Studia Podyplomowe: Java EE - produkcja oprogramowania

Produkcja i testowanie (PTE)
TDD Test-driven development
Wytwarzanie oprogramowania sterowane testami

(do użytku wewnętrznego)

Anna Derezińska,
Instytut Informatyki, Politechnika Warszawska

Semestr zimowy 2019/20
Grupa: JA19Z (zesp. JA1a, JA1b, JA1c) - semestr pierwszy

1

Literatura

1. K. Beck TDD. Sztuka tworzenia dobrego kodu, Helion 2014 (K. Beck, *Test Driven Development by Example*, Addison-Wesley, 2002)

2

2

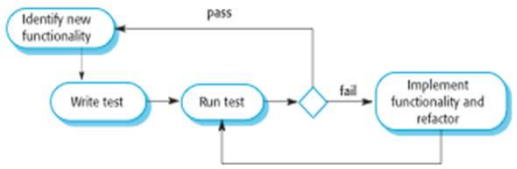
TDD

- TDD – podejście do wytwarzania programu, w którym przeplata się testowanie z tworzeniem kodu.
- Testy są pisane przed kodowaniem. Pozytywne uruchomienie testów jest podstawą sterowania procesem wytwarzania oprogramowania.
- Kod jest tworzony przyrostowo razem z testami dla danego przyrostu. Nie rozpoczynamy kodowania kolejnej partii kodu zanim aktualnie tworzony fragment kodu nie przejdzie pozytywnie swoich testów.
- TDD został spopularyzowany jako część zwinnej (ang. *agile*) metodyki wytwarzania oprogramowania, takiej jak Extreme Programming. Jest jednak używane w różnych procesach wytwarzania oprogramowania.

3

3

Test-driven development



4

4

Programowanie sterowane testami

1. Przygotowanie testu do weryfikacji nowej funkcjonalności, W razie potrzeby korzystamy z dublerów kodu (zaślepki, obiekty pozorne).
2. Uruchomienie testów
3. Implementacja (tylko kod już weryfikowany przez testy)
4. Uruchomienie testów
5. Rozwój funkcjonalności i/lub refaktoryzacja kodu
6. Ponowne uruchomienie testów
7. Powrót do p.5 lub decyzja o zakończeniu tej części funkcjonalności do p.1

5

5

TDD oczekiwane wyniki testów

1. Tworzymy testy jednostkowe.
2. Uruchamiamy testy-> oczekiwany błąd bo brak relizacji kodu — fail
3. Tworzymy fragment kodu
4. Uruchamiamy testy -> oczekiwane — pass
5. Ewolucja kodu, refaktoryzacja
6. Ponowne uruchomienie testów-> oczekiwane — pass
7. Powrót do p.5 lub koniec i przejście do p.1

6

6

Zalety TDD

- Pokrycie kodu
  - Z każdym fragmentem tworzonego kodu jest związany co najmniej jeden test, czyli każdy fragment kodu jest pokryty przez test (ale nie koniecznie każda linia czy każdy warunek .
- Testy regresyjne
  - Zbiór testów regresyjnych jest tworzony inkrementalnie razem z rozwojem programu.
- Ułatwione debuggowanie
  - Gdy test nie przechodzi, powinno być oczywiste gdzie jest problem. Nowo dodany kod powinien być sprawdzony i modyfikowany.
- Dokumentacja systemu
  - Testy same mogą tworzyć dokumentację systemu, opisującą co kod powinien robić.

7

7

Przykład - TDDExample

- Tworzymy stos liczb, rozpoczynając od pustego stosu
- W kolejnych krokach rozszerzmy jego funkcjonalność:
- odkładanie na stos *push()* i pobieranie *pop()*
- zabezpieczenie przed pobieraniem z pustego stosu
- ograniczenie rozmiaru stosu

8

8

Strategie czasu tworzenia testów

- Przed projektowaniem (z wymagań)
- Przed kodowaniem (z wymagań, z projektu), *test-driven development*
- Równolegle z kodowaniem
- Po kodowaniu jednostkowym
- Po scalaniu

9

Ćwiczenie D.1

- Kontynuacja przykładu TDD

1) Napisać test dla nowej funkcjonalności
2) Zaimplementować nową funkcjonalność w klasie *Stack*
3) Uruchomić i przetestować nową funkcjonalność

Stack.getSize() – zwraca liczbę elementów na stosie

UWAGA – mają też działać wszystkie dotychczasowe testy, (ewentualnie przy istotnych, uzasadnionych zmianach zaznaczyć wybrane testy jako @Disabled) i zastąpić nowymi

10

Ćwiczenie D.2

- Kontynuacja przykładu TDD

1) Napisać test dla nowej funkcjonalności
2) Zaimplementować nową funkcjonalność w klasie *Stack*
3) Uruchomić i przetestować nową funkcjonalność

Stack.clear() – czyści stos („usuwa" wszystkie elementy)

UWAGA – mają też działać wszystkie dotychczasowe testy, (ewentualnie przy istotnych, uzasadnionych zmianach zaznaczyć wybrane testy jako @Disabled) i zastąpić nowymi

11

Zadanie D.3 – praca domowa

- Kontynuacja przykładu TDD

1) Napisać test dla nowej funkcjonalności
2) Zaimplementować nową funkcjonalność w klasie *Stack*
3) Uruchomić i przetestować nową funkcjonalność

configurable Stack capacity – stos o zmiennej pojemności, możliwe są różne metody (np. powiększanie stosu o zadany przyrost, automatyczne powiększanie w przypadku braku miejsca na stosie, itp.)

UWAGA – mają też działać wszystkie dotychczasowe testy, (ewentualnie przy istotnych, uzasadnionych zmianach zaznaczyć wybrane testy jako @Disabled) i zastąpić nowymi

12