



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Programmierungskurs Java

Grundlagen

Raphael Allner
Institut für Telematik
15. Oktober 2019

1. Höhere Programmiersprachen
2. Warum eigentlich Java?
3. Compiler, Laufzeitumgebung, Entwicklungsumgebung
4. Ein erstes Java-Programm – „Hello World“
5. Sprachmerkmal – Sequentielle Ausführung



Grundlagen

Höhere Programmiersprachen

Höhere Programmiersprachen

Definition Algorithmen



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Algorithmen

- Eindeutige *Handlungsanweisungen* zur Lösung von Problem(klassen)
- Kann man mit Programmiersprachen „aufschreiben“
- *Details → Siehe Theorievorlesung*

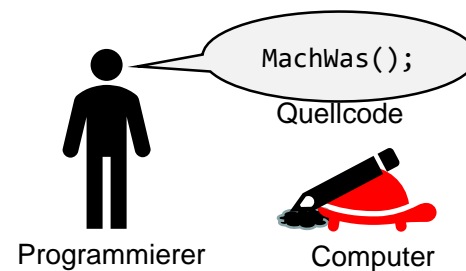
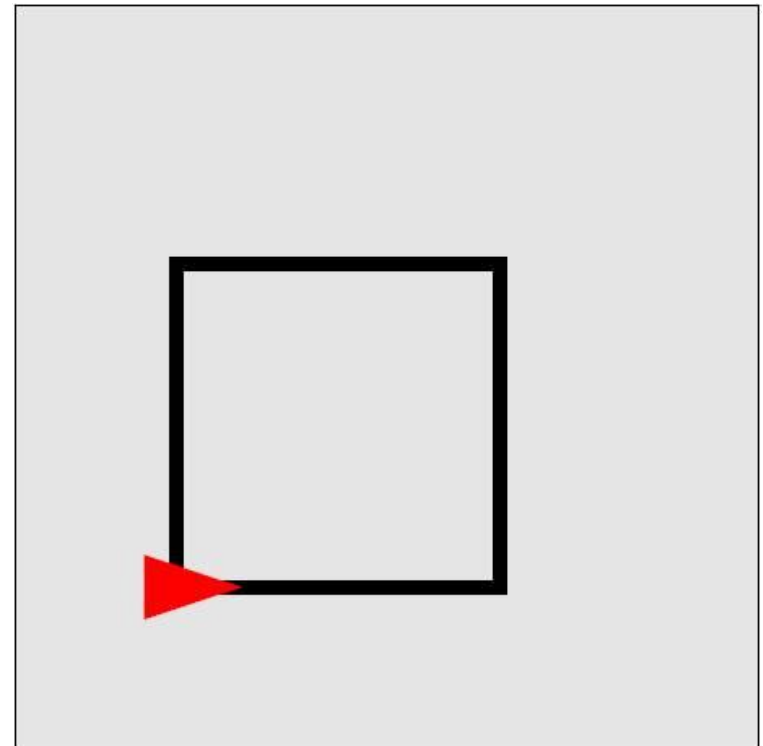
Maschinensprache genügt im Prinzip dafür

- Besteht nur aus einer **Menge von Nullen und Einsen**
- Für Menschen nur **schwer verständlich**
- Maschinenspezifisch, **schwer** auf andere Rechner **zu portieren**
- **Schwer**, in der Maschinensprache **den Algorithmus „zu sehen“**

Beispiel



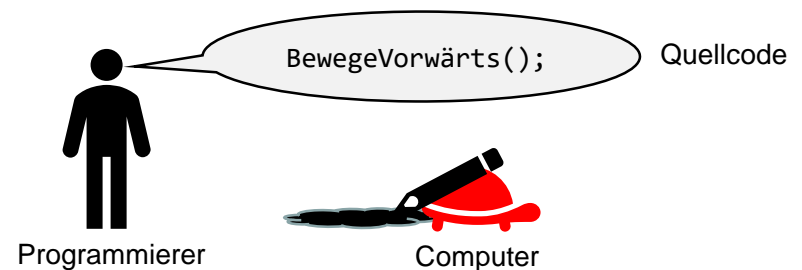
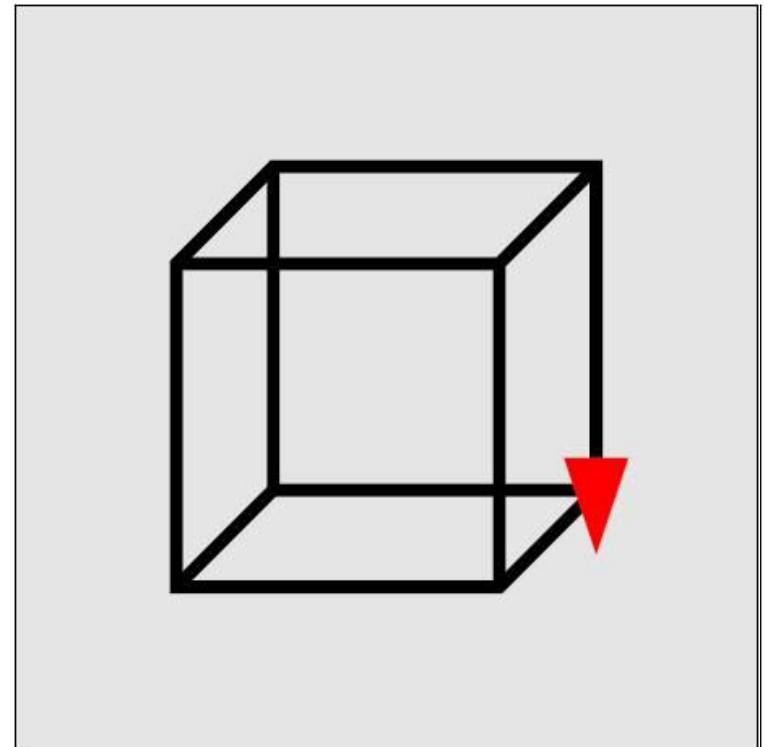
```
1 BeginneZeichnen();  
2 BewegeVorwärts();  
3 DreheNachLinks();  
4 BewegeVorwärts();  
5 DreheNachLinks();  
6 BewegeVorwärts();  
7 DreheNachLinks();  
8 BewegeVorwärts();
```



Beispiel



```
1 BeginneZeichnen();  
2 BewegeVorwärts(1);  
3 DreheNachLinks(90);  
4 BewegeVorwärts(1);  
5 DreheNachLinks(90);  
6 BewegeVorwärts(1);  
7 DreheNachLinks(90);  
8 BewegeVorwärts(1);  
9 DreheNachLinks(90 + 45);  
10 BewegeVorwärts(0.3);  
11 UnterbrecheZeichnen();  
12 BewegeNach(0.3,1.3);  
13 BeginneZeichnen();  
14 BewegeVorwärts(0.3);  
15 ...
```



Höhere Programmiersprachen

Maschinensprache



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Maschinensprache		Maschinennahesprache	Höhere Programmiersprache
Maschinencode (binär)	Maschinencode (hexadezimal)	zugehöriger Assemblercode	zugehöriger C-Code
1010101 10010001000100111100101	55 48 89 E5	<code>push rbp</code> <code>mov rbp, rsp</code>	<code>int main() {</code>
11000111010001011111110000000010	C7 45 FC 02	<code>mov DWORD PTR [rbp-4], 2</code>	<code>int a = 2;</code>
110001110100010111111100000000011	C7 45 F8 03	<code>mov DWORD PTR [rbp-8], 3</code>	<code>int b = 3;</code>
1000101101000101111111000 10001011010101010111111100 111010000 100010010100010111110100	8B 45 F8 8B 55 FC 01 D0 89 45 F4	<code>mov eax, DWORD PTR [rbp-8]</code> <code>mov edx, DWORD PTR [rbp-4]</code> <code>add eax, edx</code> <code>mov DWORD PTR [rbp-12], eax</code>	<code>int c = a + b;</code>
100010110100010111110100	8B 45 F4	<code>mov eax, DWORD PTR [rbp-12]</code>	<code>return c;</code>
1011101 11000011	5D C3	<code>pop rbp</code> <code>ret</code>	<code>}</code>

Quelle: <https://de.wikipedia.org/wiki/Maschinensprache> (Entnommen am 10.10.2018)

Höhere Programmiersprachen also...



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Höhere Programmiersprachen

- Sind näher am menschlichen Denken
- Programme bestehen aus sogenanntem „Quellcode“, „Programmcode“, „Source-Code“, „Source“ oder „Code“
- Lassen sich vom Computer in Maschinencode umwandeln (Kompilieren)

Diese Sprachen besitzen eine eigene **Syntax** und **Semantik**

Höhere Programmiersprache

Grammatik - Wiederholung



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Sprache mit eigener **Syntax** und **Semantik**

Syntax:

- **Grammatikregeln** der Sprache
- „Walfische bereisen Indien, um Wolken zu klauen!“
- Subjekt, Prädikat, Objekt

Semantik:

- **Bedeutung einzelner „Worte“** und **„Satzzeichen“** der Sprache
- Der obige Satz ist syntaktisch korrekt – aber sinnlos.

Ziel:

- **Sicheres Beherrschen** beider Aspekte
- Compiler **prüfen Syntax**, aber nur **teilweise Semantik**.

Höhere Programmiersprachen

Mächtig, Turing? Turing-mächtig?



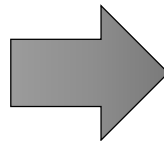
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Es existieren hunderte verschiedener Programmiersprachen

- Wikipedia listet ca. 550
- Prinzipiell sind **alle gleichmächtig** (Turing-vollständig)

Java ist **turingmächtig bzw. Turing-Vollständig!**? Bedeutet:

- Es kann **alles berechnet** werden
- Sofern **genügend Speicher** verfügbar ist





Grundlagen

Warum eigentlich Java?

Warum Java?

Die Qual der Wahl - Rankings



IEEE Spectrum Trending Web Enterprise Jobs Open Custom Mobile Embedded Create custom ranking (Click to hide)

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	Web Mobile Embedded	100.0
2	Java	Web Mobile Enterprise	96.3
3	C	Mobile Enterprise Embedded	94.4
4	C++	Mobile Enterprise Embedded	87.5
5	R	Enterprise	81.5
6	JavaScript	Web	79.4
7	C#	Web Mobile Enterprise Embedded	74.5
8	Matlab	Enterprise	70.6
9	Swift	Mobile Enterprise	69.1

Wahl der Sprache anhand:

- Verbreitung
- Anwendungszweck
 - Web
 - Mobile
 - Enterprise
 - Embedded
- Branche
 - Sicherheitskritisch
 - Medien
- Vorlieben
- **Jobwunsch**

Quelle: <https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019> Entnommen 14.10.2019

Geschichte von Java



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

1991

- Gosling et al. entwickeln *Object Application Kernel (Oak)* auf Basis von C++
- **Ziele:** Plattform-Unabhängigkeit, Erweiterbarkeit der Systeme und Austauschbarkeit von Komponenten
- Ursprünglicher Einsatzbereich ist Haushaltselektronik

1993

- Oak wird wegen rechtlicher Probleme in *Java* umbenannt.
- Zu diesem Namen wurden die Entwickler beim Kaffeetrinken inspiriert
- Java entwickelt sich durch Applet-Technologie zur „Sprache des WWW“

Seit 1995

- Sun bietet *Java Development Kit (JDK)* mit Compiler und Interpreter kostenlos an

2009

- *Oracle* übernimmt Sun Microsystems

März 2019

- Java 8 als letzte „free public“ LTS Version festgelegt
- Oracle rät von allen Versionen unter 11 ab (aus Sicherheitsbedenken)

Warum Java?



Typische Einsatzbereiche für Java heute:

- Historisch: Applets laufen direkt im Web-Browser
- Desktop-Anwendungen (Java SE)
- Web-Server
- Geschäftsanwendungen (Java EE) mit Servlets, JSF, JSP, ...
- Handy-Anwendungen (Java ME, Android, ...)

Inzwischen sehr mächtig und performant!

- Vielzahl alternativer Sprachen für VM verfügbar (JRuby, Scala ...)

Direkte Konkurrenz von Microsoft ist C#

- Gesprochen: „C Sharp“

Warum Java?

Besonderheiten

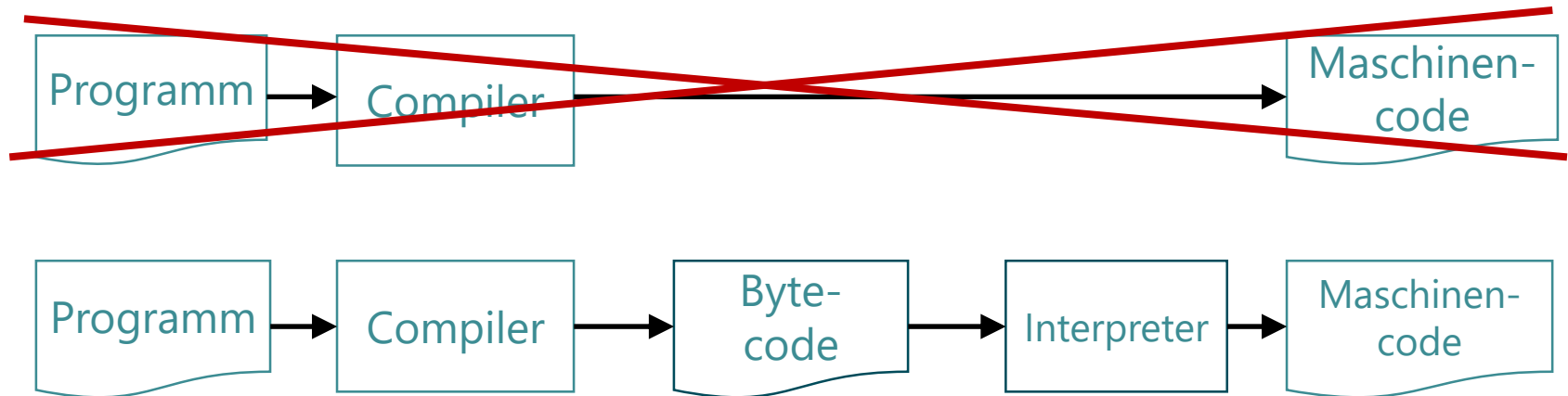


Besonderheiten von Java

- Sprache „neu“ entworfen, nur wenige Altlasten
- Daher vergleichsweise **einfach** zu erlernen
- „Write once, run anywhere“ (WORA) – Der Java Slogan



Technische Realisierung:



Warum Java?

Byte-Code



Byte-Code statt Maschinensprache

- Maschinencode für eine *Virtuelle Maschine* (VM)
- Die *Java Virtual Machine* (JVM)
 - interpretiert den Byte-Code
 - wandelt den Byte-Code in Maschinencode um



Ist eine passende JVM für einen Rechner verfügbar, dann kann **jedes kompilierte Java-Programm** ausgeführt werden

- Galt als potenziell langsamer als „echter“ Maschinencode
- Inzwischen wurden sog. Just-in-Time-Compiler entwickelt, welche den Geschwindigkeitsnachteil in vielen Bereichen kompensieren

.



Grundlagen

Compiler, Laufzeitumgebung,
Entwicklungsumgebung

Compiler und Laufzeitumgebung



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Java-Entwickler*innen benötigt **zwei** wesentliche Komponenten:

JAVA编译器

Java-Compiler

- `javac`
- *Java Development Kit (JDK)*

将JAVA文件编译为Class文件
用于创建一个Class文件来执行

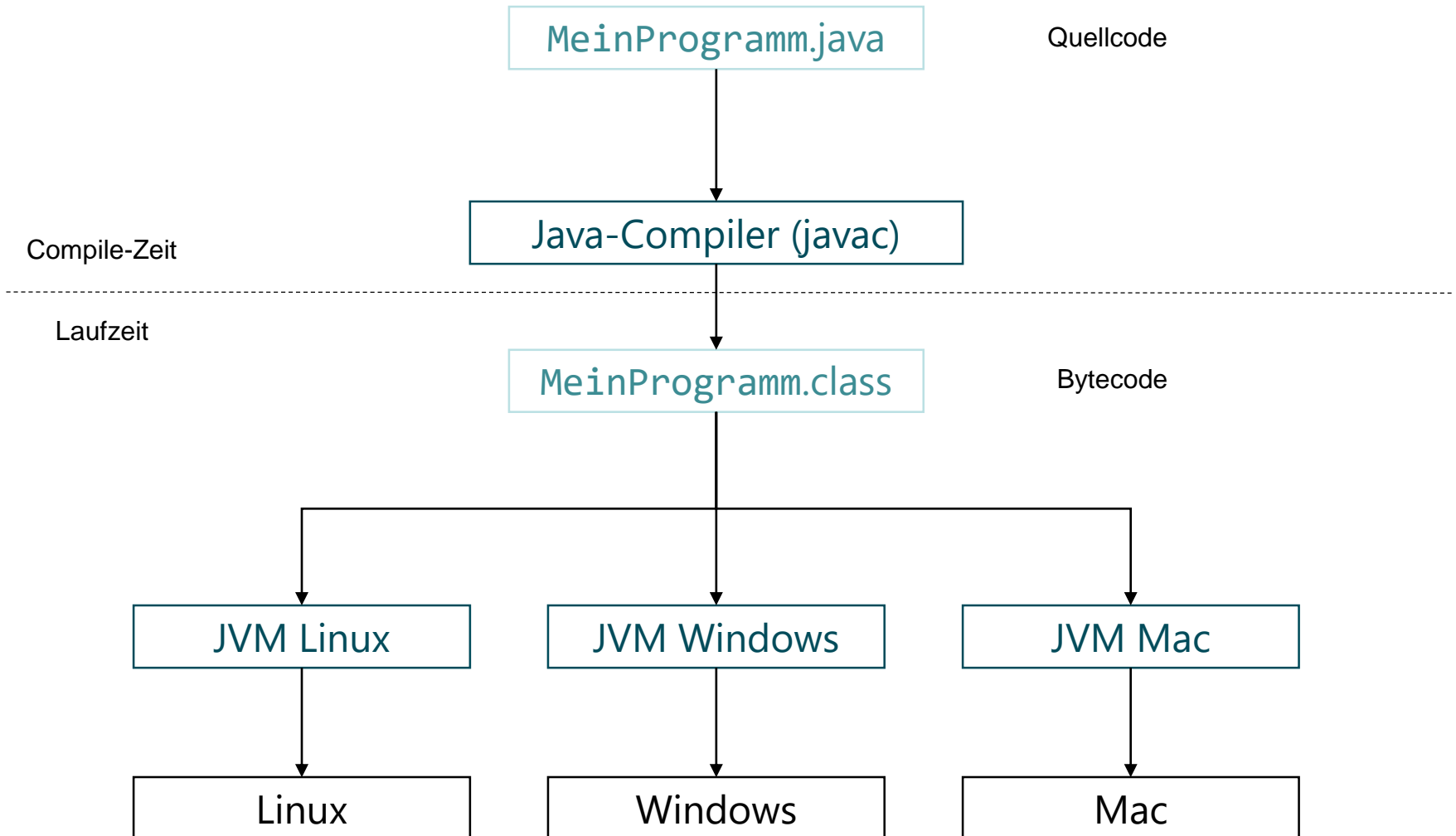
JAVA虚拟机

Java Virtual Maschine (JVM)

- `java`
- *Java Runtime Environment (JRE)*
- Auch im JDK enthalten
- Wird auch „Laufzeitumgebung“ genannt

通过Powershell来运行文件
与可以用其他软件，比如说eclipse

Compiler und Laufzeitumgebung



Welche Version? 6, 8, 11 oder 13?

- Es ist **nicht** immer sinnvoll die neuste Version einer Sprache zu verwenden
- **Empfehlung:** Version mit der **größten Verbreitung und längstem Support (LTS)**
→ Größte Community, Foren, Beispiele, Plugins und Bibliotheken
- **Wir verwenden Java 11 des OpenJDK** (<https://jdk.java.net/java-se-ri/11>)

Welche JDK-Variante:

Enterprise Edition (EE)

- Für **Webentwicklung und Business-Anwendungen**
- Deutlich mehr Bibliotheken enthalten

Mobile Edition (ME)

- Für **Mobile Anwendungen**
- Hat nichts mit Android zu tun
- Älter als Android - historisch

Standard Edition (SE)

- Alles was wir benötigen

Compiler und Laufzeitumgebung



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Schreiben von Java Programmen:

- Programme (genauer: der Quellcode) werden im Text-Editor geschrieben
- Unix: vim, nano, pico, nedit, emacs, gedit, SciTE, ...
- Windows: Atom, notepad++, edit, PSPad, UltraEdit, SciTe, ...
- Keine klassischen Textverarbeitungsprogramm (Word, OpenOffice, etc.)

```
package com.hakcode;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import javax.websocket.ContainerProvider;
import javax.websocket.DeploymentException;
import javax.websocket.Session;
import javax.websocket.WebSocketContainer;

public class App {

    public Session session;

    protected void start() {
        WebSocketContainer container = ContainerProvider.getWebSocketContainer();
        String uri = "ws://localhost:8080/websocket-glassfish-server/websocket/desktop-client";
        System.out.println("Connecting to " + uri);
        try {
            session = container.connectToServer(MyClient.class, URI.create(uri));
        } catch (DeploymentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String args[]) {
        App client = new App();
        client.start();

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String input = "";
        try {
            do {
                input = br.readLine();
                if(input.equals("quit")) {
                    client.session.getBasicRemote().sendText(input);
                } while(!input.equals("quit"));
            } catch (IOException e) {
                // TODO: Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

```
package com.hakcode;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import javax.websocket.ContainerProvider;
import javax.websocket.DeploymentException;
import javax.websocket.Session;
import javax.websocket.WebSocketContainer;

public class App {

    public Session session;

    protected void start() {
        {
            WebSocketContainer container = ContainerProvider.getWebSocketContainer();

            String uri = "ws://localhost:8080/websocket-glassfish-server/websocket/desktop-cl";
            System.out.println("Connecting to " + uri);
            try {
                session = container.connectToServer(MyClient.class, URI.create(uri));
            } catch (DeploymentException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public static void main(String args[]){
            App client = new App();
            client.start();
        }
    }
}
```

Integrated Development Environment (IDE)



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Integrated Development Environment (IDE)

- Integrieren: Editor, Compiler und VM
- Beispiele: IntelliJ, Eclipse, Netbeans, ...

Viele Zusatzfunktionen:

- Syntax Highlighting
- Code Completion
- Debugging
- Refactoring
- Plug-Ins für Spezialanwendungen

→ *Siehe Bonusvorlesung*

Für das **Erlernen** von Java **nicht nur** hilfreich

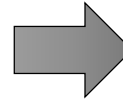




Grundlagen

Ein erstes Java-Programm – „Hello World“

Java ist turingmächtig:



Dafür erforderliche **Sprachmerkmale:**

- **Sequenz (Hintereinander-Ausführung)**
- Zuweisungen (zu Variablen)
- Elementare Rechenoperationen (Addition, Subtraktion, ...)
- Bedingte Ausführung (If-Anweisung)
- Wiederholungsanweisung (Schleifen)

Werden in den
folgenden Kapiteln
behandelt

Sprachmerkmale

Sequentielle Ausführung



Programme bestehen aus **Anweisungen** (einzelne Befehle), die in einer festen Reihenfolge nacheinander, **sequentiell**, ausgeführt werden.

Syntax:

```
1 BeginneZeichnen();  
2 BewegeVorwärts();  
3 DreheNachLinks();
```

Eine Anweisung wird durch ein ; abgeschlossen.

Semantik: $A(); B(); \rightarrow$ Mache **erst** $A()$ und **danach** $B()$.

Java-Programm – „Hello World“

Aufbau



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Grundgerüst: Wird für jedes Java-Programm benötigt!

- In Java ist alles in sog. Klassen strukturiert
- Formatierung dient der einfacheren Lesbarkeit

Klassenname 类名称
(muss mit Dateinamen übereinstimmen, inkl. Groß-/Kleinschreibung!)

main Methode
Programmeinstiegspunkt

```
1 public class MainClass {  
2     public static void main(String[] args){  
3         System.out.println("Hello World");  
4     }  
5 }
```

Anweisung
zum Ausgeben von Text

;
Ende einer Anweisung

Java-Programm – „Hello World“

Vollständiges Programm entwickeln



1. Datei mit dem Namen `MeinProgramm.java` **erstellen**
2. In einem Editor **editieren**
 - `nano` `MeinProgramm.java`
 - Groß- und Kleinschreibung beachten
3. Mit dem Java-Compiler `javac` **kompilieren**
 - `javac` `MeinProgramm.java`
 - Erzeugt Datei `MeinProgramm.class` mit Bytecode-Instruktionen
4. Mit der Java Virtual Machine (JVM) „java“ **ausführen**
 - `java` `MeinProgramm`
 - Achtung: weder „.java“ noch „.class“ am Ende
 - Groß- und Kleinschreibung beachten

Wiederkehrender Workflow:

Erstellen → Editieren → Kompilieren → Ausführen

生成==编辑==编译==执行



Java-Programm – „Hello World“

Ausgabe auf der Konsole



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
1 public class MeinProgramm {  
2     public static void main(String[] args){  
3         System.out.println("Hello World");  
4     }  
5 }
```

 MeinProgramm.class	18.09.2018 12:12	CLASS-Datei	1 KB
 MeinProgramm.java	18.09.2018 11:11	JAVA-Datei	1 KB

 Windows PowerShell

```
PS C:\dev\Java-Kurs> javac .\MeinProgramm.java  
PS C:\dev\Java-Kurs> java MeinProgramm  
Hello World  
PS C:\dev\Java-Kurs> _
```

```
1 public class MeinProgramm {public static void main(String[] args) {  
System.out.println("Hello World");System.out.println(42);}}
```

```
1 public  
2 class MeinProgramm {  
3     public static  
4 void main(String[] args)  
5     {System.out.  
6 println("Hello World")  
7 ;System.out.println(42);}}
```

Formatierung für den Compiler **egal**

- Für Menschen ergibt es Sinn, sich an gewisse **Konventionen** zu halten.

```
1 public class MeinProgramm {  
2     public static void main(String[] args){  
3         System.out.println("Hello World");  
4     }  
5 }
```

Kommentare zur Dokumentation



Kommentare, damit Menschen Code besser verstehen können 程序的注解

- Werden vom Compiler ignoriert

Varianten (in Java)

- `//` Kommentar 1-zeilig
- `/*` Kommentar potenziell über mehrere Zeilen `*/`

Vorsicht: „Verschachteln“ **nicht** erlaubt (`/* /* */ */`)

```
1 public class MeinProgramm {  
2     public static void main(String[] args){  
3         // Konsolenausgabe als Beispiel für eine Anweisung  
4         System.out.println("Hello World");  
5         /* Dies ist ein  
6             mehrzeiliger  
7             Kommentar */  
8     }  
9 }
```

Auch Kommentare müssen gepflegt werden!

Online Compiler



Kein Laptop? Kein Java JDK? Kein Problem!

Online Java-Compiler:

- Moodle
- Codiva - <https://www.codiva.io/>
- Jdoodle - <https://www.jdoodle.com/online-java-compiler>
- Browxy - <http://www.browxy.com/>
- Rextester - https://rextester.com/l/java_online_compiler

Online Compiler



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Raphael Allner



UNIVERSITÄT ZU LÜBECK
STIFTUNGSUNIVERSITÄT
SEIT 2015

Moodle der Universität zu Lübeck

Universität zu Lübeck → Start: Meine Kurse → Institut für Telematik → ITM-WS1819-Prog → Übungsmaterialien (Theorie und Java) → Verzinsung eines Guthabens für ein Jahr

Start: Meine Kurse

Informationen zu Moodle

Vorlesungsverzeichnis

Institut für Telematik

ITM-WS1819-Prog

Teilnehmer/innen

Bewertungen

Übungsmaterialien (Theorie und Java)

Verzinsung eines Guthabens für ein Jahr

- Beschreibung
- Abgabe
- Bearbeiten
- Abgabesicht
- Vorige Abgabeliste
- Abgabeliste
- Ähnlichkeit

EINSTELLUNGEN

VPL Administration

- Einstellungen
- Testfälle
- Ausführungsoptionen
- Erforderliche Dateien
- ▶ Erweiterte Einstellungen
- ▶ Testaktivität
- Virtual programming labs
- Lokale Rollen zuweisen
- Rechte ändern
- Rechte prüfen
- Filter
- Sicherung

Einführung in die Programmierung - CS1000SJ14

Beschreibung Abgabeliste Ähnlichkeit Testaktivität

Abgabe Bearbeiten Abgabesicht Bewertung Vorige Abgabeliste

★ Zinsen.java

```
1 import java.lang.Math;
2 import java.util.Scanner;
3
4 public class Zinsen {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7
8         // Variablen deklarieren und Eingabe einlesen
9         double guthaben = scanner.nextDouble();
10        double guthabenMitVerzinsung01 = 0.0;
11        double guthabenMitVerzinsung05 = 0.0;
12        double guthabenMitVerzinsung08 = 0.0;
13
14        // *** Fügen Sie hier Ihre Loesung ein:
15
16
17
18        // Ergebnisse ausgeben:
19        System.out.println(guthabenMitVerzinsung01);
20        System.out.println(guthabenMitVerzinsung05);
21        System.out.println(guthabenMitVerzinsung08);
22    }
23 }
```

Bewertungsvorschlag: 3 / 3

Ergebnisse
Das Programm bestand 11/11 Tests (1/1 offene und 10/10 versteckte).
Die bestandenen Tests liefern 3/3 Punkte.

Offene Tests
1. Korrekt. Eingabe "9137" lieferte
9146.136999999999
9182.685
9210.096


Kommentare

Ausführung

Beschreibung



Übungs- oder Hausaufgabe:

1. Laptop / PC für Übungsaufgaben in der Vorlesung vorbereiten
2. Einen einfachen Editor auswählen bzw. installieren (Keine IDE!)
3. Aktuelles Java Development Kit (JDK) installieren
 - Version: Java SE
 - Pathvariable setzen! 
4. Installation überprüfen mit der *Kommandozeile* (CI)
 - `MeinProgramm.java` (Siehe Moodle Materialien) mit dem Befehl `javac` compilieren
 - Ausführen mit `java MeinProgramm` (Ohne Dateiendung `.java/.class`)
 - „Hello World“ sollte ausgegeben werden

Für Fragen und Unterstützung: Betreute Rechnerzeit

Algorithmen - *Handlungsanweisungen* zur Lösung von Problem

Höhere Programmiersprachen – nah am menschlichen Denken

- Diese haben eigene **Syntax** und **Semantik**
- Werden in Maschinensprache übersetzt (vom Compiler)

Es gibt eine Vielzahl verschiedener Programmiersprachen

- Java ist **weit verbreitet** und viele Firmen suchen Javaentwickler
- Java ist **plattformunabhängig**

Programme laufen in der **Java Virtual Maschine (JVM)**

- JVM ist im **Java Runtime Environment (JRE)** enthalten (kostenlos)

Entwickler brauchen das **Java Development Kit (JDK)**

- Programme schreiben wir in einem **Texteditor** oder in einer **Integrated Development Environment (IDE)**

Wichtige Kommandozeilen Befehle:

- javac – Kompilieren des Quellcodes in Byte-Code
- java – Ausführen des Java-Programmes

In Java ist alles in **Klassen** strukturiert

- Der Einstieg in ein Programm ist die main-Methode

Java ist **Turing-mächtig**:

- Wenn genug Speicher vorhanden → kann alles berechnet werden
- Müssen über bestimmte **Sprachmerkmale** verfügen

Nächstes Thema:

Variablen und Datentypen

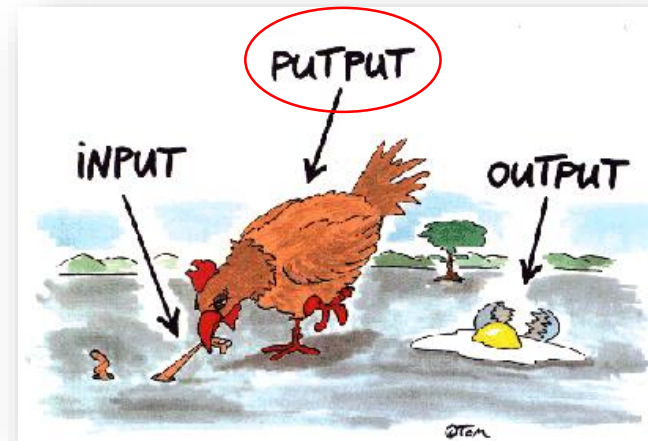


Programme verarbeiten Daten

- Werden **gelesen** und **ausgegeben**
- Notwendig: Input und Output (I/O)-Funktionen (später)

Zur Verarbeitung müssen diese im **Arbeitsspeicher** gehalten werden

- Programme müssen diese Daten referenzieren können
- **Konzept der Variablen**
 - Siehe nächste Vorlesung





UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Kontakt

Raphael Allner, M. Sc.
Wissenschaftlicher Mitarbeiter
Institut für Telematik

Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck

<https://www.itm.uni-luebeck.de/mitarbeitende/raphael-allner.html>

