

Softwareentwicklung 1

Dr. Herbert Prähofer
Institut für Systemsoftware
Johannes Kepler Universität Linz



Vortragender

Dr. Herbert Praehofer

**Institute for System Software
Johannes Kepler University
Altenbergerstrasse 69
A-4040 Linz / Austria**

**Tel.: ++43 (732) 2468 7132
Fax.: ++43 (732) 2468 7138
EMail: herbert.praehofer@jku.at
http: www.ssw.uni-linz.ac.at**

Raum: HF 306 (Hochschulfondgebäude 3. Stock)



Ziel und Inhalt

Ziel Einführung des Programmieren in Java

- Inhalte**
- Grundlagen der Programmierung
 - Programmieren in Java
 - Einfache Algorithmen
 - Grundlagen der Objektorientierung
 - Programmentwurf



Motivation programmieren zu lernen

- Software erstellen können
 - Grundlagen der Programmierung verstehen
 - Java kennen
 - komplexe Software entwerfen
 - sich neue Gebiete der Softwaretechnik erarbeiten können
- Software verstehen
 - wie arbeitet Software
 - was kann Software leisten
 - wie sind Programme aufgebaut
 - wie wird Software erstellt
- Kreativität erleben
 - Programmieren kann viel Spaß machen
 - Programmieren ist eine Herausforderung



Organisation der LVA Softwareentwicklung 1

- Vorlesung
 - Stoff wird theoretisch vorgetragen
 - Mit eingebauten Programmierbeispielen
- Übung
 - Stoff wird geübt
 - Programmieraufgaben sind wöchentlich zu programmieren
 - 2 Unterrichtsstunden für
 - Wiederholung
 - Praktische Beispiele
 - Besprechung der Programmieraufgaben
 - Übungsleiter: Hr. DI Peter Hamader

**Programmieren ist eine Fähigkeit,
die nur mit viel Übung erlernt werden kann!!**



Webpage der LVA

<http://www.ssw.uni-linz.ac.at/Teaching/Lectures/UZR/SWE1/2008W/index.html>

- Alle wichtigen Infos wie
 - Termine
 - Unterlagen
 - Studienmaterialfindet man auf dieser Seite



- P. Mössenböck, Sprechen Sie Java?, 2. Auflage, dpunkt.verlag, 2005 (Lehrbuch zur VL!)
- G. Krüger, GoTo Java 2, Addison-Wesley, 2000
On-Line Buch für Java Online Version hier Download (www.javabuch.de)
- M. Campione, K.Walrath: The Java Tutorial Online Edition (www.javasoft.com/tutorial)
- Java-Einführungskurs (<http://www.boku.ac.at/javaeinf/jein.html>)



Einheit 1: Grundlagen der Programmierung

Programmierung und Ausführung von Programmen

Algorithmen und Algorithmdarstellung

Programmiersprachen und Java

Das Java-Entwicklungssystem



Worum geht es?

Programmieren

Problem so exakt beschreiben, dass es ein Computer lösen kann

- ☞ kreative Tätigkeit
- ☞ Ingenieurtätigkeit
- ☞ Nur wenige Leute können gut programmieren

Programm = Daten + Befehle



Variablen

Programme arbeiten mit Variablen.
Sind benannte Behälter für Werte.



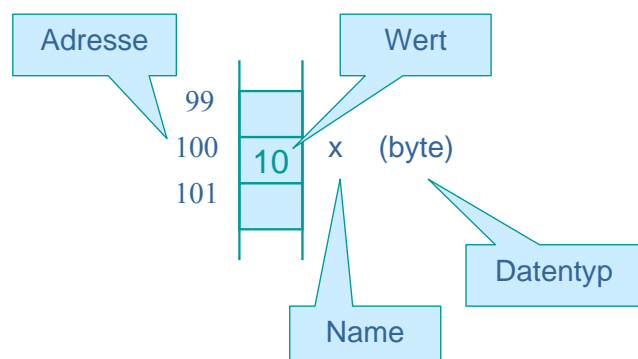
Variablen können ihren Wert ändern



Beispiel: byte x = 10







Charakterisiert durch

- Name oder Bezeichner
- Wert
- Adresse (im Hauptspeicher)
- Datentyp
 - int, char, byte, float, ...



Datentypen

Variablen haben einen Datentyp == Menge erlaubter Werte

Variablentyp	Werte	
 Zahl	 17  54 ...	Typ \cong Form - in eine Zahlenvariable passen nur Zahlen - in eine Zeichenvariable passen nur Zeichen
 Zeichen	 'a'  'x' ...	

Beispiele von Datentypen:

- char (2 Byte) [a-z][A-Z][0-9][Sonderzeichen]
- int (4 Byte) ~-2,147 Mrd. – ~+2,147 Mrd.
- boolean true, false
- float 3.14159265259



Algorithmus

Ein Algorithmus ist ein **endliches, schrittweises** Verfahren zur **Berechnung gesuchter aus gegebenen Größen**, in dem jeder Schritt aus einer Anzahl **eindeutig** ausführbarer Operationen und einer Angabe über den **nächsten Schritt** besteht.

- Berechnungsvorschrift
- schrittweise
- endlich
- ausführbare Operationen
- Angabe der Reihenfolge der Schritte

(Der Name *Algorithmus* ist abgeleitet von Al Chwarizmi, arabischer Mathematiker, ca. 800 n.Chr.)



Beispiel: Algorithmus „Sum“

Algorithmus „Sum“ zum Berechnen der Summe einer Zahlenfolge

Aufgabenstellung:

Gegeben: Eine Zahl n größer 0

Gesucht: Die Summe dieser Zahlen von 1 bis n

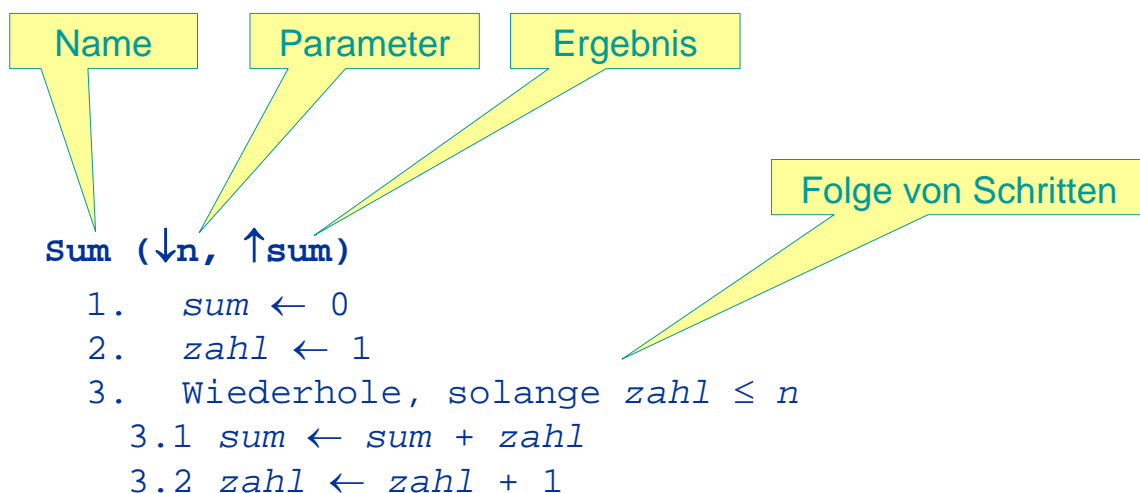
Algorithmus:

1. Addiere alle Zahlen von 1 bis n
2. Gib das Ergebnis aus

Diese Formulierung ist für einen Computer noch *zu wenig genau!*



Algorithmus „Sum“ als Verfahren in mehreren Schritten

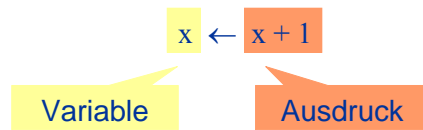


Diese Formulierung ist einem Computerprogramm schon *sehr nahe.*



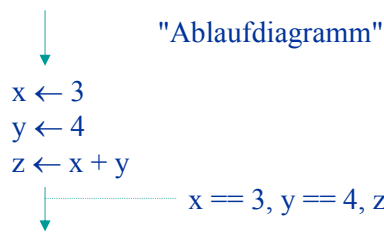
Anweisungen: Wertzuweisung und Sequenz

Wertzuweisung



1. werte Ausdruck aus
2. weise seinen Wert der Variablen zu

Anweisungsfolge (auch Sequenz)



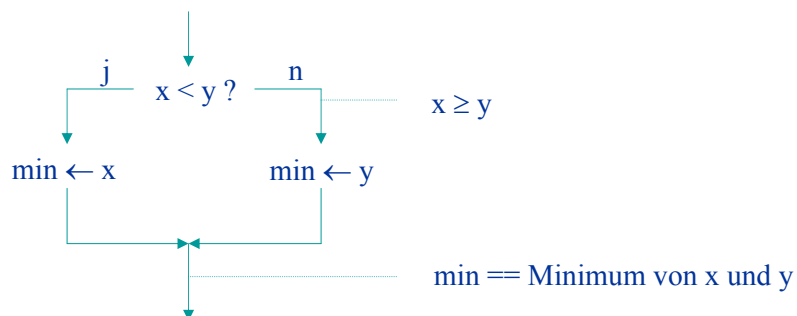
Assertion

Assertion (Zusicherung)
Aussage über den Zustand des Algorithmus an einer bestimmten Stelle



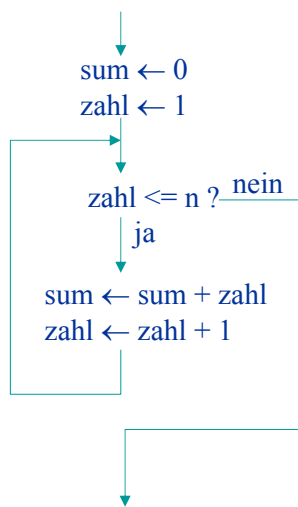
Anweisungen: Verzweigung

Auswahl (auch Verzweigung, Abfrage, Selektion)

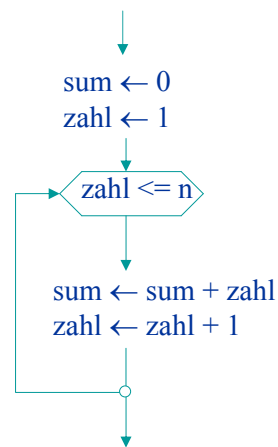


Anweisungen: Wiederholung

Wiederholung (auch Schleife, Iteration)

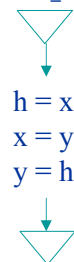


Alternative Darstellung



Beispiel: Vertauschen zweier Variableninhalte

Swap ($\uparrow x, \uparrow y$)

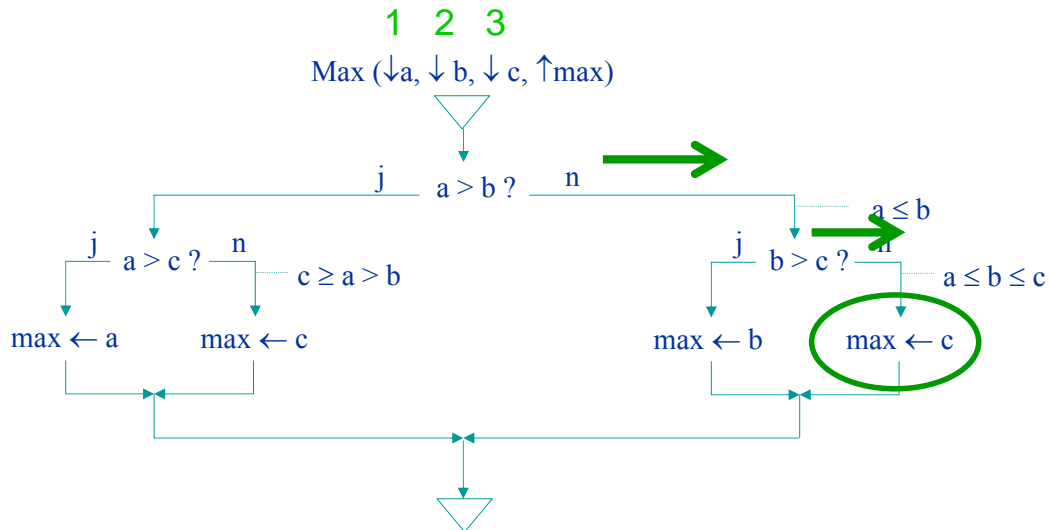


Schreibtischtest

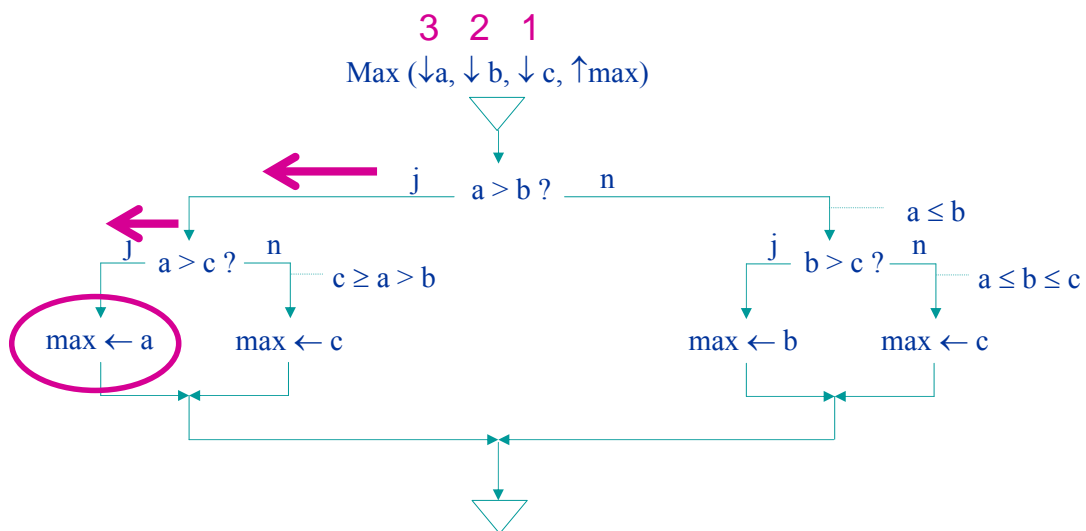
x	y	h
3	2	3
2	3	



Beispiel: Maximum dreier Zahlen bestimmen

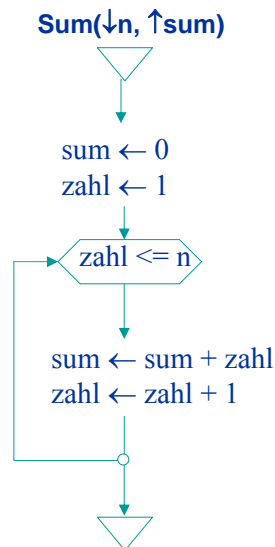


Beispiel: Maximum dreier Zahlen bestimmen



Beispiel: Summe

- Summe von 1 bis n



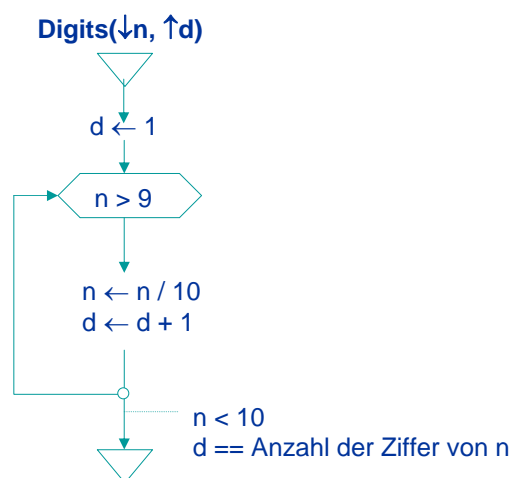
Schreibtischtest (n = 5)

	sum	zahl
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6



Beispiel: Anzahl der Ziffern einer Dezimalzahl

- Zahl solange durch 10 dividieren, bis Zahl < 10
- Anzahl der Divisionen + 1 = Anzahl der Ziffern



Schreibtischtest

n	d
12345	1
1234	2
123	3
12	4
1	5



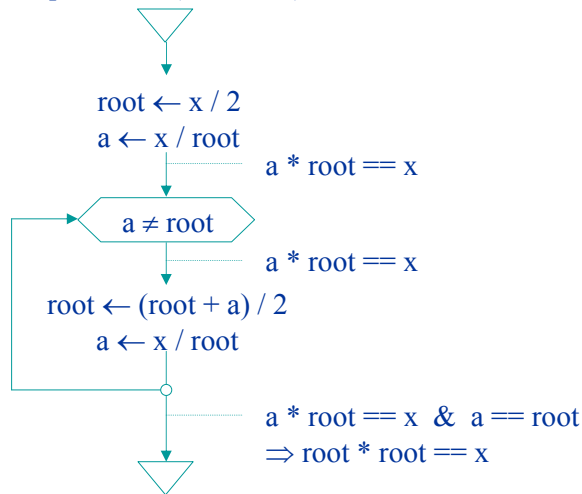
Beispiel: Quadratwurzel von x berechnen

Näherung:

$$\begin{aligned} \text{root} &\leftarrow (a + \text{root}) / 2 \\ a &\leftarrow x / \text{root} \end{aligned}$$



SquareRoot ($\downarrow x, \uparrow \text{root}$)



Schreibtischtest

x	root	a
10	5	2
	3.5	-2.85714
	3.17857	3.14607
	3.16232	3.16223
	3.16228	3.16228



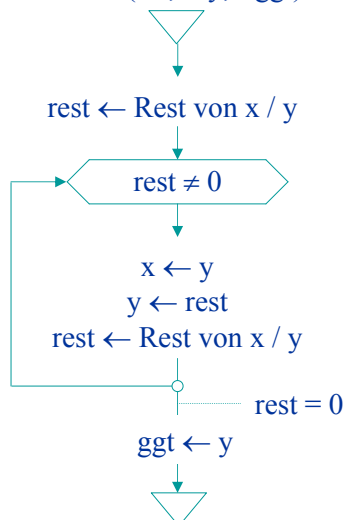
Kommazahlen sind meist nicht exakt gleich, daher besser $|a - \text{root}| > 0.0000001$



Beispiel: Euklidischer Algorithmus

Berechnet den größten gemeinsamen Teiler zweier Zahlen x und y

GGT ($\downarrow x, \downarrow y, \uparrow \text{ggt}$)



Beobachtung:

- (ggt teilt x) & (ggt teilt y)
- ⇒ ggt teilt (x - y)
- ⇒ ggt teilt (x - q*y)
- ⇒ ggt teilt Rest von x/y (rest)
- ⇒ GGT(x, y) = GGT(y, rest)

Schreibtischtest

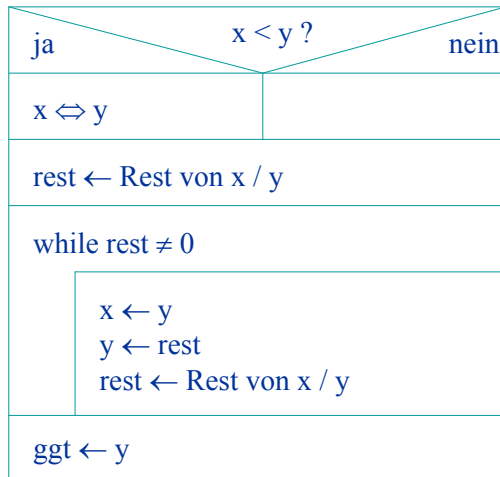
x	y	rest
28	20	8
20	8	4
8	4	0



Ablaufdiagramm (schon bekannt) ✓

Struktogramm

GGT ($\downarrow x, \downarrow y, \uparrow ggt$)



+ erzwingen strukturiertes Programmieren
(ohne beliebige Sprünge)

- aufwendig zu zeichnen
schwer zu ändern



Algorithmenschreibweisen: Stilisierte Prosa

Stilisierte Prosa

GGT ($\downarrow x, \downarrow y, \uparrow ggt$)

- S1 Mache $x \geq y$.
Wenn $x < y$ ist, vertausche x und y
- S2 Bilde Rest.
Dividiere x durch y und nenne den Rest $rest$
- S3 Ende?
Wenn $rest$ null ist, gehe zu S5
- S4 Ersetze.
Ersetze x durch y und y durch $rest$
Gehe nach S2
- S5 Fertig.
Das Ergebnis ggt ist y

+ größtmögliche Freiheit in der
Formulierung

- schreibaufwendig
- unpräzise
- Ablaufstrukturen (Schleifen,
Verzweigungen) nicht sichtbar



- + vom Computer lesbar und verarbeitbar
- + eindeutig und präzise
- + von Menschen lesbar
- exakt, ohne Freiheitsgrade
- aufwendig zu schreiben

Java-Programm

```
public class GGTProgram {  
  
    public static void main(String[] args) {  
  
        int x;  
        int y;  
        int ggt;  
  
        Out.print("Bitte x eingeben: ");  
        x = In.readInt();  
        Out.print("Bitte y eingeben: ");  
        y = In.readInt();  
        ggt = GGTEuklid(x, y);  
        Out.print("Der GGT von x und y ist: ");  
        Out.println(ggt);  
    }  
  
    static int GGTEuklid(int x, int y) {  
  
        int rest;  
        int ggt;  
  
        // %-Operator bildet den Rest  
        rest = x % y;  
        while (rest != 0) {  
            x = y;  
            y = rest;  
            rest = x % y;  
        }  
  
        // rest == 0  
        ggt = y;  
        return ggt;  
    }  
} // end GGTProgram
```



Prosa + Java

- Java-Kontrollstrukturen
- mit Anweisungen in Prosa

```
int GGTEuklid(x, y) {  
  
    rest = Rest von x/y  
    while (rest nicht 0 ist) {  
        x = y  
        y = rest  
    }  
    rest = Rest von x/y  
    ggt = y  
    return ggt  
}
```

- + nahe einem Java-Programm
- + beliebige Freiheit bei der Formulierung von Anweisungen
- + muss nicht syntaktisch korrekt sondern nur verständlich sein
- + daraus kann schrittweise ein Java-Programm entwickelt werden



Programm

Ein für die Lösung einer bestimmten Aufgabe mit einer Datenverarbeitungsanlage geeigneter Algorithmus

d.h.

- Elementare Operationen sind die durch den Computer ausführbaren Befehle
- Programm muss in einer für den Computer lesbaren Form definiert sein (==> Programmiersprache)

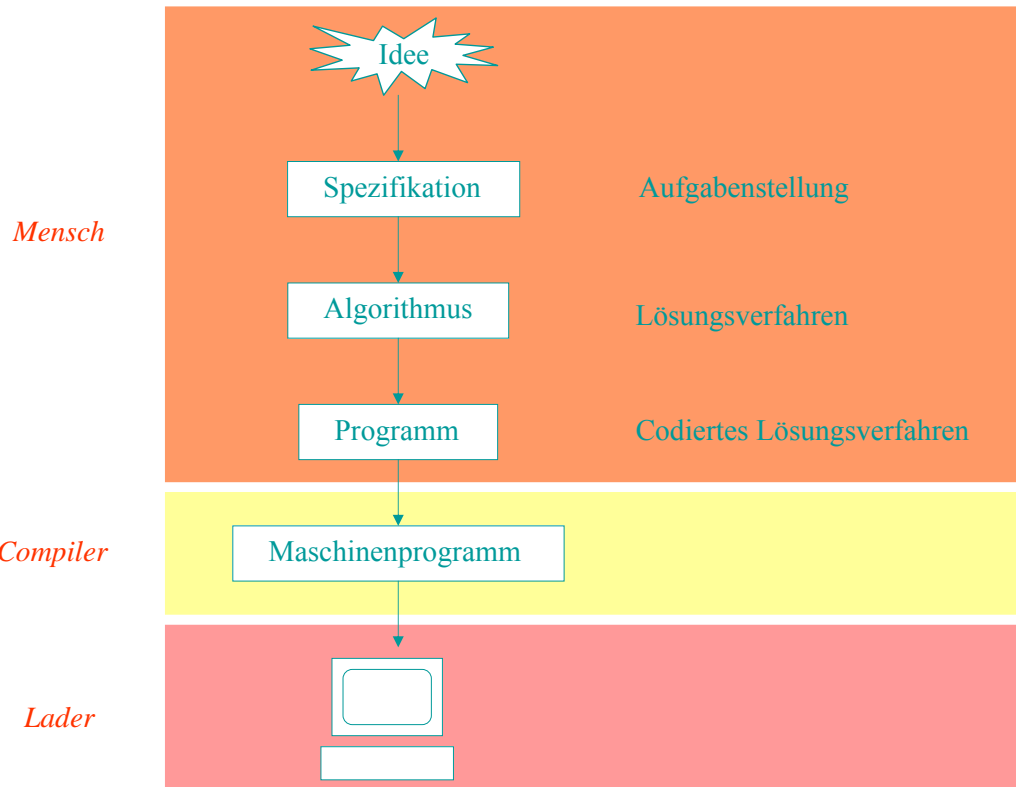


Höhere Programmiersprachen

- Höhere Programmiersprachen dienen der Formulierung von Programmen
- Sind für den Computer lesbar und verarbeitbar
 - können vom Computer gelesen werden
 - können vom Computer in ein ausführbares Programm übersetzt werden (*Compiler*)
- für den Menschen wesentlich einfacher lesbar und schreibbar als Maschinensprache
 - Programmieren wird wesentlich erleichtert
 - Programme können vom Experten gelesen und verstanden werden



Programmerstellung



Geschichte der Programmiersprachen

Jahr	Sprache	Bemerkung
1957	Fortran	technische Anwendungen
1960	Algol-60	"Algol-Familie"
	Cobol	kaufmännische Anwendungen
1962	Basic	Anfänger-Sprache
	APL	Vektoren und Matrizen
	Lisp	funktionale Sprache
1965	PL/I	Allzwecksprache
1968	Algol-68	Nachfolger von Algol-60
1971	Pascal	Einfluß auf spätere Sprachen
1972	Prolog	Wissen und Schlußregeln
1973	C	"Unix-Sprache"
1980	Modula-2	Nachfolger von Pascal
	Ada	Allzwecksprache; DoD
	Smalltalk	objektorientierte Sprache
1983	C++	objektorientierte Erweiterung von C
1987	Oberon	Weiterentwicklung von Modula-2
1995	Java	Weiterentwicklung von C++; WWW
2000	C#	Weiterentwicklung von Java durch Microsoft



Programmiersprache Java

- Java ist eine höhere, problemorientierte Programmiersprache, die laut Sun durch folgende Eigenschaften charakterisiert ist:
 - Simple: einfacher als C++
 - Architecture-neutral: läuft auf Windows-Rechner als auch auf Unix/Linux, Mac
 - Object-oriented
 - Portable
 - Distributed
 - High-performance
 - Interpreted
 - Multithreaded
 - Robust
 - Secure
 - Dynamic



Java Virtual Machine

- Java-Programme werden nicht *direkt* auf dem Computer ausgeführt sondern von einen speziellen Programm, nämlich der

Java Virtual Machine (JVM)

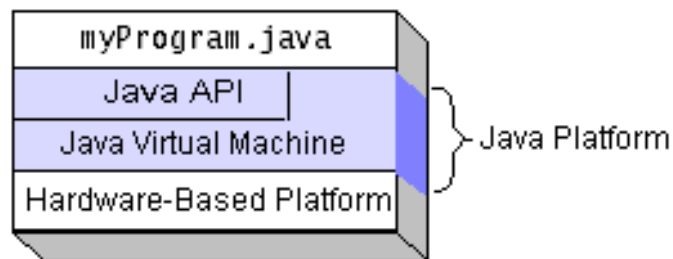
Vorteile:

- Java-Programme laufen „kontrolliert“ in der JVM ab
 - Erhöhte Sicherheit und Zuverlässigkeit
- Kann auf jede Plattformen portiert werden.
 - Vom Super Computer bis zu Handys
 - Von Unix bis Windows-CE
 - Selbes Programm läuft (mit Einschränkungen) auf alle diesen Plattformen ohne nochmals übersetzt oder portiert zu werden, da sich die VM auf jeder Plattform gleich verhält



Java Runtime Environment (JRE)

- Eine Plattform ist die Hard- und Softwareumgebung auf der ein Programm läuft.
- Die meisten Plattformen bestehen aus Hardware und Betriebssystem.
- Die Java-Plattform unterscheidet sich von den meisten anderen Plattformen dadurch, dass sie eine reine Software-Plattform ist, die für verschiedenste Hardware und Betriebssysteme verfügbar ist.
- Die Java Runtime Environment besteht aus zwei Teilen:
 - Die Java Virtual Machine (Java VM)
 - Das Java Application Programming Interface (Java API)



Erstellen und Ausführen von Java-Programmen

Beispielprogramm HelloWorld

↓ Erstellen von Java-Programm
mit Editor

Quellcode (Sourcecode) in Java

↓ Java Compiler (**javac**)

Java Zwischencode (Byte Code)

↓ Java Runtime (**java**)

Programmausführung

↓ Erstellen von Programm
HelloWorld mit Editor

Datei `HelloWorld.java` mit Java-Source

↓ `javac HelloWorld.java`

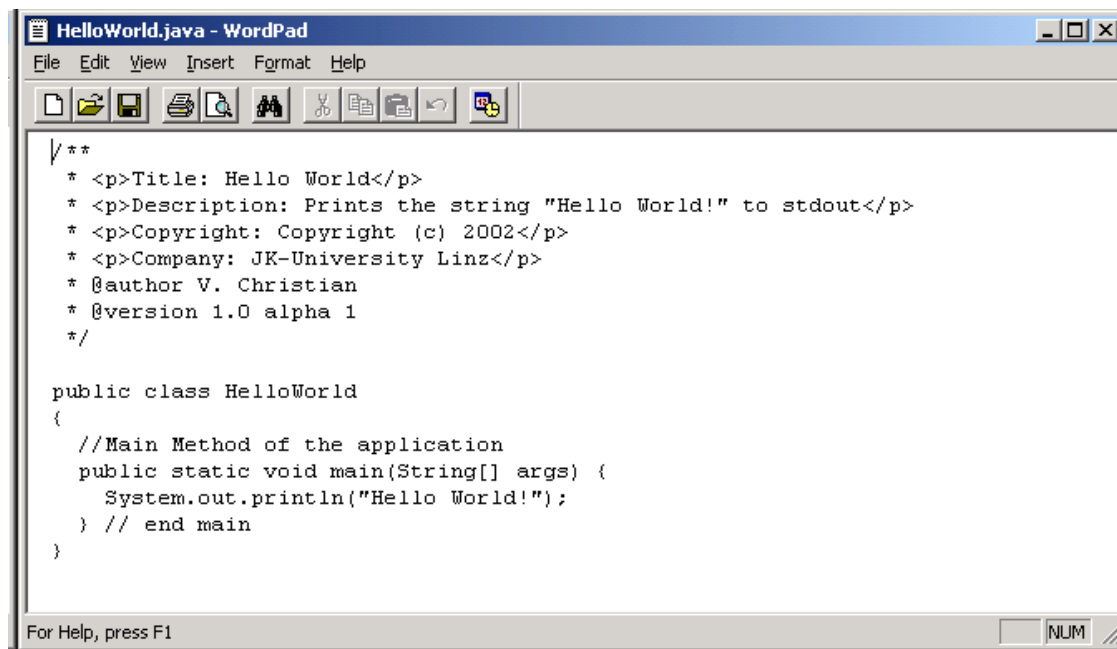
Datei `HelloWorld.class` mit Byte-Code

↓ `java HelloWorld`

Ausführung von Programm `HelloWorld`



Programm schreiben (Wordpad)

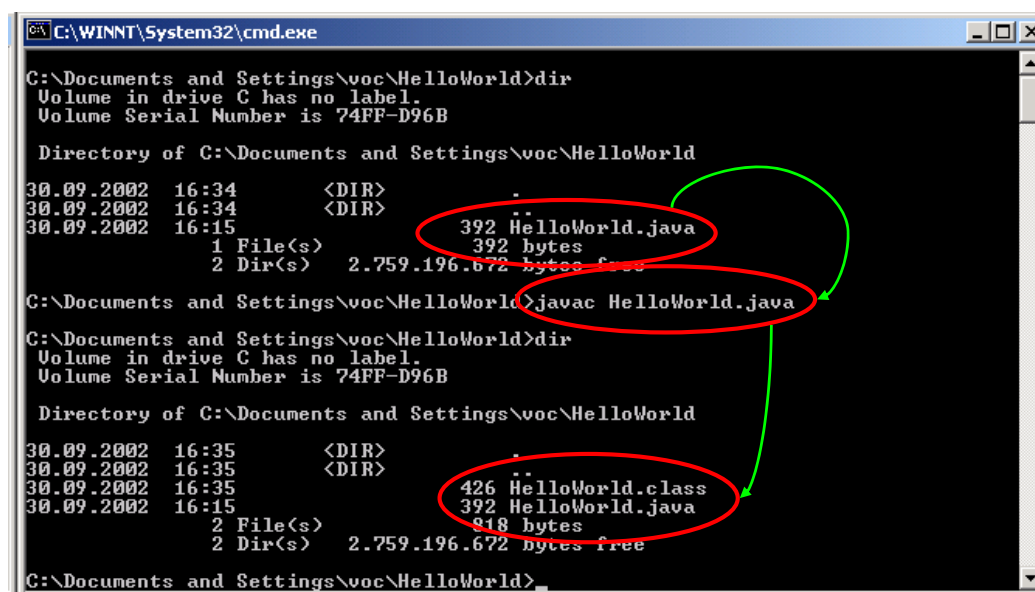


```
/**
 * <p>Title: Hello World</p>
 * <p>Description: Prints the string "Hello World!" to stdout</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: JK-University Linz</p>
 * @author V. Christian
 * @version 1.0 alpha 1
 */

public class HelloWorld
{
    //Main Method of the application
    public static void main(String[] args) {
        System.out.println("Hello World!");
    } // end main
}
```



Programm kompilieren (javac)



```
C:\WINNT\System32\cmd.exe

C:\Documents and Settings\voc\HelloWorld>dir
Volume in drive C has no label.
Volume Serial Number is 74FF-D96B

Directory of C:\Documents and Settings\voc\HelloWorld
30.09.2002  16:34         <DIR>          .
30.09.2002  16:34         <DIR>          ..
30.09.2002  16:15             392 HelloWorld.java
               1 File(s)              392 bytes
               2 Dir(s)  2.759.196.672 bytes free

C:\Documents and Settings\voc\HelloWorld>javac HelloWorld.java

C:\Documents and Settings\voc\HelloWorld>dir
Volume in drive C has no label.
Volume Serial Number is 74FF-D96B

Directory of C:\Documents and Settings\voc\HelloWorld
30.09.2002  16:35         <DIR>          .
30.09.2002  16:35         <DIR>          ..
30.09.2002  16:35             426 HelloWorld.class
30.09.2002  16:15             392 HelloWorld.java
               2 File(s)              818 bytes
               2 Dir(s)  2.759.196.672 bytes free

C:\Documents and Settings\voc\HelloWorld>
```



Programm starten (java)

```
C:\WINNT\System32\cmd.exe
C:\Documents and Settings\voc\HelloWorld>dir
Volume in drive C has no label.
Volume Serial Number is 74FF-D96B

Directory of C:\Documents and Settings\voc\HelloWorld
30.09.2002  16:35    <DIR>
30.09.2002  16:35    <DIR>
30.09.2002  16:35    .
30.09.2002  16:15    426 HelloWorld.class
30.09.2002  16:15    392 HelloWorld.java
                2 File(s)          818 bytes
                2 Dir(s)  2.759.192.576 bytes free

C:\Documents and Settings\voc\HelloWorld>java HelloWorld
Hello World!

C:\Documents and Settings\voc\HelloWorld>
```



Schritte des Algorithmenentwurfs und der Programmierung

- Problem erfassen und beschreiben
- Lösungsidee erarbeiten und niederschreiben
- Lösungsidee in einen schrittweisen Ablauf überführen (= Algorithmus)
- Algorithmus so weit verfeinern und konkretisieren, dass Umsetzung in Java direkt möglich ist
- Programmierung in Java
- Überlegen von Testfällen (dabei alle möglichen Sonderfälle, Grenzfälle betrachten)
- Testen und Verbessern
- Dokumentieren der Ergebnisse

