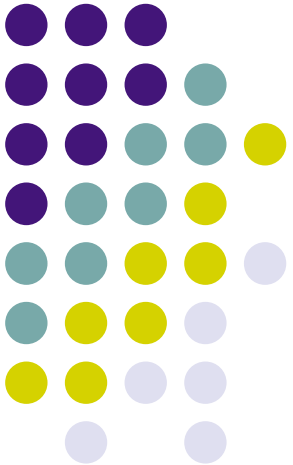
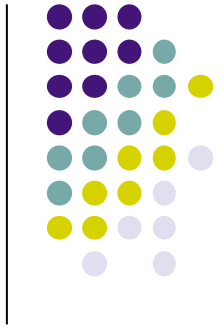
TDD
Test Driven Development

Melih Sakarya

www.sahabt.com
melih.sakarya@sahabt.com
https://github.com/melihsakarya

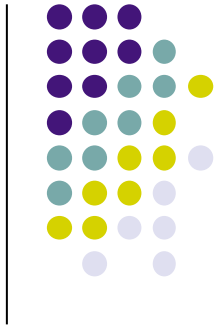


Neden Test



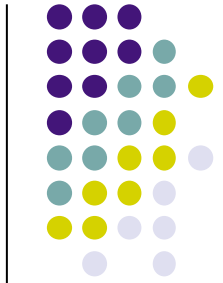
- Kaliteli bir yazılım çıktısı
- Uygulama gereksinimleri karşılıyor mu ?
- Doğru çıktıyı üretiyor mu ?
- Beklenen ortamlarda çalışıyor mu ?
- Günvenli mi ?
- Yoğun yük altında beklentileri karşılıyor mu ?
- Kaynak ihtaçları nedir ?

Test Tipleri

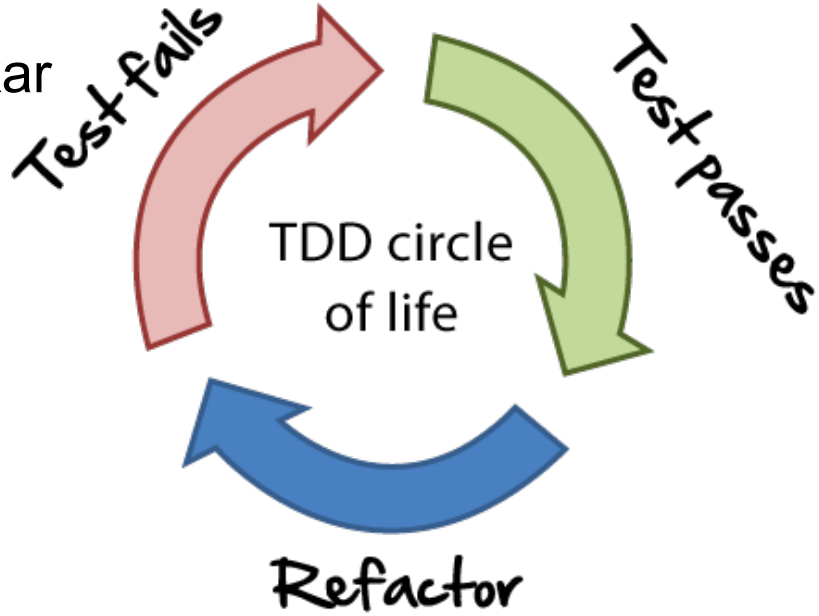


- Birim testleri
- Entegrasyon testleri
- Fonksiyonel testler
- Kabul testleri
- Yük testleri
- Performans testleri
- Güvenlik testleri
- Platform ve uyumluluk testleri
- Kullanılabilirlik testleri

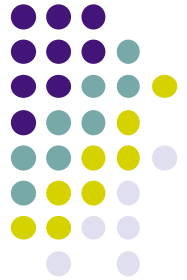
TDD
Test Driven Development



- Test yönelimli programlama
- Önce test sonra kod
- Gereksinimlerle testler belirlenir
- Testler ile uygulama ortaya çıkar
- Hatalar geliştirme anında ortaya çıkar

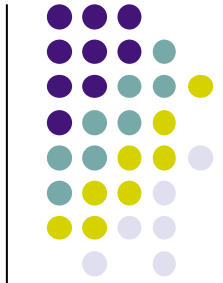


Unit Test – Birim Testi



- Bir birimi test etmeyi sağlar
- Tek birim testinin farklı fonksiyonları test etmesi beklemez
- Refactoring sürecindeki hataları minimize eder

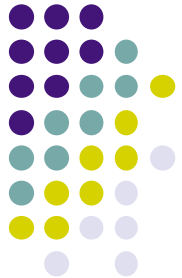
JUnit



- Java platformunda birim testleri için kullanılır.
- Hata durumunda geliştirme ortamında gözlemlenebilir.
- Test metodları @Test notasyonu ile belirlenir.

```
public class OrnekTest {

	@Test
	public void ornekTest(){
		fail("hata olustu");
	}
}
```

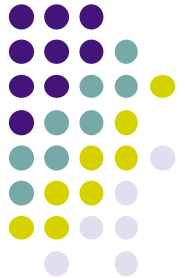
Assert Sınıfı



- Test akışında beklenen durumu sorgulamak için kullanılır.
- Beklenen durum karşılanmazsa hata verir ve testi sona erdirir.

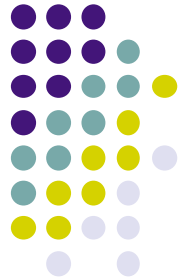
```
assertEquals(5, mat.topla(2,3));
assertTrue(true);
assertFalse(false);
assertNotEquals("aa", "bb");
```

Assert Kullanımı



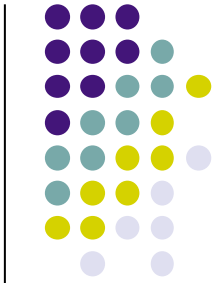
```
public class OrnekTest {

        @Test
        public void ornekTest(){
                Matematik mat = new Matematik();
                assertEquals(5, mat.topla(2,3));
        }
}
```

JUnit Yaşam Döngüsü



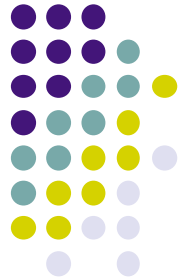
- Junit çalışma anında sınıf öncesi veya sonrası yapılması gereken işlemler tanımlanabilir.
- Annotation tanımları ile yapılır.
  - @Test
  - @BeforeClass
  - @AfterClass
  - @Before
  - @After

JUnit Yaşam Döngüsü



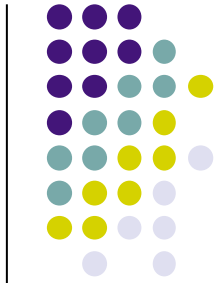
```
@BeforeClass
public static void sinifOncesi(){
}

@Before
Public void metodOncesi(){
}

@After
public void metodSonrasi(){
}

@AfterClass
Public static void sinifSonrasi(){
}
```

JUnit Yaşam Döngüsü



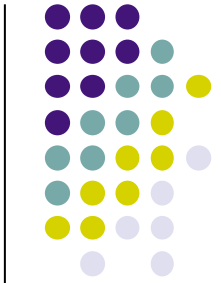
- Testlerin birbirlerine bağımlığı yoktur.
- Testler entegrasyonlardan yalıtılmış olmalıdır.

Parametreler



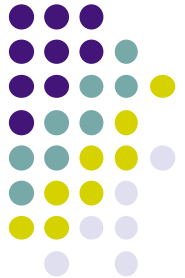
- Testler farklı değerler ile test edilmek istenebilir.
- Bu durumda her data için yeni bir karşılaştırma yapmak yerine parametrik hale getirilebilir.

Paremetreler



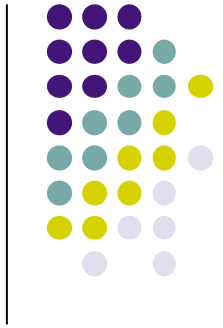
```
@RunWith(Parameterized.class)
public class OrnekTest {

        @Parameters
        public static String[][] getParameters(){
                String[][] parametreler = {
                        {"Melih", "Sakarya"},
                        {"Melih", "Sakarya"}
                };
        return parametreler;
        }

        ...
}
```

Paremetreler



```
@RunWith(JUnitParamsRunner.class)
public class DataDrivenTest {

        UserService userService;

        @Before
        public void init(){
                this.userService = new UserService();
                userService.emailService = Mockito.mock(EmailService.class);
        }



        @Test
        @Parameters({"Melih, Sakarya, melih.sakarya@gmail.com”})
        public void dataDrivenTest(String name, String lastname, String email){
                userService.register(name, lastname, email);
        }
}
```

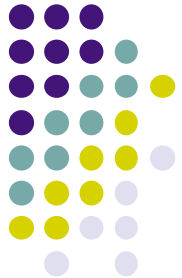
Test Suite



- Testleri bir grup olarak çalıştırmak için kullanılır.
- Sıralama dikkate alınır.
- Proje içerisinde Suite ve testler tekrar çalıştırılır.

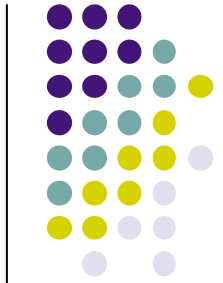
```
@RunWith(Suite.class)
@SuiteClasses({
        HavaleTest.class,
        BildirimTest.class
})
public class OrnekTestSuite {

}
```

Mock Objeler



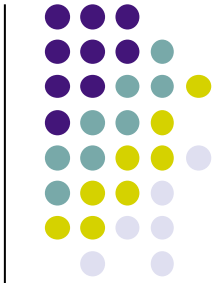
- Taklit nesne anlamına gelir.
- Nesnenin gerçek değil sahte bir kopyasını oluşturur.
- Bağıl nesnelerin davranışlarına ihtiyaç duymaz.
- Nesne davranışları kontrol edilebilir.

Mock Objeler



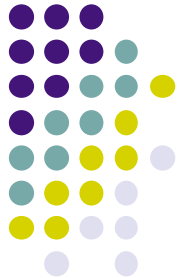
```
Hesap hesap;
Bildirim bildirim;

@Before
public void baslangic(){
        hesap = new Hesap();
        bildirim = Mockito.mock(Bildirim.class);
        hesap.setBildirim(bildirim);
}
```

Mock Objeler



```
@RunWith(MockitoJUnitRunner.class)

@InjectMocks
Hesap hesap;

@Mock
Bildirim bildirim;

@Before
public void baslangic(){

}
```

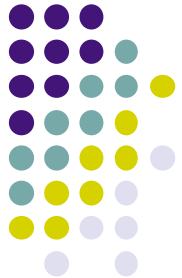
Mock Objeler - Kontroller



- Mock objeler içerisinde çağırım kontrolleri yapılabilir.

```
verify(bildirim).bildirimYap(anyString());
verify(bildirim).bildirimYap(startsWith("para"));
verify(bildirim, times(1)).bildirimYap(anyString());
verify(bildirim, never()).ornekMetod();
verify(bildirim).ornekMetod();
```

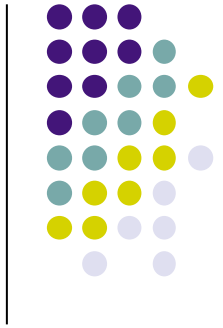
Mock Objeler –Kontroller



- Mock nesneler gerçek olmadığından metod davranışları default değerlerini döner.
- Akışın herhangi bir yerinde istediğimiz durumu gerçekleştirmek isteyebiliriz.

```
when(bildirim.bildirimYap(anyString())).thenReturn(true);
when(bildirim.ornekMetod()).thenReturn("havale");
```

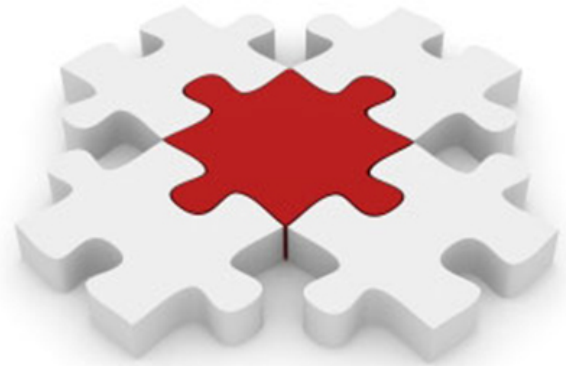
Spy Tanımlar



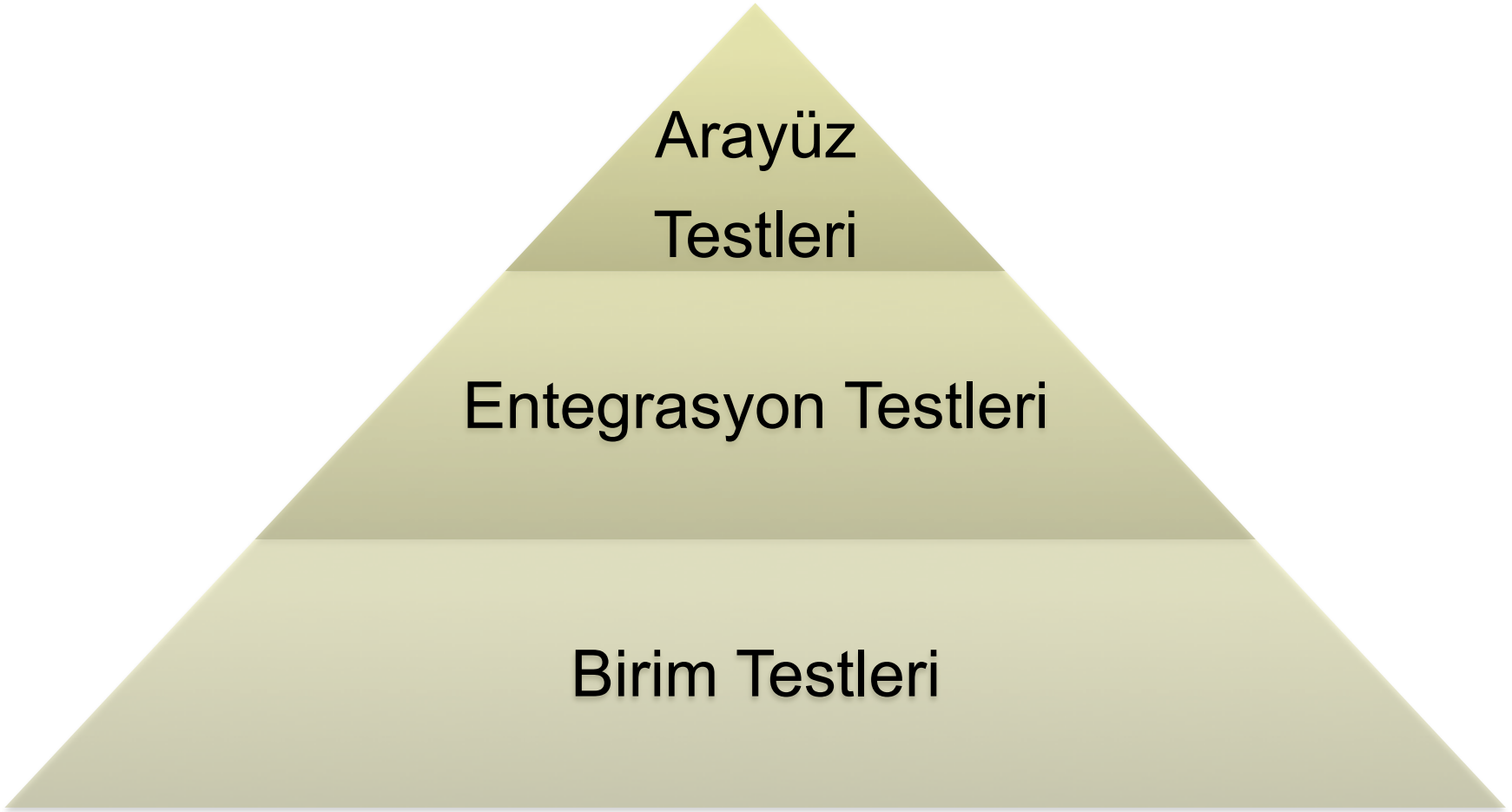
```
@InjectMocks
UserService userService;
@Spy
EmailService mailService;


@Test
public void userRegisterTest(){
        userService.register("Melih", "Sakarya",
                                        "melih.sakarya@gmail.com");
        verify(mailService, times(1)).sendEmail(anyString());
}
```

Entegrasyon Testleri

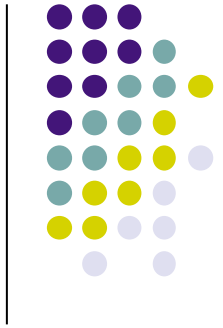
- Test birim değil alt bileşenlerle birlikte tasarlanır.
- Testleri uçtan uca diğer bileşenlerin davranışları ile test eder.
- Birim testlere göre doğrulama daha yüksektir.
- Birim testlere göre çalışması daha uzun ve masraflıdır.
- Sürekli çalıştırılması geliştirme sürecinde zaman kaybına neden olabilir.



Test Yoğunluğu

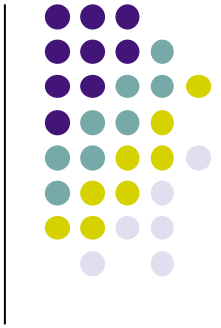


Entegrasyon Testleri

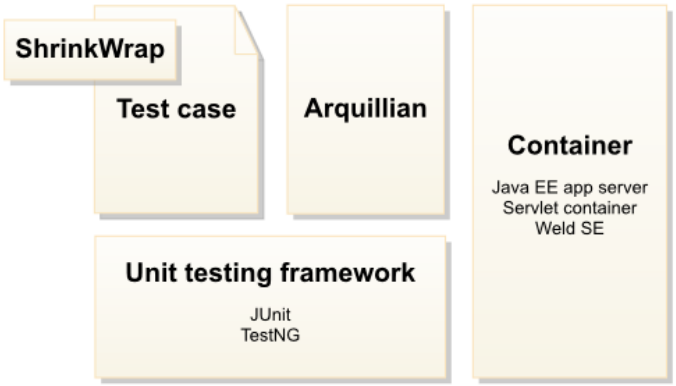


- EJB ve Spring gibi container ihtiyacı duyan katmanlar için test ihtiyacı olabilir.
- Bu durumda ilgili container üzerinde test yapılmalıdır.
- Container' ları oluşturan ve sarmallayan entegrasyon test kütüphaneleri kullanılmalıdır.
- Arquillian ve Spring Integration Test kütüphaneleri kullanılır.

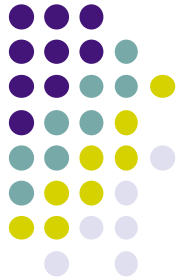
Arquillian



- JBoss ekibi tarafından yayınlanan açık kaynak entegrasyon test kütüphanesidir.
- JPA, EJB, CDI gibi katmanların testleri için ilgili container ayağa kaldırılır.
- Test süreçleri birim testlerine göre daha yavaştır.



Son



Sorular ????

melih.sakarya@sahabt.com