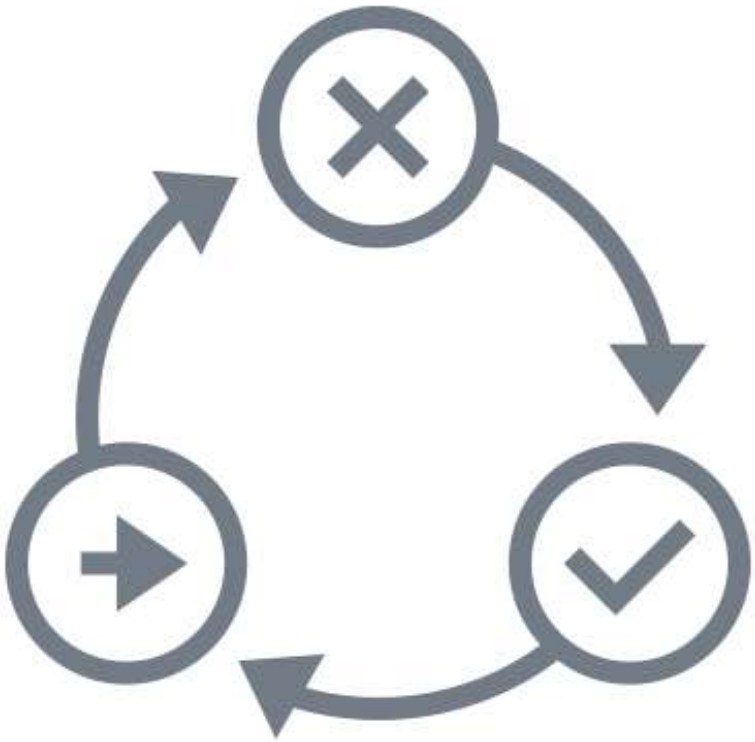
Eğitimin içeriği

- Neden test ederiz
- Test aşamaları
- Unit Test
- Unit Test yazma kuralları
- En sık kullanılan Java Unit Test araçları

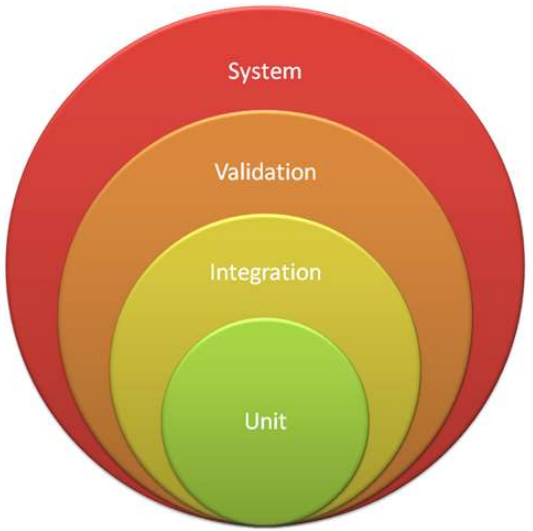


Neden Test Ederiz

- Gereksinimlerin karşılanması
- Kalite Kontrol
- Güvenlik
- Multi Device
- Zaman - Hız
- Maliyet
- Performans

Test Aşamaları

- Unit Test (Birim testi)
- Integration Test (Entegrasyon testi)
- Regression Test (Regresyon testi)
- UI Test (Arayüz testi)
- Performance Test (Performans testi)
- User Acceptance Test (Kullanıcı kabul testi)



Unit Test (Birim Testi)

- Yazılımın en küçük parçasının test eder
- Her parça bağımsız test edilir
- Yazılımcı tarafından yazılır
- Dokumantasyon sağlamak
- Çabuk geri bildirim alabilmek
- Kolay analiz ve refactoring yapma imkanı
- Birim testleri tüm hataları bulamaz

Unit Test yazma kuralları

- En küçük parçacığı test etmeli
- Sadece bir senaryo test edilir
- Kullanilan adımlar Given - When - Then
- Test method ismi test edilen senaryoyu yansıtmalı
- Test edilen parça diğer parçalardan bağımsız olmalı (mock, stub)
- Diğer test methodlarından bağımsız çalışabilmeli
- Tam otomatik

Unit Test yazma kuralları

- Hızlı çalışabilmeli ve çabuk sonuçlar vermeli
- Okunaklı, anlaşılabilir ve sürdürülebilir olmalı
- Her zaman aynı sonucu çıkarmalı

Farklı Method İsimleri

- test[Özellik ismi] (Örnek - testPozitifSayılarıToplayıncaPozitifÇıkar)
- Sadece Özellik ismi (Örnek - pozitifSayılarıToplayıncaPozitifÇıkar)
- Should_BeklenenDavranış_When_Koşul
  (should_PozitifSayıÇıkarır_When_PozitifSayıVerilirse
- When_Koşul_Expect_BeklenenDavranış
  (When_PozitifSayıVerilirse_Expect_PozitifSayıÇıkartır)
- Given_ÖnHazırlık_When_Koşul_Then_BeklenenDavranış
  (given_BirHesapMakinesi_When_PozitifSayıVerilirse_ThenPozitifSayıÇıkartır)
- Method_Senaryo_Sonuç (Örnek - topla_PozitifSayılar_Pozitif)
- Method_Sonuç_Senaryo (Örnek - topla_Pozitif_PozitifSayılar)

Java Unit Test Araçları

- JUnit (http://junit.org)
- Hamcrest (http://hamcrest.org)
- AssertJ (http://joel-costigliola.github.io/assertj/)
- Mockito (http://mockito.org)

JUnit

- Yaşam döngüsü
- Assertions (iddialar)
- Parametreler
- Hata test etme
- Suite (Gruplayarak test etme)
- Ignore (Devre dışı bırakmak)

JUnit Yaşam Döngüsü

- @BeforeClass
- @Before
- @Test
- @After
- @AfterClass

JUnit Assertions

- org.junit.Assert.*
- assertEquals - assertSame
- assertNull - AssertTrue
- assertArrayEquals

JUnit Parametreler

- @RunWith(Parameterized.class)
- @Parameters
- JUnitParams @Parameters

Hata Test Etme

- try catch
- @Test - expected
- @Rule
- CatchException framework

Hamcrest

- http://hamcrest.org/JavaHamcrest
- JUnit eklentisi
- Fluent interface (akıcı arayüzü)
- assertThat(value, matcher)

AssertJ

- http://joel-costigliola.github.io/assertj/
- Fluent assertions (Akıcı arayüzü)
- Çok geniş eşleştirme kütüphanesi
- Kendi assert yapma imkanı

Mock nedir?

- Gerçek sınıfı birebir taklit eden nesne
- İstediğiniz gibi hareket etmesini sağlayabilirsiniz
- Gerçek - Mock beraber çalışabilir (Spy)

Neden Mock?

- Birim testi sadece bir birimi test eder
- Bağımılıkların hiç bir yan etkisi olmaması gerekli
- Bir senaryo için bağımlılıkları doğru bir şekilde yönlendirme
- Gerçek nesneler ortaya beklenmedik durumlar çıkartabilir
- Gerçek nesneler yavaştır
- Gerçek nesneler tekrar edilebilir değildir
- Gerçek nesneler zor yapılandırılabilir

Java Mock kütüphaneleri

- Mockito (http://mockito.org/)
- JMock (http://www.jmock.org/)
- PowerMock (https://code.google.com/p/powermock/)
- EasyMock (http://easymock.org/)

Mockito

- http://mockito.org/
- En çok kullanılan mock kütüphanelerinden biri
- Kolay ve çabuk öğrenebilinir
- Geniş kullanım yöntemleri var
- Annotation desteği