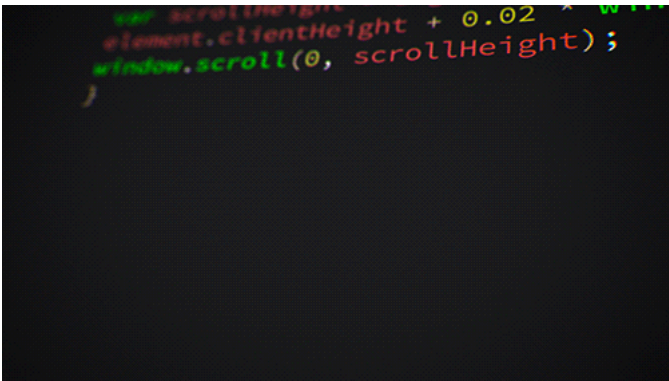


Einführung

| Parameter | Kursinformationen |
|----------------------|---|
| Veranstaltung: | Vorlesung Softwareentwicklung |
| Semester | Sommersemester 2022 |
| Hochschule: | Technische Universität Freiberg |
| Inhalte: | Motivation der Vorlesung "Softwareentwicklung" und Beschreibung der Organisation der Veranstaltung |
| Link auf den GitHub: | https://github.com/TUBAF-lfi-LiaScript/VL_Softwareentwicklung/blob/master/00_Einfuehrung.md |
| Autoren | Sebastian Zug, Galina Rudolf, André Dietrich, Fritz Apelt, KoKoKotlin |



Zielstellung der Veranstaltung



Qualifikationsziele / Kompetenzen

Studierende sollen ...

- die Konzepte objektorientierten und interaktiven Programmierung verstehen,
- die Syntax und Semantik einer objektorientierten Programmiersprache beherrschen um Probleme kollaborativ bei verteilter Verantwortlichkeit von Klassen von einem Computer lösen lassen,
- in der Lage sein, interaktive Programme unter Verwendung einer objektorientierten Klassenbibliothek zu erstellen.

[Auszug aus dem Modulhandbuch 2020]

Zielstellung der Veranstaltung

Wir lernen effizient guten Code in einem kleinen Team zu schreiben.

| Genereller Anspruch | Spezifischer Anspruch |
|---|--|
| Verstehen verschiedener Programmierparadigmen UNABHÄNGIG von der konkreten Programmiersprache | Objektorientierte (und funktionale) Programmierung am Beispiel von C# |
| praktische Einführung in die methodische Softwareentwicklung | Arbeit mit ausgewählten UML Diagrammen und Entwurfsmustern |
| Grundlagen der kooperativ/kollaborative Programmierung und Projektentwicklung | Verwendung von Projektmanagementtools und einer Versionsverwaltung für den Softwareentwicklungsprozess |

Obwohl Einstimmigkeit darüber besteht, dass kooperative Arbeit für Ingenieure Grundlage der täglichen Arbeitswelt ist, bleibt die Wissensvermittlung im Rahmen der Ausbildung nahezu aus.

Frage: *Welche Probleme sehen Sie bei der Teamarbeit?*

Spezifisches Ziel: Wir wollen Sie für die Konzepte und Werkzeuge der kollaborativen Arbeit bei der Softwareentwicklung "sensibilisieren".

- Wer definiert die Feature, die unsere Lösung ausmachen?
- Wie behalten wir bei synchronen Codeänderungen der Überblick?
- Welchen Status hat die Erfüllung der Aufgabe X erreicht?
- Wie können wir sicherstellen, dass Code in jedem Fall kompiliert und Grundfunktionalitäten korrekt ausführt?
- ...

Wozu brauche ich das?

Anhand der Veranstaltung entwickeln Sie ein "Gefühl" für guten und schlechten Code und hinterfragen den Softwareentwicklungsprozess.

Beispiel 1: Mariner 1 Steuerprogramm-Bug (1962)

[Atlas-Agena]



Mariner 1 ging beim Start am 22. Juli 1962 durch ein fehlerhaftes Steuerprogramm verloren, als die Trägerrakete vom Kurs abkam und 293 Sekunden nach dem Start gesprengt werden musste. Ein Entwickler hatte einen Überstrich in der handgeschriebenen Spezifikation eines Programms zur Steuerung des Antriebs übersehen und dadurch statt geglätteter Messwerte Rohdaten verwendet, was zu einer fehlerhaften und potenziell gefährlichen Fehlsteuerung des Antriebs führte.

[Link auf Beschreibung des Bugs](#)

Potentieller Lösungsansatz: Testen & Dokumentation

Beispiel 2: Toll-Collect On-Board-Units (2003)

Das Erfassungssystem für die Autobahngebühren für Lastkraftwagen sollte ursprünglich zum 31. August 2003 gestartet werden. Nachdem die organisatorischen und technischen Mängel offensichtlich geworden waren, erfolgte eine mehrfache Restrukturierung. Seit 1. Januar 2006 läuft das System, mit einer Verzögerung von über zwei Jahren, mit der vollen Funktionalität. Eine Baustelle war die On-Board-Units (OBU), diese konnte zunächst nicht in ausreichender Stückzahl geliefert und eingebaut werden, da Schwierigkeiten mit der komplexen Software der Geräte bestanden.

Die On-Board-Units des Systems

- reagierten nicht auf Eingaben
- ließen sich nicht ausschalten
- schalteten sich grundlos aus
- zeigten unterschiedliche Mauthöhen auf identischen Strecken an
- wiesen Autobahnstrecken fehlerhaft als mautfrei/mautpflichtig aus

Potentieller Lösungsansatz: Testen auf Integrationsebene, Projektkoordination

[Atlas-Agena] wikimedia, Autor: NASA, [Link](#)

Organisatorisches

Dozenten

| Name | Email |
|---------------|---|
| Sebastian Zug | sebastian.zug@informatik.tu-freiberg.de |
| Galina Rudolf | galina.rudolf@informatik.tu-freiberg.de |
| Nico Sonack | nico.sonack@student.tu-freiberg.de |
| Felix Busch | Felix.Busch@student.tu-freiberg.de |
| Anne Gierig | anne.gierich@student.tu-freiberg.de |

Ablauf

Jetzt wird es etwas komplizierter ... die Veranstaltung kombiniert nämlich zwei Vorlesungen:

| | <i>Softwareentwicklung (SWE)</i> | <i>Einführung in die Softwareentwicklung (EiS)</i> |
|-----------------|---|---|
| Hörerkreis | Fakultät 1 + interessierte Hörer | Fakultät 4 - Studiengang Engineering |
| Leistungspunkte | 9 | 6 |
| Vorlesungen | 28 (2 Feiertage) | 15 (bis 31. Mai 2021) |
| Übungen | ab Mai 2 x wöchentlich | ab 25. April 8 Übungen |
| | | zusätzliches Python Tutorial ab Juni |
| Prüfungsform | Klausur oder Projekt | maschinenbauspezifisches Software-Projekt (im Wintersemester 2021/22) |
| | | Prüfungsvoraussetzung: Erfolgreiche Bearbeitung der finalen Aufgabe im Sommersemester |

Ermunterung an unsere EiS-Hörer: Nehmen Sie an der ganzen Vorlesungsreihe teil. Den Einstieg haben Sie ja schon gelegt ...

Struktur der Vorlesungen

| Woche | Tag | Inhalt der Vorlesung | Bemerkung |
|-------|-----------|---|----------------------------|
| 1 | 4. April | Organisation, Einführung von GitHub und LiaScript | |
| | 8. April | Softwareentwicklung als Prozess | |
| 2 | 11. April | Konzepte von Dotnet und C# | |
| | 15. April | <i>Karfreitag</i> | |
| 3 | 18. April | <i>Ostermontag</i> | |
| | 22. April | Elemente der Sprache C# (Datentypen) | |
| 4 | 25. April | Elemente der Sprache C# (Forts. Datentypen) | |
| | 29. April | Elemente der Sprache C# (Ein-/Ausgaben) | |
| 5 | 2. Mai | Programmfluss und Funktionen | |
| | 6. Mai | Strukturen / Konzepte der OOP | |
| 6 | 9. Mai | Säulen Objektorientierter Programmierung | |
| | 13. Mai | Klassenelemente in C# / Vererbung | |
| 7 | 16. Mai | Klassenelemente in C# / Vererbung | |
| | 20. Mai | Versionsmanagement im Softwareentwicklungsprozess | |
| 8 | 23. Mai | UML Konzepte | |
| | 27. Mai | UML Diagrammtypen | |
| 9 | 30. Mai | UML Anwendungsbeispiel | |
| | 3. Juni | Testen | Ende EIS Vorlesungsinhalte |
| 10 | 6. Juni | <i>Pfingstmontag</i> | |
| | 10. Juni | Dokumentation und Build Toolchains | |
| 11 | 13. Juni | Continuous Integration in GitHub | |
| | 17. Juni | Generics | |
| 12 | 20. Juni | Container | |
| | 24. Juni | Delegaten | |
| 13 | 27. Juni | Events | |
| | 1. Juli | Threadkonzepte in C# | |
| 14 | 4. Juli | Taskmodell | |
| | 8. Juli | Language Integrated Query | |
| 15 | 11. Juli | Design Pattern | |
| | 15. Juli | | |

Durchführung

Die Vorlesung wurden im vergangenen Semester aufgezeichnet. Die Inhalte finden sich unter

<https://teach.informatik.tu-freiberg.de/b/seb-blv-unz-kxu>

Diese Materialien können der Nachbereitung der Veranstaltung dienen, ersetzen aber nicht den Besuch der Vorlesung.

Die Vorlesung findet

- Montags, 11:00 - 12:30
- Freitags, 9:15 - 10:45

im Audimax 1001 statt.

Die Materialien der Vorlesung sind als Open-Educational-Ressources konzipiert und stehen unter Github bereit.

Wie können Sie sich einbringen?

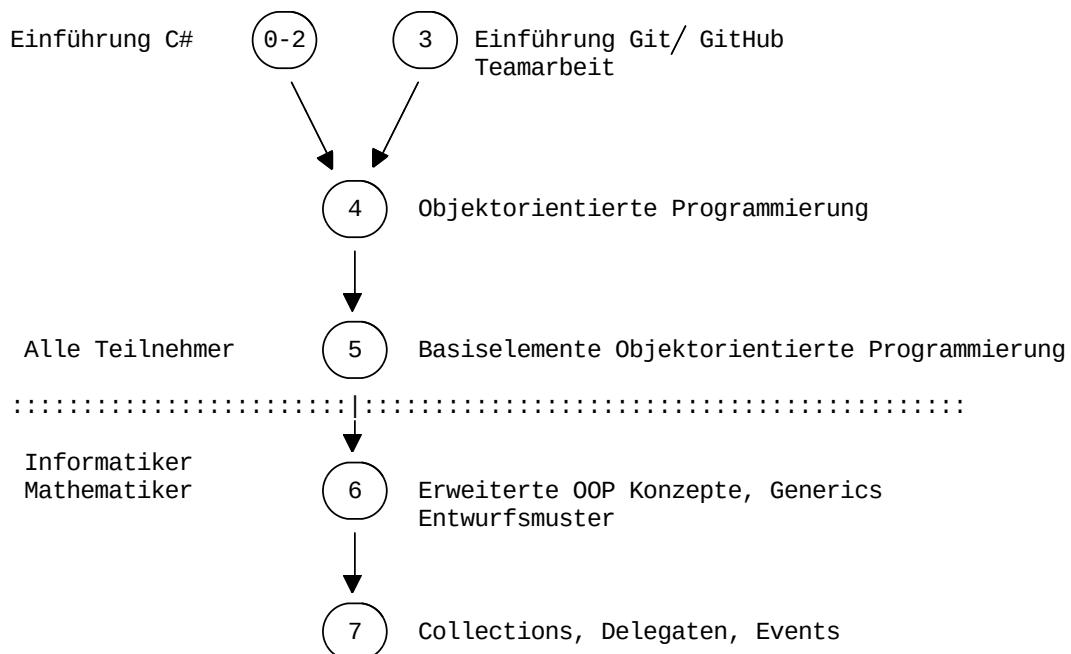
- **Allgemeine theoretische Fragen/Antworten ...** Dabei können Sie sich über github/ das Opal-Forum in die Diskussion einbringen.
- **Rückmeldungen/Verbesserungsvorschläge zu den Vorlesungsmaterialien ...** *"Das versteht doch keine Mensch! Ich würde vorschlagen ..."* ... dann korrigieren Sie uns. Alle Materialien sind Open-Source. Senden Sie mir einen Pull-Request und werden Sie Mitautor.

Die Übungen bestehen aus selbständig zu bearbeitenden Aufgaben, wobei einzelne Lösungen im Detail besprochen werden. Wir werden die Realisierung der Übungsaufgaben über die Plattform GitHub abwickeln.

Wie können Sie sich einbringen?

- **Allgemeine praktische Fragen/Antworten ...** in den genannten Foren bzw. in den Übungsveranstaltungen
- **Eigene Lösungen ...** Präsentation der Implementierungen in den Übungen
- **Individuelle Fragen ...** an die Übungsleiter per Mail oder in einer individuellen Session

Für die Übungen werden wir Aufgaben vorbereiten, mit denen die Inhalte der Vorlesung vertieft werden. Wir motivieren Sie sich dafür ein Gruppen von 2 Studierenden zu organisieren.



| Index | C# | GitHub | Teamarbeit | Inhalte / Teilaufgaben | Woche |
|-------|--------|--------|------------|--|-------|
| 0 | Basics | nein | nein | Toolchain, Datentypen, Fehler, Ausdrücke, | 5 |
| 1 | | | | Kontrollfluss, Arrays | 6 |
| 2 | | | | static Funktionen, Klasse und Struktur, Nullables | 7 |
| 3 | - | ja | ja | Github am Beispiel von Markdown | 8 |
| 4 | OOP | ja | ja | Einführungsbeispiel OOP, | 9 |
| | | | | <i>Anwendungsbeispiel:</i> Computersimulation | |
| 5 | OOP | ja | ja | Vererbung, virtuelle Methoden, Indexer, Überladene Operatoren, | 10-11 |
| | | | | <i>Anwendungsbeispiel:</i> Smartphone (Entwurf mit UML) | |
| 6 | OOP | ja | ja | Vererbung, abstract, virtuell, Generics | 12-13 |
| | | | | <i>Anwendungsbeispiel:</i> Zoo | |
| 7 | OOP | ja | ja | Generische Collections, Delegaten, Events | 14-15 |
| | | | | <i>Anwendungsbeispiel:</i> ??????? | |

Prüfungen

In der Klausur werden neben den Programmierfähigkeiten und dem konzeptionellen Verständnis auch die Werkzeuge der Softwareentwicklung adressiert!

- **Softwareentwicklung:** Konventionelle Klausur ODER Programmieraufgabe in Zweier-Team anhand einer selbstgewählten Aufgabe
- **Einführung in die Softwareentwicklung:** Teamprojekt und Projektpräsentationen (im Wintersemester 2022/23) bei bestandener Prüfungsvorleistung in Form einer Teamaufgabe im Sommersemester

Zeitaufwand und Engagement

Mit der Veranstaltung Softwareentwicklung verdienen Sie sich **9 CP** / **6 CP**. Eine Hochrechnung mit der von der Kultusministerkonferenz vorgegebenen Formel **1 CP = 30 Zeitstunden** bedeutet, dass Sie dem Fach im Mittel über dem Semester **270 Stunden** widmen sollten ... entsprechend bleibt neben den Vorlesungen und Übungen genügend Zeit für die Vor- und Nachbereitung der Lehrveranstaltungen, die eigenständige Lösung von Übungsaufgaben sowie die Prüfungsvorbereitung.

"Erzähle mir und ich vergesse. Zeige mir und ich erinnere. Lass es mich tun und ich verstehe."

— (Konfuzius, chin. Philosoph 551-479 v. Chr.)

Wie können Sie zum Gelingen der Veranstaltung beitragen?

- Stellen Sie Fragen, seien Sie kommunikativ!
- Geben Sie uns Rückmeldungen in Bezug auf die Geschwindigkeit, Erklärmuster, etc.
- Organisieren Sie sich in *interdisziplinären* Arbeitsgruppen!
- Lösen Sie sich von vermeindlichen Grundwahrheiten:
 - "in Python wäre ich drei mal schneller gewesen"
 - "VIM ... mehr Editor braucht kein Mensch!"

Literaturhinweise

Literaturhinweise werden zu verschiedenen Themen als Links oder Referenzen in die Unterlagen integriert.

Es existiert eine Vielzahl kommerzielle Angebote, die aber einzelne Aspekte in freien Tutorial vorstellen. In der Regel gibt es keinen geschlossenen Kurs sondern erfordert eine individuelle Suche nach spezifischen Inhalten.

- **Online-Kurse:**

- [Leitfaden von Microsoft für C# aber auch die Werkzeuge](#)
- [C# Tutorial for Beginners: Learn in 7 Days](#) [englisch]
- [Einsteiger Tutorials](#) [deutsch]
- [Programmierkonzepte von C#](#)

- **Video-Tutorials:**

o

Algorithmen + [Codebeispiele](#)

- Bücher:

- [C# 7.0 in a Nutshell - J. Albahari, B. Albahari, O'Reilly 2017](#)
- [Kompaktkurs C# 7 - H. Mössenböck, dpunkt.verlag](#)

Werkzeuge der Veranstaltung

Was sind die zentralen Tools unserer Veranstaltung?

- *Vorlesungstool* → BigBlueButton für die Aufzeichnungen aus dem vergangenen Semester [Introduction](#)
- *Entwicklungsplattform* → [GitHub](#)
- *Beschreibungssprache für Lerninhalte* → [LiaScript](#)

Markdown

Markdown wurde von John Gruber und Aaron Swartz mit dem Ziel entworfen, die Komplexität der Darstellung so weit zu reduzieren, dass schon der Code sehr einfach lesbar ist. Als Auszeichnungselemente werden entsprechend möglichst kompakte Darstellungen genutzt.

Markdown ist eine Auszeichnungssprache für die Gliederung und Formatierung von Texten und anderen Daten. Analog zu HTML oder LaTeX werden die Eigenschaften und Organisation von Textelementen (Zeichen, Wörtern, Absätzen) beschrieben. Dazu werden entsprechende "Schlüsselemente" verwendet um den Text zu strukturieren.

HelloWorld.md

Überschrift

eine **Hervorhebung** in kursiver Umgebung

* Punkt 1

* Punkt 2

Und noch eine Zeile mit einer mathematischen Notation $a=\cos(b)$!

Überschrift

eine **Hervorhebung** in kursiver Umgebung

- Punkt 1
- Punkt 2

Und noch eine Zeile mit einer mathematischen Notation $a = \cos(b)$!

Eine gute Einführung zu Markdown finden Sie zum Beispiel unter:

- [MarkdownGuide](#)
- [GitHubMarkdownIntro](#)

Mit einem entsprechenden Editor und einigen Paketen macht das Ganze dann auch Spaß

- Wichtigstes Element ist ein Previewer, der es Ihnen erlaubt "online" die Korrektheit der Eingaben zu prüfen
- Tools zur Unterstützung komplexerer Eingaben wie zum Beispiel der Tabellen (zum Beispiel für Atom mit [markdown-table-editor](#))
- Visualisierungsmethoden, die schon bei der Eingabe unterstützen
- Rechtschreibprüfung (!)

Vergleich mit HTML

Im Grunde wurde Markdown erfunden um nicht umständlich HTML schreiben zu müssen und wird zumeist in HTML übersetzt. Das dargestellte Beispiel zeigt den gleichen Inhalt wie das Beispiel zuvor, es ist jedoch direkt viel schwerer zu editieren, dafür bietet es weit mehr Möglichkeiten als Markdown. Aus diesem Grund erlauben die meisten Markdown-Dialekte auch die Nutzung von HTML.

| HelloWorld.html | |
|---|--|
| <pre> <h1>Überschrift</h1> <i>eine Hervorhebung in kursiver Umgebung</i> Punkt 1 Punkt 2 Und noch eine Zeile mit einer mathematischen Notation <math xmlns="http://www.w3.org/1998/Math/MathML"> <semantics> <mrow> <mi>a</mi> <mo>=</mo> <mi>c</mi> <mi>o</mi> <mi>s</mi> <mo stretchy="false">(</mo> <mi>b</mi> <mo stretchy="false">)</mo> </mrow> <annotation encoding="application/x-tex"> a=cos(b) </annotation> </semantics> </math>!</pre> | |

Vergleich mit LaTeX

Eine vergleichbare Ausgabe unter LaTeX hätte einen deutlich größeren Overhead, gleichzeitig eröffnet das Textsatzsystem (über einzubindende Pakete) aber auch ein wesentlich größeres Spektrum an Möglichkeiten und Features (automatisch erzeugte Numerierungen, komplexe Tabellen, Diagramme), die Markdown nicht umsetzen kann.

| latexHelloWorld.tex | |
|--|--|
| <pre> \documentclass[12pt]{article} \usepackage[latin1]{inputenc} \begin{document} \section{Überschrift} \textit{eine \emph{Betonung} in kursiver Umgebung} \begin{itemize} \item Punkt 1 \item Punkt 2 \end{itemize} Und noch eine Zeile mit einer mathematischen Notation $a=\cos(b)$! \end{document}</pre> | |

Das Ergebnis sieht dann wie folgt aus:

LiaScript

Das Problem der meisten Markup-Sprachen und vor allem von Markdown ist, dass die Inhalte nicht mehr nur für ein statisches Medium (Papier/PDF) geschrieben werden. Warum sollte ein Lehrinhalt (vorallem in der Informatik), der vorranig am Tablet/Smartphone/Notebook konsumiert wird nicht interaktiv sein?

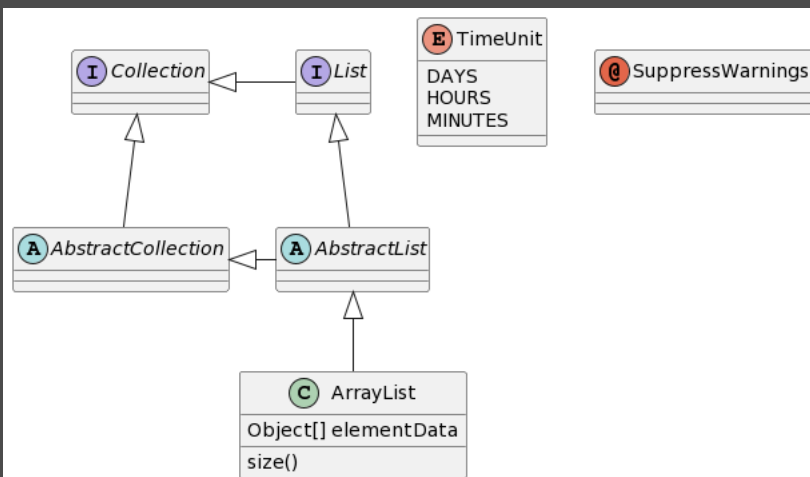
LiaScript erweitert Markdown um interaktive Elemente wie:

- Ausführbarer Code
- Animationen & Sprachausgaben
- Visualisierung
- Quizze & Umfragen
- Erweiterbarkeit durch JavaScript und Macros
- ...

Einbindung von PlantUML zur Generierung von UML-Diagrammen

Example.plantUML

```
1 @startuml
2
3 abstract class AbstractList
4 abstract AbstractCollection
5 interface List
6 interface Collection
7
8 List <|-- AbstractList
9 Collection <|-- AbstractCollection
10
11 Collection <|-- List
12 AbstractCollection <|-- AbstractList
13 AbstractList <|-- ArrayList
14
15 class ArrayList {
16     Object[] elementData
17     size()
18 }
19
20 enum TimeUnit {
21     DAYS
22     HOURS
23     MINUTES
24 }
25
26 annotation SuppressWarnings
27
28 @enduml
```



https://www.plantuml.com/plantuml/png/L0-_JWCn38TtFuL76Fe63Aqe4aX09QuDX1236mmAIIdnLx0pyuTtffLH99ik_xtCSBzKeM0u1W7PgYPoX2wIAg__srgjLTaelfn30cPy4l1YdDtACHQrju0s8czq7Etj-GGuSMMnDHeTO_HUVdSCl04kEkFMHH_77aVNgQJYKwytuCDUxc_jnUpNCBebCHkLdGzx14wi-KX8LxmgmP7dDCVm1

Ausführbarer C# Code

Wichtig für uns sind die ausführbaren Code-Blöcke, die ich in der Vorlesung nutze, um Beispielimplementierungen zu evaluieren. Dabei werden zwei Formen unterschieden:

C# 10 mit dotnet Unterstützung

Coderunner.cs9

```

1 using System;
2 using System.Collections.Generic;
3 using System.Collections;
4 using System.Linq;
5 using System.Text;
6
7 int n;
8 Console.WriteLine("Number of primes: ");
9 n = int.Parse(Console.ReadLine());
10
11 ArrayList primes = new ArrayList();
12 primes.Add(2);
13
14 for(int i = 3; primes.Count < n; i++) {
15     bool isPrime = true;
16     foreach(int num in primes) isPrime &= i % num != 0;
17     if(isPrime) primes.Add(i);
18 }
19
20 Console.WriteLine("Primes: ");
21 foreach(int prime in primes) Console.Write($" {prime}");
  
```

myproject.csproj

```

1 <Project Sdk="Microsoft.NET.Sdk">
2   <PropertyGroup>
3     <OutputType>Exe</OutputType>
4     <TargetFramework>net5.0</TargetFramework>
5   </PropertyGroup>
6 </Project>
  
```

You must install or update .NET to run this application.

App: /tmp/tmp_cpfy4ay/bin/Debug/net5.0/project
Architecture: x64
Framework: 'Microsoft.NETCore.App', version '5.0.0' (x64)
.NET location: /usr/lib/dotnet/dotnet6-6.0.110

The following frameworks were found:

6.0.10 at [/usr/lib/dotnet/dotnet6-6.0.110/shared/Microsoft.NETCore.App]

Learn about framework resolution:

<https://aka.ms/dotnet/app-launch-failed>

To install missing framework, download:

[https://aka.ms/dotnet-core-applaunch?](https://aka.ms/dotnet-core-applaunch?framework=Microsoft.NETCore.App&framework_version=5.0.0&arch=x64&rid=ubuntu.22.10-x64)

[framework=Microsoft.NETCore.App&framework_version=5.0.0&arch=x64&rid=ubuntu.22.10-x64](https://aka.ms/dotnet-core-applaunch?framework=Microsoft.NETCore.App&framework_version=5.0.0&arch=x64&rid=ubuntu.22.10-x64)

C# 8 mit mono

HelloWorld.cs

```
1 using System;
2
3 namespace HelloWorld
4 {
5     public class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Glück auf!");
10        }
11    }
12 }
```

Cannot open assembly 'main.cs': File does not contain a valid CIL image.

Frage: Welche Unterschiede sehen Sie zwischen C#8 und C#10 Code schon jetzt?

Quellen & Tools

- Das Projekt: <https://github.com/liascript/liascript>
- Die Webseite: <https://liascript.github.io>
- Nützliches
 - [Dokumentation zu LiaScript](#)
 - [YouTube Kanal zu LiaScript](#)
- Editoren
 - Für den [Atom](#)-Editor von GitHub existieren derzeit zwei Plugins:
 1. [liascript-preview](#)
 2. [liascript-snippets](#)
 - Für den Einsatz anderer Editoren eignet sich auch der [LiaScript-DevServer](#)

GitHub

Der Kurs selbst wird als "Projekt" entwickelt. Neben den einzelnen Vorlesungen finden Sie dort auch ein Wiki, Issues und die aggregierten Inhalte als automatisch generiertes Skript.

Link zum GitHub des Kurses: https://github.com/TUBAF-IfI-LiaScript/VL_Softwareentwicklung

Hinweise

1. Mit den Features von GitHub machen wir uns nach und nach vertraut.
2. Natürlich bestehen neben Github auch alternative Umsetzungen für das Projektmanagement wie das Open Source Projekt GitLab oder weitere kommerzielle Tools BitBucket, Google Cloud Source Repositories etc.

Entwicklungsumgebungen

Seien Sie neugierig und probieren Sie verschiedene Tools und Editoren aus!

- [Atom](#)

- [Visual Studio Code](#)

- [VIM/gVIM](#) / [neoVIM](#)

- weitere ...

Aufgaben

- ☐ Legen Sie sich einen GitHub Account an (sofern dies noch nicht geschehen ist).
- ☐ Installieren Sie einen Editor Ihrer Wahl auf Ihrem Rechner, mit dem Sie Markdown-Dateien komfortabel bearbeiten können.
- ☐ Nutzen Sie das Wiki der Vorlesung um Ihre neuen Markdown-Kenntnisse zu erproben und versuchen Sie sich an folgenden Problemen:
 - Recherchieren Sie weitere Softwarebugs. Dabei interessieren uns insbesondere solche, wo der konkrete Fehler direkt am Code nachvollzogen werden konnte.
 - Fügen Sie eine kurze Referenz auf Ihren Lieblingseditor ein und erklären Sie, warum Sie diesen anderen Systemen vorziehen. Ergänzen Sie Links auf Tutorials und Videos, die anderen nützlich sein können.