

## Kod szkolenia: P/TDD

## Tytuł szkolenia: Test-Driven Development w Javie

### Adresaci szkolenia:

Szkolenie adresowane jest do programistów Java, chcących podnieść swoje umiejętności tworzenia czystego, testowalnego kodu. Zapraszamy też znających Javę architektów i testerów, którzy chcą poznać zalety tworzenia kodu z pomocą testów.

### Cel szkolenia:

Uczestnicy poznają techniki tworzenia kodu wysokiej jakości.

W szczególności:

Uczestnicy dowiedzą się czym różnią się poszczególne rodzaje testów. Poznają i przyswoją sobie cykl pracy TDD, nauczą się projektować oprogramowanie pod względem testowalności i tworzyć czytelny kod. Poznają biblioteki ułatwiające stosowanie TDD oraz umożliwiające testowanie na różnych poziomach.

### Wymagania:

Od uczestników szkolenia wymagana jest umiejętność programowania w języku Java (do poznania na kursie J/JP)

### Mocne strony szkolenia:

Szkolenie prowadzone jest w formie warsztatów. Uczestnicy przyswajają wiedzę w najskuteczniejszy możliwy sposób – praktykując TDD podczas serii ćwiczeń. Po szkoleniu czują i rozumieją, czym TDD jest i jak je zastosować w pracy.

### Parametry szkolenia:

3\*7 godzin wykładów i warsztatów w proporcji 1/3.

Wielkość grupy: maks. 8-10 osób.

### Polecane szkolenia poprzedzające: J/JP

## Program szkolenia:

1. Podstawy
  - dlaczego TDD może Ci pomóc
  - stosowanie cyklu TDD Red-Green-Refactor
  - bezpieczne refaktoryzowanie kodu pokrytego testami
  - pełne wykorzystanie możliwości IDE w celu wspierania procesu TDD (generowanie kodu, refaktoryzacje automatyczne)
2. Mechanika TDD
  - jak wybrać następny test do zaimplementowania
  - nazewnictwo testów
  - zaawansowane możliwości JUnit
3. Testy współpracujących obiektów
  - wykorzystanie dublerów testowych (Test Doubles) celu izolacji klas testowanych
  - rodzaje Test Doubles: mocki, stuby
  - biblioteka Mockito - wygodny sposób generowania mocków w locie
  - tworzenie test-driven warstwy dostępu do danych
  - tworzenie test-driven interfejsu użytkownika (GUI)
4. Projektowanie obiektowe
  - testability, czyli projektowanie pod kątem testów
  - zasady wspierające dobry design (SOLID principles, Inversion of Control/Dependency Injection, powiązania, spójność)
5. Trochę szerszego spojrzenia
  - rodzaje i poziomy testów: jednostkowe, integracyjne, akceptacyjne...
  - TDD, a może BDD (Behavior-Driven Design) lub ATDD (Acceptance TDD)?
  - Podejście klasyczne vs. podejście mockowe (Classical and Mockist Testing)
6. Praca z odziedziczonym kodem (Legacy Code)
  - jak zrefaktoryzować kod, aby dało się go pokryć testem (rozcinanie zależności / Dependency Breaking)
  - zrozumienie kodu i zabezpieczenie miejsca zmiany dzięki testom charakterystycznym



### 7. Kontynuacja rozwoju

- wzorce stosowania TDD
- utrzymanie testów - jak sprawić, by służyły mi za rok równie dobrze jak dziś

