Testowanie programów z użyciem JUNIT w Eclipse

Tworzenie projektu z biblioteką JUnit

Wybieramy *File -> New ->JUnitTestCase*, podajemy nazwę projektu, *Next*
W zakładce *Libraries* wybieramy *Add Library* i dalej *JUnit*, *Next*
W oknie *JUnit Library* wybieramy wersję biblioteki i zatwierdzamy, *Finish, Finish*

Dodanie biblioteki JUnit do istniejącego projektu

Otwieramy okno własności projektu (*Properties*).
Na liście wybieramy pozycję *Java Build Path*
Wybieramy zakładkę *Libraries*
Klikamy na przycisk *Add Library* i wybieramy z listy *JUnit*.
Wybieramy wersję biblioteki i zatwierdzamy.

Alternatywnie, możemy tworzyć nowe testy (jak poniżej: *JUnit Test Case, JUnit Test Suite*).
Przy tworzeniu pierwszych testów zostanie dołączona wybrana wersja biblioteki.

Tworzenie testów jednostkowych

Dla wybranego pakiet lub klasy w projekcie Javy możemy utworzyć klasę z testami jednostkowymi.

Wybieramy *File -> New ->JUnitTestCase.*

Wybieramy wersję JUnit, określamy nazwę klasy testującej, wybieramy rodzaje szkieletów metod oraz testowaną klasę. W przypadku tworzenia testów dla wybranej klasy możemy zaznaczyć metody tej klasy, dla których tworzone będą metody testujące.

Biblioteka zawierająca junit.jar odpowiedniej wersji zostanie dołączona do projektu.

Należy uzupełnić kod metod testujących.

Uruchamianie testów jednostkowych

Uruchamiamy metody z klasy testującej wybierając *Run -> Run as -> JUnit test.*
Zostaną wykonane wszystkie metody testujące wskazanej klasy.

Prezentacja wyników

Wyniki zostaną wyświetlone w zakładce *JUnit* (*JUnit view*)
Jeśli jest zamknięta można ją otworzyć *Window-> Show view -> Other-> Java JUnit*

Refaktoryzacja kodu

1. **Warunki wstępne**

   Należy przygotować podstawowe **przypadki testowe** pozwalające sprawdzić działanie programu (testy funkcjonalne i testy jednostkowe).

2. **Zapachy w kodzie**

   Na podstawie analizy programu zidentyfikować **"zapachy w kodzie"** do refaktoryzacji.

3. **Testowanie programu**

   Uruchomić podstawowy zbiór testów dla programu.

4. **Refaktoryzacja kodu**

   Utworzyć **kopię** uruchomionego projektu.

   Na kopii wykonać ciąg wybranych przekształceń refaktoryzacyjnych (automatycznie - z listy dostępnych w środowisku Eclipse) oraz innych np. złożonych refaktoryzacji (półautomatycznie).

   Dla wybranego elementu projektu przekształcenia są dostępne z menu *Refactor*.

   Ślad wykonanych refaktoryzacji jest też dostępny w historii (w perspektywie *Java* z głównego menu wybrać *Refactor->History*).

   Obserwować zmiany w kodzie.

5. **Testowanie programu**

   Po każdym wykonaniu grupy przekształceń oraz po zakończeniu modyfikacji programu przetestować go ponownie. W razie potrzeby zmodyfikować zbiór testów.

Rezultaty

Przedstawić działanie programu, zbiory testów, wyniki refaktoryzacji.

UWAGA

Refaktoryzacja kodu w Javie jest też wspomagana przez inne narzędzia, np.:
IntelliJ IDEA

Materiały uzupełniające

Refaktoryzacja kodu - Przykładowe przekształcenia do wykorzystania

Na podstawie http://wazniak.mimuw.edu.pl/

Wydzielenie metody (Extract method)

Wykonać przekształcenie wydzielenie metody (ang. *Extract method*).

1. Znajdź metodę zbyt dużą lub o dużej złożoności.
2. Zaznacz fragment metody, w którym nie są wykorzystywane zmienne lokalne
3. Następnie z menu kontekstowego wybierz opcję *Refactor->ExtractMethod*. Pojawi się okienko, w którym należy wprowadzić nazwę metody oraz jej zakres widoczności. Zwróć uwagę, że parametry metody są obliczane automatycznie na podstawie zmiennych zdefiniowanych poza zaznaczonym fragmentem, a do których istnieją odwołania wewnątrz niego.
4. Po wybraniu klawisza *Preview* zostanie wyświetlona nowo tworzona metoda.
5. Wybór klawisza OK zatwierdza utworzenie nowej metody o podanej nazwie, a w miejscu zaznaczonego fragmentu kodu zostanie umieszczone wywołanie tej metody.

Sprawdzić przez zaznaczenie różnych fragmentów metody - kiedy Eclipse nie daje możliwości wykonania tego przekształcenia.

Zmiana sygnatury metody (Change Method Signature)

Przekształcenie to jest złożeniem dwóch refaktoryzacji: dodania parametru i usunięcia parametru, a także pozwala zmieniać typ metody i zasięg jej widoczności. Służy do modyfikowania listy parametrów przekazywanych metodzie.

1. Wskaż kursorem sygnaturę wybranej metody.
2. Z menu kontekstowego wybierz opcję *Refactor->Change Method Signature*
3. Pojawi się okienko, w którym możesz zmodyfikować wartości parametrów (zmodyfikować parametr, usunąć parametr lub dodać nowy).
4. Po wybraniu klawisza *Preview* pojawi się okienko z listą plików, które zostaną zmodyfikowane, oraz zakres zmian w każdym pliku.
5. Po wybraniu klawisza *OK* sygnatura metody oraz wszystkie wywołania tej metody zostaną zmodyfikowane.

W niektórych sytuacjach przekształcenie nie może zostać poprawnie zakończone (np. metoda wymagana przez interfejs). Zgłoszone zostanie ostrzeżenie.

Rozwinięcie (Inline)

Przekształcenie to służy do rozwinięcia zmiennej (ang. *Inline Local Variable*) lub metody (ang. *Inline Method*), czyli jest komplementarne np. w stosunku do przekształcenia *Extract Method*.

1. Wskaż kursorem wywołanie wybranej metody w innej metodzie.
2. Z menu kontekstowego wybierz opcję *Refactor->Inline.*
3. Pojawi się okienko, w którym należy dokonać wyboru pomiędzy zastąpieniem wszystkich wywołań tej metody jej ciałem, czy tylko wskazanego wywołania. W przypadku wyboru wszystkich wywołań wybrana metoda stanie się nie wykorzystywana i będzie mogła być usunięta.
4. Pojawi się okienko podglądu z listą modyfikowanych klas i zmianami, jakie zostaną wprowadzone. Przeanalizuj je. Rozwijana metoda została usunięta (jeżeli w poprzednim kroku zaznaczono opcję usunięcia metody).

Przemieszczenie składowej (Move)

Przekształcenie służy do przeniesienia składowej (pola lub metody) do innej klasy. Stosowane jest zwykle w sytuacji, gdy odwołania do tej składowej w większości pochodzą z innej klasy.

Składowa statyczna może zostać przeniesiona do dowolnej klasy.

1. Wskaż kursorem wybraną składową statyczną.
2. Z menu kontekstowego wybierz opcję *Refactor->Move*
3. Pojawi się okienko z prośbą o wskazanie klasy, do której składowa ma zostać przeniesiona.
4. Po wybraniu klawisza *Preview* pojawi się okienko podglądu z listą modyfikowanych klas oraz zakresem zmian.
5. Po wybraniu klawisza *OK* składowa zostanie przeniesiona

W przypadku składowych niestatycznych (tzn. należących do instancji klasy) składowa może zostać przeniesiona do obiektu, który jest polem klasy źródłowej (jest to jedyna możliwość w przypadku przenoszenia pól) lub argumentem przenoszonej metody (w przypadku przenoszenia metody).

Przeniesienie metody powoduje również przekazanie w metodzie referencji do obiektu źródłowego.

1. Wskaż kursorem wybraną metodę.
2. Z menu kontekstowego wybierz opcję *Refactor->Move.*
3. Pojawi się okienko, w którym zostaną wyświetlone potencjalne klasy docelowe. Okienko to pozwala na zmianę nazwy metody podczas przenoszenia oraz określenie nazwy parametru z obiektem źródłowym

4. Po wybraniu klawisza *Preview* mogą pojawić się ostrzeżenia (np. o zmianie zasięgu widoczności pola), które akceptujemy przez *Continue.*
5. Zostanie wyświetlona lista modyfikowanych klas wraz z zakresem zmian. Przeanalizuj je.
6. Po wybraniu klawisza *OK* metoda zostanie przeniesiona.

Przeniesienie składowych w obrębie hierarchii dziedziczenia

Do tej kategorii należą przekształcenia *Pull up i Push down*. Służą one odpowiednio do przeniesienia składowej do nadklasy lub do podklasy. W obu przypadkach podstawowym warunkiem wykonania przekształcenia jest istnienie w projekcie odpowiednio nadklasy (lub implementowanego interfejsu) i podklasy w postaci źródłowej.

1. Wskaż kursorem wybraną metodę.
2. Z menu kontekstowego wybierz opcję *Refactor->Pull Up*
3. Pojawi się okienko, w którym należy wskazać docelowy typ, do którego ma być przeniesiona metoda. Za pomocą klawisza *Set Required* można dołączyć inne składowe, które są wykorzystywane przez przenoszoną metodę, i które także mogą być przeniesione wraz z nią. Klawisz *Set Action* służy do określenia typu wykonywanej akcji.
4. Klikając kolejno klawisze *Next* wykonaj przekształcenie, analizując poszczególne etapy.

Narzędzie do identyfikacji zapachów w kodzie

JDeodorant to wtyczka do Eclipsa, który umożliwia automatyczną identyfikację zapachów w oprogramowaniu oraz sugeruje odpowiednią metodę refaktoryzacji dla danego zapachu. Wtyczkę można pobrać ze strony:
https://marketplace.eclipse.org/content/jdeodorant
Za pomocą JDeodorant można aktualnie wykryć pięć zapachów:

- **Zazdrość o kod** ("Feature Envy") do rozwiązania za pomocą refaktoryzacji Przenieś metodę (“Move method”)
- **Sprawdzanie stanu** ("State Checking") sugerowane rozwiązanie za pomocą zamiany wyrażenia warunkowego na polimorfizm (“Replace Conditional with Polymorphism”)
- **Duża metoda** ("Long Method") rozwiązywane za pomocą odpowiedniego przekształcenia typu: Ekstrakcja metody (“Extract Method”)
- **Duża klasa** ("God Class") rozwiązywane za pomocą odpowiedniego przekształcenia typu: Ekstrakcja klasy (“Extract Class”)
- **Duplikaty kodu** („Duplicated code”) rozwiązywane za pomocą odpowiedniego przekształcenia typu: Ekstrakcja klonu (“Extract Clone”)

Uwaga - nie wszystkie miejsca wskazywane przez narzędzie faktycznie wymagają refaktoryzacji. Możliwe jest zgłaszanie błędnych wskazań typu “false positive”!