Introductory Programming
Introduction

Anne Haxthausen[a]

IMM, DTU

Welcome!

1. Practical information for courses 02100 and 02199
2. Introduction to programming and to Java
   (a) course goal and motivation
   (b) what is a computer, a program and a programming language
   (c) example of a Java program
   (d) how to compile and execute Java programs
   (e) demo of a Java program having a graphical user interface

a. Parts of this material are inspired by/originate from a course at ITU developed by Niels Hallenberg and Peter Sestoft on the basis of a course at KVL developed by Morten Larsen and Peter Sestoft.

Course staff

Teachers:

- Anne Haxthausen: course coordinator + 02100+02199 lecturer in autumn
- Paul Fischer: lecturer 3 days in January
- Jørgen Steensgaard-Madsen: responsible for mandatory exercise
- Hans Henrik Løvengreen: responsible for project exercises
- Jens Thyge Kristensen: 02115 lecturer in autumn

Many student assistants.

Course secretary:

- Annette Bendtsen

About me (Anne Haxthausen)

Is associate professor at CSE (Computer Science and Engineering) section,
at the DTU department IMM (Informatics and Matematical Modelling).

- research area: language design and mathematical methods for software development
- civilingeniør from DTU 1985
- lic. techn. in computer science, DTU 1989
- employed at Dansk Datamatik Center and CRI 1988-1994
  (co-designer of RAISE, a software development method)
- guest researcher at a Japanese research lab 1993
- assistant professor at DTU (ID and IT) 1994-1997
- associate professor at DTU (IT and IMM) 1997-

Practical information: course activities

1. Part 1, in 13 week semester:
   - Lectures.
   - Exercise sessions.
   - Mandatory assignment at the end of the period.
2. Part 2, in 3 week period :
   - Lectures and exercises (in graphical user interfaces).
   - Project work.
   - Mandatory project report.

Remaining practical info concerns part 1 of the course.

Practical information: lectures

- Tuesdays, 9 - 10 (incl. break) in building 116, aud. 81.
- Lecture plan can be found on the WWW home page.
- Please, ask questions during the lectures!

Practical information: exercise sessions

- Tuesdays, 10 - 12.
- First exercise session takes place Tuesday, September 10th.
- Problems for an exercise session can be found on the homepage one week before the session.
- Teaching assistants will be available to help you.
- You will then be divided into classes of 25-35 students.
- The class division and class rooms will be announced on campusnet 6/9 afternoon.

Practical information: evaluation

- x = score given for mandatory assignment at the end of the 13 week period
- y = score given for mandatory report at the end of the 3 week period
- final score = 00, if x = 00 or y = 00
- final score **approx.** = $35\% * x + 65\% * y$

Practical information: teaching material etc.

Home page: http://www.imm.dtu.dk/courses/02100
contains practical information, teaching material, ...
Inspect the page each week

Text book: Lewis & Loftus: Java Software Solutions, **third** edition
can be bought at Polyteknisk Boghandel in bld. 101

Lecture notes:

- Peter Sestoft: Systematic Software Test
- ...

will be made available at the homepage.

Overheads: will be available at the homepage the day before the lecture

Exercises: will be available at the homepage one week before the exercise session

Solutions to selected exercises: will be available at the homepage sometime after the exercise session

Installations for your own PC

The cd of the book contains Java2-SDK-1.4 for Windows.

We offer:

- a webpage for Windows and Linux Users, and
- a course cd for Windows Users

from where you can download some tools for installation:

- Tools for Java: Java2-SDK-1.4, BlueJ-1.2.1, Netbeans-3.3.2
- Other useful tools: editors, word processing systems, Adobe Acrobat Reader

You can order the course cd by paying 10 kr here or in IMM's notesalg as soon as possible and **not later** than Tuesday 10/9.

IMMs notesalg

IMMs notesalg is in building 305. Opening hours are:

1. week: 9-14,
2. week: 12-14,
3. - 13. week: Mondays and Tuesdays 12-13

Advices on the weekly work

The course has reputation for being demanding.

The expected work to be made by you each week:

Lectures: up to 1 hour

Exercises: 2 hours

Homework: *at least* 5 hours

- before lecture: read the relevant sections in the book
- after lectures: look over the overheads and sum up for your-self what you have learnt. If there is something that you do not understand, ask your teaching assistant or me
- before exercises: read the problems carefully and try to solve them
- after exercises: complete your solutions and compare them with the solutions that will be put on the homepage; ask your teaching assistant, if something is not clear to you

Practical information: contact info

If you have questions of *administrative* kind, please send them to the *course secretary* (c02100@imm.dtu.dk or c02199@imm.dtu.dk).

If you have questions concerning *programming*, please address them to your *teaching assistant* or post them on *campusnet*.

If you have questions concerning *tools installation*, please address them to your *teaching assistant* or *Lars Munch Christensen* (lmc@imm.dtu.dk) who made the cd.

E-mail adresses of teaching assistants can be found on the course WWW homepage.

Course goal

- to give you knowledge about basic concepts for imperative and object-oriented programming languages, e.g.
  - input and output
  - data types and values
  - expressions and statements
  - objects, classes, methods and inheritance
- to give you skills in solving and documenting smaller programming tasks.
- to have fun!

We use the programming language Java to illustrate this.

In other words: this is not just a course in Java, but in general computer science concepts.

Motivation

- Why learn to program ?
  - Because it is a part of the toolbox of an engineer
  - Because it is only in that way that you can really understand the possibilities and limitations in using computers
  - Because you can better envisage new types of solutions and products
  - Because it is tough to be able to
  - *Know Java? It Could Help Your Salary: Salaries for IT professionals, especially those with Java training, continue to increase . . .*: in average $78750 in north east USA (ACM TechNews, August 18, 2000).
- Why Java?
  - Java is well suited for general programming (like C, Pascal, C++, Basic, ML, . . .)
  - Java is well suited for windows-oriented programming of user interfaces
  - Java is well suited for consumer programs: internet trading, home banking, interactiv graphics, chat, distributed games, . . .

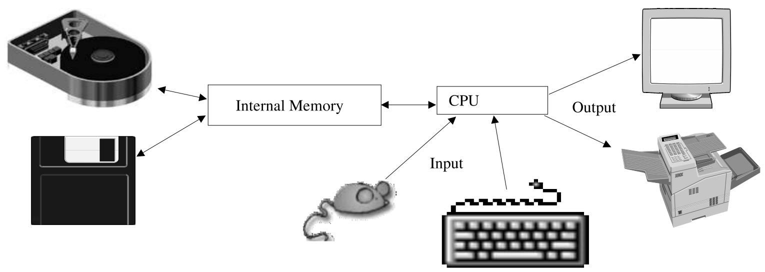
What is a computer?

- hardware: hard disks, floppy disks, keyboard, screen, printers, cpu, . . .
- software:
  - operating system: Windows 98, Windows 2000, Windows NT, Unix, PalmOS, Linux, . . .
  - applications: Microsoft Word, Star Office, Netscape, JavaEdit, . . .

Examples of computers

- PC
- Mainframe (e.g. in Danske Bank, Told-Skat)
- Server (e.g. web server, file server, mail server)
- Hearing Aid
- Palm Pilot
- Mobil phone

Simplified computer



What is a program?

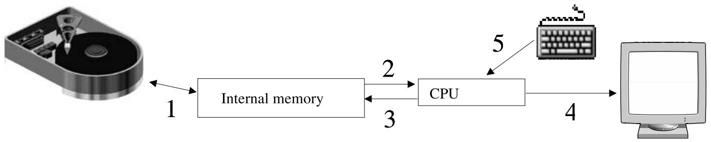
A program consists of internal instructions to a computer.

Internal instructions may imply external, visible results.
E.g. calculation and showing a balance, painting of a graph, . . .

Computers *only* do, what you ask them to do

... but they can do a lot of instructions quickly.

How is a program executed on a machine?



1. Program is read from the harddisk to the internal memory.
2. The first/next instruction in the program is stored in the CPU. The CPU execute the instruction.
3. The CPU stores maybe the result of a calculation in the internal memory.
4. The CPU outputs maybe the result of a calculation on the screen.
5. The CPU receives maybe input from the user.

What is a programming language?

A programming language is a notation for instructions to the computer, i.e. for programs.

Programming languages have

- a **syntax**: rules for, what are the allowed sentences of the language
- a **semantics**: defines the meaning of the sentences

One should be very careful when using programming languages.

Example:

- correct: `System.out.print("Hej");`
- wrong: `System.out.Print("Hej");`

A simple Java program

```
public class Hello {

  public static void main(String[] args) {
    System.out.println("This works!");
  }
}
```

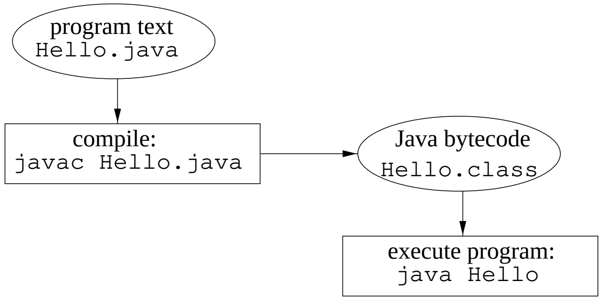
The name of the program (class) is `Hello`; the file name should be `Hello.java`.

The program text in the method `main` is executed when the program is started.

The method call `System.out.println` writes to the screen.

The meaning of `public static void` and `String[] args` will be explained later in the course.

How to compile and execute a Java program



1. *Edit* a Java program and store it in the file, e.g. as here `Hello.java`
2. *Compile* the program to Java bytecode, using the `javac` compiler; e.g.
   `javac Hello.java`
   This results in a new file `Hello.class` containing the compiled program.
3. *Execute* the compiled program, using the `java` interpreter; e.g. `java Hello`
   This results in the text `This works!` on the screen.

Kinds of errors

Three typical kinds of errors:

1. Compile-time errors
2. Runtime errors (dansk: Kørselsfejl)
3. Logical errors

Ad 1: For example syntax errors (like missing semicolon), unknown name of variable, wrong types. Are found by the compiler.

Ad 2: For example division by zero. Are found by the runtime system.

Ad 3: When you have asked the program to do something else than what you wanted it to do. Are found by the user when getting wrong output.

Remember this week

1. To fill out the questionaire and return it here after the lecture
2. To order a cd now or in IMM notesalg (if you want one) not later than 10/9
3. To buy the text book in the book store
4. To read in the text book: “Feature Walkthrough” and section (1.0–1.2 and) 1.3–1.4 and 2.0–2.8