

*Compilation 2011*

# **What Have We Learned?**

Jan Midtgaard  
Michael I. Schwartzbach  
Aarhus University

# What You Have Had the Chance to Learn

---

- Concrete skills:
  - compiler technology
  - insights into programming language design
- Technical skills:
  - all about Java
  - a new programming language: OCaml
  - building and testing software on a larger scale
  - reading huge and complex specifications
- Organizational skills:
  - handling stress and deadlines
  - finding relevant help and information
  - surviving as a group

# Compiler Technology

---

- Compiler architecture
- Typical compiler phases
- Scanners and parsers
- Scope resolvers
- Type checkers
- Static analyzers
- Code templates
  
- **You can build your own compiler**

# Programming Language Design

---

- The interplay between languages and compilers
- Features that are difficult to implement
- Limitations of current technology
- Scope rules
- Type rules
- Consequences of undecidability
  
- **You can discuss language design intelligently**
- **You can design another language**

# All About Java

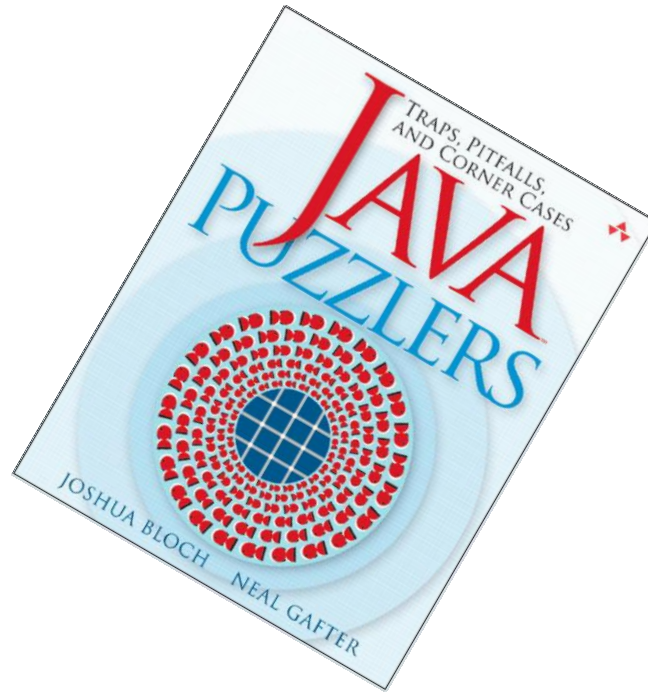
---

- The subtleties of the Java specification
- Name resolution
- Hierarchy checks
- Type checking
- Static analysis
- JVM and verification
- Code templates
  
- **You are a Java expert, not merely a user**

# All About Java

---

- The subtleties of the Java specification
  - Name resolution
  - Hierarchy checks
  - Type checking
  - Static analysis
  - JVM and verification
  - Code templates
- 
- **You are a Java expert, not merely a user**



# A New Programming Language: OCaml

---

New and well-known concepts in a new wrapping:

- Funny (unusual) syntax
- Algebraic datatypes
- Pattern matching
- Recursion
- Higher-order functions
- References vs. immutability
- Modules and interfaces
  
- **You have a more nuanced view on (statically typed) programming languages**
- **You can pick up a new programming language**

# Building and Testing Software

---

- 7,200 lines of code handed in
  - ~109K lines for 15 groups
  - Code sharing and versioning (`svn`, `cvcs`, ...)
  - Unit testing
  - Auto-generated code
  - Debugging
- 
- **You can handle a complex piece of software**

# Reading Complex Specifications

---

- The Java Language Specification (25,000 lines)
- The JVM Specification (24,000 lines)
- The ocaml yacc specification (too few lines)
  
- The Joos Languages
- 6 complex and subtle hand-in specifications
  
- **You can read huge and complex specs**

# Handling Stress and Deadlines

---

- You have met 7 hard deadlines
- Deadlines are a way of life, not a sudden crisis
- Balancing ambitions with resources
- Realistic estimations of work load
- Getting started in time
  
- **You are better at facing stress and deadlines**

# Finding Help and Information

---

- Specifications may be missing or unclear
- Don't panic!
- Help is available from many sources
- RTFM
- Help each other
- Use weekly consulting productively
- Ask useful questions on the webboard
  
- **You are better at finding help and information**

# Surviving as a Group

---

- Group work is a challenge
- Negotiating ambitions
- Exploiting different skills
- Setting aside your ego
- Respecting internal deadlines
- Ensuring your own outcome
  
- **You have more experience in group work**

# A Large Tool to Solve a Large Problem

---

- The project is **much** larger than that of comparable courses at other institutions
- This is made possible by:
  - big technology
  - explicit phase slicing
  - the online test tool
- Pros:
  - more fun, more instructive, more real
- Cons:
  - possibly confusing (compared to a C0 compiler)

# Status After the Course

---

- You have acquired nice new competences
- You have an experience to look back on
- You should be proud of yourselves
  
- You probably know by now whether you like compilers and programming languages
  
- In any case you will probably view programming languages differently from now on:
  - their definition, error messages, ...

# Want More?

---

- Look for courses by members of the PL group:
  - Olivier Danvy
  - Erik Ernst
  - Jan Midtgaard
  - Anders Møller
  - Michael Schwartzbach
- Spring 2012:
  - Introduction to Functional Programming (Q3)
  - Static Analysis (Q3)
  - Types in Object-Oriented Languages (Q3)
  - Virtual Machines for Programming Languages (Q3)
  - Abstract Interpretation (Q4)
  - Functional-Programming Techniques (Q4)
  - Software Verification (Q4)

# Want Even More?

---

- Interested in a MSc thesis in a programming language-related topic? Talk to us.
  - Compiler construction
  - Static analysis
  - Program transformation
  - Functional/OO/web/... programming
  - Type systems
  - Domain-specific languages
  - Interpreters and virtual machines
  - ...

# Want Still More?

---

- PhD studies
- 4+4 model starts after 4 years of study
- If you are fascinated by CS and want to go deep, then a PhD may be the right thing for you!
  - You will work on interesting things
  - Lots of freedom, few obligations
  - You are paid to learn!
- Talk to us early (now?), if you are interested