Kod szkolenia: **REFAKT**

Tytuł szkolenia: **Lepszy kod dzięki technikom refaktoryzacji i wzorcom projektowym**

Dni: 2

Opis:

Adresaci Szkolenia:

Szkolenie w sposób kompleksowy opisuje refaktoryzację i kontekst, w którym powinna ona być wykorzystywana. Rozpoczyna się ono od dyskusji na temat jakości kodu i metod, za pomocą których jesteśmy w stanie stwierdzić, że kod źródłowy jest niskiej jakości. Następnie, uczestnikom przedstawione są zasady, którymi powinien kierować się programista w swojej pracy, by dążyć do kodu o wysokiej jakości. Główną częścią szkolenia są warsztaty z technik refaktoryzacji (m.in. kompozycja metod, upraszczanie wyrażeń warunkowych) oraz wzorców projektowych w oparciu o zbiór GoF (Gang-of-Four).

Cel szkolenia:

- być w stanie ocenić jakość kodu źródłowego, z którym pracuje,
- wskazać w kodzie niedoskonałości, nazwać je i uargumentować, czemu negatywnie wpływają na jakość aplikacji,
- rozumieć różne techniki refaktoryzacji i potrafić je stosować na niskiej jakości kodzie,
- rozumieć kontekst, w którym należy użyć danego wzorca projektowego i potrafić go zaimplementować.

Wymagania:

Uczestnik szkolenia powinien posiadać podstawowe doświadczenie w programowaniu obiektowym. Preferowanym językiem jest Java.

Parametry szkolenia:

2*8 godzin (2*7 godzin netto) wykładów i warsztatów w proporcji: 80% warsztaty, dyskusje; 20% wykłady.

Program szkolenia:

1. Wprowadzenie
    I. Definicja wzorca projektowego
    II. Czy wzorce projektowe są odpowiedzią na braki w danym języku

programowania?

2. Jakość kodu i jej ocena
    I. Jak mierzyć jakość kodu źródłowego?
    II. Code Smells
        i. Duplikacja kodu
        ii. Metody przyjmujące dużą liczbę parametrów
        iii. Długie metody
        iv. Ogromne klasy
        v. God Class
        vi. Zależność od szczegółów implementacyjnych innej klasy
        vii. Stosowanie nazw, które niczego nie opisują
    III. Antywzorce
        i. Cut-and-Paste Programming
        ii. Spaghetti Code
        iii. Programming to Implementation
        iv. Tight coupling
        v. Golden Hammer
        vi. Poltergeist
        vii. Boat Anchor
        viii. Dead End
        ix. Ambigious Viewpoint
        x. Lava Flow
        xi. Mushroom Management
    IV. Poprawianie jakości kodu
        i. (Brakujące) Testy komponentów poddawanych zmianie
        ii. Czytelny kod (tzw. Clean Code)
        iii. Enkapsulacja
        iv. Zasada odpowiedzialności
        v. Kompozycja ponad dziedziczenie
        vi. Programowanie poprzez interfejsy
    V. Dług techniczny
3. Techniki refaktoryzacji
    I. Wprowadzenie
    II. Tworzenie metod
        i. Introduce Explaining Variable
        ii. Split Temporary Variable)
        iii. Replace Template with Query, Inline Temp, Inline Method i Extract Method
        iv. Remove Assignments to Parameters
        v. Substitute Algorithm
    III. Upraszczanie wywołań metod
        i. Replace Constructor with Factory Method, ErrorCode with Exception, Exception with Test, Parameter with Explicit Methods, Parameter with Method,
        ii. Introduce Parameter Object
        iii. Encapsulate Downcast

IV. Przenoszenie cech między obiektami
    i. Move Method i Move Field
    ii. Extract Class
    iii. Inline Class
    iv. Hide Delegate
    v. Remove Middle Man
    vi. Introduce Foreign Method
    vii. Introduce Local Extension
V. Organizacja i modelowanie danych
    i. Replace Array with Object, Data Value with Object, Magic Number with Symbolic Constant, Record with Data Class, Subclass with Fields, Type Code with Class, Type Code with Strategy/State, Type Code with Subclasses
    ii. Encapsulate Collection, Encapsulate Field
    iii. Change Bidirectional Association to Unidirectional, Unidirectional Association to Bidirectional, Reference to Value, Value to Reference
VI. Upraszczanie wyrażeń warunkowych
    i. Decompose Conditional Expressions, Consolidate Conditional Expressions i Consolidate Duplicate Conditional Expressions
    ii. Remove Control Flag
    iii. Replace Nested Conditional with Guard Clauses
    iv. Replace Conditional with Polymorphism
    v. Null Object
VII. Generalizacje
    i. Pull Up Constructor Body, Field, Method
    ii. Push Down Field, Method
    iii. Collapse Hierarchy
    iv. Extract Interface, Subclass, Superclass
    v. Replace Delegation with Intheritance, Inheritance with Delegation

4. Wzorce projektowe
    I. Wprowadzenie
    II. Wzorce GoF
        i. Kreacyjne: Builder, Prototype, Factory Method, Abstract Factory, Singleton
        ii. Strukturalne: Facade, Proxy, Composite, Adapter, Decorator, Bridge
        iii. Behawioralne: Command, Observer, State, Strategy, Chain of Responsibility, Mediator, Visitor, Template Method
    III. Niuanse wykorzystania poszczególnych wzorów
5. Podsumowanie