Einführung in die maschinennahe, imperative, funktionale, relationale und objektorientierte Programmierung

-

EMIFROP 0.25

Markus Alpers
B.Sc. und Ausbilder f. Industriekaufleute

29. Februar 2016

Inhaltsverzeichnis

Hinweis bezüglich diskriminierender Formulierungen

In diesem Text wurde darauf geachtet Formulierungen zu vermeiden, die diskriminierend verstanden werden können. Im Sinne der Lesbarkeit wurden dabei Formulierungen wie „Informatiker und Informatikerinnen“ durch „InformatikerInnen“ (mit großem i) ersetzt. An anderen Stellen habe ich Formen wie eine/einer durch eineR zusammengefasst. Hier berufe ich mich auf den Artikel „Sprache und Ungleichheit“ der Bundeszentrale für politische Bildung, kurz BpB, vom 16. April 2014, insbesondere auf den Absatz „Zum Umgang mit diskriminierender Sprache“, online abrufbar unter:

`http://www.bpb.de/apuz/130411/sprache-und-ungleichheit?p=all`

Sollten Sie dennoch Formulierungen entdecken, die diesem Anspruch nicht entsprechen, möchte ich Sie bitten, mir eine entsprechende Nachricht zu senden, denn es ist mir wichtig, Ihnen mit diesem Buch eine wertvolle Unterstützung beim Start in die faszinierende Welt der Informatik zu bieten. Das sollte nicht durch verletzte Gefühle in Folge missverständlicher Formulierungen torpediert werden.

Sie erreichen mich unter `markus.alpers@haw-hamburg.de`.

Hinweis zur Lizenz

zenz"´. Wenn Sie einen Teil finden, in dem diese „Hinweise zur Lizenz"´ nicht zu finden ist, dann dürfen Sie für den persönlichen Gebrauch dennoch Kopien davon anfertigen und Sie dürfen diese Kopien außerhalb von kommerziellen Projekten frei verwenden.

Hinweis zur Verwendbarkeit in wissenschaftlichen Arbeiten

Bitte beachten Sie dabei aber, dass die Verwendung dieses Textes im Rahmen wissenschaftlicher Publikationen zurzeit aus anderen Gründen problematisch ist: Wie viele andere Quellen, die frei im Internet verfügbar sind, wurde auch dieser Text bislang nicht durch einen nachweislich entsprechend qualifizierten Lektor verifiziert. Damit genügen Zitate aus diesem Band streng genommen noch nicht den Ansprüchen wissenschaftlicher Arbeiten.

Bitte beachten Sie außerdem, dass dieses Buch eine Konvention nutzt, die in wissenschaftlichen Arbeiten verpönt ist: Wenn in einer wissenschaftlichen Arbeit ein Begriff hervorgehoben wird, dann wird dazu kursive Schrift verwendet. In diesem Buch verwende ich dagegen Fettdruck, da es vielen Menschen schwer fällt, einen kursiv gedruckten Begriff schnell zu finden und ich mir wünsche, dass Sie es möglichst effizient auch als Nachschlagewerk nutzen können.

Teil I

Einführung in die Programmierung für alle Studierenden im Bereich MINT

(Mathematik, Ingenieurswissenschaften, Naturwissenschaften, Technik)

Kapitel 1

Typische Irrtümer darüber, was Programmieren ist.

Kapitel 2

Von der Nachrichtentechnik zur Programmierung

Kapitel 3

Vorbereitung fürs Programmieren

Im Grunde brauchen Sie zum Programmieren lediglich einen Texteditor sowie ein SDK bzw. einen Compiler oder einen Interpreter für die jeweilige Sprache. Ein Texteditor wird bei jedem beliebigen Betriebssystem mitgeliefert, während Sie sich SDK/Compiler/Interpreter häufig von der Seite des Entwicklers herunterladen können.

Sie werden im Folgenden des Öfteren die Bezeichnung **Werkzeug** (engl. **tool**) lesen. Beim Programmieren ist damit im Regelfall ein Programm gemeint, das Sie in irgendeiner Form bei der Softwareentwicklung unterstützt.

Da Sie bei der Suche nach Werkzeugen fürs Programmieren über eine Vielzahl an Begriffen stolpern werden, folgt hier eine auszugweise Aufstellung häufiger Bezeichnungen, sortiert nach Bedeutung:

- Mit **Quellcode** bezeichnet man den Programmtext, den Sie eingegeben haben.
- Ein **Interpreter** ist eine Software, der Ihren Quelltext Zeile für Zeile übersetzt und kontinuierlich jede übersetzte Zeile sofort ausführt.
- Ein **Compiler** ist eine Software, der das von Ihnen verfasste Programm vollständig übersetzt und dabei eine beschränkte Anzahl an Fehlertypen erkennen kann. Personen, die ein sehr beschränktes Verständnis von Programmierung haben sind deshalb häufig überzeugt, dass kompilierte Sprachen sicher, interpretierte Sprachen dagegen unsicher wären. Tatsächlich handelt es sich schlicht um zwei Paradigmen, die unterschiedliche Vor- und Nachteile haben. Und wenn wir von **IT-Sicherheit** sprechen, dann hat das nichts aber auch rein gar nichts mit kompilierten Sprachen zu tun.

- Ein Debugger ist eine Software, die Ihren Quelltext auf Fehler untersucht.
- Ein **SDK** (kurz für Software Development Kit bzw. Softwareentwicklungskit) ist eine Softwaresammlung, die neben einem Interpreter und/oder Compiler eine Reihe weiterer Programme beinhaltet, die Sie bei der Entwicklung von Programmen unterstützt. SDKs werden jeweils für eine bestimmte Sprache entwickelt. Gelegentlich wird auch der Begriff **Toolchain** verwendet. Im hier verwendeten Sinne entspricht eine Toolchain einem SDK.
- Eine **IDE** (kurz für Integrated Development Environment bzw. Integrierte Entwicklungsumgebung) ist ein Sammlung von Programmen, die meist unabhängig von einer bestimmten Sprache zum Programmieren nutzbar sind. Im Gegensatz zu einer SDK beinhalten IDEs in aller Regel einen eigenen Editor, der u.a. Syntaxerkennung bietet. Mehr noch: Viele IDEs sind so entwickelt, dass es möglich ist, eine Vielzahl von SDKs in ihnen zu nutzen und damit eine Vielzahl von Programmiersprachen in Ihnen zu nutzen.

Die nachfolgenden Werkzeuge gehören in den fortgeschrittenen Bereich, mit dem Sie im Rahmen dieses Buches nichts zu tun haben werden. Sie sollten Sie allerdings kennen, damit Sie wissen, wonach Sie später suchen müssen, um Probleme zu vermeiden, die sehr viel Zeit kosten können. Ihre Beherrschung unterscheidet professionelle Software Entwickler von Quereinsteigern.

- **Deployment Support/Tools** sind Werkzeuge, die Sie dabei unterstützen, ein Programm zu verteilen. Stellen Sie sich dazu vor, dass Sie ein Programm entwickeln, das von allen 15.000 Mitarbeitern Ihres Unternehmens genutzt wird. Ohne Deployment Support/Tool müssten Sie nun an jeden Arbeitsplatz gehen, die alte Version deinstallieren, u.U. den Rechner neustarten und dann die neue Version einspielen. Und wenn Sie fertig sind, gehen Sie in Rente. Mit Deployment Support/Tools dagegen lassen Sie lediglich die Änderungen (auch als **Delta**s bezeichnet) automatisiert von einem zentralen Server aus an die Rechner verteilen und dort installieren.
- **Refactoring** bezeichnet eine Überarbeitung einer Software. Hier geht es aber nicht darum, einzelnen Fehler zu beheben, sondern darum, strukturelle Änderungen in eine Software einzuarbeiten. (Hier wären wir bei einem weiteren Aspekt des Software Engineering angelangt.)

Die folgenden Abschnitte enthalten jeweils die Informationen, die sie zur Vorbereitung der Programmierung auf einem Windows-Rechner benötigen,

da auf den Rechnern unseres Departments Windows zum Einsatz kommt. Für die Linux- und MacOS-User unter Ihnen gibt es hier leider keine bzw. nur wenige Hinweise. Nutzen Sie bitte über die üblichen Quellen im Netz.

3.1 Assembler und C – Vorbereitung für die maschinennahe und imperative Programmierung

In den Veranstaltungen zur maschinennahen und imperativen Programmierung unseres Departments kommen zurzeit **ARM Cortex-M0 Prozessoren** zum Einsatz. Bei diesen Prozessoren können Sie von der Webpage von IAR ein kostenloses SDK herunterladen. Im Rahmen der Veranstaltung Informatik 3 (für Media Systems) erhalten Sie außerdem Informationen darüber, wie Sie Ihr System einrichten können und wie Sie die dort verwendete IDE beziehen können. Diese stammt von der Fa. Keil, heißt µVision und wird auch als **MDK** bezeichnet. Weiterhin benötigen Sie für die Entwicklung von Software für einen ARM Prozessor ein Entwicklerboard, auf dem der Prozessor bereits montiert ist, sowie eine debugging Unit. In beiden Fällen handelt es sich um Hardware, die Sie im Labor für Ihre Versuche erhalten.

Wenn Sie so lange nicht warten wollen und wesentlich weniger Geld ausgeben wollen, um sich in die Programmierung von ARM-Prozessoren einzuarbeiten, dann möchte ich Ihnen einen Computer mit ARM-Prozessor empfehlen, der von der Universität in Cambridge extra für Studienanfänger entwickelt wurde. Es handelt sich um den **Raspberry Pi** (kurz RasPi oder Pi), der in der Version 2 für rund 45,- € zu haben ist. (Maus und Tastatur sowie eine MicroSD-Card als Laufwerk und ein passendes Ladekabel fehlen hier noch, aber für knapp 70,- € sollten Sie ein komplettes System bekommen.) Das praktische am RasPi ist, dass Sie ihn in die Tasche stecken können.

Wenn Sie dagegen die maschinennahe Programmierung auf einem Intel- oder AMD-Prozessor mit IAx86-Architektur[1] durchführen wollen, können Sie sich die **GCC** (kurz für **GNU Compiler Collection**) kostenlos herunterladen und installieren. Hierbei handelt es sich um eine Sammlung von Compilern für die verschiedensten Sprachen. Im Gegensatz zu vielen anderen Compilern erhalten Sie die GCC unter der sogenannten **GNU GPL** (Lizenz). Kurz gesagt bedeutet das, dass Sie hierauf kostenfreien Zugang haben, dass die Entwickler Ihnen darüber hinaus aber noch wesentlich mehr Rechte einräumen. Die GCC sind in verschiedenen Paketen enthalten, die

[1] Das sind die Prozessoren, die zurzeit üblicherweise in Desktop-Rechnern verbaut werden.

eine etwas komfortablere Installation erlauben.

Windows-Nutzern sei hier das **MinGW** empfohlen, das zusätzlich zum GCC auch ein minimalistisches GNU installiert. Mit letzterem können Sie unter Windows auf der Konsole wie in einer Linux-Umgebung arbeiten. Alternativ können Sie auch auf Microsofts **Visual Studio** Professional oder Ultimate zurückgreifen, das Sie als Studierende kostenlos über das Dreamspark Programm erhalten. Davon gibt es noch die Community Edition, die grundsätzlich kostenlos von Microsoft angeboten wird. Beachten sie aber bitte, dass Sie in diesem Fall unter Umständen Software entwickeln, die ausschließlich auf Windows Rechnern lauffähig ist. Für den Aufruf des GCC-Assembler Compilers müssen Sie abschließend noch die PATH-Variable von Windows um das /bin-Verzeichnis Ihrer MinGW-Installation erweitern.

MacOS-Nutzer brauchen eine derart umfangreiche Software nicht, da Ihr Betriebssystem bereits die entsprechenden Werkzeuge an Bord hat. Allerdings müssen Sie immer noch einen Assembler Compiler installieren, wozu Sie am besten ebenfalls auf die GCC zurückgreifen. Einsteiger werden zuvor noch **XCode** installieren müssen. Dabei handelt es sich um eine IDE, die Sie über den App Store herunterladen können. Suchen Sie anschließend im Netz nach einer Anleitung, wie Sie die GCC in XCode einbinden können. Unter MAC OS X wählten Sie 2010 im XCode
`Menü -> Preferences -> Downloads` den Eintrag „Command Line Toolsünd hatten nach dem Abschluss des Downloads die GCC vollständig installiert. Die Installation sollte also auch Einsteigern problemlos möglich sein. Genau wie bei Microsofts Visual Studio sollten Sie allerdings daran denken, dass Sie bei der Nutzung von XCode unter Umständen Software entwickeln, die ausschließlich auf Apple-Rechnern lauffähig ist.

Warum es hier keine Informationen für **Linux**-User gibt? Weil Linux-User im Regelfall selbständig genug sind, um sich diese Informationen selbst zu beschaffen. Sollten Sie Linux-Neuling sein, empfehle ich Ihnen die Linux Einführung auf der Plattform edX. Dieser Kurs kann wie die meisten Kurse auf edX auch kostenlos belegt werden und wird im Gegensatz zu den meisten Kursen kontinuierlich angeboten; Sie sind hier also nicht an einen bestimmten Zeitraum gebunden. Er wurde von der FSF entwickelt und ist nach anfänglichem Marketing für die FSF eine fundamentale und sehr nützliche Einführung in das offene und freie Betriebssystem. Leider kommt GNU ein wenig zu kurz, aber Sie erhalten zumindest die entsprechenden Links.

Wenn Sie einen RasPi erworben haben, dann gibt es eine Einführung in Linux im sogenannten **Raspberry Pi Educational Manual**

http://vx2-downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf, das zwar für die erste Version des RasPi entwickelt wurde, aber auch bei der aktuellen Version genutzt werden kann, da diese weitgehend abwärtskompatibel ist. Darin ist neben der Einführung in die Grundlagen verschiedener Arten der Programmierung auch eine Einführung in die Administration von Linux enthalten. Die sollten Sie als RasPi-Nutzer auf jeden Fall durcharbeiten.

Außerdem sollten Sie noch ein gutes Lehr- und Nachschlagewerk für die Assemblerprogrammierung erwerben. Hier gibt es allerdings auch einige Bände, die Sie legal kostenlos herunterladen dürfen. Grundsätzlich möchte ich Ihnen für diese Fälle die folgende Seite ans Herz legen:
http://hackershelf.com/topics/
Suchen Sie hier bitte unter dem Suchbegriff **assembly**, nicht assembler, da im Englischen die Bezeichnung nicht Assembler Language sondern Assembly Language lautet.

Persönlicher Tipp: Greifen Sie dort zu „**Programming from the Ground up**"von Jonathan Bartlet.

Sie wollen wissen, was mit GNU gemeint ist? Gute Frage. Aber die Antwort lautet: Lesen Sie es selbst nach: https://www.gnu.org/ Es geht hier letztlich um eine der wichtigsten Institutionen, die Sie als angehende InformatikerInnen kennen sollten: Die **FSF** (kurz für Free Software Foundation). Ob nun die **GI** (kurz für Gesellschaft für Informatik), **IEEE**, die FSF oder **ITU-T** die absolute Spitzenposition bezüglich der Bedeutung für InformatikerInnen einnimmt, kann ich Ihnen nicht sagen, aber sie sind allesamt außerordentlich wichtig. Beschäftigen Sie sich deshalb damit.

Ergänzungen zu C und C++

Leider befand sich zum Zeitpunkt, als diese Zeilen geschrieben wurden noch kein für Einsteiger empfehlenswertes Buch auf hackershelf.com. Allerdings gibt es für die Programmierung in C ein Standardwerk, das u.a. vom Entwickler der Sprache verfasst wurde: „**The C Programming Language**"von **Ritchie** und **Kernighan**.

3.2 Java - Vorbereitung für die Programmierung in Java

Wichtig für Wiederholer: Stellen Sie sicher, dass Sie auf jedem Rechner die gleiche Version von Java installiert haben. Zwar ist Java weitgehend abwärtskompatibel, aber die Entwickler der Sprache erklären immer wie-

der einzelne Bestandteile für veraltet. Wenn Sie dann noch eine alte JDK verwenden, bedeutet das, dass Sie ggf. aktuellen Code nicht programmieren können und veraltete Vorgehensweisen vom Compiler akzeptiert werden. Gerade bei Hausaufgaben führt das dann schnell zu Problemen.

Für Java gibt es zunächst zwei Anbieter von SDKs. Zum einen können Sie eines der offiziellen SDKs von Oracle herunterladen, zum anderen gibt es das OpenJava, das unter einer anderen Lizenz entwickelt wird.

Auf der Webpage des Entwicklers Oracle gibt es Java2SE, Java2EE, Java for Mobile und noch ein gutes Dutzend weiterer Java Downloads. Das was Sie für den Einstieg brauchen ist **Java2SE**. Zum Grund dafür kommen wir am Ende dieses Abschnitts, für die Installation brauchen Sie es nicht zu wissen.

Nachdem Sie sich mit den Nutzungsbedingungen einverstanden erklärt haben, können Sie auf eine Reihe an Downloads zugreifen, die sich dadurch unterscheiden, ob Sie nun einen 32- oder 64-Bit-Prozessor haben, welches Betriebssystem Sie nutzen und ob Sie die Installationsdatei vollständig (offline) oder nur zum geringen Teil (für eine online-Installation) herunterladen wollen. Wenn Sie sich die Zeit nehmen, genau hinzusehen, dann sollten Sie hier die für Ihr System optimale Version wählen können. Im Grunde ist es bei Windows, Linux oder MacOS aber egal, welche Version (32/64 Bit bzw. online/offline) Sie wählen, so lange das Betriebssystem und der Prozessor stimmen.

Einen Buchtipp für die Programmierung in Java kann an dieser Stelle nicht gegeben werden, was auch daran liegt, dass mit jeder neuen Version wieder essentielle Änderungen in die Sprache eingeführt werden. Dazu kommt, dass Java durch seinen Middleware-Charakter ohnehin derart umfangreich ist, dass kein Buch existieren kann, das auch nur größtenteils vollständig ist. Dazu ist das Framework schlicht zu umfangreich. Sie glauben mir nicht? Dann werfen Sie einen Blick in die Java API; das ist die vollständige Java Dokumentation: `http://docs.oracle.com/javase/8/docs/api/` Und dort sind lediglich all die Klassen enthalten, die von Oracle selbst angeboten werden. Am besten setzten Sie in Ihrem Browser auch gleich ein Lesezeichen auf diese Seite, denn beim Lernen von und Arbeiten mit Java werden Sie wohl nichts so oft brauchen wie diese Seite. Sollte inzwischen die nächste Version von Java veröffentlicht worden sein, dann ändern Sie bitte die entsprechende Ziffer im Link ab.

Bei dieser Gelegenheit lernen Sie auch schon die nächste wichtige Abkürzung kennen: **API** steht kurz für **Application Programming Interface**, was als **Programmierschnittstelle** übersetzt wird. Eine API galt als ein typisches Kennzeichen objektorientierter Programmiersprachen, doch das ist falsch:

Eine Vielzahl an Lösungen wurden bereits von anderen Entwicklern programmiert und intensiv getestet, sodass Sie sich darauf konzentrieren können, die Spezialfälle zu programmieren, die bei Ihrer Software auftreten. Tatsächlich finden Sie so etwas jedoch selbst für Assembler. Der Begriff **Klassenbibliothek** wird des Öfteren als Synonym für eine API verwendet. An sich steht dieser Begriff aber allgemeiner für Sammlungen von Klassen, die Lösungen für häufig auftretende Probleme beinhalten.

Da der Begriff der **Klasse** untrennbar mit Java verbunden ist, hier eine kurze Erklärung, die zwar viel zu simpel ist, aber für den Anfang genügen soll: In Java bestehen Programme wie in alle objektorientierten Programmiersprachen aus sogenannten Modulen, die in Java als **Package**s und Klassen bezeichnet werden. Jedes dieser Module beinhaltet einen Programmteil, der für sich genommen bereits ein einen sinnvollen Teil des Gesamtprogramms erfüllen kann. Um bei besonders großen Projekten mehr Übersicht zu erhalten werden dort mehrere Klassen in einem Package gesammelt. Packages können Sie sich in Java wie Verzeichnisse bei einem Betriebssystem vorstellen. Genau wie dort kann ein Package mehrere Packages enthalten, die sowohl Klassen als auch Packages enthalten können.

Anschließend müssen Sie u.U. unter Windows (ähnlich wie bei der Vorbereitung für die maschinennahe Programmierung) noch die PATH-Variable erweitern. In diesem Fall müssen Sie dort das Verzeichnis angeben, in dem u.a. die Datei javac liegt. Prüfen Sie anschließend über den Befehl javac –v in der Eingabeaufforderung, ob Sie die PATH-Variable richtig gesetzt haben. Anmerkung für einen späteren Umstieg auf Linux: Dort heißt die „Eingabeaufforderung“ **Terminal**, **Konsole** oder **shell**, wird aber auch als bash, sh, z-sh und ähnliches bezeichnet, wobei das teilweise konkrete Programmnamen sind, die ein Terminal bzw. eine Konsole starten.

Jetzt aber zur Antwort auf die Frage, was denn die verschiedenen Java-Versionen unterscheidet. Hier werden wir nicht zu sehr ins Detail gehen, daher nur so viel:

- Warum steht da eine 2 in Java2SE?
  Obwohl mit jeder neuen Version (Java ist im Moment, in dem diese Zeilen geschrieben werden bei Version 8 angelangt) essentielle Änderungen an der Sprache durchgeführt werden, entschieden sich die Entwickler nach Version 2 die Zahl 2 im Namen zu behalten. Das ist alles.

- Wir brauchen ja das SDK. Aber was ist die JRE?
  Sie wissen bereits, dass Sie ein Softwareentwicklungskit zum Entwickeln von Software nutzen können. Dieses beinhaltet eine Laufzeit-

umgebung (in diesem Fall die JRE, kurz für Java Runtime Environment), die dazu dient, Java Programme auf einem Computer laufen zu lassen, auf dem kein JDK installiert ist.
Laufzeitumgebungen gibt es für eine Vielzahl von Programmiersprachen. Es handelt sich dabei um so etwas wie eine Übersetzungssoftware, die es dem Betriebssystem ermöglicht, Programme in der jeweiligen Sprache auszuführen.
Und richtig verstanden: Kein Programm wird direkt von einem Betriebssystem ausgeführt, sondern das Betriebssystem startet gegebenenfalls die Laufzeitumgebung für eine Sprache, in der das zu startende Programm dann ausgeführt wird. Ein häufiger Irrtum von Nutzern besteht dagegen darin, anzunehmen, dass beispielsweise unter Windows Dateien mit der Endung .exe von Windows selbst ausgeführt werden; vielmehr gilt auf für Dateien im .exe-Format das gleiche, was für alle Programm gilt, die ausgeführt werden sollen: Es muss auf dem Rechner eine Laufzeitumgebung (oder ein Interpreter) für die entsprechende Sprache installiert sein, sonst kann das Programm nicht ausgeführt werden. Einzige Ausnahme: Ein Programm liegt in Maschinensprache vor. Dann und nur dann kann es direkt vom Prozessor ausgeführt werden.

- Was bedeutet dieses SE in Java2SE? Was ist dieses Java2EE? Und was sollen all die anderen Versionen?
  - **Java SE** (kurz für Standard Edition) ist gewissermaßen der Kern der Sprache/der Middleware. Es enthält alles, was Sie benötigen, um Java Programme zu entwickeln bzw. um sie auf einem Rechner laufen zu lassen.
  - **Java EE** (kurz für Enterprise Edition) ist eine erweiterte Fassung von Java SE, die in Unternehmen eingesetzt werden kann, bei denen es wenigstens einen zentralen Server gibt, über den die Kommunikation von Java Anwendungen koordiniert wird.
  - **Java ME** (kurz für Micro Edition) ist dagegen eine höchst effiziente Version, die auf Geräten mit geringen Kapazitäten zum Einsatz kommen kann. (Alte Rechner, Smartphones, usw.)
  - **Java FX** (kurz für Effects) beinhaltet Klassen, mittels derer Internetapplikationen entwickelt werden können.

Nachdem Sie das JDK heruntergeladen und installiert haben, brauchen Sie noch eine Möglichkeit, um Java Programme einzugeben oder zu ändern, denn das JDK enthält keinen eigenen Editor. Hier genügt für die ersten Schritte ein **einfacher Texteditor**. Verwechseln Sie das bitte nicht mit einer

Textverarbeitung: Ein Texteditor zeigt Ihnen den eingegebenen Text standardmäßig ohne irgendwelche Hervorhebungen wie Fettdruck und ähnliches. Im Gegensatz dazu zeigt Ihnen eine Textverarbeitung Texte mit Hervorhebungen, bei denen dann der Teil des Quelltexts ausgeblendet wird, über den diese Hervorhebungen erzeugt werden.

Standardmäßig hat jedes Betriebssystem mindestens einen Texteditor an Bord. Bei Windows finden Sie diesen unter Zubehör bzw. indem Sie bei Windows 8 in der „Kachelansichtßchlicht das Wort Editor eintippen. Bei älteren Windows-Versionen können Sie das Suchfenster im Windowsmenü nutzen.

Als ersten Schritt zu mehr Komfort gibt es Texteditoren mit **Syntaxerkennung**. Beispielsweise den Notepad++. Solche Texteditoren heben automatisch syntaktische Fehler hervor, indem sie z.B. eine rote Linie unterhalb fehlerhaften Stellen anzeigen. Sie ändern dabei aber im Gegensatz zu Textverarbeitungen nichts am Quellcode.

Fortgeschrittene werden Ihnen jetzt empfehlen, doch sofort zu einer IDE wie **Eclipse** zu greifen, da die doch so viel besser zum Programmieren seien. Doch bitte lassen Sie das vorerst: Wie schon zuvor geschrieben haben selbst kostenlose IDEs inzwischen einen derartigen Umfang an Komfortfunktionen, dass sie für Einsteiger eher verwirrend als hilfreich sind. Es ist aber richtig: Nach spätestens sechs Monaten sollten Sie auf jeden Fall damit beginnen, eine IDE zu nutzen.

3.3 HTML, PHP und MySQL – Vorbereitung für die Entwicklung verteilter Anwendungen

Wenn Sie eine Webpage entwickeln wollen, dann benötigen Sie dazu folgende Dinge:

1. Einen einfachen Texteditor wie **Notepad++**.
2. Ein Softwarepaket, das Ihnen die nötige Infrastruktur für eine webbasierte Anwendung bietet. Beispiele: **XAMPP** oder **EasyPHP**
3. Eine Software, die es Ihnen ermöglicht, Dateien auf einen Webserver zu übertragen. Ein einfaches kostenloses Programm ist **FileZilla**.

Im Gegensatz zur imperativen Programmierung müssen Sie sich hier nur um wenige Dinge kümmern, obwohl hier tatsächlich wesentlich mehr Installationsschritte zu erledigen sind, als das bei C, C++ oder Java der Fall ist. Denn all die nötigen Schritte übernimmt hier die Installationsroutine

von XAMPP bzw. EasyPHP.

Da Sie alle drei Programme im Netz legal kostenlos beziehen können und die Installation keine fortgeschrittenen Kenntnisse erfordert, sei dazu an dieser Stelle nichts weiter gesagt.

Sie sollten allerdings noch wissen, welche Sprachen und Strukturen hier zum Einsatz kommen. Wie Sie zu Beginn des Kapitels über verteilte Anwendungen erfahren werden, werden Sie diese Punkte später in Veranstaltungen wie „Einführung in relationale Datenbankenäusführlich kennen lernen.

1. Zunächst benötigen Sie für eine Webanwendung einen Server, der die Daten der Seite vorhält und über das Internet erreichbar ist.
   An dieser Stelle empfehle ich Ihnen die Installation von EasyPHP. EasyPHP wird (genau wie XAMPP) Ihren Rechner so vorbereiten, dass der Apache Server auf Ihrem Rechner läuft und als Webserver genutzt werden kann. Aus Sicherheitsgründen sollten Sie diesen Server aber nur so lange laufen lassen, wie Sie ihn zum Testen benötigen. Denn wenn Sie keine fortgeschrittenene Kenntnisse in IT-Sicherheit und Serveradministration haben, werden Sie sonst mit höchster Sicherheit Sicherheitslücken auf Ihrem Rechner einrichten.

2. Dann benötigen Sie eine Sprache, mittels derer Sie die Elemente Ihrer Webanwendung definieren können. Über diese Sprache regeln Sie also, wie die einzelnen Bestandteile der Webpage zu einzelnen Ansichten verbunden werden. Hier greifen wir zu **HTML**, für das Sie im Gegensatz zu C, C++ oder Java keinen Compiler oder Interpreter installieren müssen: Den HTML-Interpreter finden Sie bereits in Form eines Webbrowsers auf praktisch jedem Rechner vor.
   HTML hat allerdings einen entscheidenden Nachteil für unsere Zwecke: Die Sprache ist durchgehend statisch. Wenn Sie also Änderungen auf der Seite einführen wollen, die beispielsweise von den Eingaben des Nutzers abhängen, dann können Sie das mit HTML alleine nicht erreichen.

3. Deshalb folgt als nächstes eine imperative Sprache, wobei wir hier **PHP** wählen, das im Gegensatz zu C und ähnlichen dynamisch typisiert ist. Sie werden sich erinnern: Das bedeutet, dass der Interpreter der Sprache vieles für Sie übernimmt, dass Sie also weniger Details selbst programmieren müssen. Dafür müssen Sie aber umso genauer bzw. umso mehr darüber Bescheid wissen, wie der Interpreter genau arbeitet.

PHP wird wird von vielen Webservern interpretiert. In unserem Fall übernimmt das der Apache Server, der Teil der Easy-PHP- bzw. der XAMPP-Installation ist.
Mittlerweile wird allerdings JavaScript mehr und mehr zu einem ernstzunehmenden Konkurrenten von PHP, weil es wesentlich mehr Möglichkeiten bietet und eine Vielzahl Beschränkungen dort nicht existieren. In HTML5 ist es als Standard für die Programmierung der Funktionalität einer Webanwendung eingestellt, sodass Sie hier zum Testen nicht unbedingt einen Webserver benötigen.

4. Außerdem brauchen wir noch eine Möglichkeit, um Daten langfristig zu speichern. Denn der Apache Server stellt zwar unsere Webpage bereit, HTML realisiert mit CSS die Darstellung und PHP realisiert die dynamische Nutzbarkeit der Anwendung, aber wenn wir nur diese Sprachen nutzen, können wir Nutzereingaben nicht dauerhaft speichern und wieder verwenden.
   Gerade wenn wir eine Webanwendung entwickeln, über die wir Verträge abschließen wollen (egal, ob es dabei um ein browserbasiertes Spiel, einen Webshop oder was auch immer geht), genügt das nicht. Und das führt uns direkt zu einer Datenbank. Datenbanken werden über eigene Sprachen angesprochen, die häufig ein SQL vorkommt. SQL steht kurz für **Structured Query Language**, übersetzt sind das also standardisierte Abfragesprachen. Und hier ist **MySQL** die Sprache, die wir im Kurs verwenden.
   Aber auch hier brauchen Sie wieder keinen Interpreter oder Compiler herunterladen; diese Aufgabe übernimmt bereits XAMPP bzw. EasyPHP für Sie.

Trotz der komfortablen Installation z.B. durch EasyPHP müssen Sie sich all diese Komponenten merken, weil Sie bei einer professionellen Entwicklung all diese Komponenten selbst installieren und konfigurieren müssen. Und schließlich geht es im Studium darum, dass Sie sich auf eine professionelle Tätigkeit vorbereiten.

3.4 Alle - Vorbereitung für die Teamarbeit

Wie im ersten Kapitel aufgeführt ist die Arbeit an einer Software heute eine Aufgabe, die in Teams durchgeführt wird. Und hier müssen Sie Werkzeuge nutzen, die über einen Server koordiniert werden, um auch nur ansatzweise effizient zu sein. Wenn sie dagegen die Vorstellung haben, Ihren Teil der Software zu entwickeln, dann eine Kopie davon an Ihre Teamkollegen zu verschicken und sich anschließend die Kopien der Arbeitsergebnisse der

Kollegen zu besorgen, dann sollten Sie sich schleunigst von diesem Dilletantentum verabschieden! So arbeiten ausschließlich Personen, die nichts aber auch gar nichts von Softwareprojekten verstehen.

Die aktuell effizienteste Möglichkeit, um ein solches Softwareprojekt über eine SCM zu verwalten heißt **Git**. Und diese Möglichkeit ist nicht nur außerordentlich effizient und praktisch, sondern obendrein noch kostenlos. Deshalb werden wir Sie hier von Anfang an einsetzen, damit Sie am Ende Ihres Studiums nicht einmal ansatzweise auf die Idee verfallen, Änderungen an Softwareprojekten per Kopie zu verteilen.

Im Arbeitsleben werden Sie allerdings häufig auf eine ältere Form des SCM treffen. Die Bezeichnung dafür lautet **SVN** bzw. **Subversion**. Subversion hat deutliche Nachteile gegenüber Git, ist aber immer noch weit verbreitet, weil viele Nutzer schlicht nicht willens sind, sich in die Änderungen einzuarbeiten, die mit einem solchen Umstieg verbunden sind. Media Systems Studierende sollten sich deshalb in jedem Fall später (z.B. in der Veranstaltung Software Engineering) auch in Subversion einarbeiten.

Wie schon im ersten Kapitel beschrieben besteht der große Vorteil von Git darin, dass Sie auf das Repository keinen Zugriff haben müssen, um an der Software zu arbeiten; Subversion geht im Kern davon aus, dass Sie sich in einem Unternehmensnetzwerk befinden und deshalb ständigen Zugriff auf das Repository haben.

Neben Git gibt es noch eine Webplattform mit dem Namen **GitHub**. Git ist an Software das einzige, das Sie installieren müssen, um ein SCM zu beginnen. Für die Speicherung des Repository können Sie dann einen beliebigen Rechner nutzen, so lange dieser über das Internet oder ein anderes Netzwerk regelmäßig erreichbar ist. GitHub selbst ist ein Service, der Ihnen anbietet, dass Sie dort Ihre Repositories lagern. Ähnlich wie andere Cloud-Dienste gilt auch hier, dass Sie bei einem kostenlosen Account keine privaten Repositories erhalten. U.U. gibt es hier aber für Studierende Sonderangebote.

Wichtig: Sollten Sie GitHub (oder einen ähnlichen Dienst) nutzen und Ihre Repositories öffentlich sein, dann bedeutet das auch, dass Ihre Projekte für jedermann/-frau frei zugänglich sind. Nutzen Sie dagegen Git mit einem Server, für den Sie Zugangsbeschränkungen erlassen können, dann gilt das natürlich nicht (automatisch).

Das praktische bei Git ist, dass Sie gar keinen Server für die Repositories brauchen, um anzufangen: Sobald Sie die Software heruntergeladen (`https://git-scm.com/`) und installiert haben, können Sie jedes be-

liebige Verzeichnis unter die Versionskontrolle stellen und können damit arbeiten, als gäbe es bereits ein Repository. Sobald Sie den nötigen Web- bzw. Cloudspace haben, können Sie Ihr/e Projekt/e dort in ein Repository einbinden, bzw. aus ihrem/n Projekt/en heraus ein Repository anlegen.

Die Einführung in die Arbeit mit Git auf der Webpage git-scm.com ist sehr gut, weshalb an dieser Stelle keine weiteren Erläuterungen erfolgen. Nur so viel: Arbeiten Sie sich dort ein, damit Sie von Beginn an damit arbeiten können.

3.5 Alle – Nutzung des Netzlaufwerks zur Speicherung eigener Daten

Leider wissen viele Studierende nicht, dass ihnen in der Hochschule ein Netzlaufwerk zur Verfügung steht, auf dem Sie alle Daten speichern können und speichern deshalb Ihre Ergebnisse „auf dem Rechnerän dem Sie in der Hochschule sitzen. Dabei werden die Daten aber gerade nicht auf dem Rechner gespeichert, an dem sie sitzen, sondern in ihrem Rechnerprofil. Und jedes Mal, wenn Sie sich in der Hochschule an einem Rechner anmelden wird dieses Profil über das Netzwerk auf den Rechner übertragen, an dem Sie sich anmelden. Wir hatten schon Studierende, die deshalb eine halbe Stunde und länger an ihrem Rechner saßen, bis sie endlich den Desktop sahen. Aber das liegt nicht etwa daran, dass die Rechner oder das Netzwerk langsam wären, sondern einzig und alleine daran, dass diese Studierenden zum Teil 30 und mehr Gigabyte an Daten in Ihrem Rechnerprofil gespeichert hatten. Nutzen Sie deshalb bitte für alle Arbeiten am Rechner Ihr Netzlaufwerk. Denn die Daten, die dort gespeichert werden werden erst dann auf Ihren aktuellen Rechner übertragen, wenn Sie sie benötigen. Und keine Sorge: Die Daten werden nicht über das WLAN übertragen, sind also im Regelfall so schnell auf Ihrem Rechner, als hätten Sie sie auf einem USB-Stick gespeichert.

An dieser Stelle auch ein Hinweis auf die Nutzung des WLANs: Leider verstehen viele Studierende nicht, dass jeder Aufruf einer Webpage eine gewisse Datenmenge über das Netzwerk überträgt, über das sie mit dem Internet verbunden sind. Und wenn nun fünfzig Studierende gleichzeitig ein Dutzend YouTube-Videos starten oder den neuesten Client für World of Warcraft herunterladen, dann bedeutet das eben, dass das Netz bis an die Grenze ausgelastet ist. Das verstehen Sie nicht? Nun, es ist genau wie im Straßenverkehr: Wenn jede/r mit dem Auto zur Arbeit fährt, dann kommt eben keine/r mehr richtig voran. Und dieser Vergleich entspricht genau dem, was beim Surfen im Netz das Problem ist: Die meisten Nutzer verhalten sich, als wenn sie alleine auf der Welt wären und als wenn Ressour-

cen unbegrenzt vorhanden wären. Wenn es dann einen Engpass gibt, dann heißt es immer: Das Netz ist schlecht, die Stadt soll mehr Straßen bauen, usw. usf. Doch die Ressourcen sind begrenzt. Ja, das gilt auch in den LTE-Netzen, egal was der Verkäufer Ihres Handy-Providers Ihnen versprochen hat.

Stichwortverzeichnis