METODY TESTOWANIA OPROGRAMOWANIA

LABORATORIUM 1

Refaktoryzacja

wersja 1.0

przygotował:
dr inż. Radosław Adamus

.

Efekty:

Po ukończeniu laboratorium będziesz:
1. Potrafił analizować kod źródłowy pod kątem problemów strukturalnych.
2. Potrafił planować kroki refaktoryzacji.
3. Potrafił refaktoryzować kod źródłowy z wykorzystaniem narzędzi dostępnych w IDE.
4. Rozumiał potrzebę systematycznego zatwierdzania zmian w systemie kontroli wersji.

Wymagania wstępne:

1. Posiadanie konta na platformie Github.

Narzędzia:

1. Eclipse IDE
2. Git

Reguły wykonywania ćwiczeń laboratoryjnych:

1. Zmiany należy zatwierdzać często. Zatwierdzenie zbiorcze zmian na koniec laboratorium równoważne jest z jego niezaliczeniem.
2. Zmiany zatwierdzone w repozytorium kontroli wersji muszą posiadać znaczące komentarze.
3. Git powinien być tak skonfigurowany, aby zatwierdzane zmiany były identyfikowane danymi studenta (adres email oraz nazwisko).

Opis laboratorium:

1. Analiza i refaktoryzacja kodu:

1. Sklonuj (operacja fork) repozytorium https://github.com/mto-lab/lab1_1 na swoje konto GitHub (a następnie na lokalny komputer). Zaimportuj projekt do Eclipse IDE.
2. W katalogu src\main\properties znajduje się diagram klas reprezentujący obecną strukturę kodu. W dwuosobowych zespołach dokonajcie analizy kodu i wypiszcie zauważone problemy. Zaproponujcie refaktoryzacje poprawiające strukturę projektu (w formie kroków).
3. Analiza propozycji na forum grupy laboratoryjnej.
4. Indywidualnie zaimplementuj uzgodnione refaktoryzacje (w dedykowanej gałęzi o nazwie: *refactorization*). Zatwierdzając każdy krok w repozytorium.
5. Dokonaj ostatecznej synchronizacji repozytoriów (lokalne -> zdalne) pozostawiając zmiany w gałęzi.

2. Wykorzystanie prostych wzorców w refaktoryzacji

1. Sklonuj (operacja fork) repozytorium https://github.com/mto-lab/lab1_2 na swoje konto GitHub (a następnie na lokalny komputer). Zaimportuj projekt do Eclipse IDE.
2. Przeanalizuj strukturę projektu. Utwórz diagram UML reprezentujący strukturę klas.
3. Utwórz gałąź w repozytorium dla zatwierdzania poniższych refaktoryzacji.
4. Zrefaktoryzuj proces tworzenia instancji klasy Payment oraz Invoice tak aby w w kodzie klienckim nie było odwołań do instrukcji **new**. Wybierz wzorzec i zaplanuj kroki refaktoryzacji. Zatwierdź kolejne zmiany w repozytorium.
5. Parametry metody issuance klasy BookKeeper są ze sobą logicznie powiązne. Zrefaktoryzuj kod poprzez utworzenie klasy InvoiceRequest reprezentującej strukturę danych związanych z fakturą.
6. Sposób obliczania podatku jest bezpośrednio zaimplementowany w metodzie issuance. Z jednej strony mamy tutaj problem z poziomami abstrakcji. Z drugiej strony sposób obliczania podatku jest zależny od wielu czynników i może ulegać zmianie. Jaki wzorzec można wykorzystać w celu odizolowania tej zmienności? Zaproponuj kroki tej refaktoryzacji. Zaimplementuj i zatwierdź je kolejno w repozytorium (w dedykowanej gałęzi o nazwie: *refactorization*).
7. Utwórz diagram reprezentujący zmodyfikowany projekt.
8. Dokonaj synchronizacji repozytoriów (lokalne -> zdalne) pozostawiając zmiany w gałęzi.