L10 Wzorce i narzędzia refaktoryzacji

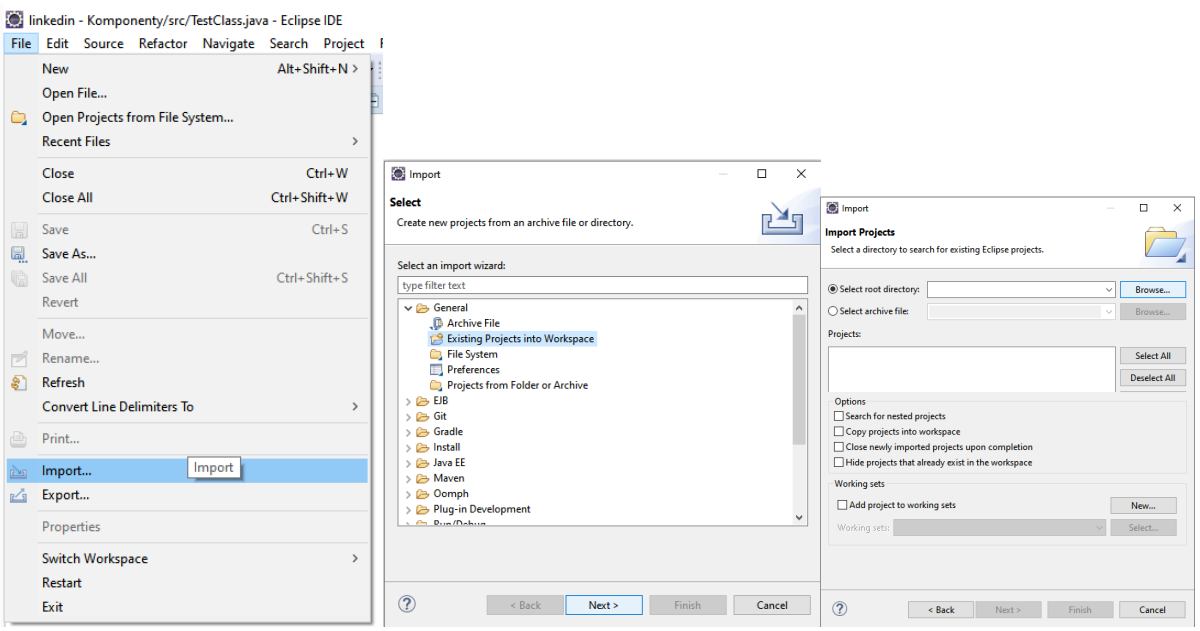
Ćwiczenia laboratoryjne

Cel

Celem ćwiczeń laboratoryjnych jest przekazanie podstawowych umiejętności refaktoryzacji na przykładzie wzorców refaktoryzacji opisanych w klasycznej pozycji Martina Fowlera *„Refaktoryzacja. Ulepszanie struktury istniejącego kodu”.*

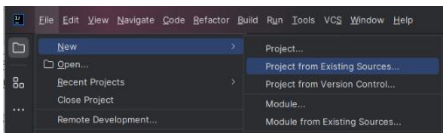
Przygotowanie

Proszę ściągnąć spakowany plik z katalogu Pliki grupy laboratoryjnej w MS TEAMS, rozpakować do wybranego katalogu i zaimportować z tego katalogu do Eclipse istniejący projekt.



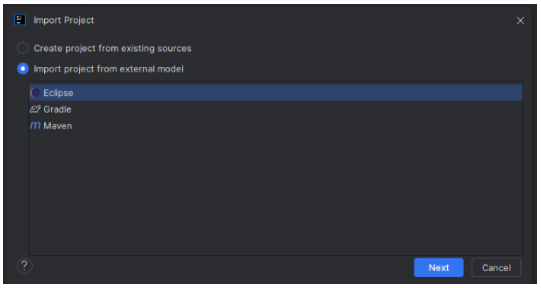
Środowisko IntelliJ

W IntelliJ uruchamianym po raz pierwszy należy utworzyć projekt bez nazwy aby uzyskać dostęp do pełnego menu. Następnie w menu wybrać *File/New/Project from Existing Sources…*.

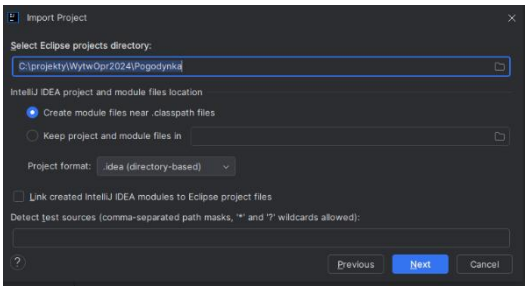


Alternatywnie można od razu, jako pierwszy krok, wybrać tę opcję za pomocą skrótu: *Ctrl+Shift+A*.

Potem wybrać katalog z projektem.
Następnie w oknie dialogowym wybrać opcję *Import project from external model* i *Eclipse* jako źródło projektu.



Kolejnym krokiem jest zaimportowanie projektu wraz z ustawieniami.



W kolejnych oknach dialogowych proszę naciskać *Next* i *Create.*

Sprawozdanie

W sprawozdaniu proszę umieścić nagłówek z grupą, imionami, nazwiskami osób. Poniżej nazwy zadań oraz zrefaktoryzowany kod sformatowany przez środowisko Eclipse/IntelliJ, który ma wyglądać tak:

```
public class TestClass {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("To jest sformatowany kod");
	}
}
```

W treści sprawozdania proszę zamieszczać tylko klasę, w której znajduje się kod powstały w wyniku refaktoryzacji oraz kod, który wywołuje ten zmieniony kod (jeśli te wywołania również się zmieniły).

Pierwszy Pakiet Przekształceń

Zadania są inspirowane podstawowymi wzorcami refaktoryzacji, które Martin Fowler określił następująco: „*Katalog otwiera pakiet przekształceń, które według mnie są najbardziej przydatne i dlatego warto je poznać w pierwszej kolejności*".

Zadanie 1. Ekstrakcja Funkcji

Ekstrakcja funkcji dotyczy również metod, procedur i podprocedur. Polega na wyodrębnieniu spójnego fragmentu kodu i przeniesieniu go do osobnej funkcji o nazwie dobrze opisującej jej działanie.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.invoice**

- Stwórz nową funkcję i nazwij ją tak, by nazwa opisywała cel jej działania (czyli to, co robi funkcja, a nie jak to robi).
- Skopiuj kod, który chcesz wyodrębnić, do ciała nowej funkcji.
- Przeszukaj wyekstrahowany kod pod kątem zmiennych lokalnych pochodzących z funkcji
- źródłowej. Będą one zmiennymi lokalnymi i parametrami nowej funkcji.
- Skompiluj kod po przekształceniu wszystkich zmiennych.
- Zastąp wyodrębniony kod w funkcji źródłowej wywołaniem funkcji wynikowej.
- Przetestuj kod.
- Poszukaj innych takich samych lub podobnych fragmentów kodu jak wyodrębniony i ewentualnie zastosuj Zastąpienie Wchłoniętego Kodu Wywołaniem Funkcji, aby wywołać nową funkcję.

Podpowiedź

.dotem hcynawohartskeyw owon hceretzc ńałowyw z ęis ćadałks anniwop gniwOtnirp adoteM

Zadanie 2. Wchłonięcie Funkcji

Celem refaktoryzacji jest otrzymanie krótkich funkcji, których nazwy odpowiadają realizowanym przez nie działaniom. Takie funkcje prowadzą do powstania przejrzystego i czytelnego kodu. Czasami jednak trafia się funkcja, której ciało jest tak samo czytelne jak jej nazwa. Bywa, że taka funkcja powstaje w wyniku refaktoryzacji. Po napotkaniu takiej funkcji należy ją wyeliminować. Pośrednictwo może być przydatne, jednak zbędne pośrednictwo po prostu irytuje.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.driver**

- Sprawdź, czy metoda nie jest polimorficzna.
- Znajdź wszystkie wywołania funkcji.
- Zastąp każde z wywołań ciałem funkcji.
- Przetestuj kod po każdej zamianie.
- Usuń definicję funkcji.

Podpowiedź

.ąnlazczsupod azcarkezrp watsod hcynoinźópo abzcil yzc ,acąjalśerko ajcknuf tsej melec myzsaN

Zadanie 3. Ekstrakcja Zmiennej

Wyrażenia bywają złożone i nieczytelne. W takich sytuacjach korzystne może się okazać wprowadzenie zmiennych tymczasowych, które pozwalają podzielić wyrażenie na mniejsze, łatwiejsze do opanowania elementy. Stosując je, można fragmentom kodu nadawać nazwy, dzięki którym łatwiej jest zrozumieć, co się w nim dzieje. Tego rodzaju zmienne ułatwiają też diagnostykę kodu, ponieważ można je wykorzystywać w debuggerze i wyświetlać ich wartości.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.order.** Proszę wyekstrahować zmienne z wyrażenia obliczającego wartość zamówienia, aby uczynić kod bardziej czytelnym i zrozumiałym.

- Sprawdź, czy wyrażenie, które zamierzasz wyodrębnić, nie wywołuje efektów ubocznych.
- Zadeklaruj niemutowalną zmienną (słowo kluczowe **final**) i przypisz jej kopię wyrażenia, któremu chcesz nadać nazwę.
- Zastąp oryginalne wyrażenie nową zmienną.
- Przetestuj kod.

Podpowiedź

.enneimz enawohcartskeyw yzrt mat ęzdiW

Zadanie 4. Zmiana Deklaracji Funkcji

Funkcje stanowią podstawowe narzędzie do dzielenia programu na części. Deklaracje funkcji określają dopasowanie tych części, czyli połączenia wewnątrz oprogramowania. Tak jak w każdej konstrukcji, połączenia te mają kluczowe znaczenie. Dobre połączenia pozwalają łatwo dodawać nowe części do systemu, a złe utrudniają zrozumienie działania programu i jego modyfikowanie w miarę zmieniających się potrzeb. Na szczęście oprogramowanie jest miękkim tworem (ang. software - miękka rzecz), dzięki czemu można zmieniać połączenia, o ile robi się to ostrożnie. Najważniejszym elementem połączenia jest nazwa funkcji. Dzięki dobrej nazwie można szybko stwierdzić, co funkcja robi, nie zaglądając w jej kod.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.newmath**. Proszę zmień nazwę funkcji obliczającej średnicę koła ze zbyt skrótowej **circum** na pełną nazwę **circumference.**

- Jeżeli usuwasz parametr, sprawdź, czy w ciele funkcji nie ma odwołań do niego.
- Nadaj deklaracji funkcji docelową postać.
- Wyszukaj wszystkie odwołania do poprzedniej deklaracji funkcji i zastąp je odwołaniem do nowej deklaracji.
- Przetestuj program.

Podpowiedź

W środowisku Eclipse można zmienić nazwę klasy, pola, metody i zmiennej, na której stoi kursor przez wciśnięcie klawiszy lewy Shift + lewy Alt + R, wpisanie nowej nazwy i wciśnięcie klawisza Enter.
Narzędzie refaktoryzacji automatycznie zastąpi wszystkie użycia w kodzie starej nazwy na nową nazwę.

Zadanie 5. Enkapsulacja Zmiennej

Refaktoryzacja polega na manipulowaniu elementami programu. Trudniej jest manipulować danymi niż funkcjami. Aby przesunąć dane i by kod dalej działał, należy w jednym kroku zmienić wszystkie odwołania do danych. W przypadku danych o bardzo małym obszarze widoczności, np. zmiennej tymczasowej w niewielkiej funkcji, nie jest to problem. Jednak, gdy zakres jest większy, pojawiają się trudności. Dlatego zmienne globalne stanowią prawdziwy kłopot. Gdy zamierzam przesunąć szeroko dostępne dane, często najpierw enkapsuluję je i udostępniam za pomocą funkcji. W ten sposób zamieniam trudne zadanie reorganizacji danych na nieco prostsze, polegające na reorganizacji funkcji.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.space**. Proszę zaenkapsuluj zmienną globalną **defaultOwner** w klasie **Space.**

- Utwórz funkcję enkapsulującą, udostępniającą i modyfikującą zmienną.
- Wykonaj statyczny test.
- Każde odwołanie do zmiennej zastąp odwołaniem do funkcji enkapsulującej. Przetestuj kod po każdej zmianie.
- Ogranicz widoczność zmiennej.
- Przetestuj kod.

Podpowiedź

W środowisku Eclipse wciśnięcie klawiszy lewy Shift + lewy Alt + S uruchamia okno wyboru narzędzi kodu źródłowego. Wciśnięcie klawisza R wywołuje funkcję generowania getterów i seterów dla klasy, w której umieszczony jest kursor.

.etavirp upętsod arotakifydom ącomop az alop od pętsod żeinwór zcinargO

Zadanie 6. Zebranie Funkcji w Klasę

Klasy są podstawowymi strukturami w językach obiektowych, ale przydają się również w innych sytuacjach. Gdy widzę grupę funkcji, które operują na tych samych danych (najczęściej przekazywanych w argumentach), wykorzystuję możliwość utworzenia klasy. Klasa stanowi jednostkę, którą funkcje mogą wykorzystywać bardziej jawnie. Wywołania funkcji wewnątrz obiektu mogą być prostsze, można usunąć wiele ich parametrów, jak również w innych miejscach kodu można stosować odwołania do obiektu.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.tea**. Proszę przenieś funkcje wyliczające cenę i podatki zamówienia do klasy zawierającej dane o spożyciu herbaty. Następnie w trzech metodach zawierających logikę biznesową wykorzystaj metody utworzone w tej klasie.

- Zastosuj Enkapsulację Rekordu współdzielonego przez funkcje. Nasz rekord już jest klasą, więc ten punkt można pominąć.
- Każdą funkcję, która wykorzystuje wspólny rekord, umieść w nowej klasie, stosując Przeniesienie Funkcji.
- Możesz usunąć wszystkie parametry funkcji znajdujące się teraz w rekordzie danych.
- Każdą operację przetwarzająca dane możesz wyodrębnić, stosując Ekstrakcję Funkcji, a następnie przenieść do nowej klasy.

Podpowiedź

.ensaj ćyb onniwop oktsyzsW

Zadanie 7. Zebranie Funkcji w Transformatę

Często programy są zasilane danymi, na podstawie których dostarczają rozmaitych informacji. Uzyskane wartości są wykorzystywane w różnych miejscach, w których czasami powtarza się wyliczenia. Preferuję umieszczanie wszystkich operacji w jednym miejscu, które można łatwo znaleźć i zmieniać. Unika się wtedy duplikowania kodu. Jednym ze sposobów osiągnięcia tego celu jest użycie funkcji transformującej, która przekształca dane wejściowe i zapisuje wyniki w polach danych wyjściowych. Aby sprawdzić wyniki, wystarczy się przyjrzeć funkcji transformującej.

Instrukcja

Kod źródłowy znajduje się w pakiecie **pl.edu.zut.wi.po.tea**. Proszę kod z zadania 6 skopiować do sprawozdania i następnie użyć go do wykonania aktualnego zadania.

- Utwórz funkcję transformującą, której parametrem jest rekord przeznaczony do przekształcenia i która zwraca te niezmienione wartości.
- Wybierz fragment kodu i przenieś go do transformaty, aby utworzyć nowe pole w rekordzie.
- Zmień kod kliencki tak, aby odwoływał się do nowego pola.
- Jeżeli kod jest skomplikowany, zastosuj najpierw Ekstrakcję Funkcji.
- Przetestuj kod.
- Powtórz proces, aby utworzyć inne potrzebne funkcje.

Podpowiedź

.lóp z icśotraw enozcilyw atyzcdo awosenzib akigol a ywokinyw lóp icśotraw yzcilyw ydotem jezcnydejop einałowyw umet ikęizD .ńezcilbo ikinyw ecąjuwohcezrp alop ćadod ytabreh uicyżops o enad jecąjareiwaz ysalk od yżelan uinadaz W