Technische Hochschule Rosenheim
Technical University of Applied Sciences

Objektorientierte Programmierung
Kapitel 1 – Einführung in Java

Prof. Dr. Kai Höfig

Inhalt

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Einführung**

- Komponenten der Java Plattform
  - Editionen
  - Virtuelle Maschine

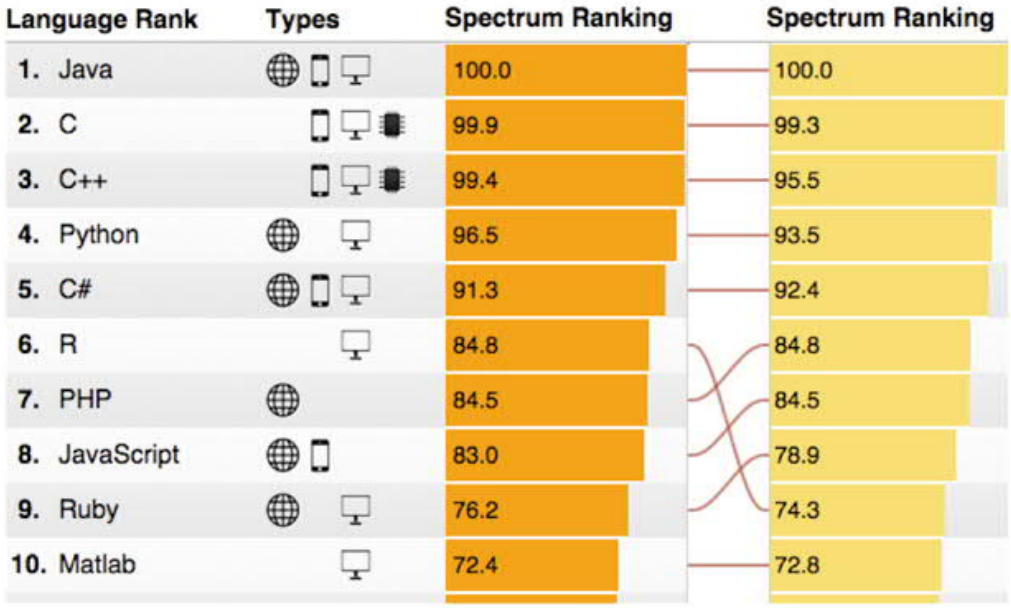
- Toolchain
  - Texteditor und Kommandozeile
  - **I**ntegrated **D**evelopment **E**nvironment (IDE)

- Syntax der Programmiersprache Java
  - Fokus: Unterschiede zu C

Zum Nachlesen: http://openbook.rheinwerk-verlag.de/javainsel/
Kapitel 1: Java ist auch eine Insel
Kapitel 2: Imperative Sprachkonzepte

Programmiersprachen: Top-Ten

Technische Hochschule Rosenheim
Technical University of Applied Sciences



Language Rank	Types	Spectrum Ranking	Spectrum Ranking
1. Java		100.0	100.0
2. C		99.9	99.3
3. C++		99.4	95.5
4. Python		96.5	93.5
5. C#		91.3	92.4
6. R		84.8	84.8
7. PHP		84.5	84.5
8. JavaScript		83.0	78.9
9. Ruby		76.2	74.3
10. Matlab		72.4	72.8

IEEE Spectrum Ranking basierend auf Metriken wie:

- # Github Repositories
- # Suchanfragen in Google
- …

Quelle: http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages, 08. März 2016

- Java ist sehr weit verbreitet!
- Java als Paradebeispiel einer objektorientierten Programmiersprache

Java Geschichte

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- 1990: Vorläufer „Oak“
  - Software für interaktives Fernsehen und Geräte der Konsumelektronik
  - Verlässlich, kompakt, Plattform-unabhängig
- 1993: „Oak“ wird Java
  - Weiterentwicklung für das World Wide Web
  - Fokus: Internet-Anwendungen (Vorläufer von Applets)
- 1996: Java JDK 1.0 wird freigegeben
  - Implementierung innerhalb des Netscape Navigators
- Klassenbibliothek wird ständig erweitert
- 2008: Android, das u.a. auf Java basiert, erscheint.
- 2009: Oracle übernimmt Sun
  - Aktuelle Version: Java SE 11.0.2(LTS), Release 15.01.2019

Grundeigenschaften von Java

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Objektorientierte Programmiersprache
  - Erlaubt Modularität, Wiederverwertbarkeit, etc.
- Ähnlichkeit zu zu C/C++
  - Aber Verzicht auf komplexe und fehlerträchtiger Konstrukte
- Automatisches Speichermanagement
  - *Garbage Collection* gibt selbstständig nicht mehr benötigten Speicherplatz frei.
  - Kein "`malloc`", kein "`free`"
- Plattformunabhängigkeit
  - Erzeugt Bytecode und keinen Maschinencode
  - Ausführung des Bytecodes innerhalb einer virtuellen Maschine.
- Weite Verbreitung von Java
  - Desktop
  - Webanwendungen
  - Android

Java Komponenten

Technische Hochschule Rosenheim
Technical University of Applied Sciences

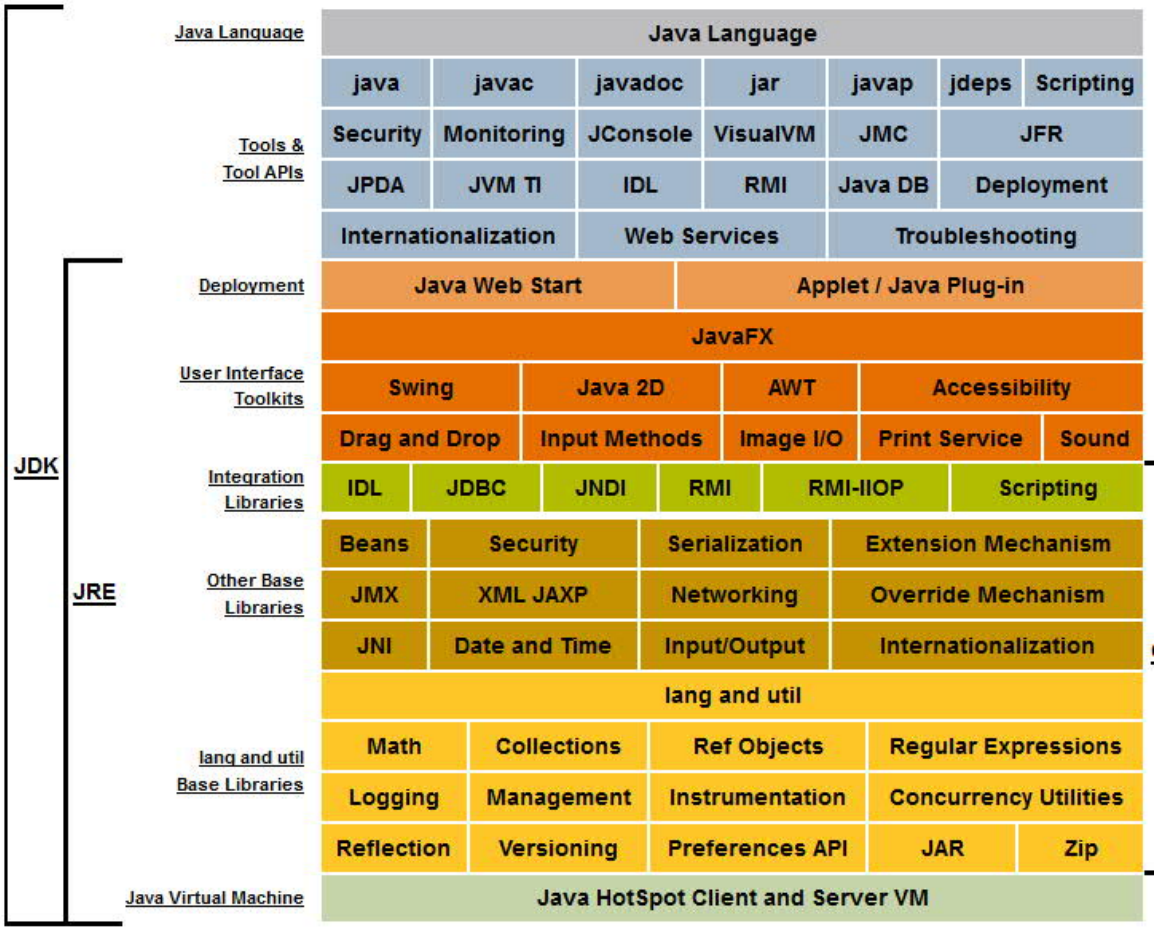
- **Programmiersprache Java**
  - Aktuell: Version 8
  - Version 9 erscheint voraussichtlich im Juli 2017

- **JRE: Java Runtime Environment** (dt. "Laufzeitumgebung")
  - Wird benötigt um Java-Programme auszuführen.
  - Übersetzt Bytecode in ausführbaren Maschinencode (Java Virtual Machine)
  - Windows: Einstellungen in Java Control Panel

- **JDK: Java Development Kit**
  - Wird benötigt, falls man selber Java programmiert und Java-Dateien in Bytecode übersetzen möchte.
  - Verschiedene Editionen, die sich im Umfang ihrer Bibliotheken unterscheiden:
    - Java Standard Edition (SE): Desktopeinsatz
    - Java Enterprise Edition (EE): Komponenten für Geschäftsanwendungen, Server
    - Java Micro Edition (ME): Für Geräte mit begrenzten Ressourcen, aktuell eher unbedeutend.

- Hinweis: Android erzeugt keinen Standard-Bytecode

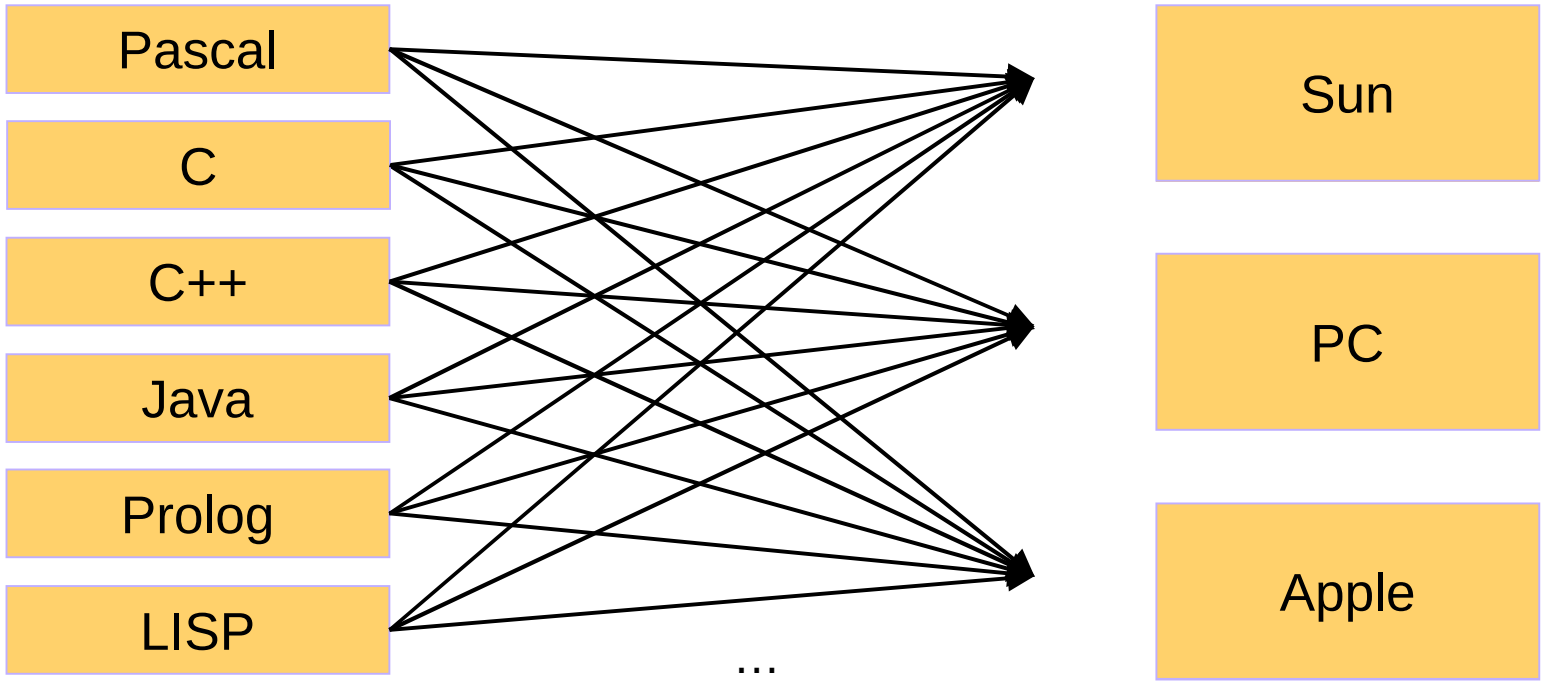
Java Standard Edition: Überblick



Quelle, 8. März 2016:
:https://docs.oracle.com/javase/8/docs/

Kompilierung: Traditioneller Ansatz

- Compiler übersetzt abstrakte Hochsprache in Maschinensprache, die auf einer bestimmten Plattform ausführbar ist.
- Problem: Viele verschiedene Rechnertypen und Sprachen

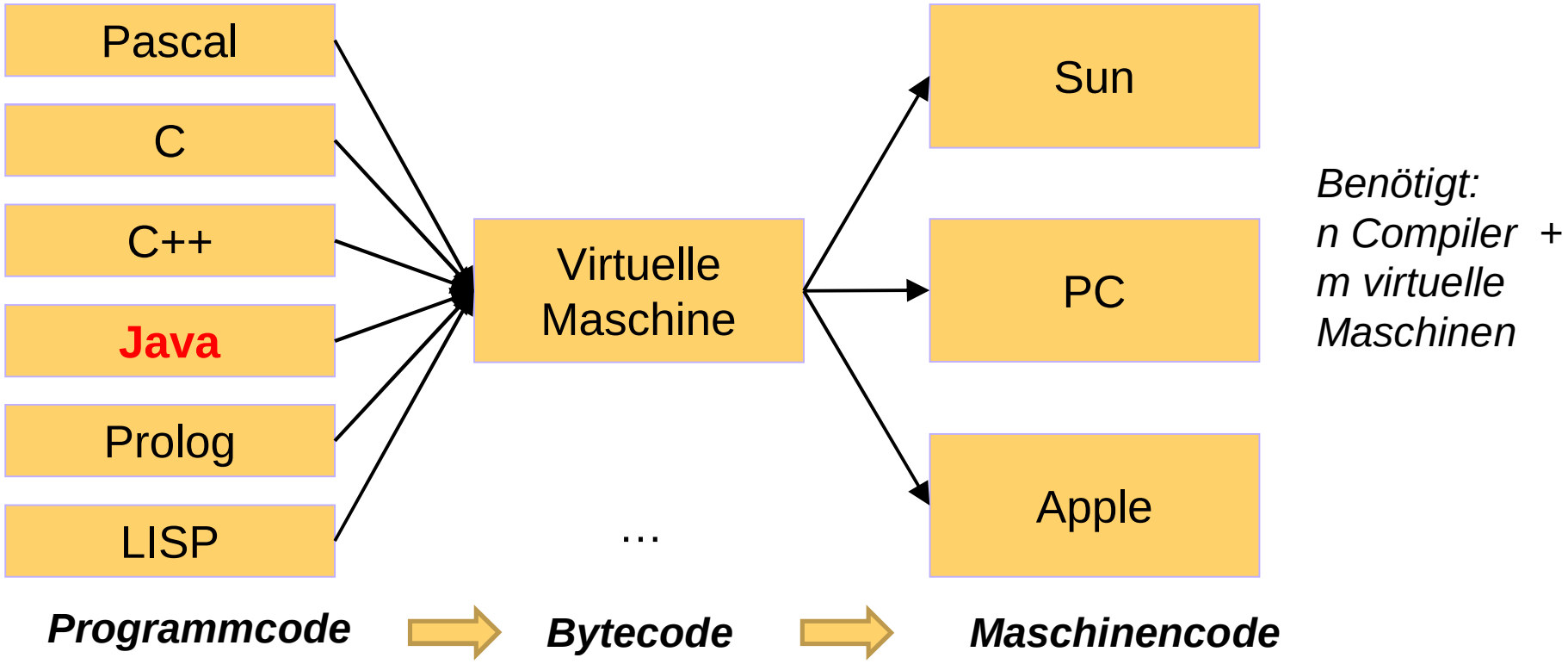


- Wie viele Compiler bei *n* Sprachen und *m* Plattformen? *n* * *m*!

Ansatz bei Java

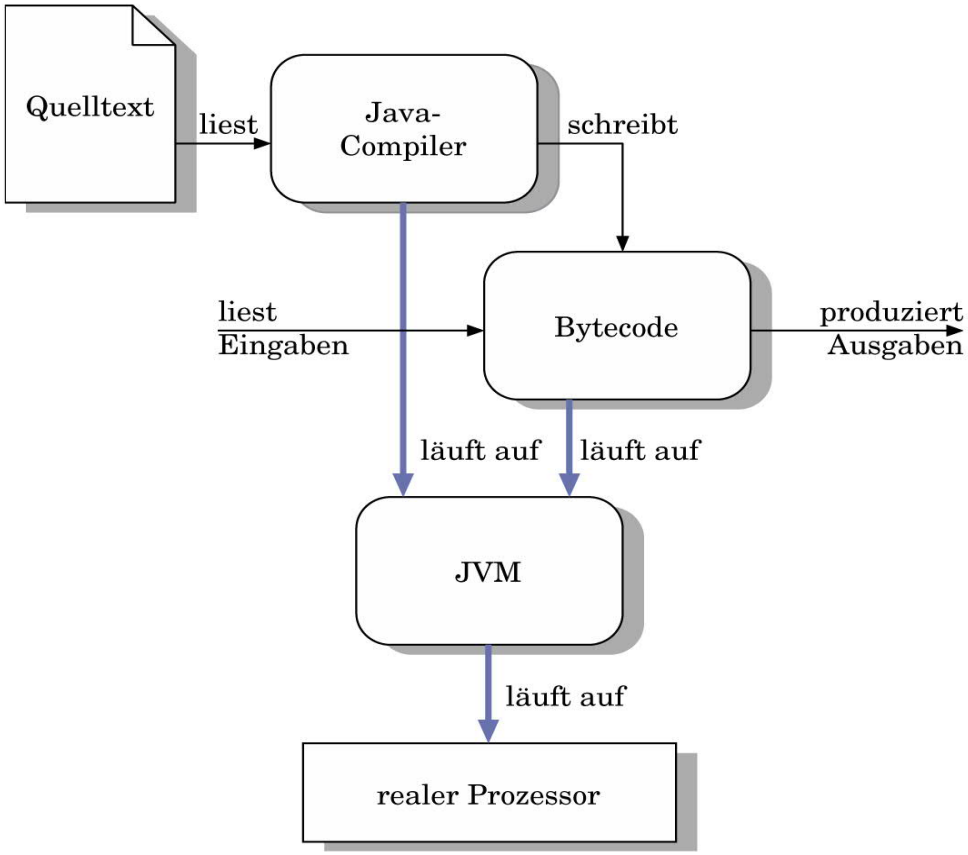
Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Compiler übersetzt Programm in Bytecode.
- Interpreter in virtueller Maschine führt Bytecode aus und übersetzt diesen (zur Laufzeit) in plattformabhängigen Maschinencode
- Virtuelle Maschine ist Bindeglied zwischen Hochsprache und Maschinensprache.
- Übersetzten Programme sind plattformunabhängig einsetzbar.



Benötigt:
n Compiler + m virtuelle Maschinen

Verarbeitungsmodell von Java



Quelle: Schiedermeier (2005), 14

- Vorteil des Java-Ansatzes:
  - Plattformunabhängigkeit
- Nachteil
  - Leicht höherer Ressourcenverbrauch, da zusätzlich zum Programm auch eine virtuelle Maschine läuft.

Erstes Java Programm

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Klassenname, sollte dem Namen der Datei entsprechen!

```
public class Quadrat {

    static int quadrat(int n) {
        return n * n;
    }

    public static void main(String[] args) {
        for (int i = 1; i <= 4; i++) {
            int result = quadrat(i);
            String s = "Quadrat von " + i +
                       ": " + result;
            System.out.println(s);
        }
    }
}
```

(Hilfs)funktion mit Integer als Parameter und Integer als Rückgabewert. Vorerst alle Funktionen mit "static" versehen.

Einstiegsmethode: Fester Name! Name darf nicht geändert werden!

Mit "+" können mehrere Strings zu einem String zusammengebaut werden.

Gibt String auf Konsole aus!

Der Syntax ist C-ähnlich!

Demo: Texteditor und Übersetzen mit Kommandozeile

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Quelltext: Quadrat.java

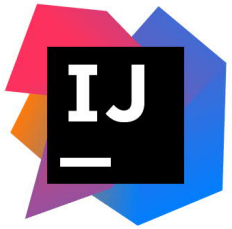
- Übersetzen in Bytecode mit:
  - `javac Quadrat.java`
  - Ergebnis: `.class`-Datei

- Ausführung des Bytecodes / des Programmes mit:
  - `java Quadrat`

Integrierte Entwicklungsumgebungen (IDE)

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Erleichtern die Arbeit des Programmierers erheblich
  - Debugging
  - Refactoring
  - Syntax Highlighting
  - …
- 3 Entwicklungsumgebungen für Java
  - **IntelliJ IDEA**
  - Eclipse
  - Netbeans
- Demo:
  - Einbinden von `Quadrat.java` in IntelliJ



Zentrale Eigenschaften von Java

Technische Hochschule Rosenheim
Technical University of Applied Sciences

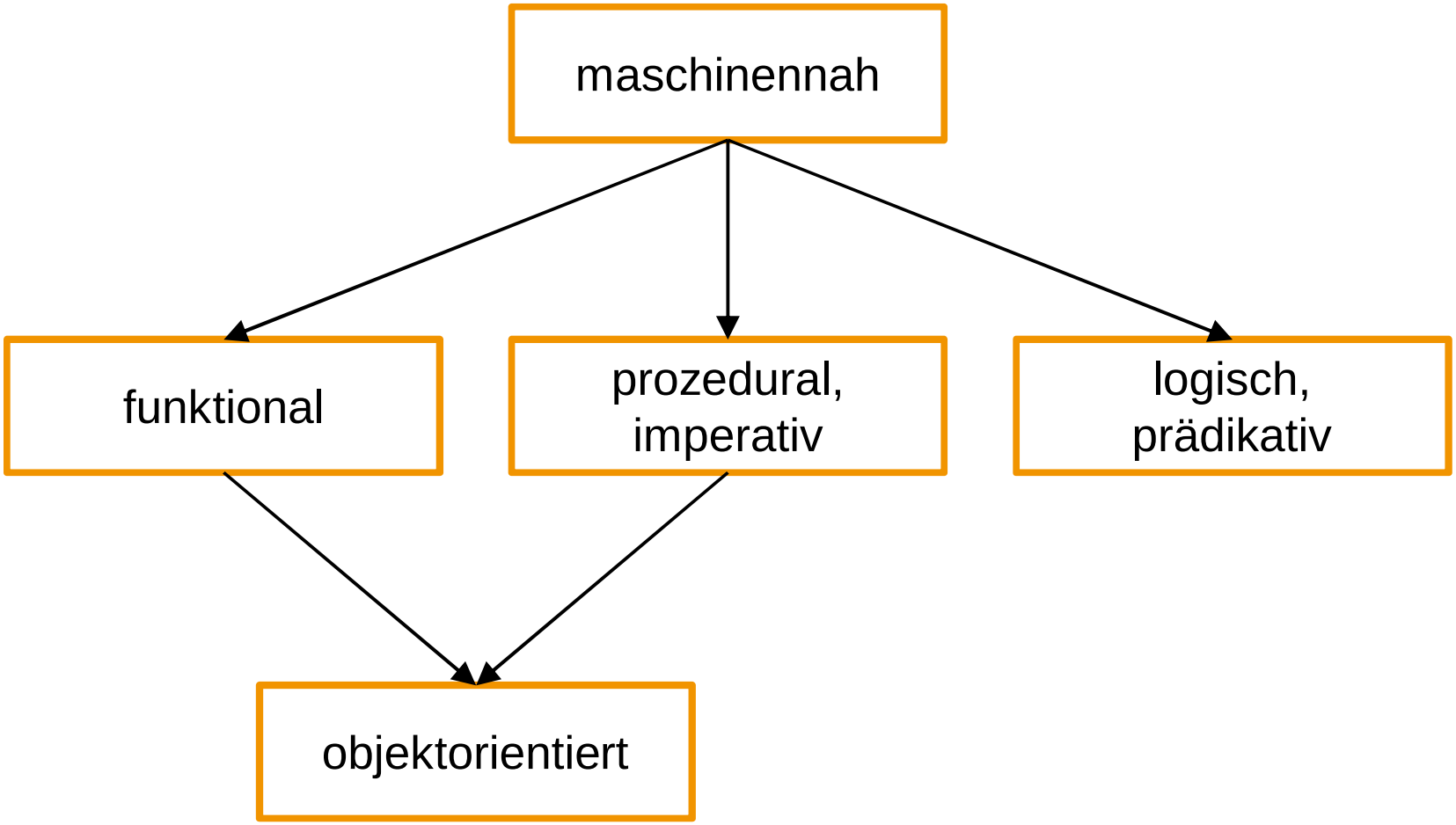
- Syntax ähnlich zu C und C++
- Kein Präprozessor
  - kein `#include` und kein `#define`
  - keine Header-Dateien
- Keine Strukturtypen (wie `struct` oder `union`)
  - Nur elementare Datentypen und Klassen
- Keine expliziten Zeiger
- Keine globalen Variablen
- Alles ist Teil einer Klasse, Java ist Objektorientiert
  - Vorerst: Maximal 1 Klasse

Programmierparadigmen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- ***Funktionale Programmierung***
  - Nur Funktionen und Rekursion; keine Zustände
  - `fac(n) = if n = 0 then 1 else n* fac(n-1) end`
- ***Imperative Programmierung***
  - Variablen, Attribute, Zustände, Steuerfluss
  - `k = n; r = 1; while k > 0  { r = r + 1;} …`
- ***Logische Programmierung***
  - Fakten, Prädikate, Invarianten, Deduktionsregeln
  - ```
    fac(0,1).
       fac(n+1, r) :- fac(n, r1), mult(n+1, r1, r)
    ```
- ***Objektorientierte Programmierung***
  - Verwendet Konzepte der imperativen Programmierung (`for-Schleife`)
  - Und der funktionalen Programmierung
  - Neue Konzepte: Objekte und Klassen
    ```

Programmierparadigmen



Strukturierung von Java-Programmen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Anweisungen** (kleinste Bausteine)
  - `int a = 5;`
  - `a = c + d;`
- **Blöcke** (Kollektion von Anweisungen)
  - Begrenzt durch geschweifte Klammern
  - Beispiel: for-Schleife
- **Methoden**
  - Menge von Anweisungen
  - Mehrere Eingabewerte
  - 1 Rückgabewert

```
public int add(int a, int b) {
    return a + b;
}
```

- **Kommentare**
  - //: Zeilenkommentar
  - /* … */: Abschnittskommentar

Rahmen für unsere ersten Java Programme

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public class Name {

    public static void main(String[] args) {
       // Hier muss programmiert werden
    }
}
```

- Unsere ersten Programme bestehen aus nur 1 Klasse (= "Programm")
- Für jede Klasse wird eine eigene Datei angelegt, deren Namen "`Name`" dem Dateinamen entspricht.
  - Dateiname = Klassenname + **`.java`**
- Konvention: Klassennamen beginnen mit Großbuchstaben.
- Es können beliebige Hilfsfunktionen hinzugefügt werden.

Kontrollstrukturen und Operatoren

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Anweisungen:** Ausdrücke mit ";" oder Blöcke oder Kontrollstrukturen

- **Verzweigungen**
  - if- und if-else
  - Switch

- **Schleifen**
  - while
  - do.. while
  - for
  - break, continue

- **Operatoren**
  - +, -, *, /, %
  - >, <, ==, !=, <=
  - !, &&, ||
  - ~, ^, >>, &, |
  - ++, --, +=, …

```
if(x > 0) a = x; // x nicht-negativ
else a = -x; // x negativ
```

```
int a;
switch(a) {   // mit break-Anweisungen !
    case 0: System.out.println("Null"); break;
    case 1: System.out.println("Eins"); break;
    default: System.out.println("default");
}
```

```
int n = 4;
int sum = 0;
for (int i = 1; i <= n; i++)
    sum += i;
System.out.println(sum);
```

IJ fori

Bezeichner

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Namen für vom Programmierer definierte Elemente
  - Beispiel: Variablen, Methoden

- **Korrekte Identifikatoren**
  - `counter`
  - `colorDepth`
  - `iso9660`
  - `xmlProcessor`
  - `MAX_VALUE`

- **Unzulässig**
  - `1stTry` Erster Buchstabe darf keine Ziffer sein
  - `Herz  Dame` Leerzeichen im Namen nicht erlaubt
  - `const` Reserviertes Wort
  - `muenchen-erding` Bindestrich im Namen nicht erlaubt

Empfehlungen zur Schreibweise

Technische Hochschule Rosenheim
Technical University of Applied Sciences

• Kleine Buchstaben für Variablen	`COUNTER`	`counter`
• Große Buchstaben für Konstanten	`max_value`	`MAX_VALUE`
• Neue Wortteile mit großen Buchstaben	`gettoken`	`getToken`
• Ganze Wörter	`c`	`counter`
• Aussagekräftige Namen	`o000oO`	`counter`
• Missverständliche Abkürzungen ausschreiben	`bup`	`binaryUpload`
• Gebräuchliche Akronyme in großen Buchstaben	`Html`	`HTML`

Schlüsselwörter in Java

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Braun: gibt's auch in C

Rot: gibt's auch in C++

abstract	do	if	package	synchronized
boolean	double	implements	private	this
break	else	import	protected	throw
byte	extends	instanceof	public	throws
case	false	int	return	transient
catch	final	interface	short	true
char	finally	long	static	try
class	float	native	strictfp	void
const (*)	for	new	super	volatile
continue	goto (*)	null	switch	while
default				

(*) const und goto sind zwar reservierte Wörter, aber ohne Verwendung

Primitive Datentypen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

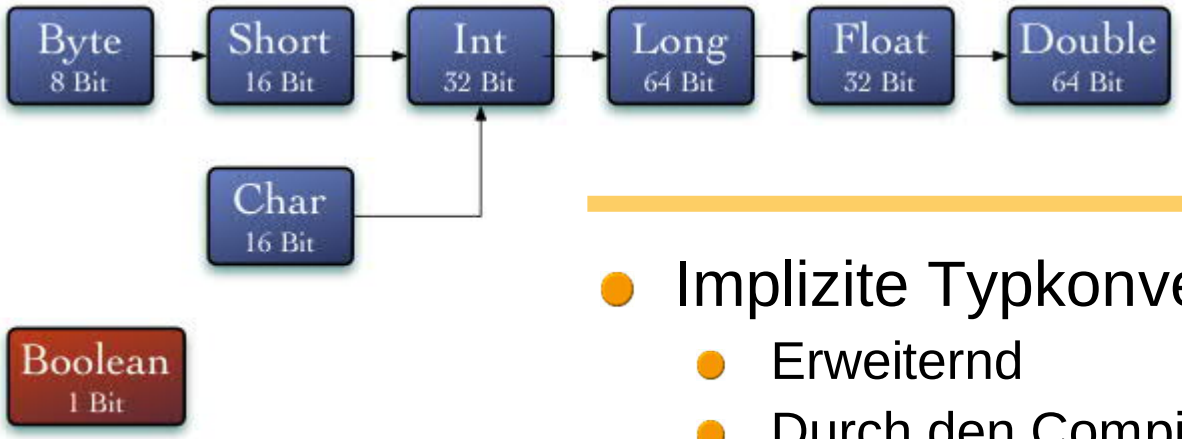
Datentyp	Bits	Format	Default	Beispiele
boolean	8		false	true, false
char	16	Unicode	'\u0000'	'a', '\u0000'
byte	8	signed	0	-128; +127
short	16	signed	0	-32768; 32767
int	32	signed	0	2147483647
long	64	signed	0	100000L
float	32	IEEE 754	0.0	0.1f
double	64	IEEE 754	0.0	0.1

```
final int N = 10;                  // Konstante
for(int i = 0; i < N; ++i)
    System.out.println(i);
```

➔ Außer primitiven Datentypen gibt es nur noch Objekte (Referenztypen) in Java!

➔ **Java kennt keine Zeiger!**

Typkonvertierung in Java



- Implizite Typkonvertierung
  - Erweiternd
  - Durch den Compiler
  - Verlustfrei

- Explizite Typkonvertierung
  - Einschränkend
  - Durch *cast*
  - *Genauigkeitsverlust*

Arrays

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Arrays erhalten ihre Länge bei der Initialisierung
  - **C**: `int[5] array;`
  - **Java**: `int[] array = new int[5];`

- Hinweise zur Länge des Arrays:
  - Länge (oben 5) muss nicht - wie in C - eine Konstante sein
  - Länge nach Initialisierung ist fest, kann nicht mehr verändert werden.
  - Abfrage möglich über Attribut `length`:
    - Beispiel: `int arrLength = array.length`

- **Mehrdimensionale** Arrays sind möglich:
  - ```
    String[][] ms = new String[4][4];
    ms[2][3] = "Otto";
    ```
    ```

Iterieren über Arrays

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Gegeben:

```
double[] array = {1.2, 3.0, 0.8};
```

- **Möglichkeit 1**

```
int sum = 0;
for (int i = 0; i < array.length; i++) {
// i indexes each element successively
    sum += array[i];
}
```

itar

- **Möglichkeit 2**

```
int sum = 0;
for (double d : array){
    // d gets each value successively in array
    sum += d;
}
```

iter

Strings, Ein- und Ausgabe auf der Konsole

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **String**
  - Kein primitiver Datentyp, sondern eine Klasse (siehe später)
  - Einfache Konkatenation mit "+„

```
String s1 = "Hallo";
String s2 = "Griasdi";
String s3 = s1 + s2;    // Ergebnis: "HalloGriasdi"
```

- **Ausgabe auf der Konsole**

```
System.out.println("Hallo" + s2);
```

- **Einlesen der Benutzereingabe (z.B. int, String)**
  - Etwas komplizierter!

```
Scanner scanner = new Scanner(System.in);
String s = scanner.next();  // liest String ein
int a = scanner.nextInt();  // liest int ein
```

Übung: Was ist falsch?

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Welche 3 (Syntax)fehler wurden gemacht?
- Was zeigt die Ausgabe, wenn die Fehler korrigiert werden?

```
bool[3] bArray;
public class Fehler {

    public static void main(String[] args) {
        bArray[0] = false;
        bArray[1] = true;
        bArray[2] = bArray[0] && bArray[1];
        System.out.println("Result of bArray[2] is: " + bArray[2]);
    }
}
```