Einführung in die Programmierung
Introduction to Programming

Prof. Dr. Bertrand Meyer

Lecture 1: Welcome and overview
Sprache für diese Vorlesung

Das ist die erste deutsche Folie dieser Vorlesung

Wer singt und tanzt in dieser Video?

- 1. Paris Hilton?
- 2. Céline Dion?
- 3. Micheline Calmy-Rey?
- 4. Samuel Schmid?

Das war die letzte deutsche Folie dieser Vorlesung
Languages spoken by assistants:

- German (several varieties)
- English
- Italian
- Romanian
- Russian
- ...

Exercise sessions (Übungsgruppen) are available in German (5) and English (4).
Goals of the course

After successfully taking this course, you will:

- Know the key concepts of programming
- Be able to tackle many different programming problems, including in new areas
- Understand basic hardware and software tools
- Master a programming language: Eiffel
- Know basic concepts of design, implementation and maintenance of software systems ("software engineering").
Schedule

Lectures:
- Monday, 13:15 – 15:00, HG E3
- Tuesday, 8:15 – 10:00, HG F1

Exercise sessions:
- 8 groups
  - Monday, 15:15 – 16:00, various rooms
  - Tuesday, 13:15 – 15:00, various rooms
The assistant team

Marco Piccioni (head)
Conrado Plano
Andreas Leitner
Michela Pedroni (back-office)
Yi Wei (Jason)
Benjamin Morandi

Wolfgang Schwedler
Hermann Lehner (quality assurance)
Ilinca Ciupa
Ivo Colombo
Nadia Polikarpova
Exercise groups

We have made up the exercise groups

Your group is based on your last name:

**A-Bro:** Nygaard *(Ilinca)*

**Bru-Fan:** Goldberg *(Conrado)*

**Fav-Guz:** Knuth *(Marco)*

**Ham-Käs:** Torvalds *(Ivo)*

**Kat-Lon:** Dijkstra *(Benjamin)*

**Lor-Ngu:** Lovelace *(Wolfgang)*

**Noe-Sche:** Liskov *(Jason)*

**Schi-Tad:** Hoare *(Nadia)*

**Tan-Z:** Wirth *(Andreas)*

If you have a good reason to change, e.g. a language problem, ask Hermann Leitner
Kristen Nygaard group: Ilinca Ciupa

Mailing list: nygaard@se.inf.ethz.ch

E-mail: Ilinca.Ciupa@inf.ethz.ch
Office: RZ J4
Phone: 044 632 44 49
Language: English
Room:
- Monday: ML J37.1
- Tuesday: ETZ E9
Mailing list: goldberg@se.inf.ethz.ch

E-mail: planoc@student.ethz.ch
Language: German
Rooms:
- Monday: ML F39
- Tuesday: LEC C18
Donald Knuth group: Marco Piccioni

Mailing list: knuth@se.inf.ethz.ch

E-mail: Marco.Piccioni@inf.ethz.ch
Language: English
Office: RZ J5
Phone: 044 632 65 32
Rooms:
- Monday: HG E21
- Tuesday: CAB H52
Linus Torvalds group: Ivo Colombo

Mailing list: torvalds@se.inf.ethz.ch

E-mail: icolombo@student.ethz.ch
Language: German
Rooms:
- Monday: ML H37.1
- Tuesday: HG E22
Mailing list: dijkstra@se.inf.ethz.ch

E-mail: benjamin.morandi@inf.ethz.ch
Language: German
Office: RZ J3
Phone: 044 632 02 68
Rooms:
- Monday: ML H34.3
- Tuesday: LFW C5
Ado Lovelace group: Wolfgang Schwedler

Mailing list: lovelace@se.inf.ethz.ch

E-mail: wolfgang.schwedler@inf.ethz.ch
       adam.darvas@inf.ethz.ch

Language: German & English
Office: RZ F6
Phone: 044 632 85 39
Rooms:
  - Monday: IFW A32.1
  - Tuesday: ETZ E9
Mailing list: liskov@se.inf.ethz.ch

E-mail: yi.wei@inf.ethz.ch
Office: RZ J3
Phone: 044 632 44 70
Language: English
Rooms:
- Monday: ML J34.1
- Tuesday: ML J34.3
Tony Hoare group: Nadia Polikarpova

Mailing list: hoare@se.inf.ethz.ch

E-mail: Nadia.Polikarpova@inf.ethz.ch
Office: RZ J8
Phone:
Language: English
Rooms:
- Monday: LEC C14
- Tuesday: IFW B42
Mailing list: wirth@se.inf.ethz.ch

E-mail: Andreas.Leitner@inf.ethz.ch
Office: RZ J4
Phone: 044 632 30 21
Language: German
Rooms:
  - Monday: IFW A 36
  - Tuesday: LEC C14
Coordinating assistant

Michela Pedroni
The quality assurance assistant

Hermann Lehner

Hermann.Lehner@inf.ethz.ch

RZ F6
044-632-8539
About me

At ETH since end of 2001, Professor of Software Engineering
In industry most of my career, last with *Eiffel Software* in Santa
Barbara, California, in 1985. Now “Chief Architect”
Assoc. Prof. at University of California, Santa Barbara in 80s
Published a number of books, in particular *Object-Oriented Software Construction* (2nd edition: 1997)
Research interests: software engineering, methods, tools, programming
languages, object-oriented programming, concurrent programming,
program proofs, testing, development environments, persistence etc.

Contact:

- E-mail: Bertrand.Meyer@inf.ethz.ch
- Office: RZ J22
- Secretary: Claudia Günthart, 044 632 83 46
  Claudia.Guenthart@inf.ethz.ch
  Office: RZ J7
Office hours: Thursdays during the semester, ask Ms. Günthart
Your most important URL for the next 4 months

Course page:

http://se.ethz.ch/teaching/2008-F/eprog-0001/

→ Check it at least twice a week

English version available, but German more up to date

Lecture material:

- Lecture slides
- Textbook: *Touch of Class* (draft)
  Available electronically from course page

Also:

Exercise material:

- Exercise sheets
- Master solutions

Video recording of lectures!
TOUCH OF CLASS

Learning to program well
with objects and contracts

Bertrand Meyer
Electronic forums

Discussion forums:
Help forum for the whole course:
http://forum.vis.ethz.ch/

Mailing list for each group

Advice and rules:

- Use the forums and mailing lists! Learning to program is hard: take advantage of every help you can get.
- Don’t be shy. There are no stupid questions.
- Criticism welcome, but always be polite to every participant and observe the etiquette.

To email the whole teaching team (professor and assistants):

eprog-assi@se.inf.ethz.ch
If you need a laptop (actually you do)

ETH has good prices through the NEPTUN program

Thinkpad (Lenovo, ex-IBM), HP or Apple

You choose your OS: Windows, Linux

Limited time window: see www.neptun.ethz.ch
Exercises

The exercises are a key part of the course

- Ca. 10 weekly assignments
- Two “mock exams”

What you must do for each task:

- Show serious effort to address the questions
- Fill out questionnaire

Military services or illness ⇒ contact your assistant.
Grading

The basic rules are by the ETH, the details of the Testat are ours.

The grade comes entirely from the exam in September.

BUT: to be permitted to take the exam you must get a “Testat”. This means that you do:

- All the weekly assignments except at most one
- Both mock exams

What you must do for each task:
- Show serious effort to address the questions
- Fill out questionnaire

Military services or illness ⇒ contact your assistant.
The software

The exercises rely on the Traffic “library”

Application domain: Transportation system in a city
    (in the book: Paris
    in this course: Zurich)

You will need to download:

- EiffelStudio: http://eiffelsoftware.origo.ethz.ch/downloads/releases/6.2/
- Traffic: http://traffic.origo.ethz.ch/download

(this can wait until next week).
Discovering Traffic
Behind the software

Michela Pedroni (current Traffic project leader)

Numerous ETH students including:
  Marcel Kessler, Rolf Bruderer, Ursina Caluori, Roger Küng, Alan Fehr, Sarah Hauser, Michele Croci, Matthias Bühlmann, Florian Geldmacher, Susanne Kasper, Lars Krapf, Valentin Wüstholz, Stefan Daniel, Etienne Reichenbach, Maria Husmann
  ...

Patrick Schoenbach (initial version)
Warning

Not everything is perfect from our side. The Traffic software probably has mistakes (“bugs”), and the textbook does, too.

BUT:

- We will correct our mistakes, as quickly as we can.
- If you try something, don’t blame the software first. It may be doing just what you told it to.
Why this approach?

Many software issues become really tough for big systems. With other approaches, in an intro course, you only see small programs.

We give you lots of software; use it as model & inspiration. You learn to use software through abstract interfaces (also known as contracts). You go from consumer to producer: outside-in. Traffic is graphical and fun! You should at the end be able to understand all of it. Then you can add to it yourself.
# Background of 1st-year CS students (2003-2007)

<table>
<thead>
<tr>
<th>Computer experience</th>
<th>2007</th>
<th>Previous years</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 1 year</td>
<td>0%</td>
<td>(0%, 0%, 1%)</td>
</tr>
<tr>
<td>2 to 4 years</td>
<td>1%</td>
<td>(3%, 4%, 1%, 6%)</td>
</tr>
<tr>
<td>5 to 9 years</td>
<td>38%</td>
<td>(35%, 48%, 35%, 55%)</td>
</tr>
<tr>
<td>≥ 10 years</td>
<td>61%</td>
<td>(62%, 48%, 63%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programming experience</th>
<th>2007</th>
<th>Previous years</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>12%</td>
<td>(19%, 18%, 14%, 22%)</td>
</tr>
<tr>
<td>No O-O</td>
<td>20%</td>
<td>(26%, 33%, 38%)</td>
</tr>
<tr>
<td>≥ 100 classes</td>
<td>8%</td>
<td>(11%, 15%, 10%, 5%)</td>
</tr>
</tbody>
</table>
Topics

- What is software?
- Objects & programs
- Interfaces and the notion of class
- Logic and contracts
- The run-time model: object creation, references
- Describing syntax
- Control structures

- Inheritance
- Genericity
- Recursion
- Data structures
- Event-driven programming & agents
- Topological sort
- Intro to software engineering
Basic advice

- Attend all lectures
- Read material — textbook, slides — before lecture
  (Note: slides often updated after lecture)
- Bring a printout of the slides to the lecture, take notes
- Attend all exercise sessions
- Do all exercises & the project
  (you’ll need them for the “Testat”)
- If you don’t understand, ask
  (again: there are no stupid questions)
Previous programming experience

If you have already programmed, take advantage of it, but using a fresh look; explore Traffic

If you are new to programming, don’t be afraid; it can be hard at the beginning but you’ll get the hang of it.

Mathematics is as useful a preparation as programming experience
Some personal advice

Succeeding at university (and specifically ETH):

- You are in charge
- Take advantage of ETH possibilities
  - Talks by visiting scientists
  - Conferences
  - Library
  - Labs
  - Projects
- Talk to professors, assistants
- Read the Web pages of the department and the Chair of Software Engineering, browse around
- Look for courses with projects, other opportunities to do personal work
More advice

- Attend lectures
- Attend exercise sessions
- Read and print slides ahead of courses
- Take notes
- Don’t neglect non-CS courses, esp. first two years
- Don’t prepare for the exam at the last minute

- Keep a critical, probing attitude
The industry of pure ideas
Software engineers build machines

You can’t touch, kick or drop our machines: they’re immaterial
But they are machines anyway
We call them programs or systems

To operate (or run or execute) a program you need a physical machine: a computer

Computers and related devices: hardware

Programs and associated intellectual value: software
Software everywhere

Banking: manage millions of accounts
Trading: decide to sell or buy
Transportation: control trains, track planes...
  ➢ Some cars have millions of lines of program code
Travel: air, train, hotel reservations
Communication: phones, Internet, ...
Government: manage taxes, track laws...
Health care: keep health record, control devices
Education
Entertainment
Information
etc.
Computers are universal machines. They execute the program that you feed them.

The only limit is your imagination.

The good news:

- Your computer will do exactly what your program says.
Working with a computer

A programmer writes a program, which a user runs on a computer.
A programmer using a computer writes a Program which runs on a computer a user
A programmer using a computer writes a Program which users run on their computers.
Programmers write a program which run on their computers using a computer that is used by users.
Computers are universal machines. They execute the program that you feed them.

The only limit is your imagination.

The good news:

- Your computer will do exactly what your program says.
- It will do it very fast.
Moore’s “Law”

Approximate doubling of computing power, for comparable price, every eighteen months

(Is this what Moore’s law says?)

(No: approximate doubling of the number of transistors)
Moore’s law (source: Intel)
Moore’s “Law”

Approximate doubling of computer power, for comparable price, every eighteen months

Speed of Intel processors

(1 Hertz = 1 clock cycle per second)

- 8008: < 1 MHz
- 80386: 33 MHz
- 80486: 50 MHz
- Pentium: 133 MHz
- Pentium IV: 1.3 GHz
- 3.8 GHz

To 1 GHz: 26 years
From 1 to 2 GHz: 8 months
Microprocessors (source: Intel)

Micro Processor Architecture

Microarchitecture Features

- i486™ 1989
  - Integrated FPU
  - Pipelining

- PENTIUM® 1993
  - Branch Prediction
  - Superscalar

- PENTIUM® Pro 1995
  - Out Of Order
  - Register Renaming
  - Speculative Execution

- PENTIUM® III 1999
  - NetBurst™
  - SSE

- PENTIUM® 4 2000
  - Hyper-Threading
  - EPIC

- XEON™ 2002
  - ITANIUM®

- XEON® 2004
  - XEON® M 2003

- POWER EFFICIENT PERFORMANCE

- POWER PERFORMANCE

*Graphics not representative of actual die photo or relative size

Laser Summer School 2008, Elba, Italia
Common myths and excuses

“Computers are intelligent”

*Fact:* Computers are neither intelligent nor stupid. They execute programs devised by humans. These programs reflect the intelligence of their authors.

The basic computer operations are extremely elementary (store this value, add these two numbers...).

“The computer has crashed”

“The computer doesn’t allow this”

“The computer lost your record”

“The computer messed up your record”
Computers don’t make mistakes *....

- Programs don’t make mistakes either
- Programmers do make mistakes

*Actually, hardware can malfunction, but this is much more rare than program errors
Computers

Computers are universal machines. They execute the program that you feed them.

The only limit is your imagination and your carefulness.

The good news:

- Your computer will do exactly what your program says.
- It will do it very fast.

The bad news:

- Your computer will do exactly what your program says.
- It will do it very fast.

“To err is human, but to really mess things up takes a computer.”
Writing software is tough

Programs “crash”
Programs that don’t crash don’t necessarily work
Badly functioning programs have killed people, e.g. in medical devices
Ariane 5 rocket, 1996: $10 billion lost because of a simple program error

Programmers are responsible for the good functioning of their programs
The purpose of this course is to teach you not just programming but good programming
The “Blue Screen Of Death”

WARNING!

The system is either busy or has become unstable. You can wait and see if it becomes available again, or you can restart your computer.

* Press any key to return to Windows and wait.
* Press CTRL+ALT+DEL again to restart your computer. You will lose unsaved information in any programs that are running.

Press any key to continue _
TOUCH OF CLASS

Learning to program well with objects and contracts

Bertrand Meyer
What computers do

- Storage and retrieval
- Operations
- Communication

Memories, processors and communication devices are the hardware.
General organization

Rest of the world

Communication devices

Processors

Also “CPU”

Keyboard, mouse, video display, network connector...

"Core" memory, disks...

"Persistent" or not

Memories
Information is what you want, e.g. a text or music.

Data is how it is encoded for the computer, e.g. MP3 audio format.

- Data: collections of symbols held in a computer
- Information: interpretation of data for human purposes
Information and data processing

Data is stored in memory
Input devices produce data from information
Output devices produce information from data
Computers come in all sizes, colors, flavors
Computers everywhere

Banks
Airplanes, cars...
Washing machines
Cell phones (70% of value)
Printers
Tomorrow: your shirt...
Computers are universal machines. They execute the program that you feed them.

(Understanding this concept can significantly enhance your skills in computer science and technology.)
Where’s the program?

**Stored-program computer:** the program is in memory

“Executable data”

The computer

(more precisely the **platform:** computer + operating system)

finds your program in memory to execute it

A program can appear in memory in different forms:

- **Source:** human-readable form (programming language)
- **Target form, machine code, object form:** form executable by the computer

**Compilers** transform source text to machine code
Writing software that’s

- **Correct**
  
  Does what it’s supposed to!

- **Extendible**
  
  Easy to change!

- **Readable**
  
  by humans!

- **Reusable**
  
  Don’t reinvent the wheel!

- **Robust**
  
  React appropriately to errors

- **Secure**
  
  Defeat attackers
Operating systems: source size

Lines of code (millions)

- **Unix V7**: 10K
- **Linux**: 10K
- **Windows 3.1**: 3 M
- **Windows NT**: 4 M
- **Windows 95**: 15
- **Windows 98**: 18
- **Windows 2000**: 40
- **Windows XP**: 45
- **Windows 2000**: 40
- **Red Hat 7.1**: 30
- **Red Hat 6.2**: 17
- **Solaris 7**: 12
- **Debian 2.2**: 55
- **Debian 3.1**: 213!
- **Vista**: 50
- **Debian 2.2**: 55
- **Debian 3.1**: 213!

In other application areas

(source: Siemens)
Writing software is tough

It is difficult to get a program right

Trial-and-error approach very inefficient
Writing software is fun

Design and build your own machines

Exert your creativity and imagination

Programs save lives and help make the world better

Experience the feeling of a program that you wrote, and that works
For next week

Read chapters 1 and 2 of *Touch of Class*

Read slides for next two lectures (2 and 3)