Program szkolenia:

Tworzenie i testowanie aplikacji z użyciem Test Driven Development

Informacje ogólne

Nazwa:	**Tworzenie i testowanie aplikacji z użyciem Test Driven Development**
Kod:	**TDD**
Kategoria:	Testowanie
Grupa docelowa:	devops, programiści, projektanci
Czas trwania:	3 dni
Forma:	40% wykłady / 60% warsztaty
Prowadzący:	Marcin Zajączkowski

Program szkolenia jest ogólną ramą - konkretne szkolenie poprzedzamy analizą przed-szkoleniową.

Zalety szkolenia:

Program szkolenia:

1. Jednostkowe testowanie kodu - niezbędnik do pisania dobrych testów w TDD

1.1. Dlaczego warto automatycznie testować kod

1.2. Dobre testy jednostkowe - FIRST

1.3. Struktura testów jednostkowych (given-when-then/arrange-act-assert)

1.4. Nazewnictwo testów (metod testowych)

1.5. Konstrukcje wspierające testowalność kodu (m.in. dziedziczenie -> kompozycja, małe klasy, wstrzykiwanie zależności)

1.6. Antywzorce dla testowalnego kodu (m.in. singletony, elementy statyczne, pola i metody final)

1.7. Elementy, których nie warto testować jednostkowo

1.8. Zadania frameworku testowego

1.9. Formułowanie naturalnie wyglądających asercji (AssertJ)

1.10. Ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

2. Spock Framework - pisanie testów szybciej, zwięźlej i czytelniej

2.1. Nowa jakość w testowaniu kodu - dlaczego warto poznać Spocka

2.2. Konstrukcje i techniki upraszczające kod testowy (Groovy w pigułce na potrzeby pisania testów)

2.3. Podział testu na bloki/sekcje

2.4. Testy parametryzowane

2.5. Testowanie wyjątków

2.6. Warunkowe wykonywanie testów

2.7. Inicjowanie i sprzątanie w testach

2.8. Porównanie trio JUnit/Mockito/AssertJ ze Spockiem

2.9. Ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

3. Test Driven Development - wprowadzenie

3.1. Historia TDD

3.2. Podstawowe założenia

3.3. Cykl red-green-refactor

3.4. Przejrzysta struktura testu

3.5. Wybór kolejnych funkcji do zaimplementowania

3.6. Sprawne uruchamianie testów z IDE (przydatne wtyczki, skróty klawiaturowe)

3.7. Programowanie w parach

3.8. Ćwiczenia praktyczne z TDD w formie Kata/Coding Dojo

3.9. Korzyści ze stosowania TDD

4. Stosowanie separacji od obiektów współpracujących przy TDD

4.1. Potrzeba, kiedy i dlaczego warto

4.2. Ttestowe zastępniki obiektów współpracujących (ang. test doubles)

4.3. Mockowanie ze Spock Framework

4.4. Mockowanie z Mockito (mocki w Spocku mają swoje ograniczenia)

4.5. Ćwiczenia praktyczne - TDD przy jednoczesnym zastosowaniu frameworku do testowania

4.6. Testowanie integracyjne aplikacji opartej o Spring Framework - temat opcjonalny

4.7. Dlaczego tylko testy jednostkowe nie wystarczą

4.8. Wsparcie w JUnit/TestNG/Spock

4.9. Konfiguracja i zarządzanie kontekstem Springa

4.10. Wstrzykiwanie zależności

4.11. Odcinanie zależności w kontekście Springa (wstrzykiwanie zależności testowych, w tym mocków)

4.12. Ręczne tworzenie kontekstu w teście (do testowania specyficznych przypadków)

4.13. Cache'owanie kontekstu między testami (i problemy z tym związane)

4.14. Wydzielanie wspólnej konfiguracji dla wielu testów

4.15. Ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

5. Test Driven Development - dobre praktyki i techniki unikania typowych problemów

5.1. Wykorzystanie TDD do pracy nad dobrym designem systemu

5.2. Wybór właściwego poziomu odcięcia zależności

5.3. Zestaw ćwiczeń praktycznych uwypuklających typowe problemy spotykane przy tworzeniu kodu produkcyjnego i testów z pomocą TDD oraz pokazanie sposób ich unikania/rozwiązywania

6. Testowanie - zaawansowane zagadnienia

6.1. Techniki testowania asynchronicznych wywołań (Awaitility, Spock)

6.2. Badanie jakości testów z testowaniem mutacyjnym (PIT)

6.3. Jednostkowe testowanie elementów bazujących na dacie

6.4. Techniki przyśpieszania testów

6.5. Uproszczenie niektórych aspektów testowania z Java 8