BO·TT·EGA
IT minds

Program szkolenia:

Codzienna Refaktoryzacja Kodu Legacy

Informacje:

Nazwa:	**Codzienna Refaktoryzacja Kodu Legacy**
Kod:	**ref-patterns**
Kategoria:	Refaktoryzacja kodu legacy
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	20% wykłady, 80% ćwiczenia

Efektywna refaktoryzacja to refaktoryzacja skuteczna. Taka która przynosi widoczne efekty dla samopoczucia programistów kiedy widzą jej rezultaty jako bardziej czytelny i zrozumiały kod. A to z kolei pozwoli im nadal rozwijać swój projekt, ponieważ będą potrafić na bieżąco zadbać o równowagę pomiędzy wartością biznesową dla klienta a jakością techniczną.

Skuteczność refaktoryzacji zależy od wielu obszarów naszej pracy. Wbrew pozorom same umiejętności techniczne programistów nie wystarczą chociaż są niezbędne. Aby refaktoryzacja miała swoje stałe miejsce, potrzeba wprowadzić ją do naszej świadomości, wprowadzić jako nawyk do zespołów oraz jako jawną część procesu tworzenia oprogramowania. Wówczas nigdy nie będzie mogła być pomijana.

W tym celu oprócz umiejętności technicznych trzeba rozwinąć także inne obszary otaczające świat programistów. Wśród nich najważniejsze to zadbanie o swoją efektywność osobistą i efektywność pracy zespołu poprzez zbudowanie świadomości jak rezultaty pracy są zależne od funkcjonowania całego zespołu.

Wprowadzanie wszystkich usprawnień zaczyna się od świadomości tego jak organizacja pracy zostaje odzwierciedlona w jakości kodu który dostarcza zespół. W rezultacie będziemy mogli mieć pozytywny wpływ na jakość kodu poprzez zmiany od strony organizacyjnej które umożliwią ciągła refaktoryzację.

Powyższe inspiracje zostaną przedstawione na podstawie takich koncepcji jak "7 Nawyków Skutecznego Działania" Stephena Coveya, "5 Dysfunkcji Pracy Zespołowej' Patricka Lencioni czy też 6 źródeł wpływu wypracowanych z laboratoriach VitalSmarts w Utah.

To praktyczno-teoretyczne szkolenie pokazuje jak podejść do refaktoryzacji w najmniejszych możliwych krokach, robiąc z niej naszą codzienną praktykę, w której codziennie będziemy wzrastać. Poprzez małe kroki, które staną się naszym nawykiem będziemy wykonywać coraz bardziej skomplikowane refaktoryzacje, aż zaczniemy odkrywać wzorce projektowe.

Wzorce projektowe to coś powszechnego. Warto więc z nich skorzystać skoro już są pewne bo sprawdzone. Najprawdopodobniej dostarczą nam oczekiwanych rezultatów polegających na otrzymaniu kodu który jest zrozumiały ale przede wszystkim łatwiej go będzie można rozwijać o kolejne elementy.

Szkolenie oparte jest na koncepcji piramidy refaktoryzacji. Umożliwia ona rozpoczęcie od najprostszych refaktoryzacji a następnie stopniowe przechodzenie do tych bardziej złożonych.

- Upraszczanie logiki algorytmów
- Ekstrakcja mniejszych metod
- Ekstrakcja mniejszych klas
- Zauważanie wzorców projektowych
- Porządkowanie architektury

Szkolenie koncentruje się głównie na pracy z kodem źródłowym z wykorzystaniem wbudowanych zautomatyzowanych refaktoryzacji w narzędziach takich jak IntelliJ lub ewentualnie Eclipse (prze mniejszym zakresie szkolenia). Kod źródłowy napisany jest w Javie. Trener wykonuje wszystkie przekształcenia refaktoryzacyjne na żywo, po czym uczestnicy mogą doświadczyć tego samego poprzez pracę z tym samym kodem na własnym laptopie. Pozostałe 25% czasu poświęcone jest na prezentacje i dyskusje o możliwościach wprowadzenia refaktoryzacji jako codziennego nawyku do pracy zespołu.

Szkolenie to nie jest szkoleniem z architektury. Różne zespoły mogą mieć różne preferencje dotyczące rozwiązań architektonicznych. Celem szkolenia jest nauczenie jak łatwo i szybko można zmieniać architekturę kodu z jednego podejścia do innego.

Zalety szkolenia:

- Refaktoryzacja w kontekście nawyków skutecznego działania
- Proces w kontekście dysfunkcji pracy zespołowej
- Praktycznie, nieksiążkowe wykorzystanie wzorców projektowych

BOTTEGA IT minds

Szczegółowy program:

1. Dzień 1 - myśl przewodnia: "7 Nawyków Skutecznego Działania"

1.1. Część 1

1.1.1. Zapoznanie z “codziennym” kawałkiem kodu który przecież szybko można zrozumieć i “rozwijać”

1.1.2. Pokazanie jak ten “szybki rozwój” metodą kopiuj-wklej zmniejszy równie szybko czytelność.

1.1.3. Refaktoryzacja od najprostszym przekształceń jak ekstrakcje zmiennych, metod, parametrów z tych metod poprzez przenoszenie tych metod do istniejących jak i nowych klas aż do ukrywania klas za ich abstrakcjami.

1.1.4. Prezentacja Piramidy Refaktoryzacji, tego w jaki sposób jej koncepcja została zastosowana przy kolejności powyższych refaktoryzacji.

1.1.5. Nauka wbudowanych automatycznych refaktoryzacji w środowisku IntelliJ lub Eclipse

1.1.6. Zastosowanie zasad SOLID jako podstawy do podziału architektury na mniejsze części

1.2. Część 2

1.2.1. Zapoznanie z rozszerzonym kodem z części 1 gdzie nowa funkcjonalność została dodana bez dokonania jakichkolwiek refaktoryzacji

1.2.2. Przedstawienie kolejnej dodatkowej funkcjonalności którą chcemy dodać do projektu

1.2.3. Dyskusja, że możemy to zrobić szybko ale dalej obniżając czytelność i jakość kodu

1.2.4. Wykonanie “ekstrakcji punktów rozszerzenia” jako wynik “podróży refaktoryzacyjnej” z użyciem koncepcji Piramidy Refaktoryzacji

1.2.5. Dostarczenie nowej funkcjonalności jako nowa implementacja “punktu rozszerzenia” z użyciem TDD

2. Dzień 2 - myśl przewodnia "5 Dysfunkcji Zespołu"

2.1. Refaktoryzacje w małych krokach oparte o koncepcję piramidy refaktoryzacji zakończone uzyskaniem implementacji poniższych wzorców projektowych

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. Dzień 3 - myśl przewodnia: "6 źródeł wpływu"

3.1. Refaktoryzacje w małych krokach oparte o koncepcję piramidy refaktoryzacji zakończone uzyskaniem implementacji poniższych wzorców projektowych

3.1.1. (Fluent) Builder

3.1.2. Proxy

3.1.3. Template

3.1.4. Bridge

3.1.5. Command

3.1.6. Adapter