Testowanie i weryfikacja oprogramowania
Projekt 2 - Test-Driven Development

Mateusz Supronowicz (kierownik)
Norbert Grzyb
Paweł Polański

3 grudnia 2014

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest zapoznanie się z metodyką wytwarzania oprogramowania jaką jest Test-Driven Development. W poniższym dokumencie odnosząc się do niej będę posługiwał się skrótem TDD. Głownymi zadaniami do wykonania w terminie 03.12.2014r. było przeprowadzenie przykładów ze stron zamieszczonych w specyfikacji projektu otrzymanej na zajęciach, opis tych projektów, a także zaproponowanie własnego programu, który będzie w następnym etapie wytwarzany zgodnie z metodyką TDD. Program ten ma za zadanie realizować 7 funkcjonalności.

1.2 Środowisko, narzędzia

System operacyjny: Windows 8.1
IDE: Netbeans 8.0.1
Biblioteki: JUnit 4.10, TestNG 6.8.1, Mockito 1.9.5

2 Projekt 1a z ISOD

2.1 Test 1

Przed pierwszym testem powstaje szkielet systemu w postaci klasy i 2 interfejsów. Następnie powstaje test, na obiektach pozornych, sprawdzający czy pojedynczy zarejestrowany klient otrzymuje wiadomość. Test nie przechodzi bo kod nie jest napisany (red). Po dopisaniu treści metod testy przechodzą (green).

2.2 Test 2

W drugim teście sprawdzamy czy każdy zarejestrowany klient otrzymuje wiadomość. Test nie przechodzi ponieważ dana funkcjonalność nie jest zaimplementowana. Dopisujemy obsługę wielu klientów i oba testy powinny przejść. Ponieważ oba testy korzystają z tych samych obiektów pozornych następuje refaktor i wyciągnięcie mocków do metody wykonywanej przed każdym testem.

2.3 Test 3

Test ten sprawdza czy osoby które nie są zarejestrowane nie dostają wiadomości. Ponieważ funkcjonalność ta została automatycznie zaimplementowana test od razu przechodzi. Po napisaniu następuje mały refaktor aby posortować metody.

2.4 Test4

Sprawdzamy czy klient zarejestrowany kilkukrotnie nie dostaje więcej niż jednej wiadomości. Test nie przechodzi więc dopisujemy funkcjonalność do głównego programu. Po zmianie rodzaju kolekcji napisany test i poprzednie przechodzą.

2.5 Test 5

Do programu dopisujemy nową pustą metodę. Piszemy test który sprawdza czy usunięci klienci dostają wiadomości. Test nie przechodzi więc zabieramy się do implementacji. Po poprawnej implementacji testu, oraz wcześniejsze, powinny przechodzić.

3 Wymagania na własne oprogramowanie z zastosowaniem metodyki TDD

Zespół decyduje się na zrealizowanie oprogramownia wykonującego proste operacje na macierzach. Główną i jedyną klasą będzie klasa Matrix zawierająca następujące funkcjonalności.

3.1 Tworzenie macierzy o zadanych wartościach.

Opis: Celem jest stworzenie obiektu typu Matrix wypełnionego liczbami wejściowymi.
Dane wejściowe: Tablica z wartościami typu double, liczba wierszy, liczba kolumn.
Wynik: Stworzene obiektu Matrix wypełnionego podanymi wartościami wejściowymi.

3.2 Obsługa niepoprawnej liczby elementów przy tworzeniu macierzy.

Opis: W przypadku, gdy liczba elementów macierzy nie wypełnia podanej liczby wierszy i kolumn, pozostałe miejsca są wypełniane zerami.
Dane wejściowe: Tablica z wartościami typu double, liczba wierszy, liczba kolumn.
Wynik: Stworzene obiektu Matrix wypełnionego podanymi wartościami wejściowymi, wypełnionego o wartości 0.

3.3 Tworzenie macierzy jednostkowej.

Dane wejściowe: n-liczba wierszy (kolumn).
Wynik: Stworzene obiektu Matrix będącego n-wymiarową macierzą jednostkową.

3.4 Dodawanie dwóch macierzy.

Opis: Funkcja realizująca dodawanie dwóch macierzy.
Dane wejściowe: Dwie macierze.
Wynik: Nowa macierz będąca rezultatem dodawania dwóch macierzy wejściowych.

3.5 Mnożenie macierzy przez skalar.

Opis: Funkcja realizująca mnożenie macierzy przez skalar, czyli mnożąca poszczególne elementy macierzy przez zadaną wartość.
Dane wejściowe: Macierz wejściowa, skalar.
Wynik: Nowa macierz będąca rezultatem pomnożenia macierzy wejściowej przez podany skalar.

3.6 Mnożenie dwóch macierzy.

Opis: Funkcja realizująca mnożenie dwóch macierzy przez siebie.
Dane wejściowe: Dwie macierze wejściowe.
Wynik: Nowa macierz będąca rezultatem pomnożenia dwóch macierzy przez siebie.

3.7 Mnożenie dwóch macierzy - obsługa niepoprawnego rozmiaru.

Opis: Obsługa błędu spowodowanego niepoprawnym rozmiarem macierzy. Liczba kolumn macierzy pierwszej nie jest równa liczbie wierszy macierzy drugiej.
Dane wejściowe: Dwie macierze wejściowe o błędnych rozmiarach.
Wynik: Rzucenie wyjątku RuntimeException z detaliczną informacją o błędzie.

3.8 Obliczanie wyznacznika zadanej macierzy kwadratowej.

Dane wejściowe: Macierz kwadratowa.
Wynik: Wyznacznik macierzy podanej na wejściu.

3.9 Błąd obliczania wyznacznika zadanej macierzy niekwadratowej.

Dane wejściowe: Macierz niekwadratowa.
Wynik: Rzucenie wyjątku RuntimeException mówiącego o niepoprawnym formacie macierzy.