BO·TT·EGA
IT minds

Training program:

Effective Refactoring to Patterns

Info:

Name:	**Effective Refactoring to Patterns**
Code:	**ref-patterns**
Category:	Working with legacy code
Target audience:	developers
Duration:	3 days
Format:	20% lecture, 80% ćwiczenia

Effective Refactoring is refactoring that indeed takes place and brings results.

It is the one that brings visible effects to the well-being of programmers when they see its results as a more readable and understandable code. And this in turn will allow them to continue development of their projects, because they will be able to keep the balance between the business value for the client and the technical quality.

Effectiveness of refactoring depends on lots of areas. Contrary to appearances, the technical skills of programmers themselves are not enough, although they are necessary. For refactoring to have a permanent place, we need to introduce it to our consciousness, introduce it as a habit to the teams and make it very visible part of the software development process. This way it can’t be overlooked.

In order to make it happen in addition to technical skills there is a need to explore other areas surrounding the world of programmers. Among them the most important is to take care of your personal effectiveness and work efficiency by building awareness of how the results of work depend on the functioning of the entire team.

Introducing all the improvements starts with the awareness of how the organization of work is reflected in the quality of the code that the team provides. As a result, we will be able to have a positive impact on the quality of the code by applying changes on the organizational side that will enable continuous refactoring.

The above inspirations will be presented on the basis of such concepts like Stephen Covey's "Seven Habits of Highly Effective People", "5 Dysfunctions of a Team” by Patrick Lencioni or 6 sources of influence developed from the VitalSmarts laboratories in Utah.

This practical and theoretical training shows how to approach refactoring in the smallest possible steps, making it our daily practice, in which we will grow every day. Through small steps that will become our habit, we will perform increasingly complex refactorings until we start to discover design patterns.

Design patterns are something common. It is worth to use them as they are reliable because they are already exercised and proven. Most likely, they are very likely to deliver the expected results by providing code that is understandable. This way it will be easier to extend it by adding new functionality.

The training is based on the concept of the refactoring pyramid. It allows you to start with the simplest refactoring and then gradually move to the more complex ones.

- Simplifying the logic of algorithms
- Extracting smaller methods
- Extracting smaller classes
- Notifying emerging design patterns
- Cleaning up the architecture

The training focuses mainly on working with source code using built-in automated refactorings within tools such as IntelliJ or Eclipse (when using Eclipse the scope will be smaller). The source code is written in Java. The trainer performs all refactoring transformations live, after which the participants can experience the same by performing the same code transformations on their laptops. The remaining 20% of the time is devoted to presentations and discussions about the possibilities of introducing refactoring as a daily teamwork habit.

This training is not about architectures. Different teams may have different preferences regarding architectural solutions. The aim of the training is to learn how to easily and quickly change the architecture of the code from one approach to another.

It's all about the content.

- Refactoring perceived in the context of the habits of effective people
- Development process in the context of teamwork dysfunctions
- Practical and non-book only usage of design patterns

BO·TT·EGA
IT minds

Training program

1. Day 1 - Keynote: "7 Habits of Highly Effective People"

1.1. Part 1

1.1.1. Getting to know the "everyday" piece of code which you can quickly understand and "extend"

1.1.2. Showing how quickly this "fast development" by copy-paste approach will reduce code readability

1.1.3. Playing with code refactorings - from the simplest transformations as extracting variables, methods, extracting parameters from these methods followed by moving these methods into existing or new classes and ending up with hiding classes behind their abstractions.

1.1.4. Presentation of the Refactoring Pyramid, how this concept was used in the order of the above applied refactorings.

1.1.5. Learning embedded automatic refactorings in the IntelliJ or Eclipse environment

1.1.6. Application of SOLID principles as a basis for dividing architecture into smaller parts

1.2. Part 2

1.2.1. Getting familiar with the extended code from part 1 where new functionality was added but no refactorings has been performed at all

1.2.2. Presentation of requirement for additional functionality to be added

1.2.3. Quick brainstorming that we can do it quickly with decreasing readability of the code

1.2.4. Performing the "extraction of extension points" as a result of "refactoring journey" using the concept of the Refactoring Pyramid

1.2.5. Providing new functionality as a new implementation of the "extension point" using TDD

2. Day 2 - keynote: "5 Dysfunctions of a Team"

2.1. Refactoring transformations performed in small steps based on the concept of refactoring pyramid that are completed with the implementation of the following design patterns

2.1.1. Interpreter

2.1.2. Chain of Responsibility

BOTTEGA IT minds

2.1.3. Composite

2.1.4. Factory Method / Abstract Factory

2.1.5. State

3. Day 3 - keynote: "6 Sources of Influence"

3.1. Refactoring transformations performed in small steps based on the concept of refactoring pyramid that are completed with the implementation of the following design patterns

3.2. (Fluent) Builder

3.3. Proxy

3.4. Template

3.5. Bridge

3.6. Command

3.7. Adapter