

Pyramid of Refactoring





About me

Włodek Krakowski

Technical and Organizational Trainer

www.refactoring.pl



@wlodekkr



A line-art illustration of a space scene. At the top center is a planet with a ring and three small circles on its surface. Surrounding the planet are five stars of varying sizes. Below the planet is a small rocket ship with a flame trail. The entire scene is set against a background of concentric circles.

Refactoring Workshops



1

What is Refactoring

Basics

Refactoring Definition

(verb) to restructure software by applying a series of refactorings without changing its observable behaviour

Martin Fowler – Refactoring (2018)



Refactoring Definition

(noun) A change made to the internal structure of the software to make it easier to understand and cheaper to modify without changing its observable behaviour

Martin Fowler – Refactoring (2018)

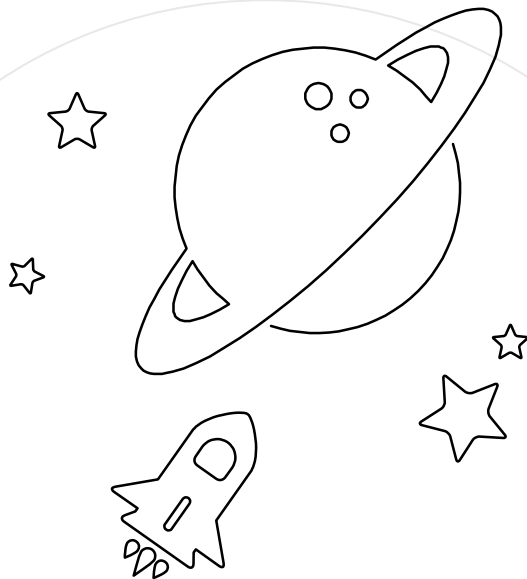




2

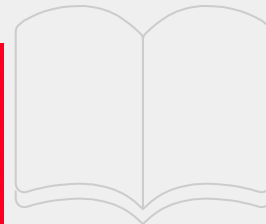
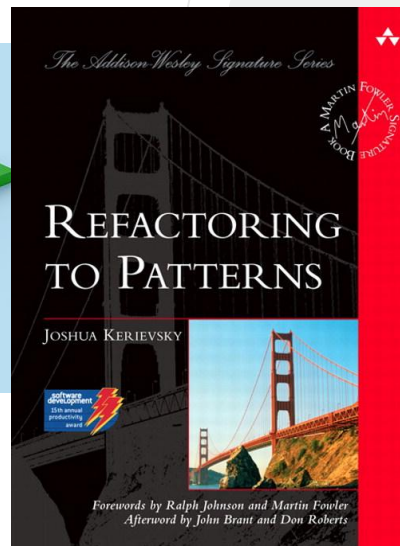
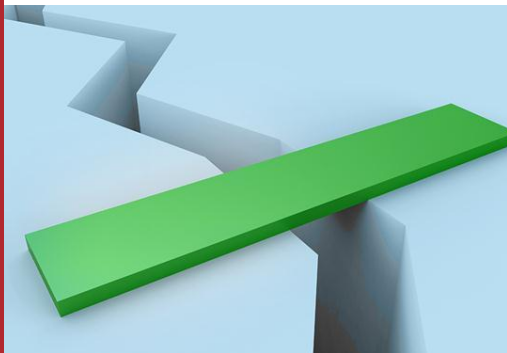
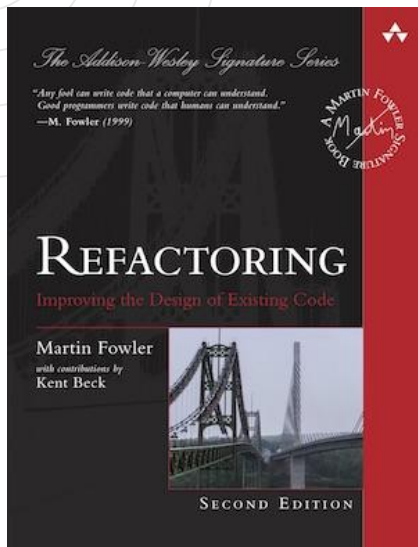
Noticing the Pyramid

My story



100+ Workshops Afterthoughts

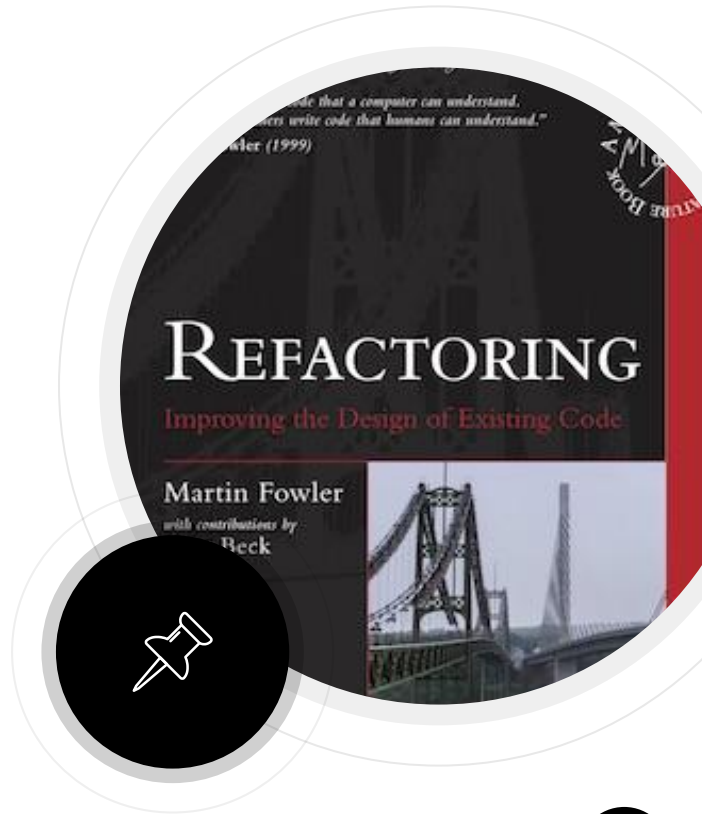
Space between two main books...



Refactoring by Martin Fowler

- Great catalogue of refactorings
- One big sample at the beginning
- Java/Javascript

This book is a great summary but it is **like**
encyklopedia / dictionary to me

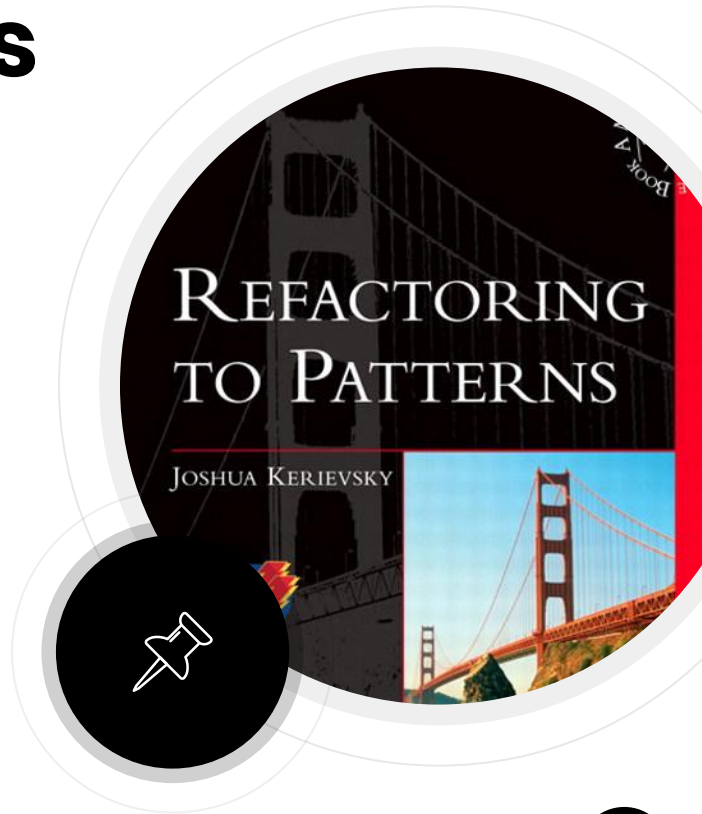


Refactoring to Patterns by Joshua Kerievsky

- Expands the subject a lot
- Contains lots of smaller samples
- Requires reading a few times...

But each sample is **already prepared** to refactoring towards given design pattern

This is rarely the case in legacy code



Working with Legacy Code by Michael Feathers

- Different perspective
- Also a kind of encyklopedia
- Allows to find a starting point !!!

Does it contain a bigger vision among lots of **useful & invaluable fixes**?



Basics... Arrangement... Order...



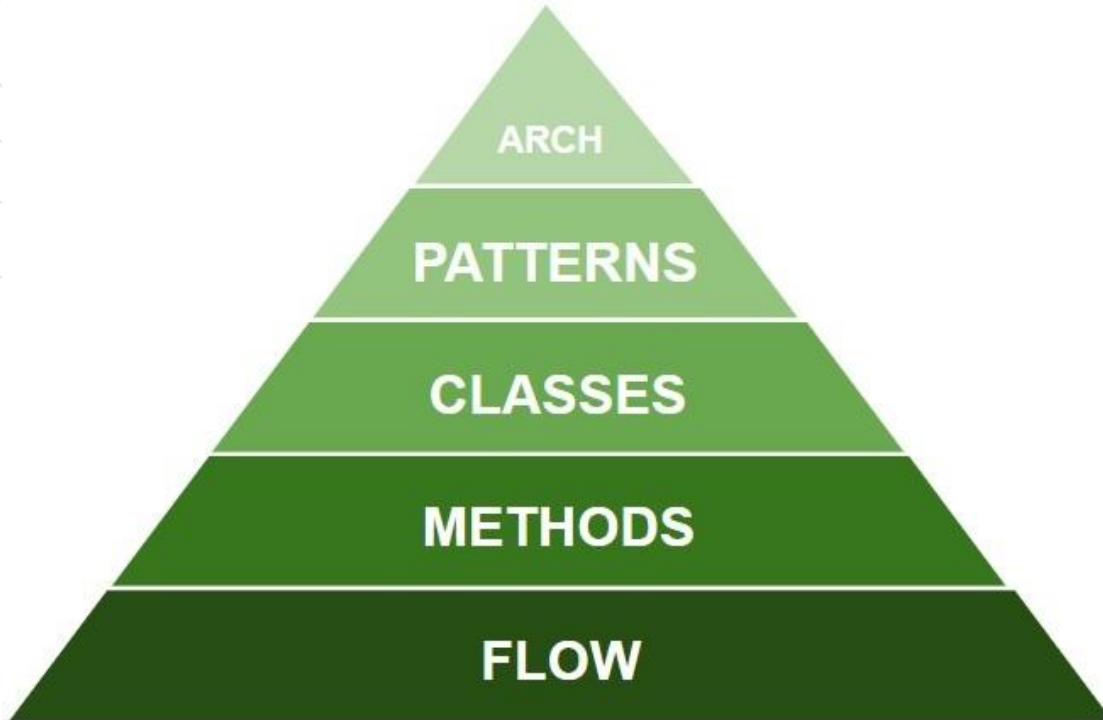


3

Pyramid of refactoring

Some theory at the beginning

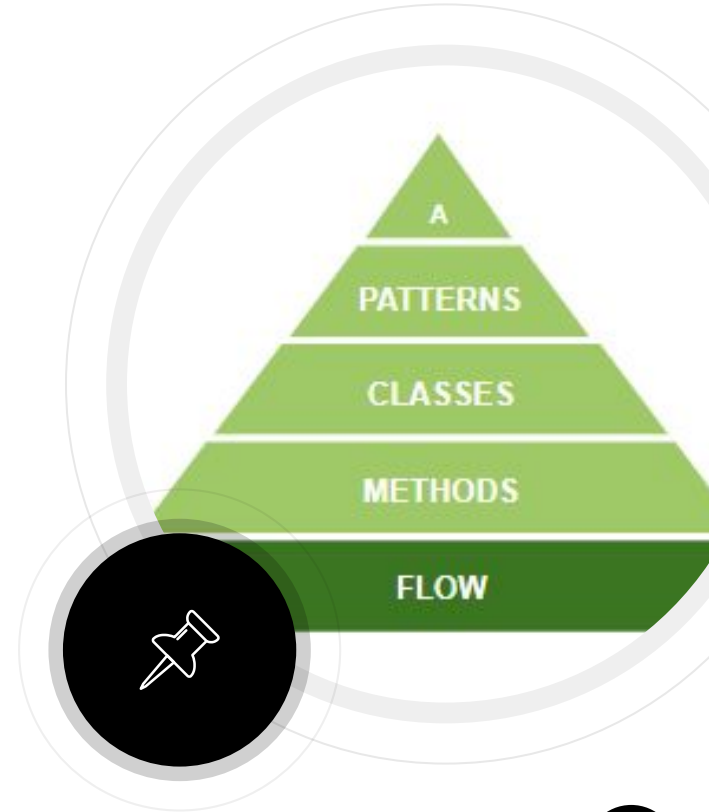
So simple?



Flow

- Nested Conditions
- Nested Loops
- Many Local Variables
- Ambiguous Names
- Single Exit Points

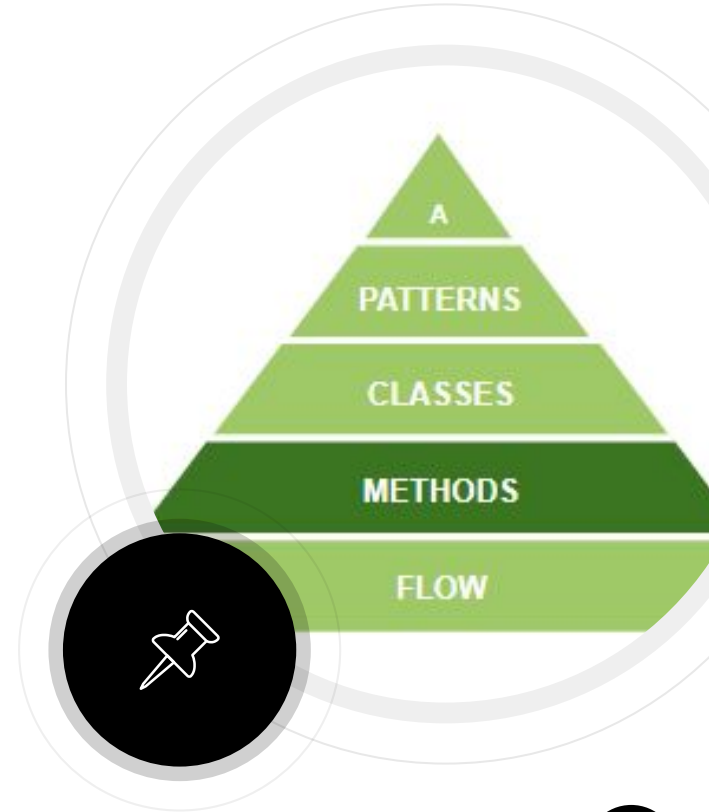
Can you read your code like a good book from the top and understand it quickly?



Methods

- Levels of Abstraction
- Extract / Remove Parameter
- Cohesion

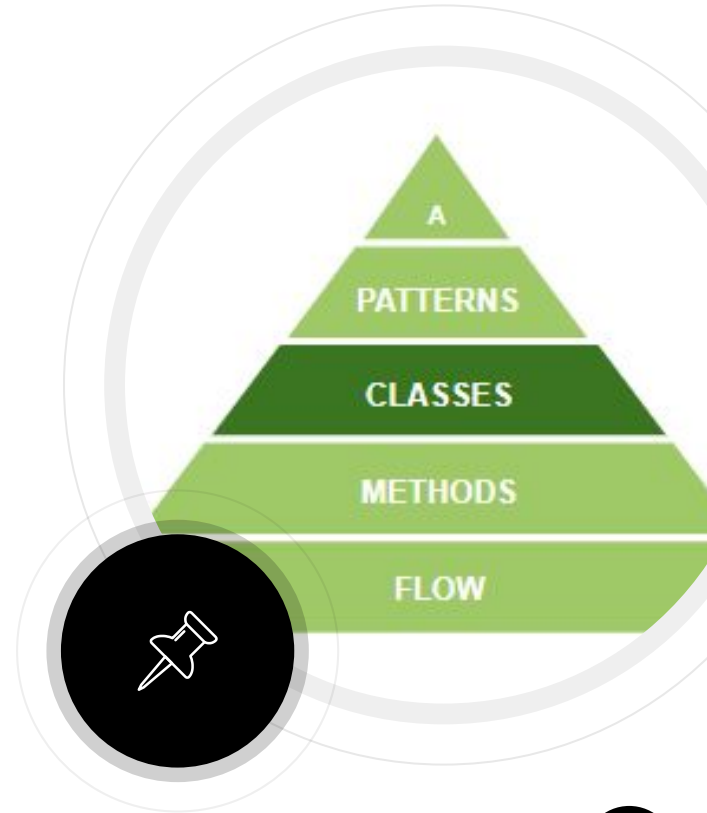
Can you understand quickly what a method does at **single level of abstraction**?



Classes

- Extract Delegate
- Extract Base Class
- Extract Subclass
- Extract field/constant

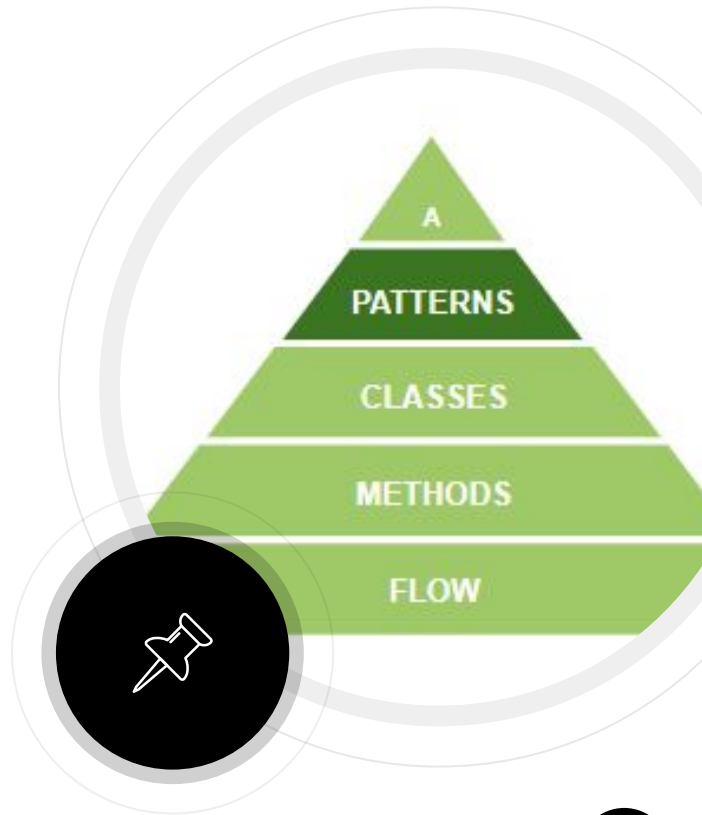
Do your classes have **distinct areas of responsibility**?



Patterns

- Abstractions
- Interfaces

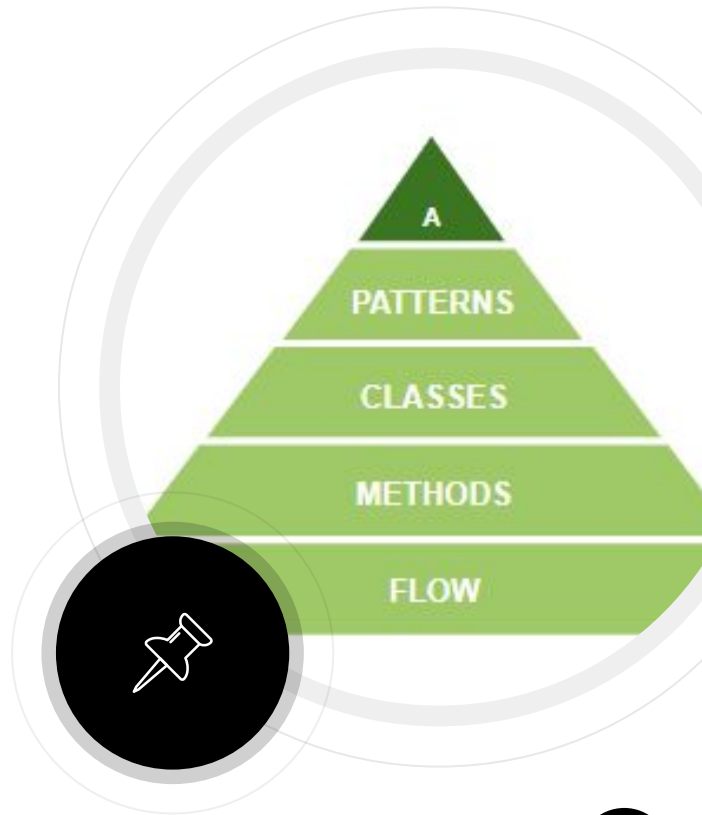
Dependencies defined as **contract instead of implementation details knowledge**



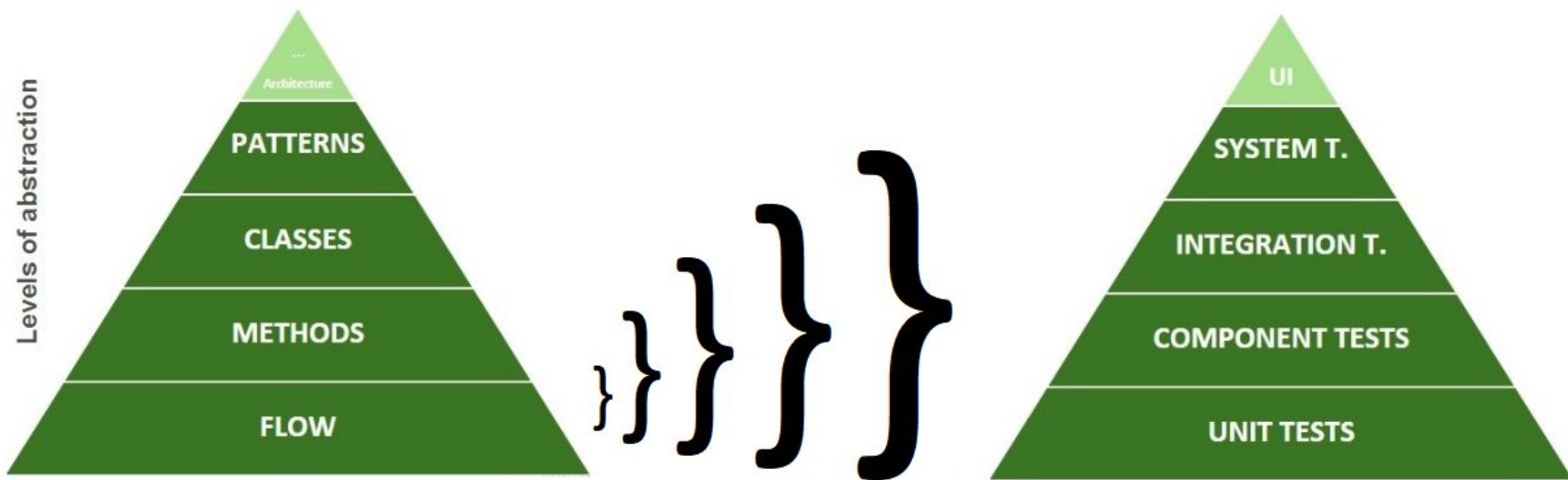
Architectures

- Packages
- Modules
- (Micro) Services

Can your **architecture grow and scale** easily by adding new or dividing existing components?



Testing and Refactoring Twins





4

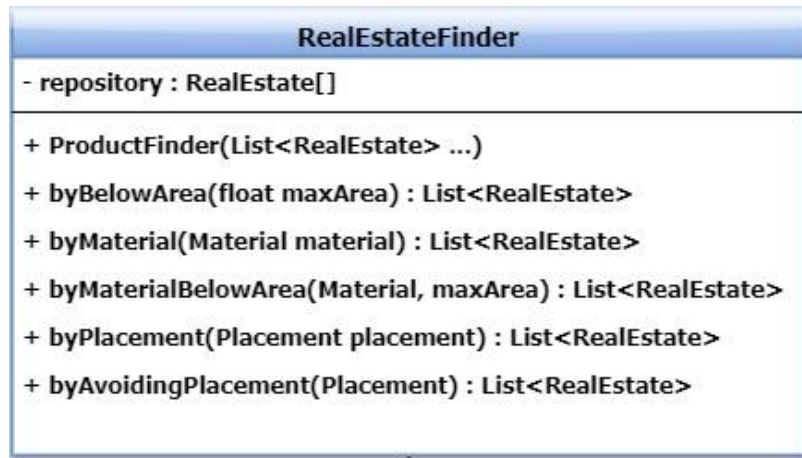
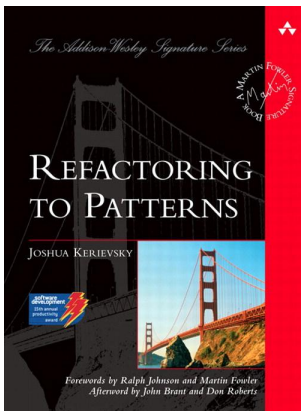
Live Refactoring

**You need to experience in order to
understand**

Real Estates Catalogue



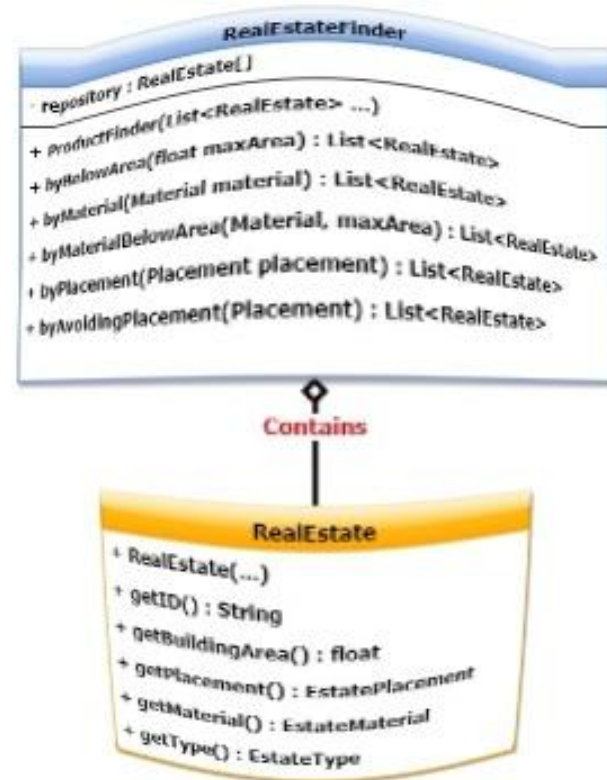
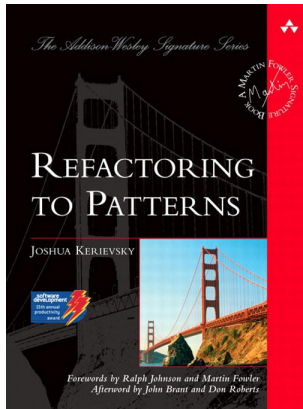
Initial Project...



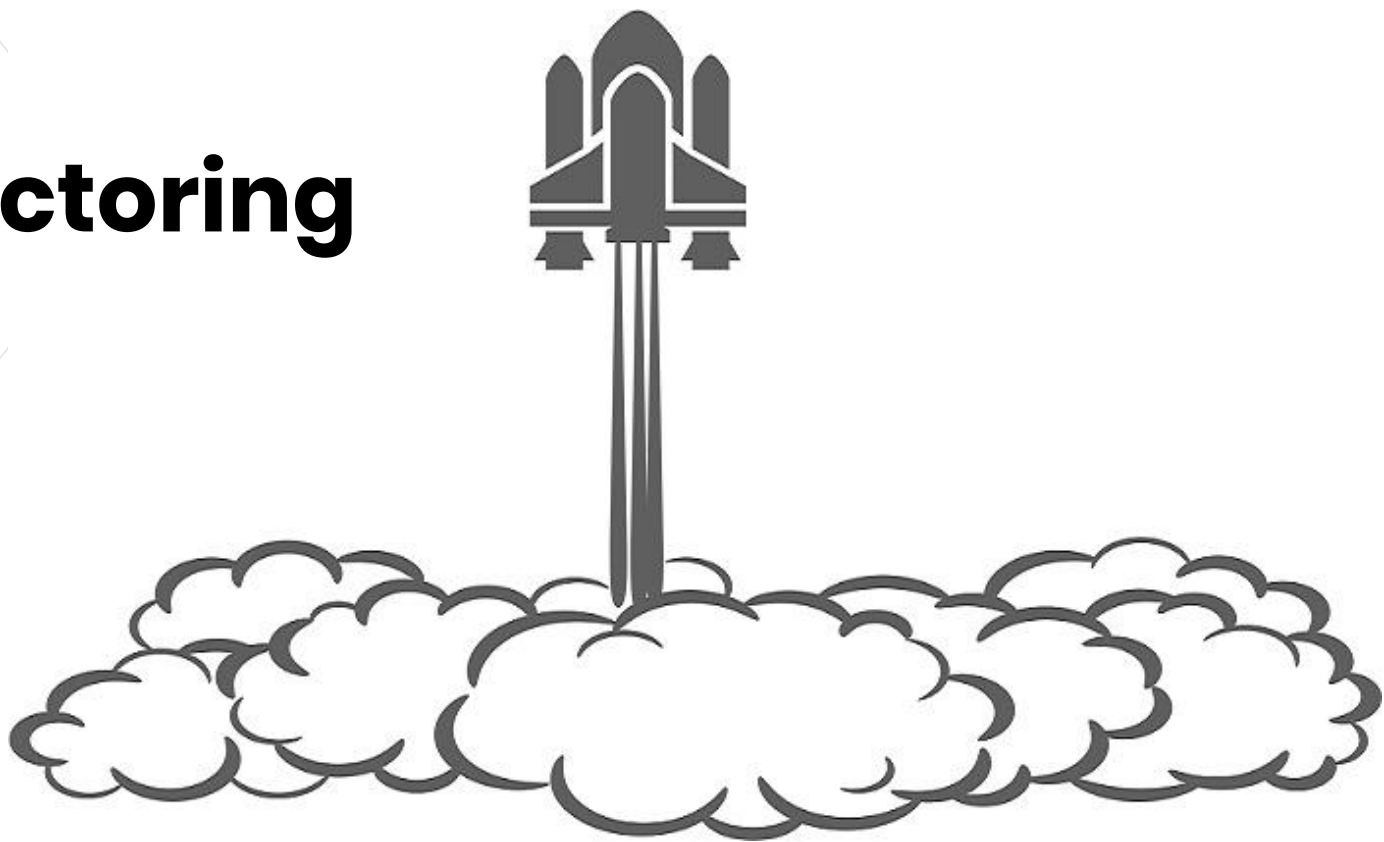
Contains



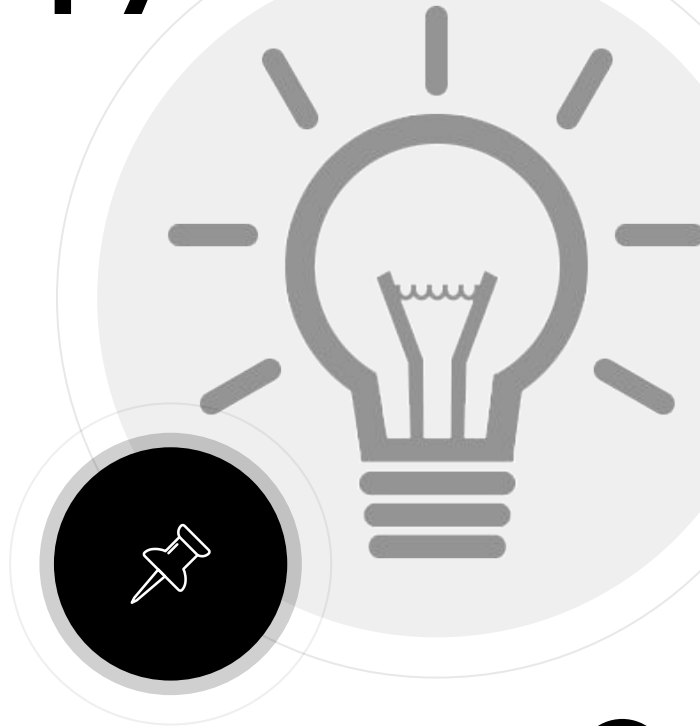
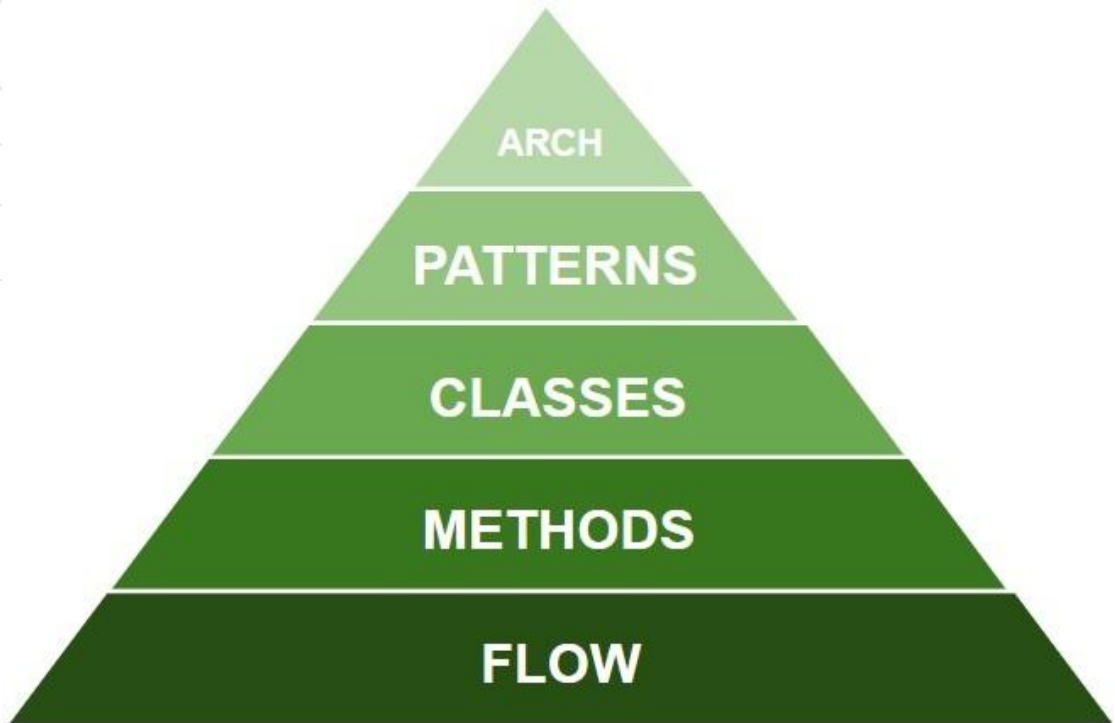
It's not me, It's the team...



Live Refactoring



We are climbing up the pyramid!



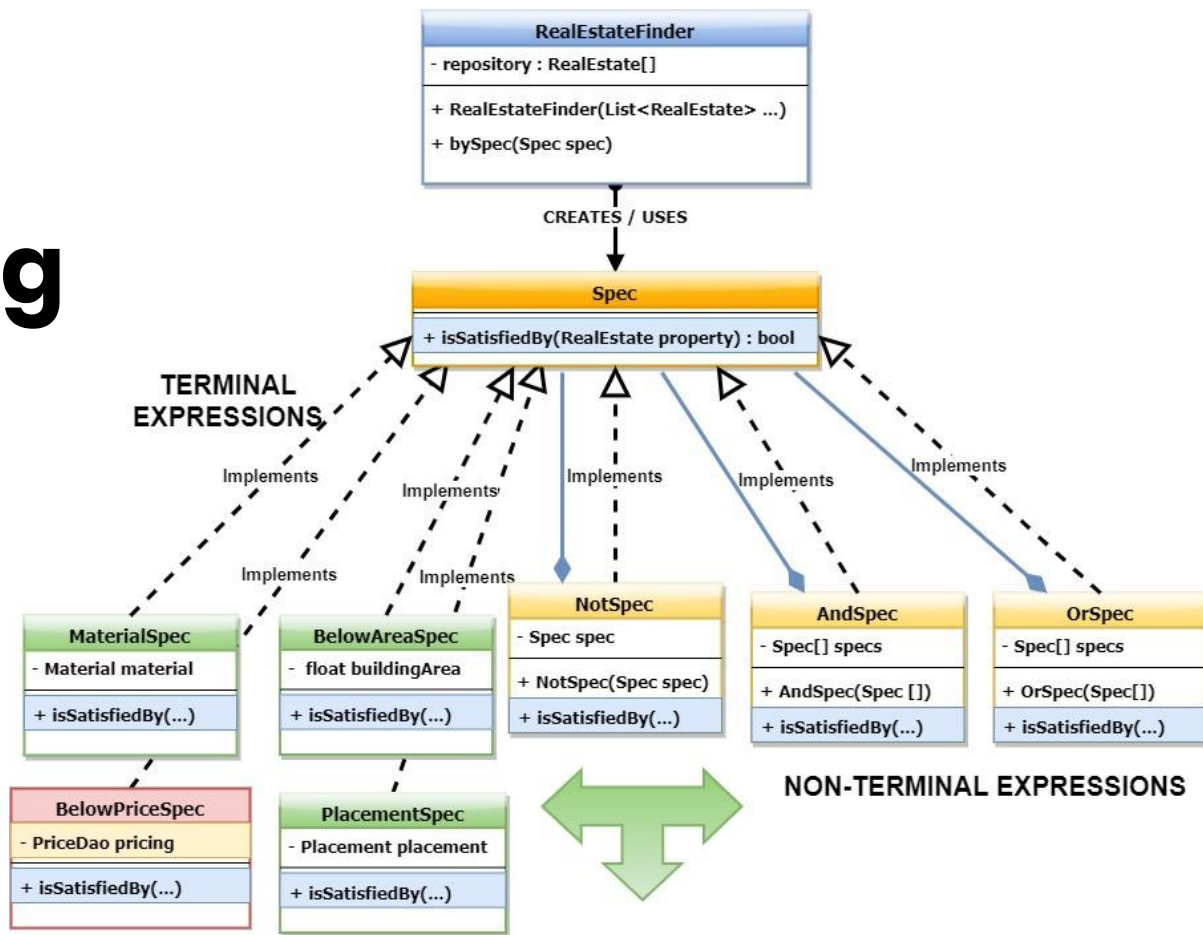
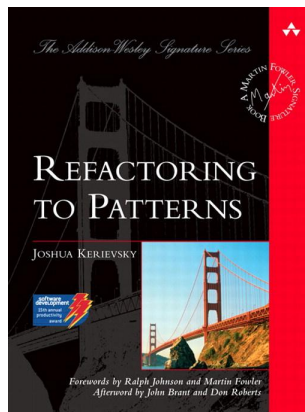
Lots of ways to refactor...

Specs classes achieved

- Create Class (manually)
- Extract Delegate / Class
- Extract Parameter Object



Live Refactoring





5

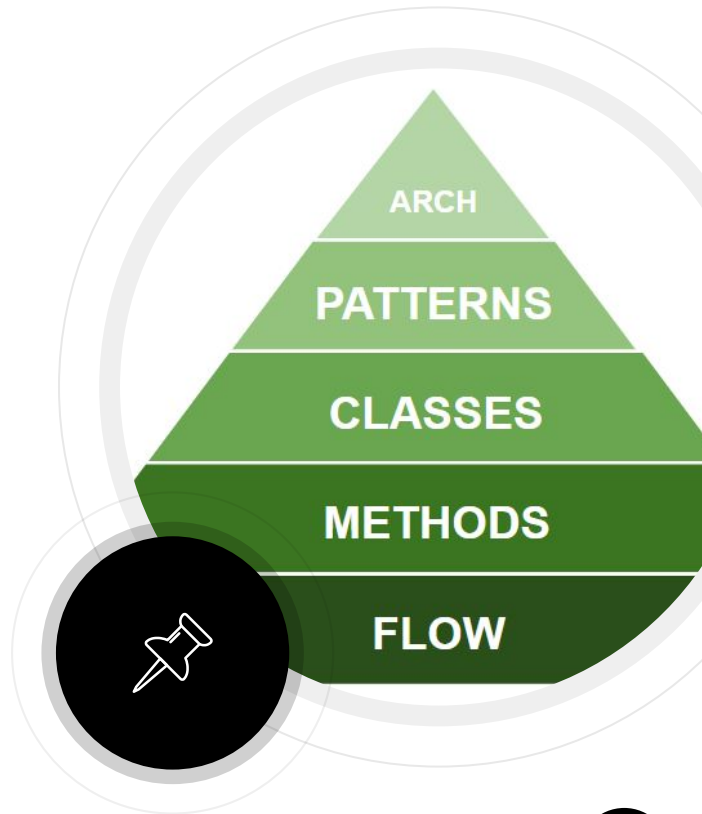
S.O.L.I.D. in Pyramid

Basics again...

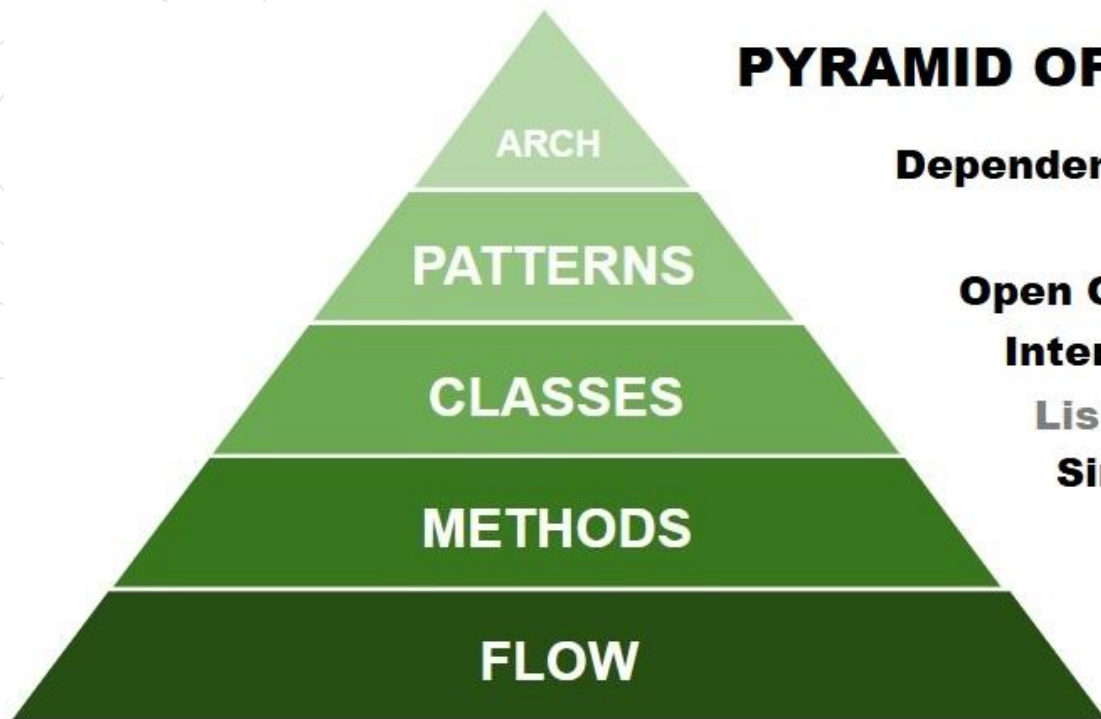


S.O.L.I.D. Refactorings

- **Single Responsibility Principle**
 - BelowAreaSpec, MaterialSpec, ...
- **Interface Segregation Principle**
 - Spec
- **Open Closed Principle**
 - ProductFinder.bySpec(Spec spec)
- **Dependency Inversion Principle**
 - pl.refactoring.search.ProductFinder
 - pl.refactoring.search.Spec
 - pl.refactoring.search.spec.ColorSpec



PYRAMID OF REFACTORING



Dependency Inversion Principle

Open Closed Principle

Interface Segregation Principle

Liskov Substitution Principle

Single Responsibility Principle



6

Make it happen

Knowledge is the beginning...

Share new experience

Emotions come first after each

- Workshop
- Conference
- Meeting
- New Experience

... when you've learned something new



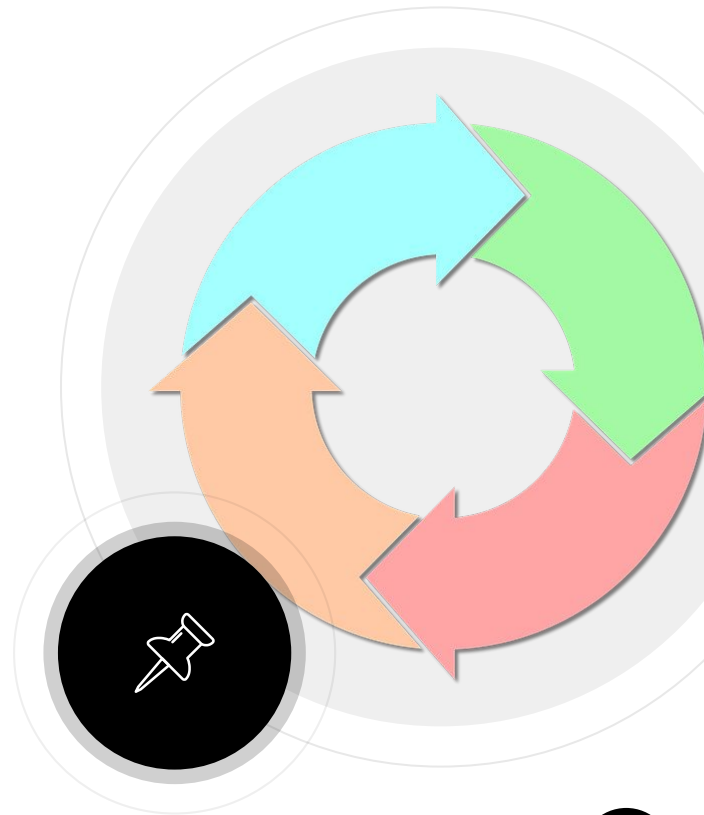
Get new skills

Knowledge and skills are needed to keep the emotions

- **Review** current skills
- Make a learning **plan**
- **Introduce** new skills step by step



Emotions and Mind



Trigger / Enable refactoring








Code review opportunities

- Readability
- Testability
- Extendibility
- Design

Teamwork – each team member has equals rights to teach and learn



Visualize Quality Activities

To Do	Dev	Code Review	Rework / Refactoring	Testing / Acceptance	Done
					
			 		





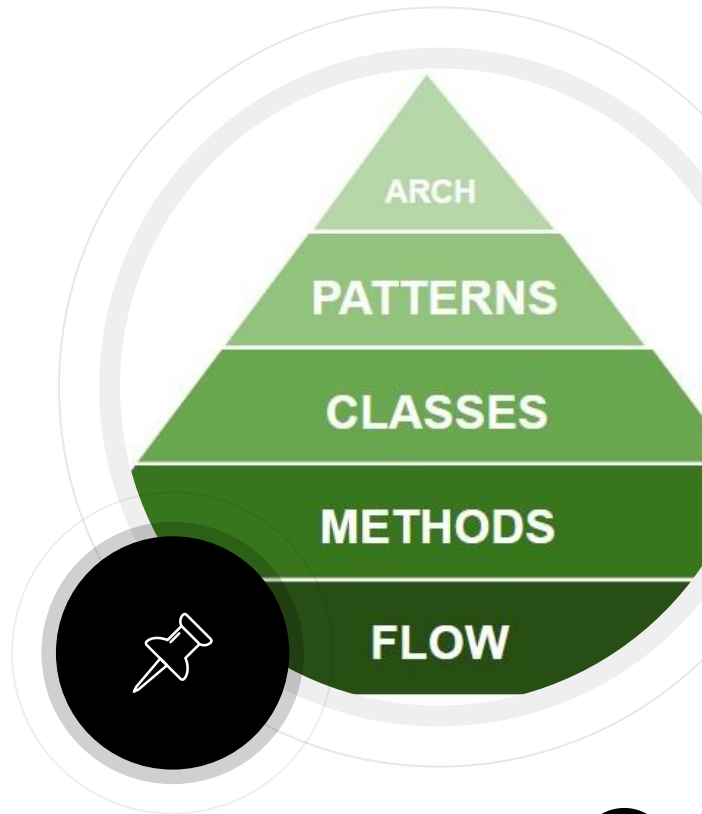
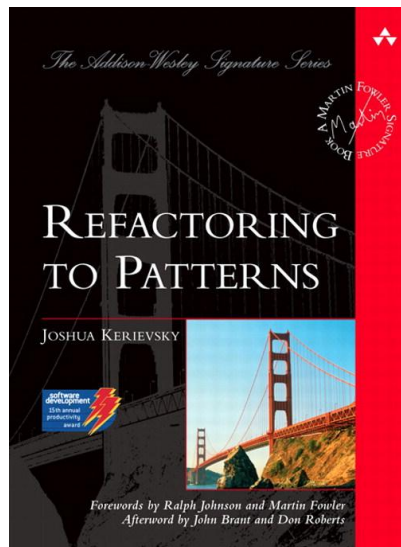
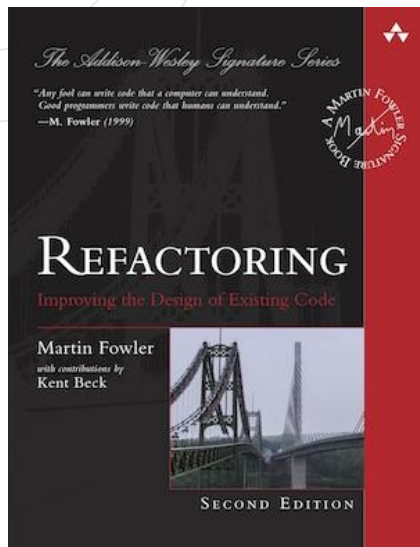
7

Let's summarize

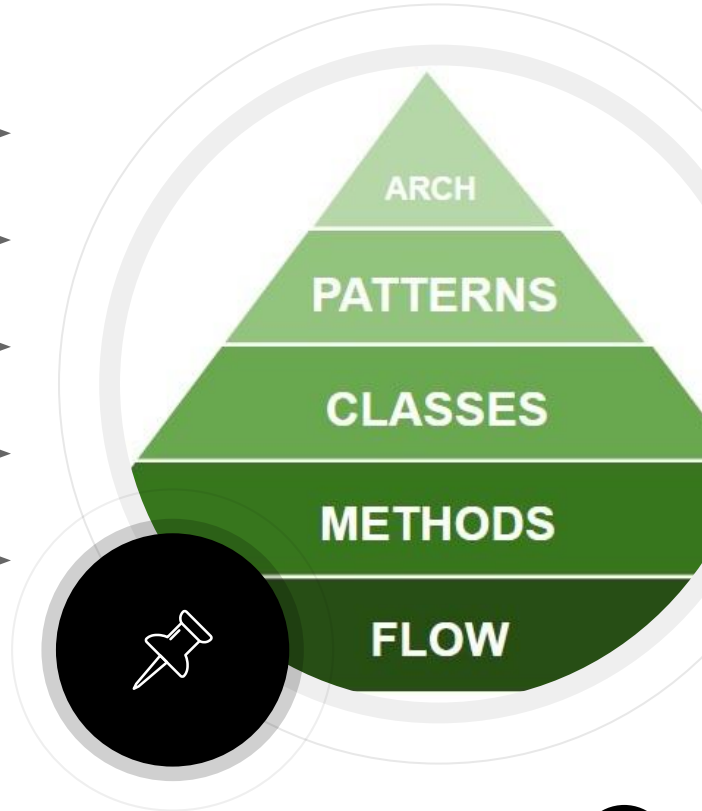
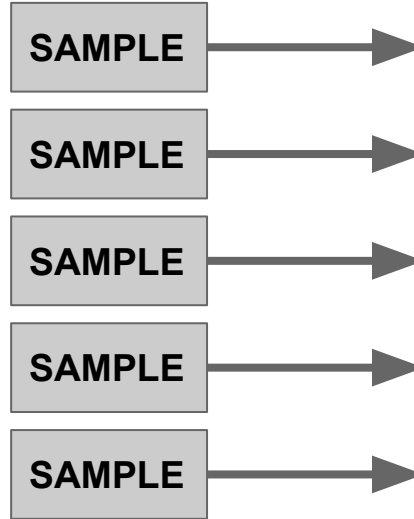
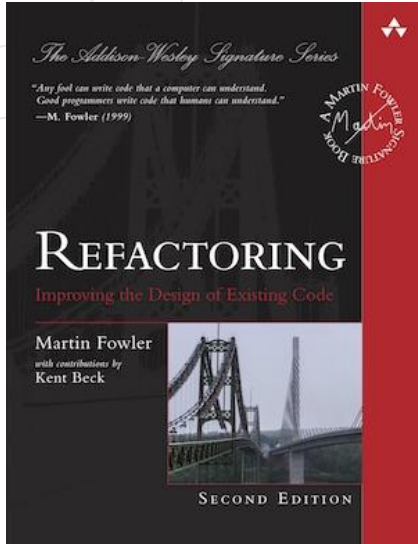
What is the gap and its fulfillment?



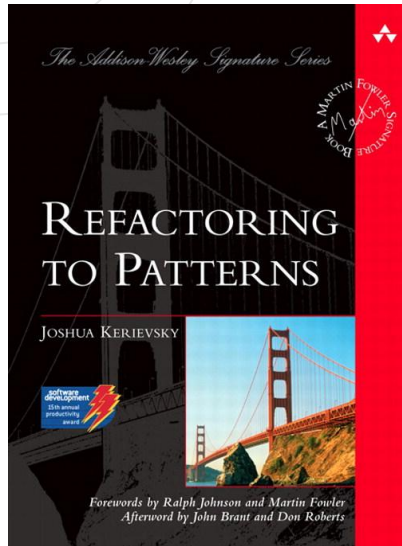
Pyramid is between the books



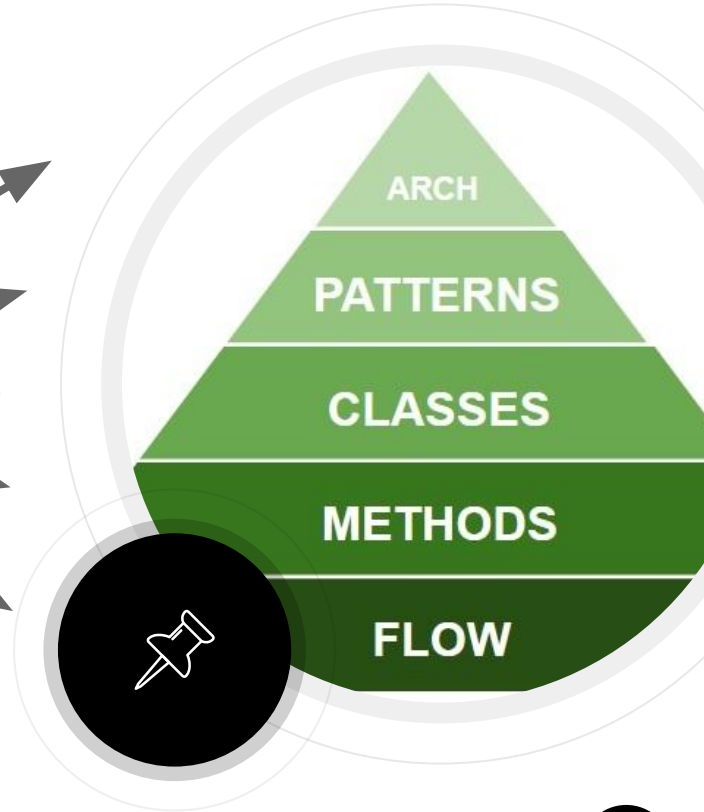
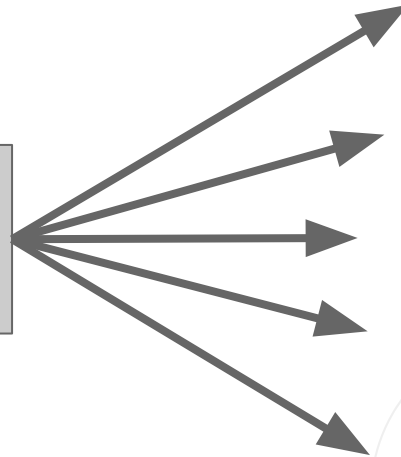
Each Sample Placement



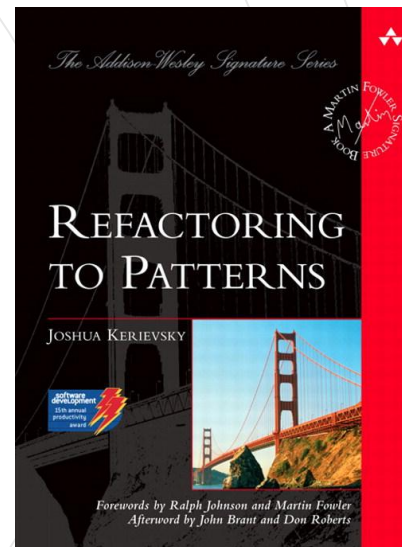
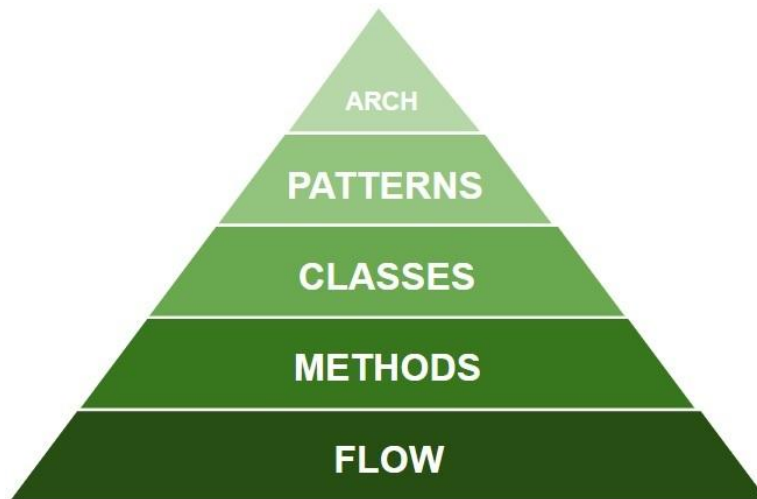
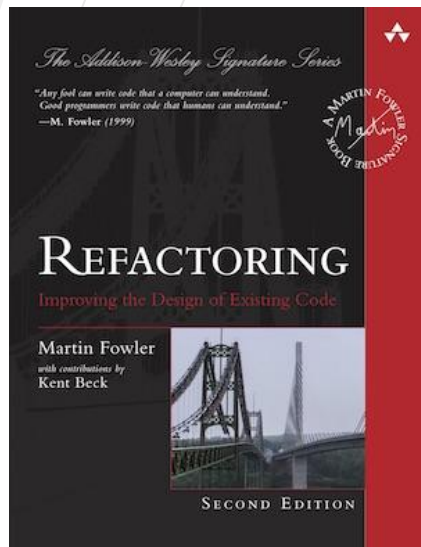
Each Sample Embracement



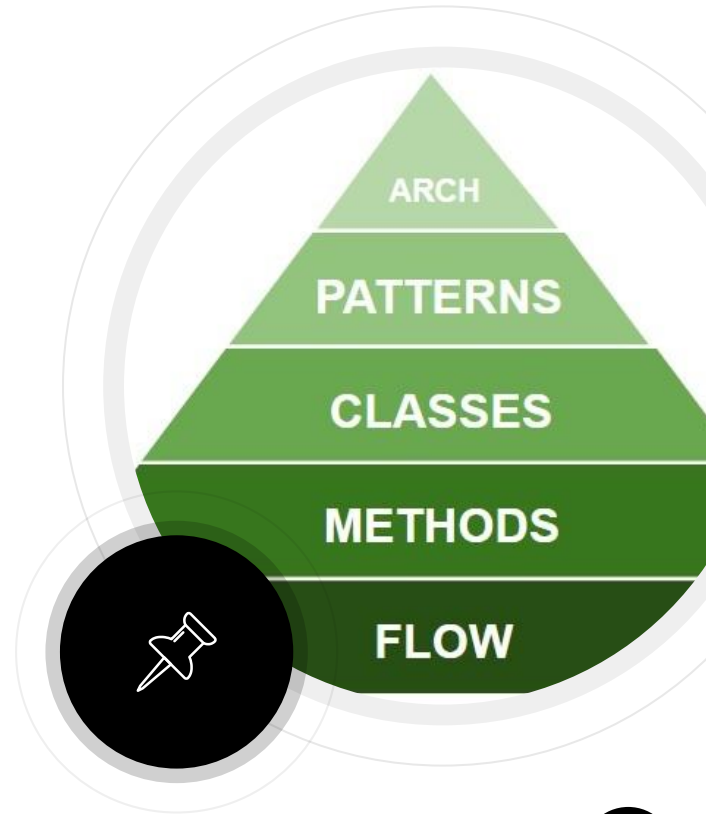
**EACH
BIG
SAMPLE**



Not named explicitly...



... but joining core books!





Thanks!

www.refactoring.pl

- Blog
- IT Trainings
- Talks



@wlodekkkr



Włodek Krakowski