## Main take-aways

- Practice, practice and once more – practice
- Divide and conquer
- Programming is much more than writing code

## Further resources

### 1. Testing

- „Pragmatic Unit Testing”, Andy Hunt, Dave Thomas
- „JUnit in Action”, Vincent Massol, Ted Husted
- „Lessons Learned in Software Testing”, Cem Kaner, James Bach, Bret Pettichord
- “JUnit Recipes”, J.B. Rainsberger

### 2. Clean Code

- “Clean Code: A handbook of Agile Software Craftsmanship”, Robert C. Martin
- “Agile Software Development: Principles, Patterns, and Practices”, Robert C. Martin
- “Holub on Patterns: Learning Design Patterns by Looking at Code”, Alan Holub

### 3. Refactoring

- “Refactoring: Improving the Design of Existing Code”, Martin Fowler
- “Working effectively with legacy code”, Michael Feathers
- “Refactoring to Patterns”, Joshua Kierevsky

### 4. TDD

- “Test-Driven Development By Example”, Kent Beck
- “Test-Driven Development: a practical guide”, David Astels
- “Behaviour Driven Development: in Ruby with RSpec”, David Chelimsky, Dave Astels
- “Test Driven: Practical TDD and Acceptance TDD for Java Developers”, Lasse Koskela
- “Growing Object-Oriented Software Guided by Tests”, Steve Freeman, Nat Pryce

### 5. Other resources/articles

- “Programming as Theory Building”, Peter Naur,
- “Transformation Priority Premise”, Bob Martin (google it)
- “Thinking, Fast and Slow”, Daniel Kahneman