Training:

**Design Patterns and effective Object Oriented techniques for developers of the business applications**

General information

- **Name:** Design Patterns and effective Object Oriented techniques for developers of the business applications
- **Code:** Patterns Biz
- **Category:** Software engineering
- **Target audience:** Developers, designers and architects of business applications
- **Duration:** 3 days
- **Form:** 50% lectures / 50% workshop

Training presents selected Design Patterns in practical terms embedded in the context of business applications. All of the patterns are illustrated with examples of use in modeling the application logic and business logic enterprise applications. Our examples deal with patterns and techniques that are used in modeling business applications unlike book examples that cover word processing or machine programming only.

During the training participants will gain an integrated understanding of the achievements of modern software engineering, including basic concepts, which shows both the structure and the need for standards.
During the practical workshop, we combine design patterns and architectural patterns, together with mechanisms that are built-in frameworks Spring or Seam (your choice) to create business models that are flexible and open to extension and characterized by a high level of testability.

Training is dedicated for developers, designers and architects, who all are involved in the process of business software development, wishing to augment their professional skills in the field of programming techniques to enhance the quality of code and design. This acquirement knowledge translates into a practical way of productivity measured in a broader term.

**Advantages:**

- Focusing on the context of business applications
- Selecting only the useful patterns and techniques
- Real examples
## Table of contents:

### 1. Techniques for Object Oriented Design

1.1. Disadvantages of the naive model verbs- nouns

1.2. Modeling hermetic aggregates - a behavioral model

1.3. Modeling invariants

1.4. Modeling of time as one of the key factors essential complexity

### 2. Anti-patterns and common pitfalls

### 3. Design patterns – practical examples based on real problems in the context of enterprise applications

3.1. Command-CommandHandler - a variant of the pattern in the the standard client-server architecture

3.2. Placing the complex business logic of the incremental nature

3.3. Wrapper - a variant of the pattern useful in the meaning-oriented modeling

3.4. Packaging basic types with handy objects

3.5. Alternative to Utils

3.6. Money Archetype

3.7. Combination of Decorator and Strategy to build a variety of incremental algorithms

3.8. Selection the variant of the algorithm without intervention in the core business

3.9. Integration with dependency injection mechanism

3.10. Defining a specific business strategy in a container Inversion of Control (XML or factory method)

3.11. Selection of business logic to the current conditions

3.12. Connection Chain and Strategy to build a variety of conditional algorithms

3.13. A consistent way to create business objects families dependent on the configuration of the system implementation

3.14. Producing the Strategy
### 3. Unified export of the domain objects

### 3.15. Unified export of the domain objects

### 3.16. Hiding the complexity of building queries

### 3.17. Technology of making common the business logic

### 3.18. Template Strategies

### 3.19. Examples of the anti-pattern

### 3.20. Implementation details of the Singletons created with delay and resistant to concurrent access

### 3.21. The implementation of state machine that represents the complex life cycle of a business object

### 3.22. State machine as Wrapper adding new functionality

### 3.23. Traps with over-generalize the State machine

### 3.24. Reducing the complexity of the systems that contain complex logic decisions

### 3.25. Case where there are many possible logical criteria

### 3.26. Case where in a given context (implementation, client) is used only a subset of the rules

### 3.27. Modeling roles in the system

### 3.28. An alternative to inheritance

### 3.29. Variation of Bridge Pattern

### 3.30. Role Object generalization for the development systems open for extension

---

**4. Testability - the impact of the use of best OOD practices and patterns for testable code**

### 4.1. JUnit

### 4.2. Mockito