



Software Construction

2014-2015

Tijs van der Storm

(storm@cwi.nl / @tvdstorm / @SoftwCons)

Actually: @JurgenVinju this time...



UNIVERSITEIT VAN AMSTERDAM



Introduction



Jurgen Vinju
(speaker)



Tijs van der Storm
(lectures + labs)



Vadim
(labs)

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
- ...

Mastering Code

- **Skill**

- design is a verb: can you do it?
- be a critic: can you judge quality?

- **Knowledge**

- algorithms, programming language
- design patterns, domain knowledge

- **Attitude**

- wanting to be a master of construction

Use the force,
read the source



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.

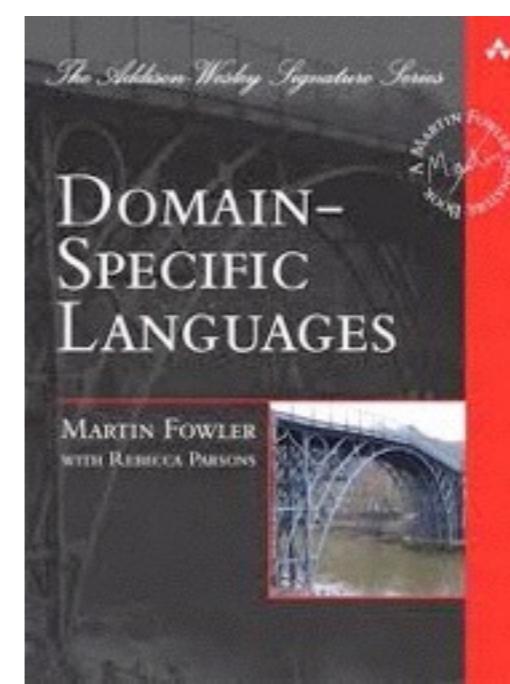
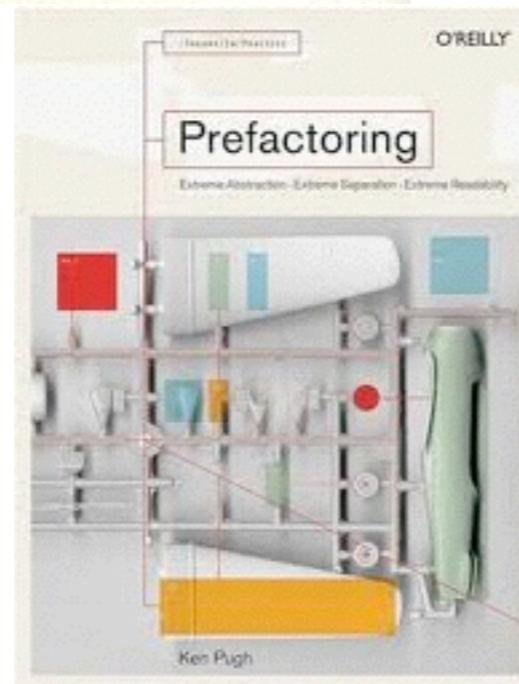
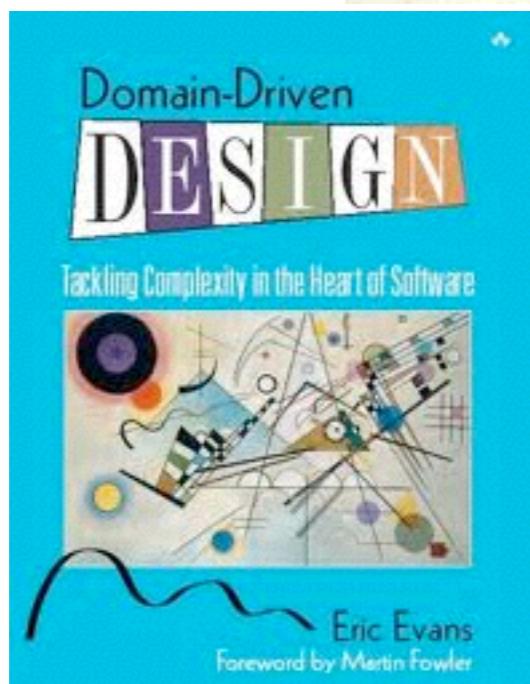
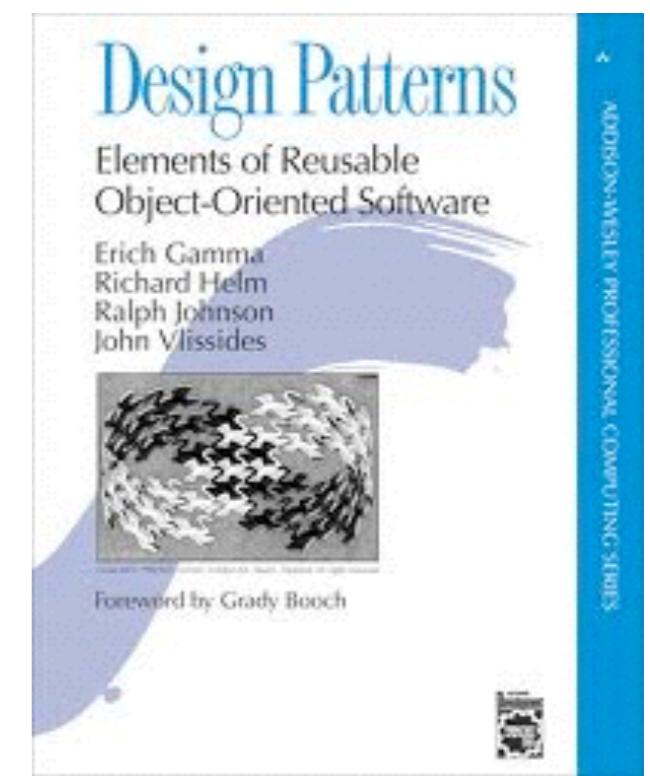
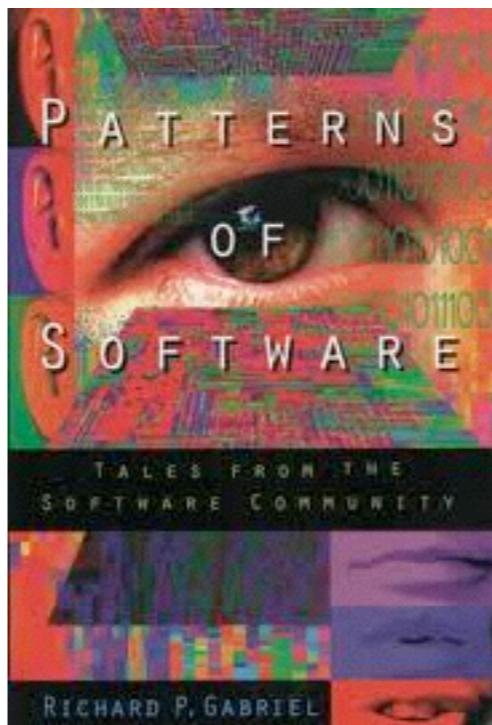
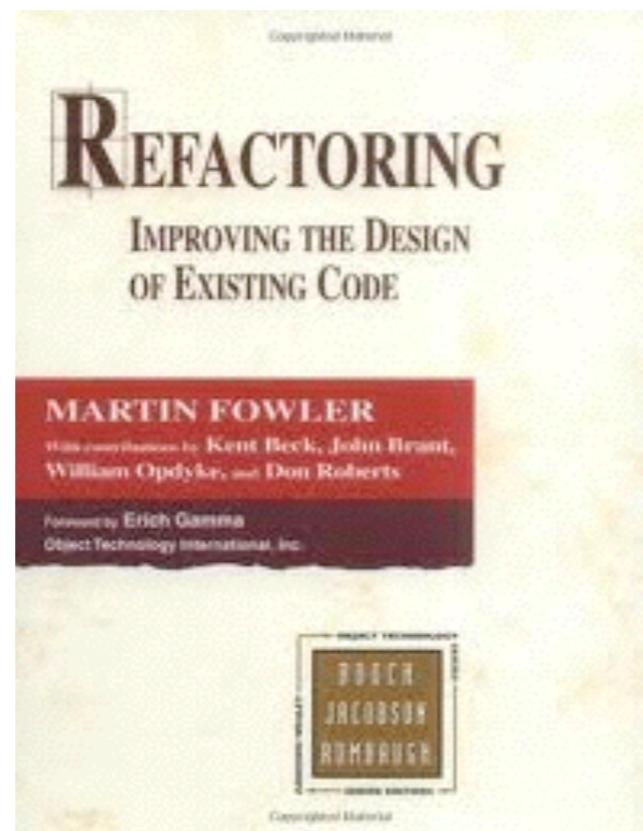
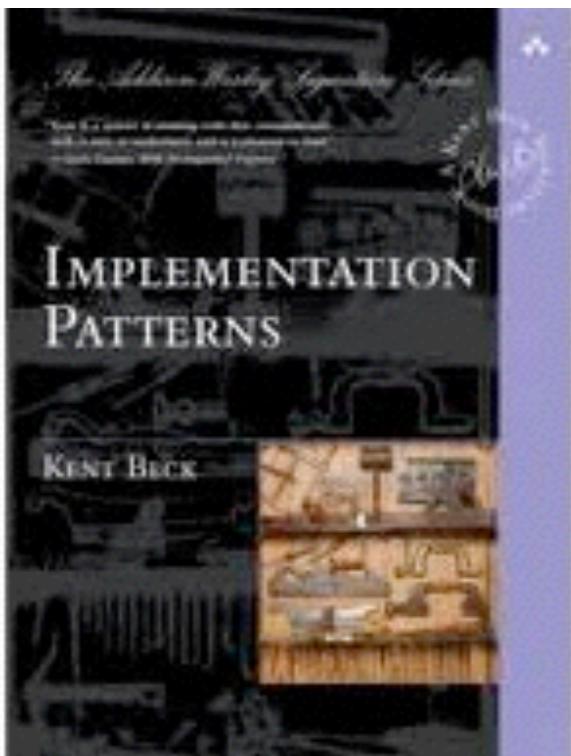
That's it. The fat lady sang. Good nite Gracy. Over and out.

**This course is *not* about the software
craftsmanship 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



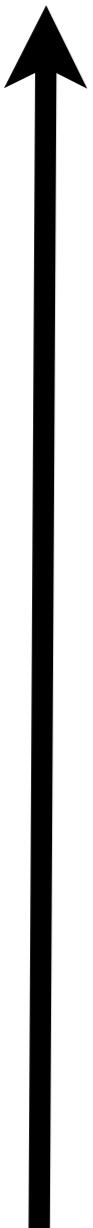
Program
something
hard

(new techniques,
concepts, tools)



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

Relentless focus
on quality





1. Distrust
Can I do it?



2. Excitement
I can do it!!!



3. Astonishment
How will I do it?



4. Enthusiasm
I got hold of the flow!!!



5. Love
I am an excellent programmer!



6. Disillusionment
Code is not functioning properly



7. Fright
Will this logic work?



8. Horror
Another A level bug!!!



9. Fury
Damn with computers
#@#\$@^

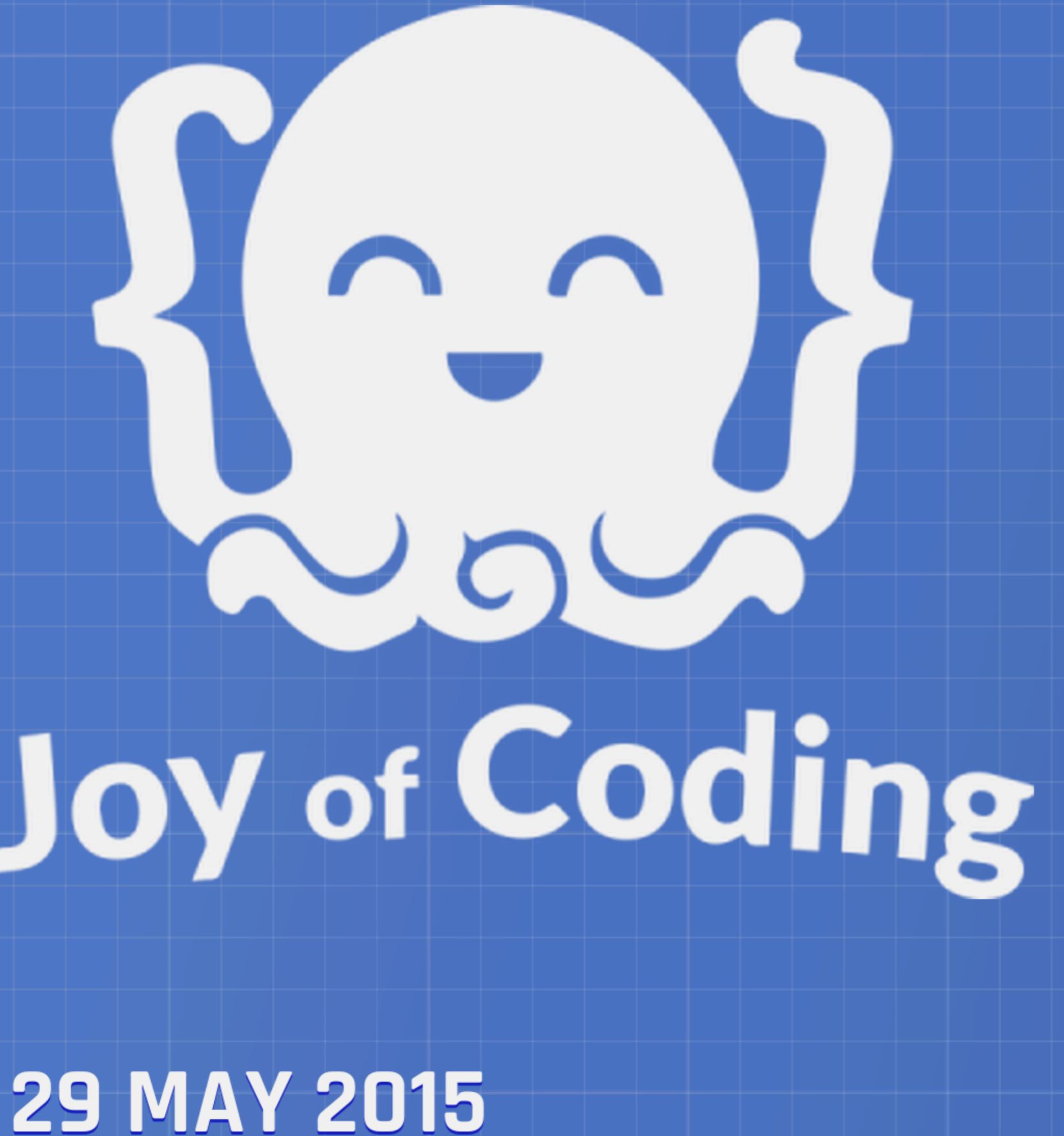


10. Frustration
It is not working in expected manner



11. The End
Project Appraisal





joyofcoding.org

29 MAY 2015

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



SUSPENSION OF DISBELIEF

If it works, it's not good enough

Working code is necessary, but not sufficient

Why

- Because... software evolution
- Because... software architecture
- Because... software process
- All of that is to try and mitigate problems introduced at construction time.
- But important details are lost in those abstractions

Why

- The restaurant owner designs a concept for a restaurant, but what if the chef makes bad food...
- Formula 1 cars all share the same architecture, but what if the mechanical engineer builds and tunes the machine badly?
- You can buy all the most expensive ski gear, and still end up in the hospital
- *To achieve real quality a master can not not think about the details.*

Overview

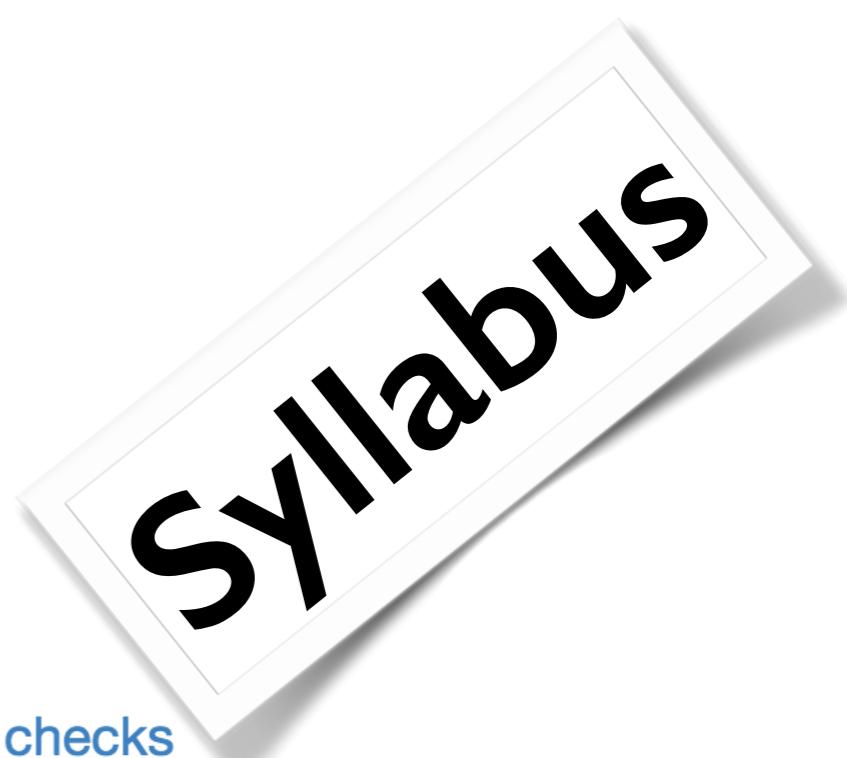
- Lectures
- Theory: papers + book
- Exam: lectures + papers + book
- Lab assignment: implement a little language
- Concluding

Lectures

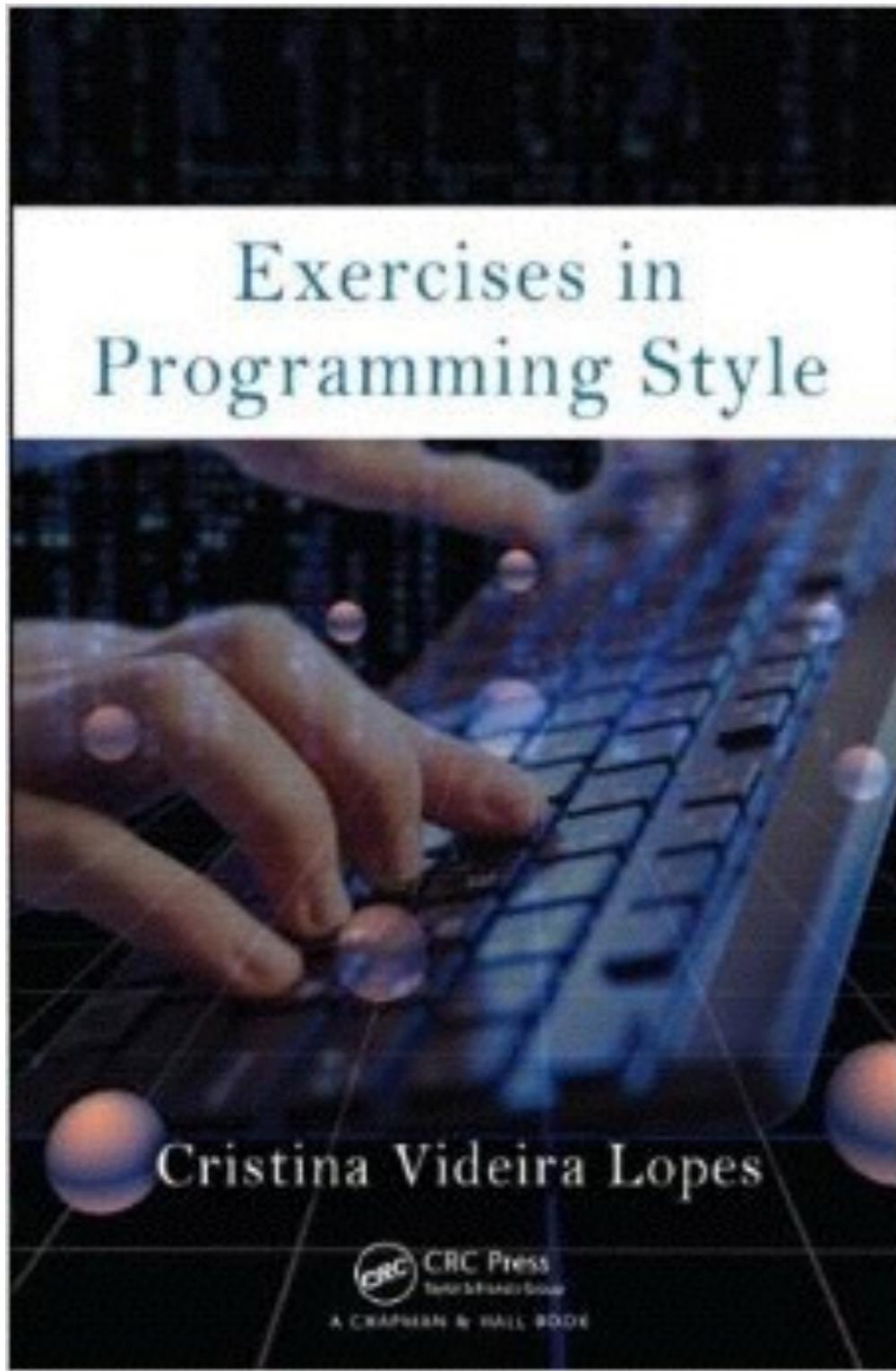


Topics of the lectures

- Syntax analysis: grammars, parsers
- Programming styles, design principles etc.
- Code quality: tangling, scattering, duplication, smells, refactoring, layout
- Modularity: information hiding, separation of concerns, encapsulation, dependency
-

- Karl J. Lieberherr, Ian M. Holland, *Assuring Good Style for Object-Oriented Programs*, 1989, [LieberherrHolland89](#)
 - D. L. Parnas, *On the criteria to be used in decomposing systems into modules* , 1972, [Parnas72](#)
 - W. Wulf and Mary Shaw, *Global variable considered harmful*, 1973, [WulfShaw84](#).
 - John Hughes, *Why functional programming matters*, 1990 [Hughes89](#)
 - Robert C. Martin, *Design principles and design patterns*, [Martin00](#).
 - Erich Gamma, Richard Helm, Ralphi Johnson, John Vlissides, *Design Patterns: Abstraction and Reuse of Object Oriented Design*, ECOOP 93 [GammaEtAl93](#)
 - Kent Beck and Martin Fowler, *Bad Smells in Code* (Chapter 3, *Refactoring*)
 - Kent Beck, A theory of programming, (Chapter 3, *Implementation Patterns*)
 - Kent Beck, *Aim, fire*, IEEE Software, [Beck01](#)
 - Jeff Bay, *Object Calisthenics*, [Bay](#).
 - Ward Cunningham, *The CHECKS Pattern Language of Information Integrity*, [checks](#)
 - Kernighan, Plauger, *Programming Style: Examples and Counterexamples*, 1974 [kernighanPlauger](#)
 - Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin, *Aspect-Oriented Programming*, [KiczalesEtAl97](#)
 - James Noble, *Arguments and Results*, [Noble97](#)
- 

Book: EiPS



Final exam

- Open book exam
- Must have grade > 5.5
- Based on
 - Exercises in Programming Style
 - Lecture material
 - Syllabus

Lab assignment

Aangifte inkomstenbelasting 2010 – Persoonlijke gegevens

Persoonlijke gegevens: Bla
Bla

Box 1: werk en woning
Box 1: andere inkomsten
Box 1: uitgaven lijfrenten e.d.
Box 2: aanmerkelijk belang
Box 3: sparen en beleggen

Aftrekposten
Vrijstellingen en verminderingen
Bijzondere situaties
Te verrekenen bedragen

Heffingskortingen: Bla

Overzicht: Bla

Voorlopige aanslag 2011

Naar ondertekenen met DigiD

Persoonlijke gegevens

Naam: Bla
Telefoonnummer: 323
Burgerservicenummer/sofinummer: 1430.95.067
Geborendatum: 11-02-1979
Nummer belastingconsulent:
Hebt u van ons bericht ontvangen om aangifte te doen? Ja Nee
Wilt u een rekeningnummer opgeven of wijzigen? Ja Nee
Uw persoonlijke situatie in 2010: Een deel van 2010 getrouwd
Periode dat u getrouwde was in 2010: 01-02 03-05
Woonde u voor of na deze periode samen met uw echtgenoot? Ja Nee
Willen u en uw echtgenoot heel 2010 als fiscale partners worden beschouwd? Ja Nee
Woonde u buiten de periode dat u getrouwde was nog met iemand anders samen? Bijvoorbeeld met uw kind van 27 jaar of ouder? Ja Nee

Akkoord

IB 602E - 2201FOL2A

Stoppen Instellingen Rekenmachine Help Printen Open bestand

Form
1040Department 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 [Dashed boxes]
If a joint return, spouse's first name and initial	Last name			Spouse's social security number [Dashed boxes]
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 <input type="checkbox"/> Single	4 <input type="checkbox"/> 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. ►		
	2 <input type="checkbox"/> Married filing jointly (even if only one had income)	5 <input type="checkbox"/> Qualifying widow(er) with dependent child		
	3 <input type="checkbox"/> Married filing separately. Enter spouse's SSN above and full name here. ►			
Exemptions	6a <input type="checkbox"/> Yourself. If someone can claim you as a dependent, do not check box 6a	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 ► [Box]		
	b <input type="checkbox"/> Spouse			
If more than four dependents, see instructions and check here ► <input type="checkbox"/>	c Dependents: (1) First name Last name [Dashed boxes] [Dashed boxes] [Dashed boxes] [Dashed boxes]	(2) Dependent's social security number [Dashed boxes] [Dashed boxes] [Dashed boxes] [Dashed boxes]	(3) Dependent's relationship to you [Dashed boxes] [Dashed boxes] [Dashed boxes] [Dashed boxes]	(4) ✓ if child under age 17 qualifying for child tax credit (see instructions) [Dashed boxes] [Dashed boxes] [Dashed boxes] [Dashed boxes]
	d Total number of exemptions claimed			
Income	7 Wages, salaries, tips, etc. Attach Form(s) W-2	7		
Attach Form(s) W-2 here. Also attach Forms W-2G and 1099-R if tax was withheld.	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		
If you did not get a W-2, see instructions.	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 ► <input type="checkbox"/>	13		
	14 Other gains or (losses). Attach Form 4797	14		
Enclose, but do not attach, any payment. Also, please use Form 1040-V.	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		

Part I: Questionnaire Language (QL)

```
form taxOfficeExample {  
    "Did you sell a house in 2010?"  
        hasSoldHouse: boolean  
    "Did you buy a house in 2010?"  
        hasBoughtHouse: boolean  
    "Did you enter a loan?"  
        hasMaintLoan: boolean  
  
    if (hasSoldHouse) {  
        "What was the selling price?"  
            sellingPrice: money  
        "Private debts for the sold house:"  
            privateDebt: money  
        "Value residue:"  
            valueResidue: money =  
                (sellingPrice - privateDebt)  
    }  
}
```

Describe the logic of
interactive questionnaires

- Did you sell a house in 2010?
- Did you buy a house in 2010?
- Did you enter a loan?

- Did you sell a house in 2010?
- Did you buy a house in 2010?
- Did you enter a loan?
- What was the selling price?
24234
- Private debts for the sold house:
34343
- Value residue:
-10109

QLS

```
stylesheet taxOfficeExample
page Housing {
    section "Buying"
        question hasBoughtHouse
            widget checkbox
    section "Loaning"
        question hasMaintLoan
}

page Selling {
    section "Selling" {
        question hasSoldHouse
            widget radio("Yes", "No")
        section "You sold a house" {
            question sellingPrice
                widget spinbox
            question privateDebt
                widget spinbox
            question valueResidue
            default money {
                width: 400
                font: "Arial"
                fontsize: 14
                color: #999999
                widget spinbox
            }
        }
    }
    default boolean widget radio("Yes", "No")
}
```

Language for styling questionnaires

Buying

Did you buy a house in 2010?

Did you enter a loan?

Previous

Next

Loaning

Selling

Did you sell a house in 2010?

Yes

No

You sold a house

What was the selling price?

232323



Private debts for the sold house:

12323



Value residue:

220000



Previous

Next

Part I: QL

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Type checker/Wellformedness checker
- Expression evaluator
- Renderer as GUI
(interpreter! Not a compiler)

Part 2: QLS

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Wellformedness checker WRT QL program
- Renderer as stylized GUI
- Challenge: modular implementation
- QL should work standalone (w/o QLS)

No server-side web apps!



- server/client distinction is a distraction
- essentially code generation all over the place

Programming language

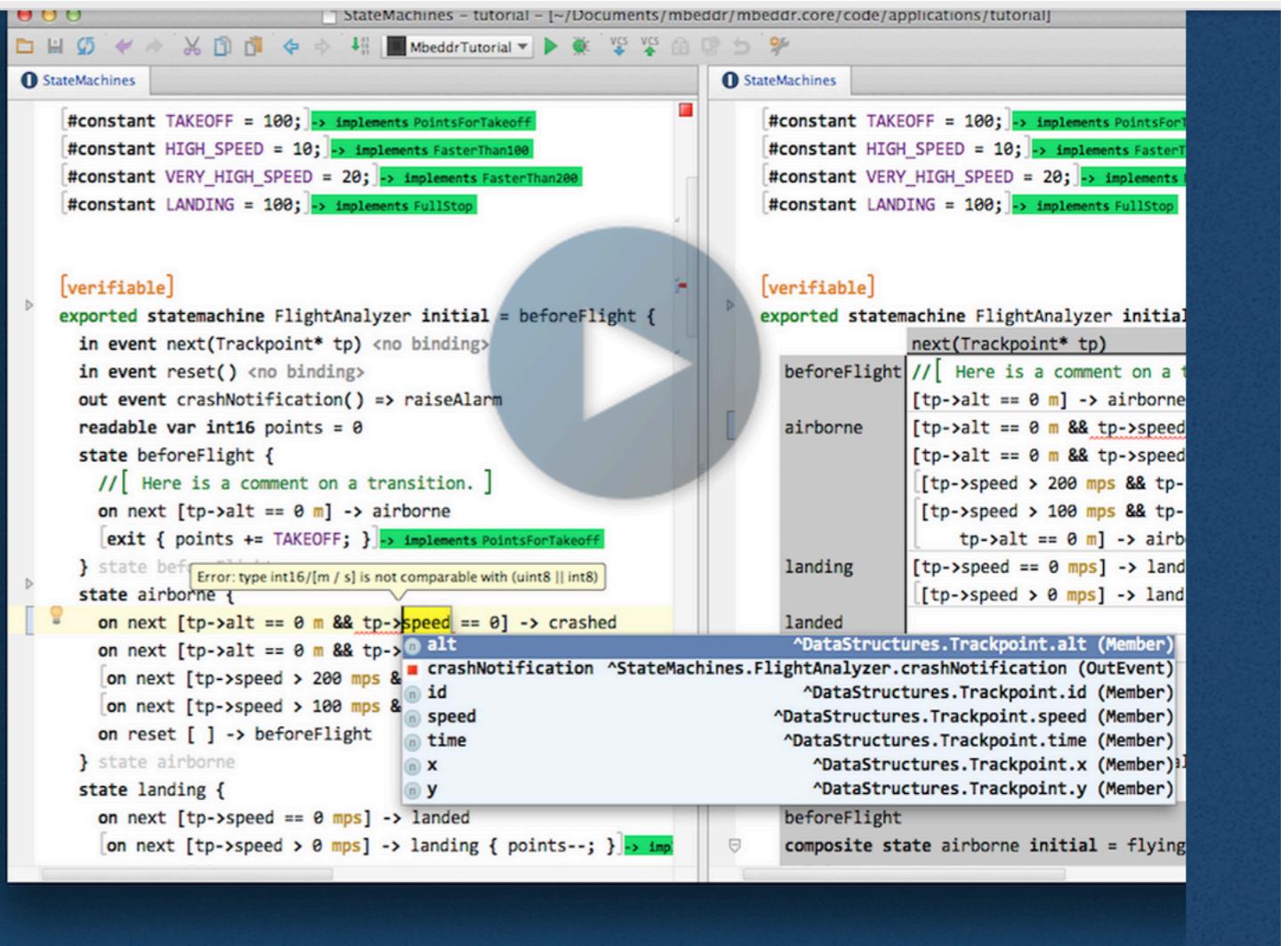
- Java, C#, Javascript, Typescript, Haskell, Scala, Clojure, Erlang, Smalltalk/Pharo, Ruby, Python, Go, Dart, Swift, Objective-C, F#, Rust, ...
- Java: you may want to use one of the provided parsing skeletons for expressions in QL
 - *Rats!*, Jacc, ANTLR

Honor's track: mbeddr in Rascal

**ENGINEERING
THE FUTURE OF
EMBEDDED
SOFTWARE**

Boosting productivity and quality by using extensible DSLs, flexible notations and integrated verification tools.

 Download



The screenshot shows the Mbeddr IDE interface with two panes. The left pane displays a Rascal script for a 'FlightAnalyzer' state machine. The right pane shows the state transition graph corresponding to the code. A large play button is overlaid on the graph. The Rascal code includes constants for speeds and altitudes, and defines states like 'beforeFlight', 'airborne', 'landing', and 'landed'. It also includes event handlers for 'next', 'reset', and 'crashNotification' events, along with readability and implementability annotations.

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop

[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight {
    in event next(Trackpoint* tp) <no binding>
    in event reset() <no binding>
    out event crashNotification() => raiseAlarm
    readable var int16 points = 0
    state beforeFlight {
        // [ Here is a comment on a transition. ]
        on next [tp->alt == 0 m] -> airborne
        [exit { points += TAKEOFF; }] -> implements PointsForTakeoff
    } state beforeFlight
    state airborne {
        on next [tp->alt == 0 m && tp->speed == 0] -> crashed
        on next [tp->alt == 0 m && tp->speed > 200 mps] -> altitude
        [on next [tp->speed > 200 mps && tp->alt == 0 m] -> airborne]
        [on next [tp->speed > 100 mps && tp->alt == 0 m] -> airspeed]
        on reset [ ] -> beforeFlight
    } state airborne
    state landing {
        on next [tp->speed == 0 mps] -> landed
        [on next [tp->speed > 0 mps] -> landing { points--; }] -> implement Landing
    } state landing
}
```

<http://mbeddr.com/>

mbeddr

- C + Language extensions
 - unit tests, modules, state machines, requirements traces, variability, ...
- Built using Jetbrains Meta Programming System (MPS)
- You will implement number of extensions in Rascal
- Modularly!!!!

Github

- Assignment to be completed *in teams of 2*
 - (except honor's track)
- <http://github.com/software-engineering-amsterdam/many-ql/>
- Use of this repository is required!
- Commit often!

“Hour of code”

- During lab sessions (Mon 14:00/Tue 9:15)
- Convene in single room
- Teams (2 per session) present their code.
- No slides. Code.
- Constructive feedback and criticism.
- Let's help each other.

Grading of lab assignments

Simplicity

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



Self pre-assessment

- Before grading moments:
- You fill out an online questionnaire
- This will help us
 - navigate the code
 - ask the right questions

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*
 - incentive 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 less important than personal improvement is

Passing this course

- Be present at all lectures
- Be present during lab sessions
- Pass the exam with grade > 5.5
- Pass lab assignment with grade > 5.5
- Final grade: average of lab and exam
- NOTE: both grades need to be > 5.5

week	7	8	9	10	11	12	13
Mon 9:15-11	lecture	lecture	lecture	lecture	lecture	lecture	Final grading
Mon 11-14	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
Mon 14-15	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Final grading
Mon 15-17	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
Tue 9:15-10:15	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Final grading
Tue 10:30-17:00	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
			First grading	First grading		Final grading	Exam on Thursday

Concluding

- All information is on Github
- Primary contact = storm@cwi.nl
- Please follow @SoftwCons

What's next

- For the coming days
 - make up your mind on language
 - start checking out parser generators
 - start coding!
- NB @tvdstorm will be back next week.
Vadim is here to help!