Einführung in die Programmierung
Introduction to Programming
Prof. Dr. Bertrand Meyer
October 2006 - February 2007
Lecture 1: Introduction & Overview

Englische & Deutsche Folien
Folien für diese und alle weiteren Vorlesungseinheiten werden in Deutsch und in Englisch verfügbar sein.
Sie können beide Varianten 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
  - French
  - Italian

Exercise sessions (Übungsgruppen) are available in German (5) and English (3).
  - If there are enough requests, we will turn one of the English groups into an Italian group.

Choose an exercise session (i.e. an assistant) in your preferred language.

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 large software systems ("software engineering").

Teaching staff & Practical information
About me

- At ETH for five years
- 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 details

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: normally Thursdays during the semester, contact Ms. Günthart

Schedule

Lectures:
- Monday, 13:15 - 15:00, HG E7
- Tuesday, 8:15 - 10:00, HG F1

Exercise sessions:
- 7 groups
  - Monday, 15:15 - 16:00, various rooms
  - Tuesday, 13:15 - 15:00, various rooms
- 1 group
  - Tuesday, 13:15 - 14:00, IFW B42
  - Thursday, 15:15 - 17:00, HG 626.3
Choosing an exercise group

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

- Preferred language
- Your schedule constraints
- 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 assistants

Ilinca Ciupa  Beat Herlig
Philipp Kraehenbuehl  Hermann Lehner
Andreas Leitner  Piotr Nienaltowski
Michela Pedroni  Marco Piccioni
Bernd Schoeller

Hermann Lehner

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

E-mail: hermann.lehner@inf.ethz.ch
Language: German
Office: RZ F6
Phone: 044 632 85 39
Rooms:
- Monday: ML J37.1
- Tuesday: ETZ E9
Philipp Krähenbühl

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

E-mail: pk@student.ethz.ch
Language: German
Rooms:
- Monday: ML F39
- Tuesday: LEC C18

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: IFW A32.1
- Tuesday: ETZ K91

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:
- Tuesday: IFW B42
- Thursday: HG 626.3
Your most important URL for the next 4 months

Course page:

se.inf.ethz.ch/teaching/ws2006/0001/index.html

→ 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!

The textbook in progress

TOUCH OF CLASS

Learning to program well
with Java Technology
and Design by Contract

AN INTRODUCTION TO SOFTWARE ENGINEERING

Herbert Myers
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 list! 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...

ETH has good prices through the NEPTUN program
Thinkpad (Lenovo, ex-IBM) or Apple
You choose your OS: Windows, Linux, MacOS
Limited time window: see www.neptun.ethz.ch

Exercises, “Testat” regulations

*Testat*: Needed for admittance to fall exam.
Tasks:
  > 7 weekly assignments
  > 2 classroom exercises
  > 1 project
To get the “Testat” you must have done at least:
  > 5 weekly assignments
  > 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: 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: eiffelsoftware.origo.ethz.ch/downloads/builds/
  > Traffic: traffic.origo.ethz.ch/download.html
(this can wait until next week).

Discovering Traffic

Behind the software

  > Patrick Schoenbach (initial version)
  > Michela Pedroni (current Traffic project leader)
  > Till Bay (EiffelMedia library)

Numerous ETH students including:
  Marcel Kessler, Rolf Bruderer, Ursina Caluori,
  Roger Kung, Alan Fehr, Sarah Hauser, Michele Croci,
  Matthias Bühmann, Florian Geldmacher, Susanne
  Kasper, Lars Kropf, Valentin Wüstholz, Stefan Daniel,
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

Tootor (Marie-Helene Ng Cheong Vee)

(Univ. of London)

Tool: helps you solve exercises, gives feedback

Data collection: help us improve the teaching.
Intro. to Programming, lecture 1: Overview

### 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
- Toplogical sort
- Intro to software engineering

### Basic advice

- Attend all lectures
- Read material — textbook, slides — before lecture (Note: slides often updated after lecture)
- 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)
- If you have already programmed, take advantage of it, but take a fresh look at the material; explore Traffic
- If you are new to programming, don't panic, it can be tough at the beginning

### 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 department's Web page, 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

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.

Programmer: writes programs
User: runs programs
Working with a computer

A programmer

writing

a program

which runs on

a computer

Working with a computer

A programmer

writing

a program

which users

run on

their computers

Working with a computer

Programmers

writing

a program

which users

run on

their computers
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 computer power, for comparable price, every eighteen months

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MHz</td>
<td>&lt; 1 MHz</td>
<td>8086: 1 MHz</td>
<td>80486: 50 MHz</td>
<td>Pentium: 133 MHz</td>
</tr>
<tr>
<td>10 MHz</td>
<td>8086: 33 MHz</td>
<td>Pentium: 1.3 GHz</td>
<td>3.8 GHz</td>
<td>1.3 GHz</td>
</tr>
<tr>
<td>100 MHz</td>
<td>to 1 GHz: 26 years</td>
<td>from 1 to 2 GHz: 8 months</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 GHz</td>
<td>3.8 GHz</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 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”

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+DELETE again to restart your computer. You will lose unsaved information in any programs that are running.

Press any key to continue...

Learning to program well

TOUCH OF CLASS

Learning to program well!
with Object Technology,
Design by Contract,
and steps to Software Engineering

Bevray & Meyer

What computers do

- Storage and retrieval
  - Operations
  - Communication

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"

Memories

"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

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.

(Human universal machine)
Where’s the program?

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

The computer, with the help of some basic programs (operating system) knows how to find 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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In other application areas

(source: Siemens)

Writing software is tough

Programs crash, etc.

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 chapter 1 of *Touch of Class*

Recommended: start reading chapter 2 and slides for next lecture

End of lecture 1