Java-Programmierkurs

Einführung

WS 2012/2013

Prof. Dr. Margarita Esponda

Inhalt

- Geschichtliche Einführung
- Grundlagen der Imperativen Programmierung mit Java
- Objektorientiertes Programmieren in Java

  Klassen, Objekte, Nachrichten, Kapselung, Vererbung

  und Polymorphie
- Ausnahmebehandlung
- Graphische Benutzeroberflächen
- Ereignisbehandlung
- Nebenläufige Prozesse (*Threads*)
- Projekt

Freie Universität Berlin

Literatur

Gries D., Gries P.: "Multimedia Introduction to Programming Using Java"
Springer-Verlag. 2005.

- Kathy Sierea & Bert Bates.: "Head First Java". 2nd Edition. O'Reilly. 2005.

- Robert Sedgewick, Kevin Wayne.: "Introduction to Programming in Java".
An Interdisciplinary Approach. Princeton University. 2

Freie Universität Berlin

Termine

Zeit	Montag bis Freitag
10 - 12:30 Uhr	**Mo**. 25.2. und **Fr**. 15.3.
14 - 18 Uhr	Rechnerräume

Freie Universität Berlin

Tutoren

Tutor	Raum
Jim Neuendorf	**K36** (Takustr. 9)
Roman Schulte-Sasse	**K38** (Takustr. 9)
Amjad Saadeh	**K44** (Takustr. 9)

Freie Universität Berlin

Kriterien für den Leistungsnachweis:

- Eine regelmäßige Teilnahme an den Übungsterminen ist unerlässlich.
- Die Abgabe der Übungsblätter erfolgt täglich bis spätestens 18:00 Uhr.
- Jede Studentin und jeder Student muss an mindestens n-1 Übungsterminen anwesend sein.
- Jede Studentin und jeder Student muss 60% der maximal erreichbaren Punktzahl aus allen Übungsblättern erreichen.
- Die Note ergibt sich aus der Punktzahl der Übungen.

Freie Universität Berlin

Vorlesungen & Übungen

Grundlegende Regeln zum Erfolg sind:

1. Regelmäßige und aktive Teilnahme in der Vorlesung

   Fragen stellen!

2. Regelmäßige und aktive Teilnahme an den Übungen

   Selbständige Lösung der Übungsblätter!

Freie Universität Berlin

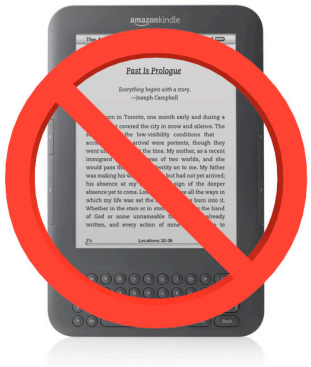
NO LAPTOPS

NO iPADS

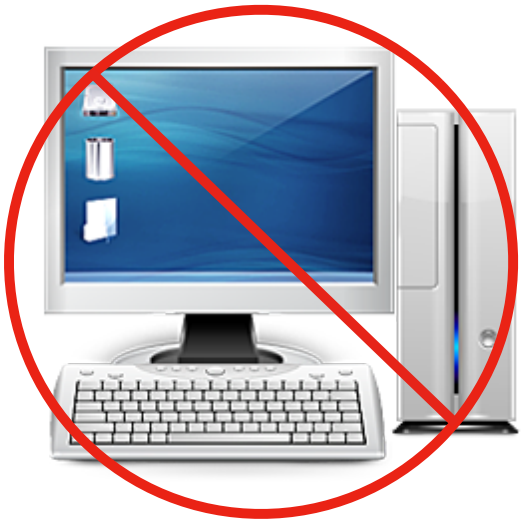
NO MUSIK-Geräte

NO HANDYS

NO KINDLE



NO DESKTOPS



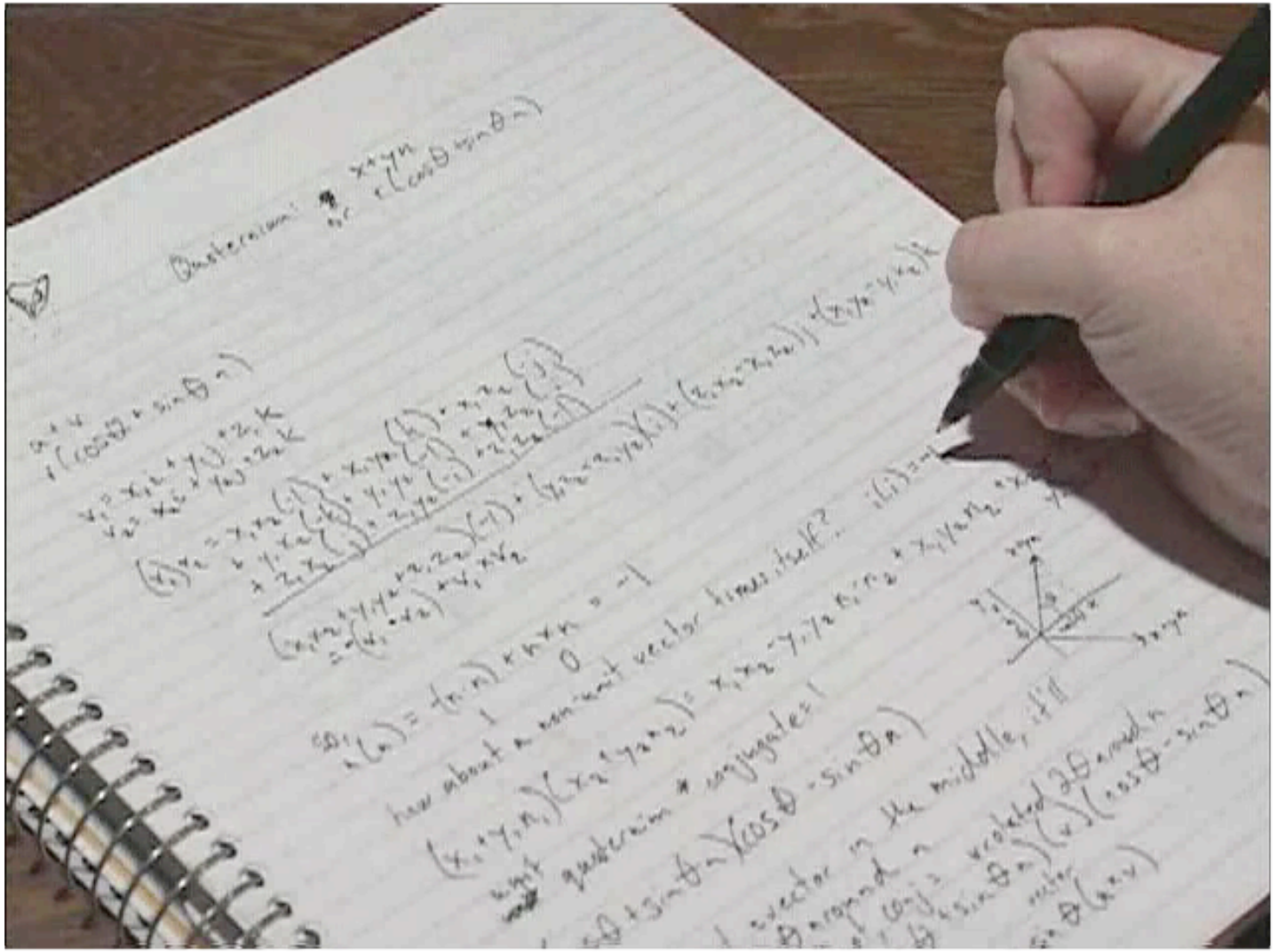
Warum ist Mitschreiben wichtig?

Das Mitschreiben hat folgende Vorteile:

- Es erhöht die Aufmerksamkeit und fördert die Konzentration
- Es regt dazu an, sich aktiv zu beteiligen
- Es individualisiert den Lernstoff und bildet oft eine gute Grundlage für die Implementierung der Übungen.

Fragen zuerst aufzuschreiben hat viele Vorteile.

Attack of the Note Sheep



Freie Universität Berlin

Fragen

Inhaltliche Fragen sind immer willkommen

- während und nach jeder Vorlesung
- in der Sprechstunde (Freitags 12 -14)
- während der Übungen am Rechner

Warum **Java**?

Warum nicht **Haskell**?

Wie so nicht **F#**?

Warum nicht **Dart** oder **Ceylon** oder **Rust**?

Wie viele Programmiersprachen?

Es gibt mehr als **2000** Programmiersprachen, die im kommerziellen Bereich bekannt sind oder gewesen sind.

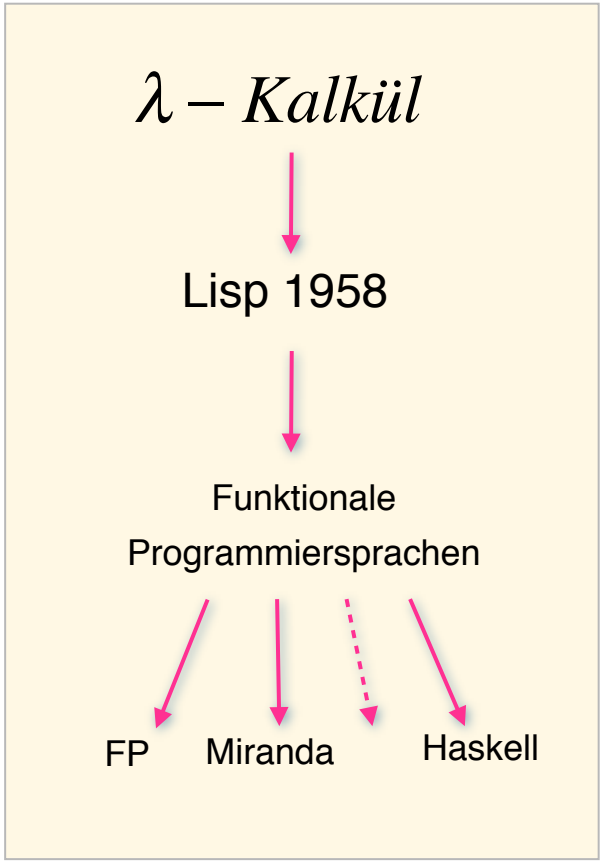
- mehr als **200** davon haben sich im Laufe der Geschichte stärker verbreitet
- (einige) Firmen entwickeln eigene private Sprachen
- es gibt viele akademische Programmiersprachen
- und es kommen immer mehr Programmiersprachen hinzu

30er Jahre

Erste Programmiersprache

Alonzo Church
1903-1995

Stephen C. Kleene
1909-1994



Lambda-Kalkül

- universelle abstrakte Programmiersprache
- nur die Hardware fehlte

30er Jahre

Alan Turing (1912-1954)

"On computable numbers, with an application to the Entscheidungsproblem".

1936

Turing-Maschine

Ein Modell, um die Klasse der intuitiv berechenbaren Funktionen zu bilden.

Freie Universität Berlin

Erste Programmgesteuerte Maschine

Lochkarten-Steuerung der Jacquard-Maschine

Jacquard Looms

1801

Programm

Programmiersprachen

- Eine ***Programmiersprache*** ist ein durch einen Rechner interpretierbarer Formalismus mit eindeutig definierter Syntax und Semantik.
- Ein ***Programm*** ist die Formulierung eines Algorithmus in einer konkreten *Programmiersprache*.
- Ein OO-Programm ist ein System kooperierender Objekte

Programmiersprachen

Die Syntax einer Programmiersprache regelt die erlaubten Kombinationen von Symbolen.

Die Semantik einer Programmiersprache ist die Definition der den zulässigen Programmen *zugeordneten Bedeutungen*.

Programmiersprachen vs. Menschensprachen

- Programmiersprachen sind viel einfacher zu lernen als natürliche Sprachen.
- Programmiersprachen haben eine einfache und eindeutige Syntax. Die Syntax natürlicher Sprachen ist viel schwieriger.
- Programmiersprachen haben vor allem eine eindeutige Semantik, natürliche Sprachen dagegen nicht!

Freie Universität Berlin

Was ist ein Programm?

Imperative Programmiersprachen

- Ein ***Programm*** ist eine Folge von Anweisungen, die auf einer Maschine ausgeführt werden können.
- Ein ***Programm*** ist die Formulierung eines Algorithmus in einer konkreten *Programmiersprache*.
- Ein **OO**-Programm ist ein System kooperierender Objekte.

50er Jahre

Imperative Programmiersprachen

Maschinensprache

0101000101101011
0101001011110101
0111000100100011
0101010001010110
0010001101000001
0000000101101011
....

Assembler-Programmiersprache

```
LOAD   #1  C
LOAD   #2  B
MULT   #1  #2  #3
LOAD   #4  A
ADD    #3  #4  #1
STORE  #1  C
```

Höhere Programmiersprachen

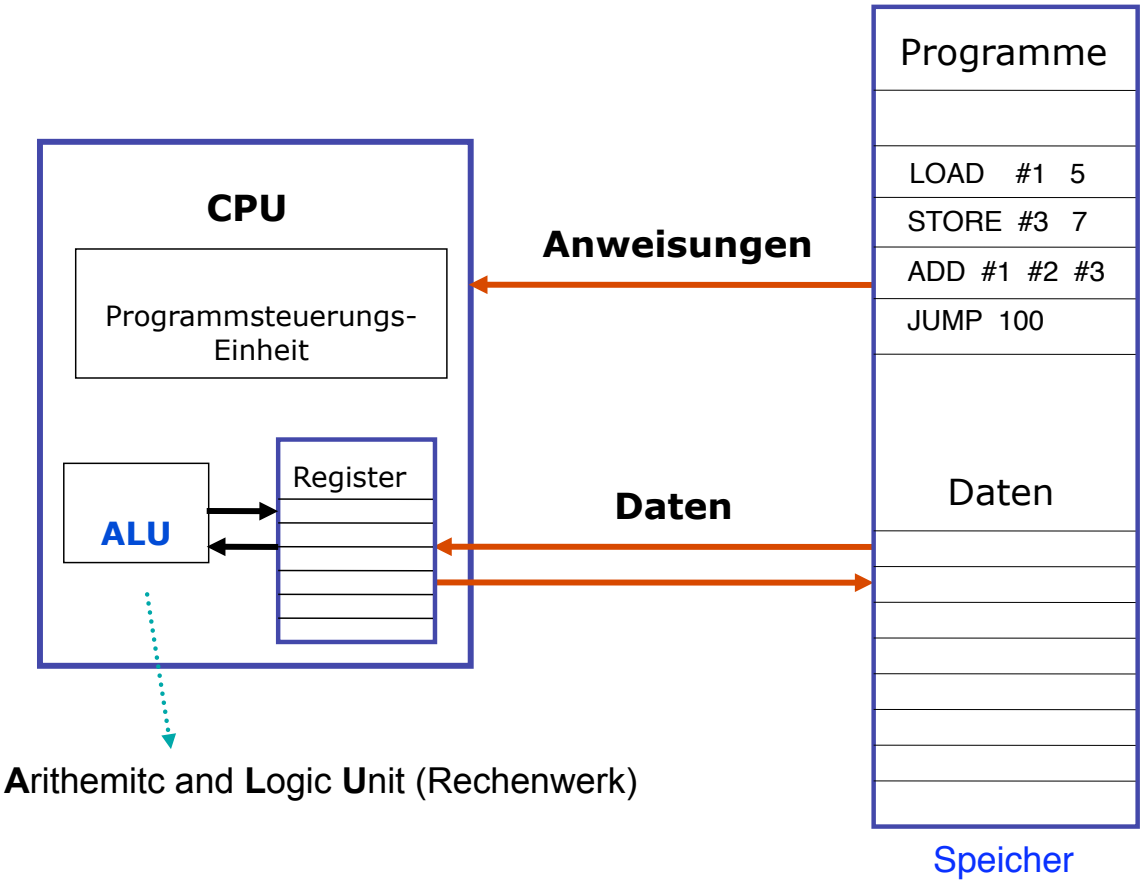
```
C  =  A + B*C
```

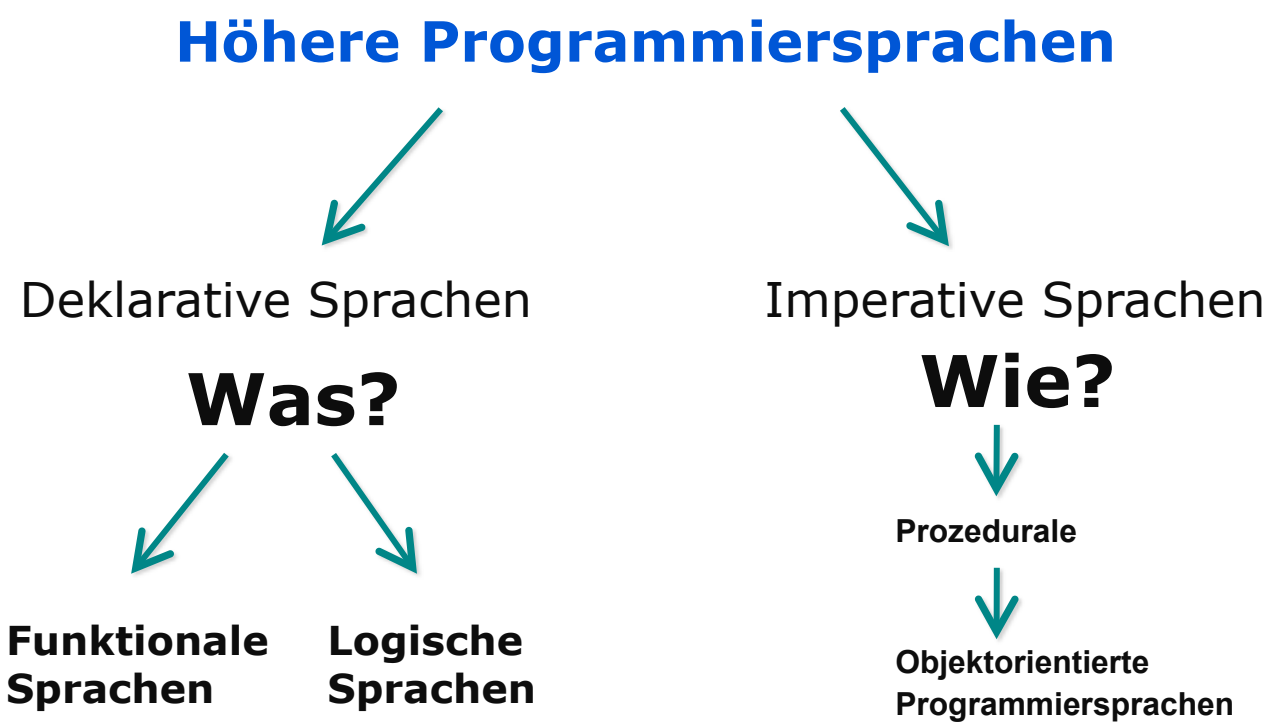
Freie Universität Berlin

Imperatives Programmieren

- Ältestes und populärstes Programmierparadigma
  - Fortran, Cobol, Algol, C, Basic, Pascal, Modula, ...
  - Simula, Smaltalk, C++, Java, C#, usw.
- Widerspiegelt die dahinter stehende Hardware-Architektur
  - von Neumann Maschine
  - gespeichertes Programm + Daten

Imperative Programmiersprachen




Höhere Programmiersprachen
Deklarative Sprachen
Was?
Funktionale Sprachen
Logische Sprachen
Imperative Sprachen
Wie?
Prozedurale
Objektorientierte Programmiersprachen

Freie Universität Berlin

deklarativ vs. imperativ

- Sprachen basieren auf einem **mathematischen Formalismus**
- Wissen über ein Problem rein deklarativ darstellbar
- Intelligentes System, das Fragen an das Programm beantworten kann

kompakter

robuster

einfacher

- Sprachen sind von der dahinterstehenden Hardware geprägt
- **Befehlssequenz (Zustände)**
- zeitlicher Ablauf im Programm sichtbar

effizienter

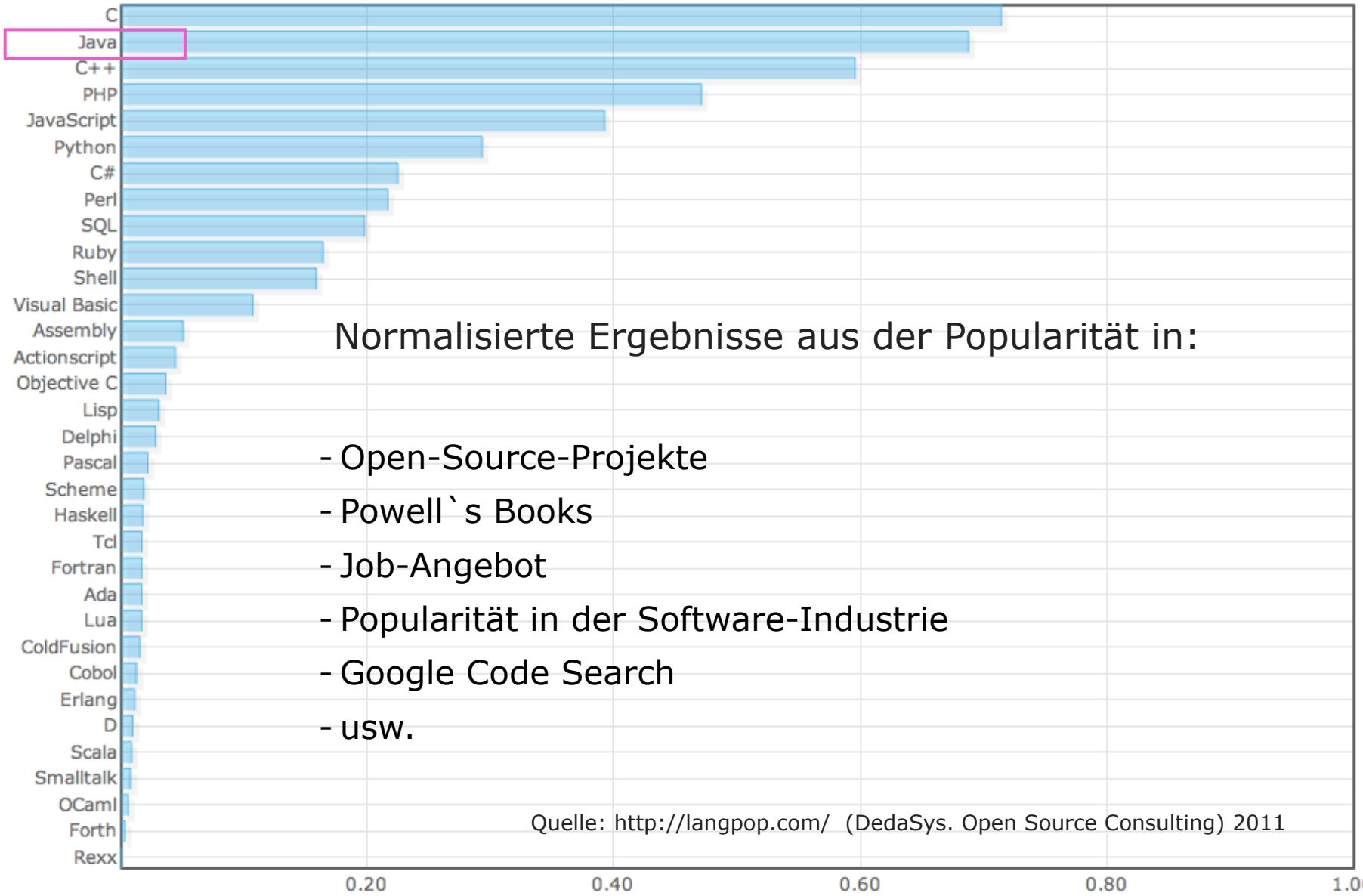
Warum Java?

Beliebteste Programmiersprachen

TIOBE Programming Community Index (Februar 2013)

Position Feb 2013	Position Feb 2012	Delta in Position	Programming Language	Ratings Feb 2013	Delta Feb 2012	Status
1	1	=	Java	18.387%	+1.34%	A
2	2	=	C	17.080%	+0.56%	A
3	5	↑↑	Objective-C	9.803%	+2.74%	A
4	4	=	C++	8.758%	+0.91%	A
5	3	↓↓	C#	6.680%	-1.97%	A
6	6	=	PHP	5.074%	-0.57%	A
7	8	↑	Python	4.949%	+1.80%	A
8	7	↓	(Visual) Basic	4.648%	+0.33%	A
9	9	=	Perl	2.252%	-0.68%	A
10	12	↑↑	Ruby	1.752%	+0.19%	A
11	10	↓	JavaScript	1.423%	-1.04%	A
12	16	↑↑↑↑	Visual Basic .NET	1.007%	+0.21%	A
13	13	=	Lisp	0.943%	+0.04%	A
14	15	↑	Pascal	0.932%	+0.12%	A
15	11	↓↓↓↓	Delphi/Object Pascal	0.886%	-1.08%	A
16	14	↓↓	Transact-SQL	0.773%	-0.07%	A–
17	75	↑↑↑↑↑↑↑↑↑↑	Bash	0.741%	+0.61%	A–
18	26	↑↑↑↑↑↑↑↑	MATLAB	0.648%	+0.15%	B
19	24	↑↑↑↑↑	Assembly	0.640%	+0.12%	B
20	19	↓	Ada	0.631%	0.00%	B

In der realen Welt verwendete Sprachen



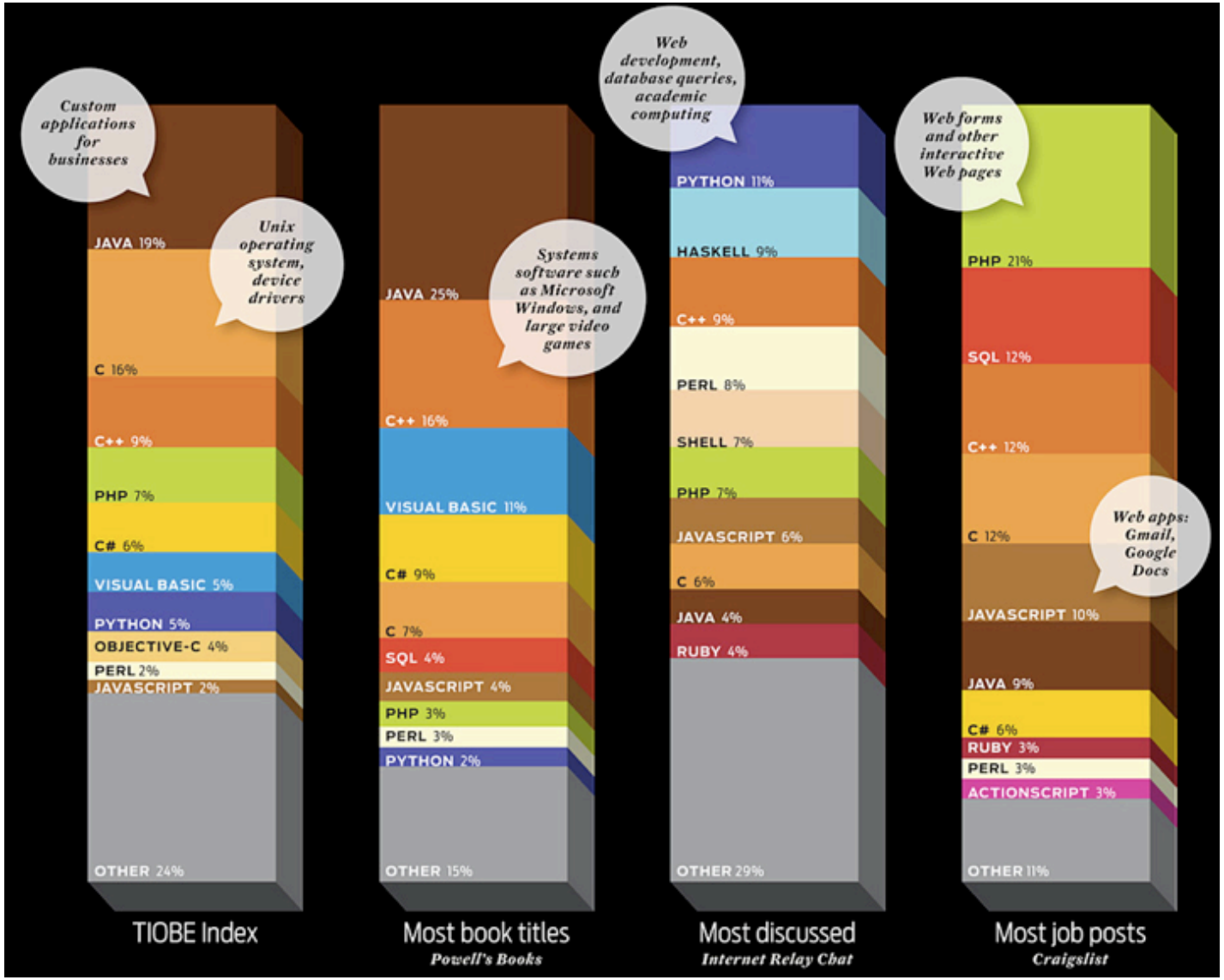
Normalisierte Ergebnisse aus der Popularität in:

- Open-Source-Projekte
- Powell`s Books
- Job-Angebot
- Popularität in der Software-Industrie
- Google Code Search
- usw.

Quelle: http://langpop.com/ (DedaSys. Open Source Consulting) 2011

Warum Java?

- Android-Anwendungen
- mehr als 700.000



Warum Java?

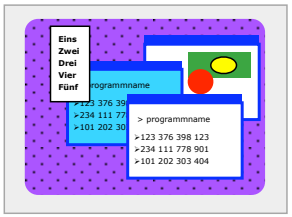
- gute Implementierung der wichtigsten objektorientierten Konzepte
- nach 17 Jahren sehr robust.
- große Portabilität durch die Java Virtuelle Maschine
- sicherer als C/C++, durch eingeschränkte Verwendung von Pointers und Pointer-Arithmetik und automatische Speicherverwaltung (GC)
- Integriertes Sicherheit-Model. *Class-Loader* und *Security-Manager*
- Große Anzahl von vorhandenen Standard-Klassen (ca. 4000)
- Standard-Bibliotheken für die Programmierung von graphischen Benutzerbibliotheken GUIs.
- Integrierte Unterstützung der Programmierung von verteilte Anwendungen
- sehr effizient geworden (*Just-in-time*-Übersetzung JIT)
- Nebenläufigkeit (*Multithreading*) in der Sprache integriert.
- breite Verwendung

OO-Konzepte können mit anderen Programmiersprachen gelernt werden.

- Eigentlich ist egal welche Sprache.
- Die wichtigsten Konzepte und Algorithmen sind sprachunabhängig.
- Java ist keine perfekte Sprache und auch nicht die beste.
- Es kommen mit Sicherheit noch bessere Programmiersprachen

Warum objektorientiertes Programmieren?

- Simulationsprobleme
  - in der Welt läuft alles parallel
- **Wiederverwendbarkeit** von Software
  - saubere Modularisierung der Software
  - mit klaren Schnittstellen
- Graphische Benutzeroberflächen
  - nicht sequentielle, interaktive Steuerung von Anwendungsprogrammen



- Programmierung verteilter Anwendungen
  - Parallelität und Kommunikation auf natürliche Weise

Was ist Java?

1991 Patrick Naughton und James Gosling

Programmierumgebung **Oak** bei Sun Microsystems

um Anwendungen für elektronische Geräte und das interaktive Fernsehen leicht zu programmieren.

1992

Green Projekt

Oak Programmierumgebung

keine Erfolg!

Was ist Java?

1996 veröffentlichte *Sun Microsystems* die erste offizielle Entwicklungsumgebung für Java.

2007 Free and Open Source Software. GNU General Public License (GPL)

Sun Microsystems → Oracle

Beispielloser Erfolg:

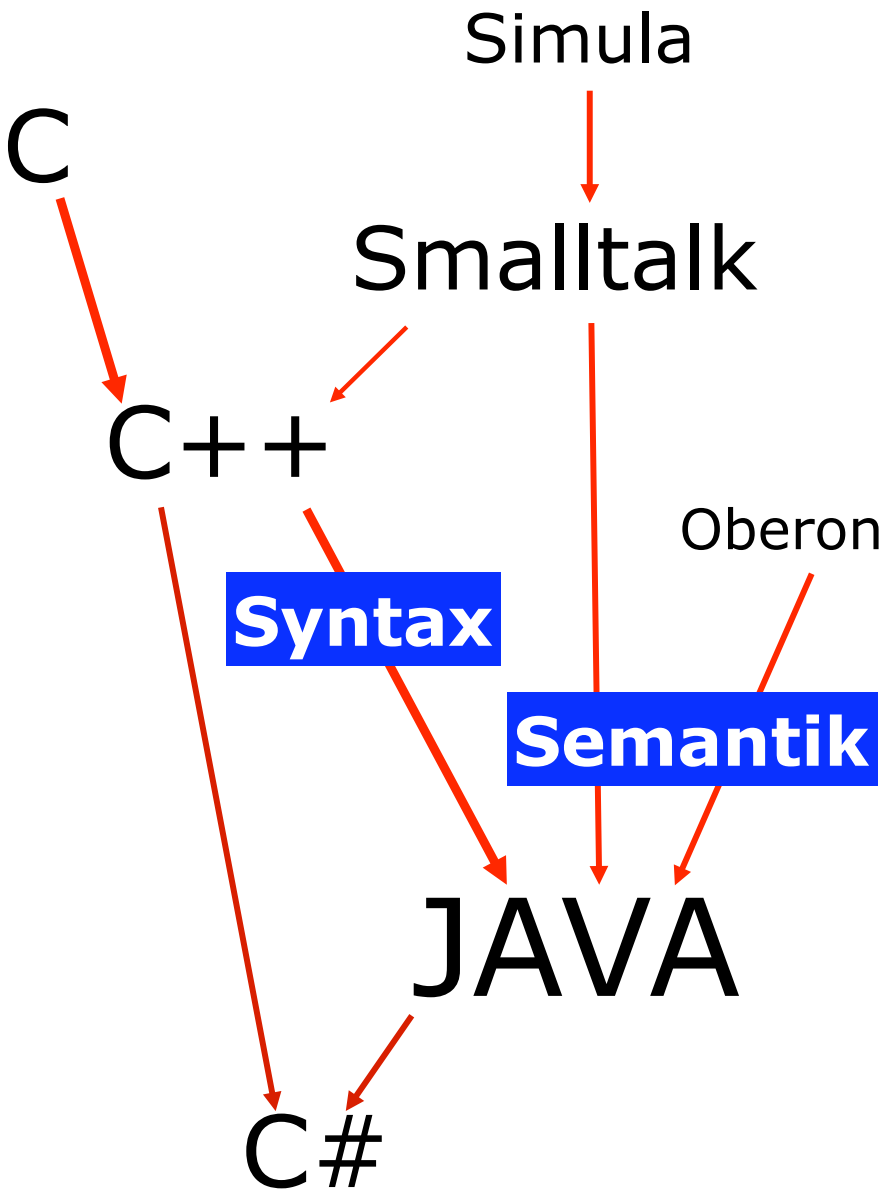
Noch nie hat eine neue Programmiersprache in so kurzer Zeit so starke Verbreitung gefunden.

Java ist 17 Jahre alt

Es gibt 930 Millionen *downloads* der *Java Runtime Environment* pro Jahr

3 Billionen Mobile-Geräte haben eine JVM Java Virtuelle Maschine

Stammbaum von Java



Objektorientiertes Programmieren

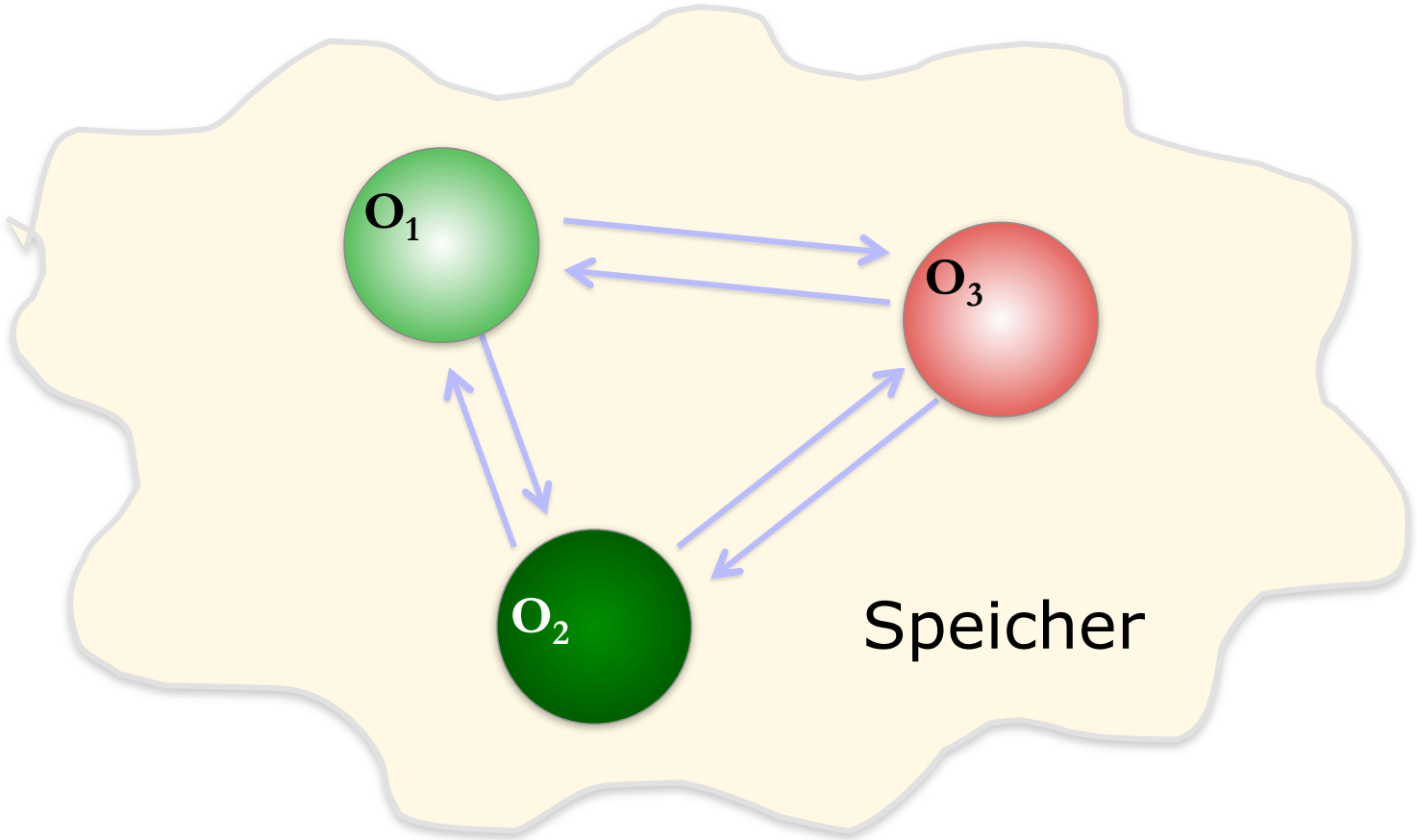
- Vorgänge der realen Welt
  - inhärent paralleles Ausführungsmodell
- Trennung von Auftragserteilung und Auftragsdurchführung
  - klar definierte Schnittstellen
- Klassifikation und Vererbung
  - Anpassbarkeit, Klassifikation und Spezialisierung von Programmteilen

Konzepte objektorientierter Programmierung

- Objekte
- Klassen
- Nachrichten
- Verkapselung
- Vererbung
- Polymorphismus

OO-Programmausführung

Die OOP betrachtet eine Programmausführung als ein System kooperierender Objekte



statisch

Was ist eine Klasse ?

Eine Klasse ist ein Bauplan, um Objekte einer bestimmten Sorte zu erzeugen.

Ohne Klassen gibt es keine Objekte in Java!

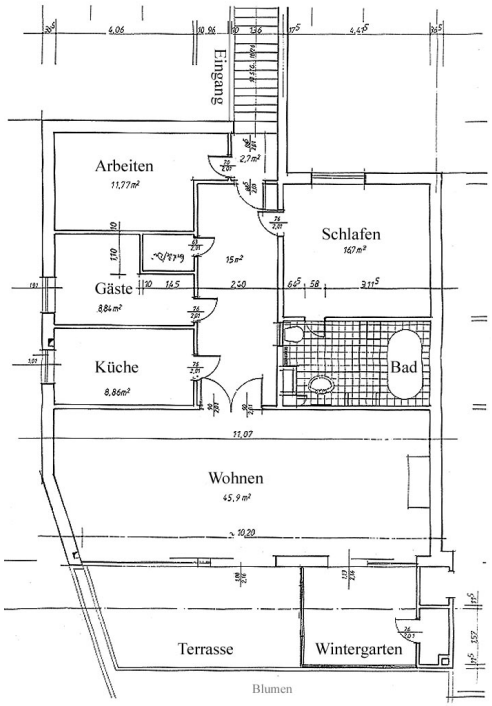
Klassen besitzen den Vorzug der Wiederverwendbarkeit.

statisch

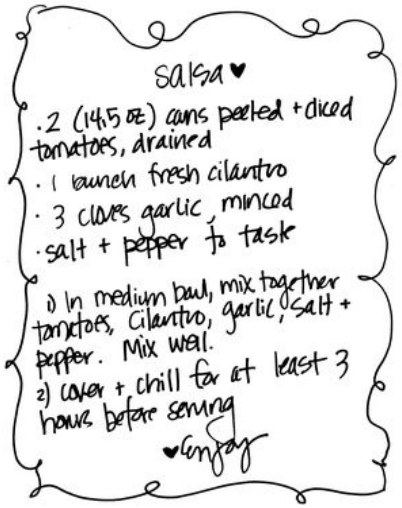
Was ist eine Klasse ?

Stempel

Bauplan



Rezept



dynamisch

Was ist ein Objekt ?

Ein Objekt ist ein Softwarebündel

aus **Variablen** und mit diesen Variablen zusammenhängenden **Methoden.**

Ein Objekt ist eine konkrete Ausprägung bzw. eine Instanz einer Klasse.

Jedes Objekt „weiß“, zu welcher Klasse es gehört.

dynamisch

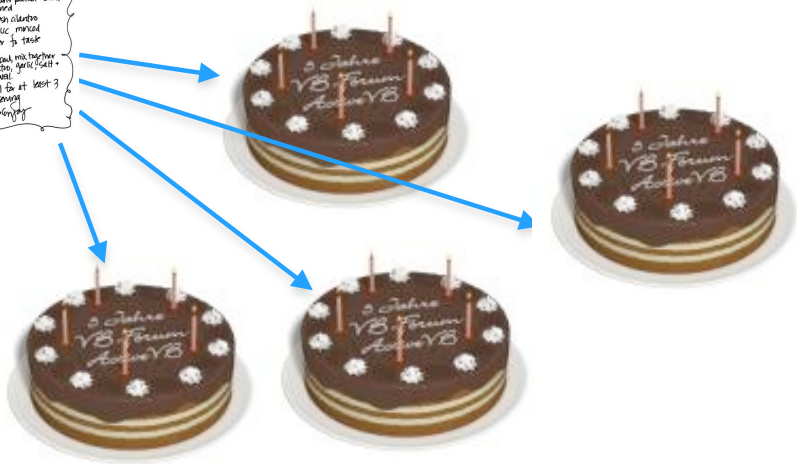
Was sind Objekte?

Klasse

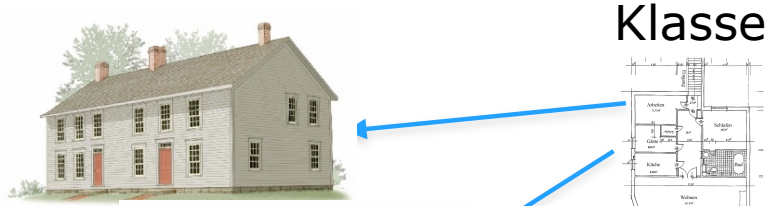
Objekte

Klasse

Objekte

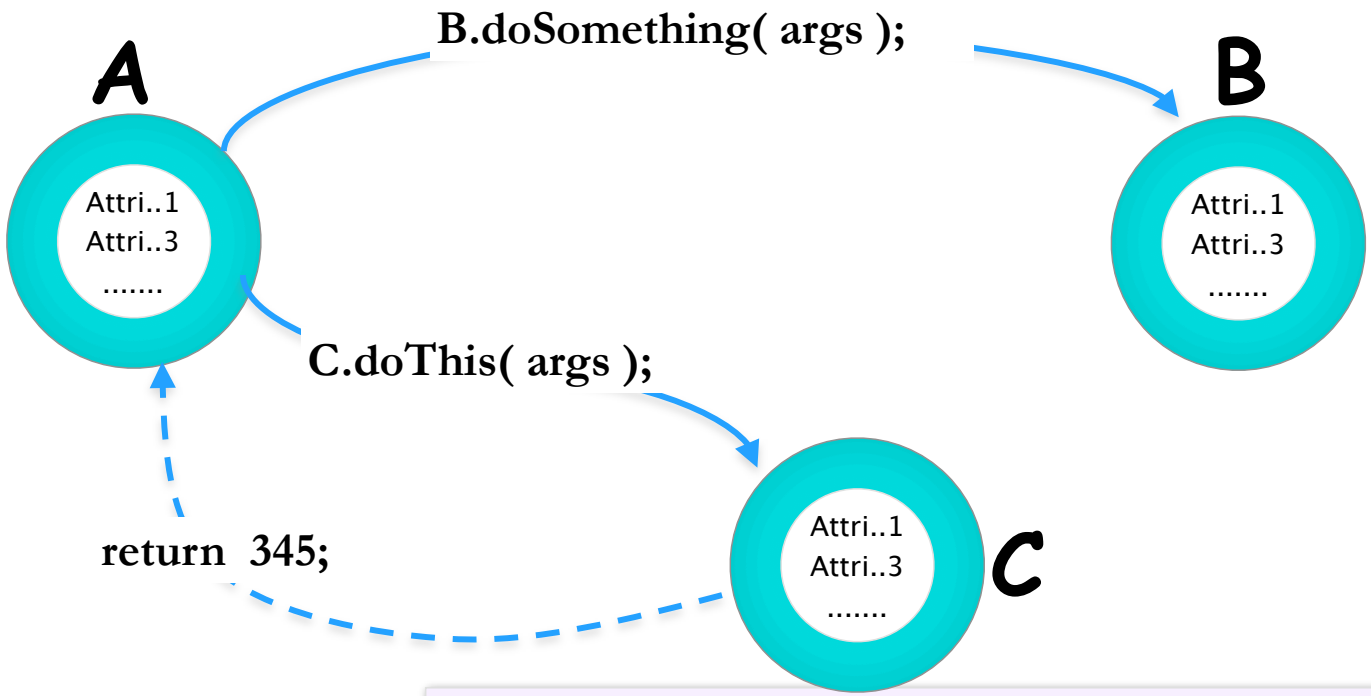


Klasse



Objekte

Was sind Nachrichten ?



Methodenaufrufe

- Empfänger
- Name der auszuführenden Methode
- Parameter

Klasse-Definition

Attribute:

- *Eigenschaft$_1$*
- *Eigenschaft$_2$*

. . . .

Verhalten:

- *Methode$_1$*
- *Methode$_2$*
- *Methode$_3$*

. . . .

Beispiel: Katze-Klasse

Attribute :

- Name
- Besitzer
- Farbe
- hungrig

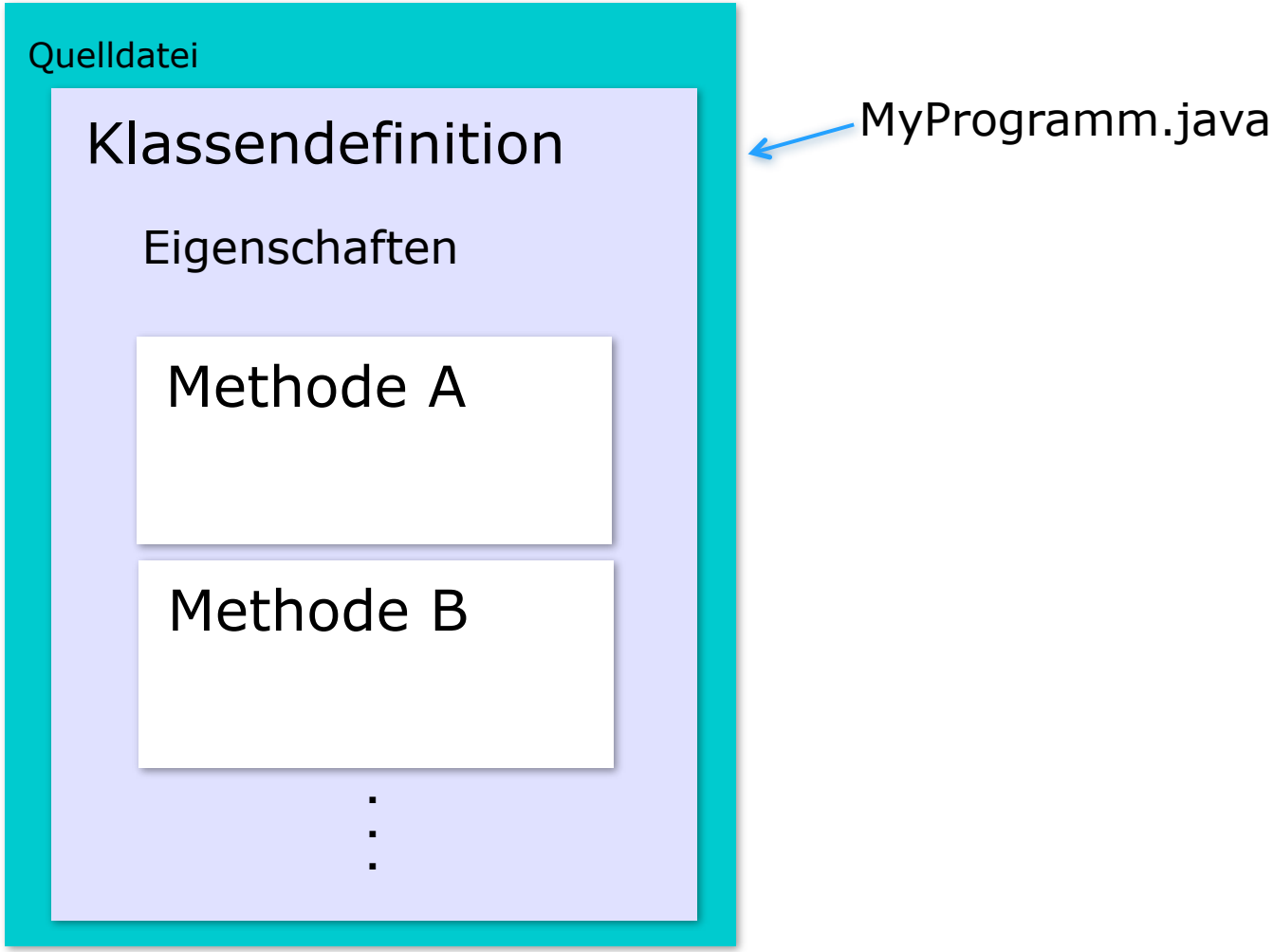
. . . . .

Verhalten:

- isst
- läuft
- kratzt
- schläft

. . .

Grundstruktur eines Java-Programms



Modellierung von Rechteck-Objekten

Eigenschaften

(x, y)

Operationen

- Fläche
- Umfang
- verkleinern
- vergrößern
- verschieben
- klonen

Modellierung von Rechteck-Objekten

```
public class Rechteck {
   // Eigenschaften
        . . .
        . . .
   // Konstruktoren
        . . .
        . . .
   // Operationen
        . . .
        . . .
        . . .
}
```

Dateiname:

Rechteck.java

Java-Anwendung

Rechteck.java

```
class Rechteck {
  // Attribute
    . . . .
  // Konstruktoren
    . . . .
  // Methoden
    . . . .
    . . . .
   . . . .
}
```

Kreis.java

```
class Kreis {
  // Attribute
    . . . .
  // Konstruktoren
    . . . .
  // Methoden
    . . . .
    . . . .
   . . . .
   . . . .
}
```

Dreieck.java

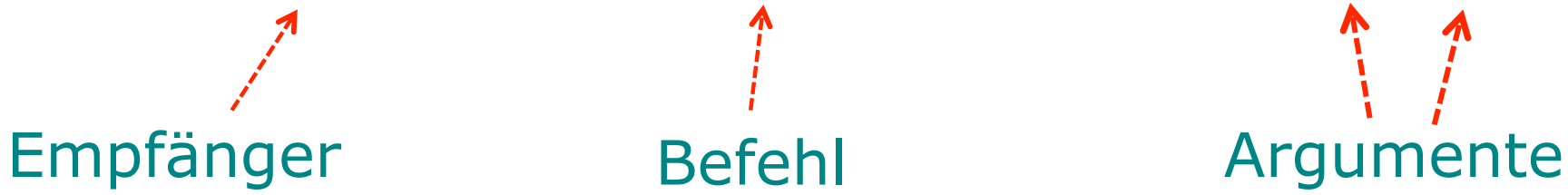
```
class Blatt {
  // Attribute
    . . . .
  // Konstruktoren
    . . . .
  // Methoden
    . . . .
    . . . .
  . . main ( . . . . ) {. .}
    . . . .

}
```

start

Nachrichten

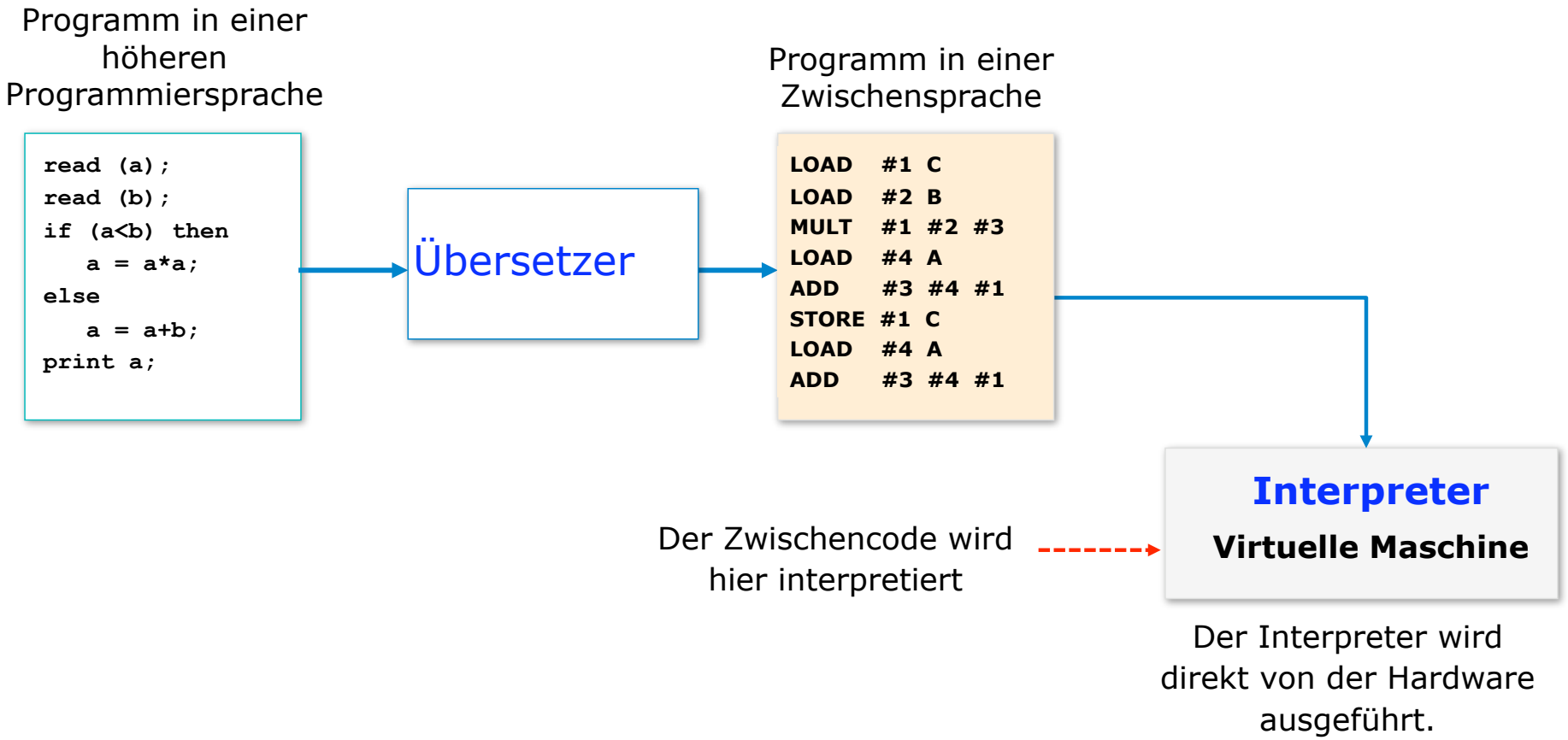
```
personX.calculateSomething( a, b )
```



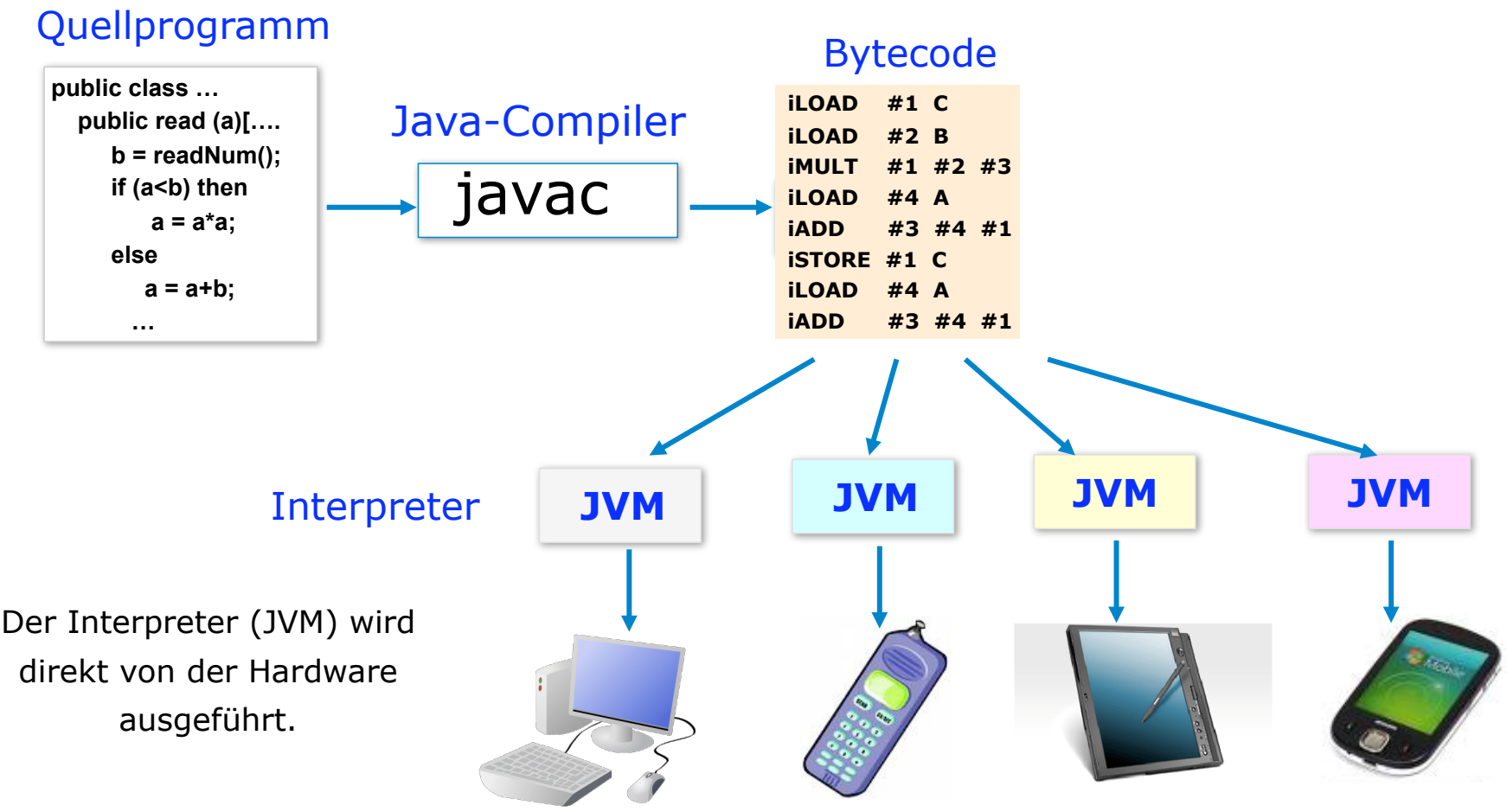
Bevor ich eine Nachricht zu einem Objekt schicke, muss dieses Empfängerobjekt im Speicher existieren.

Programmausführung

Compiler + Interpreter



Java ist plattformunabhängig



Der Interpreter (JVM) wird direkt von der Hardware ausgeführt.

Ich freue mich auf eine gute Zusammenarbeit und auf ein erfolgreiches Lernen!