PV168

Course Summary

What have you experienced?

- Let's try to remember what has happened in PV168
- What do you think about the real purpose of all that?

Programming is hard

- It's not that difficult to fulfill
  - compiler's rules
  - current requirements
- However, it's a completely different game
  - when other people are involved
  - and requirements evolve over time

Essential vs. Accidental Complexity

- Essential complexity is integral part of the problem
  - cannot be eliminated completely
  - can only be minimized
- Accidental complexity is what **we** add to the problem
  - by using inefficient tools
  - by using inefficient methodologies
  - by obscuring the intent in the code

Naming things is hard

- Good names reveal the intent (semantics)
  - not the data type used (e.g. `listEvent`)
  - not the physical position (e.g. `northPanel`)
- Good names lie in the problem domain
  - not in the solution domain (Computer Science "jargon")
- Naming things remain hard
  - even for masters
- Naming is an incremental process

Structure matters

- The same problem can typically be represented in many different forms
- Some of them are better than the others
- Deciphering bad structure is not what you want to do

Interactions with PMs are tough

- Nothing is as evident/unambiguous as demos
  - misunderstanding are clearly identifiable
  - corrective actions can be planned
- It's better to demo
  - less functionality but actually working
  - stuff you know why it's there
- "When in doubt, leave it out" (*Josh Bloch*)
  - missing things are easy to spot

Functionality vs. Graphics

- Functionality is important to have first
- Graphics is also important but not at first
  - investing into it too early is typically waste
  - developers are seldom good visual designers

Duplicity is the Root of all Evil

- Similarity is not necessarily duplicity
- Magic numbers and string literals are almost always evil
  - in production code you have to use constants
  - however, in tests the situation is different
- Repeated code blocks of the same semantics are always evil

Refactoring is hard

- We need to have solid test suite in order to do it safely
- Keeping increments small is tough and non-intuitive
- What you typically do is **Rewriting** instead

Team work is hard

- You've typically optimized the amount of code produced
  - Bus factor remains low
  - In PV168 the thing to optimize is the amount of stuff learned
- In the industry you have to optimize the amount of functionality
  - For really difficult parts it's wise to go in tandem
  - "Doctors never operate alone" (*Michael Feathers*)
- Camera turned on is a must for on-line cooperation

Copy & Paste can be dangerous

- Unless you know what you're doing
- When using sources such as Stack Overflow, Geeks for Geeks
  - you need to understand the difference in context
  - adapt the proposal to your specific case
  - and get to know the details to be able to defend the solution

Law of the Instrument

> If the only tool you have is a hammer, it is tempting to treat everything as if it were a nail. (*Abraham Maslow*)

- Especially newly gained tools/techniques tend to be "hammers"
- Not a single thing is ultimately good for everything
  - You always need to think in context

Source Code is Text

- Programming by mouse isn't efficient
  - it's not accurate enough
  - it's not fast enough
  - it's exhausting your mental capacity
- Programming is only a craft
  - not a science neither an art
- Craftsmen have good tools and operate them efficiently
  - keyboard is the ideal tool for programmers
  - IDE is the ideal tool for programmers

Thank you for your hard work!

You've learnt a lot about programming already.