TWORZENIE I TESTOWANIE APLIKACJI Z UŻYCIEM TEST-DRIVEN DEVELOPMENT

KOD: TDD

bns it

PROFIL UCZESTNIKA

Programista:

- zna język Java lub C# w stopniu podstawowym (istnieje możliwość dopasowania szkolenia do innego języka);
- chce zwiększyć niezawodność tworzonego oprogramowania;
- chce używać technik TDD podczas programowania.

KORZYŚCI ZE SZKOLENIA

1. **Poprawia się bezpieczeństwo tworzonego oprogramowania** – dzięki technikom TDD, które skłaniają programistów do testowania kodu na najbardziej elementarnym poziomie, maleje ilość błędów w aplikacji.
2. **Całkowity czas poświęconej na tworzenie i utrzymanie funkcjonalności ulega skróceniu** – dzieje jest tak, ponieważ dzięki tworzeniu testu przed implementacją pisany jest tylko niezbędny kod. Poszukiwanie błędów zajmuje mniej czasu, gdyż istnieją testy jednostkowe.
3. **Rozbudowa aplikacji jest mniej kosztowna** – stosowanie TDD wymusza dobrą jakość kodu źródłowego, m.in. stosowanie wzorców projektowych. Sprawia to, że dodawanie nowych funkcjonalności jest mniej pracochłonne. Dodatkowo istnienie testów jednostkowych czyni proces rozbudowy aplikacji bezpiecznym.
4. **Zwiększa się zaangażowanie programistów** – dzięki skierowaniu ich uwagi na cele biznesowe tworzonego oprogramowania.

PARAMETRY SZKOLENIA

Czas trwania: 3 dni - 24 godziny.
Forma zajęć: Laboratorium TDD - 60%, wykład – 40%.
Wielkość grupy: do 10 osób.
Języki programowania: Java, C#.

SZCZEGÓŁOWY PROGRAM

Moduły szkoleniowe	Nabyte wiedza i umiejętności, poruszane zagadnienia
Wprowadzenie do TDD	• Cykl programowania • Cykl TDD red-green-refactor • Programowanie przyrostowe • Zasady tworzenia testów jednostkowych • Przykład pracy z użyciem TDD
xUnit jako narzędzie testowania	• Tworzenie testów jednostkowych z użyciem xUnit • Wybrane atrybuty konfiguracji xUnit • @Test [Test] • @Before [SetUp] • @After [TearDown] • @Expected [ExpectedException] • @Ignore [Ignore] • Wybrane asercje xUnit • Assert.that • Testowanie pozytywne • Testowanie negatywne • Testowanie wyjątków • Wzorce TDD • Co testować • Testowanie stanu • Testowanie zachowania
Wzorce testowania xUnit	• Asercja stanu końcowego • Asercja pomocnicza • Asercja delta

	• Asercja własna • Asercja interakcji • Metoda fabryki • Klasy pomocnicze • Matka obiektów • Testy parametryzowane • Autopodstawianie • Uprzywilejowany dostęp • Dodatkowy konstruktor • Podklasa na potrzeby testu

Zasady TDD	• Strategie testowania
	  o Ogół - szczegół
	  o Znane - nieznane
	  o Ścieżka pozytywna – negatywna
	• Strategie implementacji
	  o Faking it
	  o Traingulation
	  o Obvious implementation
	• Pojęcia TDD
	  o Fixture
	  o Test doubles (Stubs/Fakes/Mocks)
	  o Testowanie stanu i interakcji

Refaktoring w kontekście TDD	• Refaktoryzacje przydatne w TDD • Elementy refaktoryzacji do wzorców projektowych • Paradygmaty testowalnego kodu • Refaktoryzacja testów jednostkowych • Usuwanie redundancji w testach

Testowalny kod	• Kompozycja i dziedziczenie a TDD • Elementy statyczne i singletony • Izolowanie i wstrzykiwanie zależności • Architektura warstwowa
Mockito/Moq jako narzędzie do tworzenia mocków	• Cykl życia mocka w Mockito/Moq • Testowanie zachowania • Stubbing • Weryfikacja ilości wywołań
Testowanie end-to-end	• Testowanie akceptacyjne • Testowanie end-to-end • Jakość zewnętrzna i wewnętrzna • Dobre praktyki testowania end-to-end
Behaviour-Driven Development	• Wprowadzenie do BDD • Testowanie zachowania • BDD a User Stories • Specyfikacja poprzez przykłady
Testowanie integracyjne	• Testowanie dostępu do danych • Narzędzie xDbUnit • Testowanie transakcji
Testowanie z udziałem komponentów zewnętrznych	• Strategie testowania z udziałem komponentów zewnętrznych • Warstwa adapterów
Warsztat praktyczny	• Inkrementacyjny rozwój przykładowego systemu z użyciem TDD