

Vorkurs Programmieren

HTW Berlin
SoSe 2016

Inhalt Vorlesung 1

1. Einführung

1. Allgemein: Grundlagen, Tipps zum Herangehen
2. Java: Abgrenzung, Funktionsweise

2. Grundlagen: Hello World

1. Datentypen
2. Kontrollstrukturen
3. Funktionen

1.) Einführung

1.) Allgemein

Was ist Informatik?

- Ursprünge: E-Technik, Mathematik
- Ingenieursdisziplin bedeutet:
 - (Angewandte) Wissenschaft
 - Kennen von Verfahren:
z.B.: Programmierung, Systemadministration
 - Einsatz geeigneter Mittel und Verfahren
 - Lebenslanges Lernen, Lernen aus Fehlern

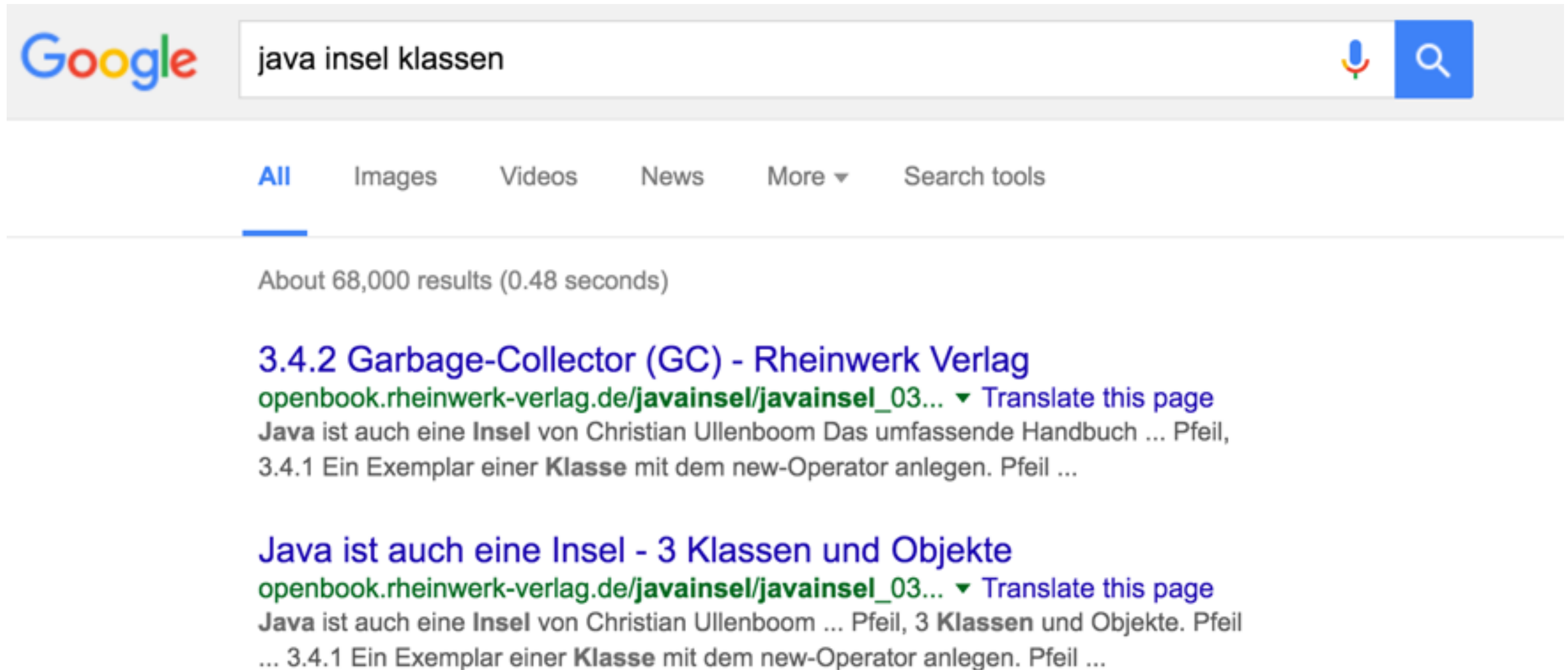
Was ist Informatik?

- Ist ein Problem lösbar?
- Mit welchem Aufwand kann ein Problem gelöst werden?
 - Rechenzeit
 - Rechenleistung
 - Speicherplatz

Was bedeutet Programmierung?

- Ein Handwerk der Informatik
- Spielen / Testen: Übung macht den Meister
- Patterns (Muster): Aufbauen auf Erfahrung anderer

Hilfe zur Selbsthilfe



The screenshot shows a Google search interface. The search bar contains the text "java insel klassen". Below the search bar, the "All" tab is selected. The search results show "About 68,000 results (0.48 seconds)". Two results are visible, both from the website "openbook.rheinwerk-verlag.de/javainsel/javainsel_03...". The first result is titled "3.4.2 Garbage-Collector (GC) - Rheinwerk Verlag" and the second is titled "Java ist auch eine Insel - 3 Klassen und Objekte". Both results include a "Translate this page" link.

Google

java insel klassen

All Images Videos News More ▼ Search tools

About 68,000 results (0.48 seconds)

3.4.2 Garbage-Collector (GC) - Rheinwerk Verlag
openbook.rheinwerk-verlag.de/javainsel/javainsel_03... ▼ Translate this page
Java ist auch eine **Insel** von Christian Ullenboom Das umfassende Handbuch ... Pfeil,
3.4.1 Ein Exemplar einer **Klasse** mit dem new-Operator anlegen. Pfeil ...

Java ist auch eine Insel - 3 Klassen und Objekte
openbook.rheinwerk-verlag.de/javainsel/javainsel_03... ▼ Translate this page
Java ist auch eine **Insel** von Christian Ullenboom ... Pfeil, 3 **Klassen** und Objekte. Pfeil
... 3.4.1 Ein Exemplar einer **Klasse** mit dem new-Operator anlegen. Pfeil ...

Hilfe zur Selbsthilfe

- **<http://openbook.rheinwerk-verlag.de/javainsel/>**
(via google: java Insel <?>)
- <https://docs.oracle.com/javase/7/docs/api/>
(via google: java api <?>)
- <https://stackoverflow.com/>
(via google: meistens von alleine)

Konventionen

- Sprache: Englisch!
- Google-Suche
- Bezeichnungen / Kommentare im Programm
- Sprachspezifische Konventionen (später...)
 - Tipp: neue Sprache; neue Konventionen, Patterns, Paradigmata
- Code-Style- / Clean-Code-Tipps (später ...)
 - Test-Driven-Development

1.) Einführung

2.) Java

Sprachen-Entwicklung

```

section .text                ;section declaration

                                ;we must export the entry point to the ELF linker or
                                ;loader. They conventionally recognize _start as their
                                ;entry point. Use ld -e foo to override the default.

    global _start

_start:

                                ;write our string to stdout

    mov     edx,len            ;third argument: message length
    mov     ecx,msg           ;second argument: pointer to message to write
    mov     ebx,1             ;first argument: file handle (stdout)
    mov     eax,4             ;system call number (sys_write)
    int     0x80              ;call kernel

                                ;and exit

    mov     ebx,0             ;first syscall argument: exit code
    mov     eax,1             ;system call number (sys_exit)
    int     0x80              ;call kernel

section .data                ;section declaration

msg db      "Hello, world!",0xa ;our dea
len equ     $ - msg           ;length
    
```

```

/* Hello World program */

#include<stdio.h>

main()
{
    printf("Hello World");
}
    
```

```

class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
    
```

Sprachen-Entwicklung

Verschiedene Abstraktionsebenen

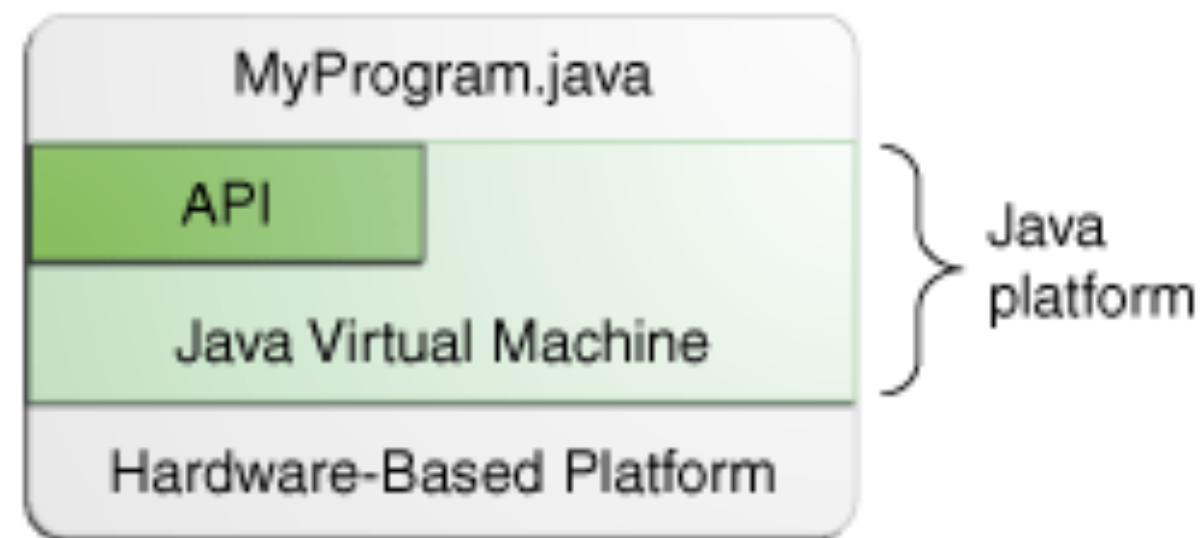
- Bessere Lesbarkeit / Wiederverwendbarkeit
- Größeren Funktionsumfang
- Java: lauffähig überall

Vom Code zum Prozess

- Compiler: Übersetzt Programmcode in Maschinensprache
- Laufzeit / Runtime: Programmausführung (Prozess)
- EVA / IPO: Eingabe - Verarbeitung - Ausgabe

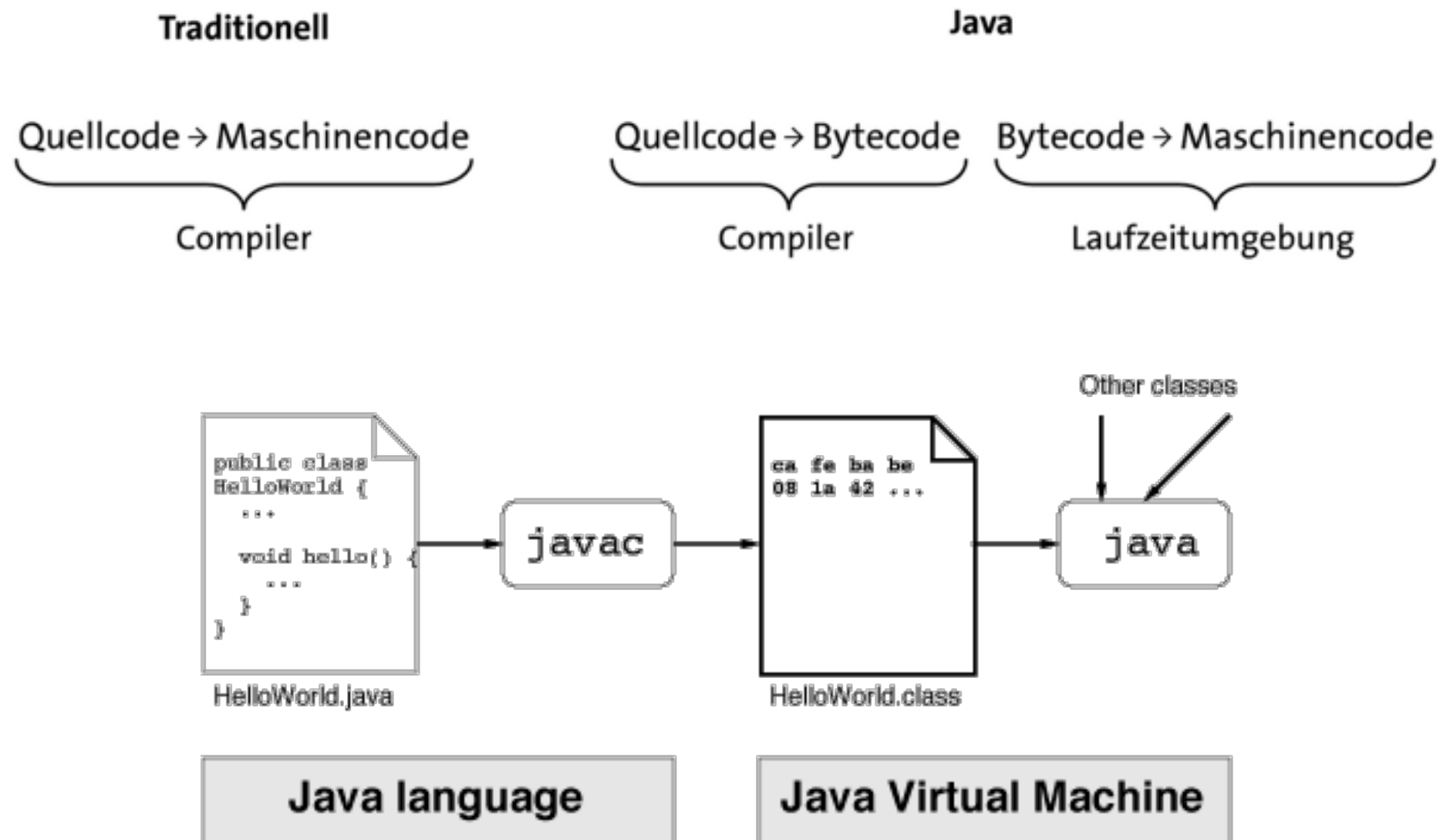
Von Code zum Prozess: Laufzeitumgebung

- Java: Abstraktion JVM, daher: Bytecode



Von Code zum Prozess: Übersetzung

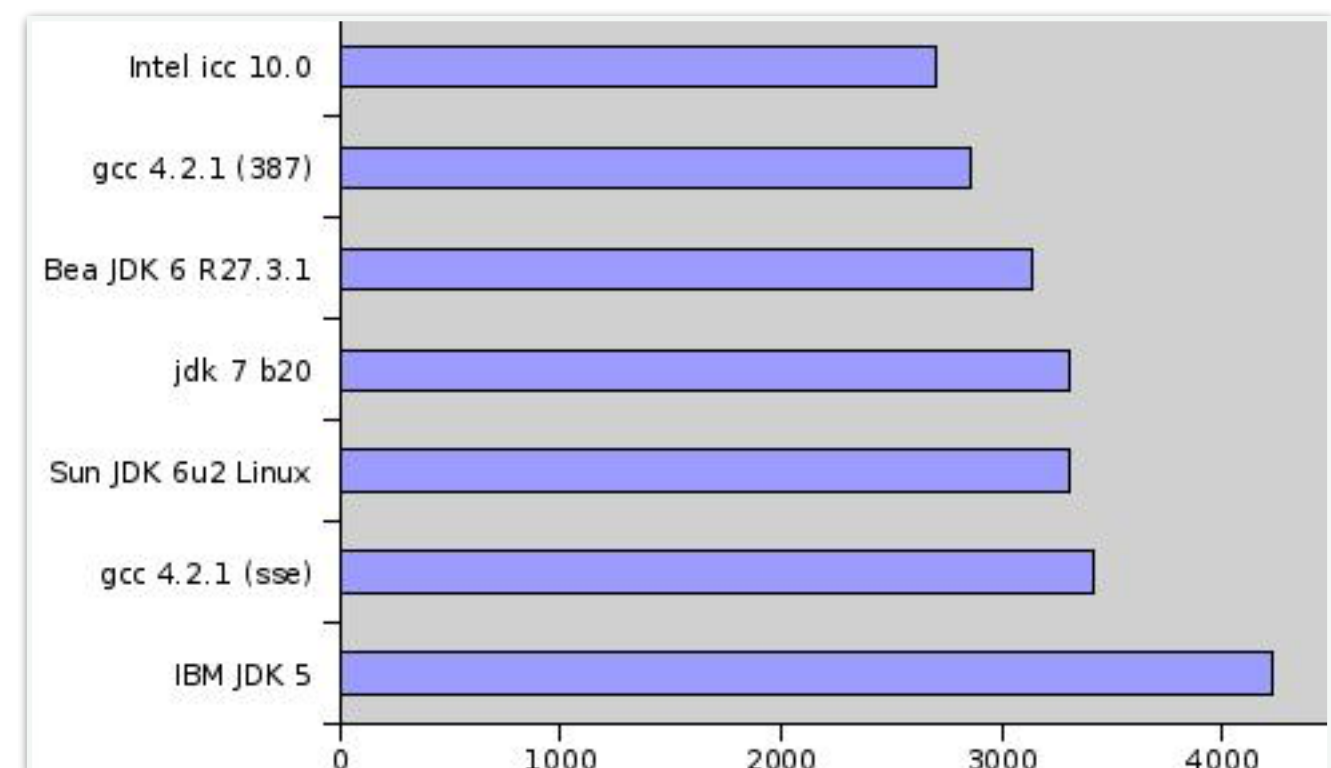
- Java: Abstraktion JVM, daher: Bytecode



Java: Zusammenfassung

- Paradigma: OOP (gut für Modularisierung)
- Typsicherheit
- Garbage Collection
- Plattformunabhängigkeit
- Nicht hardwarenah
- Beliebtheit ?!
- Geschwindigkeit ?!

Mar 2016	Mar 2015	Change	Programming Language
1	2	▲	Java
2	1	▼	C
3	4	▲	C++
4	5	▲	C#
5	8	▲	Python
6	6		PHP
7	9	▲	Visual Basic .NET
8	7	▼	JavaScript



Werkzeuge

- Programmiersprache: Java Version 7/8
(<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
- IDE: IntelliJ Idea
(<https://www.jetbrains.com/idea/>)

2.) Grundlagen: Hello World

Beispiel: Kompilierung und Ausführung

```
hello-java/1lecture [master●] » ls
```

```
Hello.java
```

```
hello-java/1lecture [master●] » cat Hello.java
```

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

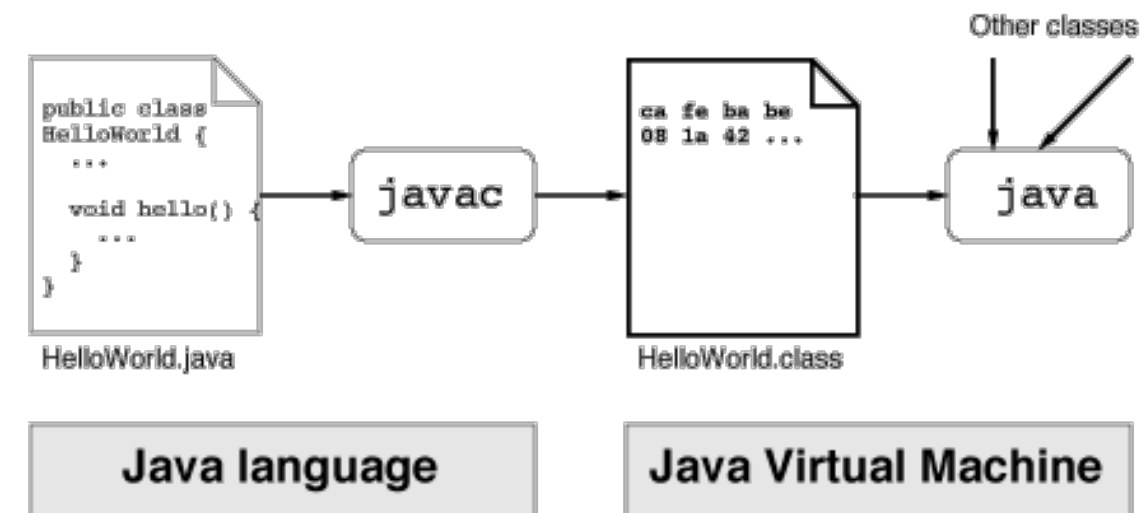
```
hello-java/1lecture [master●] » javac Hello.java
```

```
hello-java/1lecture [master●] » ls
```

```
Hello.class Hello.java
```

```
hello-java/1lecture [master●] » java Hello
```

```
Hello World
```



Grundaufbau

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

Begriffe:

- Token (kleinste lexikalische Einheit)
- Separator (Trennzeichen)
- Literale, Operatoren, Schlüsselwörter

Grundaufbau

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

Begriffe:

- Variablen mit Bezeichner und (Daten-) Typ
- Bezeichner: Variablen: `myNumber`, Methoden/Funktionen: `println`, Klassen: `Hello`
- Schlüsselwörter: `public`, `static`, `void`, `class`

Grundaufbau

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert</code>	<code>default</code>	<code>goto</code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protecte</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>const</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

Begriffe:

- Variablen mit Bezeichner und (Daten-) Typ
- Bezeichner: Variablen: `myNumber`, Methoden/Funktionen: `println`, Klassen: `Hello`
- Schlüsselwörter: `public`, `static`, `void`, `class`

Grundaufbau

```
public class Hello {  
  
    public static void main(String[] args) {  
        // TODO: translate to english  
        String msg = "Hallo World";  
        System.out.println(msg);  
    }  
}
```

Begriffe:

- Zeichensatz
- Kommentare: `//` oder `/* */`, Javadoc `/** */`
- Dokumentation: [https://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html#println\(java.lang.String\)](https://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html#println(java.lang.String))

2.) Grundlagen: Hello World

1.) Datentypen

Basis-Datentypen

- Zahlen:
 - Ganze Zahlen: (lat.: numerus integer)
 - Integer: 2^{32} Werte, $-2.147.483.648$ bis $2.147.483.647$
 - Long: 2^{64} Werte, $-/+ 9 \cdot 10^{19}$
 - *Byte, Short*
 - Rationale Zahlen: (engl.: floating point number)
 - Float: $1.4\text{E}-45$ bis $3.4028235\text{E}38$
 - Double: $4.9 \text{ E } -324$ bis $1.7976931348623157 \text{ E } 308$

Basis-Datentypen

- Zahlen:
 - Ganze Zahlen:
 - Integer: `int i = 1;`
 - Long: `long l = 2;`
 - Rationale Zahlen:
 - Float: `float f = 3.14;`
 - Double: `double d = 3.14;`

Basis-Datentypen

- Zahlen: (Standardwert: 0)
- Operatoren: `+` `-` `*` `/` `%` (...)
 - Kurzschreibweisen: `+=`, `++`
- Schreibweisen:
 - mit Datentyp: `.01F`, `100L`
 - mit Unterstrich (Java7): `100_000`
- Konstanten: `Math.PI` (begrenzte Genauigkeit)
 - Bsp.: `final int hoursPerDay = 24;`

Basis-Datentypen

- Buchstaben
- String: `String helpMsg = "Need Help?";`
 - mit Operator: `+` (Konkatenation)
- Char: `char abort = 'a';`
 - Achtung:
 - verhält sich wie Short (0 - 65535 Werte)
 - keine Konkatenation

Basis-Datentypen

- Wahrheitswerte (Standardwert: `false`)

- Boolean: `true`, `false`

```
boolean isActive = false;
```

- Operatoren: `!` `&` `|`

Beispiel

```
public class Main {  
    public static void main(String[] args) {  
        int perRow = 5;  
        int perColumn = 4;  
  
        int boxOfBeerSize = perRow * perColumn;  
        String message = "My box of beer contains: ";  
        System.out.println(message + boxOfBeerSize);  
    }  
}
```

Wie arbeitet ein Programm?

- Folge von Befehlen bzw. von Funktionen / Methoden / Prozeduren (Imperative / Prozedurale Programmierung)
- Verarbeitung von Variablen und Konstanten mit Deklaration und Initialisierung
- Wie beeinflussen wir den Programmablauf?

2.) Grundlagen: Hello World

2.) Kontrollstrukturen

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        boolean isActive = false;  
        if (isActive) {  
            System.out.println("I am active");  
        } else {  
            System.out.println("I am NOT active");  
        }  
    }  
}
```

if-else-Anweisung

Kurzschreibweise mit ?

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        boolean isActive = false;  
        if (isActive) {  
            System.out.println("I am active");  
        } else if (!!isActive) {  
            System.out.println("I am active, wtf?");  
        } else {  
            System.out.println("I am NOT active");  
        }  
    }  
}
```

if-else-Anweisung

Kurzschreibweise mit ?

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        boolean isActive = false;  
        String msg = isActive ? "I am active" :  
                                "I am NOT active";  
        System.out.println(msg);  
    }  
}
```

if-else-Anweisung

Kurzschreibweise mit ?

Kontrollfluss

```
public class Hello {  
  
    public static void main(String[] args) {  
        int hourOfDay = 9;  
        switch (hourOfDay) {  
            case 10:  
                System.out.println("Time to get up");  
                break;  
            case 12:  
                System.out.println("Breakfast");  
                break;  
            case 16:  
                System.out.println("Beer");  
                break;  
            default:  
                System.out.println("Sleep");  
        }  
    }  
}
```

switch-case-Fallunterscheidung

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        int hoursToSleep = 4;  
        for (int i = 0; i < hoursToSleep; i++) {  
            System.out.println(i + " more hours to sleep");  
        }  
    }  
}
```

for-Schleife

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        int hoursToSleep = 4;  
        while (hoursToSleep > 0) {  
            hoursToSleep--;  
            System.out.println(hoursToSleep + " more...");  
        }  
    }  
}
```

while-Schleife

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        int hoursToSleep = 4;  
        do {  
            hoursToSleep--;  
            System.out.println(hoursToSleep + " more...");  
        } while(hoursToSleep > 0);  
    }  
}
```

do-while-Schleife

Kontrollfluss

```
public class Hello {  
    public static void main(String[] args) {  
        int hoursToSleep = 4;  
        while(hoursToSleep > 0) {  
            hoursToSleep--;  
            System.out.println(hoursToSleep + " more...");  
            break;  
        }  
    }  
}
```

Schleife mit break, continue

Wie arbeitet ein Programm?

- Folge von Befehlen bzw. von Funktionen / Methoden / Prozeduren (Imperative / Prozedurale Programmierung)
- Verarbeitung von Variablen und Konstanten mit Deklaration und Initialisierung
- Springen in Schleifen oder je nach Fallunterscheidung
- Wie können wir Code wiederverwenden?

2.) Grundlagen: Hello World

3.) Funktionen

Funktionen

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

- Vorgegeben
- Aufruf mit Argumenten

Funktionen

```
public class Main {  
  
    public static void main(String[] args) {  
        int hoursToSleep = 4;  
        printSleep(hoursToSleep);  
    }  
  
    private static void printSleep(int hours) {  
        do {  
            hours--;  
            System.out.println(hours + " more...");  
        } while(hours > 0);  
    }  
}
```

Deklaration: eigene Funktionen mit Parameter

Funktionen

```
public static void main(String[] args) {  
    int hoursToSleep = 4;  
    System.out.println(printSleep(hoursToSleep));  
}  
  
private static String printSleep(int hoursToSleep) {  
    do {  
        hoursToSleep--;  
        return hoursToSleep + " more hours to sleep";  
    } while(hoursToSleep > 0);  
}
```

eigene Funktionen mit Rückgabewert **void** für keinen Rückgabewert)

Hello World: Zusammenfassung

- Folge von Befehlen bzw. von Funktionen / Methoden / Prozeduren (Imperative / Prozedurale Programmierung)
- Verarbeitung von Variablen und Konstanten mit Deklaration und Initialisierung
- Springen in Schleifen oder je nach Fallunterscheidung
- Code in Funktionen packen und aufrufen mit Argumenten