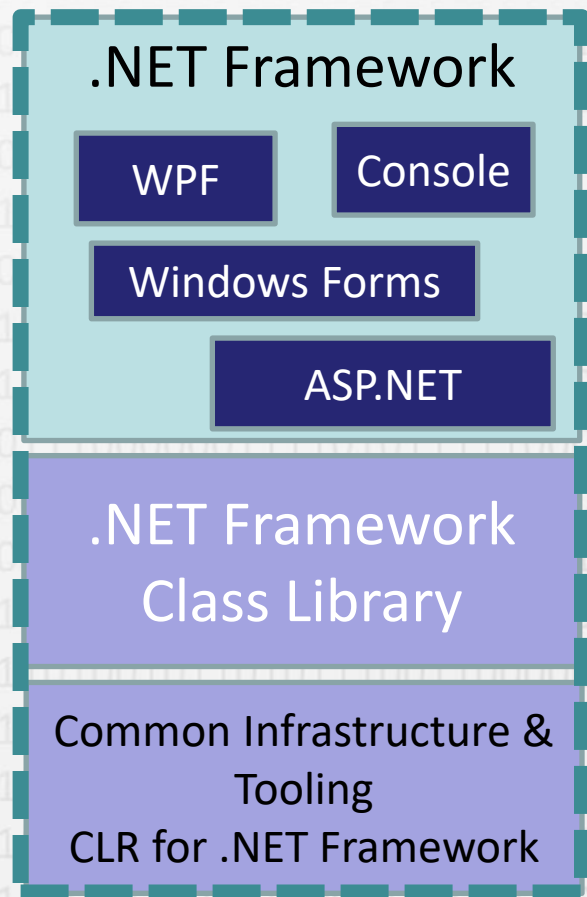
- **Crossplatform framework** (... app. workload-ok nem...)

- **Teljesítmény-orientált**

- **Crossplatform**

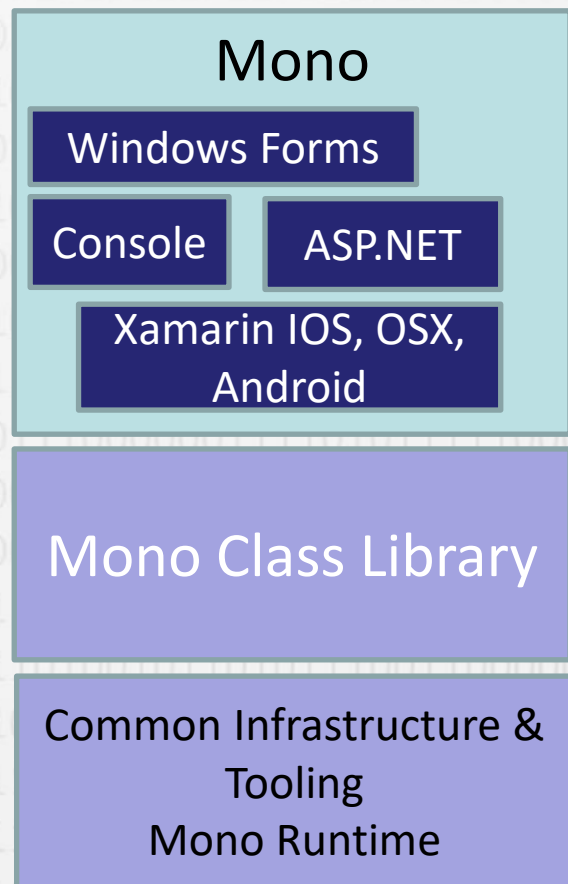
- **Xamarin: specifikus API-k: IOS, Android, Mac**

.NET Framework



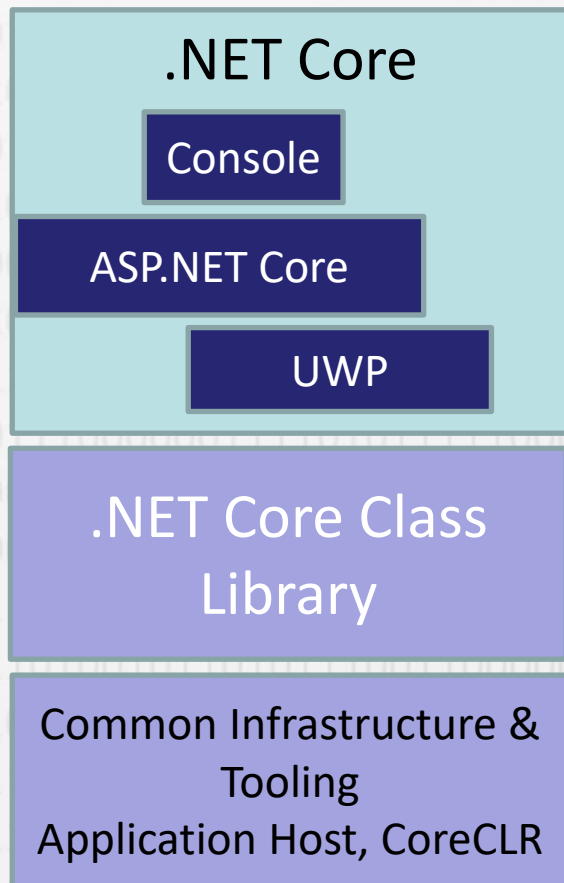
- **2002**
- **Hosszú ideig ez volt az egyetlen runtime**
- **Korlátozottan open-source:**
<https://github.com/microsoft/referencesource>
- **vs. .NET Core?**
 - Más use-case-eket támogat
 - Messze a legnagyobb workload-támogatás
 - Console
 - WCF (webszolgáltatások, kommunikáció)
 - WWF (üzleti folyamatok automatizálása)
 - WPF (komplex GUI)
 - Windows Forms (egyszerű GUI)
 - ASP.NET (Forms, MVC, WebApi) (web)
 - Azure (WebJobs, Cloud Services) (felhő)
 - Sok workload és funkcionalitás Windows-specifikus

Mono



- 2004 / 2011(Xamarin)
- Open-source:
<https://github.com/mono/mono>
- Crossplatform
- A Mono FCL a .NET Framework FCL funkcionalitásainak csak egy részét támogatja
- Workload-ok:
 - Console
 - Windows Forms
 - Apple IOS
 - Apple Mac OS X (Desktop GUI Mac-re)
 - Android

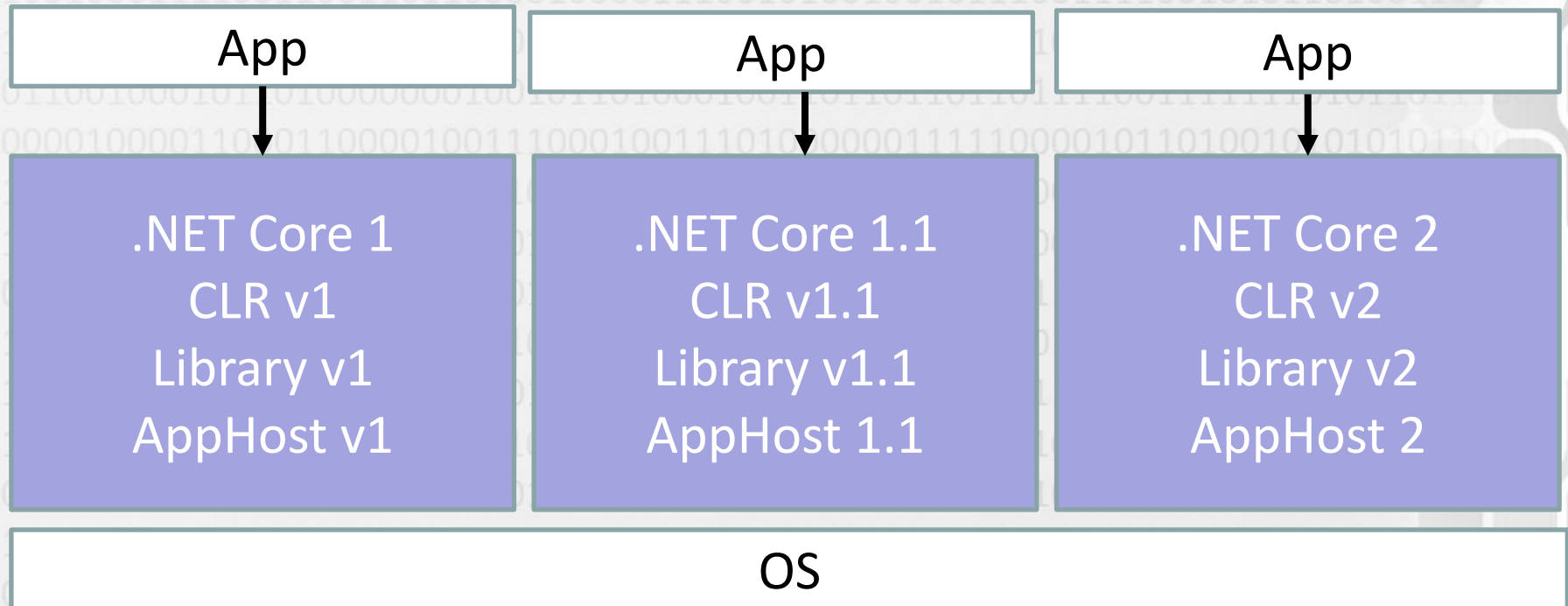
.NET Core



- 2016
- Open-source:
<https://github.com/dotnet/core>
- Crossplatform:
 - Windows client, server, IoT
 - Linuxok, FreeBSD, Mac OS X
- **KEZDETBEN** nem a .NET Framework „új változata” – más use case-eket szolgál ki
- A .NET Core FCL a .NET Framework FCL funkcionalitásainak csak egy részét támogatja
- **Workload-ok:**
 - Kezdetektől Console
 - Kezdetektől ASP.NET Core (MVC, Api)
 - 2.1 óta normális ORM (EF core)

.NET Core

- A célgépen installálva kell legyen, vagy self-contained
- Valódi side-by-side
 - User-level, computer-level



.NET Standard

.NET Framework

.NET Framework
Class Library

.NET Core

.NET Core Class
Library

Mono

Mono Class Library

- **FCL-ek funkcionalitásai különböznek**
- **Bonyolult a kódot megosztani 2 runtime között (C# kód, de...)**
 - Mert egy adott runtime-ra írt kód esetleg nem fordul a másakra (a megfelelő API-k hiánya miatt)
- **A régi Portable Class Library-k evolúciója...**
- **.NET Standard: specifikáció, amely leírja a használható API-kat/osztályokat**
 - Az összes RT implementál valamilyen .NET Standard verziót
 - Pl. .NET Framework 4.5 → .NET Standard <= 1.1

.NET Standard

- <https://docs.microsoft.com/en-us/dotnet/standard/net-standard>
- (2019.10.16.)

| .NET Standard | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 2.0 | 2.1 |
|-----------------------------|--------|--------|--------|--------|--------|--------------------|--------------------|--------------------|------------------|
| .NET Core | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 2.0 | 3.0 |
| .NET Framework ¹ | 4.5 | 4.5 | 4.5.1 | 4.6 | 4.6.1 | 4.6.1 ² | 4.6.1 ² | 4.6.1 ² | N/A ³ |
| Mono | 4.6 | 4.6 | 4.6 | 4.6 | 4.6 | 4.6 | 4.6 | 5.4 | 6.4 |
| Xamarin.iOS | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.14 | 12.16 |
| Xamarin.Mac | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.8 | 5.16 |
| Xamarin.Android | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 8.0 | 10.0 |
| Universal Windows Platform | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0.16299 | 10.0.16299 | 10.0.16299 | TBD |
| Unity | 2018.1 | 2018.1 | 2018.1 | 2018.1 | 2018.1 | 2018.1 | 2018.1 | 2018.1 | TBD |

.NET Standard

- **Additív verziózás**

```
...  
23 <NETStandard10Dependency Include="System.Threading" />  
24 <NETStandard10Dependency Include="System.Threading.Tasks" />  
25 <NETStandard10Dependency Include="System.Xml.ReaderWriter" />  
26 <NETStandard10Dependency Include="System.Xml.XDocument" />
```

1.0

```
27  
28 <NETStandard11Dependency Include="@ (NETStandard10Dependency)" />  
29 <NETStandard11Dependency Include="System.Collections.Concurrent" />  
30 <NETStandard11Dependency Include="System.Diagnostics.Tracing" />  
31 <NETStandard11Dependency Include="System.IO.Compression" />  
32 <NETStandard11Dependency Include="System.Net.Http" />  
33 <NETStandard11Dependency Include="System.Runtime.InteropServices" />  
34 <NETStandard11Dependency Include="System.Runtime.InteropServices.RuntimeInformation" />  
35 <NETStandard11Dependency Include="System.Runtime.Numerics" />  
36
```

1.1

```
37 <NETStandard12Dependency Include="@ (NETStandard11Dependency)" />  
38 <NETStandard12Dependency Include="System.Threading.Timer" />
```

1.2

- <https://github.com/dotnet/standard/blob/master/netstandard/pkg/NETStandard.Library.dependencies.props>
- **Fix, nem változó verziók: kiadott API nem változik soha többé**

.NET Standard

Extensions and Updates



- ▶ Installed
- ◀ Online
 - Visual Studio Marketplace
 - Search Results
 - ▶ Controls
 - ▶ Templates
 - ▶ Tools
- ▶ Updates (1)
- ▶ Roaming Extension Manager

Sort by: Relevance



.NET Portability Analyzer

Evaluates portability of assemblies across .NET platforms

[Download](#)

TypeScript Analyzer

Provides static analysis directly in Visual Studio for TypeScript files (.ts and .tsx) using TSLint



CLR Heap Allocation Analyzer

CLR Heap Allocation Analyzer is a Roslyn based Diagnostic Analyzer that is able to detect most allocations in code in the local method...



Serilog Analyzer

Roslyn-based analysis for code using the Serilog logging library. Checks for common mistakes and usage problems.



Inlining Analyzer

Inlining Analyzer shows if a method call will be inlined by the JIT Compiler. Method calls are highlighted right in the source code and...



Performance & Memory Diagnostics - .NET Runtime...

Performance & Memory Timeline Profiler, GDI Resource Tracker, Application / System Event Tracer & Exception Tracker



C# Code Analyzer (Roslyn)

Validations and Quick-fixes for Composite Format String at compile-time.



Audio Waveform Analyzer for .NET

.NET Winform component that adds sound waveform analysis to

1 2 3 ▶

portability analyzer



Created by: Microsoft

Version: 2.5.0.0

Downloads: 131538

Rating: ★★★★★ (15 Votes)

[More Information](#)

[Report Extension to Microsoft](#)

Scheduled For Install:

None

Scheduled For Update:

None

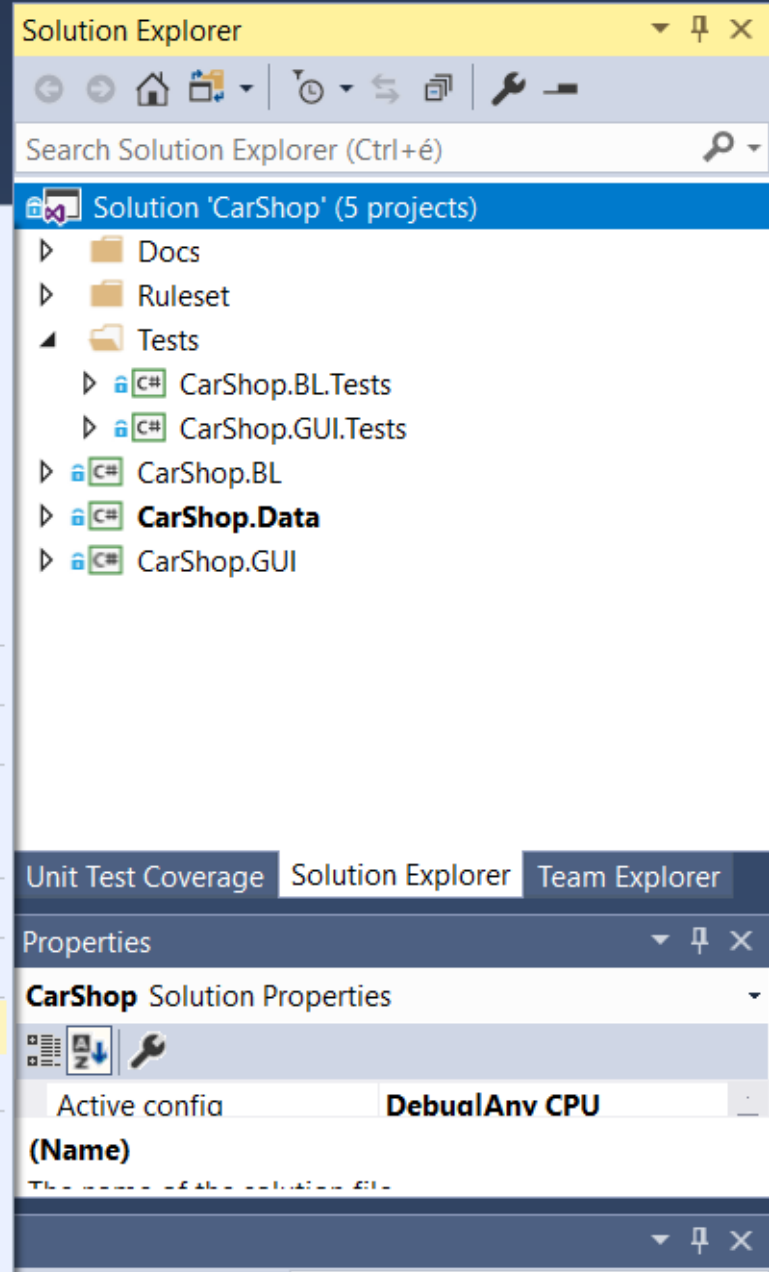
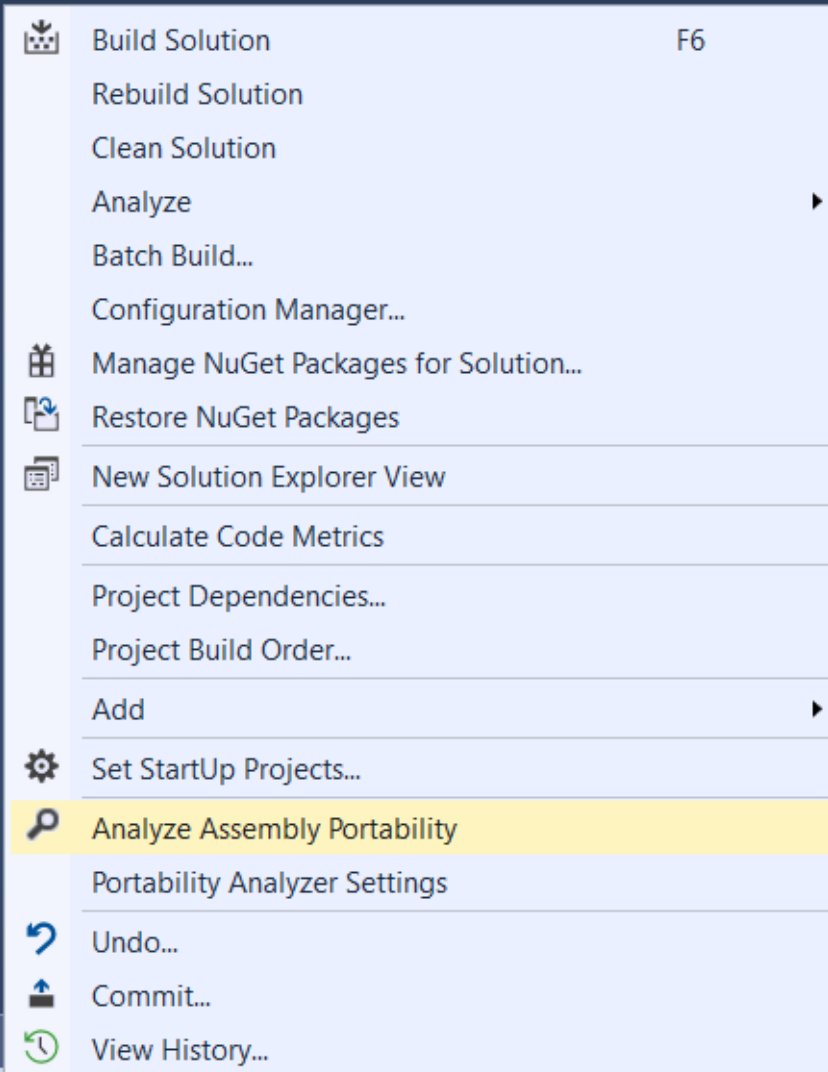
Scheduled For Uninstall:

None

[Change your Extensions and Updates settings](#)

Close

.NET Standard

























.NET Standard

| | \times | \checkmark | f_x |
|--|----------|--------------|-------|
| 1. $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$ | | | |
| 2. $\frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$ | | | |
| 3. $\frac{1}{2} \times \frac{3}{3} = \frac{1}{2}$ | | | |
| 4. $\frac{2}{2} \times \frac{1}{3} = \frac{1}{3}$ | | | |
| 5. $\frac{2}{2} \times \frac{2}{3} = \frac{2}{3}$ | | | |
| 6. $\frac{2}{2} \times \frac{3}{3} = 1$ | | | |
| 7. $\frac{3}{2} \times \frac{1}{3} = \frac{1}{2}$ | | | |
| 8. $\frac{3}{2} \times \frac{2}{3} = 1$ | | | |
| 9. $\frac{3}{2} \times \frac{3}{3} = \frac{3}{2}$ | | | |
| 10. $\frac{3}{3} \times \frac{1}{3} = \frac{1}{3}$ | | | |
| 11. $\frac{3}{3} \times \frac{2}{3} = \frac{2}{3}$ | | | |
| 12. $\frac{3}{3} \times \frac{3}{3} = 1$ | | | |

| | A | B | C | D | E | F | G | | |
|----|---|--|----------------|---------------|----------------|----------------|----------------|--|--|
| 1 | Submission Id | f73533d6-e1ef-43d7-903a-5a80b756d968 | | | | | | | |
| 2 | Description | | | | | | | | |
| 3 | Targets | .NET Framework,.NET Standard + Platform Extensions,.NET Standard,Version=v1.1,.NET Standard,Version=v1.6,.NET Standard,Version=v | | | | | | | |
| 4 | | | | | | | | | |
| 5 | Header for assembly name entries | Target Framework | .NET Framework | .NET Standard | .NET Standard, | .NET Standard, | .NET Standard, | | |
| 6 | CarShop.BL | .NETFramework,Version=v4.6.1 | 100 | 100 | 98.1 | 98.1 | 100 | | |
| 7 | CarShop.BL.Tests | .NETFramework,Version=v4.6.1 | 100 | 100 | 98.26 | 98.84 | 100 | | |
| 8 | CarShop.Data | .NETFramework,Version=v4.6.1 | 100 | 100 | 94.85 | 94.85 | 100 | | |
| 9 | CarShop.GUI | .NETFramework,Version=v4.6.1 | 100 | 81.02 | 72.22 | 73.15 | 79.17 | | |
| 10 | CarShop.GUI.Tests | .NETFramework,Version=v4.6.1 | 100 | 100 | 98.4 | 98.8 | 100 | | |
| 11 | | | | | | | | | |
| 12 | API Catalog last updated on | Friday, December 7, 2018 | | | | | | | |
| 13 | See 'http://go.microsoft.com/fwlink/?LinkId=397652' to learn how to read this table | | | | | | | | |
| 14 | | | | | | | | | |
| 15 | | | | | | | | | |
| 16 | | | | | | | | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |
| 19 | | | | | | | | | |
| 20 | | | | | | | | | |
| 21 | | | | | | | | | |
| 22 | | | | | | | | | |
| 23 | | | | | | | | | |
| 24 | | | | | | | | | |

.NET Standard

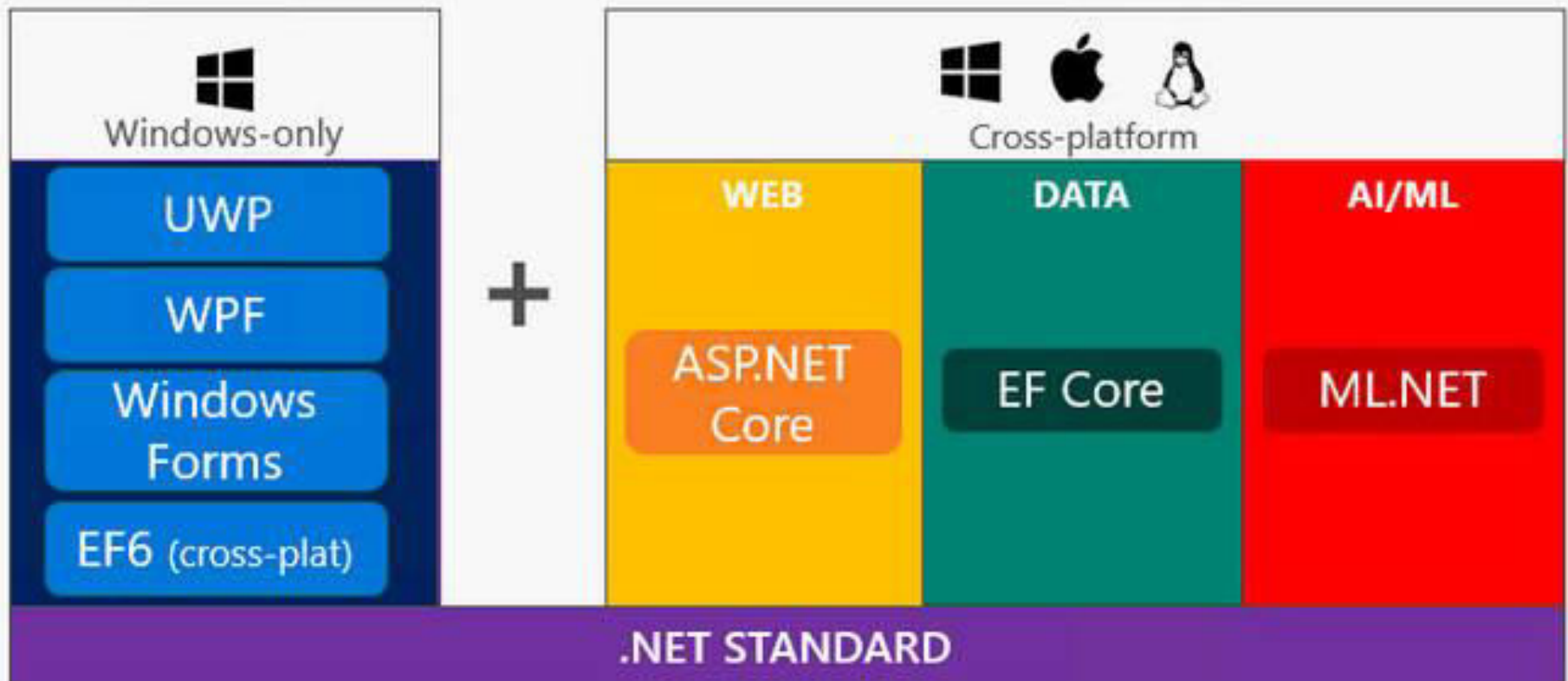
| Automatikus mentés    | | | | | | | | |
|--|---|---|-----------------|--|-----------------|---|-----------------|---|
| ApiPortAnalysis(2).xlsx - Excel | | | | | | | | |
| Fájl | Kezdőlap | Beszúrás | Lapelrendezés | Képletek | Adatok | Véleményezés | Nézet | Bővítmények |
| Súgó | Team | Mutasd meg, hogyan c | | | | | | |
| Beillesztés  | | Vágólap  | | Betűtípus  | | Igazitás  | | Stílusok  |
| Calibri 11 A^ A^ | | F D A    | | Sortöréssel több sorba  | | Általános  | | Feltételes formázás  |
| Vágólap  | | Betűtípus  | | Igazitás  | | Stílusok  | | Formázás táblázatként  |
| J16    | | | | | | | | |
| | B | C | D | E | F | G | H | |
| 1 | Target member | Header for | .NET Framework | .NET Standard | .NET Standard | .NET Standard | .NET Standard | Recommen |
| 2 | T:System.Diagnostics.ProcessStartInfo | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 3 | M:System.Diagnostics.ProcessStartInfo.set_FileN | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 4 | M:System.Diagnostics.ProcessStartInfo.set_UseS | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | Remove Sh |
| 5 | T:System.Windows.MessageBox | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 6 | M:System.Windows.MessageBox.Show(System.S | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 7 | T:System.Windows.Window | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 8 | M:System.Windows.Window.#ctor | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 9 | M:System.Windows.Window.add_Closed(System | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 10 | M:System.Windows.Window.set_DialogResult(Sy | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 11 | M:System.Windows.Window.set_Owner(System | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 12 | M:System.Windows.Window.ShowDialog | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 13 | T:System.Windows.FrameworkElement | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 14 | M:System.Windows.FrameworkElement.add_Lo | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 15 | M:System.Windows.FrameworkElement.get_Dat | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 16 | M:System.Windows.FrameworkElement.set_Dat | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 17 | T:System.Diagnostics.Process | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 18 | M:System.Diagnostics.Process.#ctor | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 19 | M:System.Diagnostics.Process.get_StartInfo | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 20 | M:System.Diagnostics.Process.Start | CarShop.GUI | Supported: 1.0+ | Supported: 1.6+ | Supported: 2.0+ | Supported: 2.0+ | Supported: 2.0+ | |
| 21 | T:System.Windows.Application | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 22 | M:System.Windows.Application.#ctor | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |
| 23 | M:System.Windows.Application.LoadComponent | CarShop.GUI | Supported: 3.0+ | Not supported | Not supported | Not supported | Not supported | |

.NET Core 3...



Desktop Packs

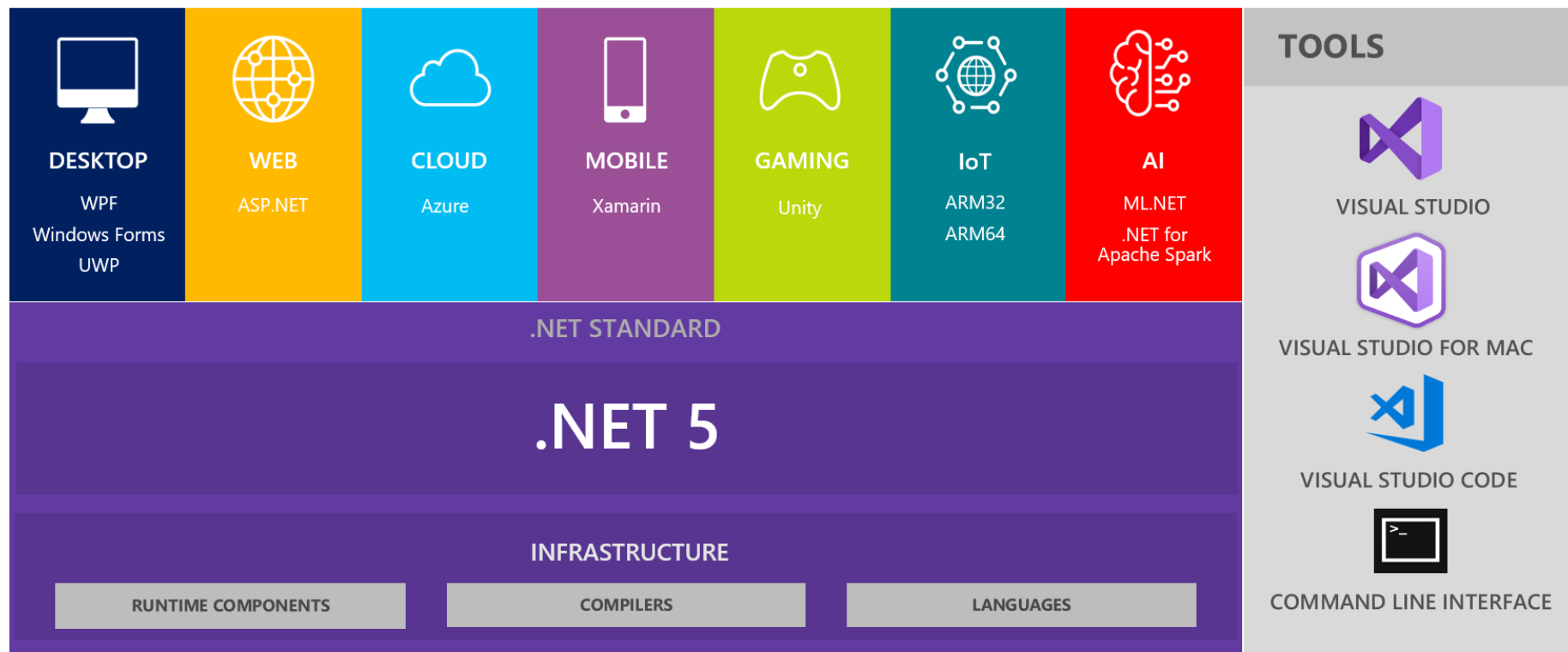
.NET Core 3



... és azon túl ...

- 2019. November: .NET Core 3.1 (bugfix, LTS)
- 2020. November: .NET FRAMEWORK 5
- 2021. November: .NET 6, LTS

.NET – A unified platform



.NET 5 és C# 9

- **v5.0.0-preview.8 (2020.08.25 = „feature complete”)**
v5.0.0-rc.1 (2020.09.14)
- **<https://devblogs.microsoft.com/dotnet/announcing-net-5-0-rc-1/>**
 - Init property accessors, Pattern matching (switch), Records, System.Text.Json
 - GetGCMemoryInfo() method to get detailed GC info
 - Better RyuJIT
- **<https://docs.microsoft.com/en-us/ef/core/what-is-new/ef-core-5.0/whatsnew>**
 - Many-to-many, Better Migrations, Better Mapper, Better inheritance, Indexes, Easy Logging, [KeyLess],
- **<https://github.com/dotnet/runtime/milestones>**
 - „99% complete, 15 open, 5,761 closed”
 - „Due by November 30, 2020”
 - <https://www.dotnetconf.net/> ➔ **November 10-12**

Források

- <https://www.red-gate.com/simple-talk/blogs/anatomy-of-a-net-assembly-pe-headers/>
- <https://blog.takipi.com/clr-vs-jvm-how-the-battle-between-net-and-java-extends-to-the-vm-level/>
- https://en.wikipedia.org/wiki/Virtual_machine
- Chris Farrell, Nick Harrison: Under the hood of .NET Memory Management, ISBN: 978-1-906434-74-8
- <http://www.informit.com/articles/article.aspx?p=1409801>
- <https://msdn.microsoft.com/en-us/library/ms973837.aspx>
- [https://msdn.microsoft.com/en-us/library/system.object.finalize\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/system.object.finalize(v=vs.71).aspx)
- <https://docs.oracle.com/javase/9/gctuning/introduction-garbage-collection-tuning.htm#JSGCT-GUID-A48F272E-A6C1-45A0-9A8B-6D5790EB454C>