Wrocław 2018
Cezary Hołub

Wrocławska Wyższa Szkoła Informatyki Stosowanej

Przedmiot	Zaawansowane praktyki programistyczne Testy jednostkowe
Semestr	Lato 2018

Materiały do ćwiczeń

Uwaga!!! Rozwiązaną listę zadań wysyłamy do tygodnia czasu od jej opublikowania

ZADANIE 0	Import projektu

1. Sklonuj repozytorium gita: https://github.com/cezary-holub/wwsis-blog
   Zawiera ono projekt mavenowy z początkiem implementacji bloga.
2. Zaimportuj projekt do Eclipse:
   a. Wybierz menu File->Import
   b. W okienku wybierz opcję Maven->Existing Maven Projects, kliknij Next
   c. Wybierz katalog ze sklonowanym repozytorium (opcja Browse) po czym kliknij Finish

ZADANIE 1	Przeglądanie istniejących testów

Zapoznaj się z zawartością projektu:

- Klasa pl.wwsis.zpp.blog.App -- zawiera funkcję main z przykładowym kontrolerem Spark-owym.
- plik src/main/resources/spark/template/freemarker/hello.ftl -- szablon widoku wykorzystany w powyższym kontrolerze
- Klasa pl.wwsis.zpp.blog.model.Post (katalog src/**main**/java) -- prosta klasa reprezentująca post na blogu
- Klasa pl.wwsis.zpp.blog.model.Comment (katalog src/**main**/java) -- reprezentuje komentarz do posta
- Klasa pl.wwsis.zpp.blog.model.PostTest (katalog src/**test**/java) -- testy jednostkowe do klasy Post. Uruchom testy i zapoznaj się z kodem -- kolejne ćwiczenie będzie polegało na napisaniu podobnych testów.

ZADANIE 2	Testowanie różnych implementacji podklas klasy Post

W pakiecie pl.wwsis.zpp.blog.model.alternatives znajduje się interfejs PostApi, określający rozszerzone API posta:

- metoda **getTags** zwraca zbiór tagów przypisanych do posta. Tag to prosta etykieta tekstowa.
- metoda **addTag** dodaje tag do posta, tzn. gdy dodamy jakiś tag do posta to powinien być on elementem zbioru zwracanego przez **getTags.** Dodatkowo nie są dozwolone duplikaty, tzn. nawet gdy wywołamy metodę **addTag** kilkakrotnie z tym samym tagiem, to i tak zostanie on dodany tylko raz. Co więcej każdy post może mieć maksymalnie 5 tagów -- gdy dodanie zostany piąty, to kolejne wywołania metody **addTag** nie mają żadnego efektu.
- metoda **getLastComments** zwraca listę 3 najnowszych (wg daty) komentarzy. Są one dodatkowo posortowane od najnowszego do najstarszego.

W pakiecie pl.wwsis.zpp.blog.model.alternatives (katalog src/main/java) znajduje się 8 implementacji opisanego wyżej interfejsu (klasy Post1...Post8). Tylko jedna z nich jest poprawna, tzn. spełnia opisane powyżej wymagania.

Zadanie polega na napisaniu zestawu testów takiego że:

- nie przejdzie go żadna z niepoprawnych implementacji (tzn. dla każdej z niepoprawnych klas nie przejdzie przynajmniej jeden test)
- poprawna implementacja przejdzie wszystkie testy

Klasa pl.wwsis.zpp.blog.model.alternatives.PostSpecificationTest (katalog src/**test**/java) zawiera szkielet klasy testowej. W linii 13 tworzony jest post który należy testować. Zmiana nazwy klasy (z Post1 na np Post2) pozwoli na uruchomienie testów dla innej wersji.

Przydatne fragmenty kodu:

Wyciąganie elementu z kolekcji	Comment c = post.getComments().get(0);	zmienna c zawiera pierwszy element z kolekcji komentarzy danego posta
	Comment c = post.getComments().get(1);	zmienna c zawiera drugi element z kolekcji komentarzy danego posta
Sprawdzenie czy kolekcja zawiera element	post.getTags().contains(“abc”)	zwróci true gdy kolekcja tagów posta zawiera tag “abc”

Negacja warunku logicznego	!post.getTags().contains(“abc”)	zwróci true gdy kolekcja tagów posta NIE zawiera tagu abc
Sprawdzenie rozmiaru kolekcji	post.getTags().size()	zwróci liczbę tagów posta
	post.getTags().size() == 3	zwróci true gdy post ma 3 tagi
Porównywanie dat	comment1.getDate().after(comment2.getDate())	zwróci true gdy comment1 ma datę późniejszą niż comment2
	comment1.getDate().before(comment2.getDate())	zwróci true gdy comment1 ma datę wcześniejszą niż comment2