Vorlesung Softwareentwicklung 2021

https://github.com/SebastianZug/CsharpCourse

André Dietrich Christoph Pooch Fabian Bär Fritz Apelt Galina Rudolf JohannaKlinke Jonas Treumer KoKoKotlin Lesestein LinaTeumer MMachel Sebastian Zug Snikker123 Yannik Höll Florian2501 DEVensiv fb89zila

.NET und Einordnung der Sprache C

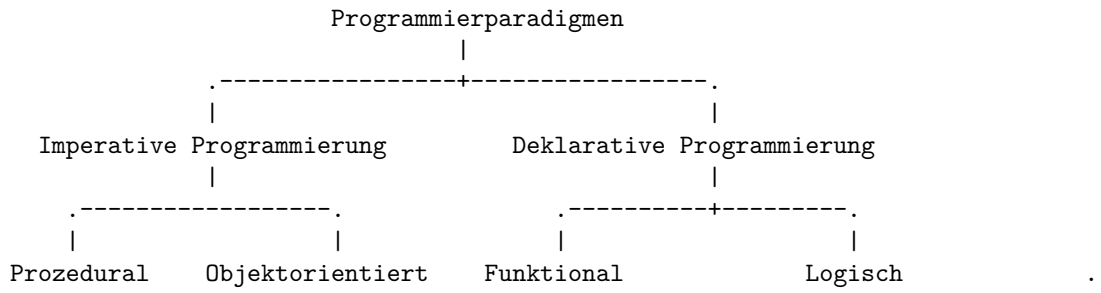
Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Softwareentwicklung`
Semester	`Sommersemester 2022`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Basiskonzepte von C# und dotnet`
Link auf den GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_Softwareentwicklung/blob/master/02_DotNet.md
Autoren	@author

Einschub: Programmierparadigmen

Ein Programmierparadigma bezeichnet die gedankliche, konzeptionelle Grundstruktur die der Darstellung des Problems in Code zugrunde liegt.

Das Programmierparadigma:

- beschreibt den fundamentalen Programmierstil bzw. Eigenschaften von Programmiersprachen
- unterscheidet sich durch die Repräsentation der statischen und dynamischen Programmelemente
- beruht auf Sprache, aber auch auf individuellem Stil.



- **Imperative Programmierung** - Quellcode besteht aus einer Folge von Befehlen die in einer festen Reihenfolge abgearbeitet werden.
  - **Prozedurale Programmierung** - Zerlegung von Programmen in überschaubare Teile, die durch eine definierte Schnittstelle aufrufbar sind (Kernkonzepte: Funktion, Prozedur, Routine, Unterprogramm)
  - **Objektorientierte Programmierung** - Kapselung von Daten und Funktionen in einem Konzept
- **Deklarative Programmierung** - es wird kein Lösungsweg implementiert, sondern nur angegeben, was gesucht ist.

  - **Funktionale Sprachen** - Abbildung der Algorithmen auf funktionale Darstellungen
  - **Logische Sprachen** - Ableitung einer Lösung aus einer Menge von Fakten, Generierung einer Auswahl von Daten

```
% Prolog Text mit Fakten
mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).

grossvater(X,Y) :-
     vater(X,Z),
     vater(Z,Y).

grossvater(adam,frank).
```

- **Weiter Konzepte** … keine spezifische Zuordenbarkeit
  - **Strukturierte Programmierung** … Verzicht bzw. Einschränkung des `Goto` Statements zugunsten von Kontrollstrukturen (Kernkonzepte: Verzweigungen, Schleifen)
  - **Nebenläufig**, **Reflektiv**, **Generisch**, …
  - **Aspektorientierte Programmierung**,

Viele Sprachen unterstützen verschiedene Elemente der Paradigmen, bzw. entwickeln sich in dieser Richtung weiter.

Sprache	imperativ	deklarativ
Pascal	prozedural	
C	prozedural	
Ada	objektorientiert	
Java	objektorientiert	
Python	objektorientiert,	funktional
C#	prozedural, objektorientiert	funktional
C++	prozedural, objektorientiert	funktional
Haskell		funktional
Prolog		Logisch
SQL		Logisch

Viele Paradigmen in einer Sprache am Beispiel eines Python Programmes … Berechnen Sie die Summe der Ziffern eines Arrays.

```
my_list = range(0,10)

# imperative
result = 0
for x in my_list:
    result += x
print("Result in imperative style      :" + str(result))

# procedural
result = 0
def do_add(list_of_numbers):
    result = 0
    for x in my_list:
```

```
        result += x
    return result
print("Result in procedural style      :" + str(do_add(my_list)))

# object oriented
class MyClass(object):
    def __init__(self, any_list):
        self.any_list = any_list
        self.sum = 0
    def do_add(self):
      self.sum = sum(self.any_list)
create_sum = MyClass(my_list)
create_sum.do_add()
print("Result in object oriented style :" + str(create_sum.sum))

# functional
import functools
result = functools.reduce(lambda x, y: x + y, my_list)
print("Result in functional style      :" + str(result))
```

@Pyodide.eval

“Das ist ja alles gut und schön, aber ich ich bin C Programmierer!”

> **Anti-Pattern “Golden Hammer”**: *If all you have is a hammer, everything looks like a nail.*

Lösungsansätze: * Individuell - Hinterfragen des Vorgehens und der Intuition, bewusste Weiterentwicklung des eigenen Horizontes (ohne auf jeden Zug aufzuspringen) * im Team - Teilen Sie Ihre Erfahrungen im Team / der Community, besetzen Sie Teams mit Mitarbeitern unterschiedlichen Backgrounds (Technical Diversity)

Weitere Diskussion unter: https://sourcemaking.com/antipatterns/golden-hammer

Warum also C#?

C# wurde unter dem Codenamen *Cool* entwickelt, vor der Veröffentlichung aber umbenannt. Der Name C Sharp leitet sich vom Zeichen Kreuz (#, englisch sharp) der Notenschrift ab, was dort für eine Erhöhung des Grundtons um einen Halbton steht. C sharp ist also der englische Begriff für den Ton *cis* (siehe Anspielung auf C++)

C#

- ist eine moderne und nur in überschaubarem Maße durch die eigene Entwicklung “verschandelte” Sprache
- enthält Elemente vieler verschiedener Paradigmen
- ist plattformunabhängig
- bietet eine breite Sammlung von Bibliotheken
- integriert Bibliotheken und Konzepte für die GUI-Programmierung
- kann mit anderen Sprachen über .NET interagieren
- unterstützt Multi-Processing problemlos
- ist typsicher
- …

Historie der Sprache C

Jahr	Version .NET	Version C#	Ergänzungen
2002	1.0	1.0	
2006	3.0	2.0	Generics, Anonyme Methoden, Iteratoren, Private setters, Delegates
2007	3.5	3.0	Implizit typisierte Variablen, Objekt- und Collection-Initialisierer, Automatisch implementierte Properties, LINQ, Lambda Expressions
2010	4.0	4.0	Dynamisches Binding, Benannte und optionale Argumente, Generische Co- und Kontravarianz
2012	4.5	5.0	Asynchrone Methoden

Jahr	Version .NET	Version C#	Ergänzungen
2015	4.6	6.0	Exception Filters, Indizierte Membervariablen und Elementinitialisierer, Mehrzeilige String-Ausdrücke, Implementierung von Methoden mittels Lambda-Ausdruck
2017	4.6.2/ .NET Core	7.0	Mustervergleiche (Pattern matching), Binärliterale 0b…, Tupel
2019	.NET Core 3	8.0	Standardimplementierungen in Schnittstellen, Switch Expressions, statische lokale Funktionen, Index-Operatoren für Teilmengen
2020	.NET 5.0	9.0	Datensatztypen (Records), Eigenschafteninitialisierung, Anweisungen außerhalb von Klassen, Verbesserungen beim Pattern Matching
2021	.NET 6.0	10.0	Erforderliche Eigenschaften, Null-Parameter-Prüfung, globale Using-Statements

Die Sprache selbst ist unmittelbar mit der Ausführungsumgebung, dem .NET Konzept verbunden und war ursprünglich stark auf Windows Applikationen zugeschnitten.

Konzepte und Einbettung

.NET ist ein Sammelbegriff für mehrere von Microsoft/Dritten herausgegebene Software-Plattformen, die der Entwicklung und Ausführung von Anwendungsprogrammen dienen. Dabei erlebt die Plattform einen permanenten Wandel. Die Bedeutung der einzelnen Teile und Technologien, die .NET umfasst, hat sich im Laufe der Zeit gewandelt. Stand November 2020 spielen folgende Frameworks eine herausgehobene Rolle in der Praxis:

- das nur unter Windows unterstützte klassische .NET Framework, das mit der Version 4.8 in der letzten Version vorliegt.
- das als dessen Nachfolger positionierte, auf verschiedenen Plattformen unterstützte Framework .NET 5 (bei dem auch einige Techniken gekündigt wurden). Es wurde mehrere Jahre parallel unter der Bezeichnung .NET Core entwickelt.
- die Plattform Mono und darauf basierende Techniken (von Microsoft meist als Xamarin bezeichnet). Diese unterstützt seit längerem .NET auf verschiedenen Plattformen (in der Vergangenheit jedoch oft unvollständig implementiert).

Mono ist eine alternative, ursprünglich unabhängige Implementierung von Microsofts .NET Standards. Sie ermöglicht die Entwicklung von plattformunabhängiger Software auf den Standards der Common Language Infrastructure und der Programmiersprache C#. Entstanden ist das Mono-Projekt 2001 unter Führung von Miguel de Icaza von der Firma Ximian, die 2003 von Novell aufgekauft wurde. Die Entwickler wurden 2011 in eine neue Firma namens Xamarin übernommen, die im Jahr 2016 eine Microsoft-Tochtergesellschaft wurde. In der Folge wurde Microsoft Hauptsponsor des Projektes.

Mono "hinkt" als Open Source Projekt der eigentlichen Entwicklung etwas nach.

```
   .NET Core 3.1
   Teile von .NET Framework
   Teile von Mono
+  neue Features
------------------
   .NET 5.0
```

Der Artikel in heise fasst diesen Status gut zusammen.

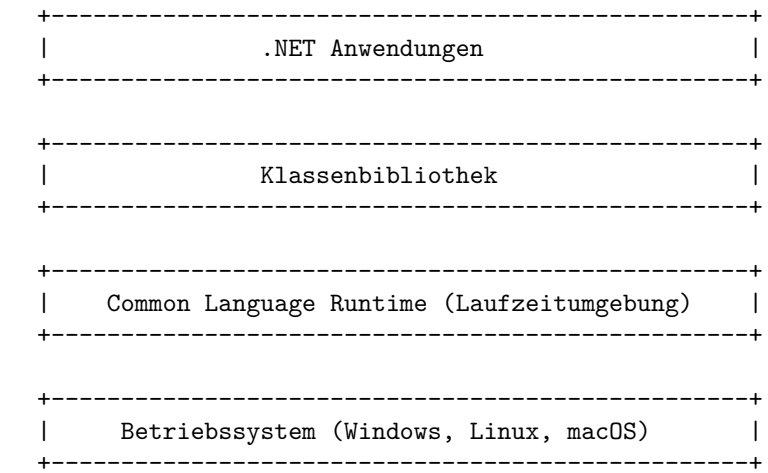
Zum Vergleich sei auf eine Darstellung der **vor dem Erscheinen von .NET 5** verwiesen:

Betriebssystem	.NET Framework	.NET Core	Xamarin
Windows 7/8	X	X	
Windows 10 Desktop	X	X	
Windows 10 Mobile Geräte			X
Linux		X	
macOS		X	
iOS			X

Einen guten Überblick über das historische Nebeneinander gibt das Video von Tim Corey (Achtung, die Darstellung greift den Stand von 2017 auf!) Link

.NET 6 ist als LTS-Version im November 2021 erscheinen und wird für 3 Jahre unterstützt.

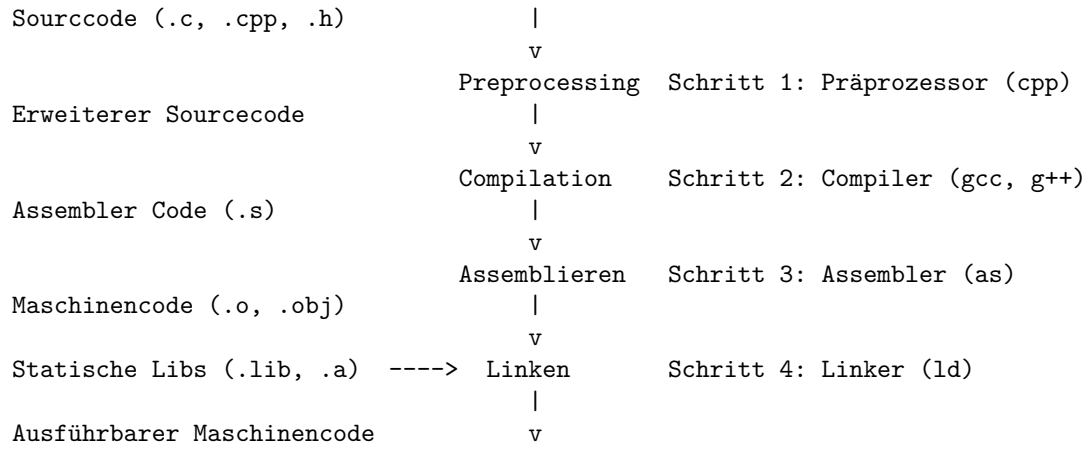
Ziel des .NET-Ökosystems ist die Erhöhung der Anwendungskompatibilität zwischen verschiedenen Systemen und Plattformen. Programme, die das .NET Framework verwenden, werden in der Regel so ausgeliefert, dass benötigte Komponenten des Frameworks automatisch mit installiert werden.



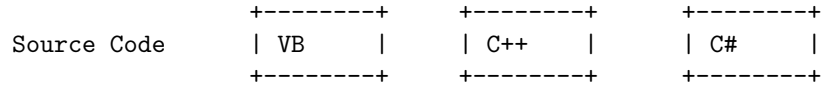
- die Laufzeitumgebung (CLR) implementiert die Ausführungsplattform des .NET Codes. Sie umfasst die Sicherheitsmechanismen, Versionierung, automatische Speicherbereinigung und vor allem die Entkopplung der Programmausführung vom Betriebssystem.
- die Klassenbibliothek gliedern sich intern in Basisklassen und eigenen Bibliotheken für verschiedene Anwendungstypen:
  - ASP.NET … ist ein Web Application Framework, mit dem sich dynamische Webseiten, Webanwendungen und Webservices entwickeln lassen.
  - Windows Forms/ WPF … ist ein GUI-Toolkit des Microsoft .NET Frameworks. Es ermöglicht die Erstellung grafischer Benutzeroberflächen (GUIs) für Windows.
  - …

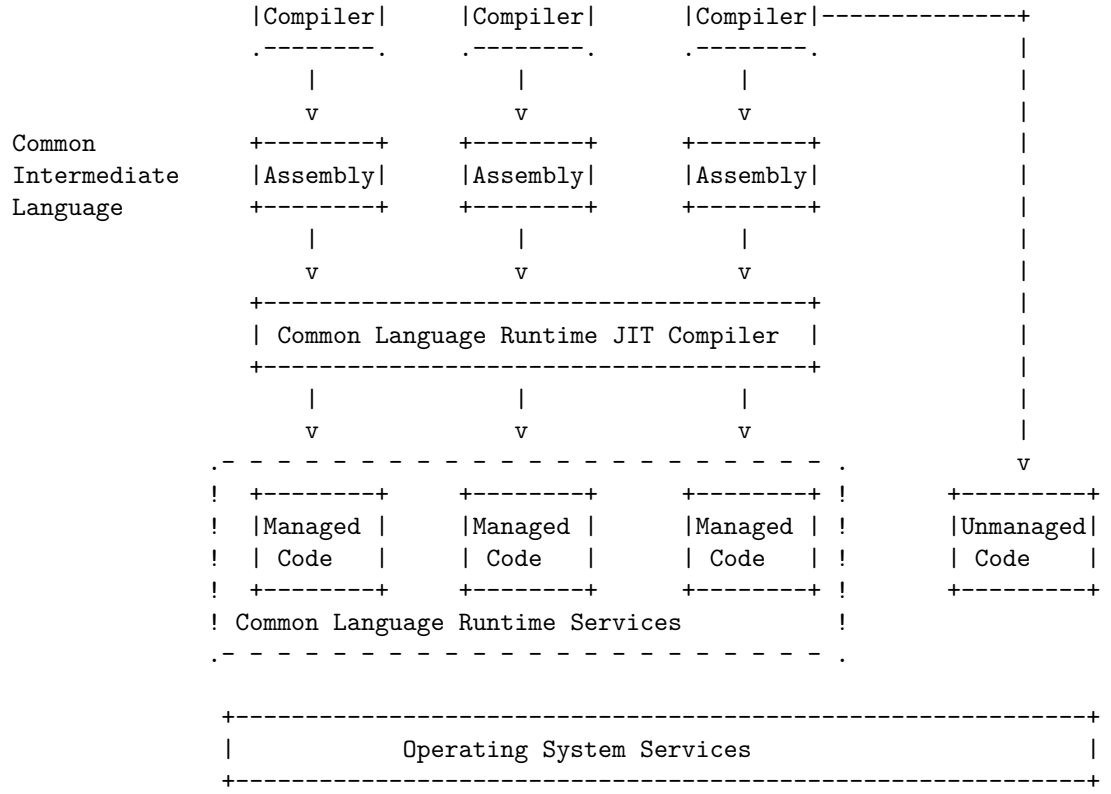
Die Open Source Community stand dem .NET Konzept kritisch gegenüber, da eine "unklare Lage" im Hinblick auf die Lizenzen bestand. Aufgrund der Gefahr durch Patentklagen seitens Microsoft warnte Richard Stallman davor Mono in die Standardkonfiguration von Linuxdistributionen aufzunehme. Ab 2013 änderte Microsoft aber seine Strategie und veröffentlichte den Quellcode von .NET komplett als Open Source unter einer MIT-Lizenz bzw. Apache-2.0-Lizenz.

Compilierung unter C (zur Erinnerung und zum Vergleich)



Build-Prozess mit C#





Die spezifischen Compiler der einzelnen .NET Sprachen (C#. Visual Basic, F#) bilden den Quellcode auf einen Zwischencode ab. Die Common Language Infrastructure (CLI) ist eine von ISO und ECMA standardisierte offene Spezifikation (technischer Standard), die ausführbaren Code und eine Laufzeitumgebung beschreibt.

Was passiert unter der Haube der CLR?

Für die *Managed Code Execution* stellt die CLR ein entsprechendes Set von Komponenten bereit:

- Class Loader … Einlesen der Assemblies in die CLR Ausführungsumgebung unter Beachtung der Sicherheits-, Versions-, Typinformationen usw.
- Just-in-Time Compiler … Abbildung der CIL auf den ausführbaren Maschinencode
- Code Execution und Debugging
- Garbage Collection … der GC ist für die Bereinigung von Referenz-Objekten auf dem Heap verantwortlich und wird von der CLR zu nicht-deterministischen Zeitpunkten gestartet.

```
.assembly HalloWelt { }
.assembly extern mscorlib { }
.method public static void Main() cil managed
{
    .entrypoint
    .maxstack 1
    ldstr "Hallo Welt!"
    call void [mscorlib]System.Console::WriteLine(string)
    ret
}
```

Ein Assembly umfasst: * das Assemblymanifest, das die Assemblymetadaten enthält. * die Typmetadaten. * den CIL-Code * Links auf mögliche Ressourcen.

Ein Assembly bildet:

- **bildet eine Sicherheitsgrenze** - Eine Assembly ist die Einheit, bei der Berechtigungen angefordert und erteilt werden.
- **bildet eine Typgrenze** - Die Identität jedes Typs enthält den Namen der Assembly, in der dieser sich befindet. Wenn der Typ `MyType` in den Gültigkeitsbereich einer Assembly geladen wird, ist dieser nicht derselbe wie der Typ `MyType`, der in den Gültigkeitsbereich einer anderen Assembly geladen wurde.
- **bildet eine Versionsgrenze** - Die Assembly ist die kleinste, in verschiedenen Versionen verwendbare Einheit in der Common Language Runtime. Alle Typen und Ressourcen in derselben Assembly bilden eine Einheit mit derselben Version.

- **bildet eine Bereitstellungseinheit** - Beim Starten einer Anwendung müssen nur die von der Anwendung zu Beginn aufgerufenen Assemblys vorhanden sein. Andere Assemblys, z. B. Lokalisierungsressourcen oder Assemblys mit Hilfsklassen, können bei Bedarf abgerufen werden. Dadurch ist die Anwendung beim ersten Herunterladen einfach und schlank.

Abgrenzung zu Java

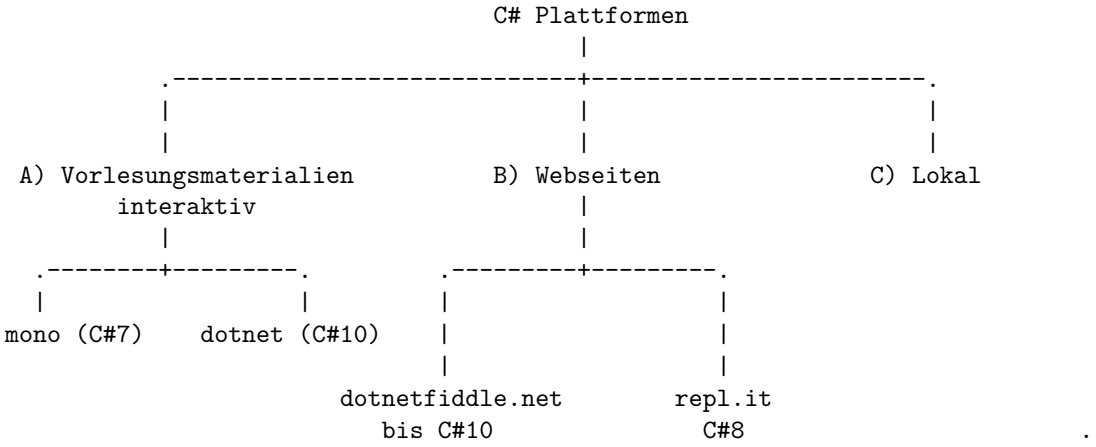
vgl. Vortrag von Mössenböck (link)

	Java	C#
Veröffentlichung	1995	2001
Plattform	(Java) Unix/Linux, Windows, MacOS, Android	(.NET) Windows, Linux, Android, iOS, MacOS
VM	Java-VM	CLR
Zwischencode	Java-Bytecode	CIL
JIT	per Methoden	per Methode / gesamt
Komponenten	Beans	Assemblies
Versionierung	nein	ja
Leitidee	Eine Sprache auf vielen Plattformen	Viele Sprachen auf vielen Plattform

Es wird konkret ... Hello World

Die organisatorischen Schlüsselkonzepte in C# sind: **Programme**, **Namespaces**, **Typen**, **Member** und **Assemblys**. C#-Programme bestehen aus mindestens einer Quelldatei, von denen mindestens eine `Main` als einen Methodennamen hat. Programme deklarieren Typen, die Member enthalten, und können in Namespaces organisiert werden.

Wenn C#-Programme kompiliert werden, werden sie physisch in Assemblys verpackt. Assemblys haben unter Windows Betriebssystemen die Erweiterung .exe oder .dll, je nachdem, ob sie Anwendungen oder Bibliotheken implementieren.



A) Vorlesungsmaterialien - LiaScript Umgebung

Die LiaScript basierte Komplierung und Ausführung kann wie bereits erläutert auf der Basis von mono und dem dotnet Framework umgesetzt werden.

```
using System;

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

```
using System;
```

```
Console.WriteLine("Hello, world!");
```

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
</Project>
```

B) Repl.it

https://replit.com/

C) .NET Kommandozeile

Das .NET Core Framework kann unter .NET für verschiedene Betriebssystem heruntergeladen werden. Das SDK umfasst sowohl die Bibliotheken, Laufzeitumgebung und Tools. An dieser Stelle sei nur auf die `dotnet` Tools verwiesen, die anderen Werkzeuge werden zu einem späteren Zeitpunk eingeführt.

```
> dotnet new console
> dotnet build
> dotnet run
```

Aus dem Generieren eines neuen Konsolenprojektes ergibt sich ein beeindruckender Baum von Projektdateien.

```
> dotnet new console

> tree
.
├── bin
│   └── Debug
│       └── net5.0
│           ├── ref
│           │   └── visual_studio_code.dll
│           ├── visual_studio_code
│           ├── visual_studio_code.deps.json
│           ├── visual_studio_code.dll
│           ├── visual_studio_code.pdb
│           ├── visual_studio_code.runtimeconfig.dev.json
│           └── visual_studio_code.runtimeconfig.json
├── obj
│   ├── Debug
│   │   └── net5.0
│   │       ├── apphost
│   │       ├── ref
│   │       │   └── visual_studio_code.dll
│   │       ├── visual_studio_code.AssemblyInfo.cs
│   │       ├── visual_studio_code.AssemblyInfoInputs.cache
│   │       ├── visual_studio_code.assets.cache
│   │       ├── visual_studio_code.csprojAssemblyReference.cache
│   │       ├── visual_studio_code.csproj.CoreCompileInputs.cache
│   │       ├── visual_studio_code.csproj.FileListAbsolute.txt
│   │       ├── visual_studio_code.dll
│   │       ├── visual_studio_code.GeneratedMSBuildEditorConfig.editorconfig
│   │       ├── visual_studio_code.genruntimeconfig.cache
│   │       └── visual_studio_code.pdb
│   ├── project.assets.json
│   ├── project.nuget.cache
│   ├── visual_studio_code.csproj.nuget.dgspec.json
│   ├── visual_studio_code.csproj.nuget.g.props
│   └── visual_studio_code.csproj.nuget.g.targets
├── Program.cs
└── visual_studio_code.csproj
```

C) .NET Visual Code

Alternativ können Sie auch die Microsoft Visual Studio oder Visual Code Suite nutzen. Diese kann man zum Beispiel auf unser gerade erstelltes Projekt anwenden

https://code.visualstudio.com/docs/languages/csharp

Aufgaben

- [ ] Installieren Sie das .NET 6 auf Ihrem Rechner und erfreuen Sie sich an einem ersten “Hello World”
- [ ] Testen Sie mit einem Kommilitonen die Features von repl.it! Arbeiten Sie probeweise an einem gemeinsamen Dokument.
- [ ] Legen Sie sich einen GitHub-Account an.