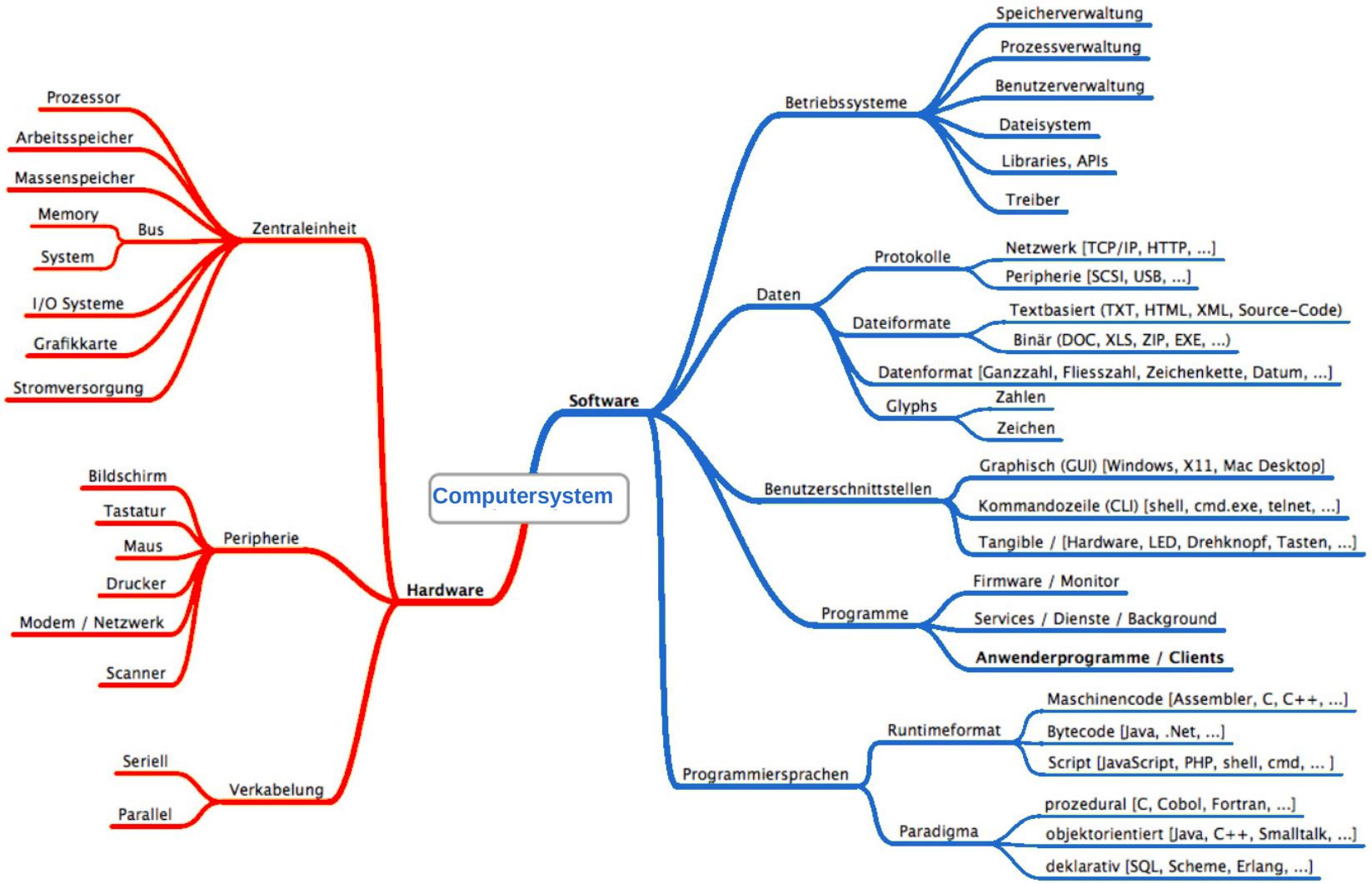
Inf1 Einführung

School of Engineering

- Hardware und Software
- Stellenwert und Geschichte der Informatik
- Daten
- Geschichte der Programmierung
- Was ist eine Programmiersprache
- Grundlegende Paradigmen
- C Programm

Hardware und Software

Übersicht Computersystem



Hardware / Software

- ***Hardware*** *bezeichnet den* ***physischen Träger****, auf dem die Daten und die Software existieren (passiv) und aufbearbeitet werden (aktiv)*
- ***Software*** *bezeichnet alle* ***nicht physischen Funktionsbestandteile*** *eines Computers bzw. eines jeden technischen Gegenstandes.*
  - *Daten (passiv): Bestandteile in denen Information gespeichert ist.*
  - *Programme (aktiv): Anweisungen/Befehlssequenzen für die Hardware die*
    - Daten verarbeitet, i.e. Daten in andere Daten umwandeln
    - Daten so aufbearbeitet, dass sie dem menschlichen Erkenntnisapparat (Sinne) zugänglich werden.

Datenträger: HW

Musik: SW

Informatik(er) und Gesellschaft

Bild des Informatikers in der Gesellschaft

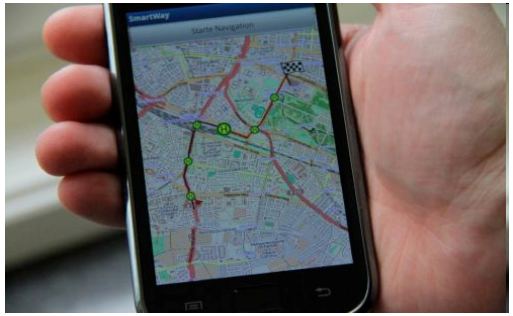
- Welche Eigenschaften hat ein Informatiker?

- Männlich
- Ein intelligenter Eigenbrötler
- Verbringt viel Zeit allein mit dem Computer
- Lebt primär in abgedunkelten Räumen
- Freizeit verbringt er mit Seinesgleichen
- Redet eine spezielle Sprache
- Kommuniziert via E-Mail, Reddit, Slack
- Legt wenig Wert auf sein Äusseres
- Vernachlässigt zuweilen die Hygiene

Er ist ein Nerd

Stellenwert der Informatik

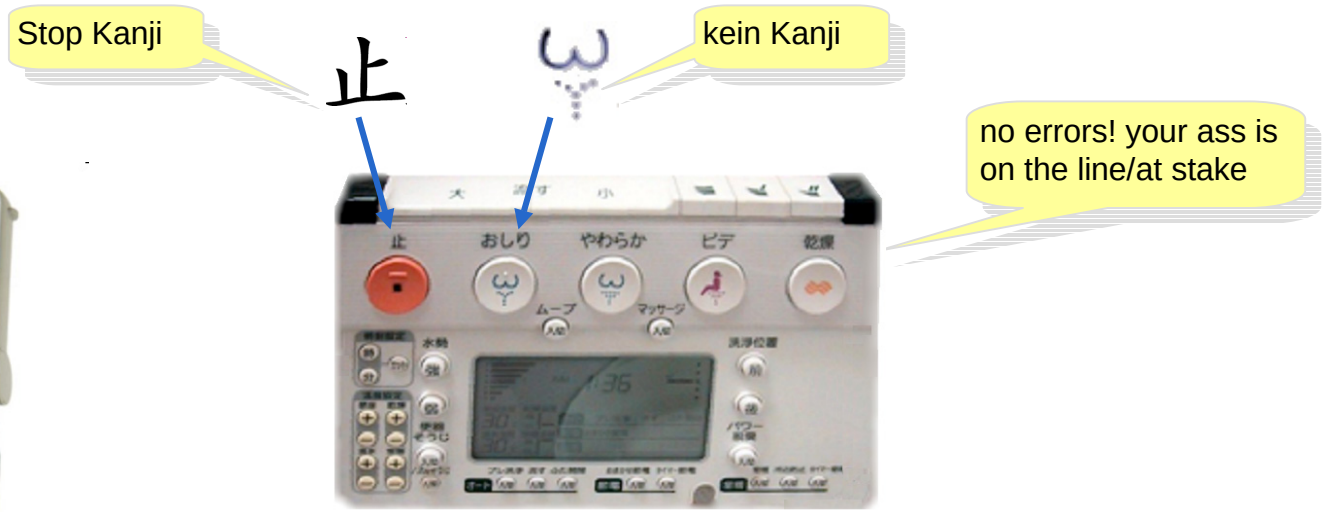
- Modernes Leben ist mit Informatikprodukten durchsetzt
- Freizeit
  - Internet
  - Smartphones
  - Tablets und e-Books als Papierersatz
  - Navigationsgeräte
  - ….
- Arbeitsplatz
  - Im Dienstleistungsbereich der PC oftmals als primäres Arbeitsmittel
  - Buchhaltungssysteme
  - Diagnosesysteme in der Medizin
  - Bordcomputer in Verkehrsmitteln
  - Reservationssysteme am Flughafen
  - ….



Stellenwert - z.B. Embedded Systems

- In vielen elektronischen Geräten ist heute ein Microprozessor eingebaut

Oszilloskop



human waste disposal system control panel in Japan

Ohne Informatik würde die moderne Gesellschaft zusammenbrechen

Wieso ist Informatik schwierig?

- 1. Ingenieur
  - **Dev**evelopment: Performante, funktionelle Implementierung
  - **Op**eration: Stabiler und Unterbruchsfreier Betrieb
  - **Sec**urity: Sicherheit des Systems

  → **DevOp oder DevSecOp**

Ingenieure sind i.d.R. primär an der Lösung von technischen Problemen interessiert

"hartes" Engineering

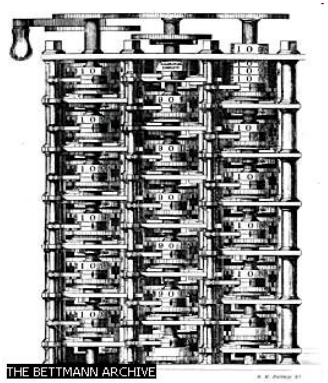
- 2. Arbeitswissenschaftler
  - Gestaltung von ergonomischen Benutzerschnittstellen
- 3. Soziologen/Psychologen
  - Soziologische und psychologische Aspekte
- 4. Künstler/Designer
  - Ansprechende Gestaltung der Oberfläche
- 5. Anwender der jeweiligen Fachdomäne
  - Verständnis der eigentlichen Aufgabe und der Abläufe

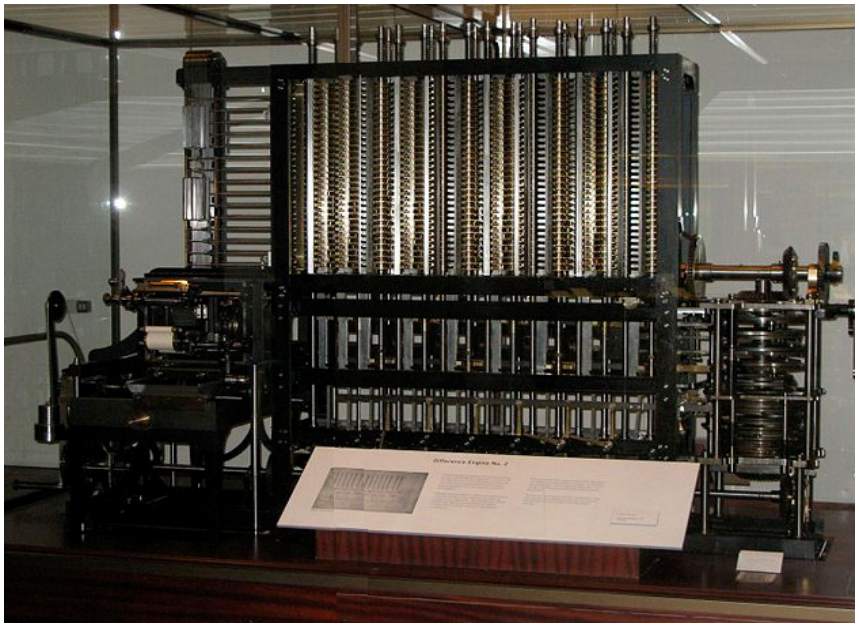
  → **User Experience Expert** (UX)

"weiche" Faktoren

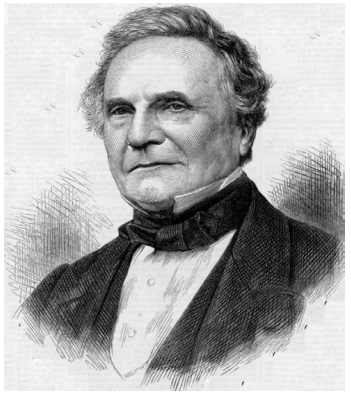
Geschichte der Informatik

1822: Charles Babbage - Mechanisch



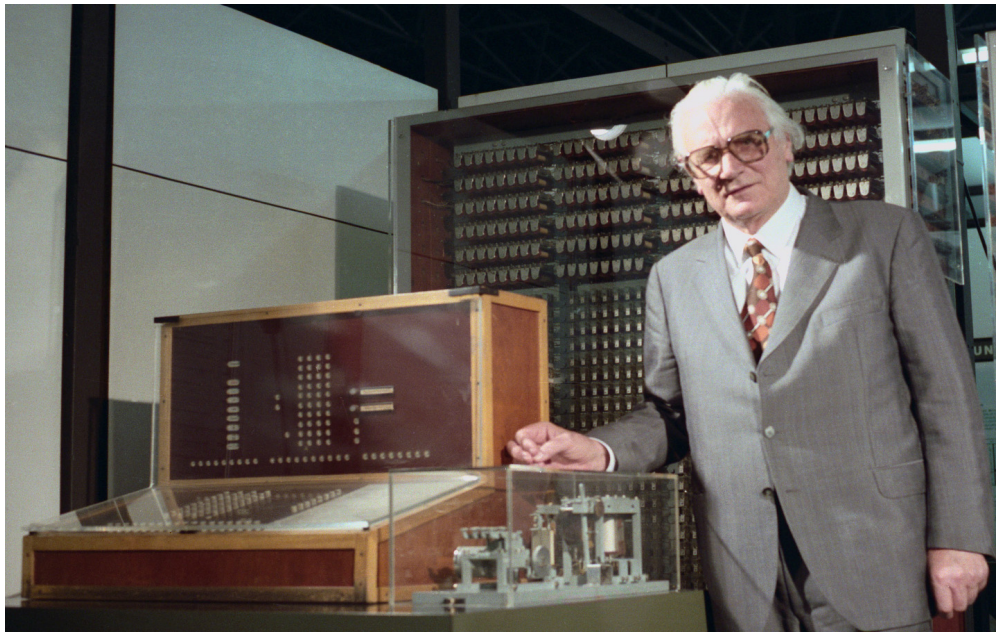


- Mechanische Rechenmaschine (Grundrechenoperationen)
- 25'000 Einzelteile
- Preis: £17470
  - Vergleich: Dampflock: £784
- Nachfolgeprojekt (Analytical Engine) wegen Geldmangel gestoppt

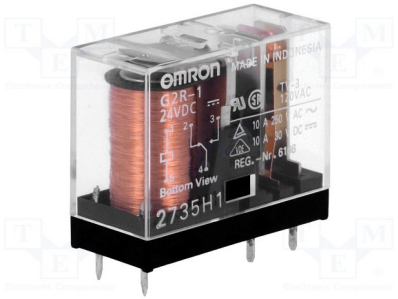


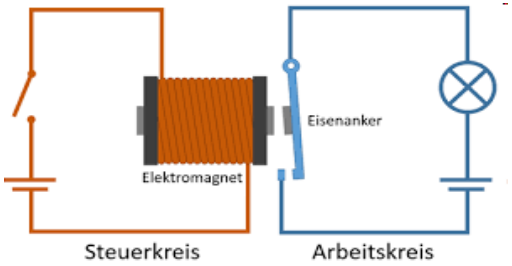
Erste Programmierer
Ada Lovelace

1938: Zuse Z1 - Z4 Relais

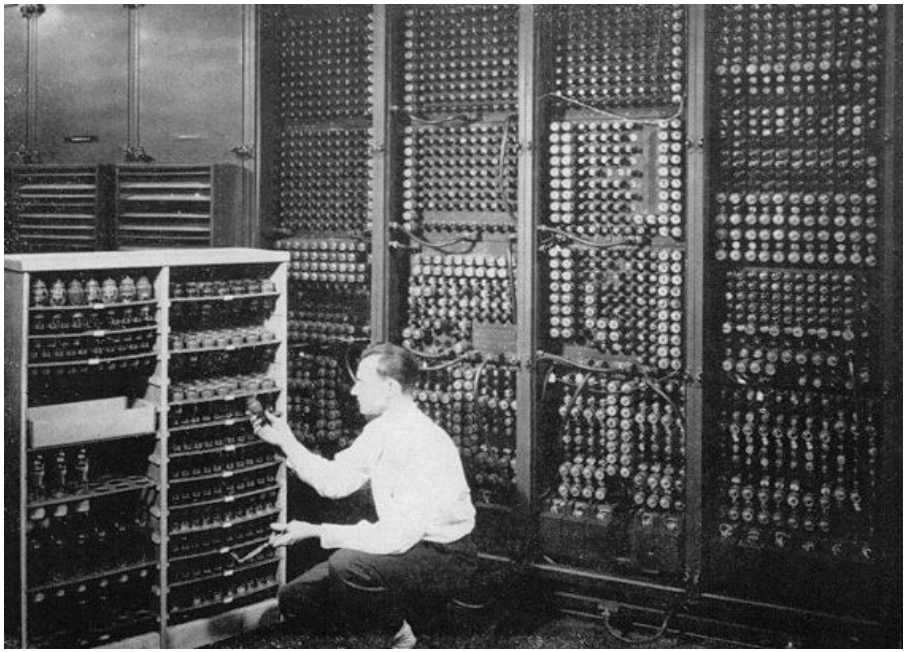


- Z1:Mit Laubsäge ausgesägte Bleche
- Z3: Relais
- Taktfrequenz: 1Hz
- Speicher 64 Zellen à 22 Bit
- Gewicht: 500 kg
- In Wohnung der Eltern

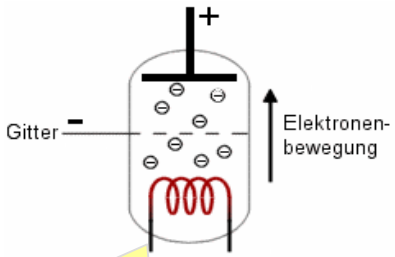




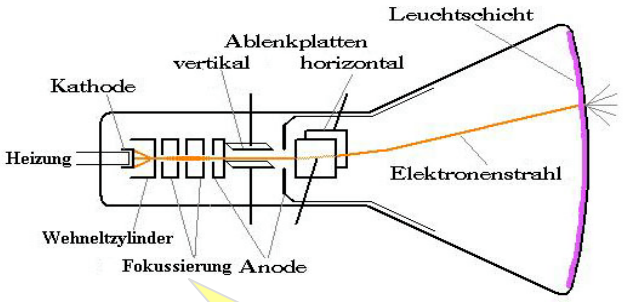
1946: ENIAC - voll Elektronisch



- Technik: 19'000 Röhren
  - Lebensdauer ca. 10'000 h
  - 50 pro Tag mussten ersetzt werden
  - Rechner war nach 30 Minuten defekt
- 175 KWh
- Gewicht 35 Tonnen
- Taktfrequenz: 100 KHz
- Dezimalarithmetik
- Programmierung mittels Steckbrett

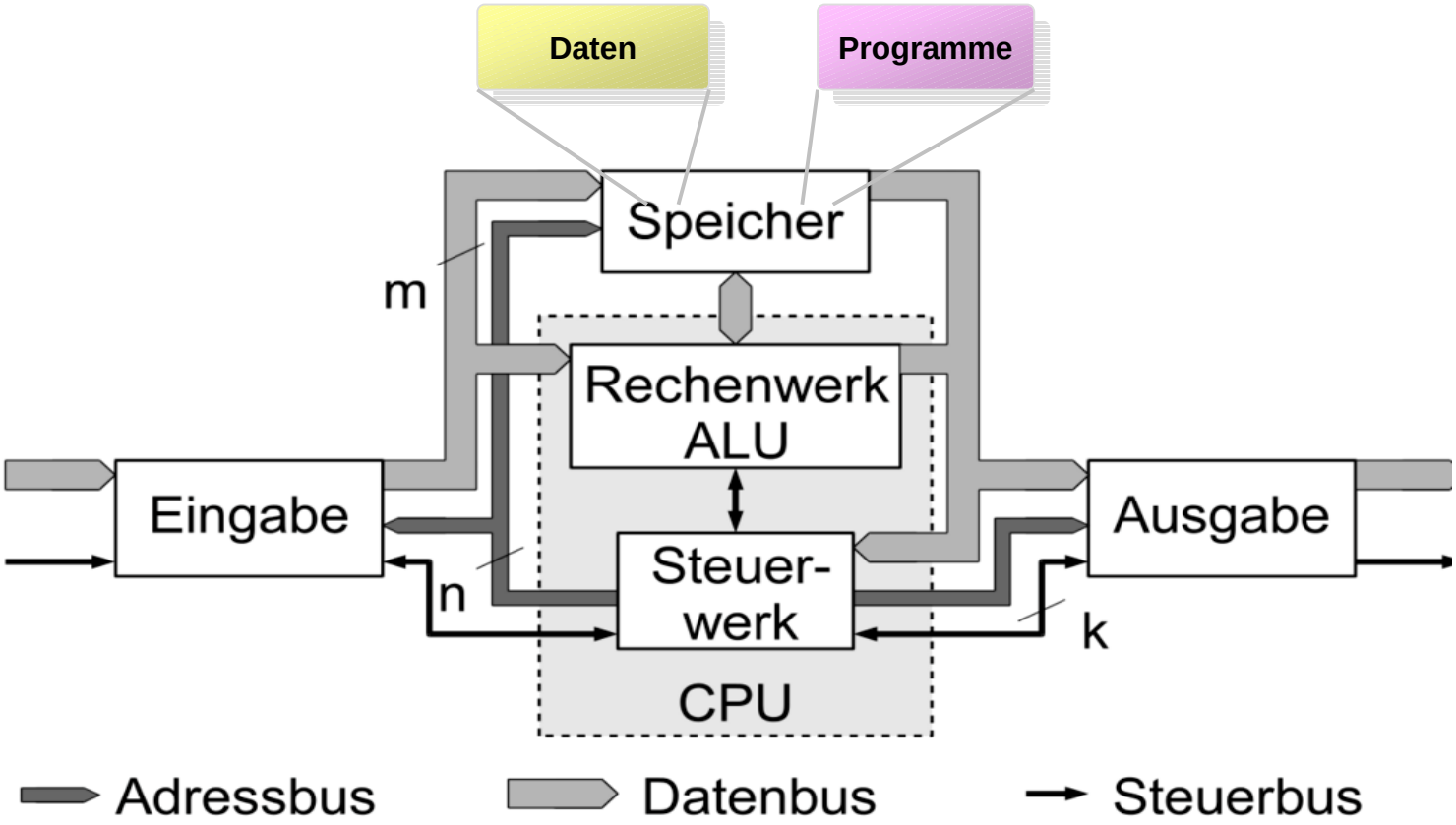


Bei hohen Temperatur können Elektronen die Glühwendel verlassen



Kathodenstrahlröhre

Von Neumann Referenzmodell 1945



von Neumann-Architektur:
Daten und Programme im gleichen Arbeitsspeicher
Program: Daten, die als Instruktionen (Anweisungen für den Rechner) interpretiert werden

Ab 1950 ...

Computer im heutigen Sinn:

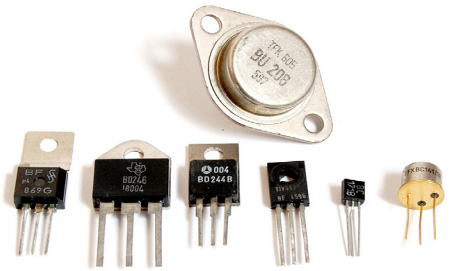
- Programm im Arbeitsspeicher
- Binärsystem für Daten und Arithmetik
- Elektronisch

Generationen

1	Elektronenröhren als Schaltelemente. Arbeitsspeicher wenige hundert “Wörter”.
2	Transistorschaltkreise. Ferritkern-Hauptspeicher; Periphere Speicher: Magnetisierbare Medien.
3	Integrierte Schaltkreise. Beginn der Miniaturisierung
4	Hochintegrierte Schaltkreise. Prozessor auf einem Chip. 8-Bit
5	Mikro-Miniaturisierung, Verdichtung. 16-, 32-, 64-Bit Architekturen.
6	Vernetzte Rechner

Hat also rund 70 Jahren Entwicklung hinter sich

1964 - 1970: Grossrechner: IBM System360



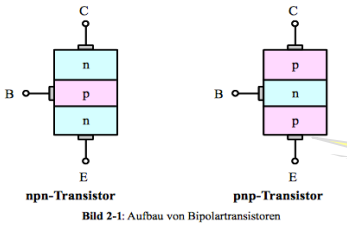


Bild 2-1: Aufbau von Bipolartransistoren

- IBM360-67
- Halbleiter, integrierte Schaltkreise
- 360/67 mit Virtual Machine Computing
- 20 KByte bis 768 KByte Hauptspeicher
- Takt ~ 1MHz
- Binärarithmetik
- Programmierung mittels Lochkarten oder Terminal

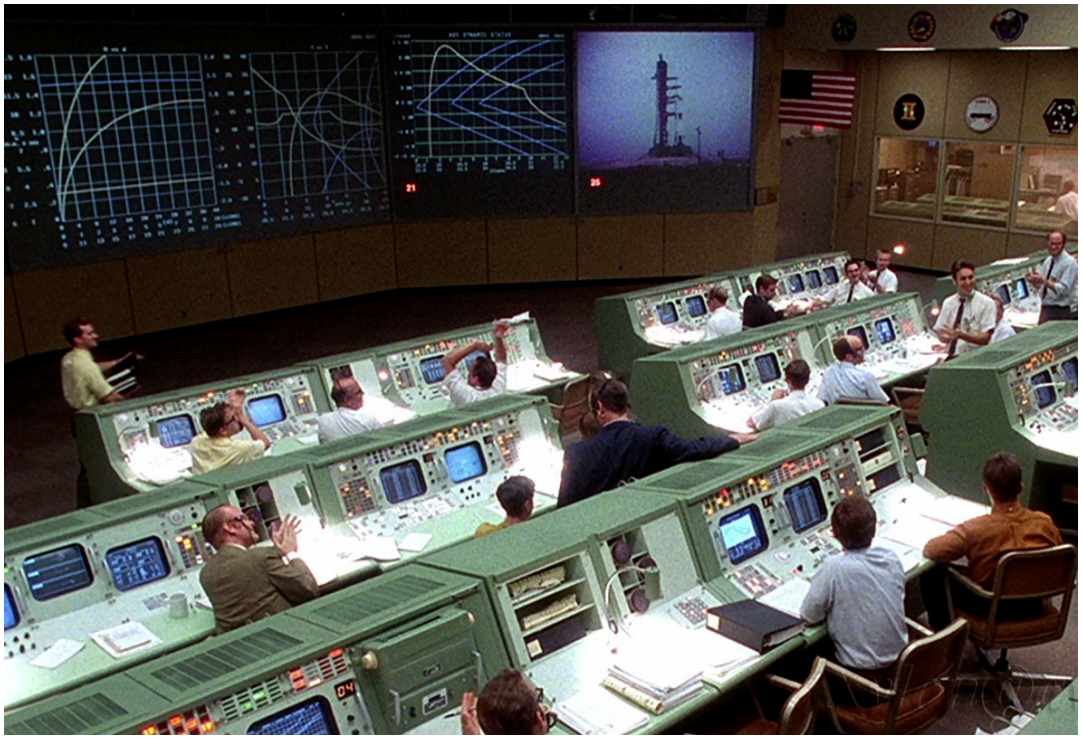
Bipolartransistor mit drei unterschiedlich dottierten Schichten

Apollo 11 1969 - NASA Mission Control

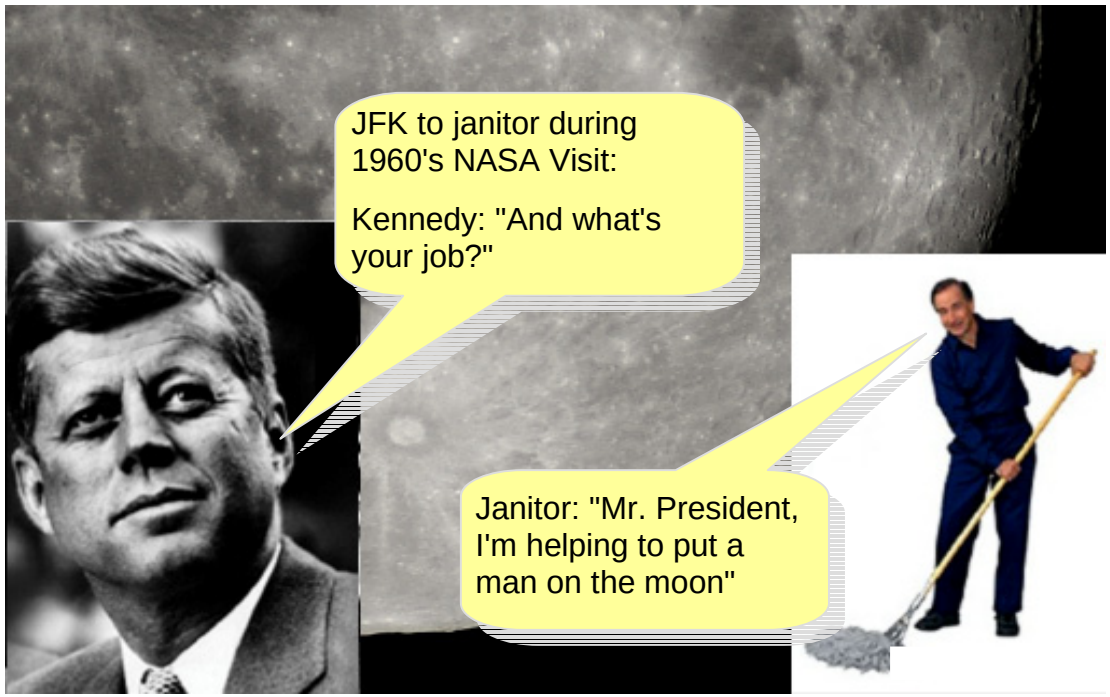
- 5 IBM System/360 Model 75
- 1 MIPS (x 5)
- 1 MB Memory (x 5)
- Leistungsaufnahme: Megawatt Bereich
- Programm: 6 Million Codezeilen

- 3'000 IBM Angestellte für Programmierung und Betrieb

- Miete $50,000 to $80,000 p.m.
- Kauf $2.2 Million to $3.5 Million
- 25,000h MTBF (~3 Jahre)



Der NASA Arbeitsethos



all in all 400'000 engineers and technicians engaged in Apollo project

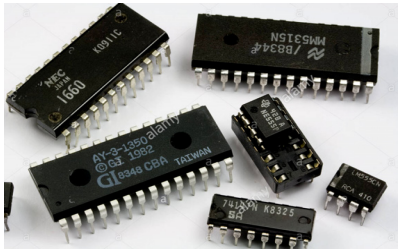
http://enodoglobal.com/2017/12/28/the-key-to-transforming-company-culture-unlocking-the-power-of-identity/

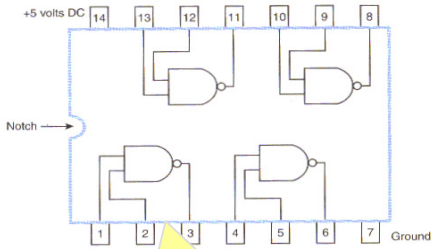
- A staggering 87% of employees worldwide are not engaged. Many companies are experiencing a crisis of engagement and aren't aware of it.

  https://www.gallup.com/services/190118/engaged-workplace.aspx

1970 - 1976: Abteilungsrechner: PDP-11

- Rechner für ca. 20 Pers.
- Integrierte Schaltkreise
- 16 Bit Maschine
- 64 KBytes bis 256 KBytes
- Betriebssystem: Unix
- Eingabe mittels Terminal
- $100'000

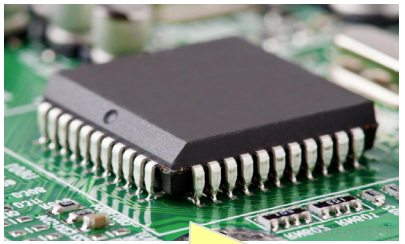




Integrierte Schaltung (IC): Mehrere hundert Transistoren in einem Gehäuse

PC Zeitalter: ab ca. 1980

- Personal Computer
- 8 Bit Microprozessor
  - Intel: 8088 (8 Bit Variante von 8086)
- 4.77 MHz Takt
- >128 KBytes Speicher
- 5" Floppy mit 360 KB
- MS DOS
- Preis $1000-$5000

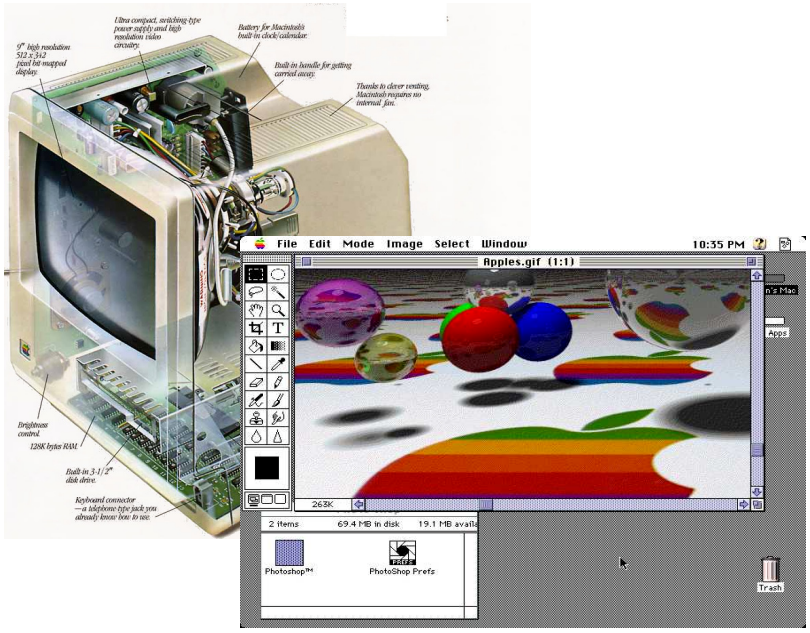


Hochintegrierte Schaltung: Tausende bis (heute) mehrere Milliarden Transistoren in einem Gehäuse

Apple Lisa (1983), Macintosh (1984)

Apple Lisa

- war ein Fehlschlag: zu teuer, zu langsam
- Apple ging dabei fast bankrott



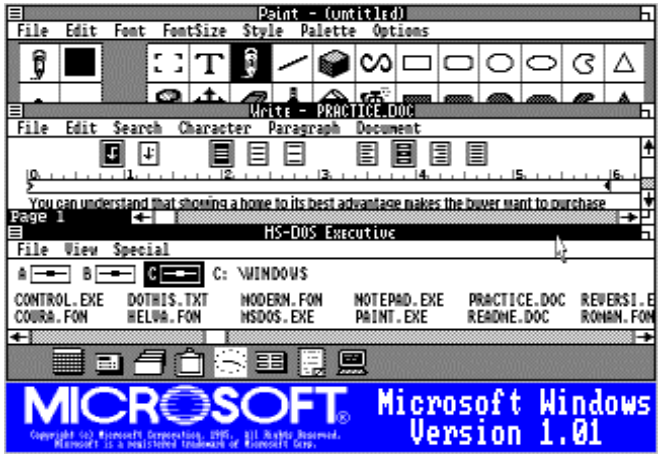
Macintosh

- Kommerzieller Erfolg kam überraschend
- relativ preisgünstig: $2000

Microsoft Windows 1(1985), 2 (87), 3.11(93)

Windows 1 und 2

- Auf DOS aufsetzend
- juristische Auseinandersetzung mit Apple
- kein kommerzieller Erfolg



Windows 3.11

- Service Release für fehlerbehaftetes 3.1
- Erste reine Windows Anwendungen entstehen
- Kommerzieller Erfolg

World Wide Web 1994

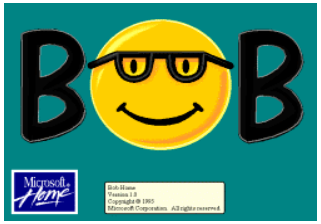
- Ein Problem am CERN war, dass sich ein Teil der Laboratorien auf F, ein anderer auf CH Gebiet befindet.
- Unterschiedliche Netzwerk-Infrastrukturen, die den Austausch von Informationen erschwerten.
- Erfindet HTML und B.L. gründete das World Wide Web Consortium
- WWW nach Grundprinzipien wie Gesellschaft nach B.L. funktionieren sollte:
  - flache Hierarchien
  - harmonische Kooperation
  - Toleranz und Offenheit für Vielfalt
  - Vernunftgebrauch
  - Zuversicht in die Mitstreiter

Timothy John Berners-Lee

Erster Web Server am CERN

Microsoft Bob Frühjahr 1995

- Neue Oberfläche von Microsoft
- für Computern-Anfänger
- Es wurde strikt an Metaphern festgehalten
- Bedienung durch Einschränkungen der realen Welt bestimmt.



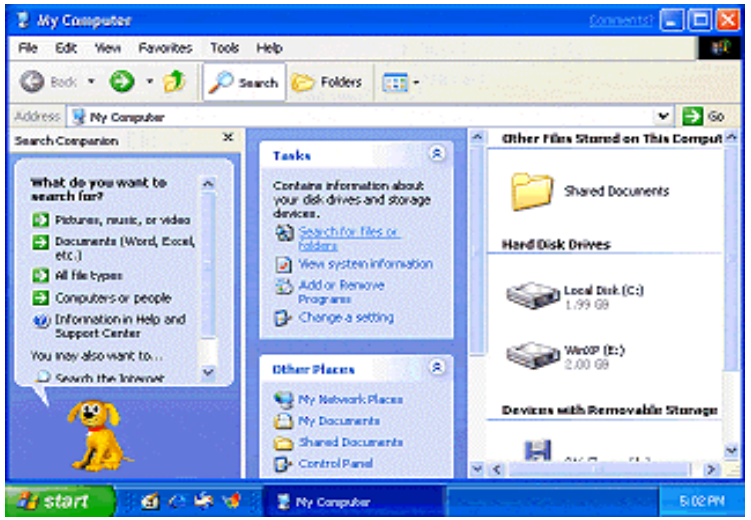
https://www.youtube.com/watch?v=RkU4WWEUj-Y

- Kommerzieller Misserfolg
- Programm Manager war Bill's Frau Melinda
- Teile davon später wieder auferstanden: "nervender" gelber Hund

Windows 95 , XP , Win 7, Win 10

Windows 95 (Herbst 1995)

- Bezüglich Usability mit Apple - endlich - gleichauf
- Aber immer noch Aufsatz auf DOS
- kommerzieller Erfolg



NT 4.0 (1996), XP, Win 7, Win 10 ...

- kein Aufsatz auf DOS mehr
- Assistenten:
  - Hund, Büroklammer,Cortana

Apples Desktops (ab 2000)

- Basierend auf (stabilem) UNIX Mach-Kernel
- Verwendung von Multimedia und sieht "cool" aus

Apples Aqua Design

Apples OS X El Capitan

Portable Computer

- 1981 Osborne
  - Prozessor:Zilog Z80 4.0 MHz
  - Arbeitsspeicher: 128 kB RAM
  - 2 * 160 KB Floppy
  - Bildschirm: 5"
  - Betriebssystem: CP/M 2.2
  - Stromnetz
  - 11 Kg

Rege's Laptop

- 2020 Dell XPS 15
  - 9th Generation Intel Core i9-9980HK
    - *8 Cores - 5 GHz*
  - Bildschirm: 15" Full HD (1920 x 1080)
  - 32GB LPDDR3 RAM, 1T SSD
  - Windows 10
  - Up to 7-hours Battery Life
  - 2 Kg

Smartphones ab 2000

- Frühere Geräte von z.B. Nokia
  - einfach "miniaturisierte PC"

- Erstes "echtes" Smartphone vorgestellt von S. Jobs 9. Januar, 2007

Smartphone technische Daten

- Dual Core CPU (TI OMAP 4430 ~1 GHz)
- 500 MIPS -> **x 10..100**
- 1 GB RAM/ 16 GB SSD **x 100..200**
- 1.6 Watt **x 10^{-7}**
- 12 Millionen Codezeilen **x 2**
- Lautsprecher/Mikrophon
- Kamera
- Display
- Verbindungen
  - WLAN
  - GSM, GPS

Vergleich NASA Rechner

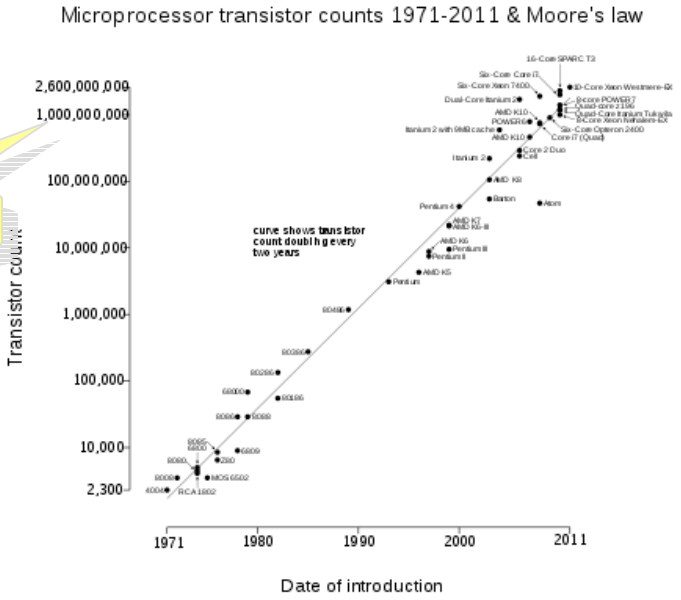
vs

Jeder Smartphone Besitzer hat mehr Rechenleistung zur Verfügung als die gesamte NASA mit mehreren 10'000 Mitarbeiten für die Mondlandung

Moor'sches Gesetz

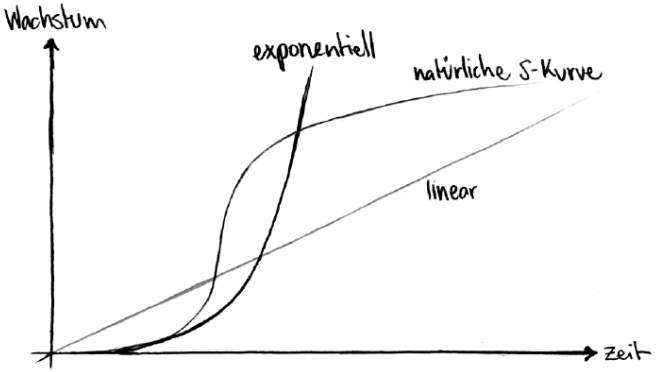
- Anzahl Transistoren verdoppeln sich alle zwei Jahre -> Exponentielles Wachstum
- Formuliert im Jahr 1965

logarithmische Skala



Kenneth Boulding: nur Idioten und Ökonomen glauben an unendliches, exponentielles Wachstum

- Anhaltendes exponentielles, also unendliches und immer schneller steigendes Wachstum ist in einer endlichen Welt nicht möglich.



Daten
Digitalformat

Daten (Analog vs Digital)

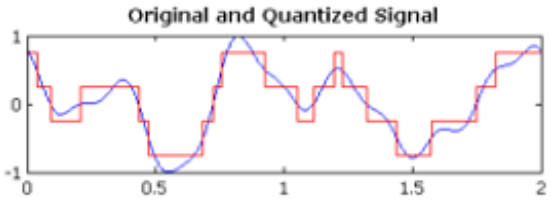
Analog — Beispiel: Mess-, Fühlgrösse in kontinuierlichem Bereich. Geschwindigkeit, Temperatur, Musik, Farbe, Zeit

Digital — Wenn für die Erfassung eines Zustandes, eines Wertes, u.dgl. in den Wertebereichen oder auf den Skalen eine endliche Anzahl unterscheidbare „Werte“ verfügbar sind.

„Digital“

„Genauigkeit“ ist im Digitalcomputer immer beschränkt
i.e. analog ist eigentlich genauer

Durch die Digitalisierung wird ein sog.
Quantisierungsfehler eingeführt.



Wie werden Daten "physikalisch" gespeichert?

- Speicher eines (Digital-) Computers können nur zwei Zustände annehmen (je nach Anwendung und Technologie)

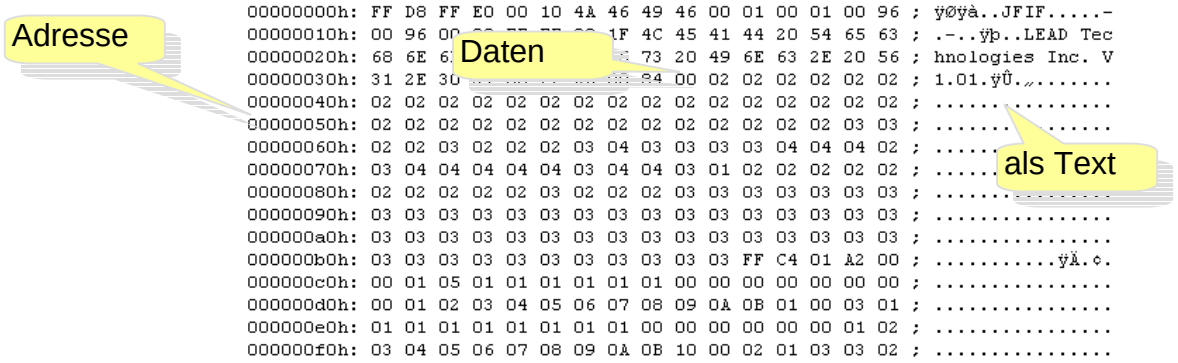
- Grund: bei vielen elektrischen Vorgänge sind klar 2 Zustände unterscheidbar.
  - Spannung oder keine Spannung (positiv/negativ)
  - Ladung oder keine Ladung (bzw. positive/negative Ladung)
  - Magnetisierung N oder S

- Dies wird entsprechend in eine binäre Logik übersetzt:
  - 1 oder 0
  - Wahr oder Falsch

- Jede Speicherzelle kann ein Bit (Zustand 1 oder 0) speichern

Bit und Byte

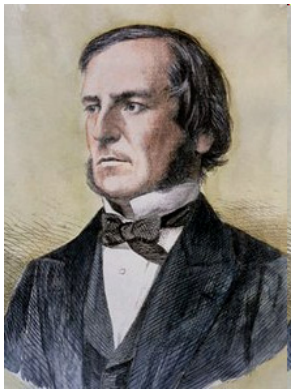
- Acht Bit werden zu einem *Byte* zusammengefasst,
  - 256 mögliche Werte (2^8)
- Die Bytes können mit Adressen im Hauptspeicher wahlfrei (über Adresse) angesprochen werden
  - Sekundärspeicher wird i.d.R. sequentiell zugegriffen
- Mehrere Bytes werden dann zu 16, 32 oder 64 Bit zu Einheiten (**Wörtern**) zusammengefasst
  - Bei grossen Speicher: Wörter als kleinste **adressierbare Einheit** nehmen (z.B. 64 Bit = 8 Bytes)



Rechnen mit Bits: Boole-sche Algebra

- Algebra mit Bits (0 = false, 1 = true)

Not A	A	B	A and B	A or b	A xor B
1	0	0	0	0	0
1	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	1	0



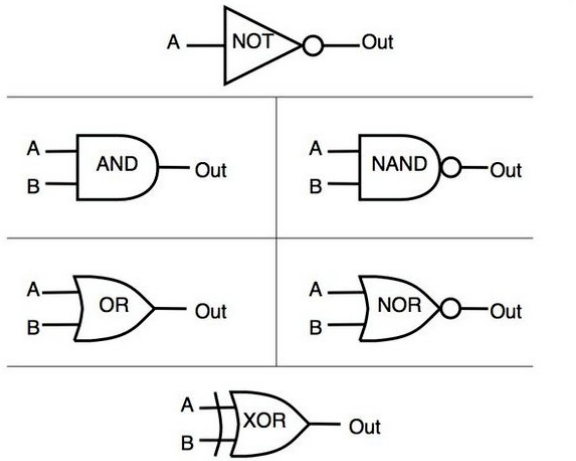
George Boole
1815-64

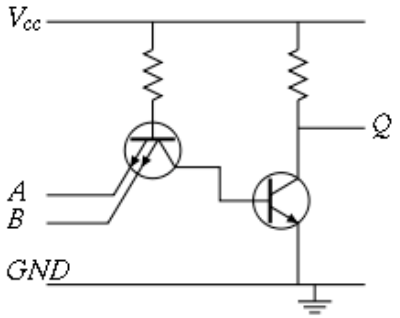
- Die Boole-sche Algebra ist die Grundlage der mathematischen Logik
- Weil alles im Computer letztendlich binär codiert ist, funktionieren alle Operationen mit irgendwelchen Daten dank dieser Grundlage.
- Zum Glück ist die Arithmetik für diese Algebra sehr elegant und auch einfach mit elektronischen Bauelementen realisierbar.

Bool-sche Operationen mit Binärzahlen

- Bitfolgen (oder Bytes) können logisch verknüpft werden
- Bsp: AND ($\wedge$) Verknüpfung von zwei Bitfolgen

```
  0 0 0 1 1 0 1
∧ 0 1 0 1 1 0 0
-------------------
  0 0 0 1 1 0 0
```

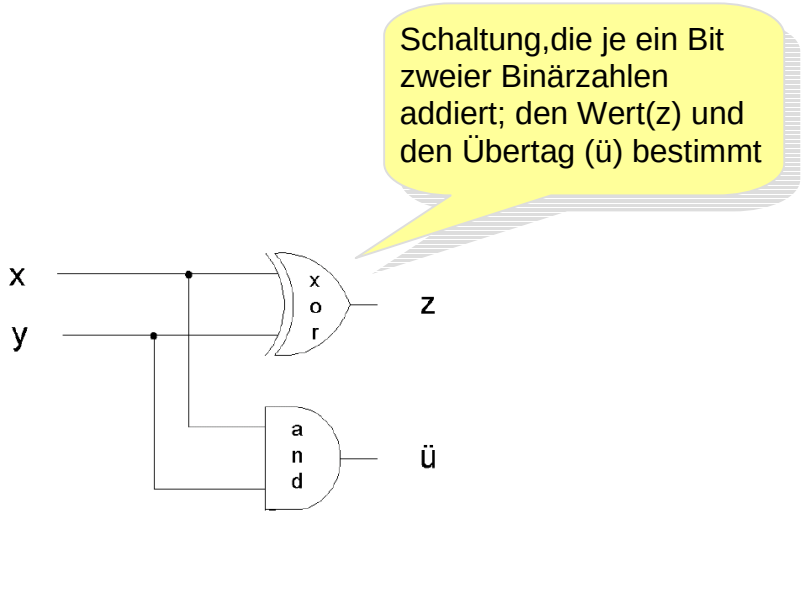




Rechnen mit Binärzahlen

- Im Binärenzahlensystem kann auch (wie gewohnt) gerechnet werden
- Z.B. Addition

```
  0 1 0 1 1 0 1
  0 0 0 0 1 0 1
        1 1     1   ü
-----------------
  0 1 1 0 0 1 0
              z
```



Zahlensysteme

Wieso benutzt der Mensch das Dezimalsystem?

School of Engineering

Dezimalsystem (Zehnersystem)	**Binärsystem (Zweiersystem)**
Alphabet: {0,1,2,3,4,5,6,7,8,9}	Alphabet: {0,1}
Basis: 10	Basis: 2
Symbole: Ziffern	Symbole: Binärziffern = Bits (Binary Digits)
1597_{10} = 1000 + 500 + 90 + 7	1101_2 = $1\bullet 2^3 + 1\bullet 2^2 + 0\bullet 2^1 + 1\bullet 2^0$
= $1\bullet 10^3 + 5\bullet 10^2 + 9\bullet 10^1 + 7\bullet 10^0$	= $8 + 4 + 0 + 1 = 13_{10}$
Anz. Stellen: Anz. mögliche Werte: 1 10^1 2 $100 = 10^2$ 3 $1000 = 10^3$ = 1k (Kilo) ... 6 1'000'000= 10^6 = 1M (Mega) ... 9 10^9 = 1G (Giga) ... 12 10^{12} = 1T (Tera)	Anz. Stellen Anz. mögliche Werte 1 $2 = 2^1$ 2 $4 = 2^2$ 3 $8 = 2^3$ 4 $16 = 2^4$ ... 8 256 (8 Bit entspr. 1 Byte) ... 10 1024 (= 1 KiBit) ... 16 65536 ... 20 1'048'576 (= 1MiBit)

Amendment 2 of IEC International Standard IEC 60027-2 vom November 2000 definiert, welche Präfixe verendet werden sollen, um den Basistyp zu potenzieren

Präfixe mit Basis 10

K	Kilo	10^3	1'000
M	Mega	10^6	1'000'000
G	Giga	10^9	1'000'000'000
T	Tera	10^{12}	1'000'000'000'000
P	Peta	10^{15}	1'000'000'000'000'000
E	Exa	10^{18}	1'000'000'000'000'000'000

Präfixe mit Basis 2 (bi = binary)

Ki	Kibi	2^{10}	1'024
Mi	Mebi	2^{20}	1'048'576
Gi	Gibi	2^{30}	1'073'741'824
Ti	Tebi	2^{40}	1'099'511'627'776
Pi	Pebi	2^{50}	1'125'899'906'842'624
Ei	Exbi	2^{60}	1'152'921'504'606'846'976

Umwandlung zwischen Zahlensystemen

- Binär nach Dezimal
  - Schema: Multiplikation der Stelle mit der Wertigkeit und anschliessende Addition

$$1011_2 = 1*2^3+0*2^2+1*2^1+1*2^0=8_{10}+0_{10}+2_{10}+1_{10}=11_{10}$$

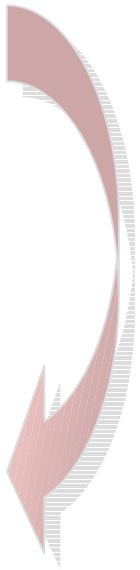
- Dezimal nach Binär
  - Schema: Man dividiert die Dezimalzahl fortlaufend durch 2 (ganzzahlige Division)
  - bis das Resultat 0 ergibt. Merke den Rest.
  - Schreibe die Restwerte von rechts nach links nebeneinander

Umwandlungen Dezimal nach Binär

33_{10}

33 : 2 = 16 Rest: 1
16 : 2 = 8 Rest: 0
8 : 2 = 4 Rest: 0
4 : 2 = 2 Rest: 0
2 : 2 = 1 Rest: 0
1 : 2 = 0 Rest: 1

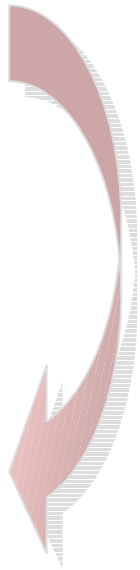
100001_2



11_{10}

11 : 2 = 5 Rest: 1
5 : 2 = 2 Rest: 1
2 : 2 = 1 Rest: 0
1 : 2 = 0 Rest: 1

1011_2



Gegenprobe 33

$1*2^5+0*2^4+0*2^3+0*2^2+0*2^1+1*2^0 = 32+0+0+0+0+1 = 33$

Gegenprobe 11

$1*2^3+0*2^2+1*2^1+1*2^0 = 8+0+2+1 = 11$

Hexadezimaldarstellung

- Die Binärdarstellung ist für die Darstellung von Speicherinhalten o. ä. unhandlich.
- Die Dezimaldarstellung ist ungeeignet weil die Umwandlung kompliziert.
- *Hexadezimaldarstellung*: Basis 16
- Vier Bit werden zu einer Hexadezimalziffer 0 ... 9, A ... F zusammengefasst
- Damit kann ein Byte durch zwei Hexadezimalziffern dargestellt werden:

$255_{10} = FF_{16} = 11111111_2$
$124_{10} = 7C_{16} = 01111100_2$

Dezimal	Binär	Hexa-dezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Interpretation von Daten im Computer

2E	78	6D	6C
.	x	m	l
.xml			

$6C6D782E_{16}$

$1'819'113'518_{10}$

$6C6D782E_{16}$

$1.148331807 * 10^{27}$

ASCII Text

int

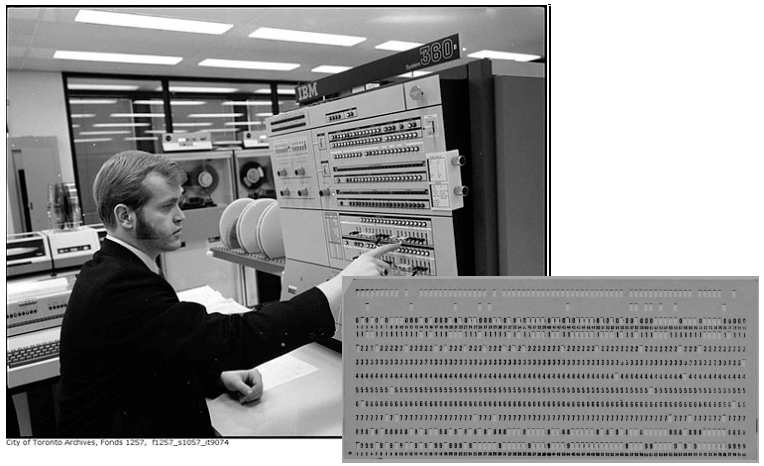
float

Geschichte der Programmierung

Was sind Programme?

- Ein **Programm** ist eine *Anleitung zur Lösung einer Aufgabenstellung*, die so präzise formuliert ist, dass sie “mechanisch" (automatisch) ausgeführt werden kann:
- Mögliche Beschreibungen sind
  - Prosa, Bytefolge, Lochkarte, Programmiersprache
- Müssen auf permanentem Medien gespeichert werden
- Lochkarten zur Beschreibung von Web Mustern (19 Jahrhundert)

Erste Elektronenrechner
Ebenfalls auf Lochkarten



Wie sag ich's meinem Computer?

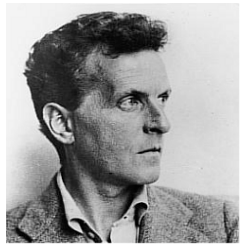
- Die Sprache, mit der wir mit dem Computer "sprechen"
- Anforderungen
  - **aussagekräftig**: alles ausdrückbar
  - **verständlich** für Mensch und Computer
  - **eindeutig**: Computer macht genau das, was man ihm sagt
  - **kompakt**, leicht erlernbar für Menschen (falls nicht natürliche Sprache)
- Problem: Mensch und Computer "sprechen" sehr unterschiedliche Sprachen

Zweck einer Sprache?

- Die ganze Kunst der Sprache besteht darin, verstanden zu werden. (Konfuzius, 551-479 BC)

- Wie viele Sprachen du sprichst, sooft mal bist du Mensch (Johann Wolfgang Goethe)

- Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt (Wittgenstein,1889-1951)



- Neusprech: In Ausdrucksmöglichkeiten beschränken, um damit die Freiheit des Denkens aufzuheben. (G. Orwell 1948)

Problem

Maschinencode

- Folge von Bytes im Speicher
- Sprache, die der Prozessor direkt versteht
  - `78 87 A6 7A`
- einfach, eindeutig für die Maschine

Assembler

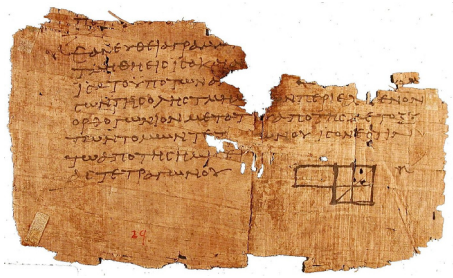
- textuelle Beschreibung der "Bitmuster"
  - `LDA 67` lade Wert aus Speicherzelle 67
  - `MULA #3` multipliziere Wert mit 3
  - `STA 68` speichere Resultat in Speicherzelle 68
- Problem: für Mensch schwer lesbar unübersichtlich

Menschliche Sprache

- Sprache, die der Mensch direkt versteht
- alles ausdrückbar, mächtig
- Problem: die menschliche Sprache ist komplex und per se nicht eindeutig; Satz steht immer in einem impliziten Kontext:
  - Frau S. trägt einen Hut
  - Herr Q. trägt es mit Fassung
  - Frau N. schraubt eine neue Birne in die Fassung

Wie beschreibe ich ein Programm?

Griechisch auf Papyrus



Euklid: ggT Papyrusfragment aus Stoicheia

Schritt 1: Man dividiere p durch q. Dabei erhält man einen Rest r, der zwischen 0 und $q-1$ liegt. (Hier ist die ganzzahlige Division gemeint. Zum Beispiel ergibt die Division von 24 durch 18 den Quotienten 1 und den Rest 6.)

Schritt 2: Wenn $r = 0$ ist, dann ist q der gesuchte größte gemeinsame Teiler. Wenn $r \neq 0$ ist, dann benenne man das bisherige q in p um, das bisherige r in q und wiederhole Schritt 1 und Schritt 2 so lange, bis r = 0 geworden ist.

Assembler

Adresse	Befehl		Kommentar
			Speicherplatzreservierung
00	–		Ein Speicherplatz für p
04	–		Ein Speicherplatz für q
08	–		Ein Speicherplatz für r (am Anfang beliebig)
			Schritt 1
12	LOAD	00	Lade AC mit p
16	MODULO	04	Bilde den Rest von AC/q in AC
20	STORE	08	Speichere AC in r
			Schritt 2
24	IFZERO	48	Wenn $AC = 0$ dann gehe nach 48
28	LOAD	04	Lade AC mit q
32	STORE	00	Speicher AC nach p
36	LOAD	08	Lade AC mit r
40	STORE	04	Speichere AC nach q
44	JUMP	12	Springe zurück nach 12
48	STOP		Halte an; q enthält den größten gemeinsamen Teiler

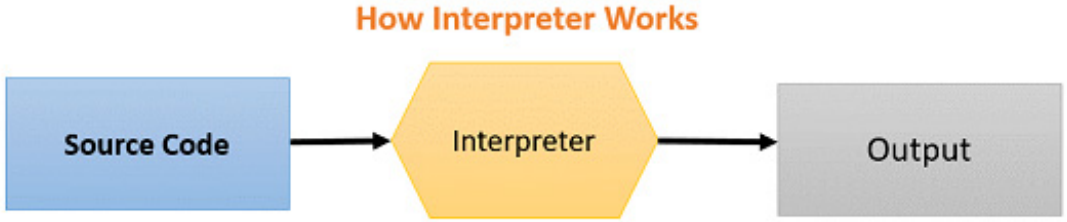
„höhere“ Programmiersprache

```
1  //=== GGT nach Euklid
2  int ggt(int a, int b) {
3     int q, r;
4     while (b > 0) {
5        q = a/b;
6        r = a - q*b;
7        a = b;
8        b = r;
9     }//while
10    return(a);
11 }//ggt
12
13 println ggt(387,216)
```

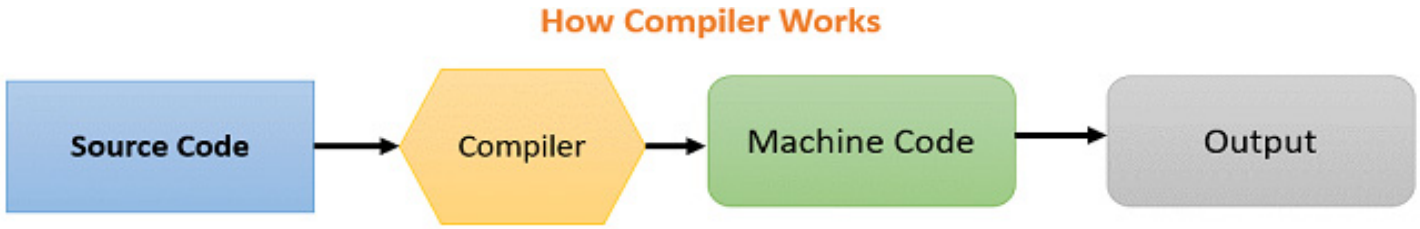
Am Anfang ist $p = 378, q = 216$.
Schritt 1 ergibt $r = 162$.
Schritt 2 ergibt $p = 216, q = 162$.
Schritt 1 ergibt $r = 54$.
Schritt 2 ergibt $p = 162, q = 54$.
Schritt 1 ergibt $r = 0$.

Höhere Programmiersprache

- Sprache die vom Menschen besser verstanden wird.
- Diese Sprache kann aber nicht direkt ausgeführt werden sondern muss **vom Computer** durch ein spezielles Programm (vor-)verarbeitet werden.
- vom **Interpreter**
  - Sprache wird Anweisung für Anweisung ausgeführt

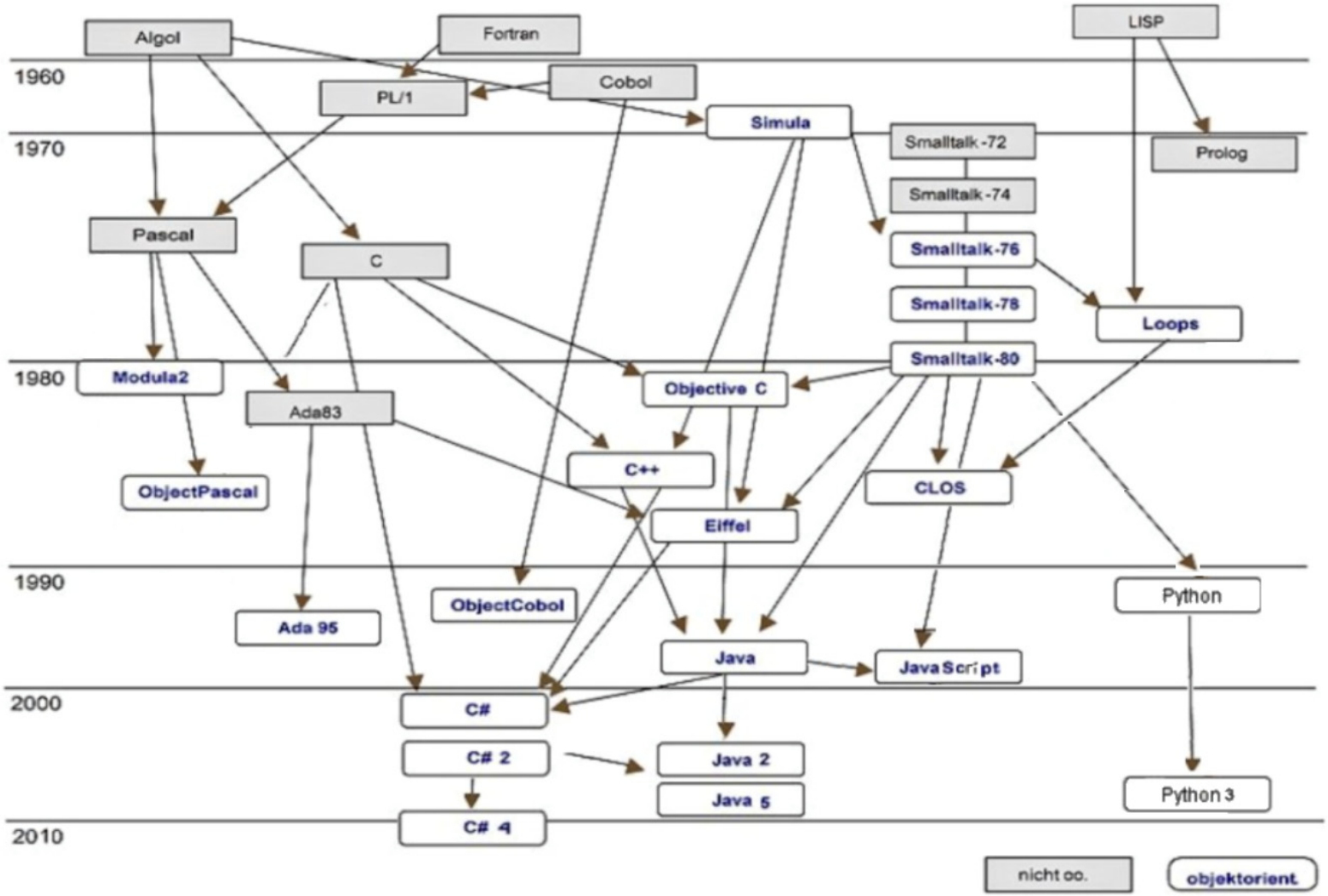


- vom **Compiler** in Maschinen Code übersetzt (compiliert)
  - Als Ganzes in Maschinen Code übersetzt (compiliert)



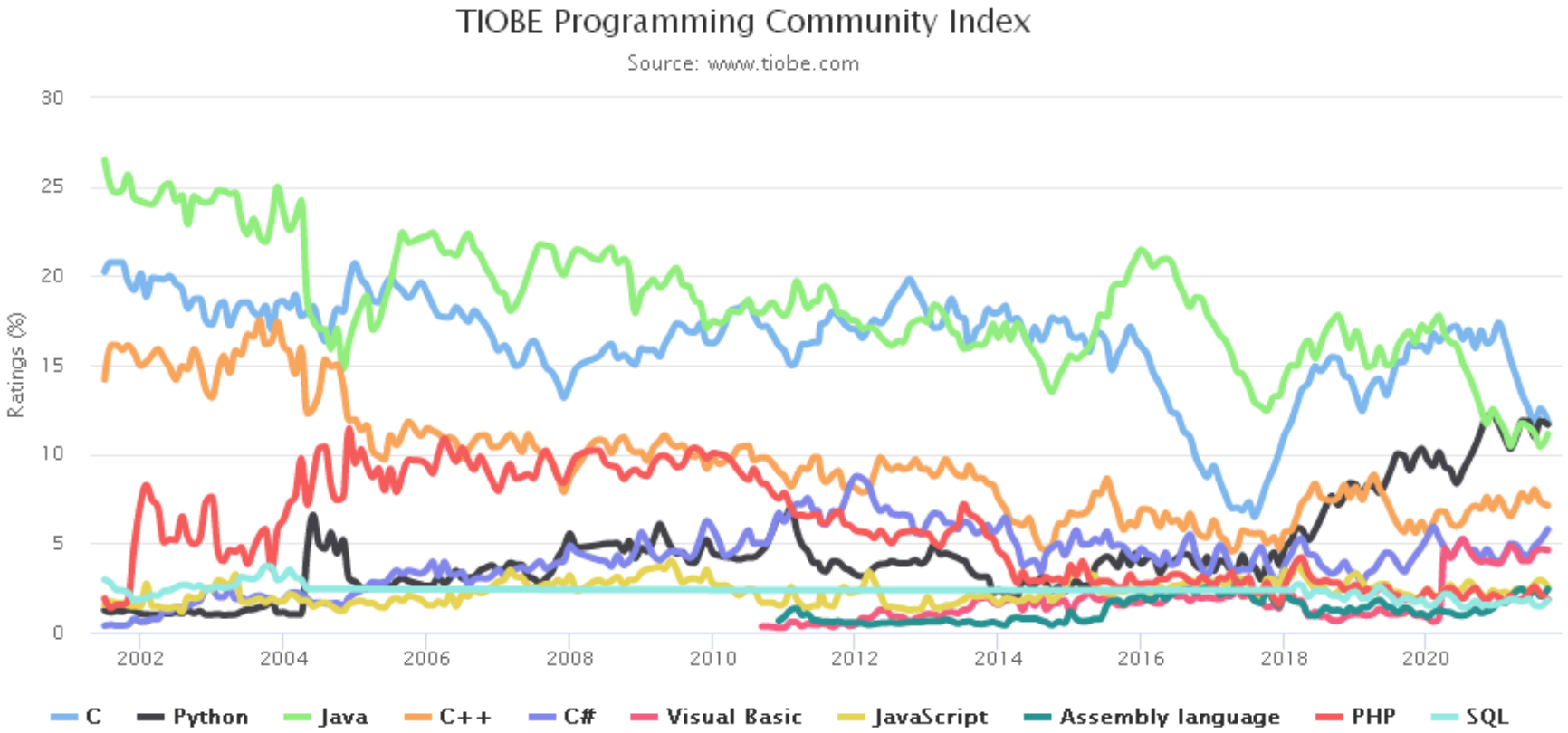
Programmiersprachen

Stammbaum der Programmiersprachen



Popularität von Programmiersprachen Sept. 21

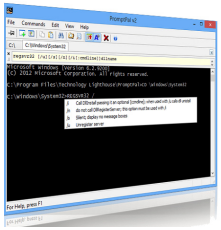
- Gemessen anhand Aktivitäten auf dem Web und in Newsgroups



http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

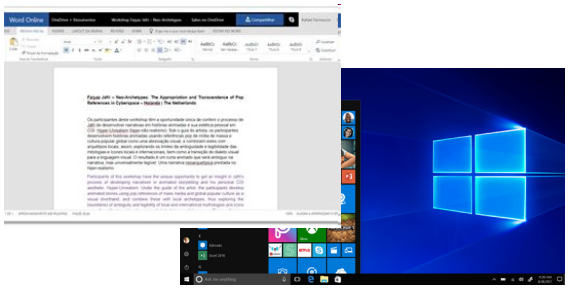
Popularität nach Einsatzgebiet

- Ad-hoc Programme, Skripte
  - schnelle Erstellung, einfache Konstrukte
  - Shell Skripte, einfache Programme
  - **Strukturierte Sprachen**
  - CLI Shell Sprachen, Basic, Python, JavaScript



- Gerätesteuerung
  - kompakt, Ressourcen schonend, effizient
  - Komplexität: Elektrische Zahnbürste bis Flugzeug
  - Früher Assembler heute meistens C (ev. C++)
  - **Strukturierte oder Objektorientierte Sprachen**

- Professionelle Programme, "Enterprise Scale"
  - Hohe Qualität (Fehlerfreiheit), Wartbarkeit
  - Word, Betriebssystem, Enterprise Applications
  - **Objektorientierte Sprachen**
  - Java, C#, C++



Gender Preference

- For new development 2004

Language	Overall	Male	Female	Difference	Significance
HTML	31.1%	30.3%	34.2%	-3.9%	F
SQL	20.1%	20.0%	20.3%	-0.4%	
JavaScript	18.9%	19.3%	17.7%	1.6%	
Visual Basic	18.3%	19.1%	15.2%	3.9%	MMM
Java	17.6%	18.3%	15.0%	3.3%	MM
C++	16.0%	16.8%	12.9%	3.9%	MMM
C	13.2%	13.8%	10.9%	2.9%	M
Oracle	9.7%	9.7%	9.4%	0.3%	
ASP	9.4%	9.7%	8.5%	1.2%	
Basic	7.4%	7.5%	7.1%	0.4%	
Visual C++	7.4%	8.0%	5.2%	2.8%	MMM
Active X	6.2%	6.8%	3.7%	3.1%	MMMM
Perl	6.0%	6.6%	3.8%	2.7%	MMM
OOP	5.0%	5.4%	3.5%	1.9%	MM
Cobol	4.2%	4.4%	3.6%	0.8%	M

ActiveX/COM =

COMMUNICATIONS OF THE ACM January 2004/Vol. 47, No. 1

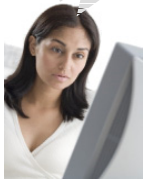
Programmentwicklung

Schritte bei der Programmentwicklung

Fachexperte/Kunde

Informatiker

AnwenderIn



Business Analyse

Programmieren

Problem/Aufgabe

Lösungsweg → **Algorithmus**
(Schrittweises, präzises Verfahren zur Lösung eines Problems)

Folge von Rechenschritten, Wiederholungen, Entscheidungen, bedingten Sprünge
(kann in beliebiger Sprache oder graphisch beschrieben werden)

Programm in einer Programmiersprache
(Java, Python, …..)

Input
Tastatur, Maus, Datei, Netzwerk →

Ausführung des Programms im Computer

→ Output
Bildschirm, Drucker, Datei, Netzwerk

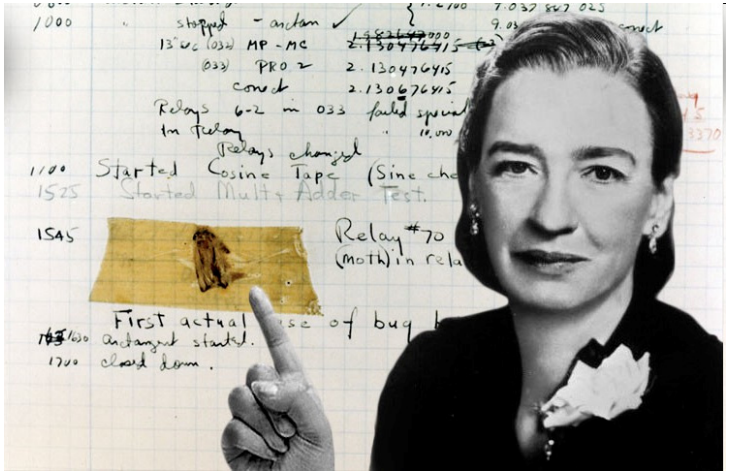
Komplexität: Million Lines of Code (Vergleich)

Ab einer bestimmten Grösse reichen (Struktur) Mittel der einfachen strukturierten Programmierung nicht mehr

Zum Vergleich Mensch: Gene 3.3 Milliarden "Lines of Code"

Programmfehler = Bug

- Fehlersuche und Behebung
- Einer der ersten Computerfehler durch Motte ausgelöst, die einen Kurzschluss verursacht hat.
- Entdeckt von Grace Hopper (Informatikpionierin)
  - hat besagte (tote) Motte in den Fehlerbericht geklebt



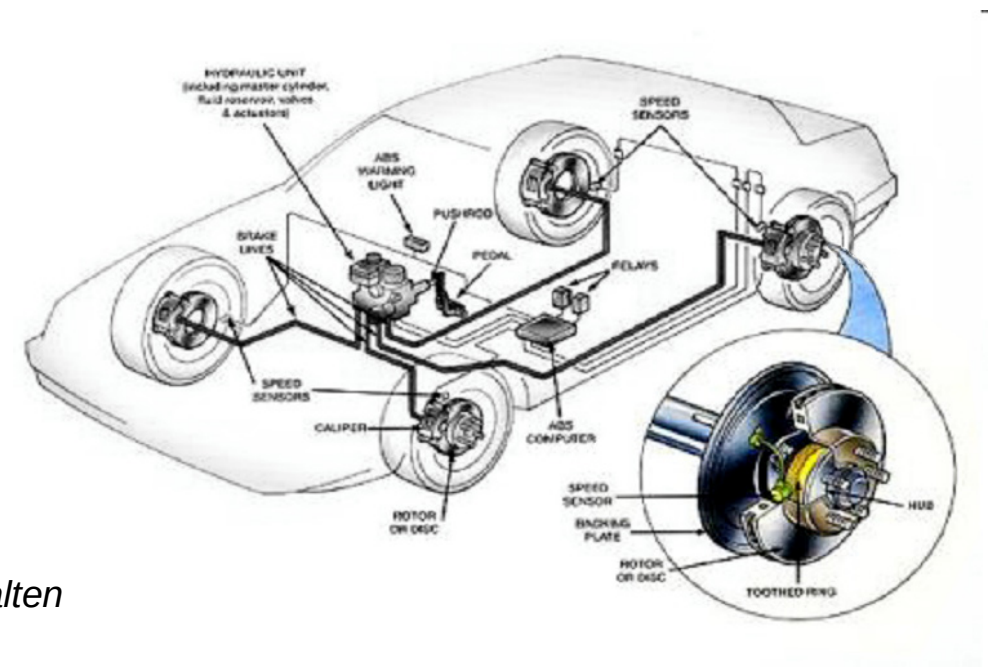
Komplexität von Programmen

- Programme werden schnell gross und damit unübersichtlich
- Die Komplexität führt zu Problemen bei
  - Entwicklung
  - Testen
  - Wartung
  - Wiederverwendung
- Um Komplexität in den Griff zu bekommen werden grundsätzliche Denkansätze (Paradigmen) angewandt
- **Strukturierte** Programmierung = **Prozedurale** Programmierung
  - **Prozedur** und **Funktion** (hier) als Synonyme gebraucht
- **Objektorientierte Programmierung (später)**

Strukturierte Programmierung

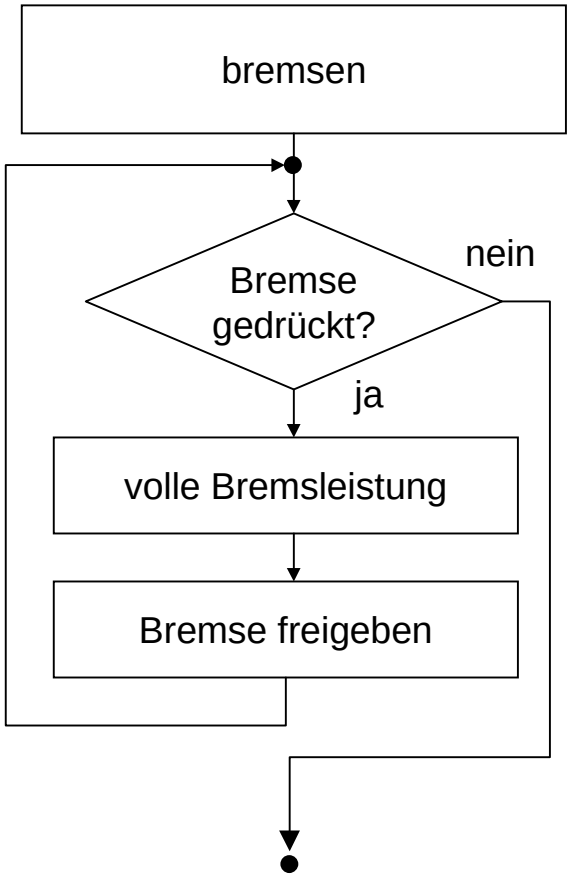
- ABS
  - Daten festlegen:
    - *Bremspedalstand*
    - *Hydraulik*
  - Ablauf beschreiben
    - *Bremspedal wird voll gedrückt*
    - *Volle Bremsleistung auf Bremse*
    - *Bremse blockieren*
    - *Bremse kurz freigeben*
    - *Volle Bremsleistung auf Bremse*
    - *...*
  - für komplexere Schritte wird ein **zusammenfassender benannte Unterschritt eingeführt**
    - *Bremse in "optimalen Zustand" halten*
- Das Ist das Paradigma der **prozeduralen/strukturierten Programmierung**

Ablauf wird in Unterabläufe strukturiert



Graphische strukturierte Programmierung

Ablaufdiagram (Flow-Chart)

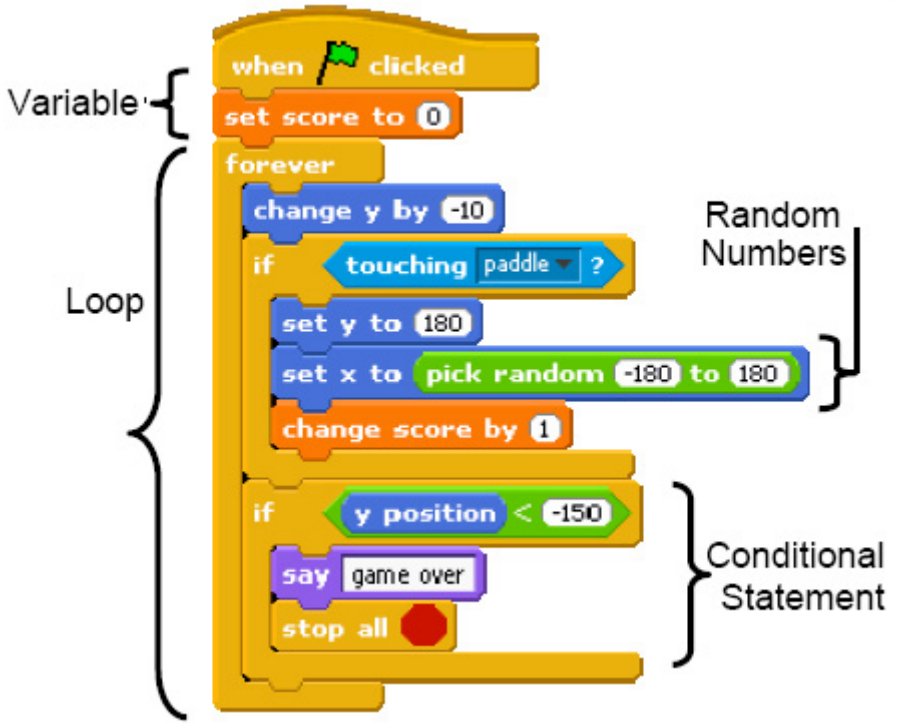


Struktogramm

+ erzwingen strukturiertes Programmieren
  (keine beliebige Sprünge → kein "Spagetti-Code"
- aufwendig zu zeichnen, schwer zu ändern

… graphische strukturierte Programmierung

- Scratch als graphische Programmiersprache (LP 21)
  - an Struktorgrammen angelehnt



Textuelle strukturierte Programmierung

```c
/* helloworld.c */
#include <stdio.h>

/* Hauptprogramm */
int main() {
  printf("Hello World in C\n");
  return 0;
}
```

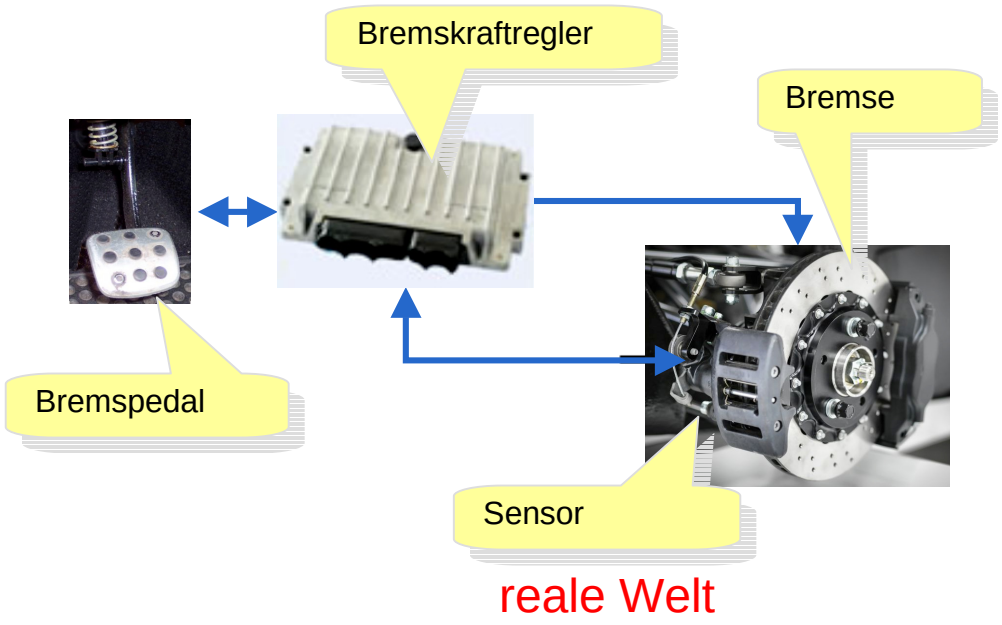
- Kommentar
- Bibliothek einfügen
- Funktionsdefinition
  Funktion *main* gibt einen Wert vom Typ *int* zurück (eine Zahl)
- Funktionsaufruf
  Funktion *printf* gibt einen Text (in "...") aus

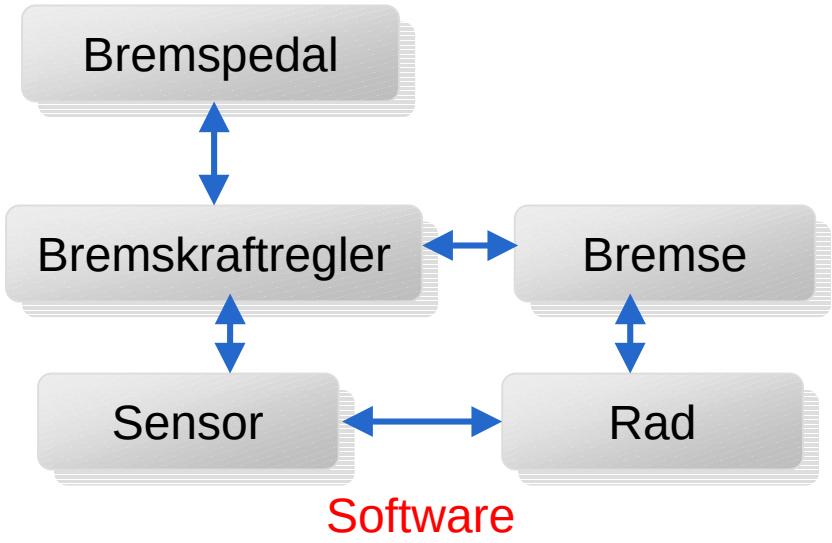
Objektorientierte Programmierung (OOP)

outlook

Die Software wird analog den Objekten der realen Welt modelliert bzw. strukturiert

- Man spricht von Objekten
  - mit Zustand und Verhalten
- Die untereinander interagieren
  - sie rufen sich gegenseitige auf





Objektorientierte Programmierung

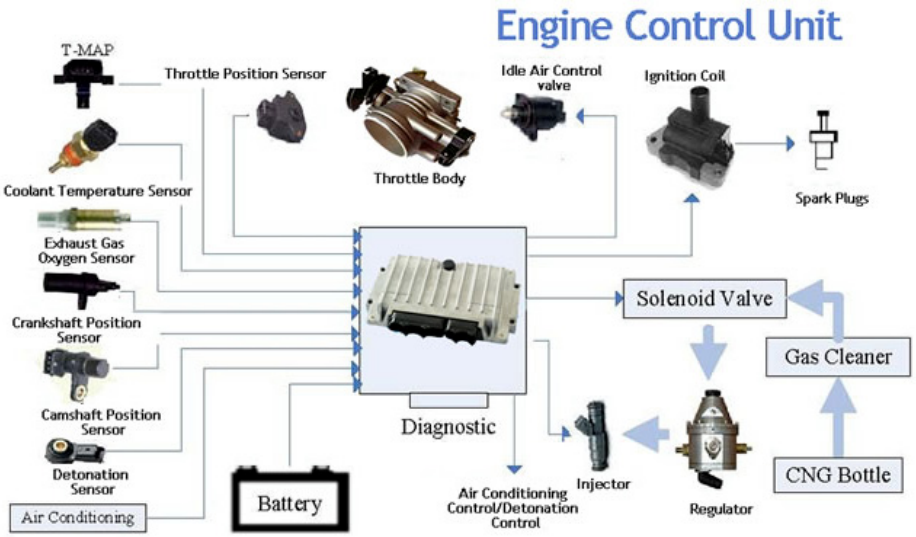
outlook

- **Heute wichtigstes Programmierparadigma**
- analog einem technischen System, z.B. Auto:
  - Antriebssystem (Motor, Kupplung, Räder,...)
  - Bremssystem (Bremsen, Bremspedal, Bremsflüssigkeit,...)
  - Lichtanlage (Lichter, Batterie,...)
- Die Einzelteile sind die **Objekte**
  - Haben einen **Zustand**
  - Haben ein **Verhalten**

- **Klassen sind die "Baupläne" für Objekte**
  - Objekte werden zur Laufzeit anhand der Klassen erzeugt (instanziert)
  - Der gewünschte Programmablauf ergibt sich durch die Interaktion der Objekte

- **Objektorientierte Sprachen**
  - Java, C#, C++,

Softwaresystem aus Objekten/Klassen aufbauen

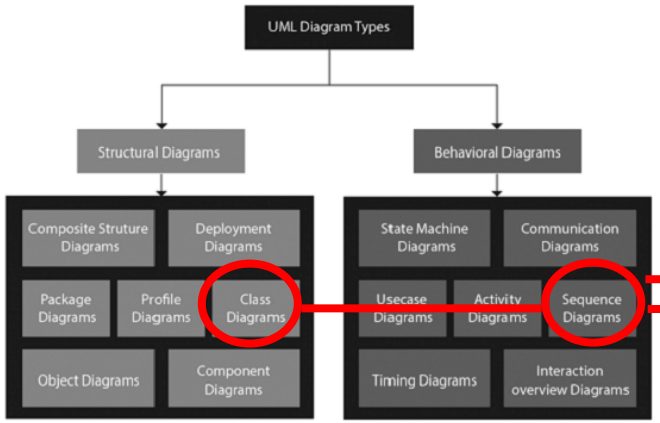


Special Version

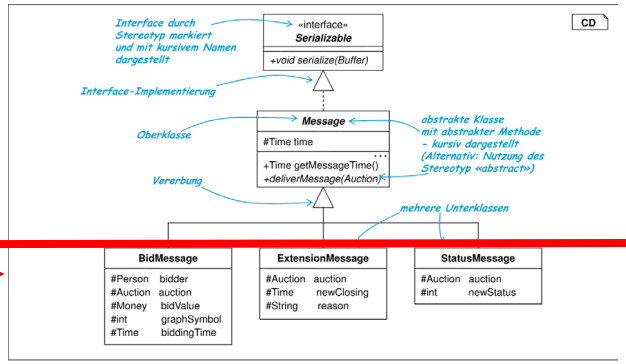
… die UML Notation

outlook

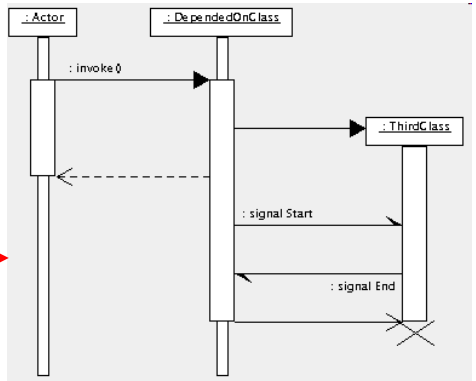
- UML **Unified Modeling Language**
  - Verschiedene Diagramme um strukturelle und dynamische Aspekte zu modellieren
  - Zwei Gruppen von Diagrammen
    - *solche die die Struktur beschreiben*
    - *solche die das Verhalten beschreiben*



Diagrammtypen



Klassendiagramm
statische Struktur



Sequenzdiagramm
dynamischer Ablauf

Textuelle objektorientierte Programmierung

outlook

- Java Programm mit Klassen (Bauplan) und Methoden

```
public class BreakControler {
          Break[] breaks;
          double[] breakPressure;
   long timeElapese;
   double[] sensorValue;

   void break() {
     if (timeElapses < 20) {
                    if
sensorValue[0] > 10 {

break[0].press();
                    }
     }
   }
}
```

Zustand in Form von (Instanz-) Variablen

Verhalten in Form von Methoden

Erstes C-Programm

Hello World

```c
/* helloworld.c */
#include <stdio.h>

/* Hauptprogramm */
int main() {
  printf("Hello World in C\n");
  return 0;
}
```

- Kommentar
- Bibliothek einfügen
- Funktionsdefinition
  Funktion *main* gibt einen Wert vom Typ *int* zurück (eine Zahl)
- Funktionsaufruf
  Funktion *printf* gibt einen Text (in "...") aus

Programm Editor

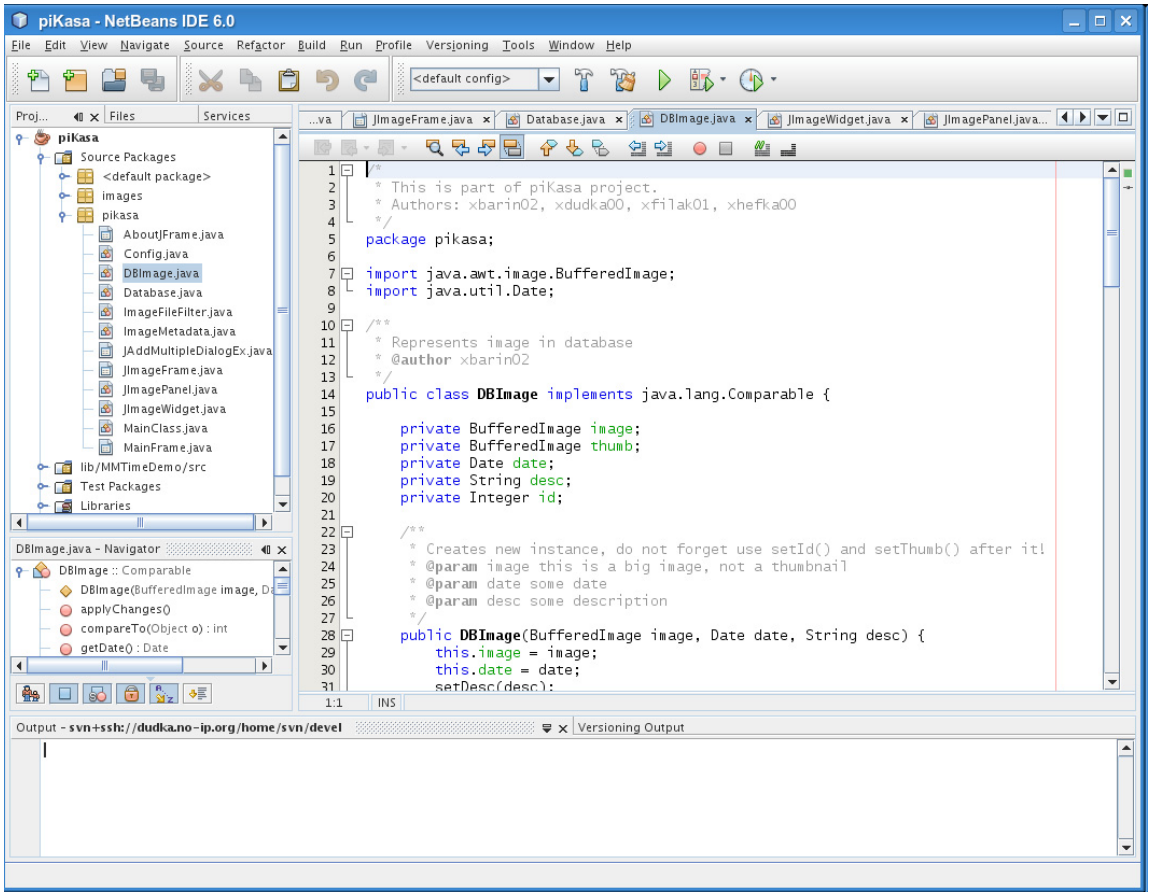
- Programm, das es erlaubt, in Dateien gespeicherte Texte anzulegen und zu bearbeiten
- Im Gegensatz zur Textverarbeitung wie Word: Editor verarbeitet reinen Text, keine Formatierungen (Schriftgrösse etc.)
- Beispiele:
  - Windows: *Notepad*
  - Linux/Unix: vi

IDE - Entwicklungsumgebung

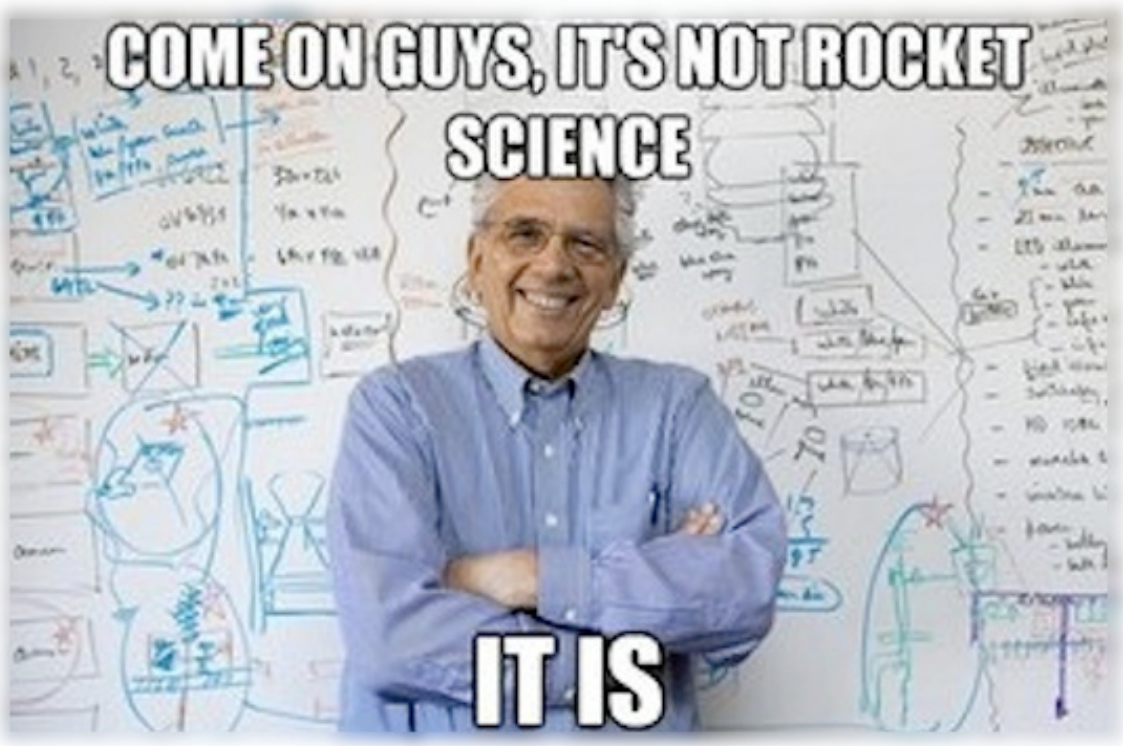
- Eine Entwicklungsumgebung (IDE, Integrated Development Environment) stellt Werkzeuge zum Entwickeln von Programmen zur Verfügung

- Dazu gehören:
  Quellcode-Editor,
  Compiler,
  Debugger

- Beispiele:
  Visual Studio,
  NetBeans,
  XCode, Eclipse, …



Fragen ?



C