1. Einführung

Informatik

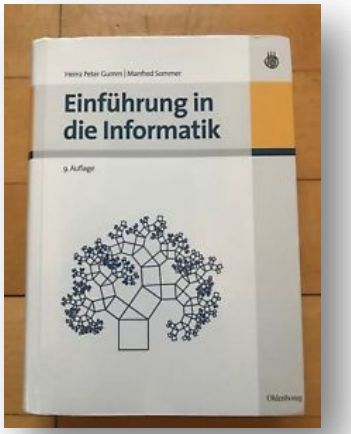
- „ist die **Wissenschaft**, **Technik** und **Anwendung** der maschinellen **Verarbeitung**, **Speicherung** und **Übertragung** von **Information**.“ (Broy, 1992)
- Der Name **Informatik** ist ein Kunstwort aus **Information** und **Mathematik** (geprägt in den 60er Jahren).

Wichtige Aspekte in der Informatik

- Repräsentation von Information
- Informationsverarbeitung
- Verarbeitungsvorschriften
- Informationsverarbeitende Maschinen

Repräsentation von Information

- Viele Abstraktionsebenen



Interpretation

Repräsentation

ISBN 978-3-486-70641-3

```
01001001 01010011 01000010 01001110
00100000 00111001 00110111 00111000
00101101 00110011 00101101 00110100
00111000 00110110 00101101 00110111
00110000 00110110 00110100 00110001
00101101 00110011
```

Informationsverarbeitung

- Reales Problem
  - Gegeben: Tagesumsätze eines Unternehmens
  - Gesucht: Wochen-, Monats- und Jahresumsatz

- Software-Werkzeuge als Hilfsmittel, um schneller, einfacher, besser reale Probleme zu lösen.
  - Programmiersprache
    - Formale Sprache zur „einfachen“ Lösung des Problems
  - Andere Hilfsmittel
    - Editoren, Entwicklungsumgebungen, Software-Bibliotheken

Typische Aufgabe in der Informatik

- Auswahl der geeigneten Hilfsmittel

- Verwendung der Hilfsmittel um reale Probleme zu lösen

Die zwei Gesichter der Informatik

- Informatik als Mittel zum Zweck
  - Lösung konkreter Probleme aus der realen Welt
- Informatik als Methodenwissenschaft
  - Was sind gute Werkzeuge zur Lösung konkreter Probleme?

Informatik zur Problemlösung

- Anwender hat ein konkretes Problem, das von einem Informatiker unter Verwendung eines Computers gelöst werden soll.

- Vorgehensweise des Informatikers
  1. Analysieren des Problems
  2. Erstellung eines Modells der realen Welt
  3. Umsetzung des Modells auf einen Rechner
     - Auswahl der Implementierungswerkzeuge
  4. Testen und Dokumentieren der Lösung

Typische Beispiele realer Probleme

- Erstellung eines Online-Portals für Bank XYZ

- Erstellung eines Energiecockpits für ein mittelständisches Unternehmen.

- Erstellung eines Raumverwaltungssystems für die Uni Marburg

Informatik zur Werkzeugentwicklung

- Gruppe von ähnlichen Problemen aus der realen Welt
  1. Generierung eines abstrakten Problems
  2. Lösung des abstrakten Problems
  3. Bereitstellung der Lösung als Werkzeug
  - → „Framework“

- Wichtige Eigenschaft
  - Lösung konkreter Probleme wird einfacher, wenn die Lösung des abstrakten Problems verwendet wird.

Beispiel

- Entwicklung eines Informationssystems
  - Problemklasse
    - Viele Unternehmen besitzen wichtige Ressourcen, aber es gibt keine einheitliche Verwaltung der Ressourcen
  - Lösung
    - SAP/R3-System zur einfachen Verwaltung der Ressourcen

1.1 Programmiersprachen

- **Die Programmiersprache bildet die Schnittstelle zwischen Mensch und Computer bei der Entwicklung von Lösungen konkreter Probleme und Werkzeugen**
  - Mittels einer Sprache werden die zuvor entwickelten Lösungen auf einem Computer umgesetzt.
- **Analogie zu einem Dolmetscher**

Anforderungen an eine Sprache

- einfach erlernbar
- ausdrucksstark
  - Es können damit ganz viele Probleme gelöst werden.
- effiziente Ausführung der Befehle auf dem Computer
  - Kurze Laufzeit eines Programms
- schnelles Verarbeitung während der Entwicklung

Maschinensprache

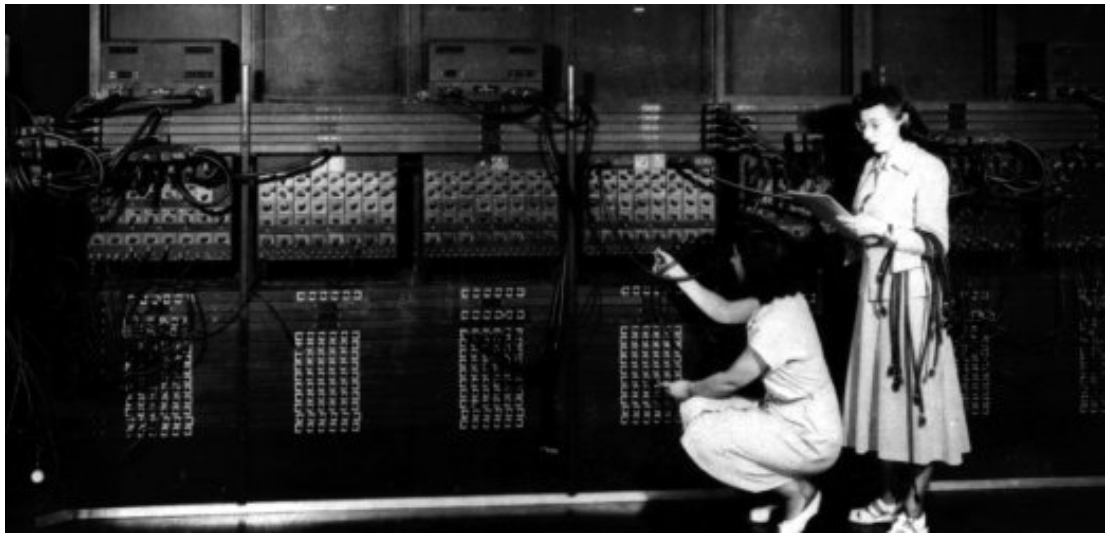
Computer verfügen bereits über

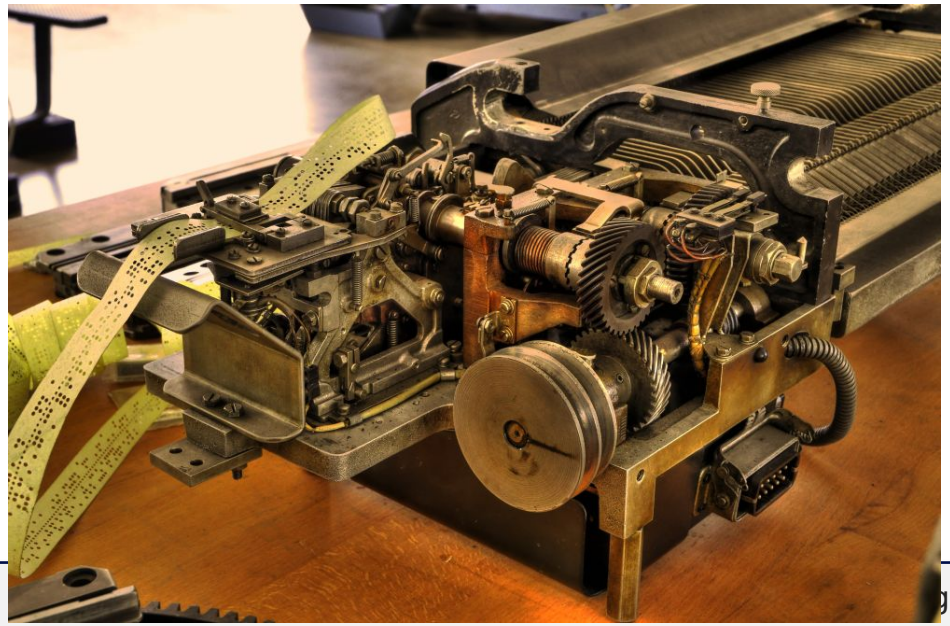
- einen Satz von elementaren Befehlen (Maschinensprache),
  - die direkt vom Computer verarbeitet werden können.
- und einfache Datentypen
  - die zur Repräsentation von Information genutzt werden können.
    - z. B. ganze Zahlen (bis zu einer gewissen Größe).

Maschinenprogramm

- Folge von Maschinenbefehlen zur Lösung eines Problems

Programmieren in der Antike





Beispiel

- Beispiel von Kommandos in Maschinensprache

```
ADD AX, FFh
INC AX
JMP 0A3
MOV AX, 5
```

- Diese Kommandos sind als Zahlen im Binärsystem kodiert (d.h. mit den Ziffern 0 und 1), etwa:

```
10001010 10100111 11010101 10010110 11010110 10101001 11110101 01011110
10110010 10100101
```

- Mit Maschinensprache kann man zwar prinzipiell alle Probleme lösen, die man auch in anderen Sprachen lösen kann, aber für Menschen sind Programme in Maschinensprache schwierig zu verstehen.

➔ **zu hohe Erstellungs- und Wartungskosten**

Höhere Programmiersprachen

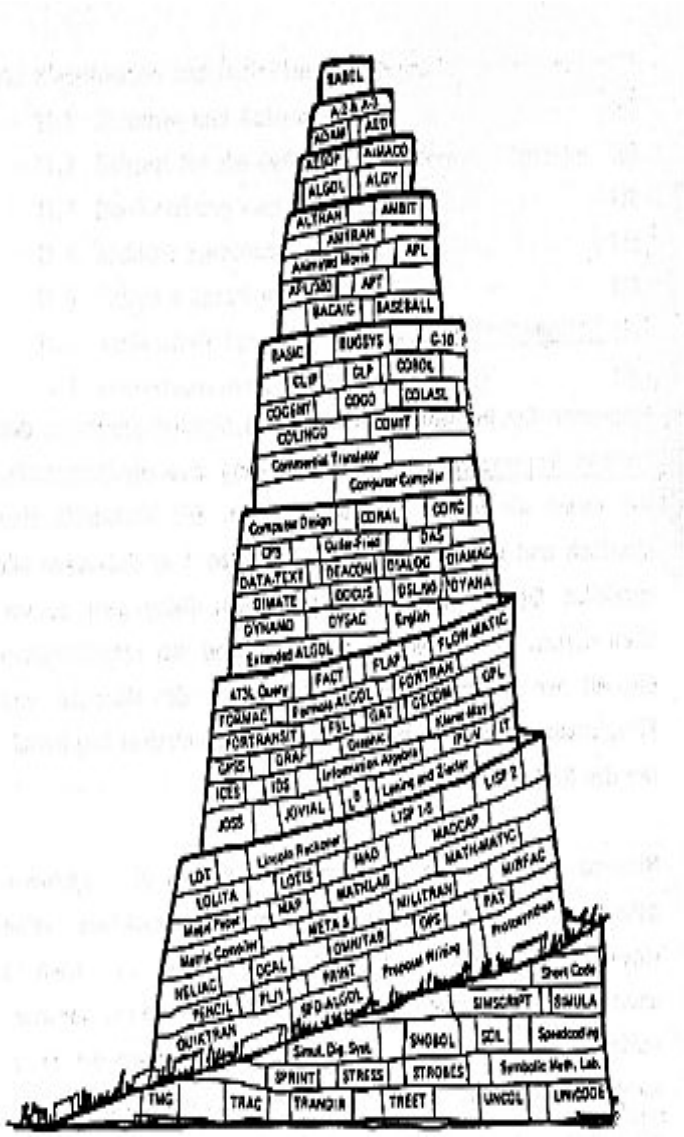
Idee

- Entwicklung von höheren Programmiersprachen, die eine einfache Umsetzung von Problemlösungen erlauben.
- Programmiersprachen bieten mächtige Methoden zur Abstraktion
  a) Möglichkeit der Definition problemorientierter Datentypen (Datenabstraktion)
  b) Mehrere Schritte, die sequentiell ausgeführt werden sollen, können zu einem Schritt zusammengefasst werden (Kontroll- und Funktionsabstraktion).

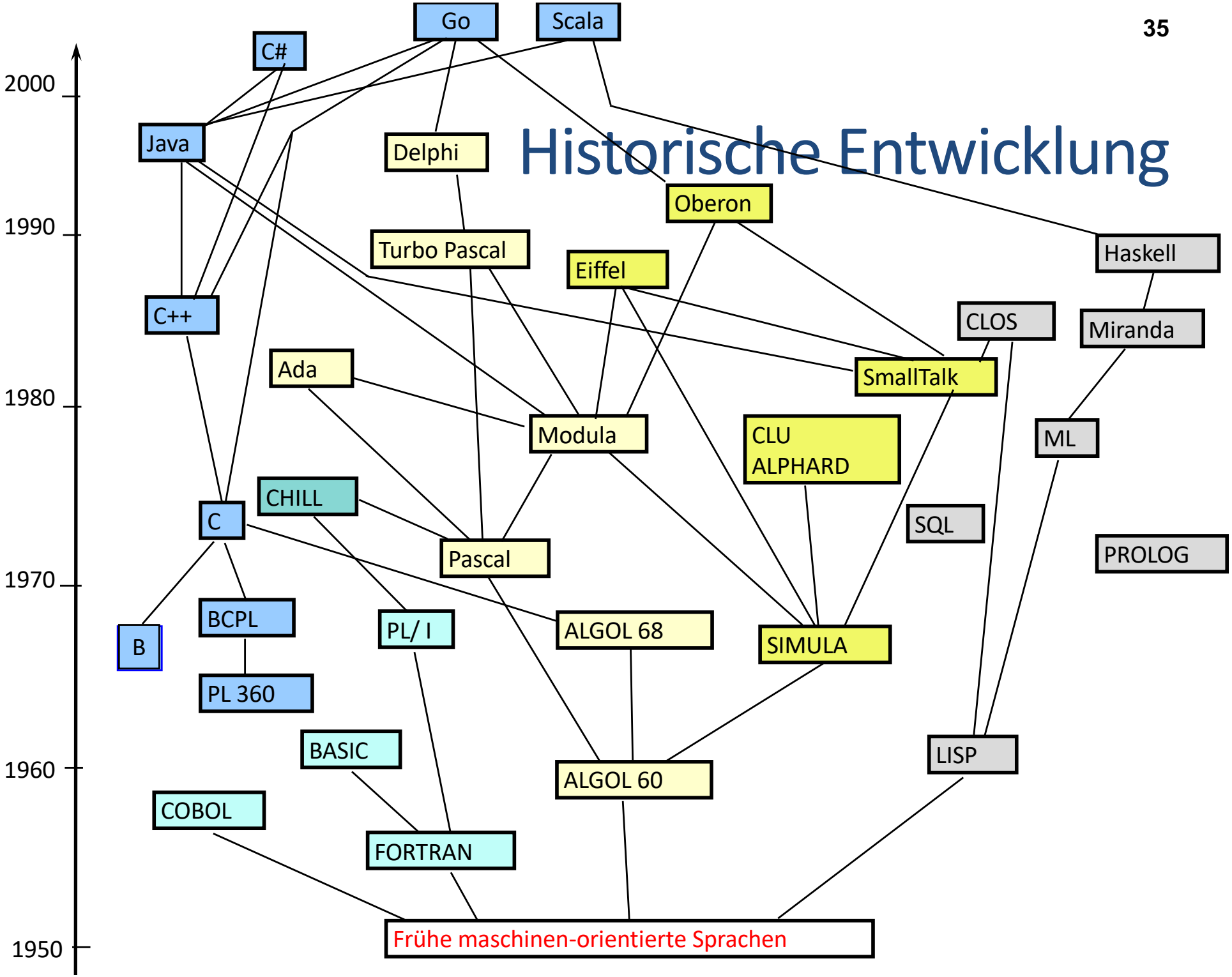
- Unterschiedliche Probleme erfordern verschiedene Programmiersprachen.
  - One-size-does-not-fit-all

Programmiersprachen

- Es gibt Tausende von Programmiersprachen
  - Allgemeine Sprachen, mit denen prinzipiell alle Fähigkeiten eines Rechners zugänglich sind
    - C, Pascal, C++, **Java**, C#, Scala, Go, Rust,…
    - Lisp, Prolog, …
    - FORTRAN, ALGOL, LISP, BASIC, PL1, …
    - Visual Basic, VB.NET
  - Spezialsprachen für bestimmte Anwendungen
    - JavaScript, ECMAScript, VBScript, …
    - PHP, Perl, Python, Ruby …
    - SQL, TeX, TCL/TK, …
    - HTML, XHTML, XML, …



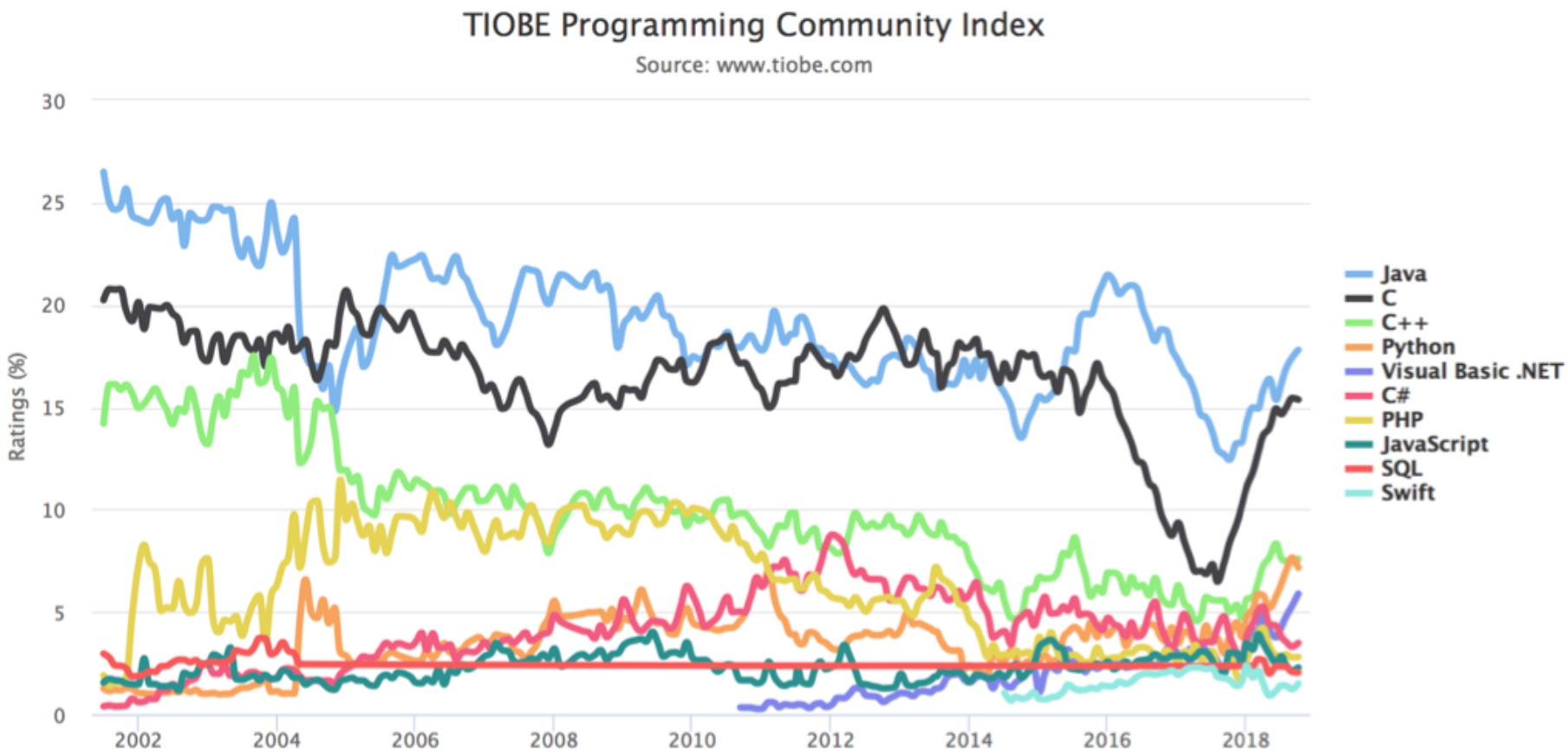
Historische Entwicklung



Klassen von Programmiersprachen

- imperative Sprachen unterstützen das zustandsorientierte Programmieren
  - Werte können in Variablen zwischengespeichert und verändert werden, auf die über einen Namen oder Adresse zugegriffen werden kann.
  - Beispiele: Pascal, C, Fortran, Cobol, …
- objektorientierte Sprachen sind eine Weiterentwicklung imperativer Sprachen
  - Werte können nur innerhalb eines Objekts zwischengespeichert werden. Der Zustand eines Objekts kann durch Methoden verändert werden.
  - Beispiele: Eiffel, Smalltalk, C++, **Java**
- funktionale Sprachen betrachten ein Programm als eine mathematische Funktion, die zu einer Eingabe eine Ausgabe produziert
  - In diesen Sprachen gibt es keine explizite Zwischenspeicherung von Ergebnissen.
  - Beispiele: Lisp, Miranda, ML, Haskell
- objekt-funktionale Sprachen sind ein Hybrid zwischen objektorientierten und funktionalen Sprachen
  - Beispiele: C++14, C#, Scala, **Java9**
- logik-basierte Sprachen
  - Regeln zur Definition von Relationen.
  - Beispiel: Prolog, Datalog, SQL

Ranking von Programmiersprachen (?)



Die Geschichte von Java

- **1991**: James Gosling entwickelt mit einem kleinen Team bei der Kultfirma SUN die Programmiersprache OAK (Object Application Kernel).
  - Ziel zunächst: Programmierung von elektronischen Geräten der Konsumgüterindustrie
- **1995**: Java wird öffentlich vorgestellt und im gleichen Jahr bietet SUN kostenlos den Kern eines Programmiersystems JDK zusammen mit einer Implementierung der JVM an.
- **2010**: Mit der Übernahme von SUN durch die Firma Oracle verliert die Entwicklung von Java an Dynamik.
  - Andere JVM-basierende Programmiersprachen wie Scala werden als Alternative zu Java interessant.
- Im **Sept. 2018** wurde die Version 11 freigegeben und veröffentlicht.

Gründe für den Erfolg von Java

- Java ist zum richtigen Zeitpunkt veröffentlicht worden.
  - C-ähnliche Syntax hat viele dazu bewegt, den Schritt von C und C++ nach Java zu gehen.
  - Gelungene Umsetzung der objektorientierten Konzepte
- Freie Verfügbarkeit von Compiler und Infrastruktur
- Großes Angebot an vorgefertigten Werkzeugen in Java
- Java gilt als die Internet-Programmiersprache
  - Entwicklung einer Vielzahl von Apps und Spielen
    - z. B. Minecraft
- Java ist eine Anforderung bei vielen Jobs

  → aktuelle Jobbörsen