Einführung in die Programmierung
Introduction to Programming
Prof. Dr. Bertrand Meyer

Lecture 1: Introduction & Overview

Englische Folien
Folien für diese und alle weiteren Vorlesungseinheiten werden in Deutsch und Englisch verfügbar sein.
Sie können sie auf der Webseite der Vorlesung finden.

Language
Das war die erste deutsche Folie dieser Vorlesung
Das ist die letzte deutsche Folie dieser Vorlesung
Choose your language

Languages spoken by assistants:
- German (several varieties)
- English
- Italian

Exercise sessions (Übungsgruppen) are available in German (6) and English (2).
- If there are enough requests, we will turn one of the English groups into an Italian group.

Goals of the course

After successfully taking this course, you will:
- Know the key concepts of programming
- 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 E7
- 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
Choosing an exercise group

Registration lists are available during the break. Choose your group according to

- Preferred language
- Availability

We may have to reassign students to a different group to keep the numbers balanced.

The first exercise session takes place this afternoon!

The official assistant team of the 2008 Olympics

Beat Herlig
Ilinca Ciupa (head)
Conrado Plano
Andreas Leitner
Michela Pedroni

Hermann Lehner
& Adam Darvas
Yann Müller
Marco Piccioni
Volkan Arslan

Kristen Nygaard group: Ilinca Ciupa

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

E-mail: Ilinca.Ciupa@inf.ethz.ch
Office: RZ 2.34
Phone: 044 632 44 49
Language: English
Room:
- Monday: IFW A32.1
- Tuesday: ETZ K91
Adele Goldberg group: Conrado Plano

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

E-mail: planoc@student.ethz.ch
Language: German
Rooms:
- Monday: ML F39
- Tuesday: LEC C1B

Donald Knuth group: Marco Piccioni

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

E-mail: Marco.Piccioni@inf.ethz.ch
Language: English
Office: RZ 75
Phone: 044 632 65 32
Rooms:
- Monday: CHN G42
- Tuesday: CAB H52

Linus Torvalds group: Beat Herlig

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

E-mail: bherlig@student.ethz.ch
Language: German
Rooms:
- Monday: ML H37.1
- Tuesday: LFW C5
Intro to Programming, lecture 1: Overview

Niklaus Wirth group: Andreas Leitner

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

About me

At ETH since end of 2001
In industry most of my career
Founded Eiffel Software in Santa Barbara, California, in 1985. Now "Chief Architect"
Published a number of books, in particular Object-Oriented Software Construction (2nd edition: 1997)
Plan: help the industry build the best software possible

Contact:
- E-mail: Bertrand.Meyer@inf.ethz.ch
- Office: RZ 322
- Secretary: Claudia Günthart, 044 632 83 46
- Claudia.Guenthart@inf.ethz.ch
- Office: RZ 37
- Office hours: Thursdays during the semester, ask Ms. Günthart
Course page:
http://se.inf.ethz.ch/teaching/2007-F/eprog-0001/

English version available, but German more up to date

Lecture material:
> Lecture slides
> Textbook: Touch of Class (draft)
  Available electronically from course page

Also: Video recording of lectures!

Exercise material:
> Exercise sheets
> Master solutions

Video recording

We are looking for student volunteers to record the lectures

The ETH e-learning service will provide free training

If interested, talk to Ilinca Ciupa

The textbook in progress

TOUCH OF CLASS

Learning to program well
with Object Technology
and Design by Contract

AN INTRODUCTION TO SOFTWARE ENGINEERING

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@cs.inf.ethz.ch

If you need a laptop...

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 and project

The exercise and project are a key part of the course
- Five or six weekly assignments
- Two "classroom exercises" (like mini-exams)
- A programming project in the last 5 weeks of the semester

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

(Subject to minor adaptations, see Web page for final information)

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 classroom exercises
- The project

What you must do for each task:

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

Military services or illness \(\Rightarrow\) contact your assistant.

The project

Extended programming exercise

Extension to Traffic (see next), very open-ended

To be turned in end of Fall semester

Public presentation in the second week of the Summer semester (date to be announced)

"Object-Oriental Bazaar"

The software

The exercises rely on the Traffic "library"

Application domain: Transportation system in a city

(\(\text{in the book: Paris}\)

(\(\text{in this course: Zurich}\))

You will need to download:

- EiffelStudio:
  
  http://eiffelsoftware.origo.ethz.ch/downloads/releases/6.0/

- 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,

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.

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

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.
User: runs programs.
Computers

Programmers

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 computer power, for comparable price, every eighteen months

- 1970: < 1 MHz
- 1980: 1 MHz
- 1990: 10 MHz
- 2000: 1 GHz

(1 Hertz = 1 clock cycle per second)

- 8086: 5 MHz
- 80386: 33 MHz
- Pentium: 133 MHz
- Pentium IV: 1.3 GHz

- To 1 GHz: 26 years
- From 1 to 2 GHz: 8 months

---

**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 the menu and wait.

Press [CTRL+ALT+DEL] again to restart your computer. You will lose unsaved information in any program that was running.

Press any key to continue.
Learning to program well

TOUCH OF CLASS

Learning to program well with classes, objects and contracts

Bernard Meyer

What computers do

- Storage and retrieval
- Operations
- Communication

Storage and retrieval \(\equiv\) memories
Operations \(\equiv\) processors
Communication \(\equiv\) communication devices

Memories, processors and communication devices are the hardware.

General organization

Communication devices

Rest of the world

Keyboards, mice, video display, network connector...

Also "CPU"

"Core" memory, disks...

"Persistent" or not
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.

Computers come in all sizes, colors, flavors
Computers everywhere

*Computers everywhere*

- Banks
- Airplanes, cars...
- Washing machines
- Cell phones (70% of value)
- Printers
- Tomorrow: your shirt...

Computers

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

\[ \text{Universal machine} + \text{Program} = \text{Specialized machine} \]

Where’s the program?

*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
Software Engineering

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

Operating systems: source size

<table>
<thead>
<tr>
<th>Year</th>
<th>OS</th>
<th>Source Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>Unix V7:</td>
<td>10K</td>
</tr>
<tr>
<td>1992</td>
<td>Red Hat 7.1:</td>
<td>17</td>
</tr>
<tr>
<td>1995</td>
<td>Solaris 7:</td>
<td>12</td>
</tr>
<tr>
<td>1998</td>
<td>Windows 95:</td>
<td>15</td>
</tr>
<tr>
<td>2000</td>
<td>Windows NT 4:</td>
<td>4 M</td>
</tr>
<tr>
<td>2001</td>
<td>Windows 98:</td>
<td>18</td>
</tr>
<tr>
<td>2000</td>
<td>Windows 2000:</td>
<td>40</td>
</tr>
<tr>
<td>2002</td>
<td>Windows XP:</td>
<td>45</td>
</tr>
<tr>
<td>2001</td>
<td>Red Hat 6.2:</td>
<td>17</td>
</tr>
<tr>
<td>2003</td>
<td>Red Hat 7.1:</td>
<td>30</td>
</tr>
<tr>
<td>2005</td>
<td>Vista:</td>
<td>50</td>
</tr>
<tr>
<td>2006</td>
<td>Debian 3.1:</td>
<td>213</td>
</tr>
<tr>
<td>2006</td>
<td>Debian 2.2:</td>
<td>55</td>
</tr>
<tr>
<td>2006</td>
<td>Debian 2.1:</td>
<td>39</td>
</tr>
<tr>
<td>2006</td>
<td>Debian 3.1:</td>
<td>213</td>
</tr>
</tbody>
</table>

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)