

TWORZENIE I TESTOWANIE APLIKACJI Z UŻYCIEM TEST-DRIVEN DEVELOPMENT

KOD: TDD

Profil uczestnika

Programista:

- zna język Java lub C# w stopniu podstawowym;
- 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 programowania ulega skróceniu** – dzieje się 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	<ul style="list-style-type: none">• Cykl programowania• Cykl TDD red-green-refactor• Programowanie przyrostowe• Zasady tworzenia testów jednostkowych• Przykład pracy z użyciem TDD
xUnit jako narzędzie testowania	<ul style="list-style-type: none">• 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
Refaktoring w kontekście TDD	<ul style="list-style-type: none">• Refaktoryzacje przydatne w TDD• Elementy refaktoryzacji do wzorców projektowych• Paradygmaty testowalnego kodu• Refaktoryzacja testów jednostkowych• Usuwanie redundancji w testach

Zaawansowane aspekty TDD

- Testowanie z użyciem test doubles
- Mock Objects
- Rozwijanie istniejących systemów (legacy) z użyciem TDD
- Testowanie usług dostarczonych przez kontener Spring
- Techniki tworzenia testów integracyjnych (DbUnit)
- Testowanie współpracy z baza danych (JDBC, Hibernate, JPA)
- Testowanie aplikacji webowej (serwlety, JSP, akcje Struts)
- Tworzenie testów akceptacyjnych

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