ETH zürich
Chair of Software Engineering

Einführung in die Programmierung

Prof. Dr. Bertrand Meyer

Vorlesung 1: Willkommen & Übersicht

Ziele der Vorlesung

Nach erfolgreichem Abschluss dieser Vorlesung werden Sie:

- Die Schlüsselkonzepte des Programmierens kennen
- Viele verschiedene Programmierprobleme aus verschiedenen Bereichen lösen können
- Einige grundsätzlichen Hardware- und Softwarewerkzeuge kennen
- Eine Programmiersprache beherrschen: Eiffel
- Die Grundkonzepte des Designs, der Implementierung und der Wartung von Softwaresystemen kennen ("software engineering")

Über mich

An der ETH seit Ende 2001, Professor für Software Engineering

Grossteil meiner Karriere in der Industrie, zuletzt bei *Eiffel Software* in Santa Barbara, Kalifornien

Professor an der University of California, Santa Barbara in den 80ern

Forschungsthemen: Software Engineering, Programmiersprachen, OO Programmierung, Nebenläufige Programmierung (*concurrency*), Programbeweise, Testen, Entwicklungsumgebungen, Persistenz

Kontaktdaten:

- E-mail: Bertrand.Meyer@inf.ethz.ch, Büro: RZ J22
- Sekretariat: Claudia Günthart, 044 632 83 46
  Claudia.Guenthart@inf.ethz.ch, Büro: RZ J7

Sprechstunden: Mittwochs während des Semesters, kontaktieren Sie Frau Günthart

Sprache

Die Sprache für diese Vorlesung ist SÜF

Geschwindigkeit auf Deutsch

Der Turbo-Knopf (nicht noch gefunden)

Auf der positiven Seite

30 Sekunden

Das Team der Assistenten

Marco Piccioni
(Back Office and TA)

Alexey Kolesnichenko
(Koordinator and TA)

Loic Ciccone

Salvatore di Girolamo

Jan Hązła

Yu Pei (Max)

Roman Schmocker

Petar Tsankov

Felix Laufenberg

Tim Linggi

Sandro Marcon

Sonja Menzi

Nicolas Truessel

Christian Vonrüti

Lorin Weilenmann

Philipp Wirth

Marie Woon

Back Office

Marco Piccioni

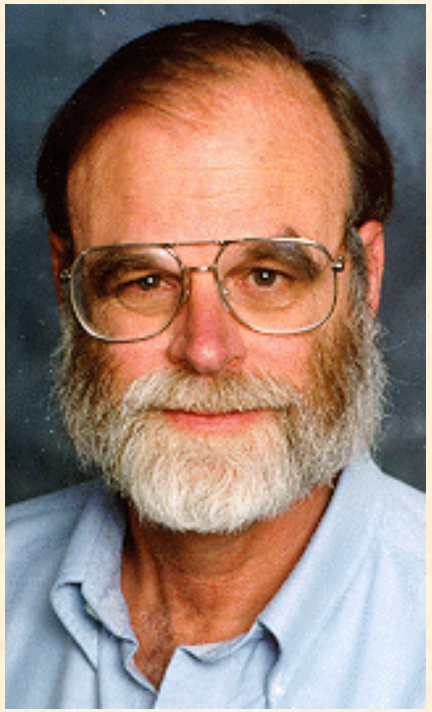
Koordinator

Alexey Kolesnichenko

Gruppe Jim Gray: Marco Piccioni

Mailingliste: se-info1-gray@lists.inf.ethz.ch

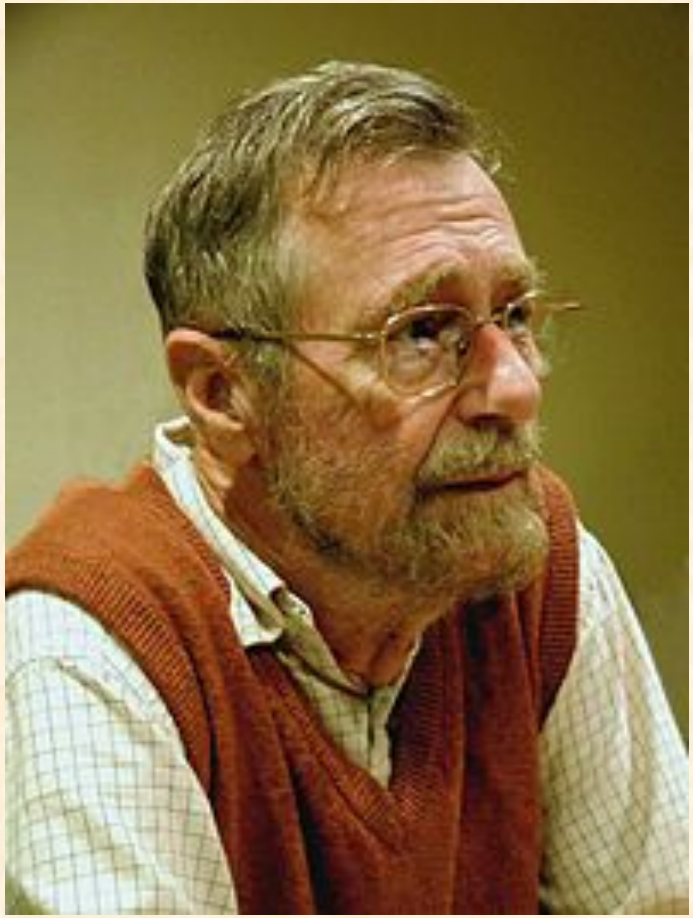
E-mail: marco.piccioni@inf.ethz.ch



Gruppe Edsger Dijkstra: Alexey Kolesnichenko

Mailingliste: se-info1-dijkstra@lists.inf.ethz.ch

E-mail: alexeyk@inf.ethz.ch



Gruppe John von Neumann: (Max) Yu Pei

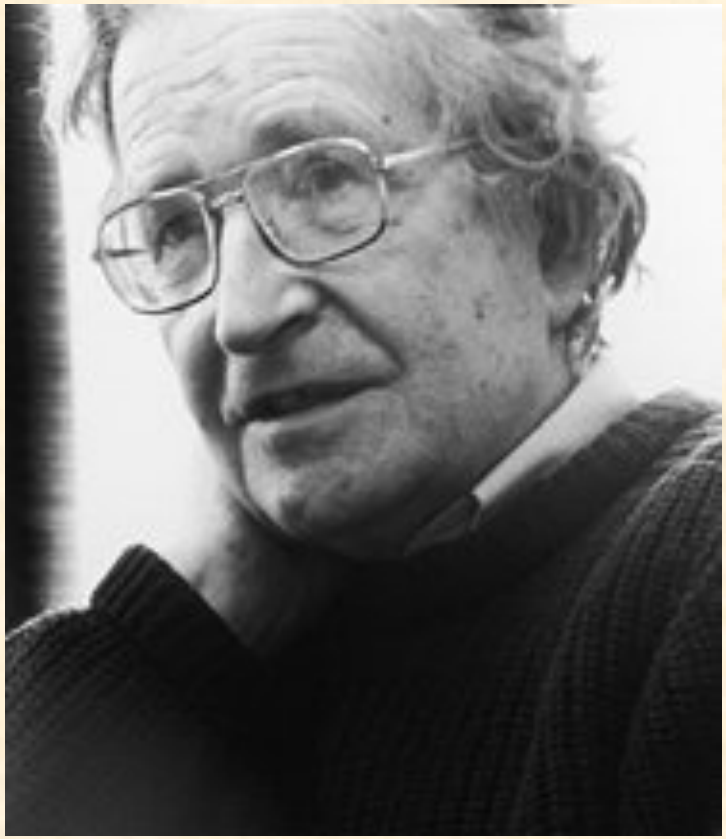
Mailingliste: se-info1-neumann@lists.inf.ethz.ch

E-mail: yu.pei@inf.ethz.ch

Gruppe Noam Chomsky: Loic Ciccone

Mailingliste: se-info1-chomsky@lists.inf.ethz.ch

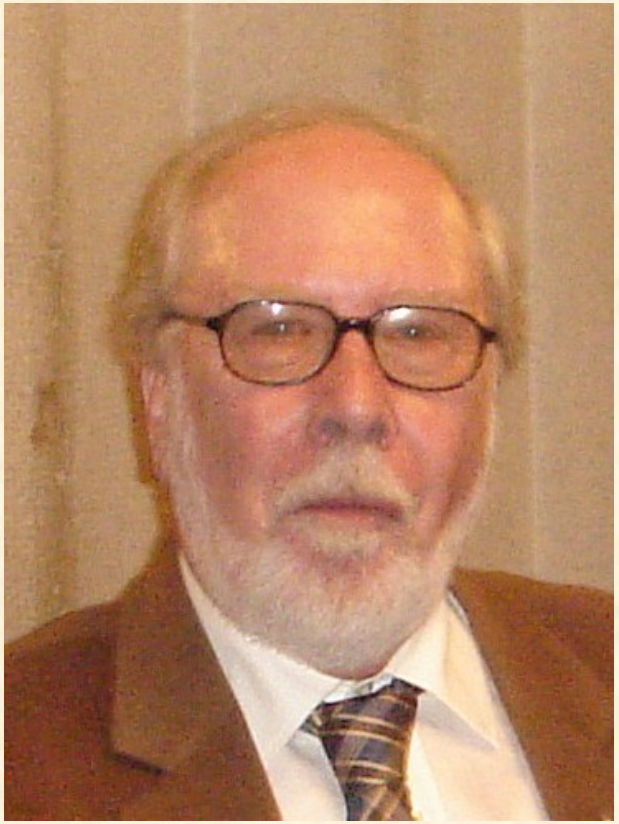
E-mail: loic.ciccone@inf.ethz.ch



Gruppe Niklaus Wirth: Salvatore di Girolamo

Mailingliste: se-info1-wirth@lists.inf.ethz.ch

E-mail: salvatore.digirolamo@inf.ethz.ch



Gruppe Alan Turing: Jan Hązła

Mailingliste: se-info1-turing@lists.inf.ethz.ch

E-mail: jan.hazla@inf.ethz.ch

Gruppe Tony Hoare: Roman Schmocker

Mailingliste: se-info1-hoare@lists.inf.ethz.ch

E-mail: romasch@student.ethz.ch

Gruppe John Backus: Petar Tsankov

Mailingliste: se-info1-backus@lists.inf.ethz.ch

E-mail: ptsankov@inf.ethz.ch

Gruppe Vinton Cerf: Felix Laufenberg

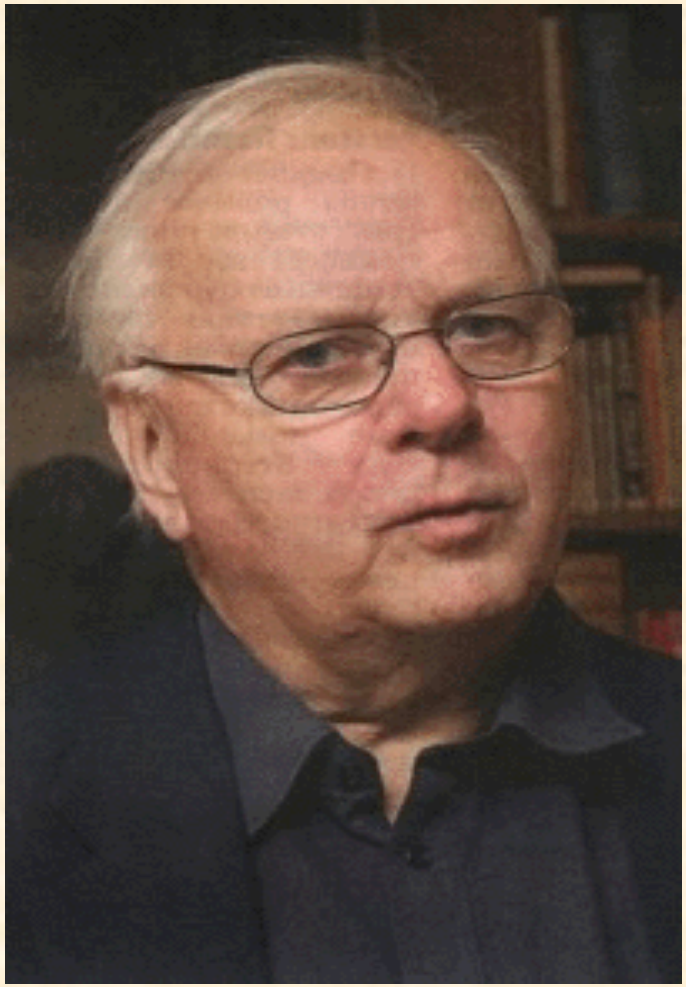
Mailingliste: se-info1-cerf@lists.inf.ethz.ch

E-mail: felix.laufenberg@gmx.de

Gruppe Kristen Nygaard: Tim Linggi

Mailingliste: se-info1-nygaard@lists.inf.ethz.ch

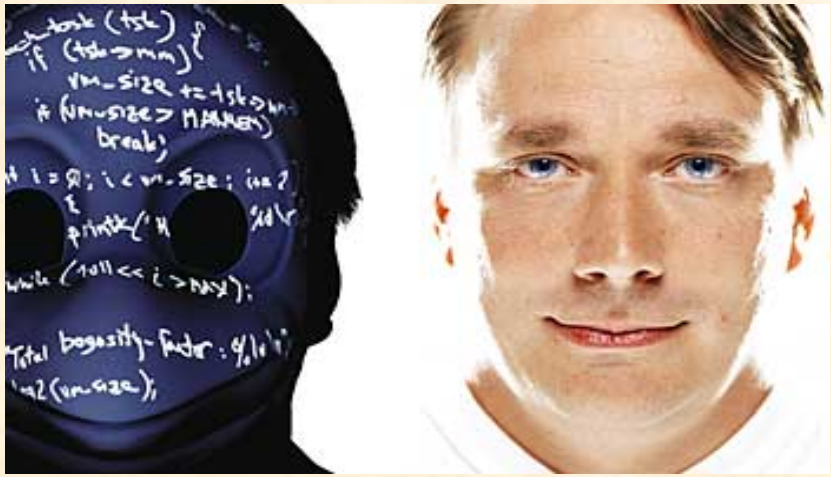
E-mail: linggit@student.ethz.ch



Gruppe Linus Torvalds: Sandro Marcon

Mailingliste: se-info1-torvalds@lists.inf.ethz.ch

E-mail: smarcon@student.ethz.ch



Gruppe Ada Lovelace: Sonja Menzi

Mailingliste: se-info1-lovelace@lists.inf.ethz.ch

E-mail: smenzi@student.ethz.ch

Gruppe Adele Goldberg: Nicolas Truessel

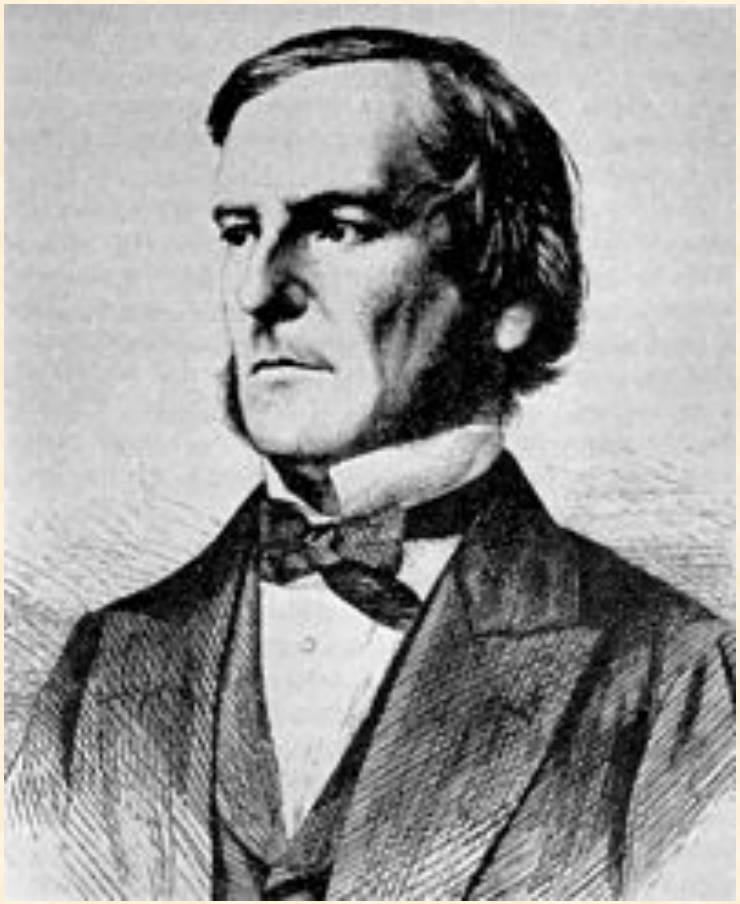
Mailingliste: se-info1-goldberg@lists.inf.ethz.ch

E-mail: nicolast@student.ethz.ch

Gruppe George Boole: Christian Vonrüti

Mailingliste: se-info1-boole@lists.inf.ethz.ch

E-mail: christian.vonrueti@gmail.com



Gruppe Donald Knuth: Lorin Weilenmann

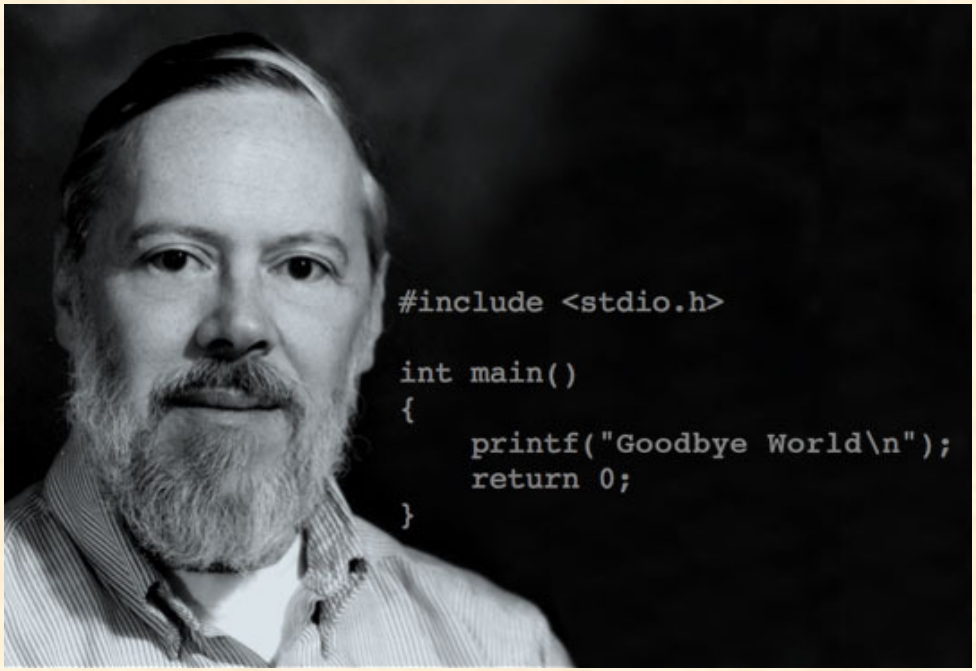
Mailingliste: se-info1-knuth@lists.inf.ethz.ch

E-mail: wlorin@student.ethz.ch

Gruppe Dennis Ritchie: Philipp Wirth

Mailingliste: se-info1-ritchie@lists.inf.ethz.ch

E-mail: pwirth@ethz.ch



Gruppe Barbara Liskov: Marie Woon

Mailingliste: se-info1-liskov@lists.inf.ethz.ch

E-mail: mwoon@student.ethz.ch

Übungsgruppen

Sie füllen eine Umfrage aus. Den Link dazu finden sie auf der Webseite, oder wird Ihnen heute Nachmittag per E-Mail geschickt. Sie wählen zwischen

- Drei Niveaus
- Zwei Sprachen

Tragen Sie sich bis Mittag auf mystudies.ethz.ch für den Kurs ein, um das Email zu erhalten

Füllen Sie die Umfrage bis Donnerstag Abend aus. Am Montag finden Sie die Einteilung in die Übungsgruppen auf der Webseite

Falls Sie gute Gründe für einen Gruppenwechsel haben: fragen Sie Alexey

Sprachen

Unsere Assistenten sprechen verschiedene Sprachen:

- Deutsch
- Englisch
- Italienisch
- Chinesisch
- Russisch
- …

Die Übungsgruppen werden auf deutsch oder auf englisch gehalten.

Stundenplan

Vorlesungen:

- Montags, 13:15 – 15:00, HG E7
- Dienstags, 8:15 – 10:00, HG E7

Übungsstunden:

- 17 Gruppen
  - Mittwoch, 8:15 – 10:00, in verschiedenen Räumen
  - Mittwoch, 15:15 – 17:00, in verschiedenen Räumen

Webseite der Vorlesung

Webseite:

http://se.ethz.ch/courses/2015b_fall/eprog/

→ Zweimal wöchentlich anschauen

Deutsche und Englische Versionen, beide sind aktuell

Vorlesungsunterlagen:

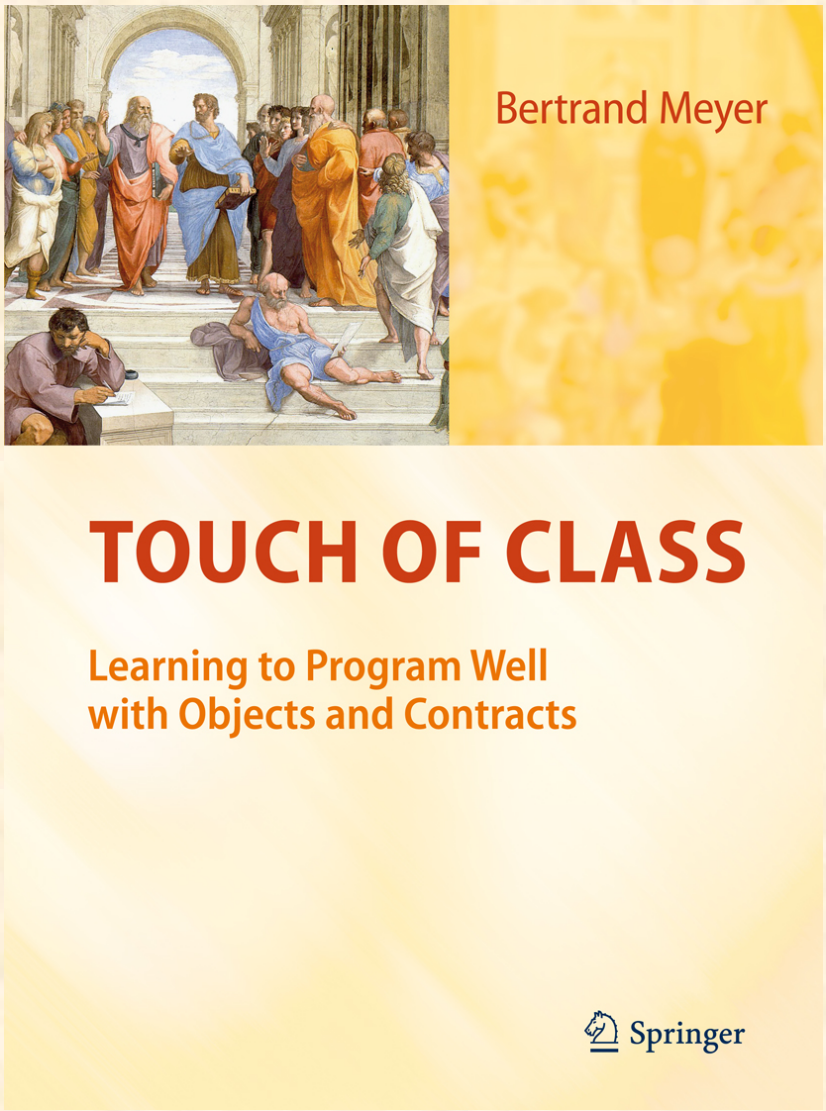
- Folien der Vorlesung
- Buch: *Touch of Class*

  Siehe nächste Folie

Übungsunterlagen:

- Übungen
- Musterlösungen

Das Buch zur Vorlesung



Online-Version des Buches

Möglich aus dem Netz der ETH

URL: siehe Vorlesungswebseite

Buchverkauf: nächsten Montag (in der Pause)

Montag, 21.9.15 um 14:00

vor dem Hörsaal HG E

Berichtigte Version 2013!

ETH STORE

Meyer, Bertrand

Touch of Class

Fr. 72.00 / **mit Legi Fr. 54.00**

Büchertisch vor dem Hörsaal

Nur Barbezahlung möglich!

ETH Store Genossenschaft
Leonhardstrasse 34, MM C 88.1, 8092 Zürich
Öffnungszeiten: Montag – Freitag: 09.00 – 18.00 Uhr

Email: store@store.ethz.ch

Neu für dieses Jahr

Der Info-1 MOOC (3rd edition, 2015/2016)
http://webcourses.inf.ethz.ch/se_courses/introduction_to_programming/main_page/

Der edX MOOC (3rd edition), startet am 22. September 2015,
https://www.edx.org/course/computing-art-magic-science-ethx-cams-2x

(Marco Piccioni)

Elektronische Foren

Diskussionsforen:
Hilfeforum für die gesamte Vorlesung:
http://forum.vis.ethz.ch/

Mailingliste für jede Übungsgruppe

Ratschläge und Regeln:

- Benutzen Sie das VIS-Forum und die Mailinglisten! Programmieren zu lernen ist schwierig: Nutzen Sie jede Hilfe, die Ihnen angeboten wird.
- Es gibt keinen Grund, schüchtern zu sein. Es gibt keine dummen Fragen.
- Kritik ist willkommen, seien Sie aber immer freundlich und halten Sie sich an die Etiquette.

Falls Sie einen Laptop brauchen...

Das NEPTUN-Programm der ETH verkauft Laptops zu guten Preisen

Thinkpad (Lenovo), HP oder Apple

Sie wählen das Betriebssysstem: Windows, Linux, MacOS

Zeitlich begrenzter Verkauf: siehe www.neptun.ethz.ch

Übungen

Die Übungen sind ein wichtiger Bestandteil der Vorlesung

- Eine Übung pro Woche (ca 10 insgesamt)
- Zwei Mock exams

Für Ihre Übungsabgabe sollten Sie:

- nachweisen, dass Sie die Aufgaben zu lösen versucht haben

Absenzen wegen Militärdienst oder Krankheit: kontaktieren Sie Ihren Assistenten.

Benotung

Die Grundregeln sind von der ETH diktiert, die Feinheiten von uns bestimmt:

- **Die Note beruht auf der Leistung in der Prüfung vom kommenden August**

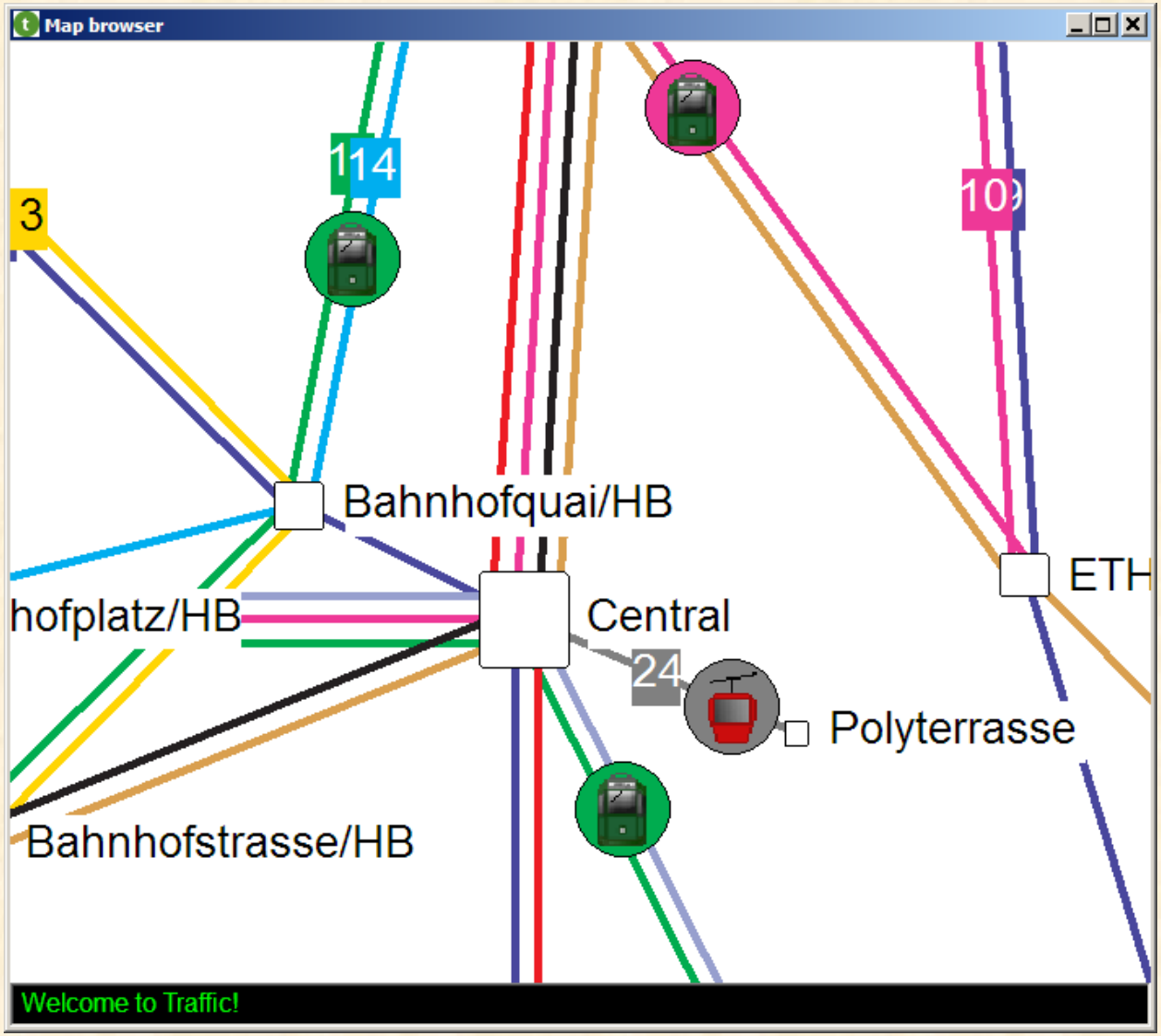
Die Software

Die Übungen bauen auf der Bibliothek Traffic auf

Anwendungsgebiet: öffentlicher Verkehr in einer Stadt (benutzt Zürich als Beispiel)

Übung 1 (auf der Webseite) führt Sie durch die Installation von EiffelStudio und zeigt Traffic

Entdecken Sie Traffic



Warnung

Natürlich ist nicht alles perfekt.

Traffic beinhaltet wahrscheinlich Fehler (“bugs”), und das Buch wahrscheinlich auch.

(Fehlerliste: http://touch.ethz.ch -> *Errata*)

Aber:

- Wir versuchen, die Fehler so schnell wie möglich zu korrigieren.
- Schieben Sie beim Ausprobieren die Schuld jeweils nicht zuerst der Software in die Schuhe. Vielleicht folgt sie bloss Ihren Anweisungen.

Weshalb diese Lehrmethode?

Mit anderen Lehrmethoden würden Sie nur mit kleinen selber geschriebenen Programmen arbeiten

Wir verwenden ein vorgegebenes Softwaresystem; benutzen Sie dieses als Beispiel und Inspiration

Sie benutzen die Software durch ihre abstrakten Schnittstellen (auch bekannt als **Verträge** (contracts))

Sie verwandeln sich vom Konsumenten zum Produzenten: outside-in

Traffic ist grafisch und macht Spass!

Im besten Fall verstehen Sie am Ende die gesamte Software.

Dann können Sie auch neues hinzufügen

Programmiererfahrung

Falls Sie bereits programmiert haben, nutzen Sie diesen Vorteil, aber seien Sie auch offen für eine neue Sichtweise; erkunden Sie Traffic

Falls Sie noch nie programmiert haben, keine Angst; es kann anfangs schwierig sein, aber Sie werden es schaffen.

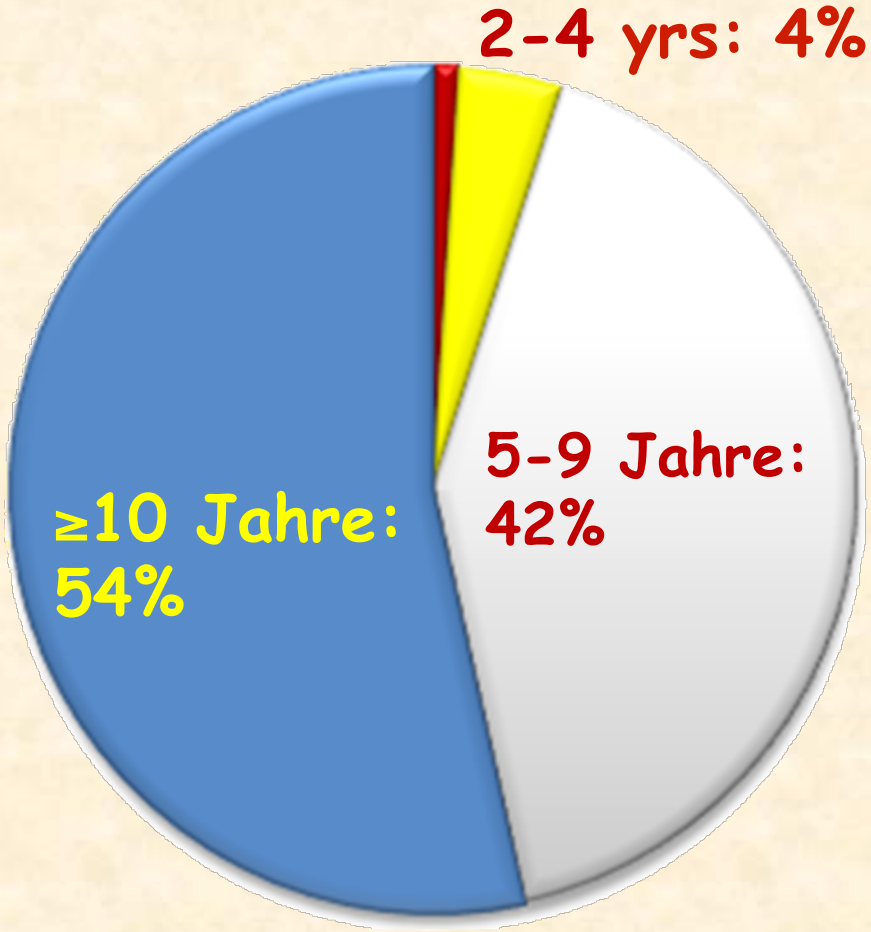
Mathematisches Wissen ist genauso nützlich wie Programmiererfahrung

Vorkenntnisse eines Informatikstudenten im ersten Semester (2003-2008)

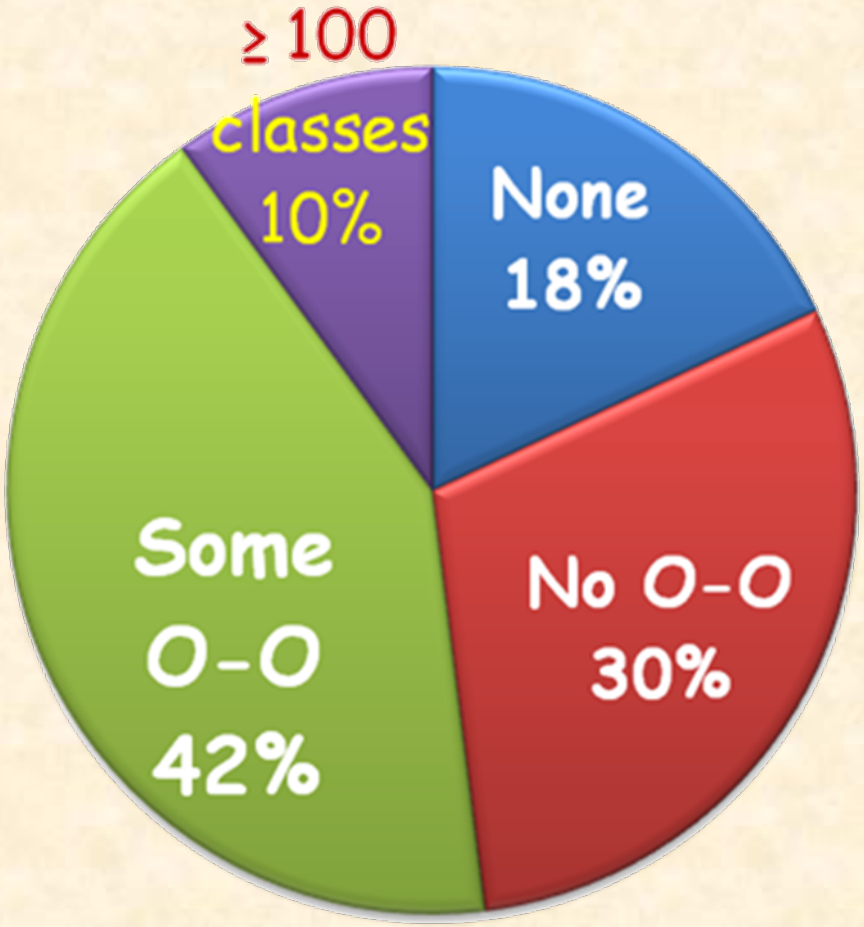
	2008	Frühere Jahre
Erfahrung mit Computern	≤ 2 Jahre: 1%	(0%, 0%, 0%, 1%)
	2 bis 4 Jahre: 3%	(1%, 3%, 4%, 1%, 6%)
	5 bis 9 Jahre: 34%	(39%, 35%, 48%, 35%)
	≥ 10 Jahre: 62%	(61%, 62%, 48%, 63%)
Programmier-erfahrung	Keine: 18%	(12%, 19%, 18%, 14%)
	Keine OOP: 22%	(20%, 26%, 33%, 38%)
	≥ 100 Klassen: 17%	(8%, 11%, 15%, 10%, 5%)

Vorkenntnisse eines Informatikstudenten im ersten Semester

Erfahrung mit Computern



Programmiererfahrung



Durschnitt über 6 Jahre, 2003-2008

Ein paar Regeln

Kein Web-Browsing, Email usw. in der Vorlesung

Keine persönliche Gespräche

Minds open...

... Laptops closed.

Ratschläge

- Besuchen Sie alle Vorlesungen
- Lesen Sie die Unterlagen — das Buch und die Folien — jeweils vor den Vorlesungen
  (Bem.: Folien werden häufig nach der Vorlesung aktualisiert)
- Nehmen Sie eine Druckversion der Folien mit und machen Sie sich Notizen
- Besuchen Sie alle Übungsstunden
- Machen Sie alle Übungen
- Falls Sie etwas nicht verstehen, fragen Sie nach
  (es gibt keine dummen Fragen)

Persönliche Ratschläge

Erfolgreich studieren (insbesondere an der ETH):

- Sie bestimmen selbst, was Sie wann tun
- Nutzen Sie die Möglichkeiten der ETH
  - Talks gehalten von Forschern von anderen Unis
  - Konferenzen
  - Bibliotheken
  - Experimente
  - Projekte
- Sprechen Sie mit Professoren und Assistenten
- Lesen Sie die Webseiten des Departements und des Chair of Software Engineering
- Halten Sie Ausschau nach Vorlesungen mit Projekten und anderen Möglichkeiten, individuell zu arbeiten

Noch mehr persönliche Ratschläge

- Besuchen Sie die Vorlesungen
- Besuchen Sie die Übungsstunden
- Lesen und drucken Sie die Folien vor der Stunde
- Machen Sie sich Notizen
- Schliessen Sie sich zu Studiengruppen zusammen
- Besuchen Sie auch Vorlesungen zu informatikfremden Themen, vor allem während den ersten zwei Jahren
- Bereiten Sie sich früh auf die Prüfung vor

- Bewahren Sie eine kritische, forschende Einstellung

Ziele der Vorlesung

Nach erfolgreichem Abschluss dieser Vorlesung werden Sie:

- Die Schlüsselkonzepte des Programmierens kennen
- Viele verschiedene Programmierprobleme aus verschiedenen Bereichen lösen können
- Die grundsätzlichen Hardware- und Softwarewerkzeuge kennen
- Eine Programmiersprache beherrschen: Eiffel
- Die Grundkonzepte des Designs, der Implementierung und der Wartung von Softwaresystemen kennen ("software engineering").

Themen

- Was ist Software?
- Objekte & Programme
- Schnittstellen und das Klassenkonzept
- Logik und Verträge (contracts)
- Das Laufzeitmodell: Objekterzeugung, Referenzen
- Syntaxbeschreibung
- Kontrollstrukturen
- Vererbung
- Generik
- Rekursion
- Datenstrukturen
- Ereignisgesteuerte Programmierung & Agents
- Topologisches Sortieren
- Einführung ins Software Engineering

Die Industrie der reinen Ideen

Softwareingenieure bauen Maschinen

Diese Maschinen kann man nicht berühren, treten oder fallen lassen: sie sind immateriell

Aber es sind trotzdem Maschinen

Wir nennen sie Programme oder Systeme

Um ein Programm auszuführen, benötigt man eine materielle Maschine: einen Computer

Computer und technische Geräte: Hardware

Programme und der damit verbundene intellektuelle Wert: Software

Software, wohin man auch schaut

Banken: verwaltet Millionen von Konti

Handel: entscheidet über Kauf und Verkauf

Verkehr: kontrolliert Züge, überwacht Flugzeuge…

- Einige Autos beinhaltet Software mit Millionen von Zeilen Programmcode

Reisen: Flug-, Zug-, und Hotelbuchungen

Kommunikation: Telefonie, Internet, …

Behörden: verwaltet Steuern, überwacht Gesetze…

Gesundheitswesen: verwaltet Krankenakten, überwacht Geräte

Unterricht

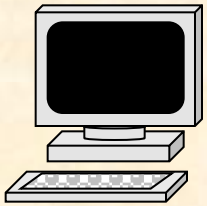
Unterhaltung

Information

usw.

Computer überall...

- Banken
- Flugzeuge, Autos...
- Waschmaschinen
- Smartphone
- Drucker
- Morgen: Ihr T-Shirt...



Computer in allen Grössen, Farben und Formen

http://www.go-gulf.com/wp-content/themes/go-gulf/blog/6oseconds.jpg

Computer

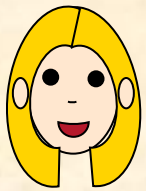
Computer sind universelle Maschinen. Sie führen das Programm aus, das Ihnen gegeben wird.

Ihre Vorstellungskraft ist die einzige Grenze

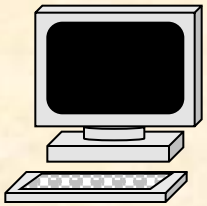
Die guten Nachrichten:

- Ihr Computer tut genau das, was in Ihrem Programm steht

Programme erstellen und ausführen



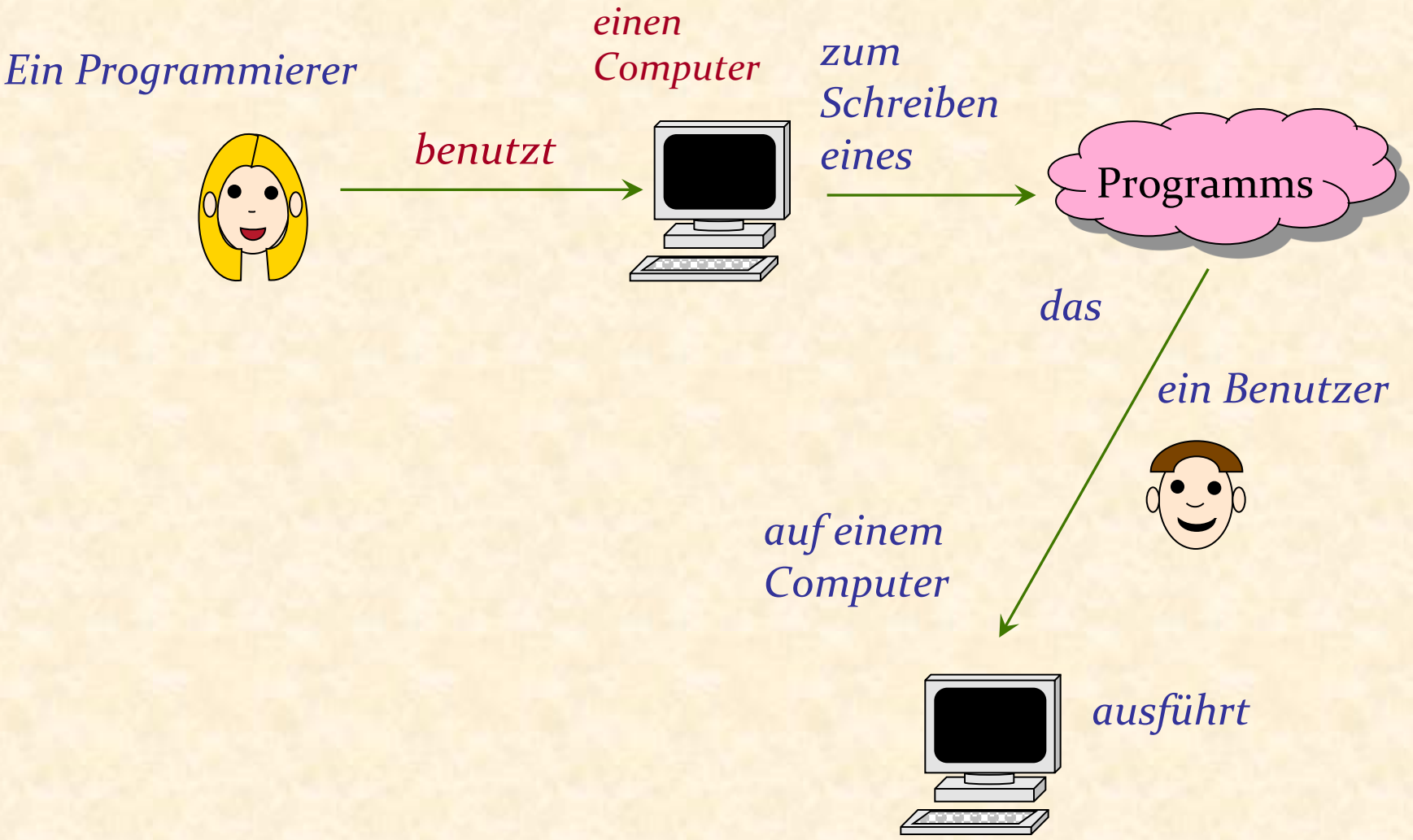
Ein Programmierer *schreibt ein* Programm *das* *ein Benutzer* *auf einem Computer* *ausführt*



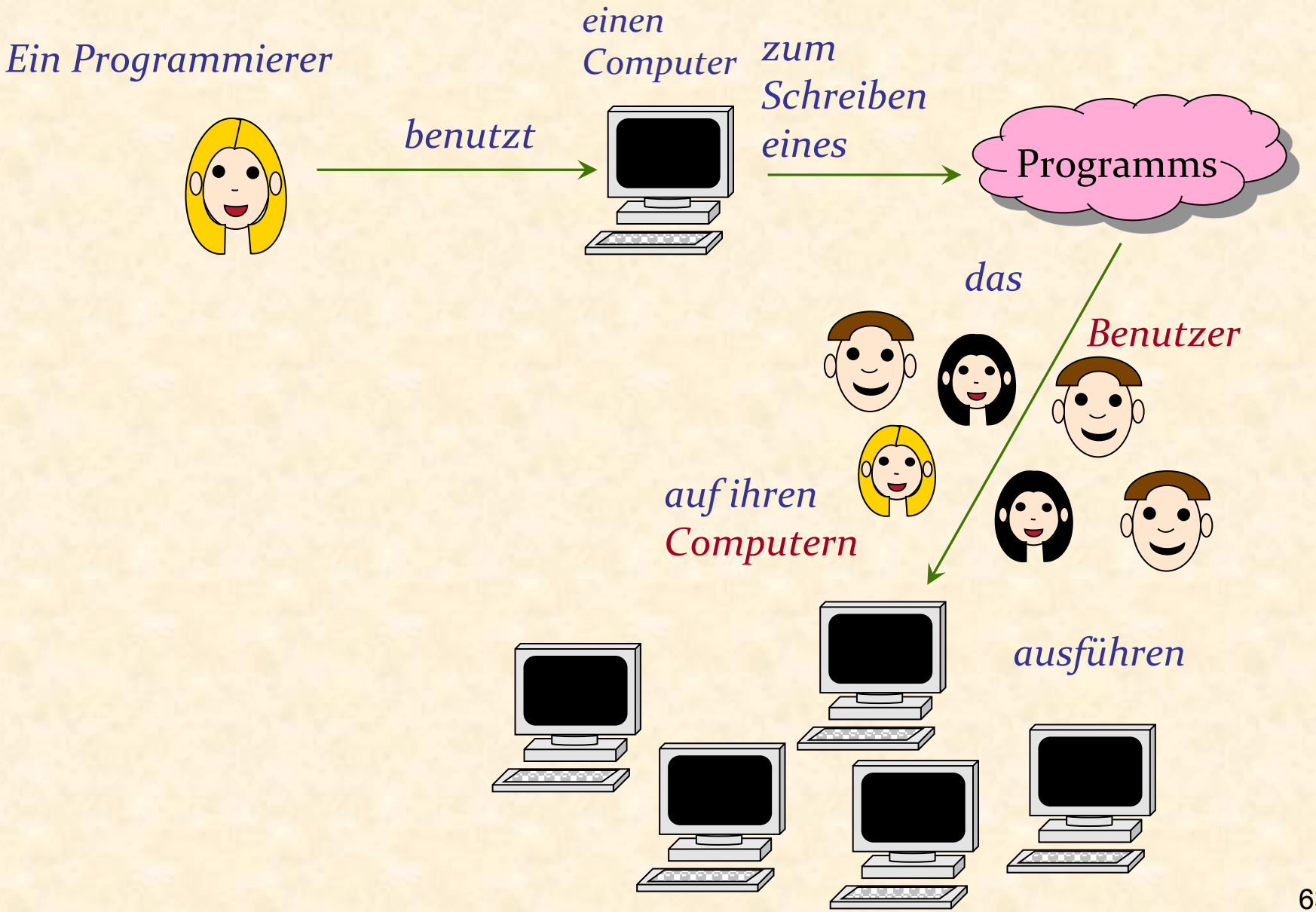
Programmierer: schreibt Programme

Benutzer: führt Programme aus

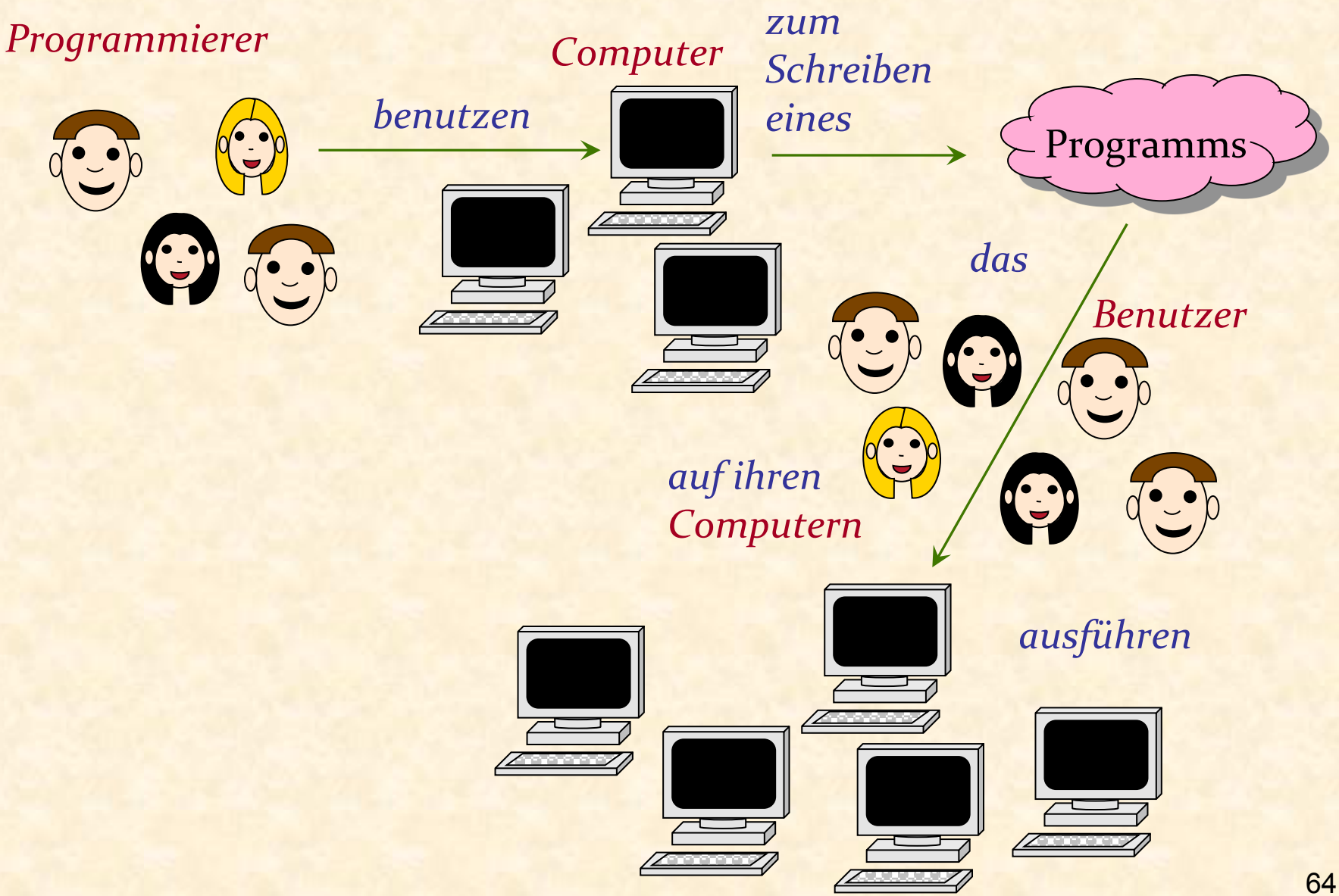
Programme erstellen und ausführen



Programme erstellen und ausführen



Programme erstellen und ausführen



Computer

Computers sind universelle Maschinen. Sie führen das Programm aus, das Ihnen gegeben wird.

Ihre Vorstellungskraft ist die einzige Grenze

Die guten Nachrichten:

- Ihr Computer tut genau das, was in Ihrem Programm steht
- Er tut es sehr schnell

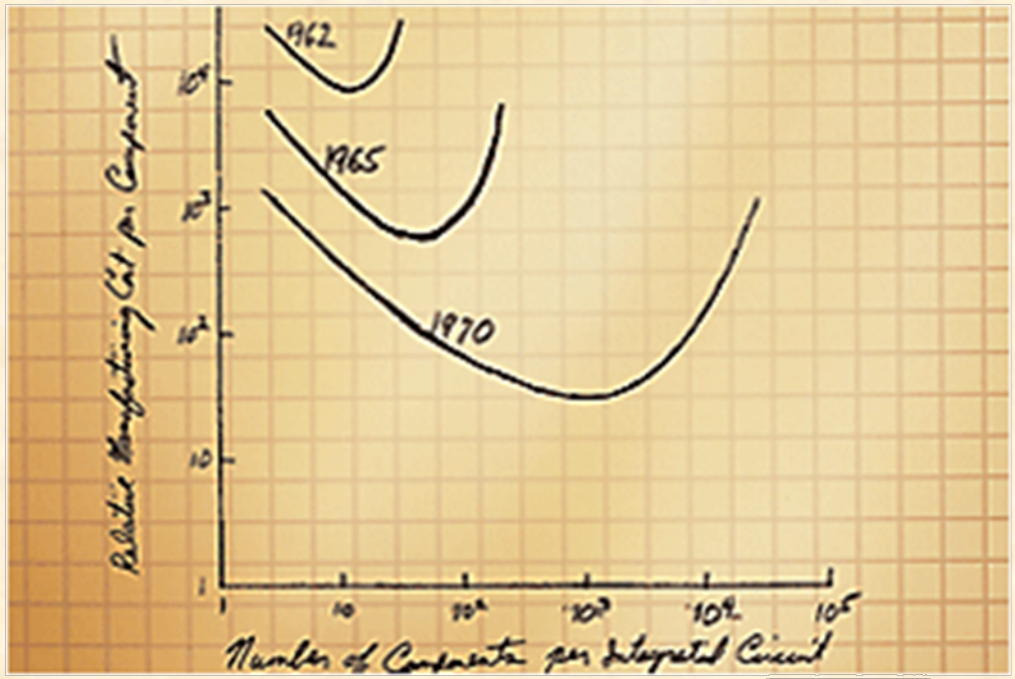
Moore's "Gesetz"

Etwa alle 18 Monate: Verdopplung der Rechenleistung bei gleichbleibendem Preis

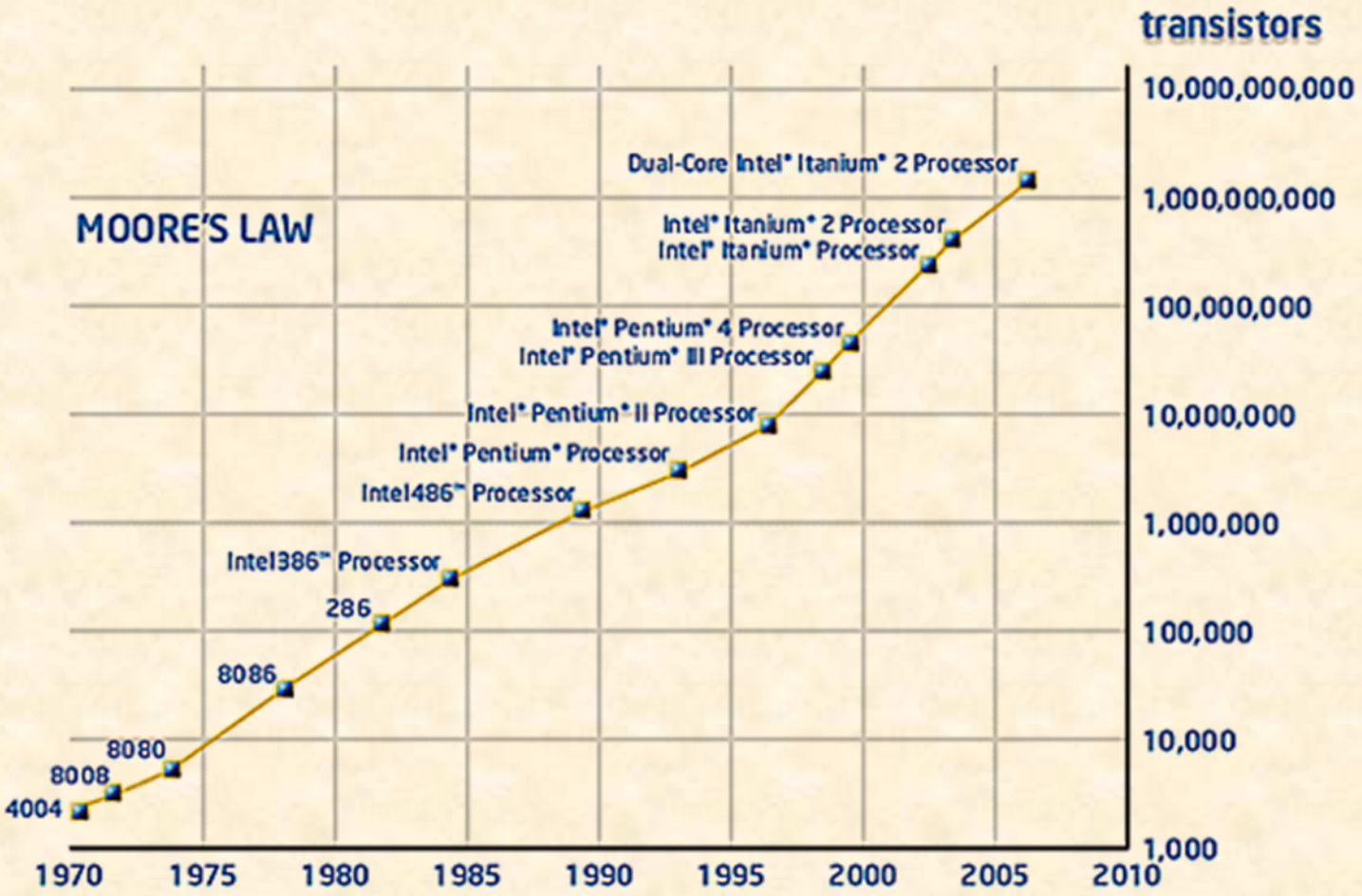
(Ist das die Aussage von Moore's Gesetz?)

(Nein: Verdopplung der Anzahl Transistoren)

Gordon Moore's original graph from 1965

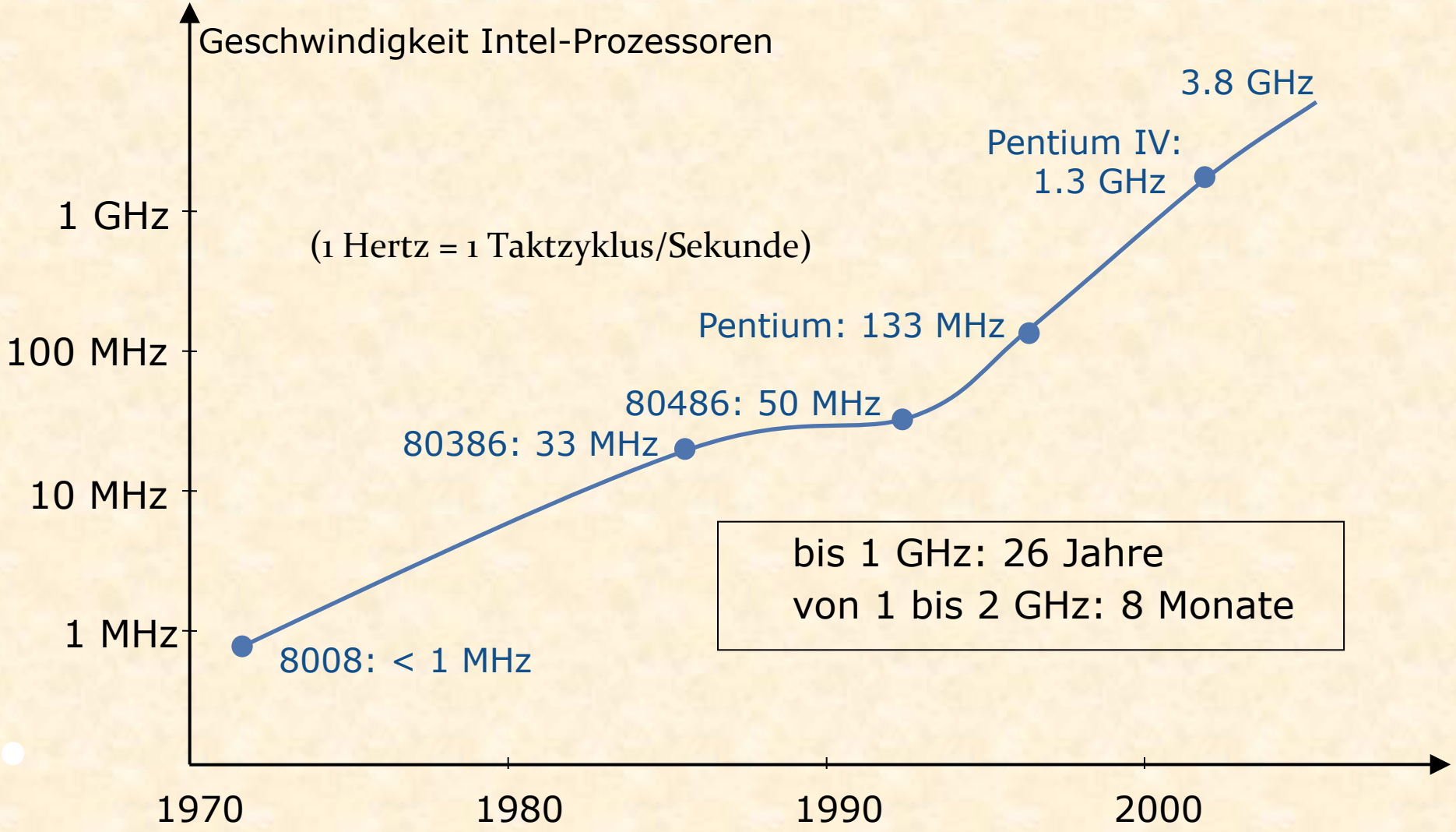


Moore's Gesetz (Quelle: Intel)

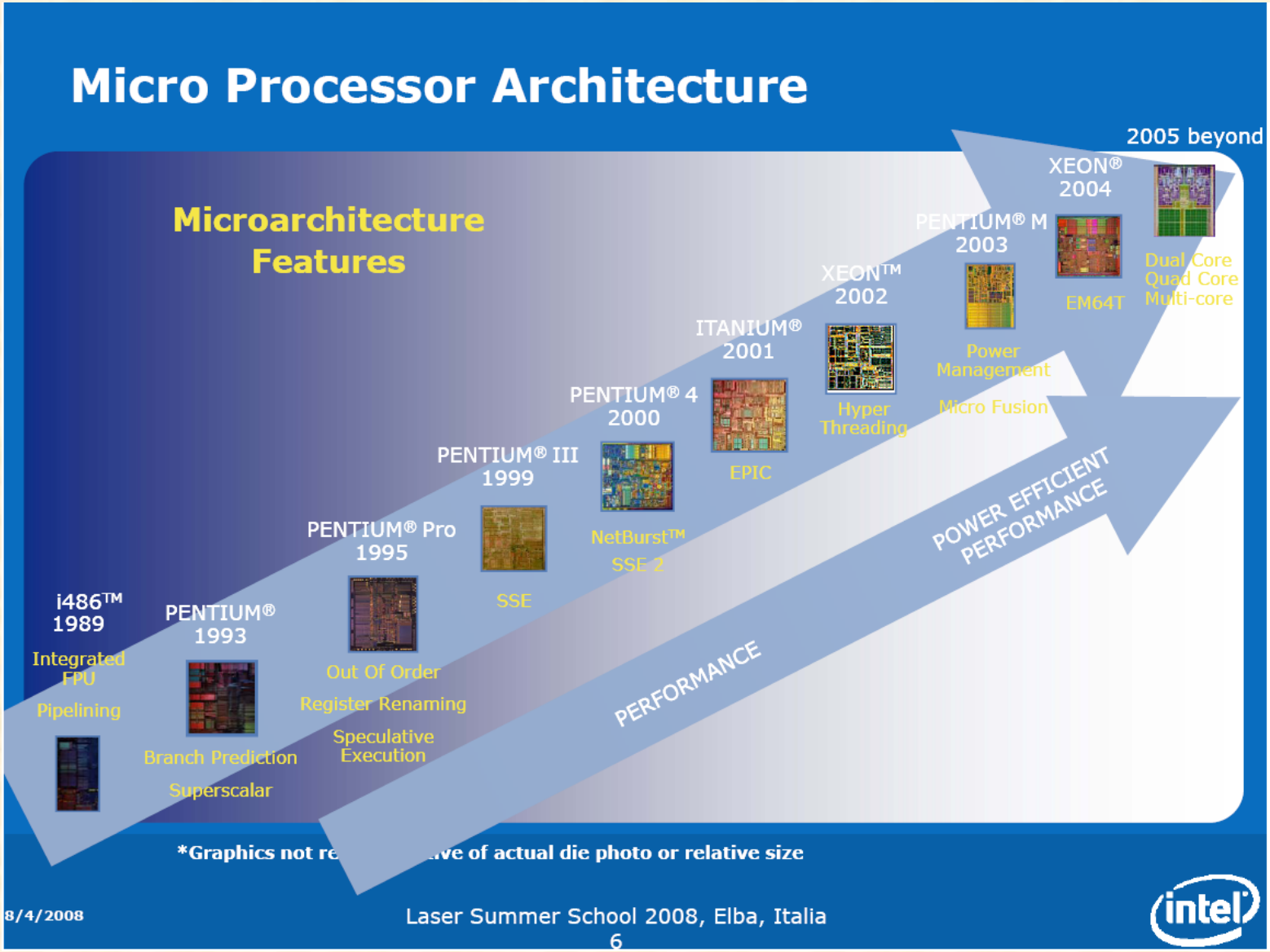


Moore's "Gesetz"

Etwa alle 18 Monate: Verdopplung der Rechenleistung bei gleichbleibendem Preis



Microprozessoren (Quelle: Intel)



Typical times*

1 nanosecond (ns) = 10^{-9} second, 1 millisecond (ms) = 10^{-3} second:

		If this were 1 second…
➢ Basic computer operation	1 ns	
➢ Main memory access	100 ns	Almost 2 minutes
➢ Disk access: one item	10 ms	4 months
1 MB	20 ms	8 months
➢ SSD access: 4 K	0.15 ms	2 days
1 MB	1 ms	Half a month
➢ Local packet send	0.5 ms	One week
➢ Packet send Europe → US	150 ms	Almost 5 years

*After Peter Norvig, for more see https://gist.github.com/jboner/2841832; thanks also to Erik Meijer

Legenden und Entschuldigungen

"Computer sind intelligent"

> *Tatsache: Computer sind weder intelligent noch dumm. Sie führen Programme aus, die von Menschen entwickelt wurden. Diese Programme spiegeln die Intelligenz ihrer Autoren wieder.*
>
> *Die Grundoperationen eines Computers sind elementar (speichern eines Wertes, addieren zweier Zahlen...).*

"Der Computer ist abgestürzt"

"Der Computer erlaubt das nicht"

"Der Computer hat Ihre Datei verloren"

"Der Computer hat Ihre Datei kaputt gemacht"

Computer machen keine Fehler *....

- Programme machen auch keine Fehler
- Programmierer machen die Fehler

*Hardware kann zwar defekt sein, aber das ist viel seltener der Fall als Fehler in der Software

Computer

Computer sind universelle Maschinen. Sie führen das Programm aus, das Ihnen gegeben wird.

Die einzige Grenze ist Ihre Vorstellungskraft und Ihre Sorgfalt

Die guten Nachrichten:

- Ihr Computer tut genau das, was in Ihrem Programm steht
- Er tut es sehr schnell

Die schlechten Nachrichten:

- Ihr Computer tut genau das, was in Ihrem Programm steht
- Er tut es sehr schnell

"To err is human, but to really mess things up takes a computer"

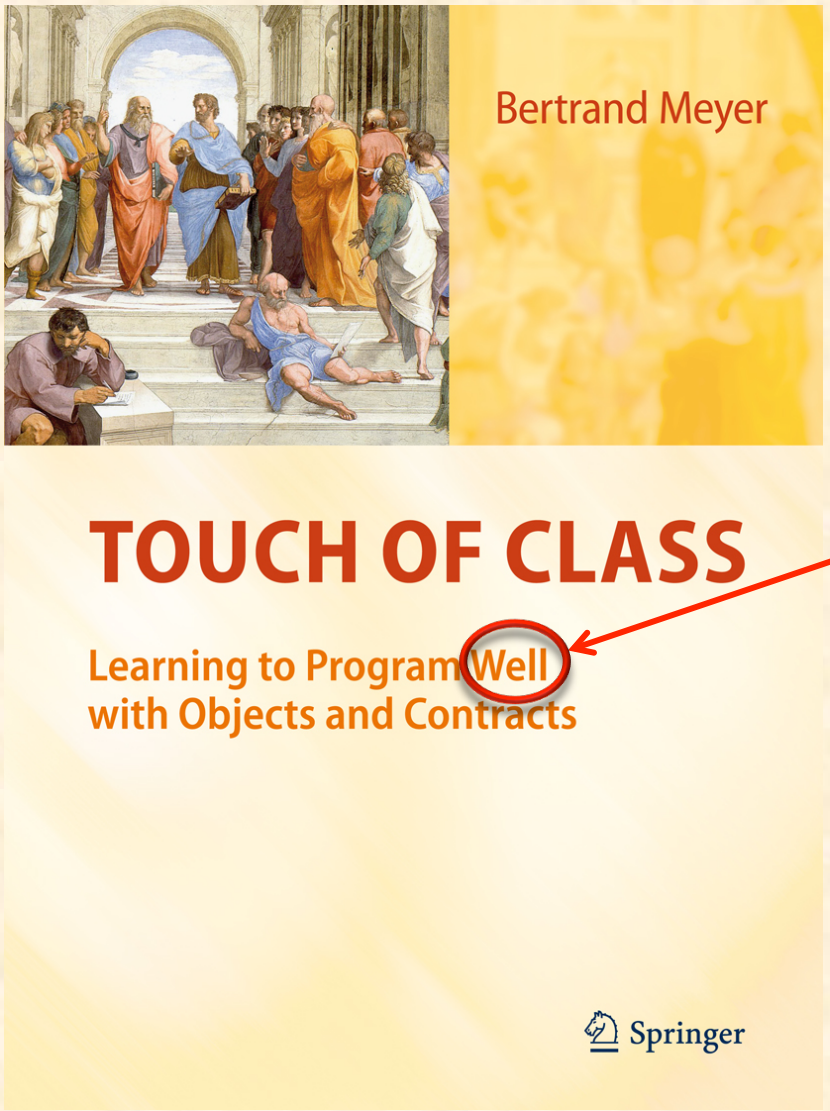
Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: HAL_INITIALIZATION_FAILED

Software zu schreiben ist schwierig

- Programme “stürzen ab”
- Wenn Programme nicht abstürzen, bedeutet das nicht unbedingt, dass sie richtig funktionieren
- Fehlerhafte Programme haben Menschen umgebracht, z.B. medizinische Geräte
- Ariane 5 Rakete, 1996: $10 Mia. verloren, wegen einem einfachen Programmfehler
- US Patriot-Raketen, 1990: 28 Soldaten getötet, wegen einem numerischen Fehler
- Programmierer sind für die korrekte Funktionsweise ihrer Programme verantwortlich
- Der Zweck dieser Vorlesung ist es nicht nur Ihnen programmieren beizubringen, sondern dass Sie gut programmieren lernen.

Learning to program *well*



Ariane 5

http://www.youtube.com/watch?v=kYUrqdUyEpI

Programm-Fehler: sehen

- Jean-Marc Jézéquel & Bertrand Meyer: *Design by Contract: The Lessons of Ariane*, in *Computer* (IEEE), vol. 30, no. 1, January 1997, pages 129-130, archive.eiffel.com/doc/manuals/technology/contract/ariane/.

Was Computer tun

Speichern und wieder abrufen

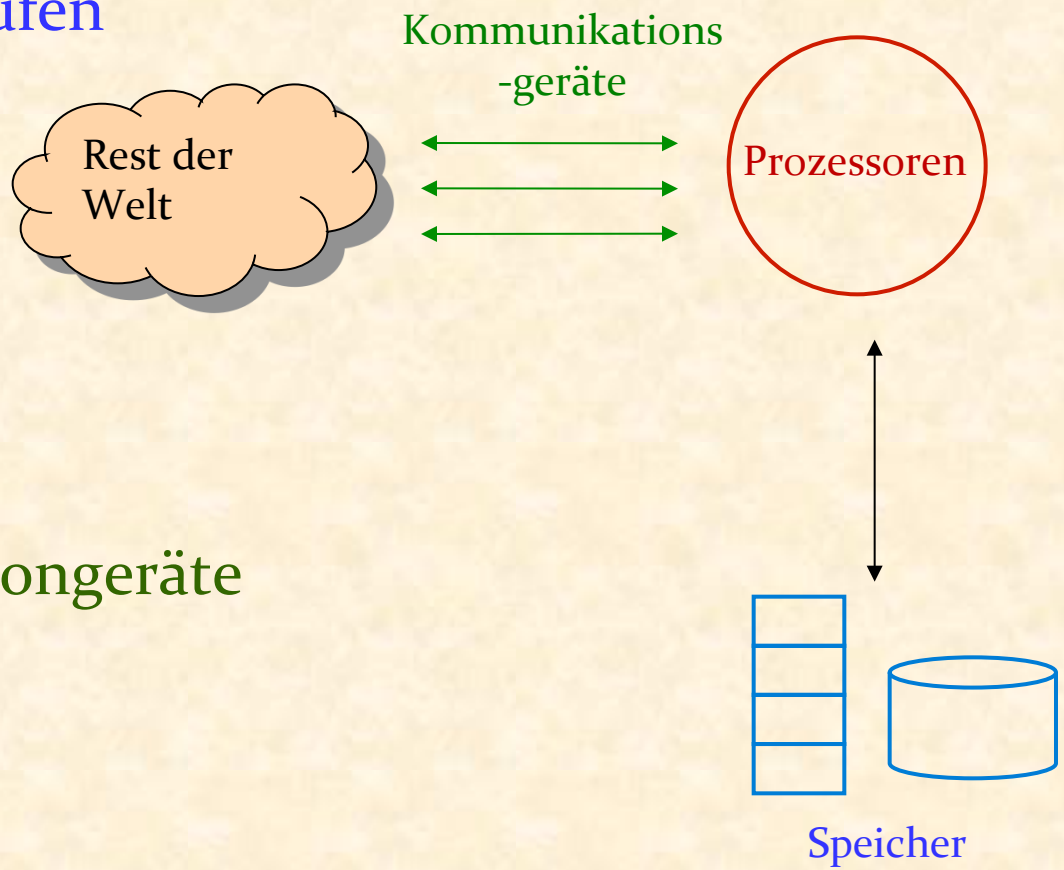
- Speicher

Operationen ausführen

- Prozessoren

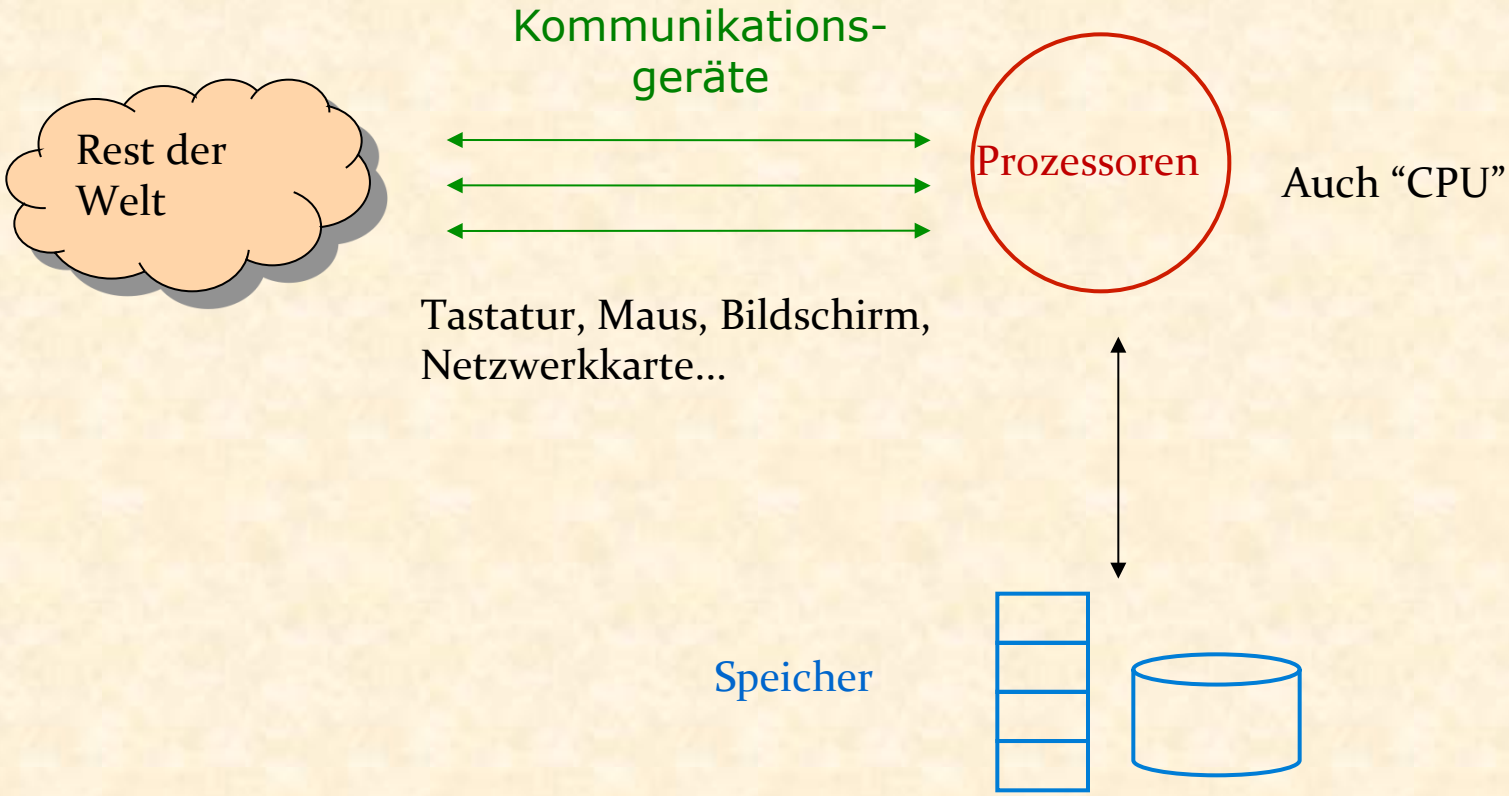
Kommunikation

- Kommunikationgeräte



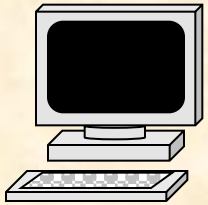
Speicher, Prozessoren und Kommunikationsgeräte gehören zur Hardware

Genereller Aufbau



Computer

Computer sind universelle Maschinen. Sie führen das Programm aus, das Ihnen gegeben wird.



(Universelle Maschine)

\+

Programm

Software

=

Spezialisierte Maschine

Information und Daten

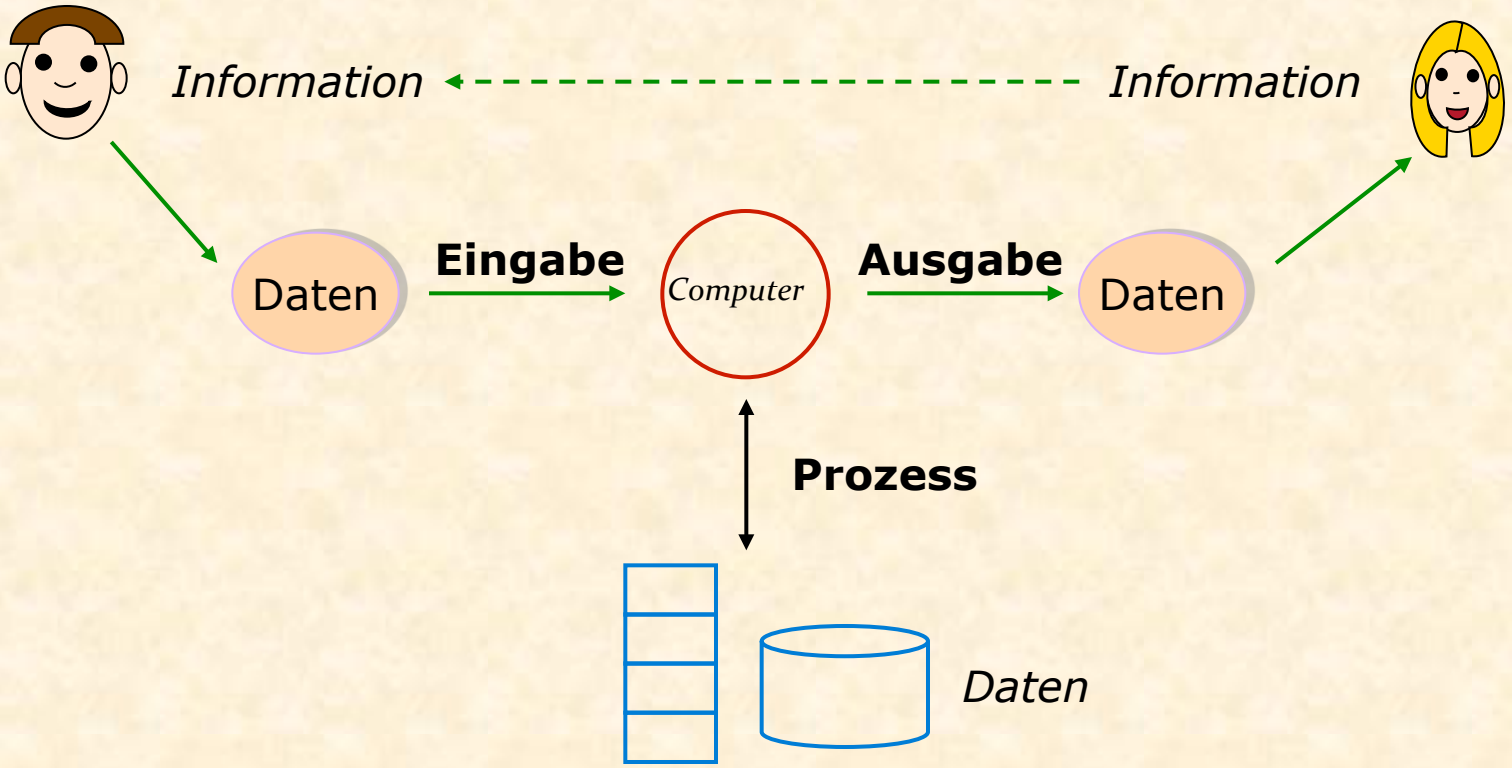
Information ist, was Sie wollen, z.B. Text oder Musik

Daten sind Informationen, die für den Computer kodiert sind, z.B. im MP3-Audioformat

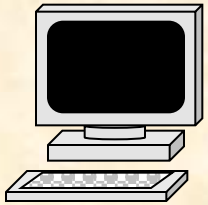
- Daten: Menge von Symbolen auf dem Computer gespeichert
- Informationen: Interpretation der Daten für menschliche Zwecke

Verarbeitung von Daten und Information

Daten werden im Speicher gehalten
Eingabegeräte produzieren Daten aus Informationen
Ausgabegeräte produzieren Informationen aus Daten



Wo ist das Programm?



(Universelle Maschine)

\+

Programm

=

Spezialisierte Maschine

Wo befindet sich ein Programm?

Speicherprogrammierte Computer: Das Programm befindet sich im Speicher

- "Executable data"

Ein Programm kann im Speicher in verschiedenen Formen vorhanden sein:

- Quellcode: von Menschen lesbar (Programmiersprache)
- Maschinencode: vom Computer ausführbar

Ein Compiler übersetzt Quellcode in Maschinencode

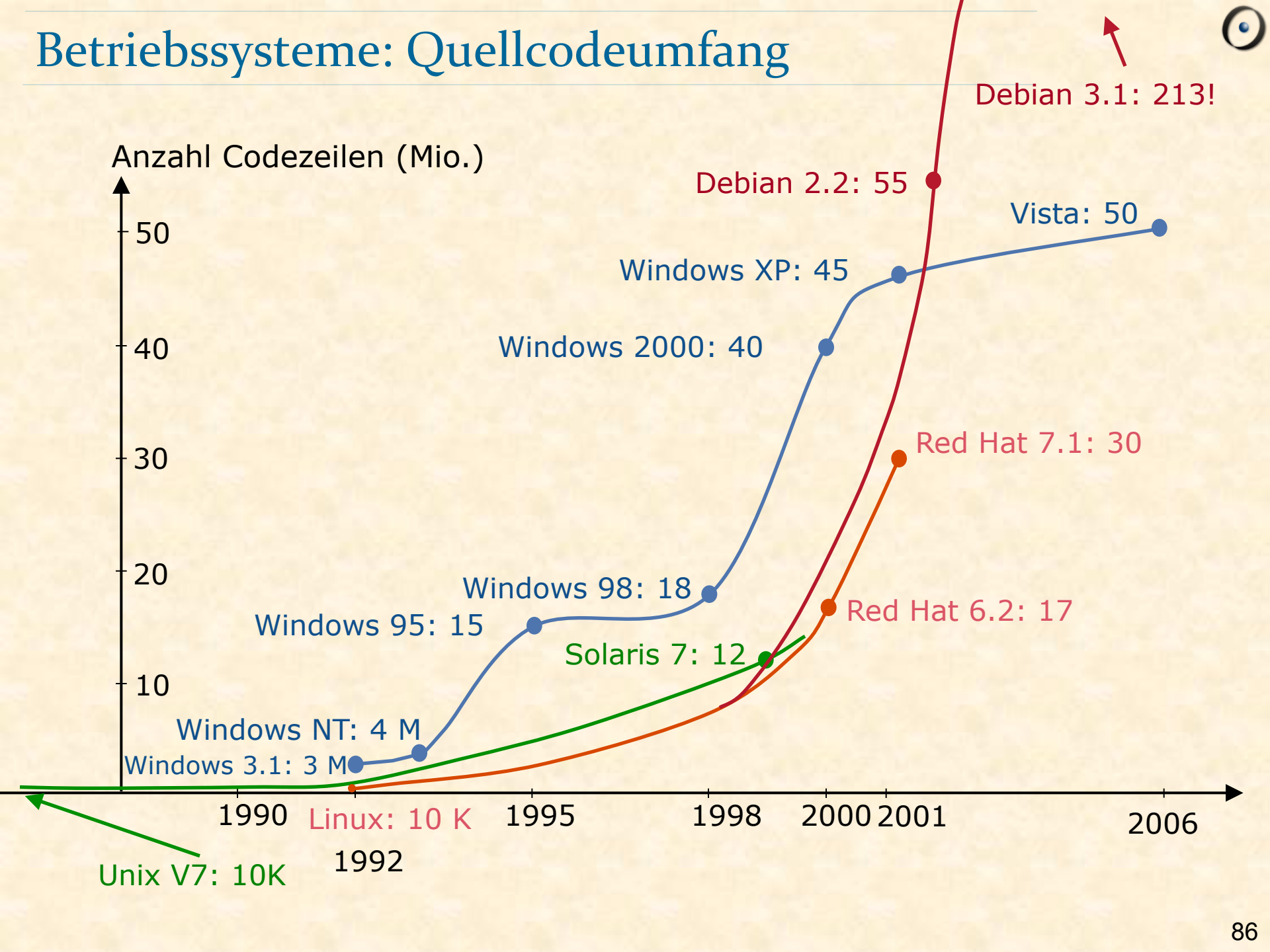
Der Computer
(genauer: die Plattform aus Computer + Betriebssystem)
findet Ihr Programm im Speicher und führt es aus

Software Engineering

Schreiben von Software, die folgende Eigenschaften besitzt:

- Korrekt
  Macht was sie soll!
- Erweiterbar
  Einfach zu ändern!
- Lesbar
  Von Menschen!
- Wiederverwendbar
  Das Rad nicht wiedererfinden!
- Robust
  Reagiert angemessen auf Fehler
- Sicher
  Gegen Angreifer

Betriebssysteme: Quellcodeumfang



Software zu schreiben ist schwierig

Es ist schwierig, ein korrektes Programm zu schreiben

“Trial-and-error” ist ineffizient

Software zu schreiben macht Spass

Entwerfen und entwickeln Sie Ihre eigenen Maschinen

Leben Sie Ihre Kreativität und Ihren Ideenreichtum aus

Programme retten Leben und helfen, die Welt zu einem besseren Ort zu machen

Erleben Sie das Gefühl, wenn ein Programm, das Sie geschrieben haben, funktioniert

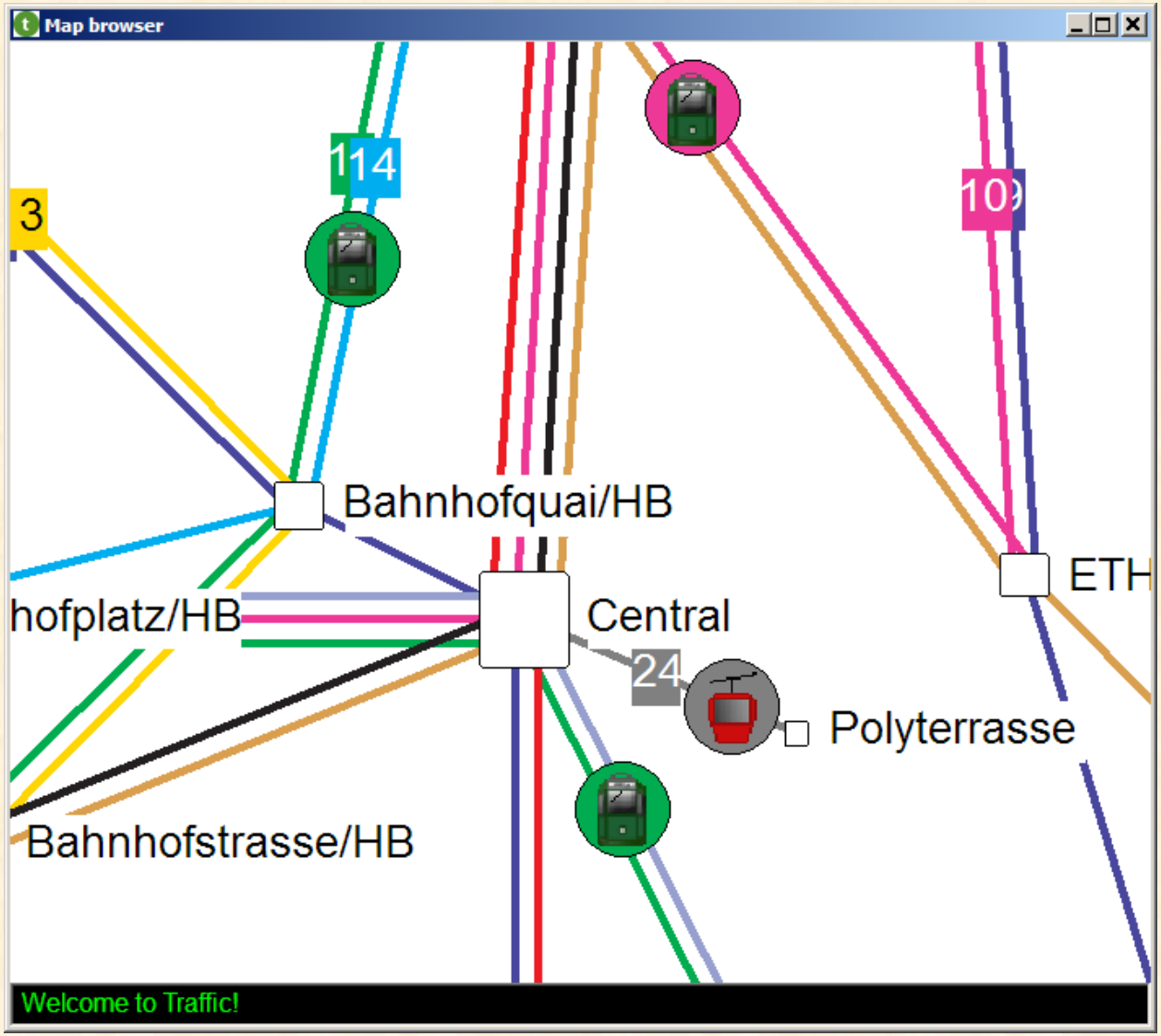
Die Software

Die Übungen bauen auf der Bibliothek Traffic auf

Anwendungsgebiet: öffentlicher Verkehr in einer Stadt (benutzt Zürich als Beispiel)

Übung 1 (auf der Webseite) führt Sie durch die Installation von EiffelStudio und zeigt Traffic

Entdecken Sie Traffic



Warnung

Natürlich ist nicht alles perfekt.

Traffic beinhaltet wahrscheinlich Fehler (“bugs”), und das Buch wahrscheinlich auch.

(Fehlerliste: http://touch.ethz.ch -> *Errata*)

Aber:

- Wir versuchen, die Fehler so schnell wie möglich zu korrigieren.
- Schieben Sie beim Ausprobieren die Schuld jeweils nicht zuerst der Software in die Schuhe. Vielleicht folgt sie bloss Ihren Anweisungen.

Weshalb diese Lehrmethode?

Mit anderen Lehrmethoden würden Sie nur mit kleinen selber geschriebenen Programmen arbeiten

Wir verwenden ein vorgegebenes Softwaresystem; benutzen Sie dieses als Beispiel und Inspiration

Sie benutzen die Software durch ihre abstrakten Schnittstellen (auch bekannt als **Verträge** (contracts))

Sie verwandeln sich vom Konsumenten zum Produzenten: outside-in

Traffic ist grafisch und macht Spass!

Im besten Fall verstehen Sie am Ende die gesamte Software.

Dann können Sie auch neues hinzufügen

Warum wir Eiffel benutzen

- Einfaches, reines Objektmodell
- Design by Contract Mechanismen
- Erlaubt Ihnen auf Begriffe, nicht die Sprache selbst, zu fokussieren
- Geringes «Sprachgepäck»
- Gute Entwicklungsumgebung (EiffelStudio)
- Portabilität: Windows / Linux / Mac
- Realismus: Eiffel ist keine «akademische» Sprache

Eiffel bereitet Sie darauf vor, andere O-O Sprachen wenn nötig zu studieren, z.B. C++, Java, C#

Diese Vorlesung lehrt nicht eine Programmiersprache, sondern die **Programmierung**

Warum nicht Java?

Erstes Java Programm:

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

Du wirst verstehen, wenn du aufwächst!

Tue wich ich sage, nicht wie ich tue!

Erstes Eiffel programm

```
class
        PREVIEW
inherit
        ZURICH_OBJECTS
 feature
        explore
                        -- Show city info and route.
                do
                        -- “To be filled in (by you!)”
                end
end
```

Warum nicht Java?

- Warum nicht C++?
- Warum nicht Ruby?
- Warum nicht Python?
- Warum nicht C#?
- Warum nicht C?
- Warum nicht Fortran?
- Warum nicht Cobol?
- ...

Einen Job finden

In den kommenden Jahren werden Sie alle wichtige Programmiersprachen lernen

Gute Arbeitgeber suchen nicht spezifische Kenntnisse, sondern starke Ingenieur- und Systemfähigkeit

Viele Leute können Programme schreiben

(tatsächlich: ca. 24 Millionen)

Arbeitsgeber suchen Leute, die **gute** Programme schreiben können

Bis zur nächsten Vorlesung

Lesen Sie Kapitel 1 und 2 von *Touch of Class*

Schauen Sie sich die Folien der nächsten zwei Vorlesungen (2 und 3) an