**Einführung in .NET und Visual Studio**

Am Anfang dieses Jahrhunderts machte sich der Softwarehersteller Microsoft intensive Gedanken über eine mögliche Konkurrenz zu der damals vorherrschenden Java-Plattform. Einerseits sollten die neuen Sprachen Visual Basic 6 – Entwickler in eine objektorientierte Welt bringen, andererseits wollte man bestehende C++ und Java-Programmierer auch für sich gewinnen.  
Beides scheint gelungen zu sein, denn das am **13. Februar 2002** veröffentlichte **.NET-Framework** (= .NET-Plattform)unterstütztvom ersten Moment an die beiden Welten angelehnte Sprachen: **VB.NET** als Ersatz für veraltetes Visual Basic 6 (VB6) und **C#** als Ersatz für C++ und Java.  
Es dauerte nicht lange, so sprossen auf der .NET-Plattform, wie Pilze aus dem Boden neue Programmiersprachen, wie z.B. **J#** (noch stärker an Java angelehnt). Dank eines gemeinsamen Standards ist .NET **offen** für weitere Programmiersprachen.  
Praktisch jeder kann „seine“ .NET-Sprache entwickeln und auch veröffentlichen – solange er sich klarerweise an einer **gemeinsamen Sprachspezifikation** hält.

**Was ist .NET?**

Die Programmiersprache C# „läuft“ auf der sogenannten .**NET-Plattform** oder „unter .NET.“ und ist somit eine **.NET-Sprache.**

.NET ist eine Open-Source-Plattform zur Entwicklung von Software, die für die Entwicklung von **verschiedenen Programmiersprachen** (wie C#, F#, VB.NET, …) genutzt werden kann.  
Es bietet eine **Laufzeitumgebung**, mit der Programme auf unterschiedlichen Plattformen (Windows, Linux, macOS, …) ausgeführt werden können.

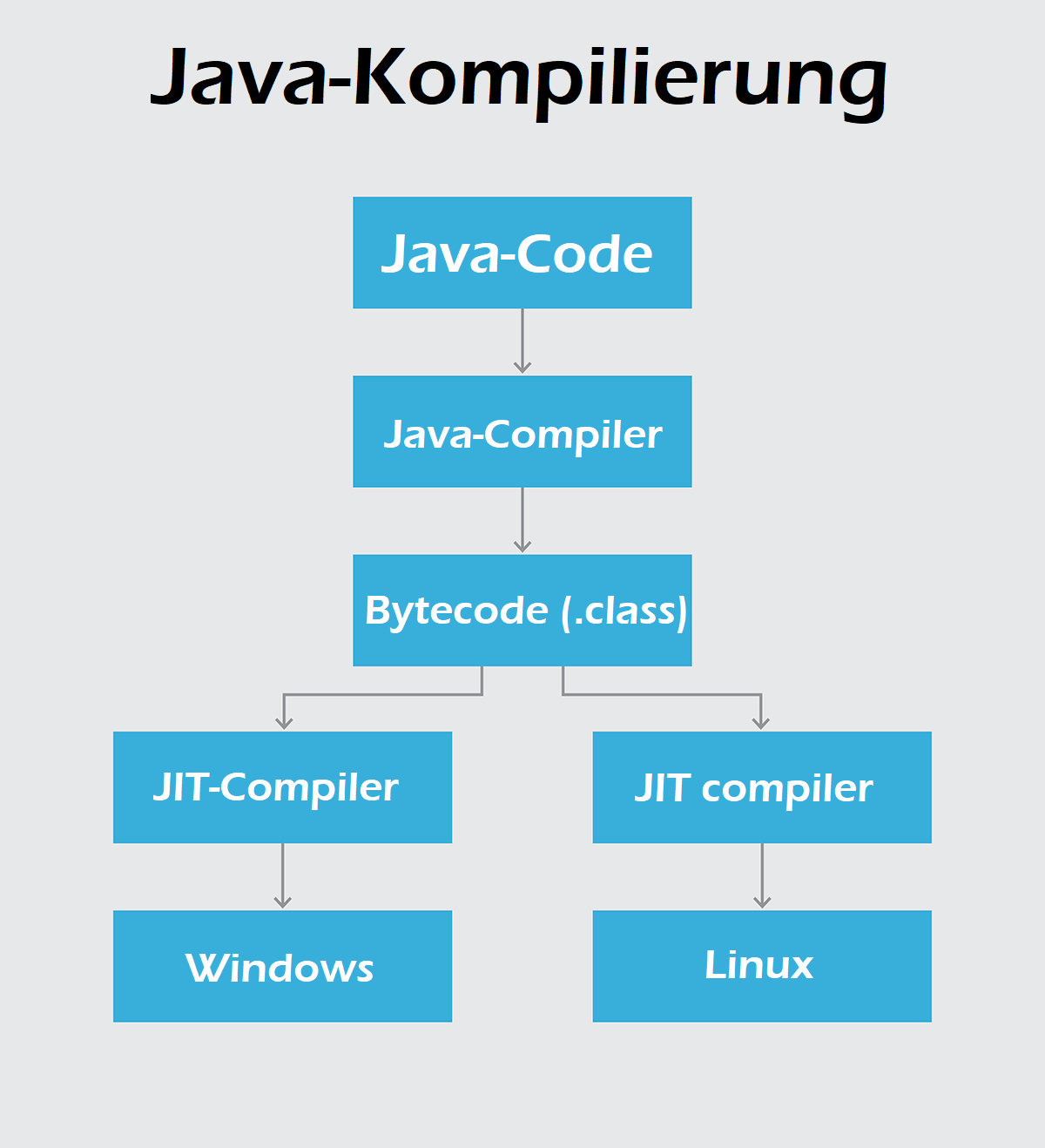
* **.NET Framework vs. .NET Core vs. .NET 5 und höher**
  + **.NET Framework:**
    - Enthält eine große Menge an Bibliotheken und eine **Laufzeitumgebung** (**CLR – Common Language Runtime**).
    - Wird zunehmend durch **.NET Core** und **.NET 5+** ersetzt, insbesondere da .NET Core plattformübergreifend ist.
  + **.NET Core:**
    - Cross-Platform (Windows, macOS, Linux).
    - Open-Source und modularer als .NET Framework.
    - Sehr schnell und optimiert für Cloud-Entwicklung.
    - Modularität: Im Gegensatz zum .NET Framework, das eine große, monolithische Codebasis hatte, ermöglichte .NET Core eine modulare Architektur. Entwickler können nur jene Teile des Frameworks verwenden, die sie tatsächlich benötigten, was zu einer besseren Performance und kleineren Anwendungen führt.
  + **.NET 5 und höher:**
    - Ab .NET 5 wurde **.NET Core mit .NET Framework zusammengeführt**.
    - Eine vereinheitlichte Plattform, die sowohl Desktop- als auch Web-Anwendungen unterstützt.

**.NET-Sprachen: C#, F#, VB.NET**

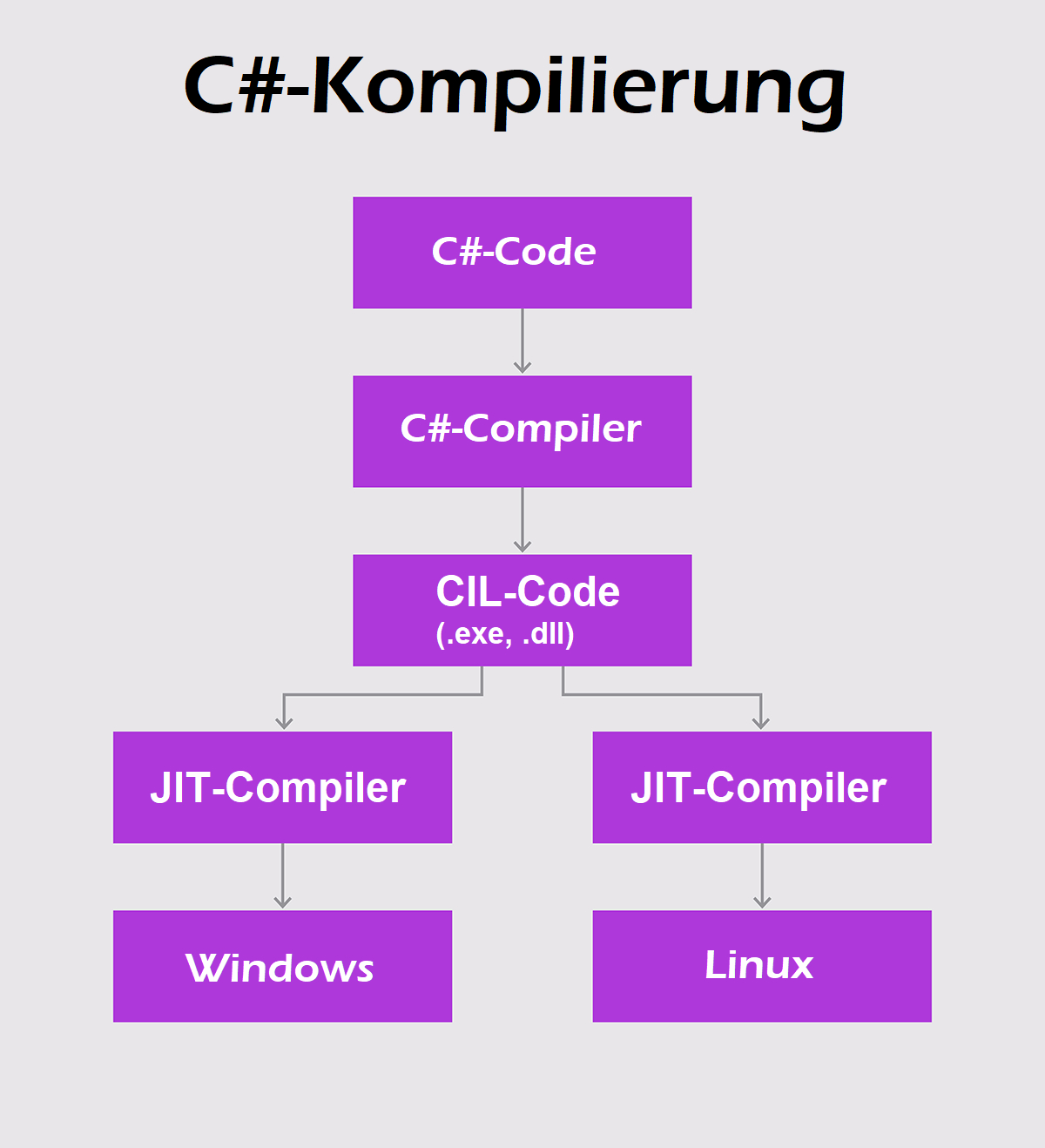
* **C#:**
  + Eine objektorientierte Programmiersprache, die derzeit in .NET am häufigsten verwendet wird.
  + Wird in den meisten modernen .NET-Anwendungen verwendet, z. B. in **Desktop-**Apps, **Web-**Applikationen und **mobilen** Apps.
* **F#:**
  + Eine **funktionale** Programmiersprache, die vor allem in wissenschaftlichen und mathematisch orientierten Anwendungen verwendet wird.
  + Gute Unterstützung für funktionale und objektorientierte Programmierung.
* **VB.NET:**
  + War die erste voll entwickelte Programmiersprache, die sich an das noch ältere VB6 (das noch keine .NET-Sprache war) orientierte. VB.NET ist ähnlich wie C#, aber mit einer anderen Syntax.

**Kompilierung**

**Java kompilieren**



**.NET kompilieren**

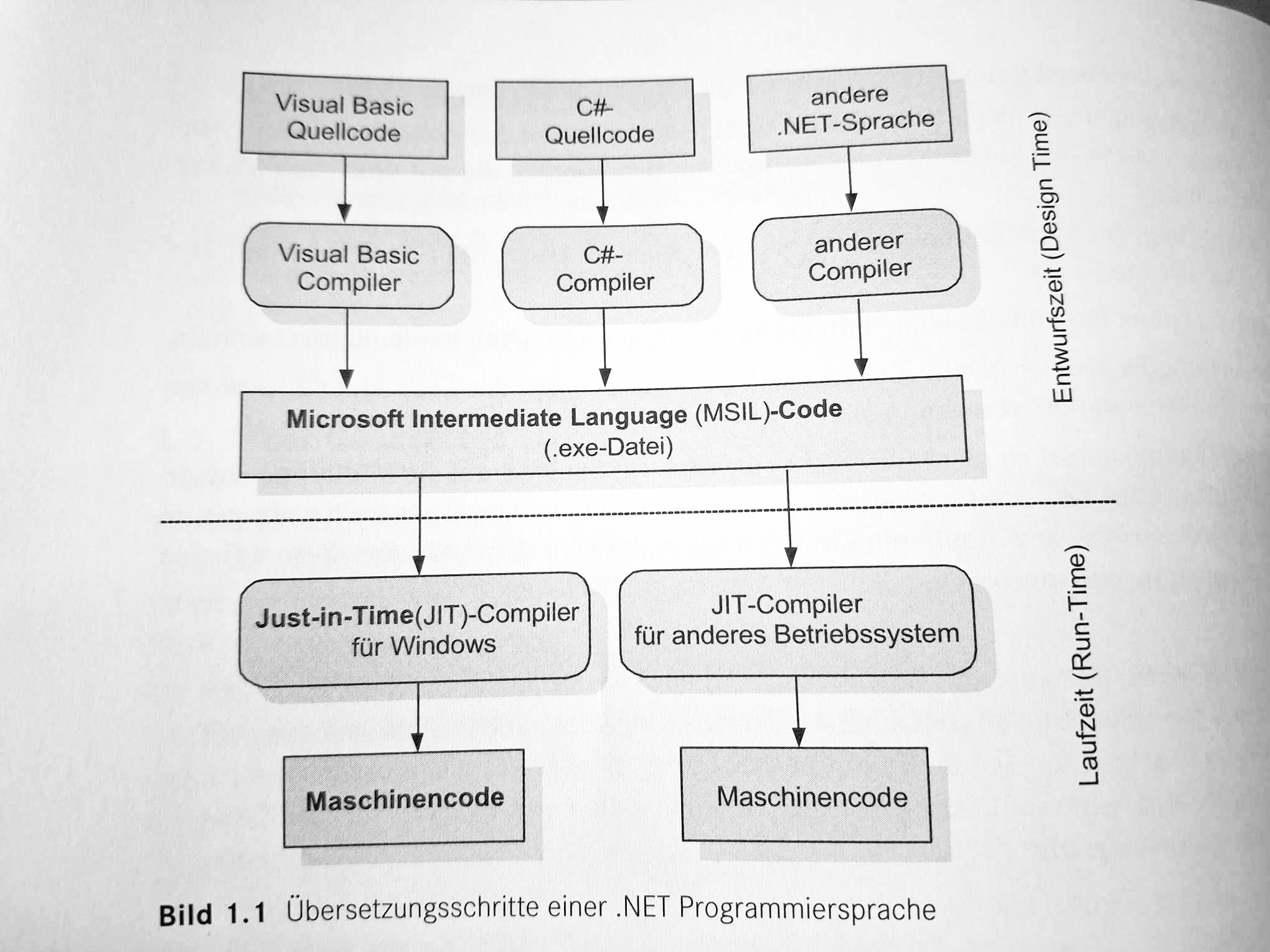


Beim Kompilieren eines .NET-Programmes wird der jeweilige **Quelltext** in **Common Intermediate Language** - Gemeinsame Zwischensprache **(**zwischen)**kompiliert**.

Es gibt nur ein **gemeinsames Laufzeitsystem für alle .NET-Sprachen**, die sogenannte **CLR** (Common Language Runtime – Laufzeitumgebung der Gemeinsamen Sprache). **CLR führt CIL– Programme** aus**.** Ein anderer (älterer) Name für CIL lautet **MSIL**.

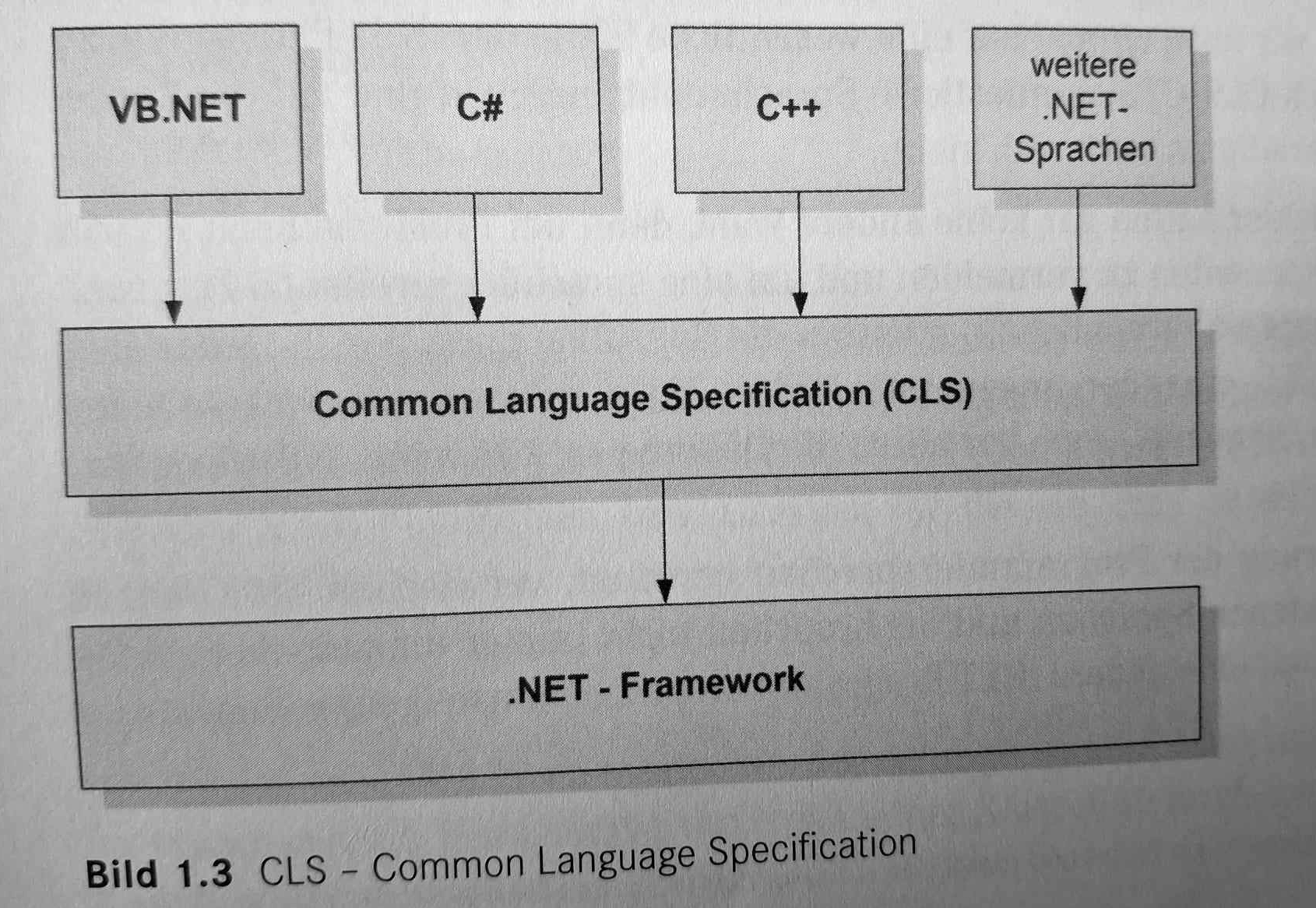
Erst wenn der **CIL**-Code von einem Programm zur Ausführung genutzt werden soll (z.B. eine .exe doppelklickt), wird er von **Just-in-Time Compiler (JIT)** in den jeweiligen (Windows, Linux, MacOS, …) **Maschinencode** übersetzt.

Ein Programm wird also von Entwurf bis zu seiner Ausführung tatsächlich **zweimal kompiliert.**



**Common Language Specification (Gemeinsame Sprachspezifikation)**

Natürlich gibt es **allgemeine Richtlinien und Standards,** an denen man sich halten muss um eine **gültige .NET-Sprache** zu entwickeln.



**.NET Core: Vertiefung**

Wichtige Merkmale von .NET Core:

1. **Plattformübergreifend**: .NET Core kann auf verschiedenen Betriebssystemen ausgeführt werden, was es zu einer viel flexibleren Lösung machte als das traditionelle .NET Framework, das hauptsächlich auf Windows beschränkt war.
2. **Modularität**: Im Gegensatz zum .NET Framework, das eine große, monolithische Codebasis hatte, ermöglicht **.NET Core** eine modulare Architektur. Entwickler können nur jene Teile des Frameworks verwenden, die sie **tatsächlich benötigten**, was zu einer besseren **Performance** und kleineren Anwendungen führt.
3. **Leistung**: .NET Core wurde mit Fokus auf **hohe Performance** entwickelt. Es ist schneller und effizienter als das .NET Framework, besonders bei der Web- und Cloud-Anwendungsentwicklung.
4. **Open Source**: .NET Core war unter der MIT-Lizenz als Open-Source-Software verfügbar, was es der Entwicklergemeinschaft ermöglichte, zur Weiterentwicklung beizutragen.

**Gemeinsamkeiten und Unterschiede Java vs. C#**

**1. Allgemein**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Erscheinungsjahr | 1995 | 2000 |
| Entwickler | Sun Microsystems (jetzt Oracle) | Microsoft |
| Laufzeitumgebung | Java Virtual Machine (JVM) | Common Language Runtime (CLR, .NET) |
| Plattform | Plattformunabhängig (Write Once, Run Anywhere) | Plattformübergreifend (ab .NET Core/.NET 5+) |

**2. Syntax und Sprachfeatures**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Datentypen | Primitive Typen (z. B. int, double) und Objekttypen | Einheitliches Typsystem (z. B. int ist ein Alias für System.Int32) |
| Eigenschaften | Keine direkten Eigenschaften (nur Getter/Setter-Methoden) | Unterstützt Eigenschaften (get/set-Syntax) |
| Null-Handling | Seit Java 8: Optional (limitierter Ansatz) | Nullable Reference Types seit C# 8.0 |
| Lambda-Ausdrücke | Seit Java 8 verfügbar | Seit C# 3.0, weiterentwickelter und flexibler |
| Generics | Typerasure (Generics sind nur zur Kompilierungszeit vorhanden) | Generics werden zur Laufzeit beibehalten (stärker typisiert) |
| Schlüsselwort var | Nicht verfügbar | Unterstützt implizite Typen mit var |

**3. Objektorientierung**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Abstrakte Klassen und Interfaces | Beides verfügbar | Beides verfügbar + Default-Implementierungen in Interfaces (seit C# 8.0) |
| Multiple Vererbung | Nicht erlaubt, aber durch Interfaces simuliert | Nicht erlaubt, aber durch Interfaces simuliert |
| Delegates | Nicht verfügbar (nur über funktionale Interfaces) | Unterstützt Delegates (starke Funktionalität für Events) |

**4. Laufzeit und Performance**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| JIT-Kompilierung | Bytecode wird von der JVM in Maschinencode übersetzt | Ähnlich, wird von der CLR in Maschinencode übersetzt |
| AOT-Kompilierung | Nicht direkt (außer in bestimmten Java-Subprojekten) | Unterstützt (z. B. mit .NET Native) |
| Garbage Collection | Automatisches Speichermanagement | Automatisches Speichermanagement, mit optimierten GC-Modi |
| Native Performance | Abhängig von der JVM und Hardware | Durch .NET Core/.NET 5+ oft performanter |

**5. Ökosystem und Tools**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Entwicklungsumgebung | Eclipse, IntelliJ IDEA, NetBeans | Visual Studio, JetBrains Rider, Visual Studio Code |
| Build-Systeme | Maven, Gradle, Ant | MSBuild, NuGet, Cake |
| Frameworks | Spring, Hibernate, Jakarta EE | ASP.NET Core, Entity Framework, Blazor |
| Bibliotheken/Packages | Maven Central Repository | NuGet |

**6. Plattformunabhängigkeit**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Plattform | JVM ermöglicht plattformübergreifende Ausführung | .NET Core/.NET 5+ ermöglicht plattformübergreifende Entwicklung |
| Mobile Entwicklung | Android-Entwicklung über native Tools | Xamarin/.NET MAUI für plattformübergreifende Apps |

**7. Community und Einsatzgebiete**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Verbreitung | Stark in Unternehmenssoftware, Android-Apps | Stärker in Microsoft-Ökosystemen, aber auch vielseitig einsetzbar |
| Community | Große Open-Source-Community | Große Entwicklergemeinschaft mit starkem Microsoft-Support |
| Anwendungsbereiche | Web (Spring), Mobile (Android), Enterprise | Web (ASP.NET Core), Mobile (Xamarin/.NET MAUI), Spiele (Unity) |

**Kompilierung Vergleich von Java und C#**

|  |  |  |
| --- | --- | --- |
| Sprache | Java | C# |
| Compiler | javac | csc |
| Zwischenformat | Bytecode (.class) | Common Intermediate Language (CIL) |
| Virtuelle Maschine | Java Virtual Machine (JVM) | Common Language Runtime (CLR) |
| Laufzeitkompilierung | Interpreter oder JIT | JIT |
| Plattformunabhängigkeit | Bytecode auf JVMs für viele Plattformen nutzbar | IL auf .NET Core/Runtime für verschiedene Plattformen |

**Zusammenfassung**

* **Java** ist ideal für plattformübergreifende Anwendungen und bietet eine langjährige Stabilität in der Enterprise-Entwicklung. Es wird auch stark in der Android-Entwicklung genutzt.
* **C#** ist moderner und flexibler, insbesondere seit der Einführung von .NET Core/.NET 5+, das die Plattformunabhängigkeit deutlich verbessert hat. Es wird bevorzugt in Microsoft-Umgebungen, bei der Webentwicklung (ASP.NET Core) und in der Spieleentwicklung (Unity) eingesetzt.

Beide Sprachen sind leistungsstark und hängen stark von den Projektanforderungen und dem Ökosystem ab.

**Das erste C#-Programm**

Ganz **ohne Zusatztools** und nur mit Bordmitteln des Windows-Betriebssystems:

* Texteditor: **Notepad.exe**
* C#-Compiler **csc.exe**

1. Schreiben Sie im **Texteditor** folgenden Code:

*class Program*

*{*

*static void Main(string[] args)*

*{*

*System.Console.WriteLine("Hallo Welt");*

*System.Console.ReadLine();*

*}*

*}*

1. Speichern Sie diese Datei an beliebigem Ort als **HalloWelt.cs** ab
2. Rufen Sie die Eingabeanforderung **cmd.exe** auf und kompilieren Sie die **HalloWelt.cs** in **HalloWelt.exe**, indem Sie in cmd.exe folgende Zeile ausführen:

C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe /out:HalloWelt.exe HalloWelt.cs

Roter Teil: C#-Compiler

Grüner Teil: Pfad, wo HalloWelt.exe zu erzeugen ist

Blauer Tei: Pfad zu HalloWelt.cs

1. Doppelklicken Sie die soeben erzeugte HalloWelt.exe. Herzlichen Glückwunsch, Ihr erstes C# -Programm läuft!

**Kurze Einführung in Visual Studio Community (auf Deutsch)**

**Visual Studio Community** ist die kostenlose Edition von Visual Studio und richtet sich an Einzelentwickler, Studenten, Open-Source-Projekte und kleine Teams. Es bietet eine vollständige integrierte Entwicklungsumgebung (IDE) mit leistungsstarken Tools für die Softwareentwicklung.

**Hauptfunktionen von Visual Studio Community**

1. **Sprachunterstützung**:
   * Entwickle in C#, F#, VB.NET, C++, Python, JavaScript und mehr.
2. **Projekttypen**:
   * Erstelle Desktop-Anwendungen, Webanwendungen, mobile Apps, Spiele (Unity), Konsolenprogramme und mehr.
3. **IntelliSense**:
   * Automatische Codevervollständigung und intelligente Vorschläge für schnelles und fehlerfreies Programmieren.
4. **Debugger**:
   * Setze Breakpoints, führe Code Schritt für Schritt aus und überwache Variablen.
5. **Versionskontrolle**:
   * Integrierte Git-Unterstützung für Quellcodeverwaltung (inkl. GitHub).

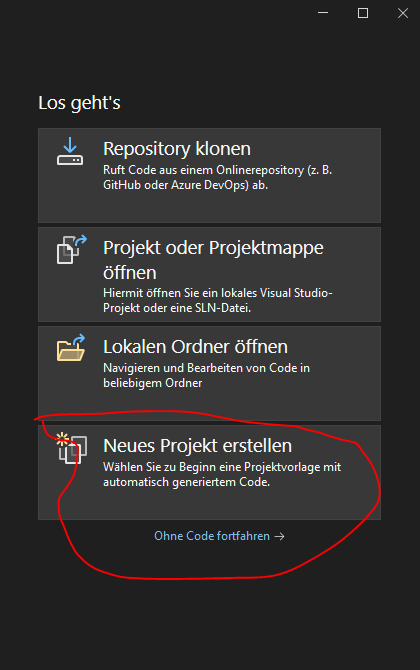
**Erste Schritte mit Visual Studio Community**

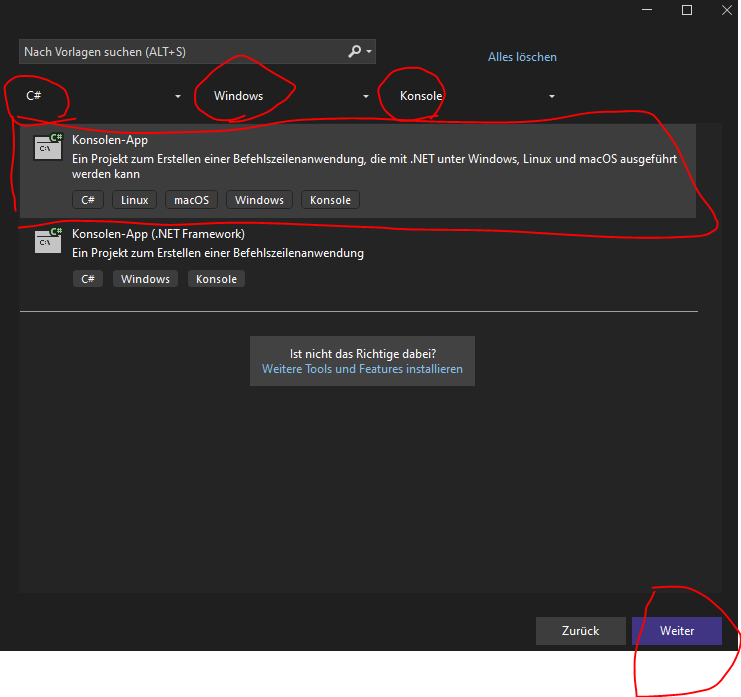
**1. Installation**

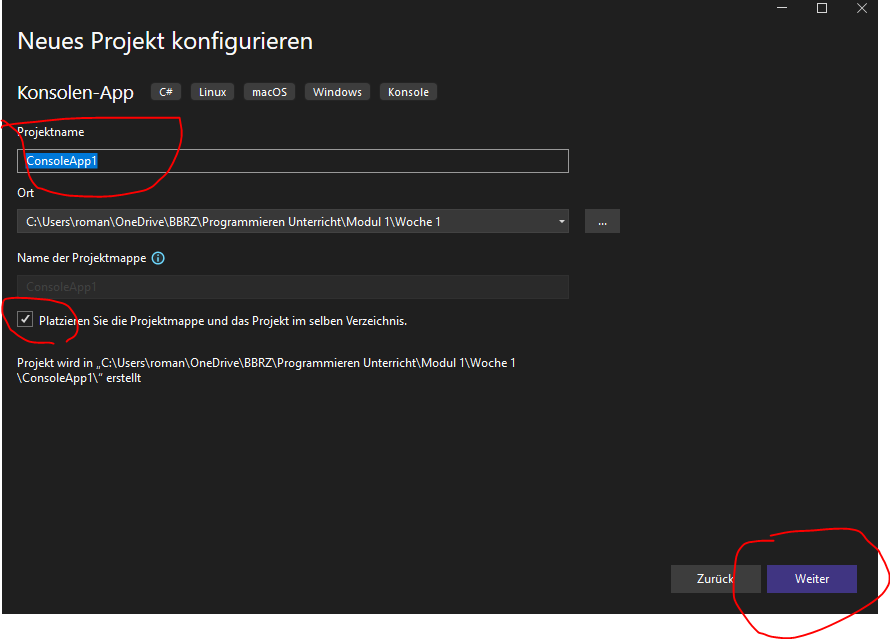
1. Lade Visual Studio Community von [visualstudio.microsoft.com](https://visualstudio.microsoft.com/) herunter.
2. Starte den Installer und wähle die gewünschten **Workloads**:
   * **.NET-Desktopentwicklung** für C#-Anwendungen.
   * **ASP.NET und Webentwicklung** für Webanwendungen.
   * **Spieleentwicklung mit Unity** für Spiele.

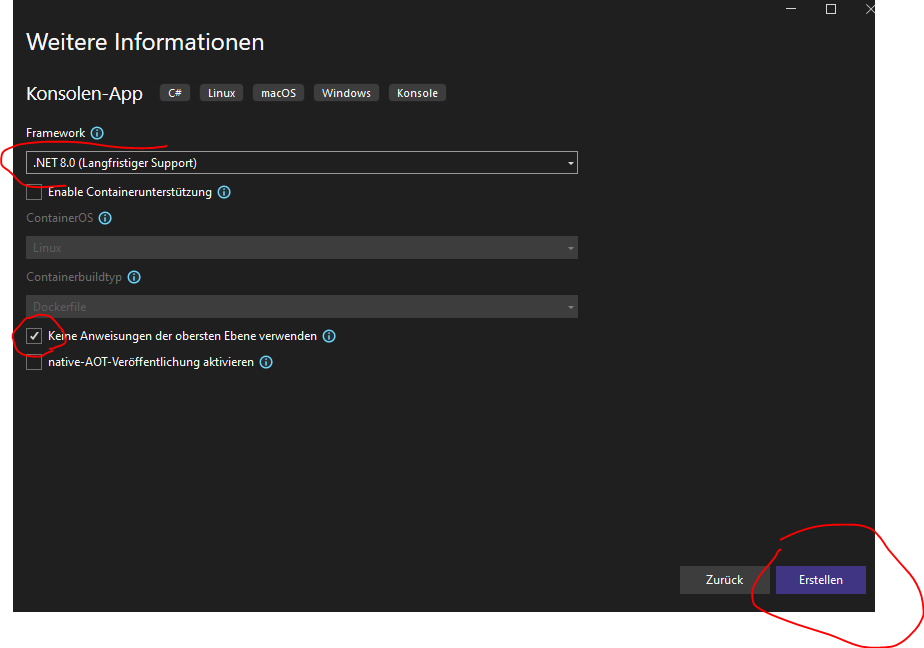
**2. Neues Projekt erstellen**

Starten Sie Ihr **Visual Studio**









**3. Code schreiben**

Erweitern Sie den vorhandenen Code um eine Nächste **Console.WriteLine** – Zeile mit Text Ihrer Wahl

**4. Programm ausführen**

* Drücken Sie **F5**, um das Programm auszuführen.

***Visual Studio auf Deutsch umstellen***

1. *Gehe zu* ***Extras → Optionen → Internationale Einstellungen****.*
2. *Wähle* ***Deutsch*** *als Sprache aus.*
3. Starte Visual Studio neu, damit die Änderungen wirksam werden.

**Fortgeschrittene Übung**

1. Erstellen Sie eine neue Konsolen-App (wie im Punkt 2 oben).
2. Versuchen Sie nun alleine oder unter Anleitung Ihres Trainers den Java-Code des Beispiels „Wald„ in C# zu übersetzen.
3. Führe Sie das Programm aus.

<https://github.com/RomanPeterP/Programmieren-JET-/tree/master/Modul%201/Woche%201>