TDD: Discover Implementation by Stepping Small

Vaidas Pilkauskas

Agile Tour, 2016

Who am I?

Server Guild Master at Wix.com

Vilnius Java User Group organizer

Twitter @liucijus

Motivation for this talk

- Rigorous application of Test-Driven Development
- Algorithm implementation without knowing algorithm steps
- Importance of feedback loop (aka “listen to your code”)

Test-Driven Development cycle

Red

Green

Refactor

Roman Numerals

I - 1, II - 2, III - 3,

IV - 4, V - 5, VI - 6, VII - 7, VIII - 8,

IX - 9, X - 10, XI - 11, etc

3497 - MMMCDXCVII

Transformation Priority Premise (TPP)

“As the tests get more specific, the code gets more generic.”

Transformations

1. ({} → nil) no code at all → code that employs nil
2. (nil → constant)
3. (constant → constant+) a simple constant to a more complex constant
4. (constant → scalar) replacing a constant with a variable or an argument
5. (statement → statements) adding more unconditional statements.
6. (unconditional → if) splitting the execution path
7. (scalar → array)
8. (array → container)
9. (statement → tail-recursion)
10. (if → while)
11. (statement → non-tail-recursion)
12. (expression → function) replacing an expression with a function or algorithm
13. (variable → assignment) replacing the value of a variable.
14. (case) adding a case (or else) to an existing switch or if

Q&A