BS450 Yazılım Test Mühendisliği

Ön Bilgi

GitHub Deposu

Bu derse ait bütün içeriğie şu GitHub deposundan erişebilirsiniz:

https://github.com/omursahin/bs450-2024

En son içeriği elde etmek için her hafta bir "pull" yapmanız gerekecektir.

Java

Tüm kod örnekleri Java dilindedir ve testler JUnit'tir.
Repository'deki Java örneklerini kullanmak için, bilgisayarınızda **Java 11 veya daha yeni bir sürümünün** kurulu olması gerekmektedir.

Gradle

Java kod örnekleri **Gradle** kütüphanesinde bulunmaktadır.

Depoyu klonladıktan sonra, terminalden `code` dizininde derleyebilir ve testleri çalıştırabilirsiniz.

Bu komutlar Mac/Linux/WSL'de çalışmaktadır.

(Windows için "`./`" başlangıcını "`.\`" ile değiştirin)

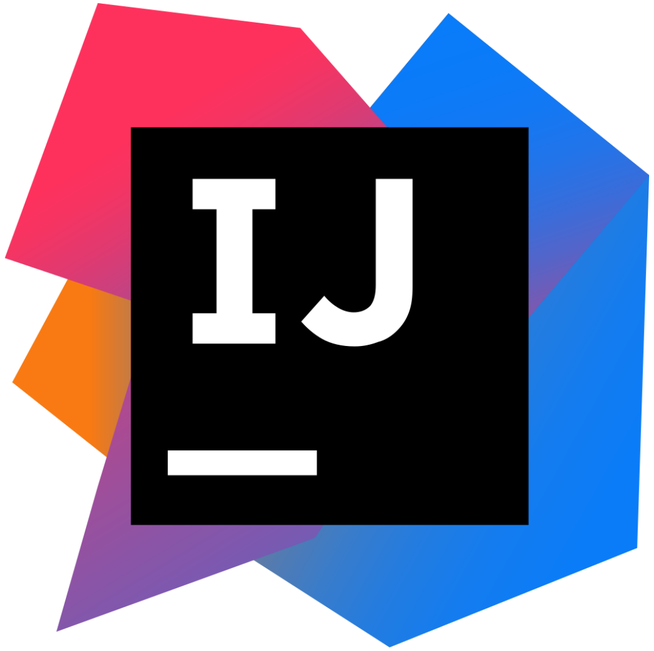
```
./gradlew build   ———▸ tüm kodu derler
./gradlew test    ———▸ tüm testleri çalıştırır
./gradlew test --tests bs450.TriangleTest
./gradlew test --tests bs450.TriangleTest.shouldClassifyEquilateral
```

Spesifik sınıftaki tüm testleri çalıştırır

Spesifik testi çalıştırır

Daha fazla bilgi için:
https://gradle.org

Integrated Developer Environments (IDEs) kullanımı

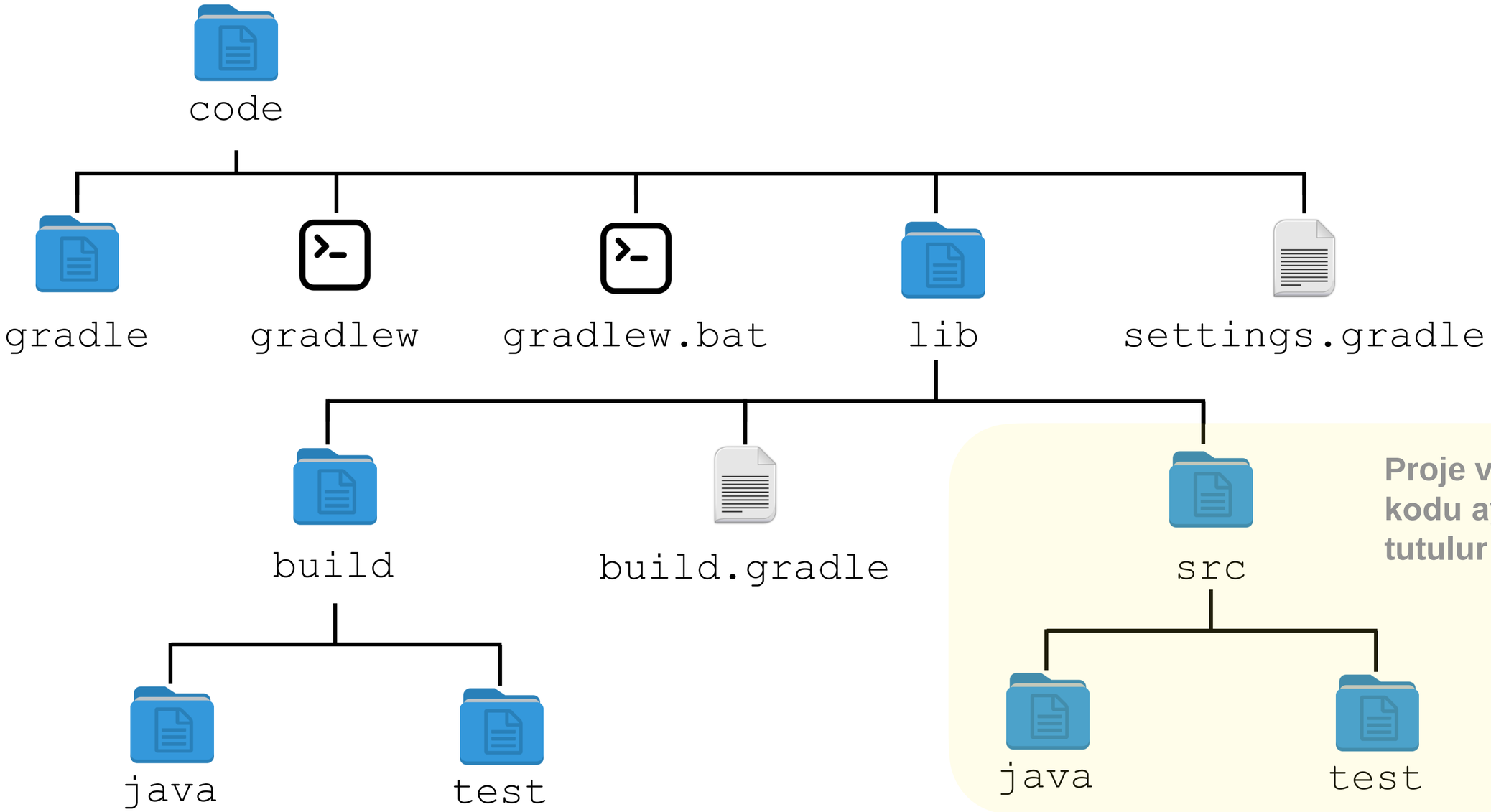


Pek çok modern IDE Graddle destekler, ör. **IntelliJ IDEA** (önerim)

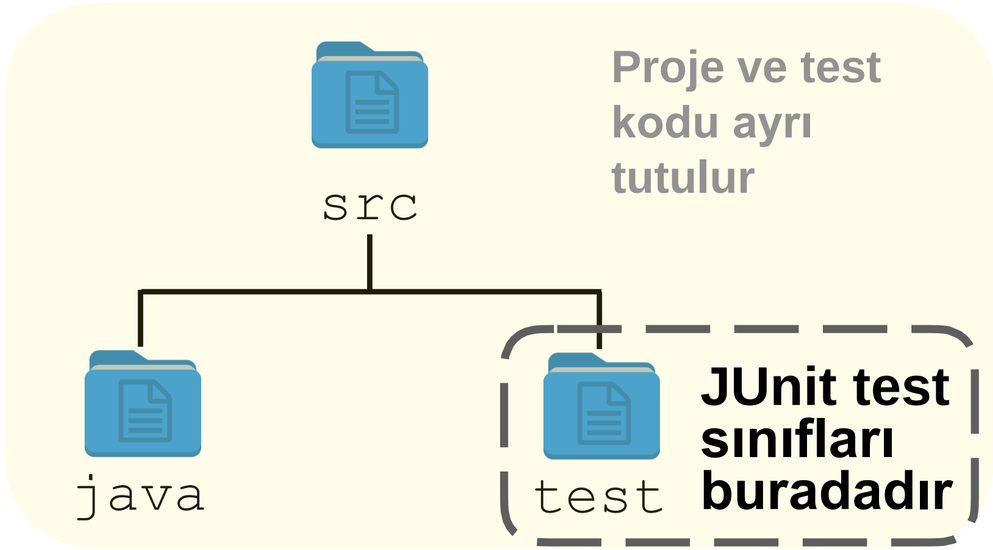
Code klasöründe yeni proje oluşturulur yalnızca Gradle configuration'ını bulmalıdır.

Buradan itibaren, kod otomatik olarak derlenecek ve IDE üzerinden belirli testleri çalıştırabileceksiniz.

Gradle Project Structure



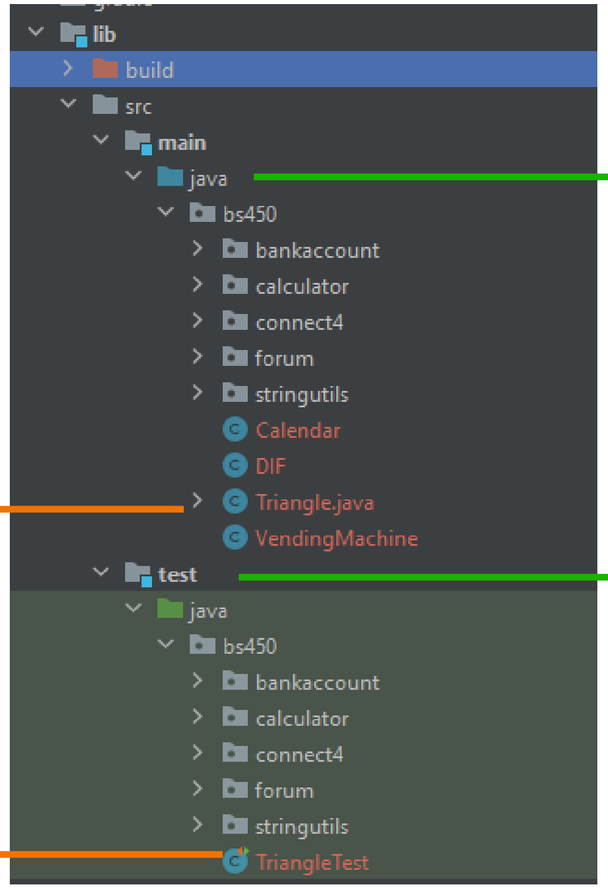
JUnit



Bu modül boyunca JUnit 5'i kullanacağız

Dahası için: https://junit.org/junit5/docs/current/user-guide/

Test Edelim!



Kodlar: `src/main/java`

Testler: `src/test/java`

`Triangle.java` sınıfını `TriangleTest` isimli JUnit test sınıfıyla test edeceğiz.

Bir JUnit Test Sınıfı ve bir Test

```java
import org.junit.jupiter.api.Test;

public class TriangleTest {

    @Test
    public void shouldClassifyEquilateral() {
      // Test code goes here...
    }

    // ...
```

Testler `@Test` anotasyonu ile işaretlenir böylelikle JUnit hangi sınıfın test hangisinin ise yardımcı sınıf olduğunu bilir.

Bir testin içeriği

```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class TriangleTest {

    @Test
    public void shouldClassifyEquilateral() {
        Triangle.Type result = Triangle.classify(10, 10, 10);
        assertEquals(Triangle.Type.EQUILATERAL, result);
    }

    // ...
```

Test oluşturmak ve sistemin istediğimiz parçasını test etmek için metot çağrıları yapılarak başlanır.

Bir testin içeriği

```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class TriangleTest {

    @Test
    public void shouldClassifyEquilateral() {
        Triangle.Type result = Triangle.classify(10, 10, 10);
        assertEquals(Triangle.Type.EQUILATERAL, result);
    }

    // ...
```

Ardından **actual (mevcut) sonuç** **beklenen (expected)** sonuca eşit mi kontrol etmek için assertion yazılır.

JUnit Assertion'ları

```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class TriangleTest {

    @Test
    public void shouldClassifyEquilateral() {
        Triangle.Type result = Triangle.classify(10, 10, 10);
        assertEquals(Triangle.Type.EQUILATERAL, result);
    }

    // ...
```

`assertEquals` metodu, JUnit'in bir parçasıdır ve özellikle **beklenen bir değerin**, test edilen birimden döndürülen **gerçek değere eşit** olup olmadığını kontrol eder.

JUnit, gerçek ve beklenen çıktılar arasındaki ilişkileri kontrol etmek için çok sayıda onaylama türüne sahiptir.

Bunlar `assertTrue(booleanVariable)`, `assertNull(reference)`, gibi pek çok assertion'dan oluşmaktadır. Ayrıntı için:

https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html

assertThrows ile İstisnaları Kontrol Etme

```java
@Test
public void shouldThrowExceptionWithInvalidTriangle() {
    assertThrows(InvalidTriangleException.class, () -> {
        Triangle.classify(0, 0, 0);
    });
}
```

```java
@Test
public void shouldThrowExceptionWithInvalidTriangle() {
    Exception e = assertThrows(InvalidTriangleException.class, () -> {
        Triangle.classify(0, 0, 0);
    });
    assertEquals("(0, 0, 0) is not a valid triangle", e.getMessage());
}
```

İstisna mesajını da denetleyen daha detaylı bir versiyonu.

Ancak bu türde bir kontrol testleri daha **kırılgan** yapacaktır.

"`import static`", başka bir sınıftan bir statik yöntemi içe aktarmayı ve bunu sanki mevcut sınıfta gibi kullanmayı ifade eder.

Bir test `@Test` ile annotate edilir.

Metottan dönen değerin beklendiği gibi olduğunu iddia eden: `assertEquals`

Beklendiği gibi bir istisna fırlatıldığını iddia eden `assertThrows`

```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

public class TriangleTest {

    @Test
    public void shouldClassifyEquilateral() {
        Triangle.Type result = Triangle.classify(10, 10, 10);
        assertEquals(Triangle.Type.EQUILATERAL, result);
    }

    @Test
    public void shouldClassifyIsoceles() {
        Triangle.Type result = Triangle.classify(5, 10, 10);
        assertEquals(Triangle.Type.ISOSCELES, result);
    }

    @Test
    public void shouldClassifyIsocelesWhenSidesAreOutOfOrder() {
        Triangle.Type result = Triangle.classify(10, 10, 5);
        assertEquals(Triangle.Type.ISOSCELES, result);
    }

    @Test
    public void shouldThrowExceptionWithInvalidTriangle() {
        assertThrows(InvalidTriangleException.class, () -> {
            Triangle.classify(0, 0, 0);
        });
    }
}
```