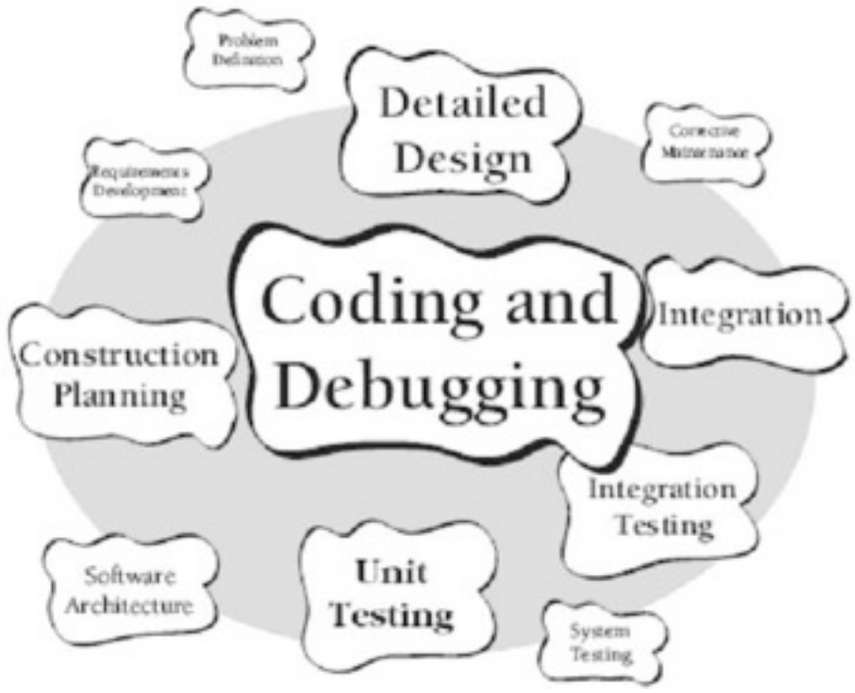


Software Construction

2012-2013
Tijs van der Storm (storm@cwi.nl / @tvdstorm)

UNIVERSITEIT VAN AMSTERDAM

CWI

What this course is about

- You all know programming, right?
- But what is good code?
- How to *reason* about good code?
- What is *beautiful* code?
- Think about it.

This course is *not* about

- Data structures
- Algorithms
- Programming language X
- Paradigm X (though: OO)
- GUI programming
- Web applications
- Concurrency
- Performance
- Graphics programming
- Mathematics
- Computational complexity
- ...

Uncle Bob*

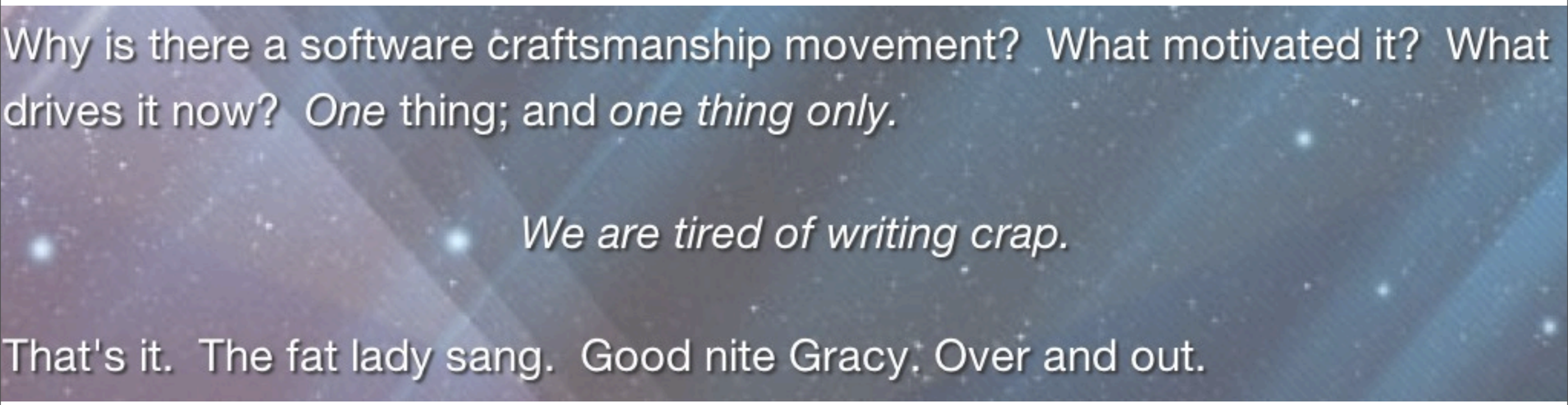
Why is there a software craftsmanship movement? What motivated it? What drives it now? *One* thing; and *one thing only.*

We are tired of writing crap.

That's it. The fat lady sang. Good nite Gracy. Over and out.

*Robert Martin, http://cleancoder.posterous.com/software-craftsmanship-things-wars-commandmen

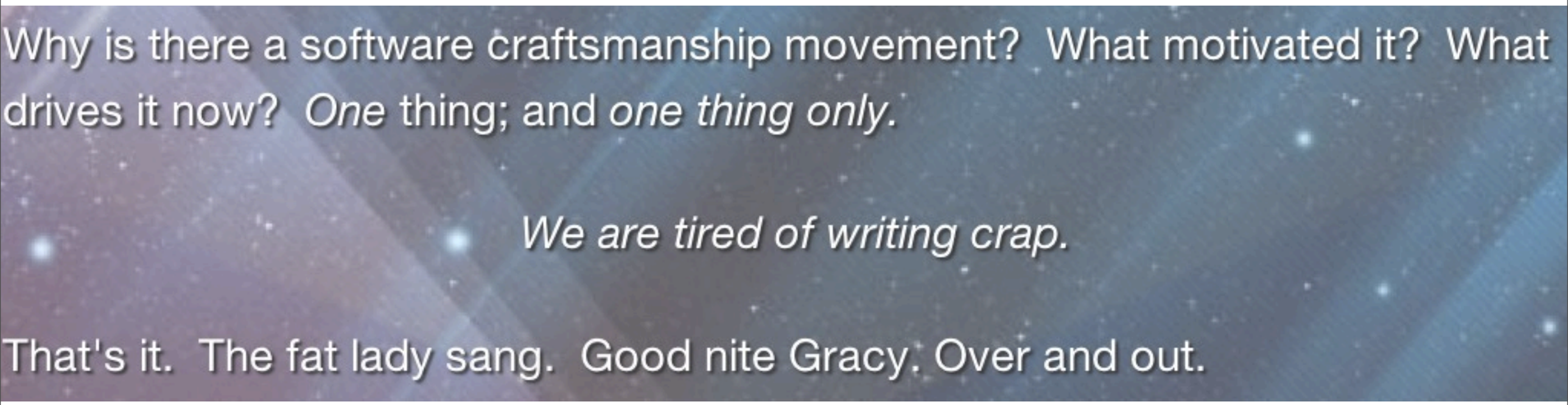
Uncle Bob*



This course is *not* about the software craftmanship movement...

*Robert Martin, http://cleancoder.posterous.com/software-craftsmanship-things-wars-commandmen

Uncle Bob*

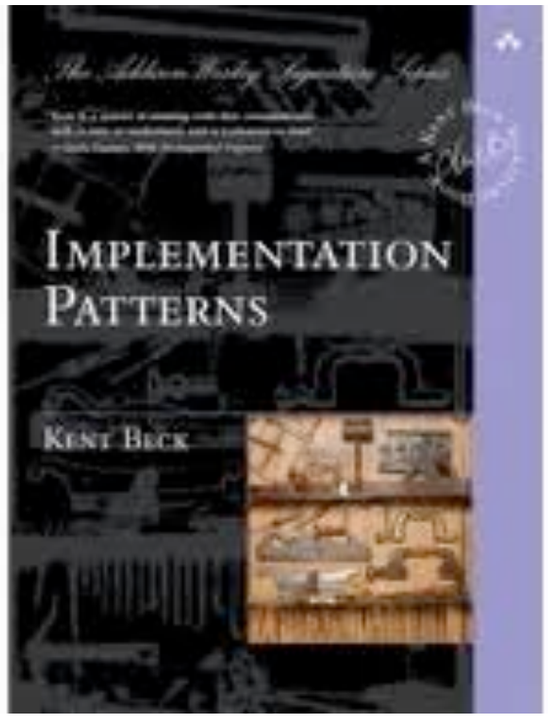


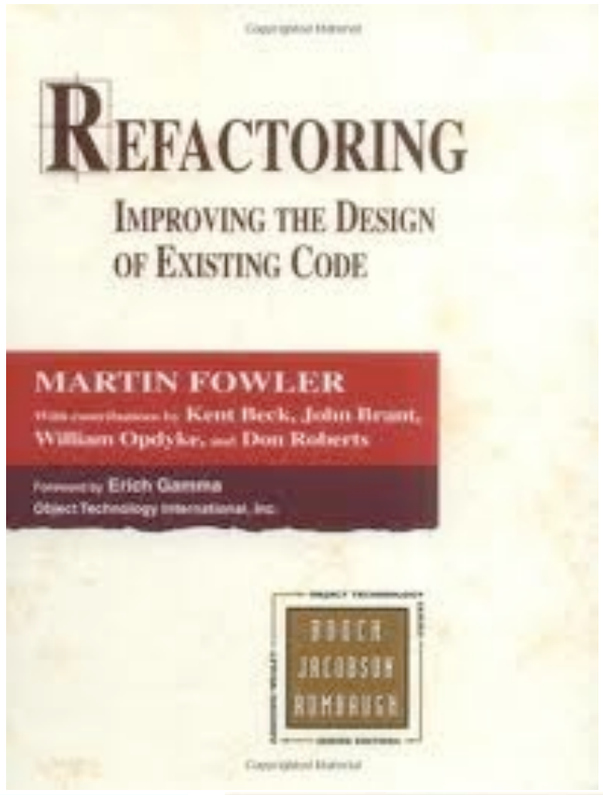
This course is *not* about the software craftmanship movement...

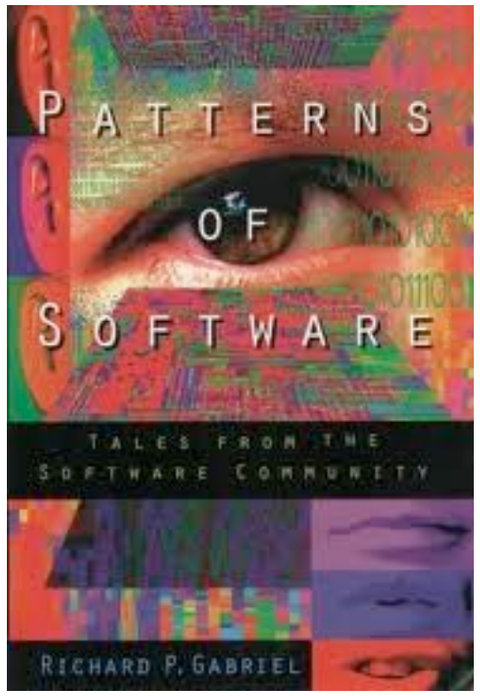
This course is about *not* writing crap.

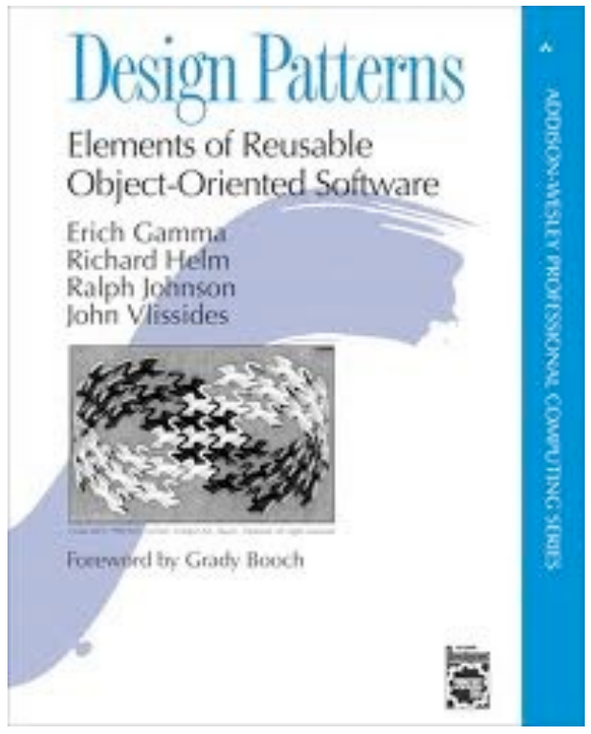
*Robert Martin, http://cleancoder.posterous.com/software-craftsmanship-things-wars-commandmen

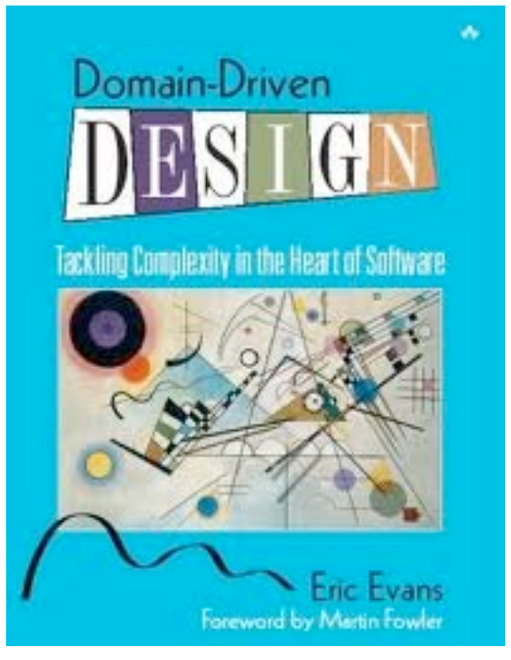
Representative books

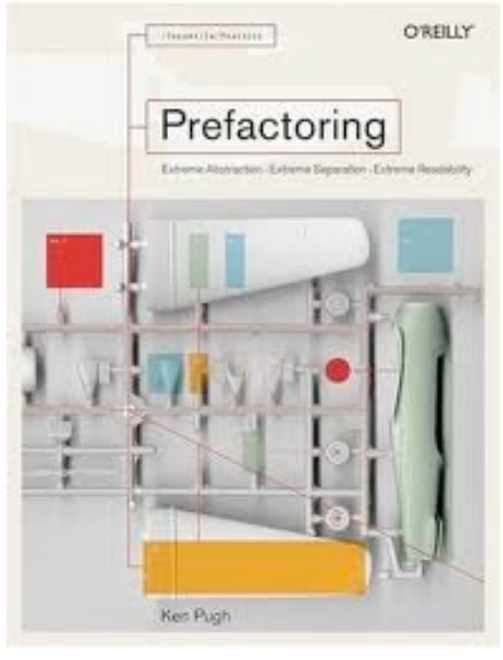


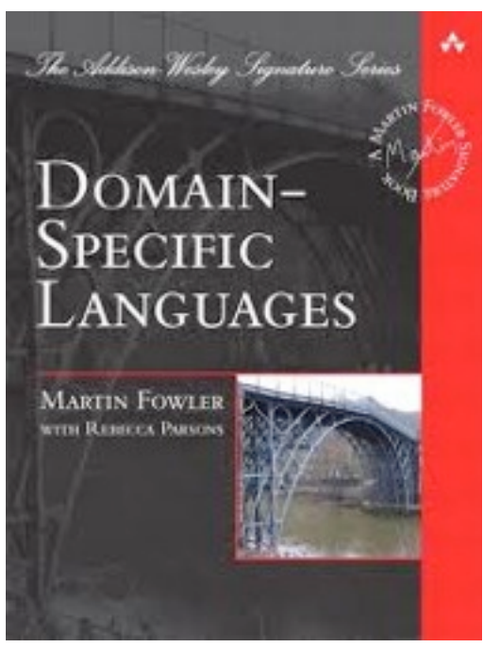










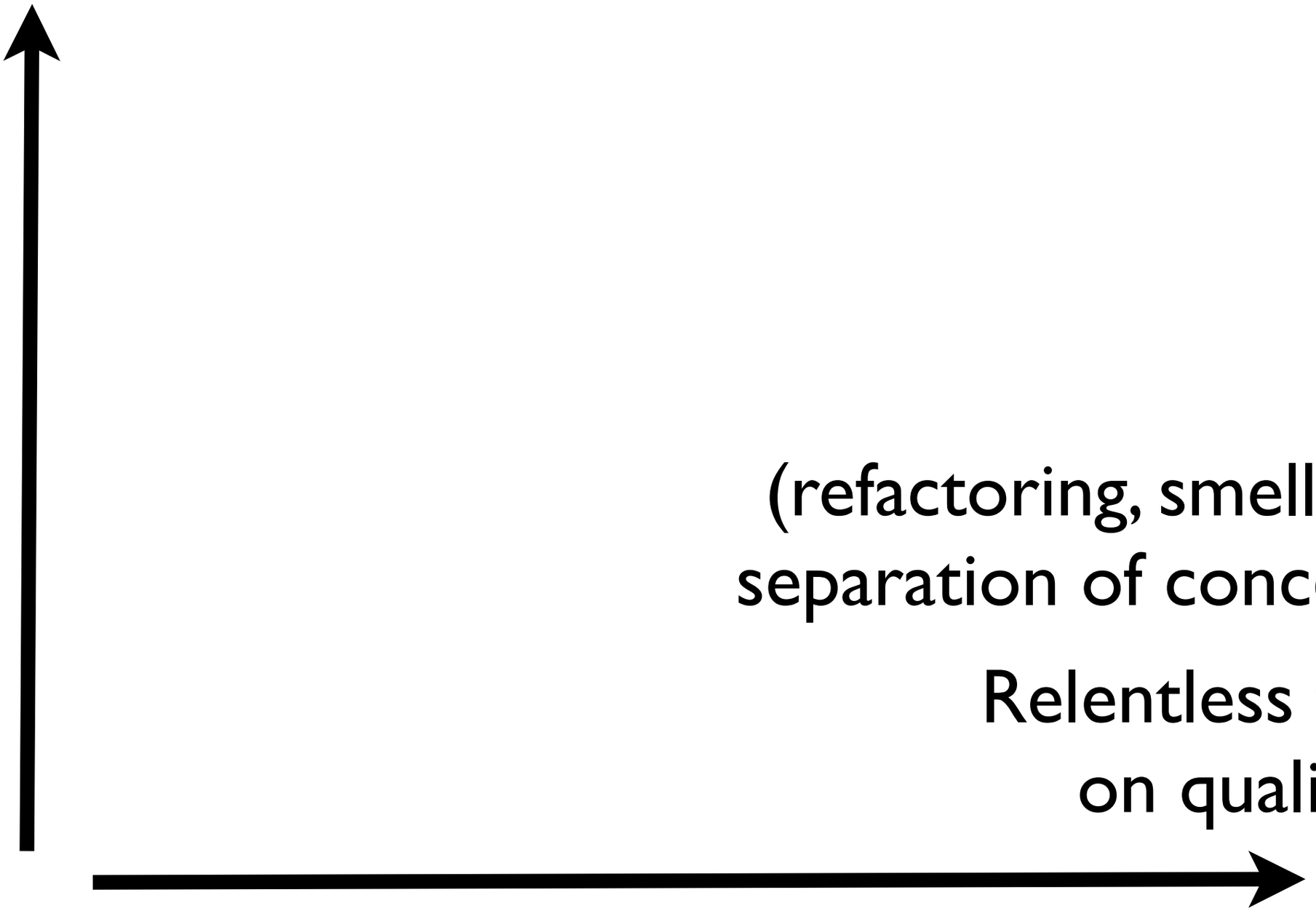


Learning goals

- Create good low level designs
- Produce clean, readable code
- Reflect upon techniques, patterns, guidelines etc.
- Assess the quality of code
- Apply state of the art software construction tools

Program something hard

(new techniques, concepts, tools)



(refactoring, smells, design, separation of concerns, etc.)

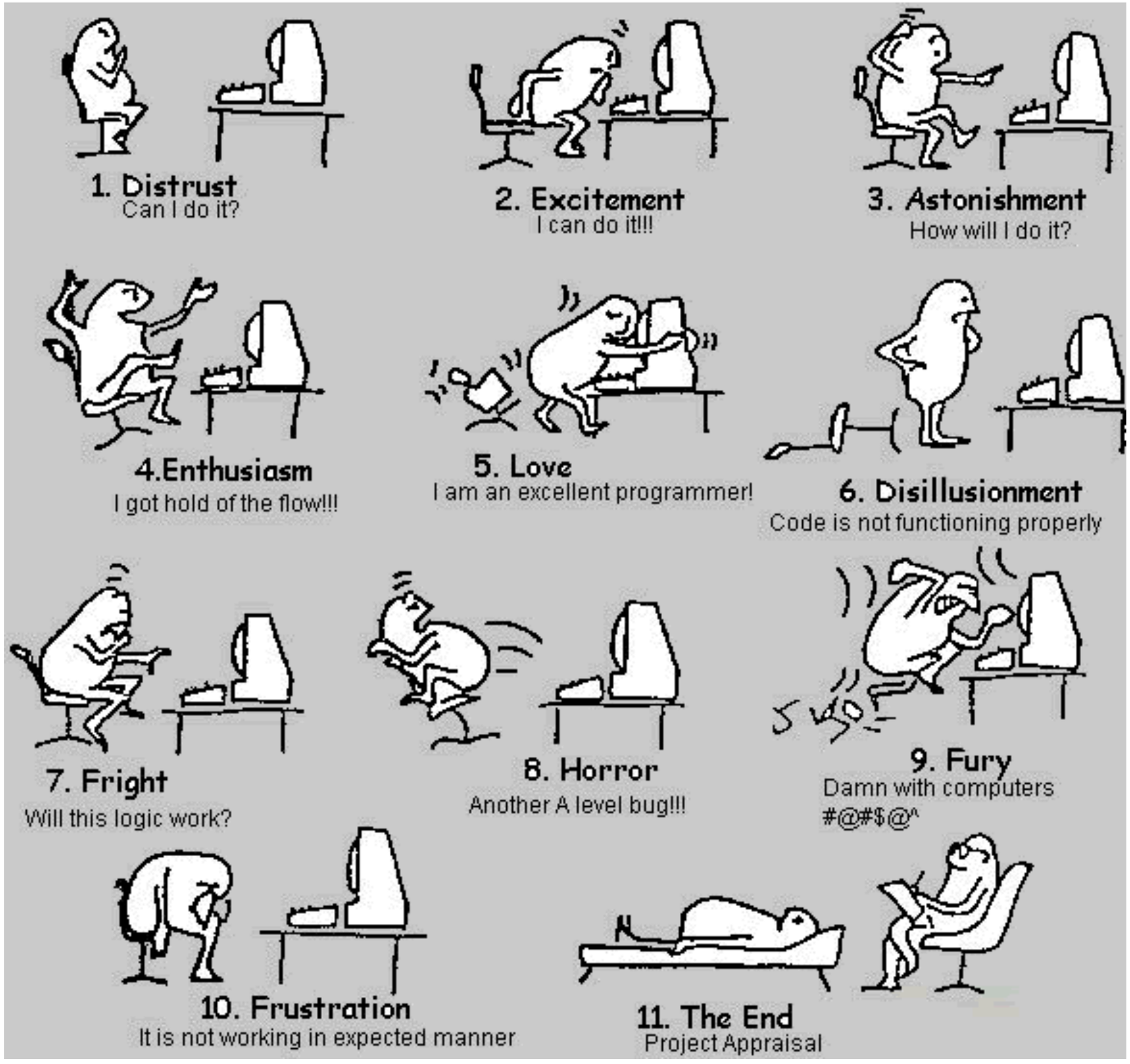
Relentless focus on quality

Program something hard

(new techniques, concepts, tools)

(refactoring, smells, design, separation of concerns, etc.)

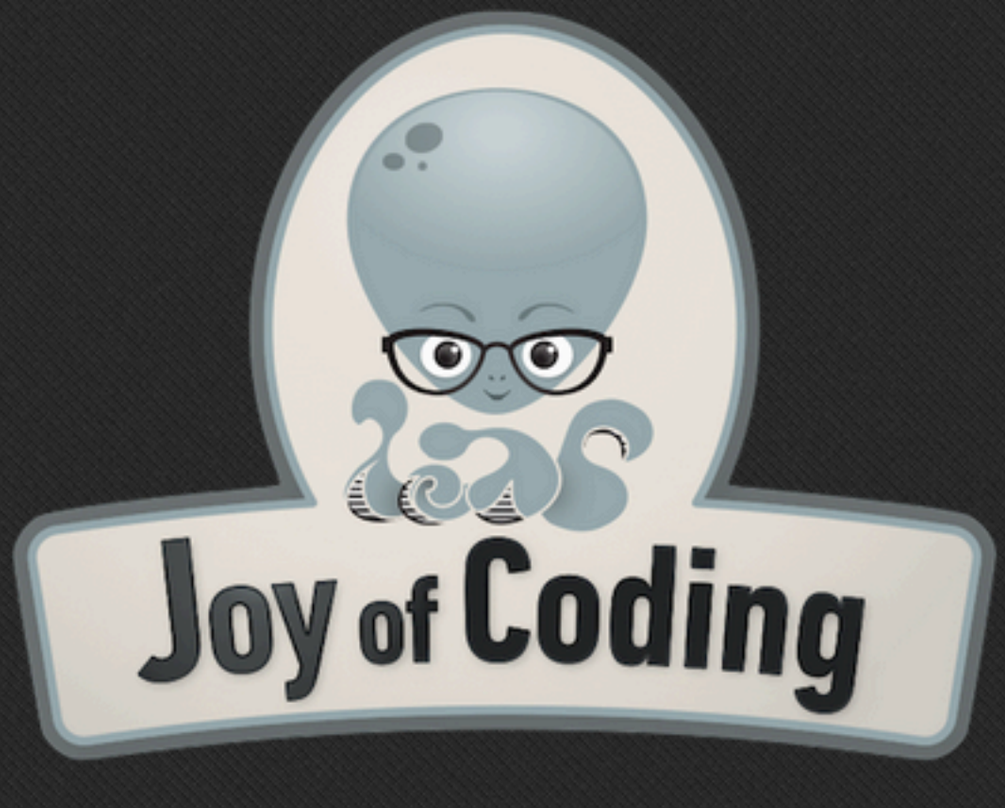
Relentless focus on quality



http://candraadi.wordpress.com/2012/10/17/programmer/

Buy your ticket now, only € 130!

save money, buy before January 15th



March 1st, 2013

Rotterdam (NL)

A one-day conference that celebrates the art, craft, science but foremost the joy of software development

The Pragmatic Bookshelf

PRAGMATIC LIFE

THE PASSIONATE PROGRAMMER

CREATING A REMARKABLE CAREER IN SOFTWARE DEVELOPMENT

CHAD FOWLER

FOREWORD BY DAVID HEINEMEIER HANSSON

This course

- Quality comes first
- Be your own worst critic
- Refactor mercilessly
- Aim to become code literati
- Better to read code, than to write code
- If it works it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's

not good enough

If it works, it's not good enough

If it works, it's not good enough

Working code is necessary, but not sufficient

Why?

Fact 41. Maintenance typically consumes 40 to 80 percent of software costs. It is probably the most important life cycle phase of software.

Fact 44. Understanding the existing product is the most difficult task of maintenance.

Fact 21. For every 25 percent increase in problem complexity, there is a 100 percent increase in solution complexity.

Robert Glass, *Facts and fallacies of Software Engineering*, Addison-Wesley 2003

Overview

- Lectures
- Theory
- Research paper + review
- Lab assignment
- Concluding

Lectures

Lectures

- Introduction (today)
- Syntax analysis
- Domain-specific languages
- Code quality
- Debugging

Guest lectures

- Jeroen van den Bos: DSL for digital forensics
- ?

“Theory”

Two tests

- Oldskool: on paper
- No grade, but you must *pass* them
- Selection from known questions

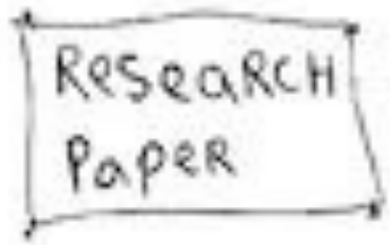
1st test: technique

- Bertrand Meyer, *Applying "Design by Contract"*, 1992, Meyer92.
- Karl J. Lieberherr, Ian M. Holland, *Assuring Good Style for Object-Oriented Programs*, 1989,LieberherrHolland89.
- Robert C. Martin, *The Open-Closed Principle*, 1996, Martin96.
- Ralph Johnson, Brian Foote, *Designing reusable classes*, 1988, JohnsonFoote88.
- Marjan Mernik et al. *When and How to Develop Domain Specific Languages*, 2005,MernikEtAl05.

2 test: philosophy

- D. L. Parnas, *On the criteria to be used in decomposing systems into modules* , 1972, Parnas72
- William R. Cook, *On understanding data abstraction, revisited*, 2009, Cook09.
- Herbert Simon, *The Architecture of Complexity*, Simon62
- Carliss Y. Baldwin, Kim B. Clark, *Modularity in the Design of Complex Engineering Systems*,BaldwinClark06
- Horst Rittel, Melvin Webber, *Dilemmas in a General Theory of Planning*, RittelWebber84.

Research paper



Exercise in argumentation

- Paper: 3-5 pages
- Topics from predefined list
- Identify the central claim
- Argue **against** it (debunking)
- Required: find 2 more papers for support
- Must be clear you've read all 4 papers

Structured programming with or without gotos?

- E.W. Dijkstra *Goto considered harmful*, 1968, Dijkstra68;
- D. Knuth, *Structured Programming with go to statements*, 1974, Knuth74.

State Transactional Memory

- Simon Peyton Jones, *Beautiful Concurrency*, 2007, PeytonJones07.
- Calin Cascaval et al. *Software Transactional Memory: Why is it Only a Research Toy?*, 2008,CascavalEtAl08.
- Bryan Cantrill and Jeff Bonwick, *Real-world Concurrency*, 2008, CantrillBonwick08.

Internal vs external DSLs

- Marjan Mernik et al. *When and How to Develop Domain Specific Languages*, 2005,MernikEtAl05.
- Martin Fowler, *Implementing an Internal DSL*, 2007 Fowler07.

Aspect-Oriented Programming

- Gregor Kiczales et al. *Aspect-Oriented Programming*, 1997, KiczalesEtAl97.
- Robert E. Filman, Daniel P. Friedman, *Aspect-Oriented Programming is Quantification and Obliviousness*, 2000, FilmanFriedman00.

Literate Programming in the 21st Century

- Bentley, Knuth and McIllroy, *A Literate Program*, 1986, BentleyEtAl86.
- Knuth, *Literate Programming*, 1984, Knuth84.

Prototype-based vs class-based object-orientation

- James Noble, Brian Foote, *Attack of the Clones*, 2002, NobleFoote02.
- Henry Lieberman, *Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems*, 1986, Lieberman86.

Design by Contract

- Bertrand Meyer, *Applying "Design by Contract"*, 1992, Meyer92.
- Jean-Marc Jézéquel, Bertrand Meyer, *Design by Constract: The Lessons of Ariane*, 1997,JezequelMeyer97.

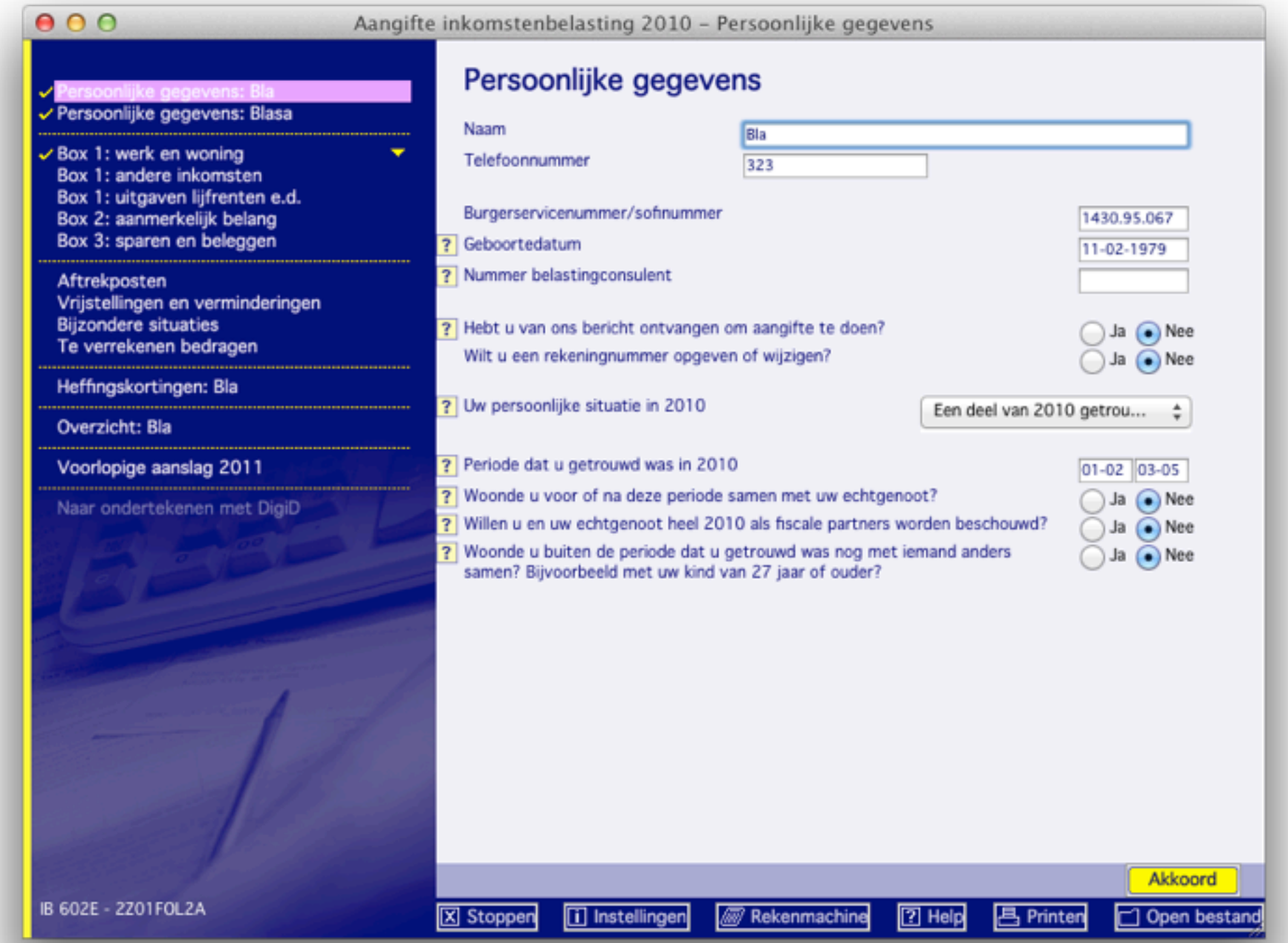
Fluent or Law-of-Demeter?

- Martin Fowler, *FluentInterface*, 2005, Fowler05.
- Karl J. Lieberherr, Ian M. Holland, *Assuring Good Style for Object-Oriented Programs*, 1989,LieberherrHolland89.

Reviewing

- Next to paper: write a review of someone else's paper
- Structured review
- We'll use EasyChair (pending approval)
- Paper and review not graded, by okayed.

Lab assignment



Form **1040** Department of the Treasury—Internal Revenue Service (99)

U.S. Individual Income Tax Return

2012 OMB No. 1545-0074 IRS Use Only—Do not write or staple in this space.

For the year Jan. 1–Dec. 31, 2012, or other tax year beginning , 2012, ending , 20 See separate instructions.

Your first name and initial	Last name	**Your social security number**
If a joint return, spouse's first name and initial	Last name	**Spouse's social security number**
Home address (number and street). If you have a P.O. box, see instructions.	Apt. no.	▲ Make sure the SSN(s) above and on line 6c are correct.
City, town or post office, state, and ZIP code. If you have a foreign address, also complete spaces below (see instructions).		**Presidential Election Campaign** Check here if you, or your spouse if filing jointly, want $3 to go to this fund. Checking a box below will not change your tax or refund. ☐ **You** ☐ **Spouse**
Foreign country name	Foreign province/state/county / Foreign postal code	

Filing Status

Check only one box.

1 ☐ Single
2 ☐ Married filing jointly (even if only one had income)
3 ☐ Married filing separately. Enter spouse's SSN above and full name here. ▶
4 ☐ Head of household (with qualifying person). (See instructions.) If the qualifying person is a child but not your dependent, enter this child's name here. ▶
5 ☐ Qualifying widow(er) with dependent child

Exemptions

6a ☐ **Yourself.** If someone can claim you as a dependent, **do not** check box 6a
b ☐ **Spouse**

c **Dependents:** (1) First name / Last name	(2) Dependent's social security number	(3) Dependent's relationship to you	(4) ✓ if child under age 17 qualifying for child tax credit (see instructions)
			☐
			☐
			☐
			☐

If more than four dependents, see instructions and check here ▶ ☐

d Total number of exemptions claimed

Boxes checked on 6a and 6b
No. of children on 6c who:
• lived with you
• did not live with you due to divorce or separation (see instructions)
Dependents on 6c not entered above
Add numbers on lines above ▶

Income

Attach Form(s) W-2 here. Also attach Forms W-2G and 1099-R if tax was withheld.

If you did not get a W-2, see instructions.

Enclose, but do not attach, any payment. Also, please use **Form 1040-V.**

7	Wages, salaries, tips, etc. Attach Form(s) W-2		7
8a	**Taxable** interest. Attach Schedule B if required		8a
b	**Tax-exempt** interest. **Do not** include on line 8a	8b	
9a	Ordinary dividends. Attach Schedule B if required		9a
b	Qualified dividends	9b	
10	Taxable refunds, credits, or offsets of state and local income taxes		10
11	Alimony received		11
12	Business income or (loss). Attach Schedule C or C-EZ		12
13	Capital gain or (loss). Attach Schedule D if required. If not required, check here ▶ ☐		13
14	Other gains or (losses). Attach Form 4797		14
15a	IRA distributions 15a	b Taxable amount	15b
16a	Pensions and annuities 16a	b Taxable amount	16b
17	Rental real estate, royalties, partnerships, S corporations, trusts, etc. Attach Schedule E		17
18	Farm income or (loss). Attach Schedule F		18
19	Unemployment compensation		19
20a	Social security benefits 20a	b Taxable amount	20b
21	Other income. List type and amount		21
22	Combine the amounts in the far right column for lines 7 through 21. This is your **total income** ▶		22

Adjusted Gross Income

23	Reserved	23	
24	Certain business expenses of reservists, performing artists, and fee-basis government officials. Attach Form 2106 or 2106-EZ	24	
25	Health savings account deduction. Attach Form 8889	25	
26	Moving expenses. Attach Form 3903	26	
27	Deductible part of self-employment tax. Attach Schedule SE	27	
28	Self-employed SEP, SIMPLE, and qualified plans	28	
29	Self-employed health insurance deduction	29	
30	Penalty on early withdrawal of savings	30	
31a	Alimony paid b Recipient's SSN ▶	31a	
32	IRA deduction	32	
33	Student loan interest deduction	33	
34	Reserved	34	
35	Domestic production activities deduction. Attach Form 8903	35	
36	Add lines 23 through 35		36
37	Subtract line 36 from line 22. This is your **adjusted gross income** ▶		37

Questionnaire Language (QL)

- A DSL for specifying questionnaires

```
form Box1HouseOwning {
  "Did you sell a house in 2010?" hasSoldHouse: boolean
  "Did you by a house in 2010?" hasBoughtHouse: boolean
  "Did you enter a loan for maintenance/reconstruction?"
hasMaintLoan: boolean
  if (hasSoldHouse) {
    "Private debts for the sold house:" privateDebt: money
    "Price the house was sold for:" sellingPrice: money
    "Value residue:" valueResidue = sellingPrice - privateDebt
  }
}
```

Did you sell a house in 2010? ☑
Did you by a house in 2010? ☐
Did you enter a loan for maintenance/reconstruction? ☐
Private debts for the sold house:

20

Price the house was sold for:

200

Value residue:

180

LWC 2013

Navigation

Main page
Community portal
LWC 2013
LWC 2012
LWC 2011
Presentation Videos
Recent changes
Random page
Help

Contents [hide]

1 Language Workbench Challenge 2013
1.1 The assignment
1.2 Reference implementation
1.3 Submitting and participating
1.4 Demonstrations
1.5 Program - April 9th, 2013
1.6 Registration
1.7 Important dates
1.8 Announced participants

Language Workbench Challenge 2013

Part 1: front end

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Type checker
- Two variants:
  - Java
  - Rascal (but: you have to do a little more)

Java

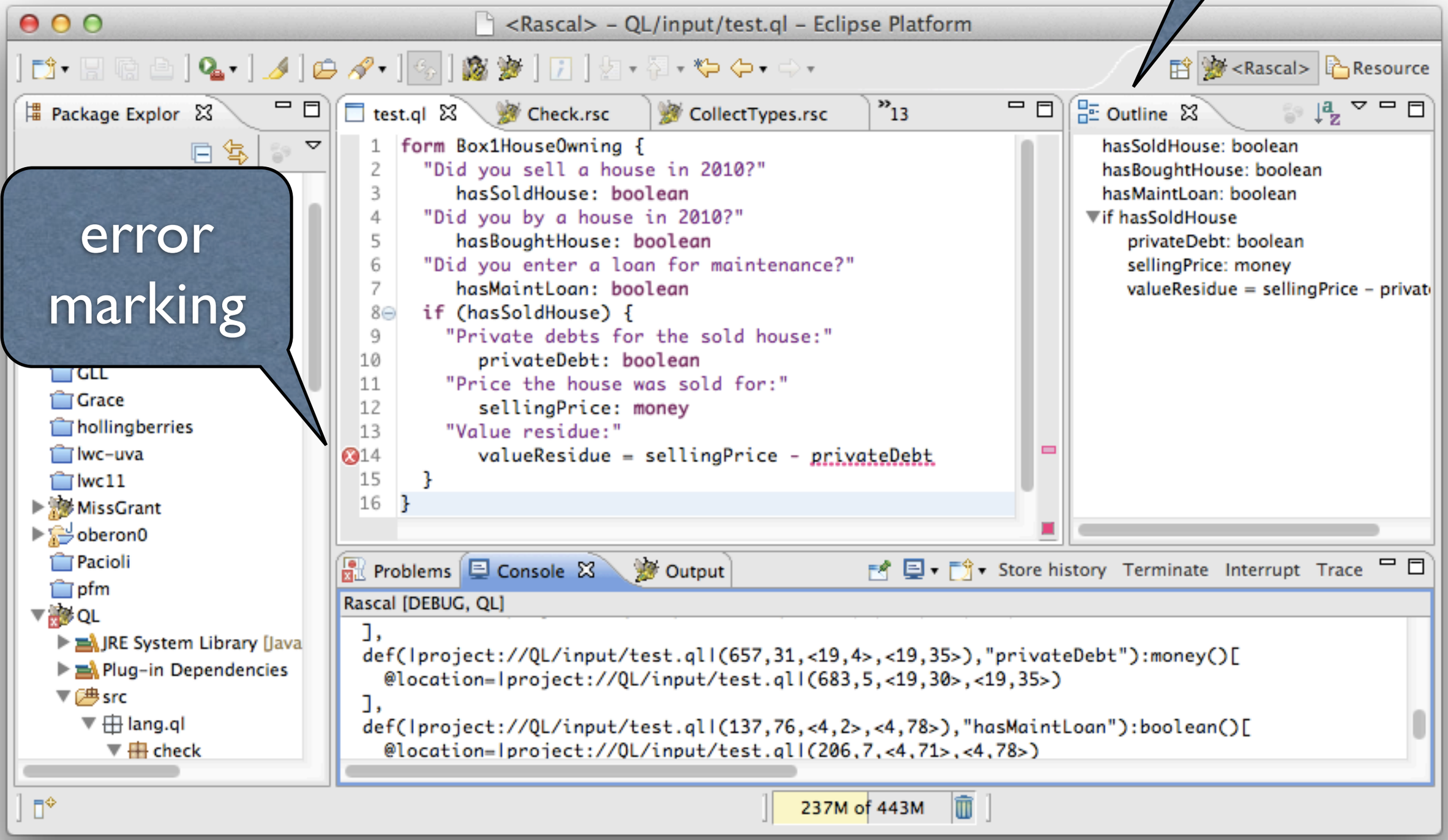
- Use one of the provided parsing skeletons: *Rats!*, Jacc, ANTLR
- Design AST class hierarchy (required!)
- Visitor/interpreter for checking well-formedness

Rascal

- Rascal is *designed* for making DSLs:
  - syntax definition
  - AST data types
  - type checking
  - desugaring
  - IDE features
- You have to do all of this.

outline

error marking



Part 2: Back end

- Implement questionnaire engine
- GUI representation of questionnaire
- Java: build an *interpreter*
- Rascal: build a *compiler*
- *See course info for more detail*

Advanced track

- For excellent students
- = Rascal variant
- + additional layout and styling language QLS
- Close collaboration with CWI (= me)
- *See course info for details*

Github

- Assignment to be completed *individually*
  - (except advanced track)
- http://github.com/software-engineering-amsterdam/sea-of-ql/
- Use of this repository is **required**!
- Email your Github name to me for access

Grading of lab assignments

- Functionality
- Tests
- Simplicity
- Modularity
- Layout and style
- Separation of concerns

Grading of la
assignmen

- Functi ity
- Te
- imp icity
- Modularity
- Layout and style
- Separation f conce s

Simpli city

Some advice up-front

- Naming, layout, indentation
- Encapsulation, modularity, separation of concerns, reuse
- Don’t repeat yourself (DRY)
- Library and tool selection and use
- Unit testing

More advice

- Use asserts sensibly
- No global, static, non-final variables
- You ain't going to need it (YAGNI)
- Avoid premature optimization
- Use comments for rationale
- *Compiling **and** working code*

Grading (ctd.)

- First part: your grade is *indicative*
  - hint to improve your code
- Second part: we review all code
  - this will be your *final* grade for the lab
- Grading is on-site: *you* show your code
- Grade is not important, improvement is

Passing this course

- Be present at all lectures
- Pass the 2 theory tests
- Successfully complete lab assignments
  - = beautiful, working code
- Write a good research paper and review
- Final grade: *FinalLab*

Schedule

Date	Lecture	Tests/Grading
7-1	Introduction	
14-1	Grammars and parsing	
21-1	Domain-specific languages	Part I
28-1	Code quality	Test I
4-2	Debugging (maybe)	
11-2	*Guest lecture*	Test II
18-2	*Guest lecture*	Part II
25-2	Extra paper writing time	due: 3-3-12

Concluding

- All information is in Blackboard under *Course Information* (links to Github wiki)
- Primary contact = me (storm@cwi.nl)

What's next

- study QL exercise
- get access to Github
- select study skeleton code
- setup you project using Eclipse
- start reading papers
- start programming!