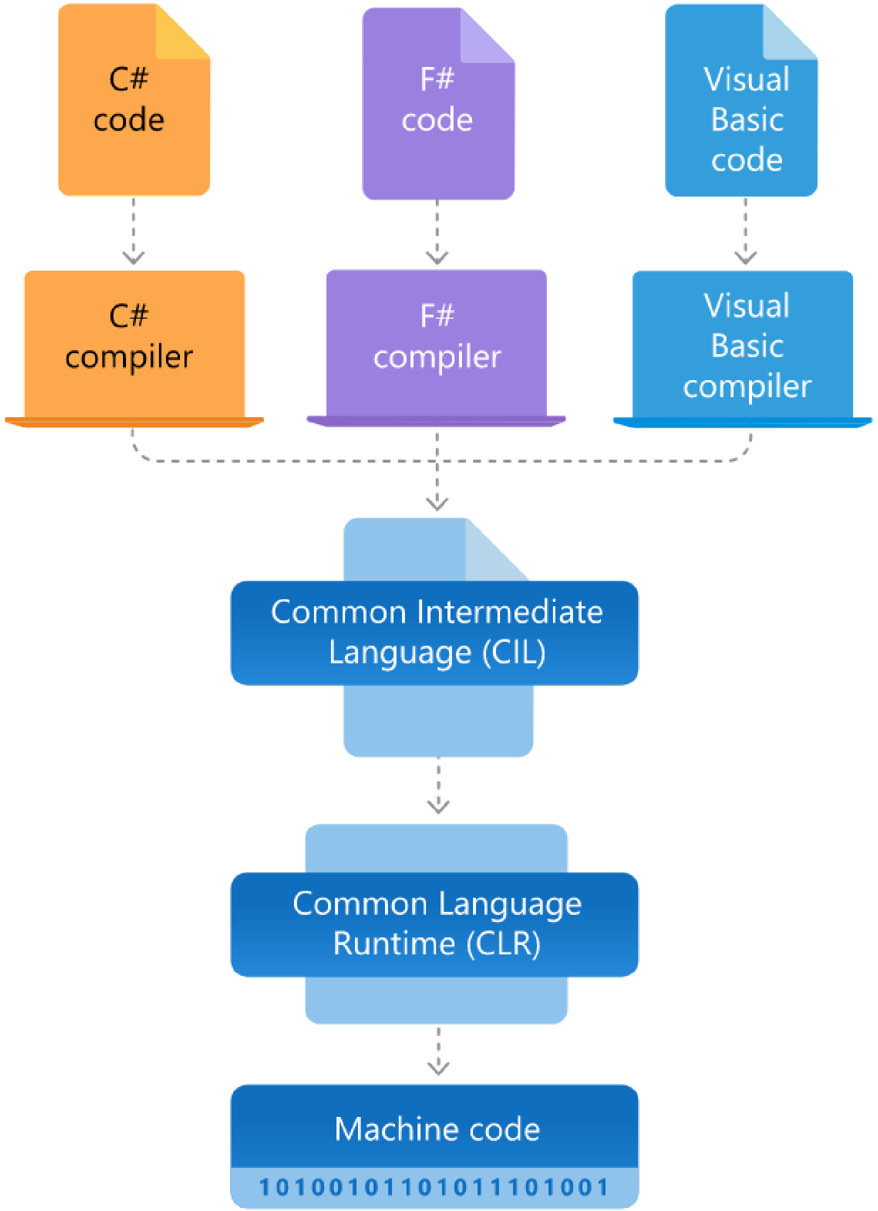
Day 1 : .NET & C# Basics

.NET Framework

- A runtime execution environment to manage apps targeting the framework
- Original implementation of .NET
- Supports running websites, services, desktop apps on Windows platforms only
- Includes two major components
  - Common Language Runtime(CLR)
  - Base Class Library

- Common Type System
- Common Language Specification
- Current version – 4.8

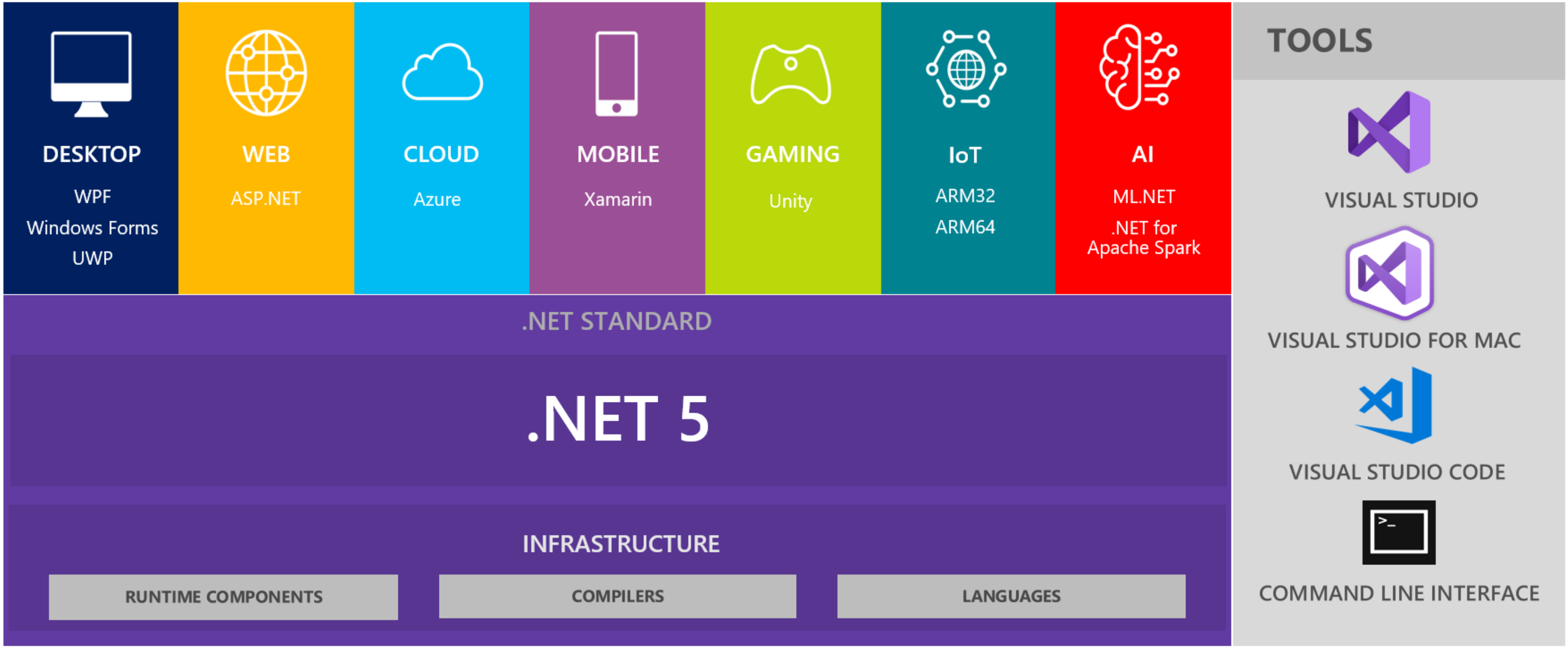
.NET Framework



.NET 5.0

- Is a free, open-source development platform
- Cross platform
  - Operating Systems: Windows, macOS, Linux, Android, iOS, tvOS, watchOS
  - Processor Architecture: x64, x86, ARM32, ARM64
- Can be used to build Web Apps, Web APIs, Cloud native apps, Mobile Apps, Desktop Apps, Games, IoT
- Supports C#, F#, Visual Basic
- Supports AOT(Ahead of time) compilation
- Automatic memory management

.NET – A unified platform



DESKTOP

WPF

Windows Forms

UWP

WEB

ASP.NET

CLOUD

Azure

MOBILE

Xamarin

GAMING

Unity

IoT

ARM32

ARM64

AI

ML.NET

.NET for Apache Spark

.NET STANDARD

.NET 5

INFRASTRUCTURE

RUNTIME COMPONENTS

COMPILERS

LANGUAGES

TOOLS

VISUAL STUDIO

VISUAL STUDIO FOR MAC

VISUAL STUDIO CODE

COMMAND LINE INTERFACE

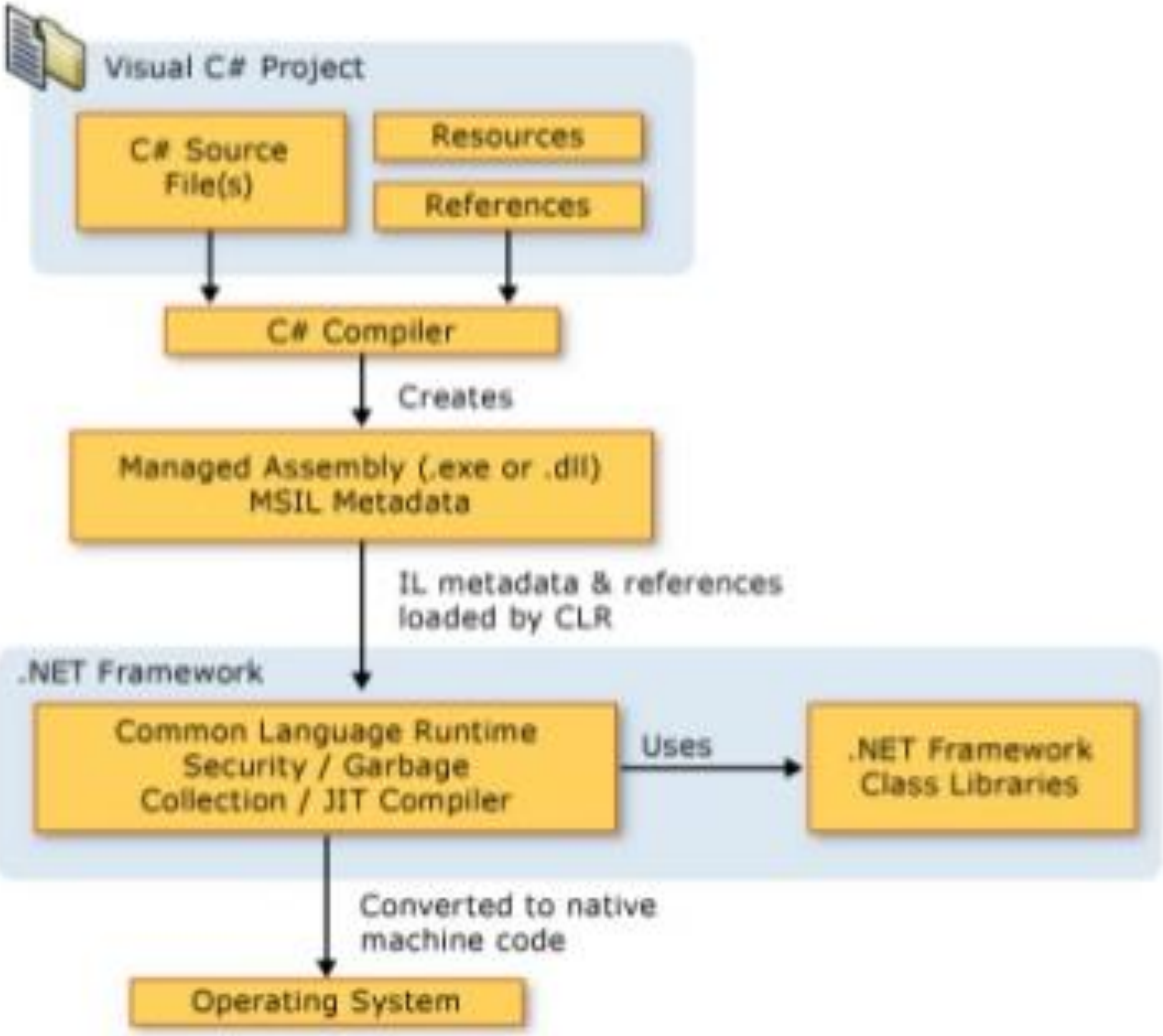
C#

- Developed by Microsoft as part of .NET Framework initiative
- Approved as a standard by ECMA -ECMA-334
- Designed by Anders Hejlsberg and is now lead by Mads Torgersen
- Major Versions

Version	Year	Framwork	IDE
1.0	2002	.NET 1.0	Visual Studio.NET 2002
2.0	2005	.NET 2.0	Visual Studio 2005, 2008
3.0	2008	.NET 3.0	Visual Studio 2008
4.0	2010	.NET 4.0	Visual Studio 2010
5.0	2012	.NET 4.5	Visual Studio 2012, 2013
6.0	2015	.NET 4.6, .NET Core 1.0	.NET Core 1.0, Visual Studio 2015
7.0	2017	.NET 4.7, .NET Core 2.0	.NET Core 2.0, Visual Studio 2017 v15.0
8.0	2019	.NET 4.8, .NET Core 3.0	.NET Core 3.0, Visual Studio 2017 v16.3

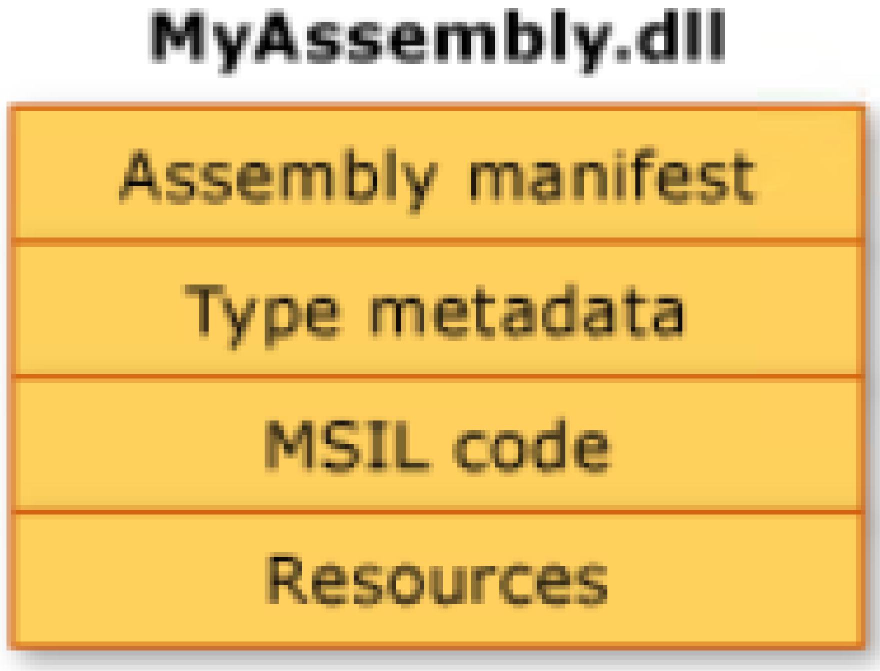
https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

Execution Cycle


Visual C# Project
C# Source File(s)
Resources
References
C# Compiler
Creates
Managed Assembly (.exe or .dll) MSIL Metadata
IL metadata & references loaded by CLR
.NET Framework
Common Language Runtime Security / Garbage Collection / JIT Compiler
Uses
.NET Framework Class Libraries
Converted to native machine code
Operating System

Assembly

- Fundamental unit of deployment
- Collection of types and resources
- Assemblies take the form of a .DLL or .EXE
- Assemblies that are part of .NET framework is put in Global Assembly Cache(GAC)



MyAssembly.dll

Assembly manifest

Type metadata

MSIL code

Resources

Structure

```
using System;
namespace SampleNamespace
{
    class SampleClass
    {
        static void Main(string[] args)
        {
            //Your program here...
        }
    }
}
```

Type System

Value Types

- Stores data directly
- Cannot be null

Reference Types

- Stores references to objects
- Can be null

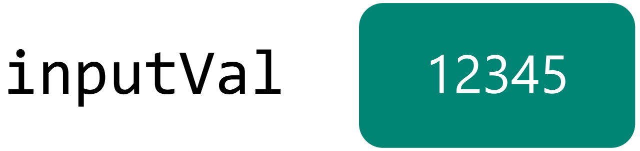
Dynamic Types

- Stores any type of value
- Type checking take place at runtime

Type System

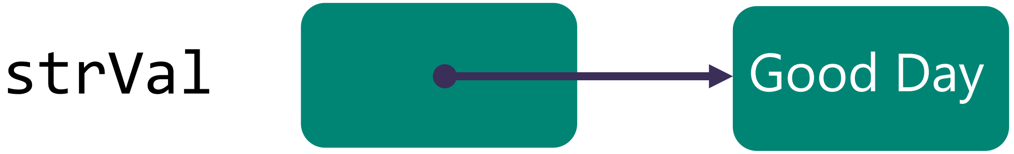
Value Type

```
int inputVal = 12345;
```



Reference Type

```
string strVal = “Good Day”;
```



Value Types

· Primitives	· `int i;`
· Enums	· `enum Selected { “off”, “on” }`
· Structs	· `struct Point { int x, int y; }`

Reference Types

· Classes	· `class Foo : Ifoo { … }`
· Interfaces	· `interface Ifoo : Ibar { … }`
· Arrays	· `string[] arr = new string[10];`
· Delegates	· `delegte void OnClicked()`

Predefined Types

Reference	object, string
Signed	sbyte, short, int, long
Unsigned	byte, ushort, uint, ulong
Character	char
Floating Point	float, double, decimal
Logical	bool

Comments

```
//Single line comment

/*
Multi line
…
…
Comments
*/

/// <summary>
/// Documentation single line comment.
/// </summary>
public string FirstName { get; set; }

/**
* <summary> Documentation Multi line comment</summary>
*/
public string Last Name { get; set; }
```

Statements

- A single line of code that ends with semi colon(;)
- Series of single line of statements in a block
- A statement block is enclosed in {} brackets
- Can contain nested blocks

```
//Declaration
int age;

//Assignment
age = 25;
```

Variables

Syntax

```
<access specifier> <data type> <name>;
```

Example

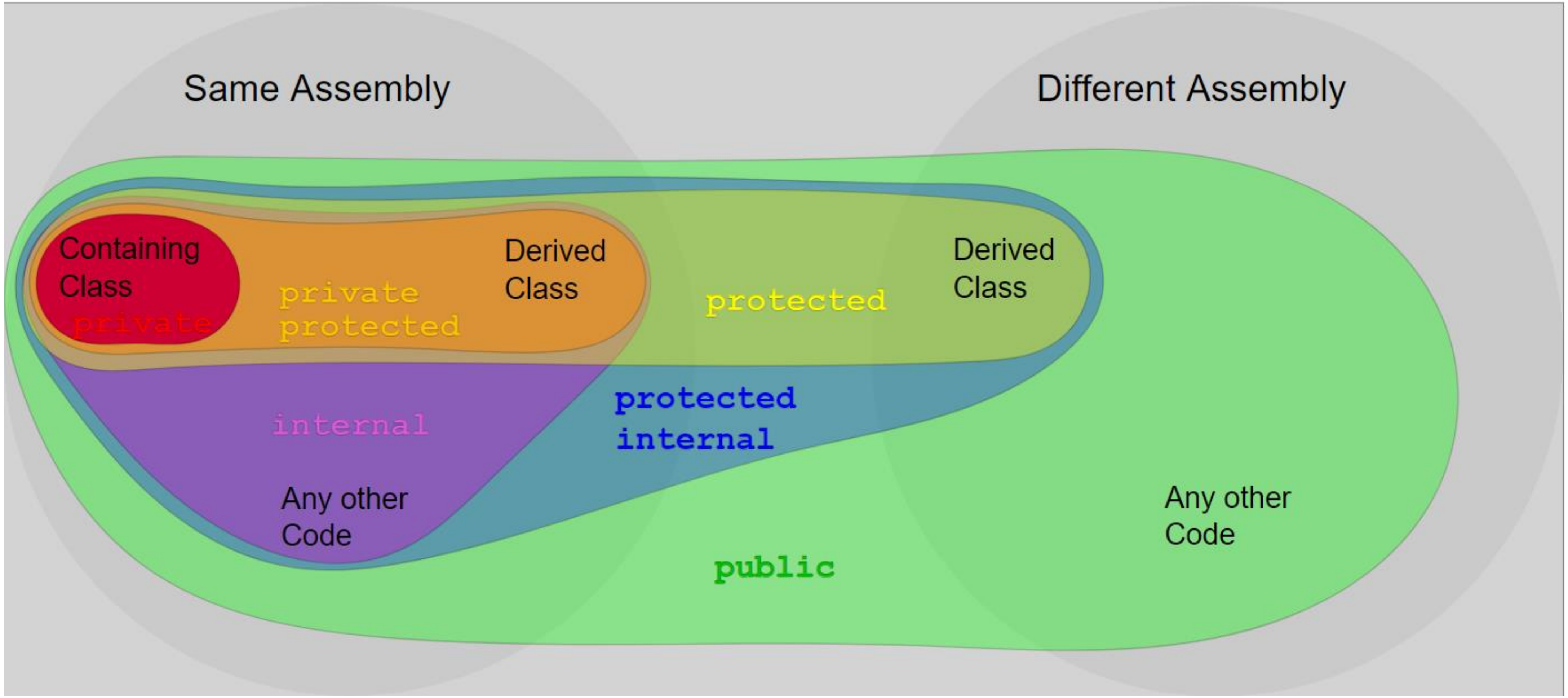
```
private int age; //declaration

private int age = 10; //declaration and assignment
```

Access Specifiers

public	can be accessed by any other code in the same assembly or another assembly that references it
private	can be accessed only by code in the same class or struct
protected	can be accessed only by code in the same class, or in a class that is derived from that class.
internal	can be accessed by any code in the same assembly, but not from another assembly.
protected internal	can be accessed by any code in the assembly in which it's declared, or from within a derived class in another assembly
private protected	can be accessed only within its declaring assembly, by code in the same class or in a type that is derived from that class.

Access Specifiers


Same Assembly
Different Assembly
Containing Class
private
private protected
Derived Class
protected
Derived Class
protected internal
internal
Any other Code
public
Any other Code