Einführung in die Programmierung
Introduction to Programming

Prof. Dr. Bertrand Meyer

Lecture 1: Welcome & overview
Sprache für diese Vorlesung

Das ist die erste deutsche Folie dieser Vorlesung

Das war die letzte deutsche Folie dieser Vorlesung
Language

Languages spoken by assistants:

- German (several varieties)
- English
- Italian
- Chinese
- Russian
- French
- Bulgarian
- ...

Exercise sessions (Übungsgruppen) are available in German (7) and English (3).
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 in diverse 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:
- 10 groups
  - Monday, 15:15 - 17:00, various rooms
  - Tuesday, 13:15 - 15:00, various rooms
Exercise groups

Information was sent to you by email. You choose between

- Three levels
- Two times
- Two languages

We assign you to a group.

If you have a good reason to change, e.g. a language problem: ask Nadia
The assistant team

Nadia Polikarpova (head)
Michela Pedroni
Scott West
Marco Trudel
Julian Tschanne
Benjamin Morandi

Nikolay Kazmin
Florian Köhl
Alexandre Lagadec
Jeremie Miserez
Yi Wei (Jason) (quality assurance)
Marco Piccioni (back office)
Coordinating assistant

Nadia Polikarpova
Quality assurance assistant

Yi (Jason) Wei
Back-office assistant

Marco (Paco) Piccioni
Kristen Nygaard group: Michela Pedroni

Mailing list: se-info1-nygaard@lists.inf.ethz.ch

E-mail: michela.pedroni@inf.ethz.ch
Office: RZ J8
Phone: 044 632 76 84
Language: German
Room:
  ➢ Tuesday: IFW A34
Adele Goldberg group: Alexandre Lagadec

Mailing list: se-info1-goldberg@lists.inf.ethz.ch

E-mail: lagadeca@student.ethz.ch
Language: German
Rooms:
- Tuesday: ETZ K91
Donald Knuth group: Florian Köhl

Mailing list: se-info1-knuth@lists.inf.ethz.ch

E-mail: koehlf@student.ethz.ch
Language: German
Rooms:
  ➢ Tuesday: CAB H52
Linus Torvalds group: Scott West

Mailing list: se-info1-torvalds@lists.inf.ethz.ch

E-mail: scott.west@inf.ethz.ch
Language: English
Office: RZ J8
Phone: 044 632 74 09
Rooms:
  ➢ Tuesday: LFW C5
Edsger Dijkstra group: Benjamin Morandi

Mailing list: se-info1-dijkstra@lists.inf.ethz.ch

E-mail: benjamin.morandi@inf.ethz.ch
Language: German
Office: RZ J9
Phone: 044 632 78 28
Rooms:
  - Monday: ML H41.1
Ada Lovelace group: Marco (Taco) Trudel

Mailing list: se-info1-lovelace@lists.inf.ethz.ch

E-mail: marco.trudel@inf.ethz.ch
Language: German
Office: RZ J9
Phone: 044 632 85 92
Rooms:
  ➢ Monday: LFV E41
Barbara Liskov group: Nikolay Kazmin

Mailing list: se-info1-liskov@lists.inf.ethz.ch

E-mail: nkazmin@student.ethz.ch
Language: English
Rooms:
  ➢ Tuesday: ETZ E9
Mailing list: se-info1-hoare@lists.inf.ethz.ch

E-mail: Nadia.Polikarpova@inf.ethz.ch
Office: RZ J8
Phone: 044 632 65 74
Language: English
Rooms:
  ➢ Tuesday: IFW B42
Mailing list: se-info1-wirth@lists.inf.ethz.ch

E-mail: julian.tschannen@inf.ethz.ch
Office: RZ J3
Phone: 044 632 44 49
Language: German
Rooms:
  ➢ Tuesday: CAB H53
Mailing list: se-info1-turing@lists.inf.ethz.ch

E-mail: jmiserez@student.ethz.ch
Language: German
Rooms:
  ➢ Tuesday: ETZ F91
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
Course page:

http://se.ethz.ch/teaching/2009-H/eprog-0001/

→ Check it at least twice a week

German version available, but English more up to date

Lecture material:
- Lecture slides
- Textbook: Touch of Class
  See next

Exercise material:
- Exercise sheets
- Master solutions

Also:

Video recording of lectures!
The textbook

TOUCH OF CLASS

Learning to Program Well with Objects and Contracts

Bertrand Meyer
Purchasing the textbook: tomorrow (or later)

Meyer
Touch of Class
Fr. 58.00

Büchertisch vor dem Hörsaal
Nur Barbezahlung möglich!

In der Buchhandlung akzeptieren wir Bargeld, Maestro, Postcard, Visa und Eurocard/Mastercard

Polybuchhandlung
ETH Zentrum MM B96
Öffnungszeiten: Mo-Do 9.30 – 16.30, Fr. 9.30-15.30
Internet: www.books.ethz.ch, Email: shop@books.ethz.ch

ETH-Zentrum
MM B 96
8092 Zürich

Öffnungszeiten
Montag – Donnerstag
9:30 – 16:30 Uhr
Freitag
9:30 – 15:30 Uhr
Accessing the textbook online

Possible for free from an IP address (access point) at ETH
URL: see course page
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.
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, MacOS

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

Repetenten: last year’s Testat valid, but you are advised to take the assignments anyway.
The software

The exercises rely on the Traffic "library"

Application domain: Transportation system in a city
(Paris is used as an example)

You will need to download:

- EiffelStudio: http://tinyurl.com/EiffelStudio6-4
- 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 (2003-2008)

<table>
<thead>
<tr>
<th>Computer experience</th>
<th>2008</th>
<th>Previous years</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 2 year: 1%</td>
<td></td>
<td>(0%, 0%, 0%, 1%)</td>
</tr>
<tr>
<td>2 to 4 years: 3%</td>
<td></td>
<td>(1%, 3%, 4%, 1%, 6%)</td>
</tr>
<tr>
<td>5 to 9 years: 34%</td>
<td></td>
<td>(39%, 35%, 48%, 35%)</td>
</tr>
<tr>
<td>≥ 10 years: 62%</td>
<td></td>
<td>(61%, 62%, 48%, 63%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programming experience</th>
<th>2008</th>
<th>Previous years</th>
</tr>
</thead>
<tbody>
<tr>
<td>None: 18%</td>
<td></td>
<td>(12%, 19%, 18%, 14%)</td>
</tr>
<tr>
<td>No O-O: 22%</td>
<td></td>
<td>(20%, 26%, 33%, 38%)</td>
</tr>
<tr>
<td>≥ 100 classes: 17%</td>
<td></td>
<td>(8%, 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
  (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 personal advice

- Attend lectures
- Attend exercise sessions
- Read and print slides ahead of courses
- Take notes
- Form study groups
- 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

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.
Building and running programs

A programmer writes a Program which a user runs on a computer.

Programmer: writes programs
User: runs programs
Building and running programs

A programmer using a computer writes a Program which runs on a computer...
Building and running programs

A programmer

using

a computer

writes a Program

which

users

run on

their computers
Building and running programs

Programmers using computers write a Program which users run on their computers.
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

(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)
“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."
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 →
Writing software is tough

Programs “crash”
Programs that don’t crash don’t necessarily work
Incorrect 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 correct functioning of their programs
The purpose of this course is to teach you not just programming but good programming
Learning to program **well**

**TOUCH OF CLASS**

Learning to Program Well with Objects and Contracts

Bertrand Meyer

Springer
Computers everywhere

Banks
Airplanes, cars...
Washing machines
Cell phones (70% of value)
Printers
Tomorrow: your shirt...
Computers come in all sizes, colors, flavors
What computers do

Storage and retrieval
- Memories

Operations
- Processors

Communication
- Communication devices

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

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
Where’s the program?

Stored-program computer: the program is in memory
- “Executable data”

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

A compiler transforms source text to machine code

The computer (more precisely the platform: computer + operating system) finds your program in memory to execute it
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
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)