JUNIT4 VE JUNIT 5 İLE UNIT TEST

Unit Test Nedir?

Unit test ; yazlımın birimlerini test etmek olarak tanımlayabiliriz. Birim olarak tanımladığımız yapılar method yada methodun için yer alan kod bloğu olabilir. Unit testler kullanıcılar için değildir yazlımcılar için kodun test edilmesidir.

Bir unit test metodu genellikle üç aşamadan oluşur. Bu aşamalar yabancı kaynaklarda **The AAA(Arrange-Act-Aspect) Pattern** olarak geçer.

Arrange: Test edilecek koda verilecek olan input parametrelerinin belirlendiği ve test edilecek olan kodun bağımlı olduğu diğer bileşenlerin test anındaki bulunacakları durumlarının tanımlandığı kısımdır.

Act: Test edilecek olan kodun çalıştırıldığı aşamadır. Bu aşamada test edilecek olan fonksiyonu/metodu tetikleriz.

Assert: Test sonuçlarının doğrulanması aşamasıdır. Tetiklenen fonksiyon doğru sonucu üretiyor mu veya bağımlı olduğu bileşenler üzerinde beklenen aksiyonları tetikliyor mu kontrolünü bu aşamada yaparız.

Unit testleri **“Assert” fazında** doğrulayacağı davranışa göre iki kategoriye ayrılır;

state-based: Test edilen kodun çıktılarının veya sistemde oluşturduğu durum (state) değişikliğinin kontrol edilmesi durumunda “state-based” test yazmış oluyoruz.

interaction-based: Test edilen kodun belirli fonksiyonları doğru şekilde tetiklediğini (doğru etkileşim) doğrulayan bir test yazdığımızda “interaction-based” bir test yazmış oluyoruz.

Bazı Unit Test Framework’ leri;

1. Robot Framework
2. JUnit Spock
3. NUnit
4. TestNG
5. Jasmin
6. Mocha

Unit Test nasıl yazılır?

1. Unit Test yazmak için de bazı kurallara uymamız gerekiyor bu kurallar;
2. En küçük parçacığı test edilmeli Sadece bir senaryo test edilir.
3. Kullanılan adımlar belirlenir. Test method ismi test edilen senaryonun yansıması olmalıdır.
4. Test edilen kısım diğer kısımlardan bağımsız olmalıdır.
5. Testlerimiz tam otomatik şekilde çalışmalıdır.
6. Hızlı çalışabilmeli ve çabuk sonuçlar vermelidir.
7. Okunaklı, anlaşılabilir ve sürdürebilir olmalıdır.
8. Test başarısız olduğunda durmalı ve iyi bir hata raporu döndürmelidir. Bu hata raporunda neyi test ettin ? ne yapmalı ? beklenen çıktı neydi ve gerçekte ne yaptığıdır ?

JUNIT 4 & JUNIT 5

JUnit 5, Java 8 kodlama stilini uyarlamayı ve JUnit 4'ten daha sağlam ve esnek olmayı amaçlamaktadır.

JUNIT 4 & JUNIT 5 arasındaki farklar

Feature	JUnit 4	Junit 5
Declare a test method	`@Test`	`@Test`
Execute before all test methods in the current class	`@BeforeClass`	`@BeforeAll`
Execute after all test methods in the current class	`@AfterClass`	`@AfterAll`
Execute before each test method	`@Before`	`@BeforeEach`
Execute after each test method	`@After`	`@AfterEach`
Disable a test method/class	`@Ignore`	`@Disabled`
Test factory for dynamic tests	NA	`@TestFactory`
Nested tests	NA	`@Nested`
Tagging and filtering	`@Category`	`@Tag`
Register custom extensions	NA	`@ExtendWith`

Mimari

JUnit 4, tek bir jar dosyasında paketlenmiş her şeye sahiptir.

JUnit 5, JUnit Platform, JUnit Jupiter ve JUnit Vintage olmak üzere 3 alt projeden oluşmaktadır.

JDK Gereksinimleri

JUNIT 4 java 5 ve üzeri versiyonlarda çalışır

JUNIT 5 Java 8 ve üzeri versiyonlarda çalışır

Assertions

Junit 4'te, org.junit.Assert, beklenen ve ortaya çıkan sonuçları doğrulamak için tüm onaylama yöntemlerine sahiptir. Method signatur için İLK argüman olarak hata mesajları için ekstra parametreleri kabul ederler. Örneğin;

```
public static void assertEquals(long expected, long actual)
public static void assertEquals(String message, long expected, long actual)
```

JUnit 5'te, **org.junit.jupiter.Assertions**, ,çoğu assert() yöntemini içerir ayrıca assertThrows() ve assertAll() yöntemlerini de içerir.

JUnit 5 assretions method lar ayrıca, testin başarısız olması durumunda yazdırılacak hata mesajlarının ayrıştırılmasını desteklemek için overload method lara sahiptir. Örneğin;

```
public static void assertEquals(long expected, long actual)
public static void assertEquals(long expected, long actual, String message)
public static void assertEquals(long expected, long actual, Supplier
messageSupplier)
```

Assumptions

JUNIT 4'te, org.junit.Assume, bir testin anlamlı olduğu koşullar hakkında varsayımları belirtmek için yöntemler içerir. Aşağıdaki beş yönteme sahiptir:

1. assumeFalse()
2. assumeNoException()
3. assumeNotNull()
4. assumeThat()
5. assumeTrue()

JUNIT 5'te, org.junit.jupiter.api.Assumptions, bir testin anlamlı olduğu koşullar hakkında varsayımları belirtmek için yöntemler içerir. Aşağıdaki üç yönteme sahiptir:

1. assumeFalse()
2. assumingThat()
3. 3.assumeTrue()

Tag Kullanımı

Junit 4'te @category ek açıklaması kullanılır.

Haziran 5'te @tag ek açıklaması kullanılır.

Test Kullanımları

JUNIT 4'te @RunWith ve @Suite ek açıklaması. Örneğin.

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
        ExceptionTest.class,
        TimeoutTest.class
})
public class JUnit4Example
{
}
```

JUNIT 5'te, @Suite, @SelectPackages ve @SelectClasses. Örneğin;

```
import org.junit.platform.runner.JUnitPlatform;
import org.junit.platform.suite.api.SelectPackages;
import org.junit.runner.RunWith;

@Suite
@SelectPackages("com.howtodoinjava.junit5.examples")
public class JUnit5Example
{
}
```

Public Olmayan Metodlara İzin Verir

- JUnit 5 test sınıfları ve test metotlarının public olması gerekli değildir. Onları paket korumalı hale getirebiliriz.
- JUnit, test sınıflarını ve test metotlarını bulmak için dahili olarak Reflection kullanır. Reflection, sınırlı görünürlükleri olsa bile onları keşfedebilir, bu nedenle public olmalarına gerek yoktur.
- JUnit test sınıfları ayrıca genel olmayan kuruculara sahip olabilir. Hatta tartışabilirler. Bu, JUnit 5'te genel tartışmasız bir kurucuya sahip olmanın zorunlu olmadığı anlamına gelir.

```
class AppTest {

    private AppTest(TestInfo testInfo) {
        System.out.println("Working on test " + testInfo.getDisplayName());
    }

  @Test
  void test(){
        assertTrue(true);
    }

}
```

3.Parti Entegrasyon

Junit 4'te 3. taraf eklentiler ve IDE'ler için entegrasyon desteği yoktur. Reflection a güvenmek zorundalar.

JUnit 5'in bu amaç için özel bir alt projesi vardır, yani JUnit Platformu. Platformda çalışan bir test fremework geliştirmek için TestEngine API'sini tanımlar.

Yararlanılan Kaynaklar;

https://howtodoinjava.com/junit5/junit-5-vs-junit-4/#

https://serdarkuzucu.com/2021/05/18/unit-test-nedir/

https://medium.com/@cengizhandumlu.35/unit-testing-nedir-ve-nas%C4%B1l-yaz%C4%B1l%C4%B1r-446073767e60