SZKOLENIE ŚREDNIO ZAAWANSOWANE

Lepszy kod dzięki technikom refaktoryzacji i wzorcom projektowym

REFAKT

Czas trwania: 2 dni

Cele szkolenia

- Możliwość oceny jakości kodu źródłowego, z którym pracuje,
- Wskazać w kodzie niedoskonałości, nazwać je i uargumentować, dlaczego 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

Zalety

Dla kogo?

- 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)

Wymagania

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

Program

1. Wprowadzenie
    a. Definicja wzorca projektowego
    b. Czy wzorce projektowe są odpowiedzią na braki w danym języku programowania?
2. Jakość kodu i jej ocena
    a. Jak mierzyć jakość kodu źródłowego?
    b. Code Smells
        -Duplikacja kodu
        -Metody przyjmujące dużą liczbę parametrów
        -Długie metody
        -Ogromne klasy
        -God Class
        -Zależność od szczegółów implementacyjnych innej klasy
        -Stosowanie nazw, które niczego nie opisują
    c. Antywzorce
        -Cut-and-Paste Programming
        -Spaghetti Code
        -Programming to Implementation
        -Tight coupling
        -Golden Hammer
        -Poltergeist
        -Boat Anchor
        -Dead End
        -Ambigious Viewpoint
        -Lava Flow
        -Mushroom Management
    d. Poprawianie jakości kodu
        -(Brakujące) Testy komponentów poddawanych zmianie
        -Czytelny kod (tzw. Clean Code)
        -Enkapsulacja
        -Zasada odpowiedzialności
        -Kompozycja ponad dziedziczenie
        -Programowanie poprzez interfejsy
    e. Dług techniczny
3. Techniki refaktoryzacji
    a. Wprowadzenie
    b. Tworzenie metod
        -Introduce Explaining Variable
        -Split Temporary Variable)
        -Replace Template with Query, Inline Temp, Inline Method i Extract Method
        -Remove Assignments to Parameters
        -Substitute Algorithm
    c. Upraszczanie wywołań metod

-Replace Constructor with Factory Method, ErrorCode with Exception, Exception with Test, Parameter with Explicit Methods, Parameter with Method,
-Introduce Parameter Object
-Encapsulate Downcast

d. Przenoszenie cech między obiektami
-Move Method i Move Field
-Extract Class
-Inline Class
-Hide Delegate
-Remove Middle Man
-Introduce Foreign Method
-Introduce Local Extension

e. Organizacja i modelowanie danych
-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
-Encapsulate Collection, Encapsulate Field
-Change Bidirectional Association to Unidirectional, Unidirectional Association to Bidirectional, Reference to Value, Value to Reference

f. Upraszczanie wyrażeń warunkowych
-Decompose Conditional Expressions, Consolidate Conditional Expressions i Consolidate Duplicate Conditional Expressions
-Remove Control Flag
-Replace Nested Conditional with Guard Clauses
-Replace Conditional with Polymorphism
-Null Object

g. Generalizacje
-Pull Up Constructor Body, Field, Method
-Push Down Field, Method
-Collapse Hierarchy
-Extract Interface, Subclass, Superclass
-Replace Delegation with Intheritance, Inheritance with Delegation

4. Wzorce projektowe
a. Wprowadzenie
b. Wzorce GoF
-Kreacyjne: Builder, Prototype, Factory Method, Abstract Factory, Singleton
-Strukturalne: Facade, Proxy, Composite, Adapter, Decorator, Bridge
-Behawioralne: Command, Observer, State, Strategy, Chain of Responsibility, Mediator, Visitor, Template Method
c. Niuanse wykorzystania poszczególnych wzorów
5. Podsumowanie