

2023 땅울림 썸머코딩

JUnit5를 이용한 단위테스트

진행자: 김선우

목차

01 단위테스트와 JUnit5

02 JUnit5 기본문법

03 JUnit5 심화문법

04 테스트 주도 개발(TDD)

05 Mockito

06 테스트 커버리지와 Jacoco

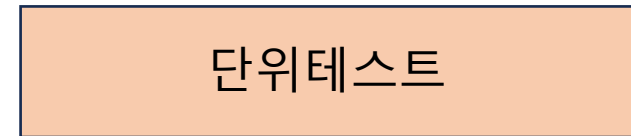
01

단위테스트와 JUnit5

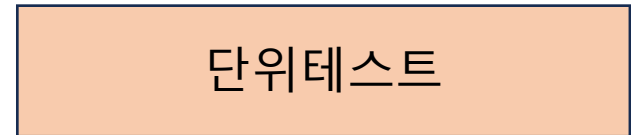
단위테스트와 JUnit5

```
public boolean isValid(String input){  
  
    // 입력을 검증하는 메소드  
  
    // 길이, 특수문자 여부 등...  
  
}
```

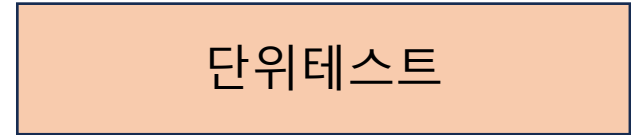
input → result



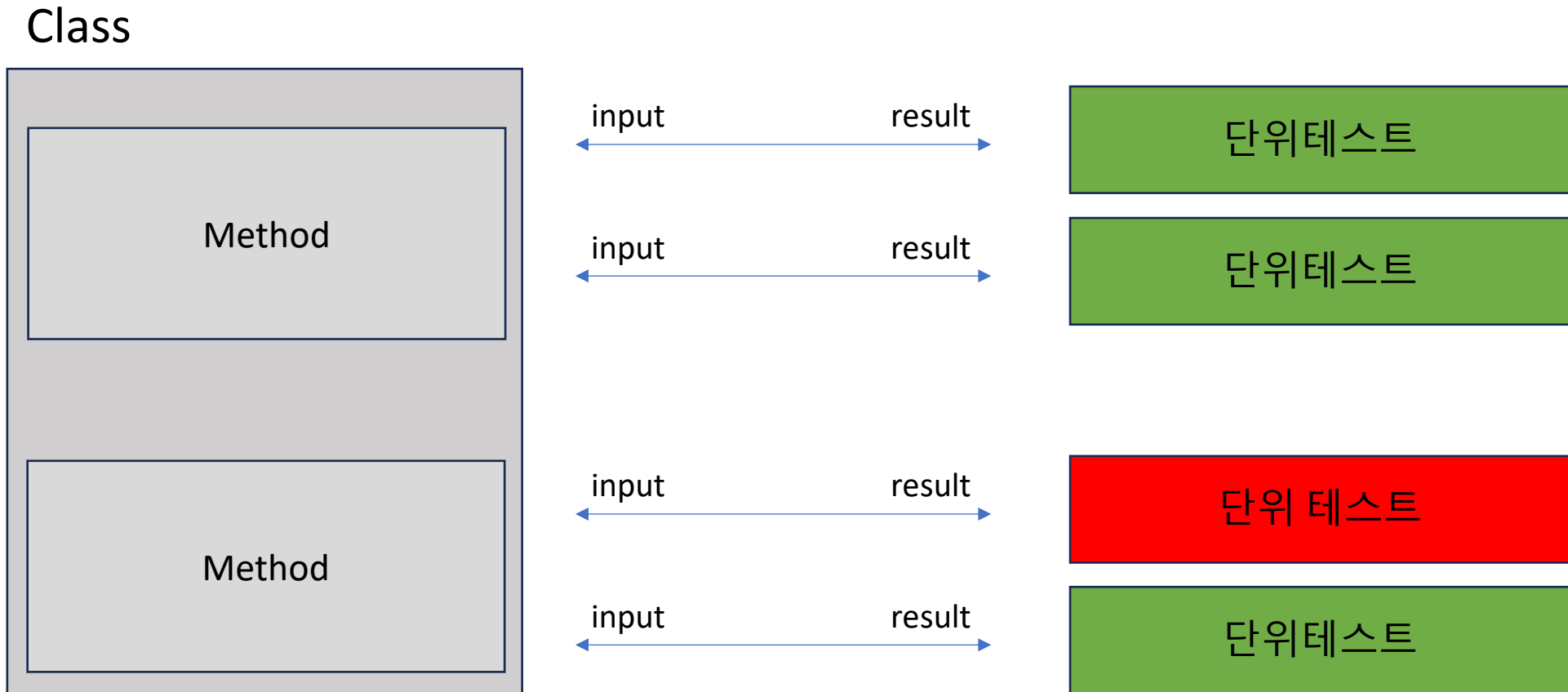
input → result



input → result




단위테스트와 JUnit5



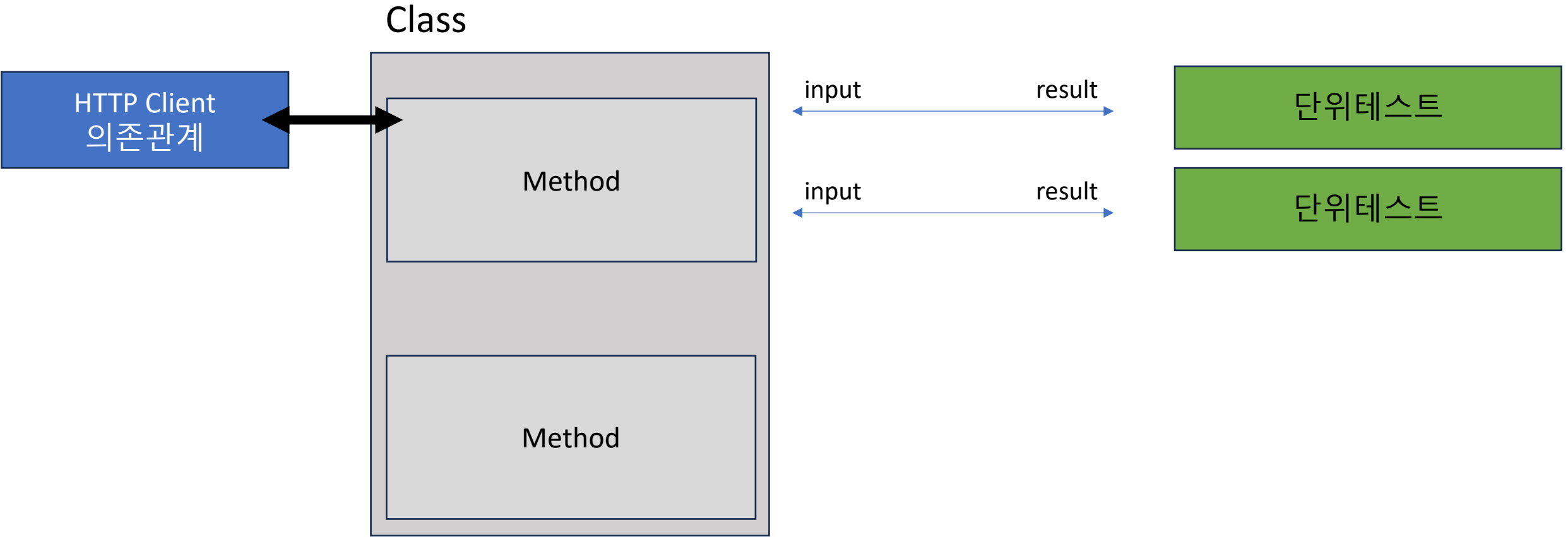
단위테스트와 JUnit5

```
public class Calculator {  
  
    public int integerDivision(int dividend, int divisor) {  
        return dividend / divisor;  
    }  
  
}
```

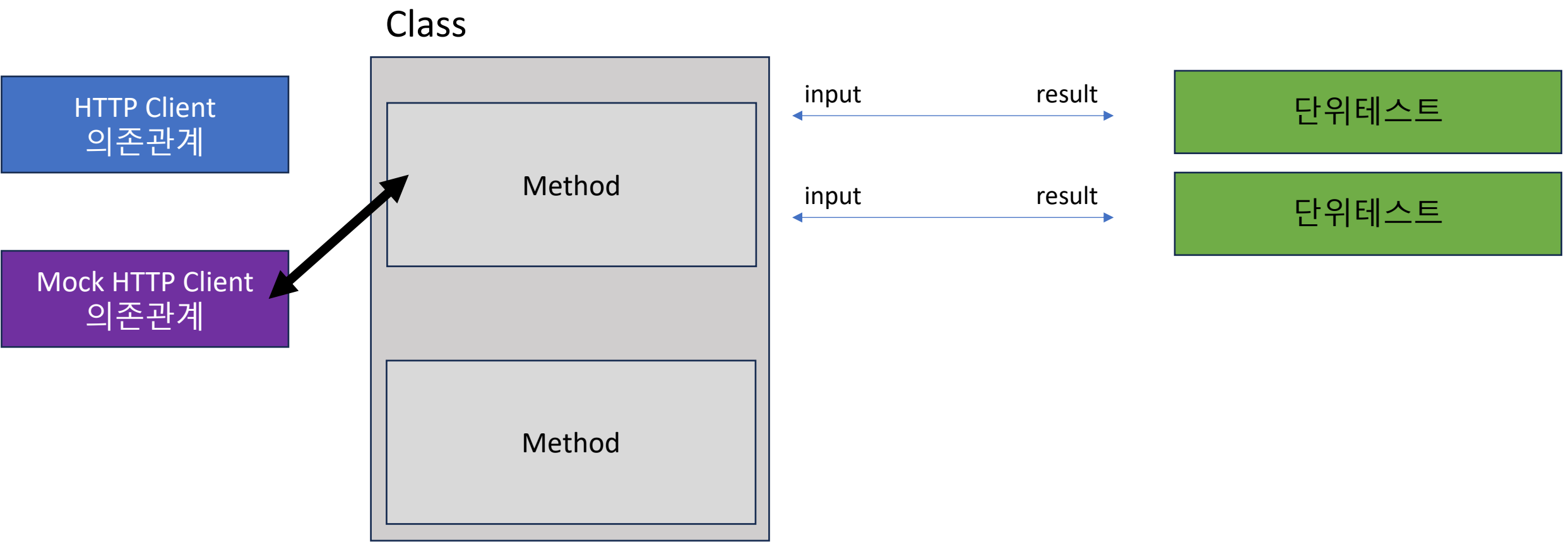


```
@Test  
void testIntegerDivision(){  
    // Arrange  
    Calculator calculator = new Calculator();  
  
    // Act  
    int result = calculator.integerDivision(dividend: 4, divisor: 2);  
  
    //Assert  
    Assertions.assertEquals(expected: 2, result, message: "4/2 should be 2!");  
}
```

단위테스트와 JUnit5



단위테스트와 JUnit5



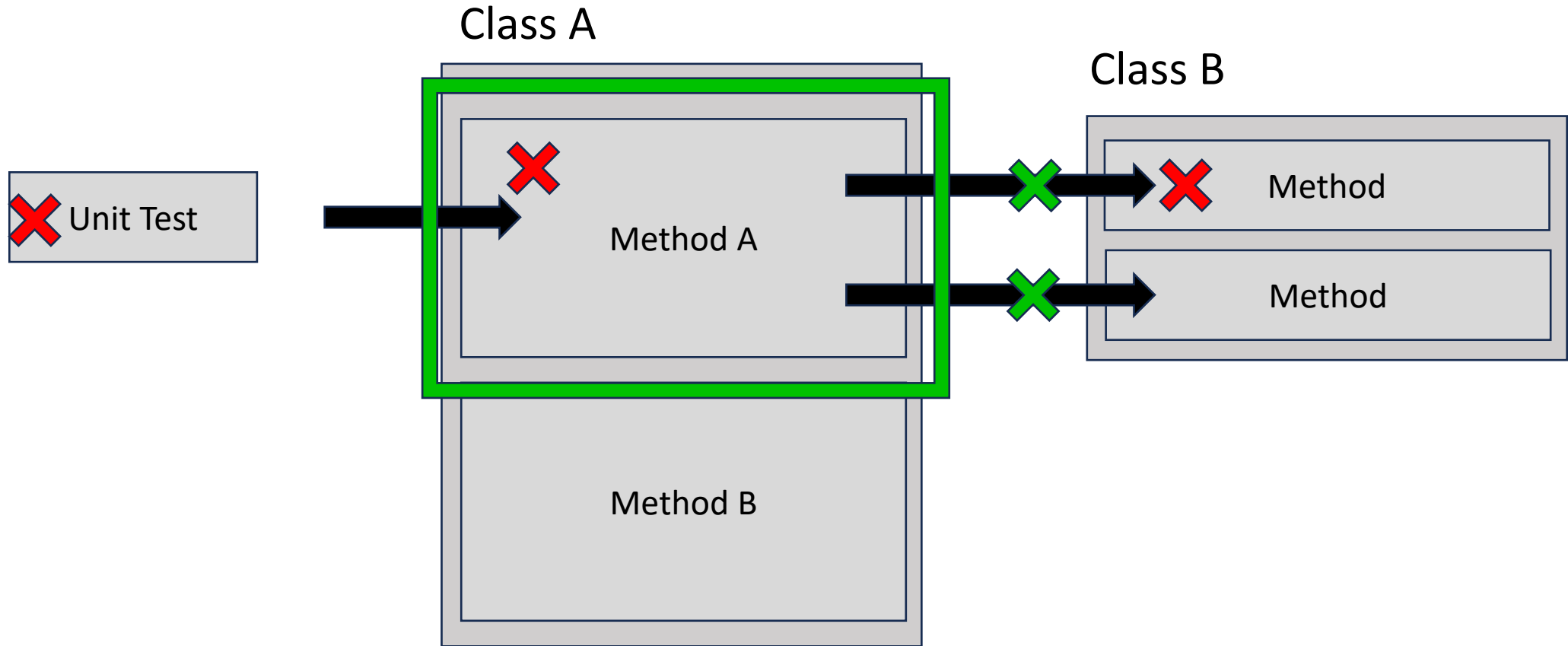
단위테스트를 작성해야하는 이유

- 코드가 제대로 동작하는지 확인하기
- 리팩토링 해도 제대로 동작하는지 확인
- 프로그램의 신뢰성 확보
- 개발 초기 단계에서 버그 발견

FIRST

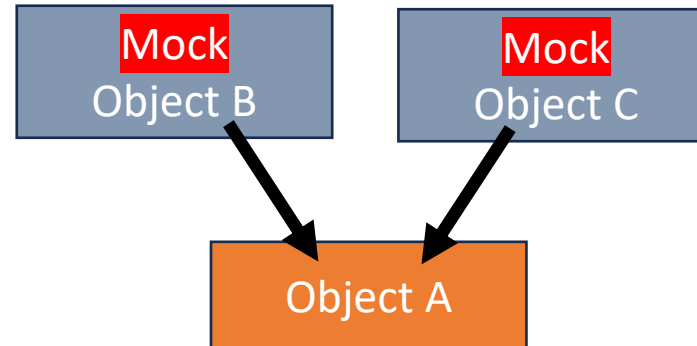
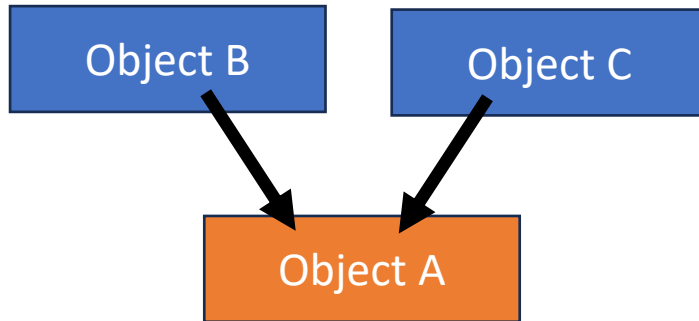
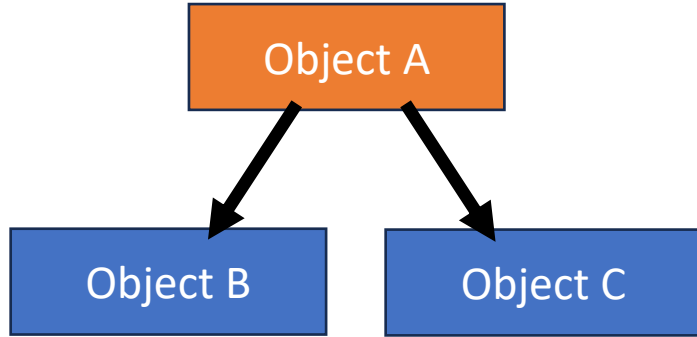
- **Fast** – 빠르게 동작
- **Independent** – 다른 테스트 의존 X 독립적이어야함
- **Repeatable** – 여러번 동작해도 같은 결과
- **Self-validating** – 단위테스트는 자신을 검증해야한다
- **Thorough & Timely** – 적재적소에 작성되어야한다

단위테스트와 JUnit5



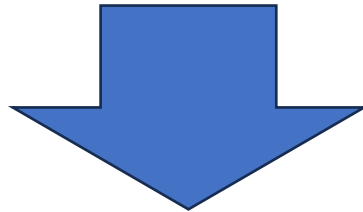
단위테스트와 JUnit5

Dependency Injection



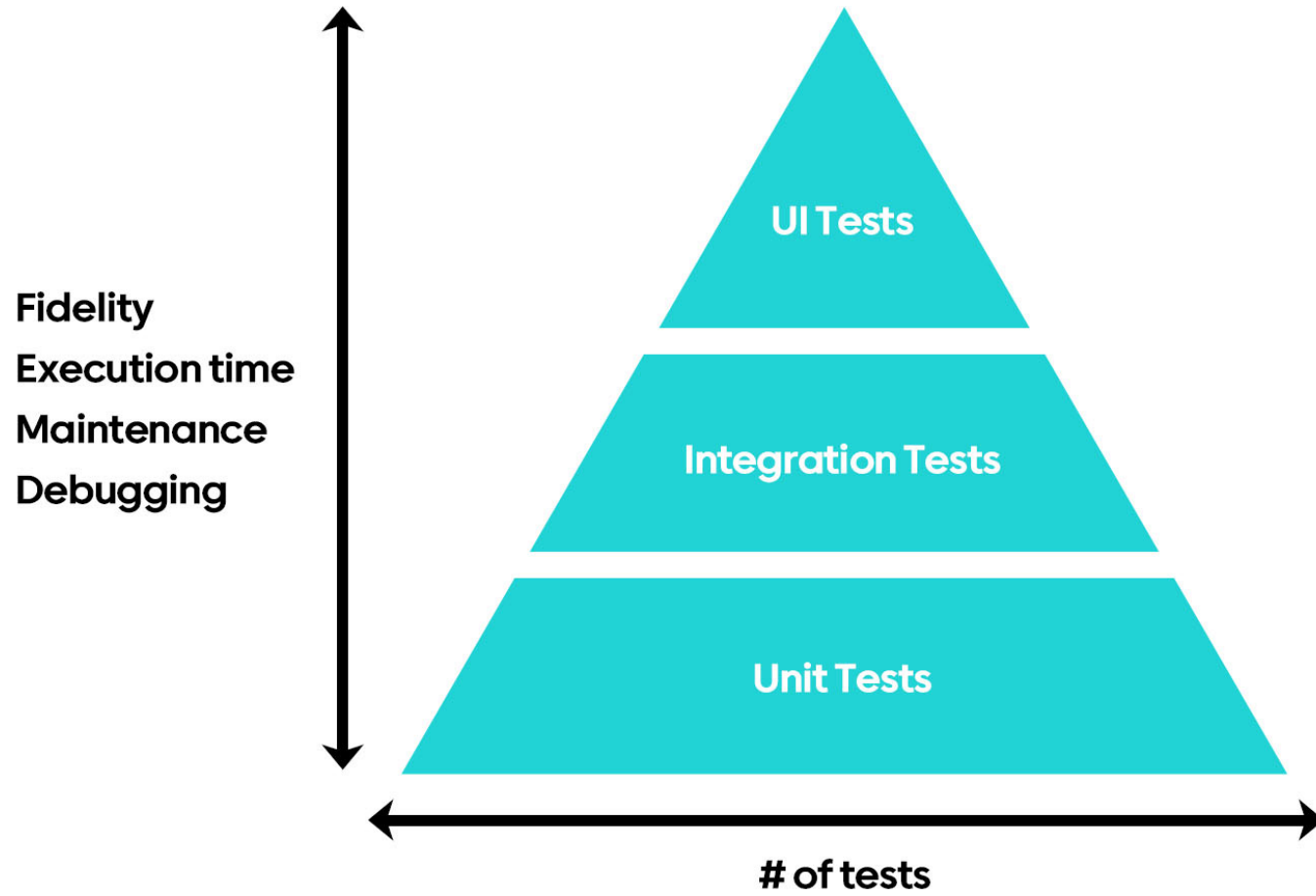
단위테스트와 JUnit5

```
public SignupResult processUserSignUp(FormData formData){  
  
    SignupWebService webservice = new SignupWebserviceImpl();  
  
    return webservice.performSignup(formData)  
}
```

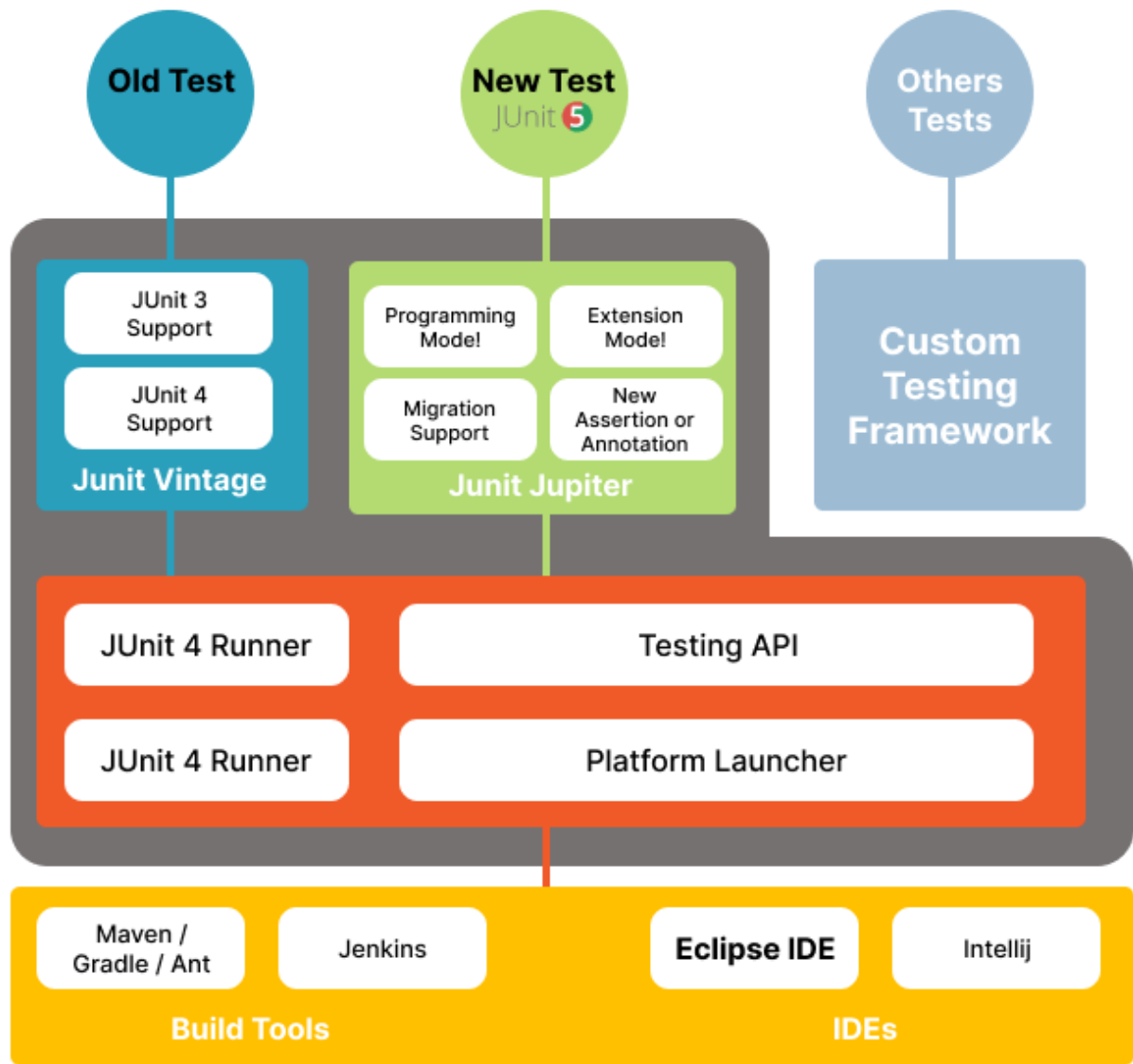


```
public SignupResult processUserSignUp(FormData formData, SignupWebService signupWebService){  
  
    return webservice.performSignup(formData)  
}
```

단위테스트와 JUnit5



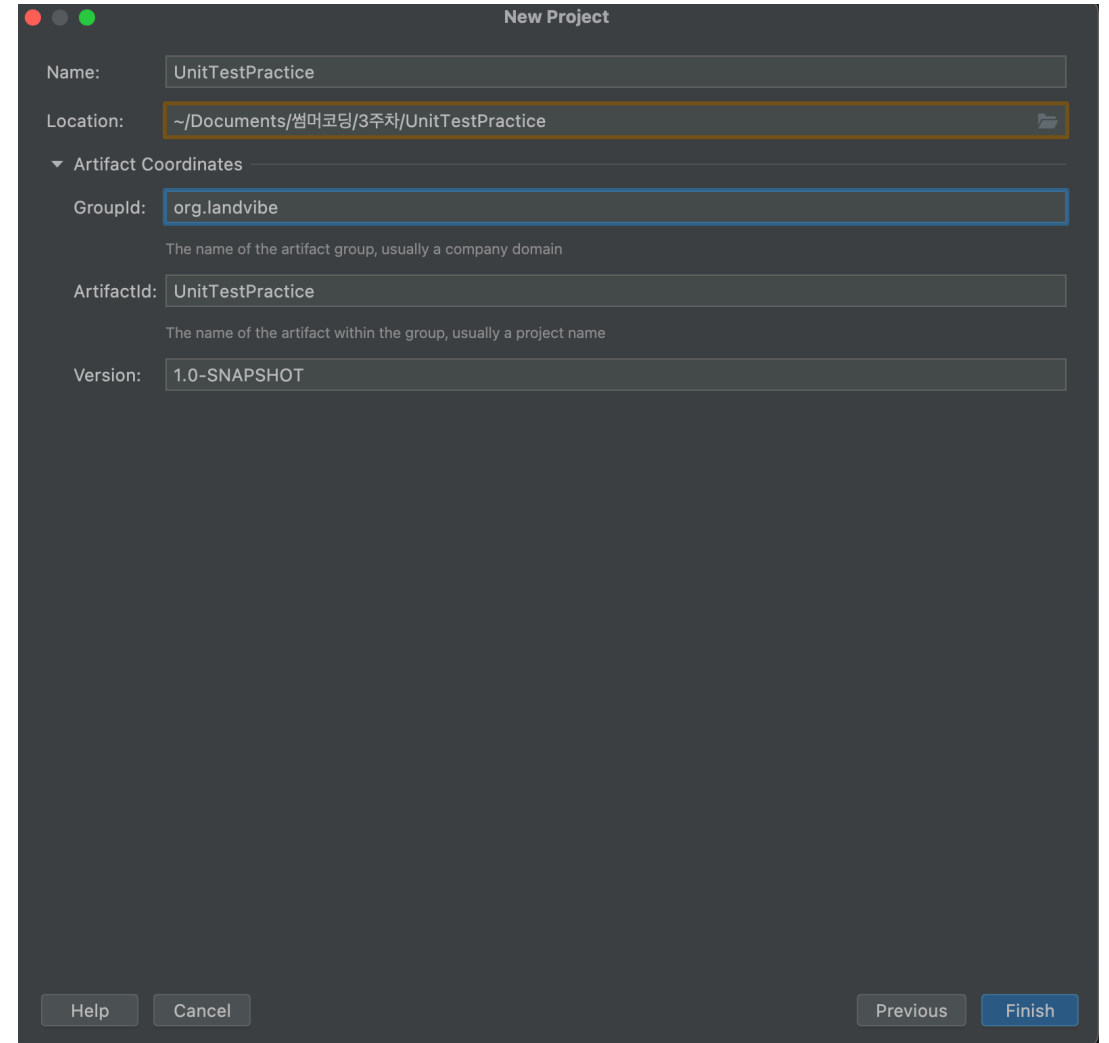
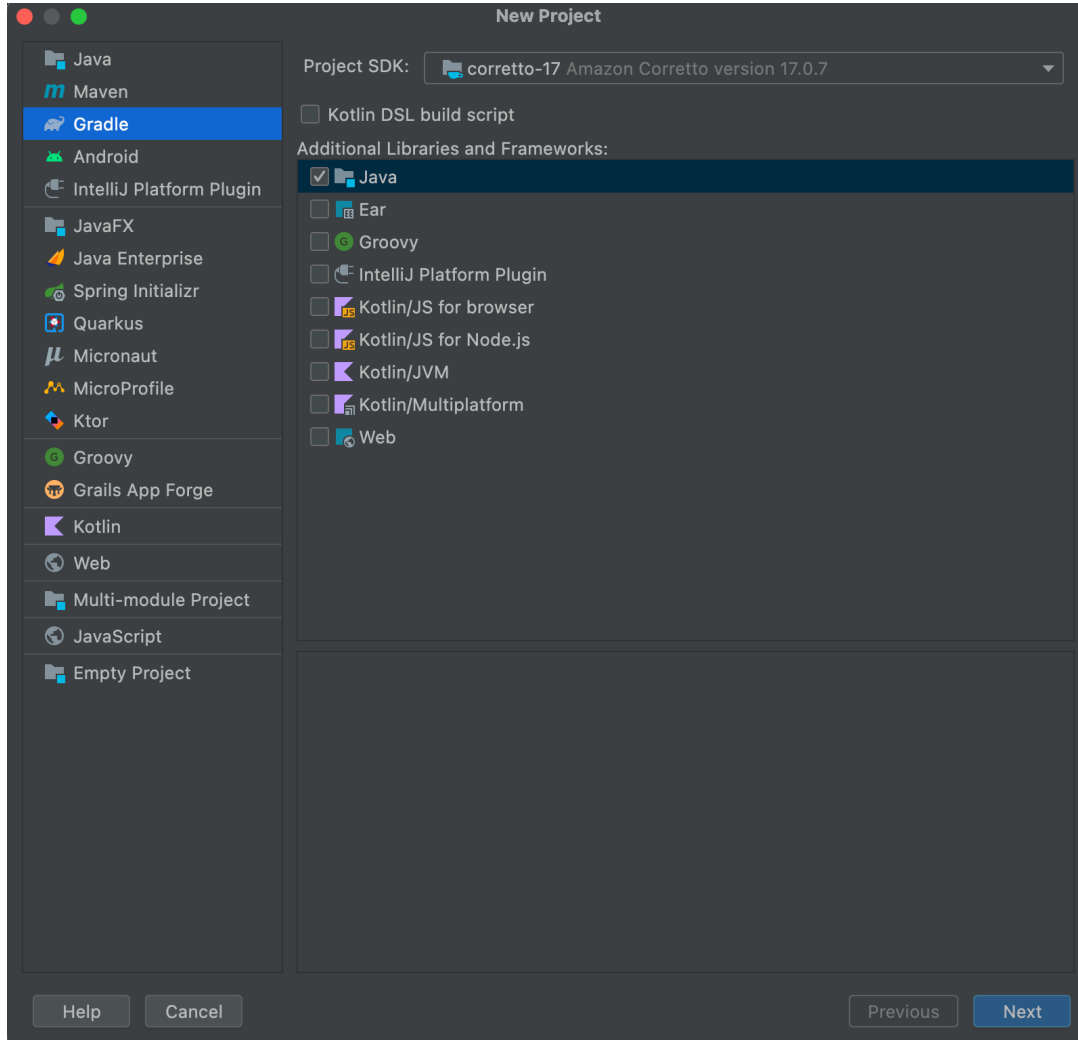
단위테스트와 JUnit5



02

JUnit 기본 문법

JUnit5와 기본문법



JUnit5와 기본문법

```
build.gradle (UnitTestPractice) x
1  plugins {
2      id 'java'
3  }
4
5  group 'org.landvibe'
6  version '1.0-SNAPSHOT'
7
8  repositories {
9      mavenCentral()
10 }
11
12 dependencies {
13     testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
14     testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
15 }
16
17 test {
18     useJUnitPlatform()
19 }
```

JUnit5와 기본문법

The screenshot shows the Maven Repository search results for the query 'junit:jupiter'. The interface includes a search bar at the top with the text 'junit:jupiter' and a 'Search' button. Below the search bar, the results are sorted by 'relevance'. The first three results are highlighted with a red border:

- 1. JUnit Jupiter API** (13,245 usages, EPL license)
org.junit.jupiter » junit-jupiter-api
JUnit Jupiter is the API for writing tests using JUnit 5.
Last Release on Jul 6, 2023
- 2. JUnit Jupiter Engine** (13,155 usages, EPL license)
org.junit.jupiter » junit-jupiter-engine
Core package for the JUnit Jupiter test engine.
- 3. JUnit Jupiter (Aggregator)** (6,856 usages, EPL license)
org.junit.jupiter » junit-jupiter
Module "junit-jupiter" of JUnit 5.
Last Release on Jul 6, 2023

The left sidebar shows the repository structure with sections for 'Repository', 'Group', and 'Category'. The 'Repository' section lists various repositories like Central, Sonatype, and Spring Plugins. The 'Group' section lists groups like com.github, io.github, and org.apache. The 'Category' section lists categories like Maven Archetype, Android Package, and Maven Plugins.

JUnit5와 기본문법

MVN REPOSITORY

Search for groups, artifacts, categories

Home » org.junit.jupiter » junit-jupiter » 5.9.3

JUnit Jupiter (Aggregator) » 5.9.3

Module "junit-jupiter" of JUnit 5.

License	EPL 2.0
Categories	Testing Frameworks & Tools
Tags	testing junit
HomePage	https://junit.org/junit5/
Date	Apr 26, 2023
Files	pom (3 KB) jar (6 KB) View All
Repositories	Central
Ranking	#71 in MvnRepository (See Top Artifacts) #9 in Testing Frameworks & Tools
Used By	6,856 artifacts

Note: There is a new version for this artifact

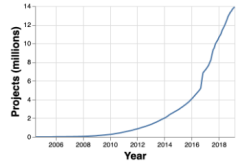
Maven Gradle **Gradle (Short)** Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
// https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter
testImplementation 'org.junit.jupiter:junit-jupiter:5.9.3'
```

Include comment with link to declaration

Copied to clipboard!

Indexed Artifacts (34.5M)

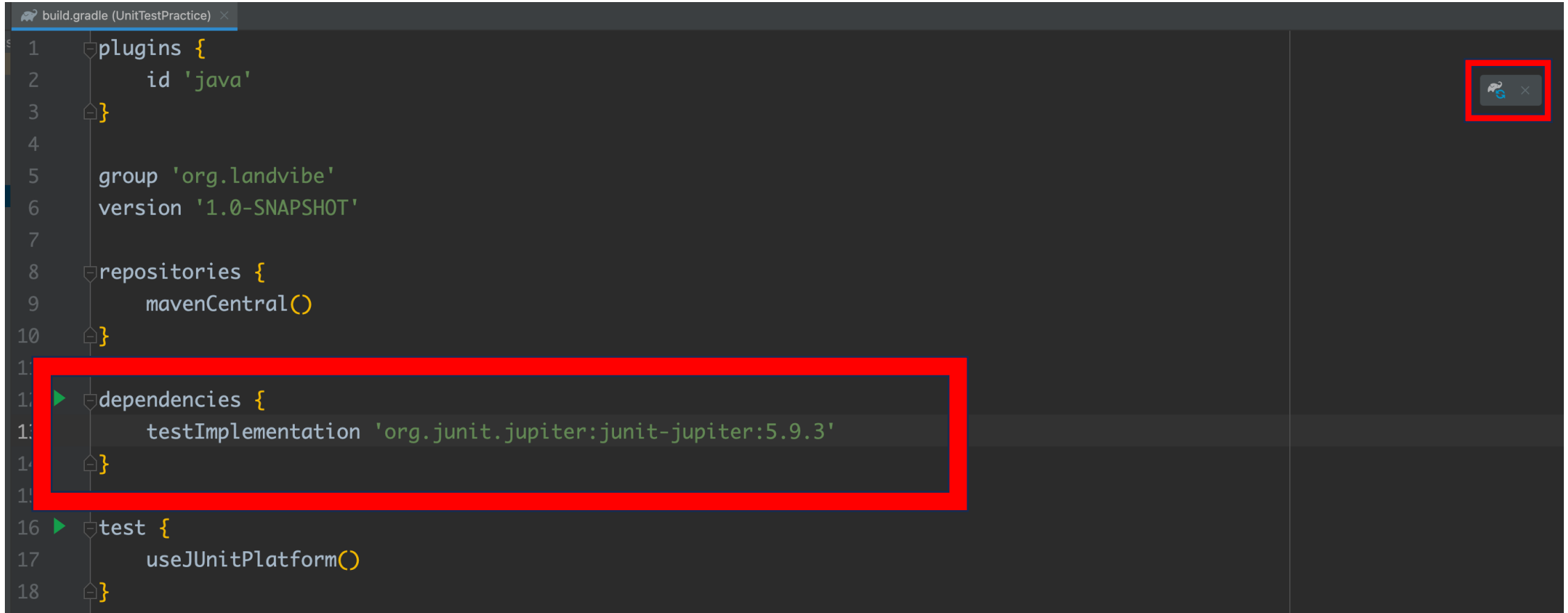


Popular Categories

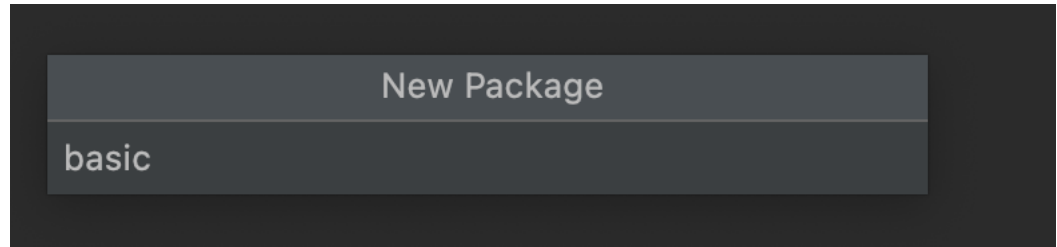
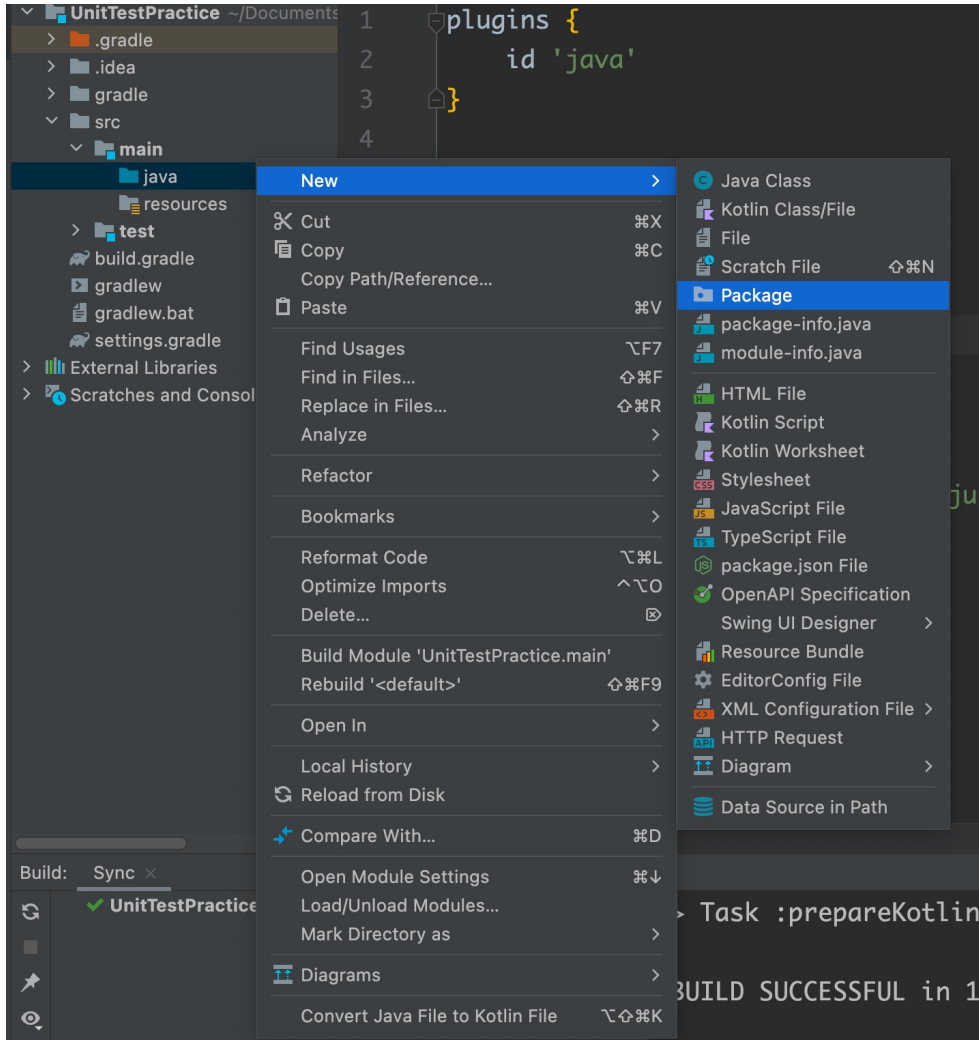
- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- JVM Languages
- Core Utilities
- Language Runtime
- Mocking
- Web Assets
- Annotation Libraries
- Logging Bridges
- HTTP Clients
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Defect Detection Metadata
- Configuration Libraries

JUnit5와 기본문법

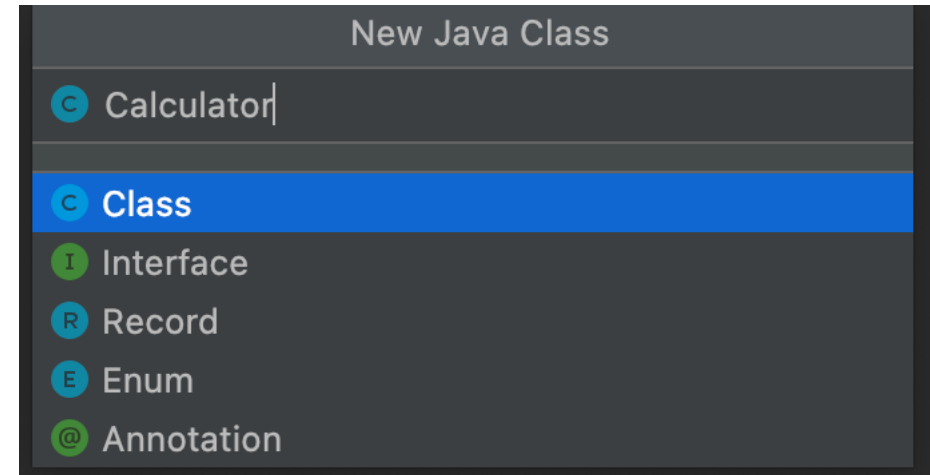
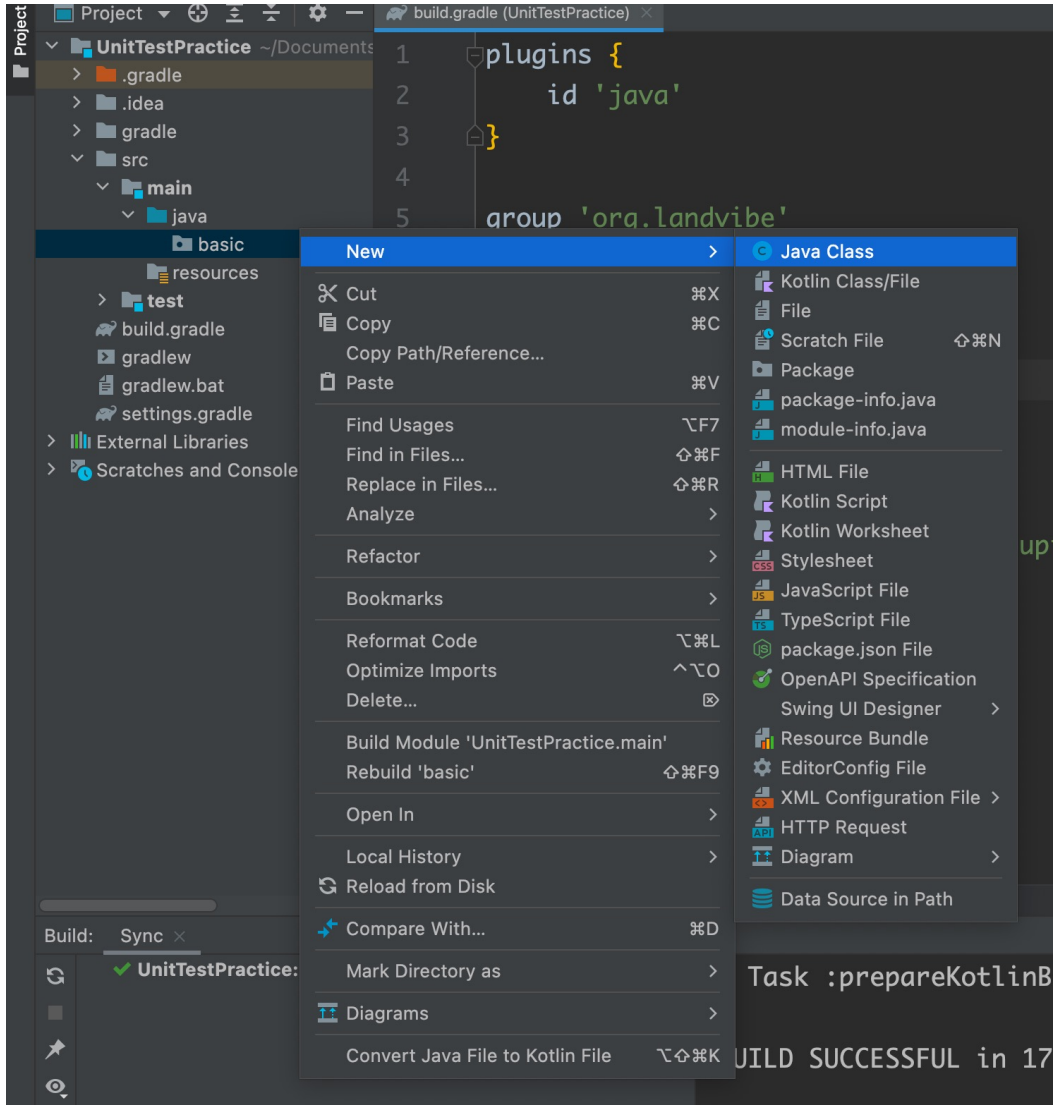
```
build.gradle (UnitTestPractice) x
1  plugins {
2      id 'java'
3  }
4
5  group 'org.landvibe'
6  version '1.0-SNAPSHOT'
7
8  repositories {
9      mavenCentral()
10 }
11
12 dependencies {
13     testImplementation 'org.junit.jupiter:junit-jupiter:5.9.3'
14 }
15
16 test {
17     useJUnitPlatform()
18 }
```



JUnit5와 기본문법



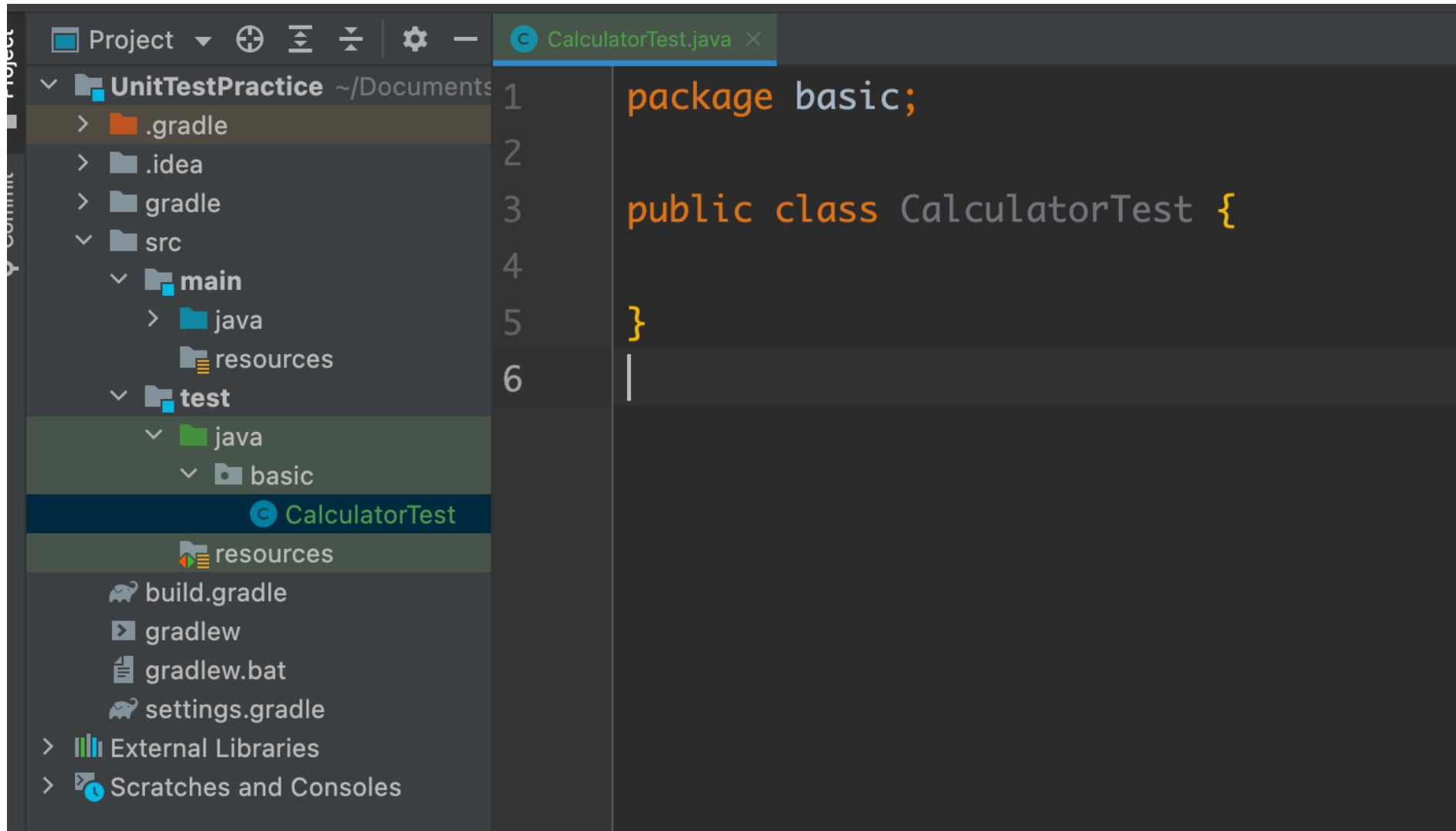
JUnit5와 기본문법



JUnit5와 기본문법

```
Calculator.java x
1 package basic;
2
3 public class Calculator {
4     public int integerDivision(int dividend, int divisor) {
5         return dividend / divisor;
6     }
7
8     public int integerSubtraction(int minuend, int subtrahend){
9         return minuend - subtrahend;
10    }
11 }
```


JUnit5와 기본문법



JUnit5와 기본문법

```
5  
6 public class CalculatorTest {  
7  
8     @Test  
9     void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo(){  
10         Calculator calculator = new Calculator();  
11         int result = calculator.integerDivision( dividend: 4, divisor: 2);  
12         Assertions.assertEquals( expected: 2, result);  
13     }  
14 }
```

The screenshot shows the IDE's interface during a test run. At the top, a 'Run' window displays the test name: 'CalculatorTest.integerDivision_WhenFourIsDividedByT...'. Below this, a 'Test Results' tree shows a successful test: 'basic.CalculatorTest.integerDivision_WhenFourIsDividedByTwo_ShouldRe' with a duration of 7ms. To the right, the build output window shows the following tasks: 'Task :compileJava', 'Task :processResources NO-SOURCE', 'Task :classes', 'Task :compileTestJava', 'Task :processTestResources NO-SOURCE', 'Task :testClasses', and 'Task :test'. The final output is 'BUILD SUCCESSFUL in 889ms'. The status bar at the top right of the IDE indicates 'Tests passed: 1 of 1 test - 7ms'.

JUnit5와 기본문법

```
@Test
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo(){
    Calculator calculator = new Calculator();
    int result = calculator.integerDivision(dividend: 4, divisor: 2);
    Assertions.assertEquals(expected: 9999, result, message: "4 나누기 2의 결과가 2가 아닙니다!");
}
```

Run: CalculatorTest.integerDivision_WhenFourIsDividedByT... x

Tests failed: 1 of 1 test - 7 ms

Test Results 7 ms

- basic.CalculatorTest 7 ms
 - integerDivision_WhenFourIsDividedByTwo_ShouldR... 7 ms

4 나누기 2의 결과가 2가 아닙니다! ==> expected: <9999> but was: <2>

Expected :9999
Actual :2
[<Click to see difference>](#)

org.opentest4j.AssertionFailedError: 4 나누기 2의 결과가 2가 아닙니다! ==> expected: <9999> but was: <2>
at app//org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:151)
at app//org.junit.jupiter.api.AssertionFailureBuilder.buildAndThrow(AssertionFailureBuilder.java:132)

JUnit5와 기본문법

```
@Test
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {
    // Arrange (given)
    Calculator calculator = new Calculator();
    int dividend = 4;
    int divisor = 2;
    int expectedResult = 2;

    // Act (when)
    int result = calculator.integerDivision(dividend, divisor);

    // Assert (then)
    assertEquals(expectedResult, result, message: "4 나누기 2의 결과가 2가 아닙니다!");
}
```

JUnit5와 기본문법

```
2
3 ▶ @Test
4 void integerDivision_WhenDividendIsDividedByZero_ShouldThrowArithmeticException() {
5     // Arrange (given)
6     Calculator calculator = new Calculator();
7     int dividend = 4;
8     int divisor = 0;
9
10    // Act, Assert
11    Assertions.assertThrows(ArithmeticException.class, () -> {
12        | calculator.integerDivision(dividend, divisor);
13    }, message: "0으로 나눌 경우에는 ArithmeticException을 발생시킨다");
14 }
15
```

JUnit5와 기본문법

```
@Test
void integerDivision_WhenDividendIsDividedByZero_ShouldThrowArithmeticException() {
    // Arrange
    Calculator calculator = new Calculator();
    int dividend = 4;
    int divisor = 0;
    String expectedExceptionMessage = "/ by zero";

    // Act
    ArithmeticException actualException = assertThrows(ArithmeticException.class, () -> {
        calculator.integerDivision(dividend, divisor);
    }, message: "0으로 나눌 경우에는 ArithmeticException을 발생시킨다");

    // Assert
    assertEquals(expectedExceptionMessage, actualException.getMessage(),
        message: "Unexpected exception message");
}
```

JUnit5와 기본문법

```
public class CalculatorTest {  
  
    @Test  
    void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {  
        // Arrange (given)  
        Calculator calculator = new Calculator();  
        int dividend = 4;  
    }  
}
```

The screenshot shows an IDE interface with two main panels. The top panel displays the test results, and the bottom panel shows the build output.

Test Results Panel: A tree view showing the following structure:

- Test Results (8 ms)
 - basic.CalculatorTest (8 ms)
 - integerDivision_WhenDividendIsDividedByZero_ShouldReturnTwo (7 ms)
 - integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo (1 ms)

The test results are highlighted with a red box. The status bar at the top of this panel indicates "Tests passed: 2 of 2 tests - 8 ms".

Build Output Panel: Shows the following tasks:

- > Task :compileJava UP-TO-DATE
- > Task :processResources NO-SOURCE
- > Task :classes UP-TO-DATE
- > Task :compileTestJava UP-TO-DATE
- > Task :processTestResources NO-SOURCE
- > Task :testClasses UP-TO-DATE
- > Task :test

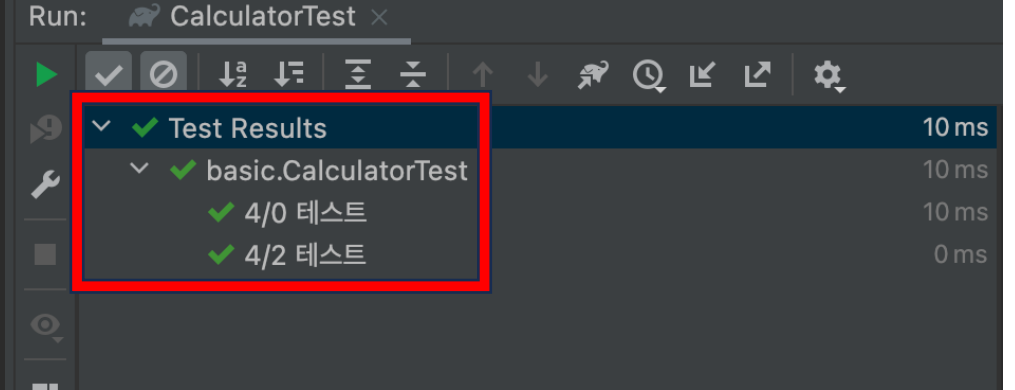
JUnit5와 기본문법

```
@DisplayName("4/2 테스트")
@Test
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {
    // Arrange (given)
    Calculator calculator = new Calculator();
    int dividend = 4;
    int divisor = 2;
    int expectedResult = 2;

    // Act (when)
    int result = calculator.integerDivision(dividend, divisor);

    // Assert (then)
    assertEquals(expectedResult, result, message: "4 나누기 2의 결과가 2가 아닙니다!");
}

@DisplayName("4/0 테스트")
@Test
void integerDivision_WhenDividendIsDividedByZero_ShouldThrowArithmeticException() {
    // Arrange
```



Test Name	Duration	Status
Test Results	10 ms	Passed
basic.CalculatorTest	10 ms	Passed
4/0 테스트	10 ms	Passed
4/2 테스트	0 ms	Passed

JUnit5와 기본문법

```
@DisplayName("33-1 테스트")
@Test
void integerSubtraction_When33Minus1_ShouldReturn32() {
    // Arrange
    Calculator calculator = new Calculator();
    int minuend = 33;
    int subtrahend = 1;
    int expectedResult = 32;

    // Act
    int result = calculator.integerSubtraction(minuend: 33, subtrahend: 1);

    // Assert
    assertEquals(expected: 32, result,
        message: minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}
```

JUnit5와 기본문법

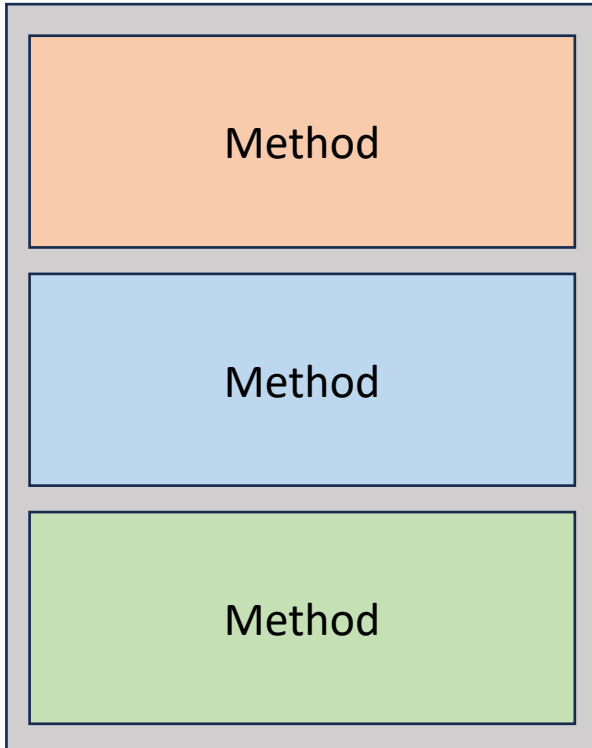
```
@DisplayName("33-1 테스트")
@Test
void integerSubtraction_When33Minus1_ShouldReturn32() {
    // Arrange
    Calculator calculator = new Calculator();
    int minuend = 33;
    int subtrahend = 1;
    int expectedResult = 32;

    // Act
    int result = calculator.integerSubtraction(minuend: 33, subtrahend: 1);

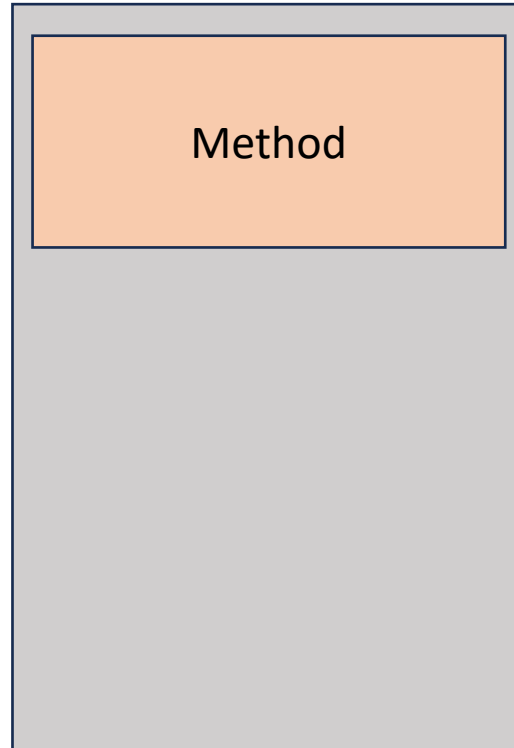
    // Assert
    assertEquals(expected: 32, result,
        () -> minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}
```

JUnit5와 기본문법

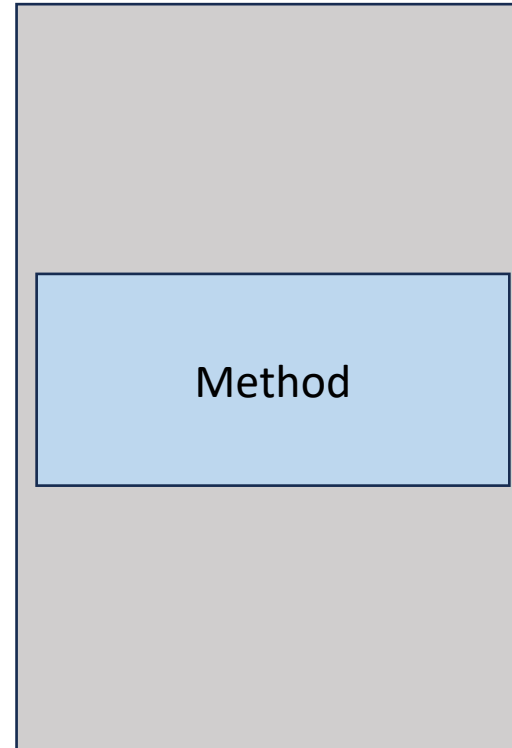
Test Class



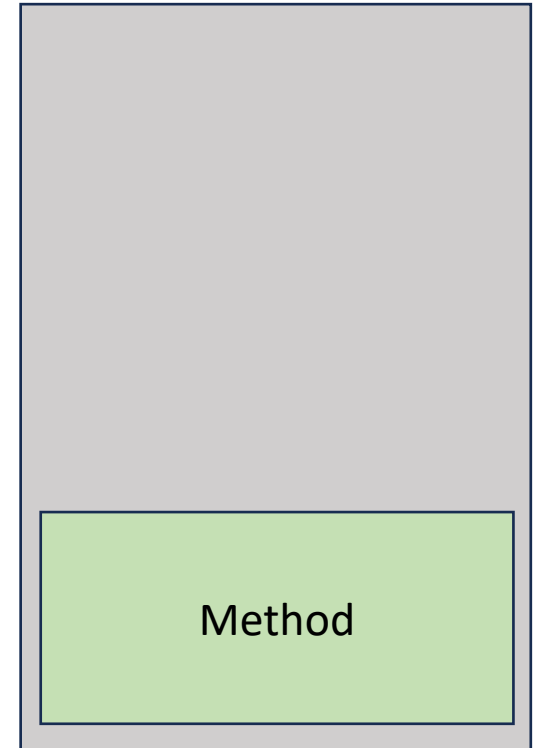
Class Instance 1



Class Instance 2



Class Instance 3

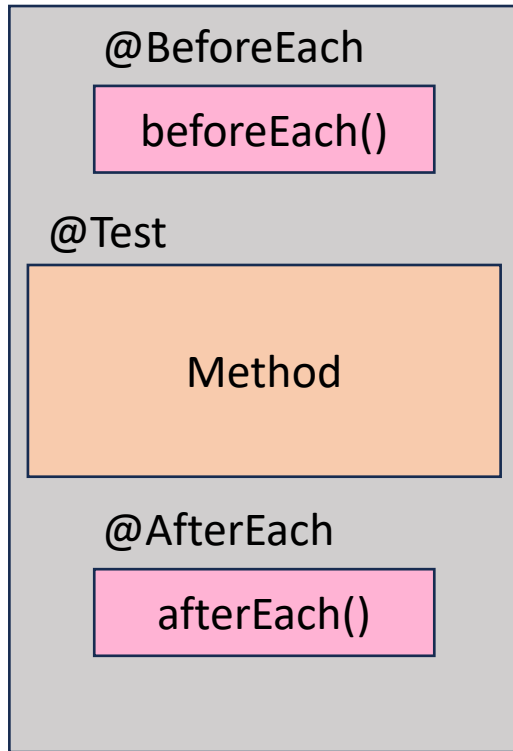


JUnit5와 기본문법

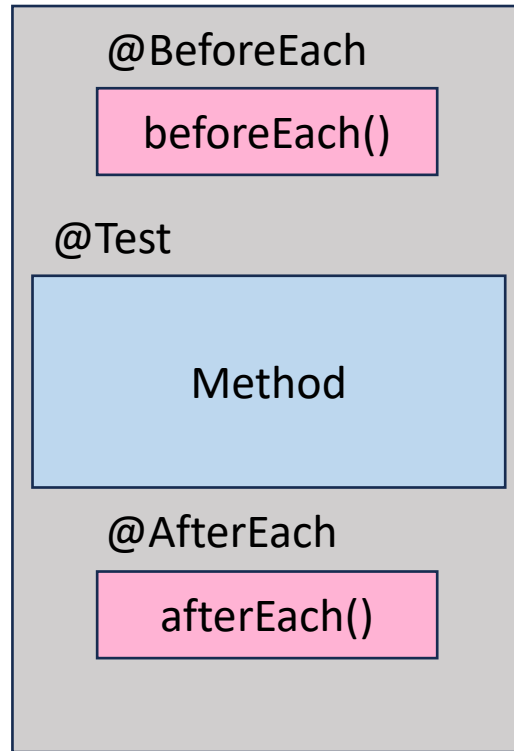
@BeforeAll

setup()

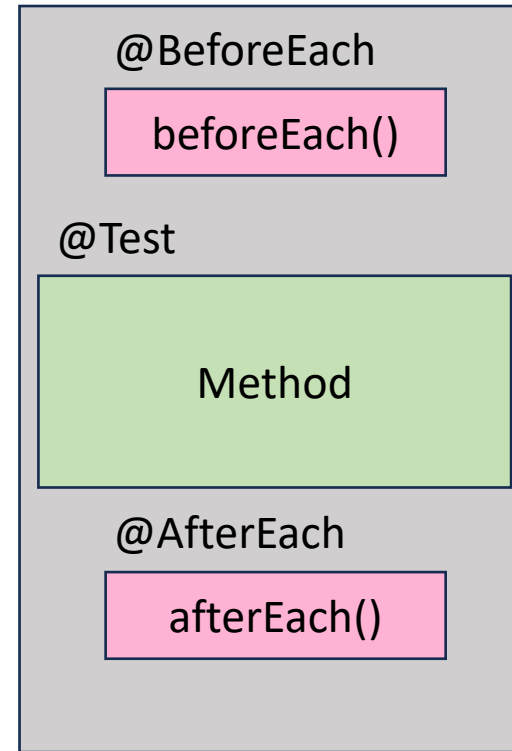
Class Instance 1



Class Instance 2



Class Instance 3



@AfterAll

cleanUp()

JUnit5와 기본문법

```
public class CalculatorTest {  
  
    @BeforeAll  
    static void setup() {  
        System.out.println("@BeforeAll 실행");  
    }  
  
    @AfterAll  
    static void cleanup() {  
        System.out.println("@AfterAll 실행");  
    }  
  
    @BeforeEach  
    void beforeEachTestMethod() {  
        System.out.println("@BeforeEach 실행");  
    }  
  
    @AfterEach  
    void afterEachTestMethod() {  
        System.out.println("@AfterEach 실행");  
    }  
}
```

```
@DisplayName("4/2 테스트")  
@Test  
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {  
  
    System.out.println("4/2 테스트 실행");  
}
```

```
@DisplayName("4/0 테스트")  
@Test  
void integerDivision_WhenDividendIsDividedByZero_ShouldThrowArithmeticException() {  
  
    System.out.println("4/0 테스트 실행");  
}
```

```
@DisplayName("33-1 테스트")  
@Test  
void integerSubtraction_When33Minus1_ShouldReturn32() {  
  
    System.out.println("33-1 테스트 실행");  
}
```

JUnit5와 기본문법

Run: CalculatorTest x

Tests passed: 3 of 3 tests – 7 ms

Test Results	Duration
Test Results	7 ms
basic.CalculatorTest	7 ms
4/0 테스트	6 ms
33-1 테스트	1 ms
4/2 테스트	0 ms

@BeforeAll 실행
@BeforeEach 실행
4/0 테스트 실행
@AfterEach 실행
@BeforeEach 실행
33-1 테스트 실행
@AfterEach 실행
@BeforeEach 실행
4/2 테스트 실행
@AfterEach 실행
@AfterAll 실행

JUnit5와 기본문법

```
@DisplayName("4/2 테스트")
```

```
//@Test
```

```
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {
```

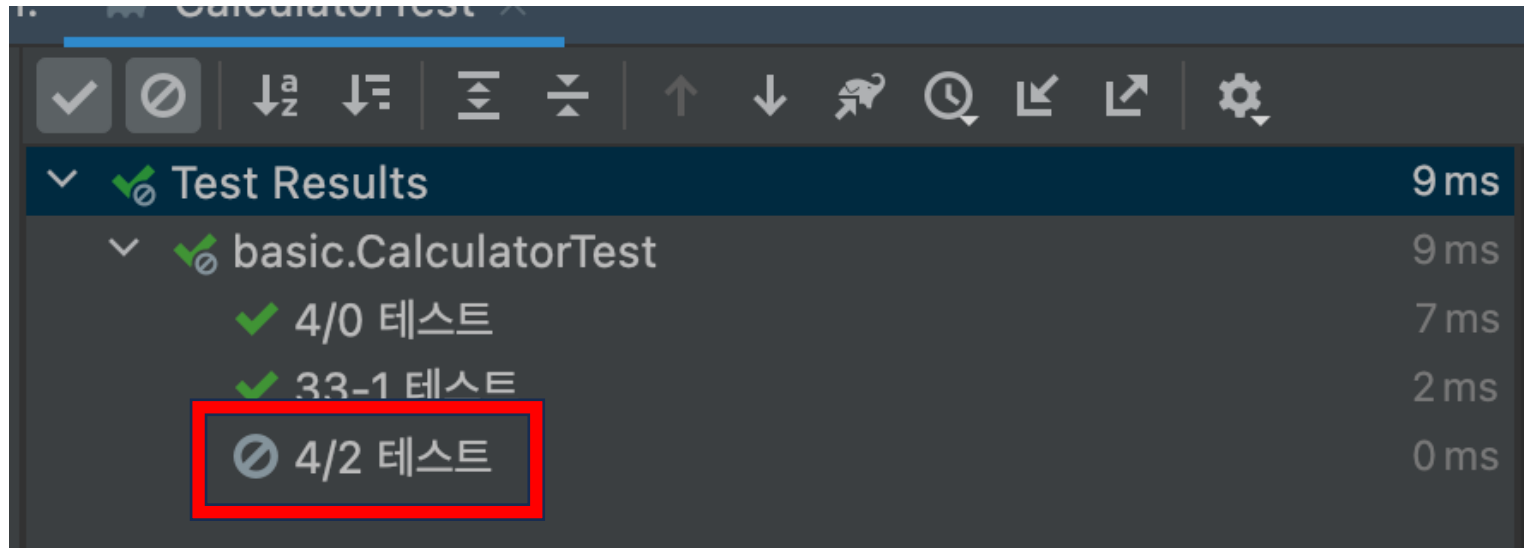
The screenshot shows the Test Results window in an IDE. At the top, there is a toolbar with icons for test actions: a checkmark, a circle with a slash, a downward arrow with 'a', a downward arrow with a list icon, a double arrow, a double arrow with a slash, an upward arrow, a downward arrow, a search icon, a magnifying glass, a left arrow, a right arrow, and a gear icon. Below the toolbar, the test results are listed in a tree view:

- Test Results (8 ms)
 - basic.CalculatorTest (8 ms)
 - 4/0 테스트 (6 ms)
 - 33-1 테스트 (2 ms)

A red box highlights the empty space below the '33-1 테스트' entry.

JUnit5와 기본문법

```
@Disabled
@DisplayName("4/2 테스트")
@Test
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {
```



JUnit5와 기본문법

```
public class CalculatorTest {

    Calculator calculator;

    @BeforeAll
    static void setup() {
        System.out.println("@BeforeAll 실행");
    }

    @AfterAll
    static void cleanup() {
        System.out.println("@AfterAll 실행");
    }

    @BeforeEach
    void beforeEachTestMethod() {
        calculator = new Calculator();
        System.out.println("@BeforeEach 실행");
    }

    @AfterEach
    void afterEachTestMethod() {
        System.out.println("@AfterEach 실행");
    }
}
```

```
@DisplayName("4/2 테스트")
@Test
void integerDivision_WhenFourIsDividedByTwo_ShouldReturnTwo() {

    System.out.println("4/2 테스트 실행");

    // Arrange (given)
    calculator = new Calculator();
}
```

```
@DisplayName("4/0 테스트")
@Test
void integerDivision_WhenDividendIsDividedByZero_ShouldThrowArithmeticException() {

    System.out.println("4/0 테스트 실행");

    // Arrange
    calculator = new Calculator();
}
```

```
@DisplayName("33-1 테스트")
@Test
void integerSubtraction_When33Minus1_ShouldReturn32() {

    System.out.println("33-1 테스트 실행");

    // Arrange
    calculator = new Calculator();
}
```

03

JUnit 심화문법

JUnit5 심화문법

```
@DisplayName("빨셈 테스트")
@ParameterizedTest
@MethodSource("integerSubtraction")
void integerSubtraction_1(int minuend, int subtrahend, int expectedResult) {

    int result = calculator.integerSubtraction(minuend, subtrahend);

    assertEquals(expectedResult, result,
        () -> minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}

private static Stream<Arguments> integerSubtraction() {
    return Stream.of(
        Arguments.of(33, 1, 32),
        Arguments.of(54, 1, 53),
        Arguments.of(24, 1, 23)
    );
}
```

✓ Test Results	16 ms
✓ CalculatorTest	16 ms
✓ 빨셈 테스트	16 ms
✓ [1] 33, 1, 32	15 ms
✓ [2] 54, 1, 53	
✓ [3] 24, 1, 23	1 ms

JUnit5 심화문법

```
@DisplayName("뺄셈 테스트")
@ParameterizedTest
@MethodSource(())
void integerSubtraction_1(int minuend, int subtrahend, int expectedResult) {

    int result = calculator.integerSubtraction(minuend, subtrahend);

    assertEquals(expectedResult, result,
        () -> minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}

private static Stream<Arguments> integerSubtraction_1() {
    return Stream.of(
        Arguments.of(33, 1, 32),
        Arguments.of(54, 1, 53),
        Arguments.of(24, 1, 23)
    );
}
```

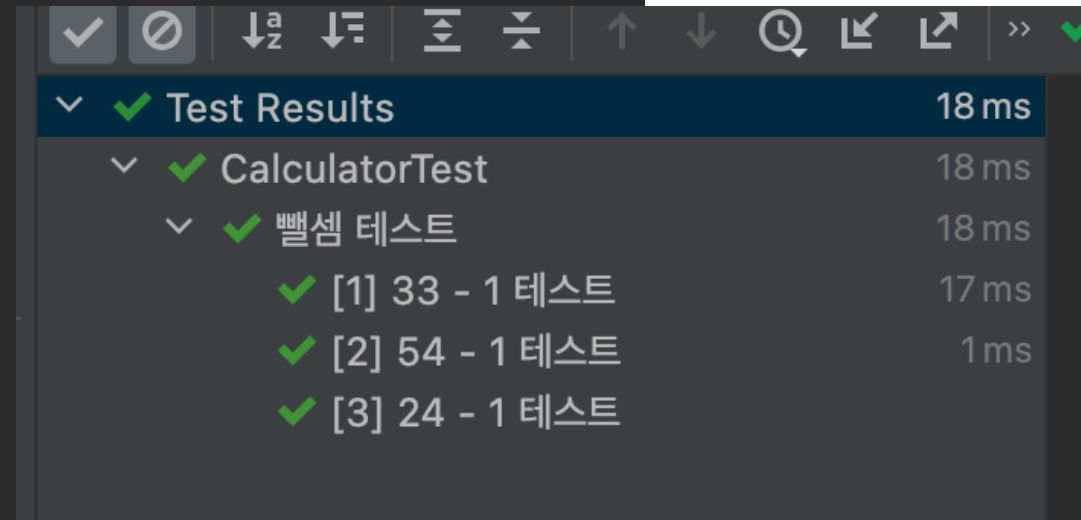
JUnit5 심화문법

```
@DisplayName("뺄셈 테스트")
@ParameterizedTest(name = "[{index}] {0} - {1} 테스트")
@MethodSource()
void integerSubtraction_1(int minuend, int subtrahend, int expectedResult) {

    int result = calculator.integerSubtraction(minuend, subtrahend);

    assertEquals(expectedResult, result,
        () -> minuend + "-" + subtrahend + " 결과가 " + expectedResult + "와 다릅니다.");
}

private static Stream<Arguments> integerSubtraction_1() {
    return Stream.of(
        Arguments.of(33, 1, 32),
        Arguments.of(54, 1, 53),
        Arguments.of(24, 1, 23)
    );
}
```



Test Results	Duration
Test Results	18 ms
CalculatorTest	18 ms
뺄셈 테스트	18 ms
[1] 33 - 1 테스트	17 ms
[2] 54 - 1 테스트	1 ms
[3] 24 - 1 테스트	

JUnit5 심화문법

```

@DisplayName("뺄셈 테스트 CsvSource")
@ParameterizedTest(name = "[{index}] {0} - {1} 테스트")
@CsvSource({
    "20, 17, 3",
    "17, 12, 5",
    "26, 24, 2"
})
void integerSubtraction_2(int minuend, int subtrahend, int expectedResult) {

    int result = calculator.integerSubtraction(minuend, subtrahend);
    assertEquals(expectedResult, result,
        () -> minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}

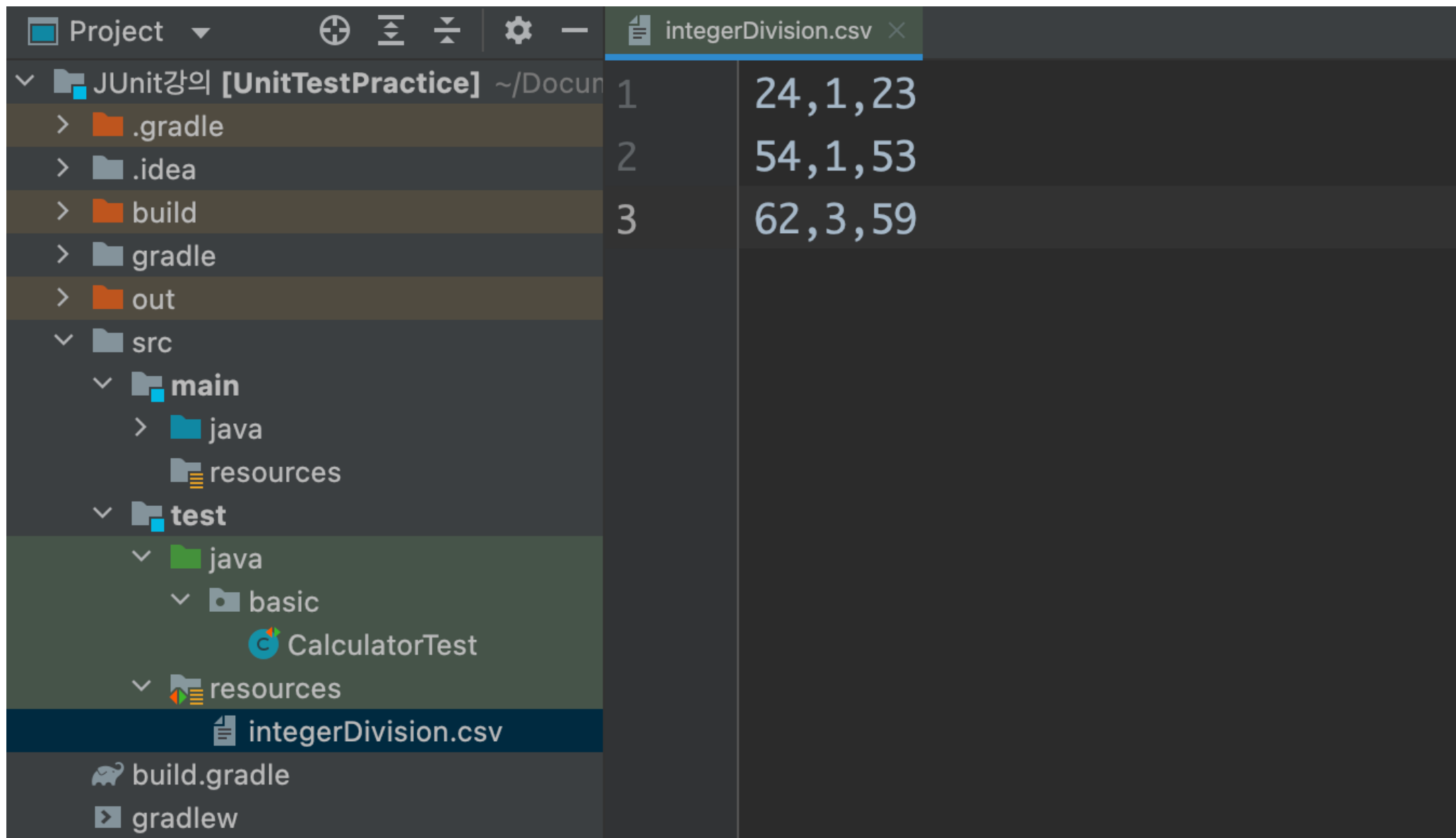
```

✓ Test Results	18 ms
✓ CalculatorTest	18 ms
✓ 뺄셈 테스트 CsvSource	18 ms
✓ [1] 20 - 17 테스트	16 ms
✓ [2] 17 - 12 테스트	1 ms
✓ [3] 26 - 24 테스트	1 ms

JUnit5 심화문법

The image shows an IDE interface with a 'New File' dialog box in the foreground. The dialog contains the text 'integerDivision.csv'. Behind the dialog, a file explorer shows a project structure with folders like 'src', 'main', 'resources', and 'test'. A context menu is open over the 'resources' folder, with 'New' selected, and a sub-menu showing various file types like 'Kotlin Class/File', 'File', 'Scratch File', etc.

JUnit5 심화문법



The screenshot shows an IDE interface with a project structure on the left and a CSV file on the right. The project structure is as follows:

- Project
- JUnit강의 [UnitTestPractice] ~/Docum
 - .gradle
 - .idea
 - build
 - gradle
 - out
 - src
 - main
 - java
 - resources
 - test
 - java
 - basic
 - CalculatorTest
 - resources
 - integerDivision.csv
 - build.gradle
 - gradlew

The CSV file, `integerDivision.csv`, contains the following data:

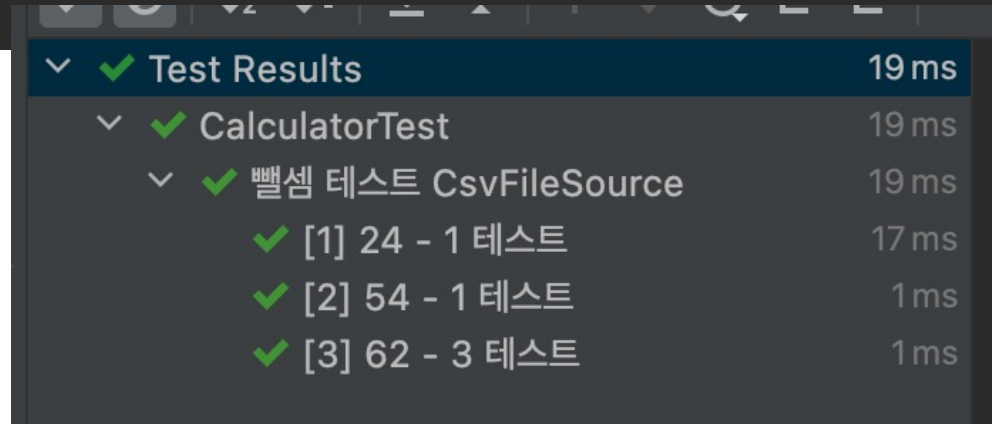
1	24,1,23
2	54,1,53
3	62,3,59

JUnit5 심화문법

```
@DisplayName("빨셈 테스트 CsvFileSource")
@ParameterizedTest(name = "[{index}] {0} - {1} 테스트")
@CsvFileSource(resources = "/integerDivision.csv")
void integerSubtraction4(int minuend, int subtrahend, int expectedResult) {

    int result = calculator.integerSubtraction(minuend, subtrahend);

    assertEquals(expectedResult, result,
        () -> minuend + "-" + subtrahend + " 결과값이 " + expectedResult + "와 다릅니다.");
}
```

A screenshot of the JUnit5 Test Results window. It shows a hierarchical tree of test results. The root is 'Test Results' with a green checkmark and a duration of 19 ms. Underneath is 'CalculatorTest' with a green checkmark and 19 ms. Below that is '빨셈 테스트 CsvFileSource' with a green checkmark and 19 ms. This category contains three sub-items, each with a green checkmark: '[1] 24 - 1 테스트' (17 ms), '[2] 54 - 1 테스트' (1 ms), and '[3] 62 - 3 테스트' (1 ms).

✓ Test Results	19 ms
✓ CalculatorTest	19 ms
✓ 빨셈 테스트 CsvFileSource	19 ms
✓ [1] 24 - 1 테스트	17 ms
✓ [2] 54 - 1 테스트	1 ms
✓ [3] 62 - 3 테스트	1 ms

JUnit5 심화문법

```
@DisplayName("ValueSource 테스트")
@ParameterizedTest
@ValueSource(strings={"김선우", "김건희", "땅울림"})
void valueSourceDemonstration(String firstname){
    System.out.println(firstname);
    assertNotNull(firstname);
}
```

```
Tests passed: 3 of 3 tests - 17 ms
Test Results 17 ms
  CalculatorTest 17 ms
    ValueSource 테스트 17 ms
      [1] 김선우 16 ms
      [2] 김건희
      [3] 땅울림 1 ms
@BeforeAll 실행
@BeforeEach 실행
김선우
@AfterEach 실행
@BeforeEach 실행
김건희
@AfterEach 실행
@BeforeEach 실행
땅울림
@AfterEach 실행
@AfterAll 실행
```

JUnit5 심화문법

```
6  
7 @TestMethodOrder(MethodOrderer.Random.class)  
8 public class MethodOrderedRandomlyTest {  
9  
10     @Test  
11     void testA(){  
12         System.out.println("랜덤 A 실행");  
13     }  
14  
15     @Test  
16     void testB(){  
17         System.out.println("랜덤 B 실행");  
18     }  
19  
20     @Test  
21     void testC(){  
22         System.out.println("랜덤 C 실행");  
23     }  
24  
25     @Test  
26     void testD(){  
27         System.out.println("랜덤 D 실행");  
28     }  
29 }
```

Test Results 10 ms

- MethodOrderedRandomlyTest 10 ms
 - testD() 8 ms
 - testB() 1 ms
 - testC() 1 ms
 - testA() 1 ms

/Users/jameskim/
랜덤 D 실행
랜덤 B 실행
랜덤 C 실행
랜덤 A 실행

JUnit5 심화문법

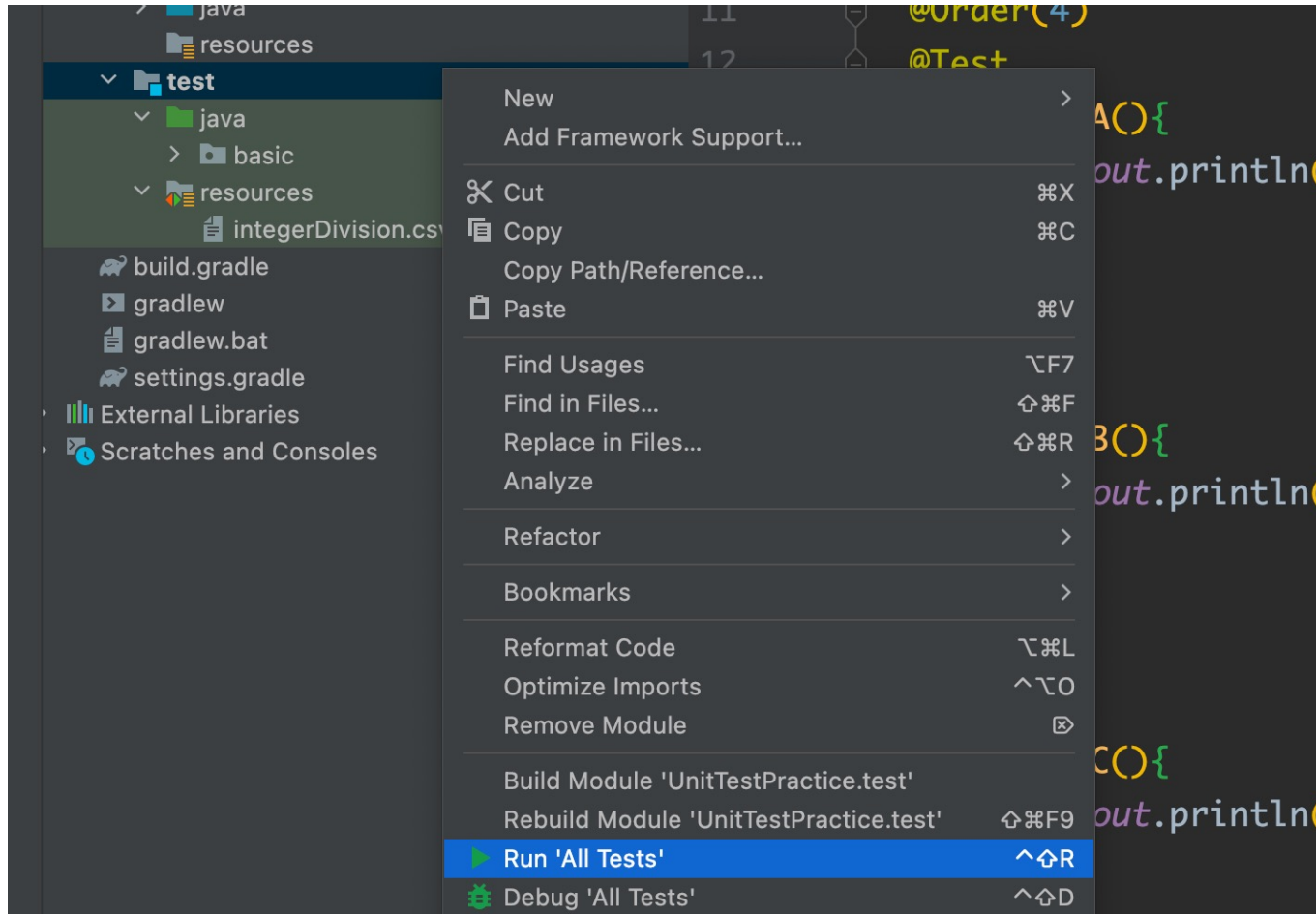
```
Project
├── JUnit강의 [JUnitPractice] ~/Documents/샘머코딩
│   ├── .gradle
│   ├── .idea
│   ├── build
│   ├── gradle
│   ├── out
│   └── src
│       ├── main
│       │   ├── java
│       │   └── resources
│       └── test
│           ├── java
│           │   └── basic
│           │       ├── CalculatorTest
│           │       ├── MethodOrderedByOrderIndexTest
│           │       └── MethodOrderedRandomlyTest
│           └── resources
│               └── integerDivision.csv
├── build.gradle
├── gradlew
├── gradlew.bat
├── settings.gradle
├── External Libraries
└── Scratches and Consoles
```

```
MethodOrderedByOrderIndexTest.java
8   @TestMethodOrder(MethodOrderer.OrderAnnotation.class)
9   public class MethodOrderedByOrderIndexTest {
10
11      @Order(4)
12      @Test
13      void testA(){
14          System.out.println("Order 어노테이션 A 실행");
15      }
16
17      @Order(3)
18      @Test
19      void testB(){
20          System.out.println("Order 어노테이션 B 실행");
21      }
22
23      @Order(2)
24      @Test
25      void testC(){
26          System.out.println("Order 어노테이션 C 실행");
27      }
28
29      @Order(1)
30      @Test
31      void testD(){
32          System.out.println("Order 어노테이션 D 실행");
33      }
34  }
```

Test Results		9 ms
MethodOrderedByOrderIndexTest		9 ms
testD()	✓	7 ms
testC()	✓	
testB()	✓	1 ms
testA()	✓	1 ms

```
/Users/jameskim/Library/Java/Java7
Order 어노테이션 D 실행
Order 어노테이션 C 실행
Order 어노테이션 B 실행
Order 어노테이션 A 실행
```

JUnit5 심화문법



✓ <default package>	31 ms
<ul style="list-style-type: none"> ✓ MethodOrderedByOrderIndexTest <ul style="list-style-type: none"> ✓ testD() 9 ms ✓ testC() 1 ms ✓ testB() ✓ testA() ✓ CalculatorTest 20 ms <ul style="list-style-type: none"> ✓ 4/0 테스트 5 ms > ✓ 뱀셈 테스트 CsvFileSource 12 ms > ✓ ValueSource 테스트 3 ms <ul style="list-style-type: none"> ✓ 33-1 테스트 ✓ 4/2 테스트 > ✓ 뱀셈 테스트 > ✓ 뱀셈 테스트 CsvSoruce ✓ MethodOrderedRandomlyTest 1 ms <ul style="list-style-type: none"> ✓ testD() ✓ testC() ✓ testB() 1 ms ✓ testA() 	

JUnit5 심화문법

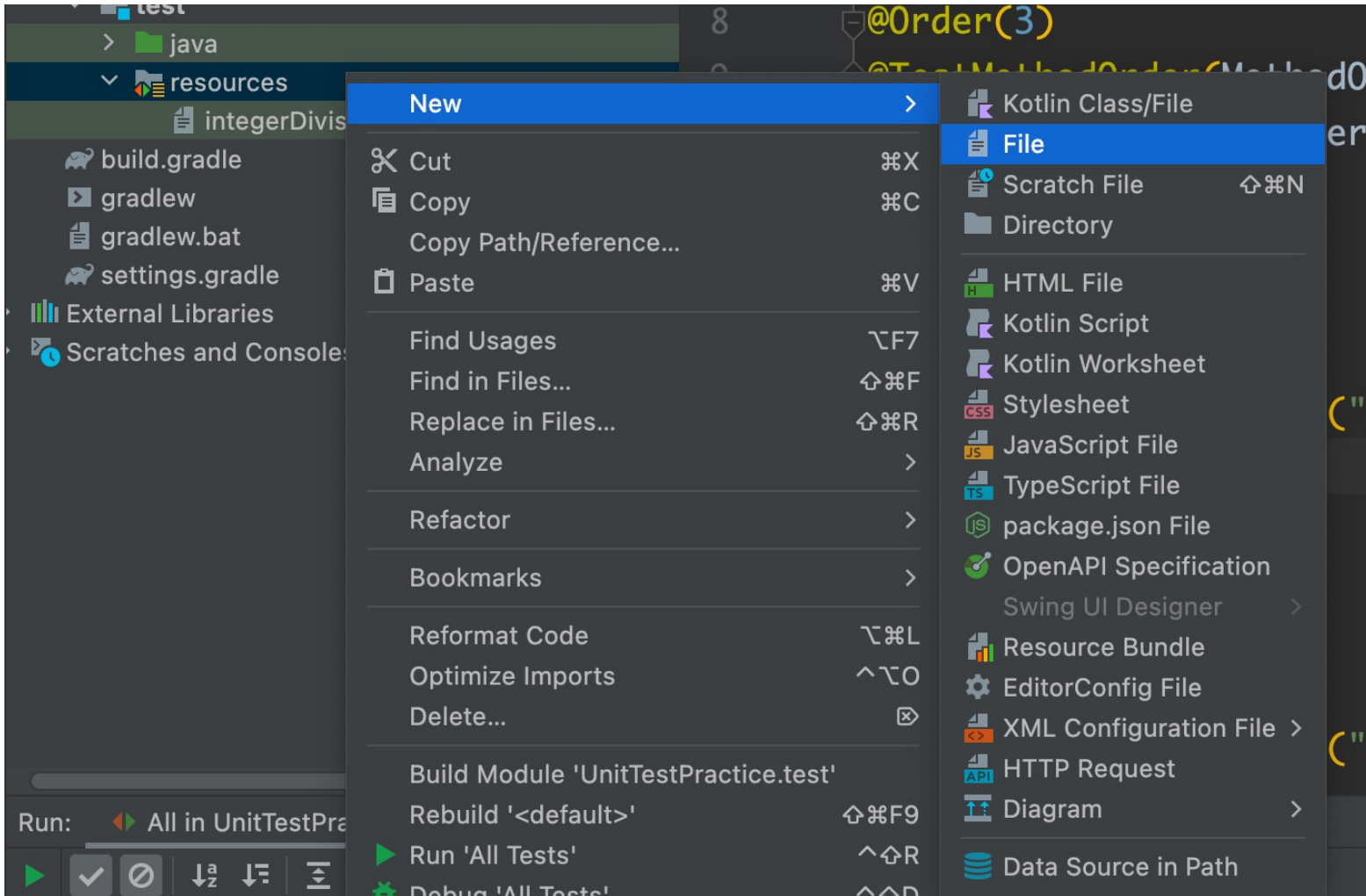
```
@Order(1)
@TestMethodOrder(MethodOrderer.Random.class)
public class MethodOrderedRandomlyTest {
```

```
@Order(2)
public class CalculatorTest {
```

```
@Order(3)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class MethodOrderedByOrderIndexTest {
```

JUnit5 심화문법

New File
junit-platform.properties



JUnit5 심화문법

The screenshot shows an IDE with a project named 'JUnit강의 [JUnitTestPractice]'. The file explorer on the left shows the project structure, including 'src/main/java' and 'src/test/java'. The 'junit-platform.properties' file is open in the editor, containing the property: `junit.jupiter.testclass.order.default=org.junit.jupiter.api.ClassOrderer$OrderAnnotation`. A test results window is overlaid on the right, showing a list of tests with their execution times:

Test Name	Execution Time
<default package>	34 ms
MethodOrderedRandomlyTest	10 ms
CalculatorTest	23 ms
MethodOrderedByOrderIndexTest	1 ms
testD()	
testC()	1 ms
testB()	
testA()	

JUnit5 심화문법

```
10
11 @TestInstance(Lifecycle.PER_METHOD)
12 @TestMethodOrder(MethodOrderer.OrderAnnotation.class)
13 public class LifeCycleTest {
14
15     StringBuilder string = new StringBuilder("");
16
17     @AfterEach
18     void afterEach() {
19         System.out.println("string 상태: " + string);
20     }
21
22     @Order(1)
23     @Test
24     void testA(){
25         System.out.println("Order 어노테이션 A 실행");
26         string.append(1);
27     }
28
29     @Order(2)
30     @Test
31     void testB(){
32         System.out.println("Order 어노테이션 B 실행");
33         string.append(2);
34     }

```

Tests passed: 4 of 4 tests - 13 ms

Test Results	Time
LifeCycleTest	13 ms
testA()	10 ms
testB()	1 ms
testC()	1 ms
testD()	1 ms

Order 어노테이션 A 실행
string 상태: 1
Order 어노테이션 B 실행
string 상태: 2
Order 어노테이션 C 실행
string 상태: 3
Order 어노테이션 D 실행
string 상태: 4

```
@Order(3)
@Test
void testC(){
    System.out.println("Order 어노테이션 C 실행");
    string.append(3);
}

@Order(4)
@Test
void testD(){
    System.out.println("Order 어노테이션 D 실행");
    string.append(4);
}

```

JUnit5 심화문법

```
@TestInstance(Lifecycle.PER_CLASS)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class LifeCycleTest {

    StringBuilder string = new StringBuilder("");
```

✓ Test Results	14 ms	/Users/jameskim/Library/Java/Java
✓ LifeCycleTest	14 ms	Order 어노테이션 A 실행
✓ testA()	11 ms	string 상태: 1
✓ testB()	1 ms	Order 어노테이션 B 실행
✓ testC()	1 ms	string 상태: 12
✓ testD()	1 ms	Order 어노테이션 C 실행
		string 상태: 123
		Order 어노테이션 D 실행
		string 상태: 1234

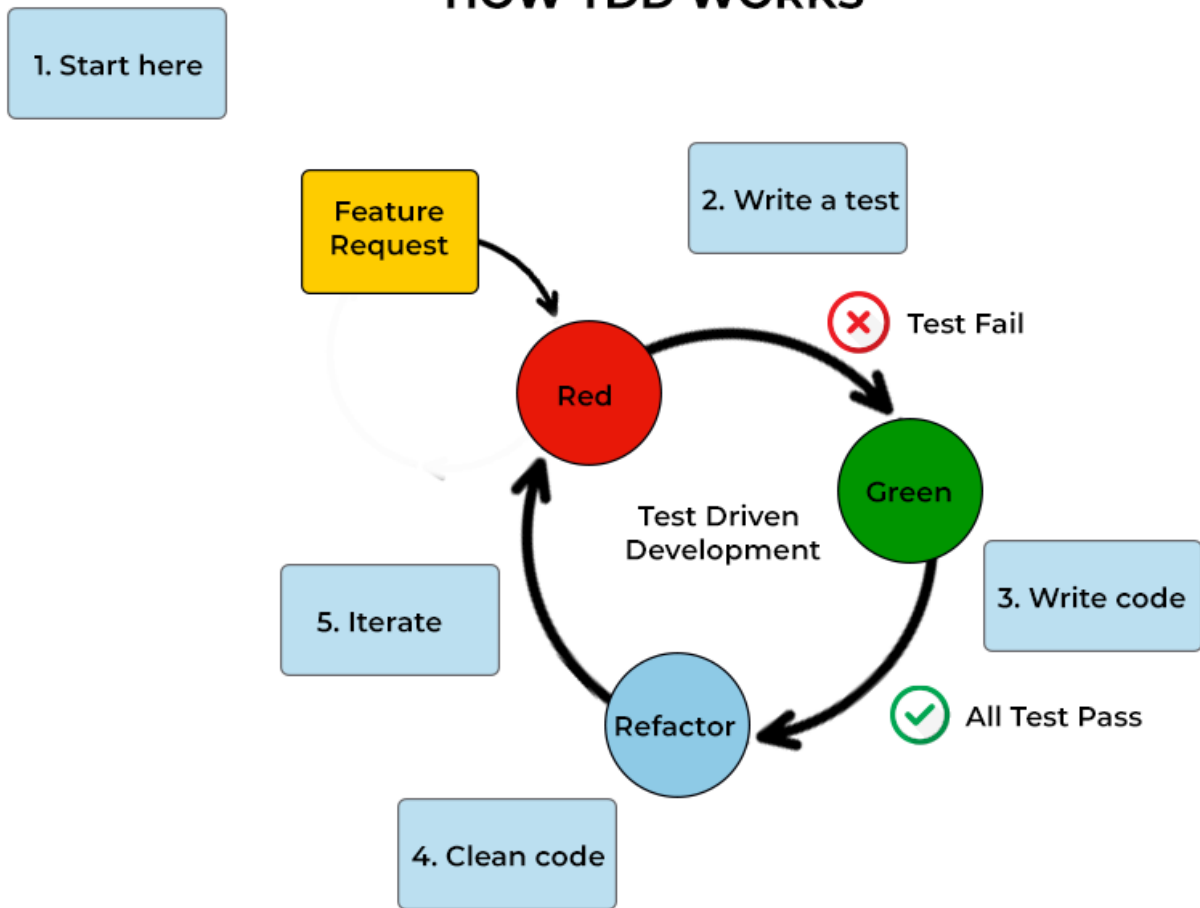
04

테스트 주도 개발(TDD)

테스트 주도 개발(TDD)



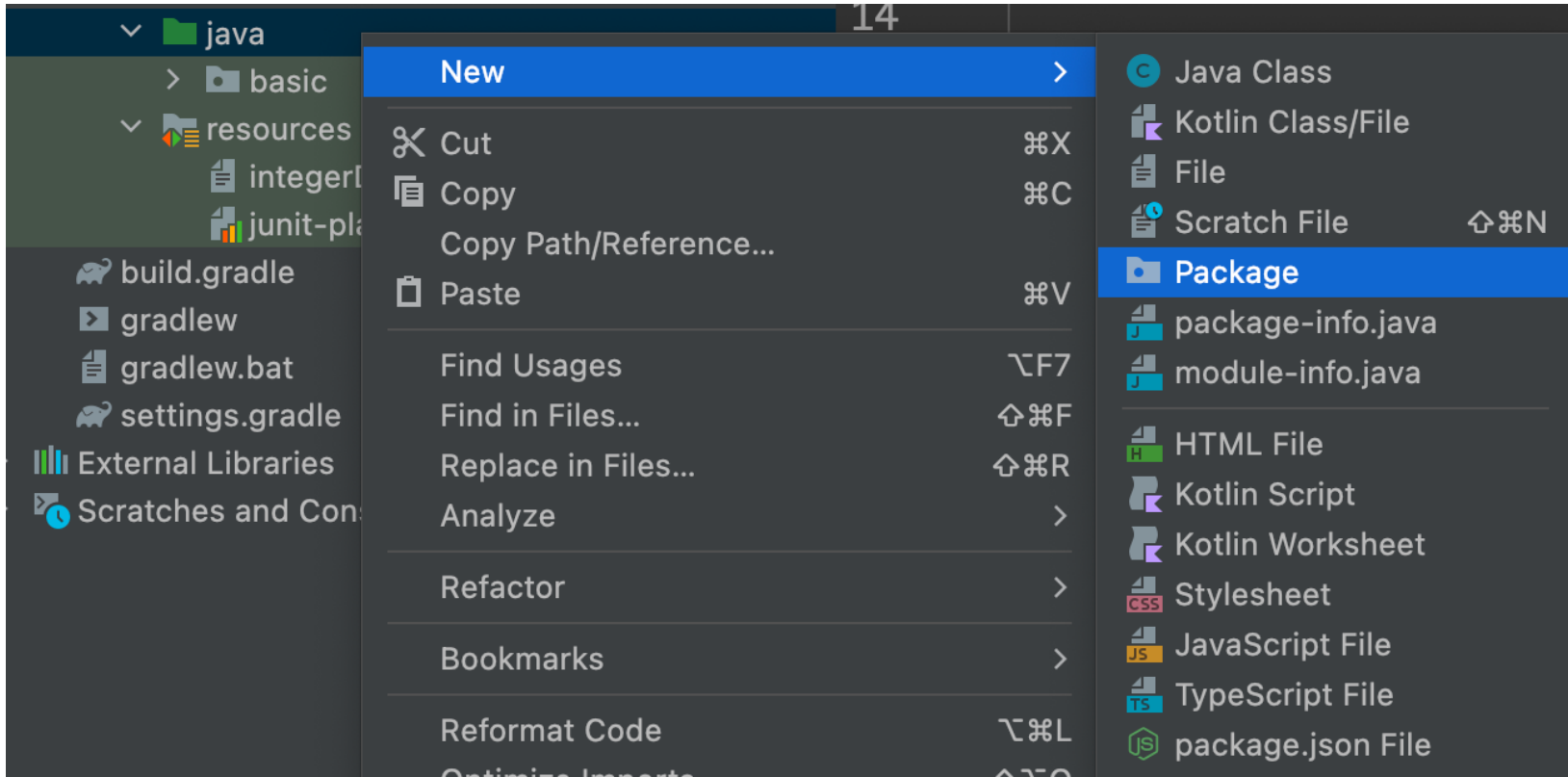
HOW TDD WORKS



요구사항

회원가입을 구현하시오
회원가입시 이름, 이메일, 비밀번호만 요구

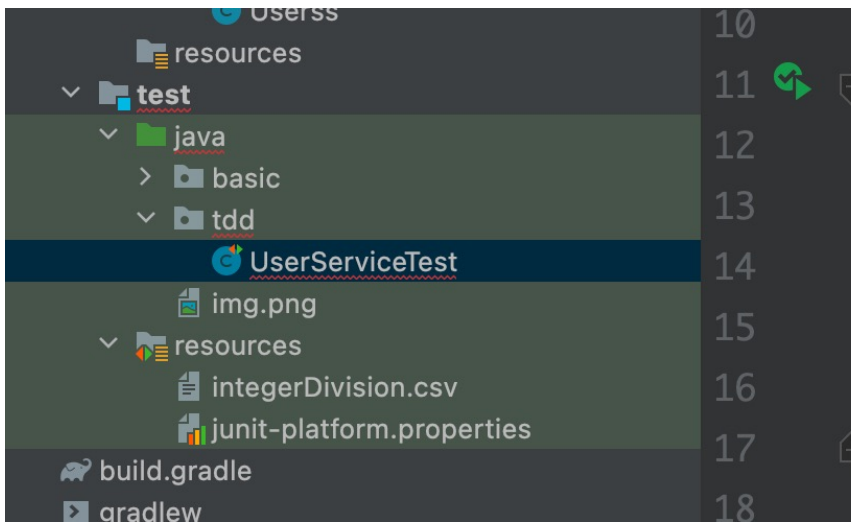
테스트 주도 개발(TDD)



New Package

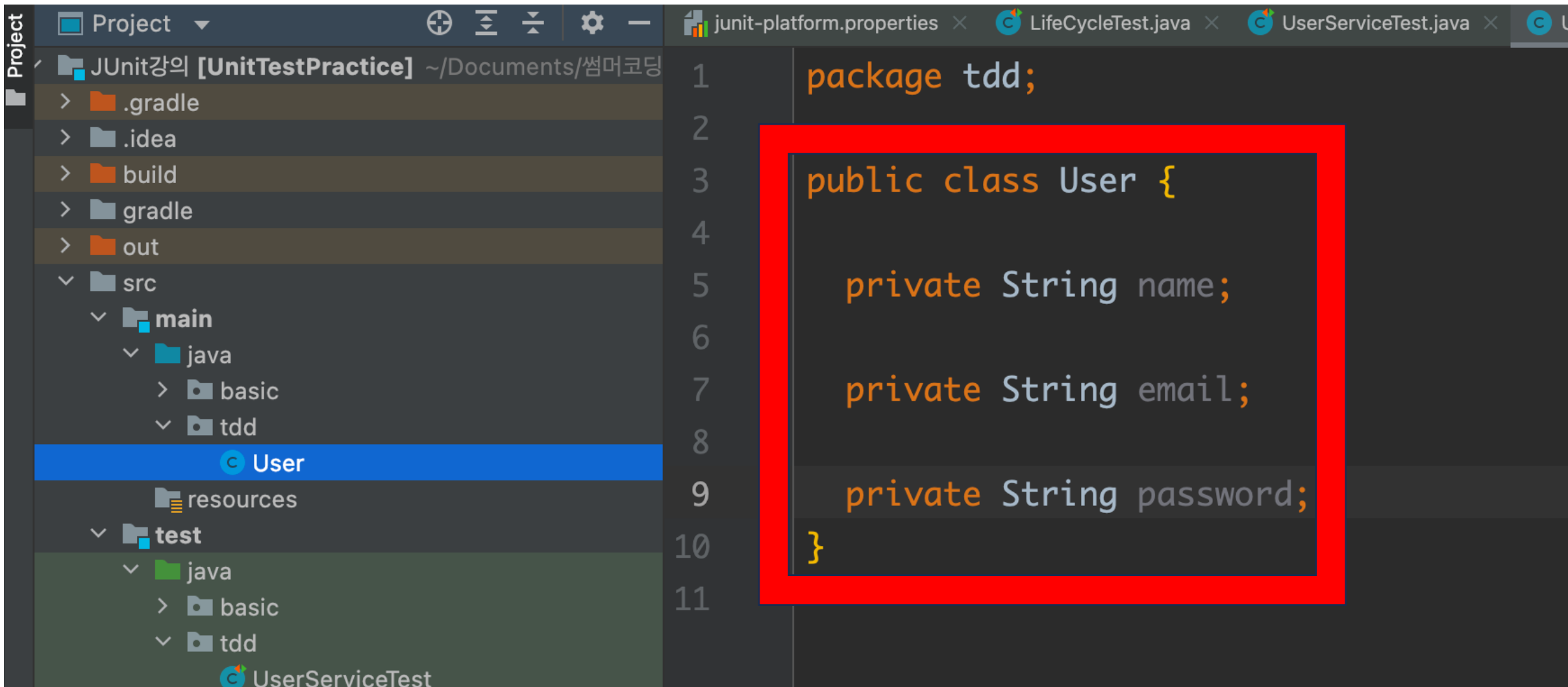
tdd

테스트 주도 개발(TDD)



```
10  
11  
12  
13  
14  
15  
16  
17  
18  
  
@Test  
public void createUser_whenUserDetailsProvided_returnUserObject(){  
  
    User user = userService.createUser();  
  
    assertNotNull(user, "회원가입한 사용자는 null이면 안됩니다.");  
  
}
```

테스트 주도 개발(TDD)



The screenshot shows an IDE interface with a project explorer on the left and a code editor on the right. The project explorer displays the following structure:

- Project
- JUnit강의 [UnitTestPractice] ~/Documents/쌤머코딩
 - .gradle
 - .idea
 - build
 - gradle
 - out
 - src
 - main
 - java
 - basic
 - tdd
 - resources
 - test
 - java
 - basic
 - tdd

The code editor shows the following Java code:

```
1 package tdd;  
2  
3 public class User {  
4  
5     private String name;  
6  
7     private String email;  
8  
9     private String password;  
10 }  
11
```

The code editor also shows tabs for `junit-platform.properties`, `LifeCycleTest.java`, and `UserServiceTest.java`. The `User` class is highlighted in blue in the project explorer, and the code editor shows a red box around the class definition.

테스트 주도 개발(TDD)

```
1 package tdd;
2
3 public interface UserService {
4
5     1 related problem
6     User createUser();
7
8 }
```

The screenshot shows an IDE interface. On the left is a file explorer for a project named 'JUnit강의 [UnitTestPractice]'. The project structure includes folders for '.gradle', '.idea', 'build', 'gradle', 'out', 'src', 'resources', and 'test'. Under 'src/main/java', there are subfolders 'basic' and 'tdd'. The 'tdd' folder contains two files: 'User' and 'UserService'. The 'UserService' file is selected and highlighted in blue. On the right is a code editor showing the content of 'UserService.java'. The code is as follows:

테스트 주도 개발(TDD)

The screenshot shows an IDE window with a project structure on the left and a code editor on the right. The project structure includes a 'src/main/java/tdd' directory containing 'User', 'UserService', and 'UserServiceImpl' files. The code editor displays the following Java code for 'UserServiceImpl.java':

```
1 package tdd;
2
3 public class UserServiceImpl implements UserService {
4
5     @Override
6     public User createUser() {
7         return new User();
8     }
9 }
10
```

A red rectangular box highlights the class definition and the `createUser()` method implementation.

테스트 주도 개발(TDD)

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes folders for 'src/main/java/basic', 'src/main/java/tdd', and 'src/test/java/tdd'. The 'UserServiceImpl' class is highlighted in the 'src/main/java/tdd' folder. The code editor shows the following Java code:

```
4 import org.junit.jupiter.api.Test;
5
6 public class UserServiceTest {
7
8     UserService userService = new UserServiceImpl();
9
10    @Test
11    public void createUser_whenUserDetailsProvided_returnUserObject(){
12
13        User user = userService.createUser();
14
15        Assertions.assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
16    }
17
18 }
19
```

At the bottom, the 'Run' window shows the test execution results:

```
Run: UserServiceTest.createUser_whenUserDetailsProvided... x
> Tests passed: 1 of 1 test - 12 ms
Test Results 12 ms
  UserServiceTest 12 ms
    createUser_whenUserDe 12 ms
```

The process finished with exit code 0.

테스트 주도 개발(TDD)

```
@Test
```

```
public void createUser_whenUserDetailsProvided_returnUserObject(){
```

```
    String name = "홍길동";
```

```
    String email = "gildong@naver.com";
```

```
    String password = "qwerty123";
```

```
    User user = userService.createUser(name, email, password);
```

```
    Assertions.assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
```

```
}
```

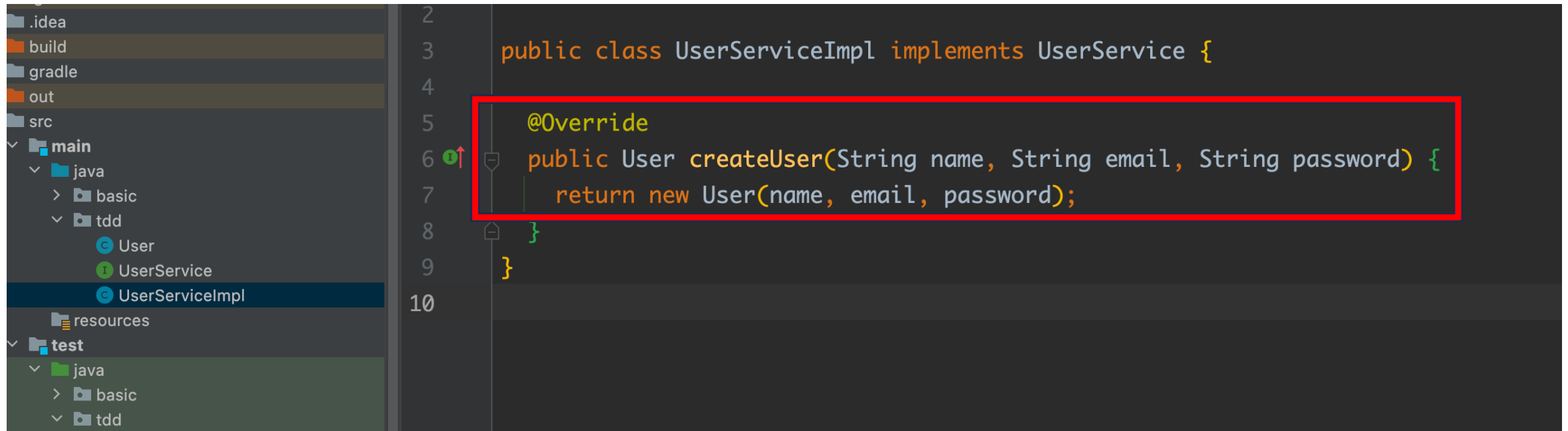
```
}
```

테스트 주도 개발(TDD)

The screenshot shows an IDE with a project named 'JUnit강의 [UnitTestPractice]'. The file explorer on the left shows the project structure, including a 'test' directory with a 'tdd' subdirectory containing 'UserServiceTest'. The main editor displays the 'User' class with the following code:

```
2  
3 1 related problem  
4 public class User {  
5     private String name;  
6  
7     private String email;  
8  
9     private String password;  
10  
11 1 related problem  
12 public User(String name, String email, String password) {  
13     this.name = name;  
14     this.email = email;  
15     this.password = password;  
16 }
```

테스트 주도 개발(TDD)



```
2  
3 public class UserServiceImpl implements UserService {  
4  
5     @Override  
6     public User createUser(String name, String email, String password) {  
7         return new User(name, email, password);  
8     }  
9 }  
10
```

The screenshot shows an IDE interface. On the left is a project explorer with a tree view containing folders like .idea, build, gradle, out, src, main, java, basic, tdd, resources, and test. The file 'UserServiceImpl.java' is selected. The main editor area shows the code for 'UserServiceImpl' which implements 'UserService'. The method 'createUser' is highlighted with a red box, showing the '@Override' annotation and the implementation 'return new User(name, email, password);'. A green arrow icon is visible next to line 6.

테스트 주도 개발(TDD)

The screenshot displays an IDE with a project structure on the left and a code editor on the right. The project structure includes a `test` directory with a `UserServiceTest` class. The code editor shows the following Java code:

```
9
10 @Test
11 public void createUser_whenUserDetailsProvided_returnUserObject(){
12
13     String name = "홍길동";
14     String email = "gildong@naver.com";
15     String password = "qwerty123";
16
17     User user = userService.createUser(name, email, password);
18
19     Assertions.assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
20 }
21
22
```

Below the code editor, the IDE shows the execution results for the test. The `Run` tab indicates that the test passed successfully:

```
Run: UserServiceTest.createUser_whenUserDetailsProvided... x
>> Tests passed: 1 of 1 test - 10 ms
```

The `Test Results` tab shows the following details:

- Test Results: 10 ms
- UserServiceTest: 10 ms
- createUser_whenUserDe: 10 ms

The output window at the bottom shows the process finished with exit code 0:

```
/Users/jameskim/Library/Java/JavaVirtualMachines/corretto-17.0.7/Contents/Home/bin/java ...
Process finished with exit code 0
```

테스트 주도 개발(TDD)

```
@Test
```

```
public void createUser_whenUserDetailsProvided_returnUserObject(){
```

```
    String name = "홍길동";
```

```
    String email = "gildong@naver.com";
```

```
    String password = "qwerty123";
```

```
    User user = userService.createUser(name, email, password);
```

```
    assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
```

```
    assertEquals(name, user.getName());
```

```
}
```


테스트 주도 개발(TDD)

```
public class User {  
  
    private String name;  
  
    private String email;  
  
    private String password;  
  
    public User(String name, String email, String password) {  
        this.name = name;  
        this.email = email;  
        this.password = password;  
    }  
  
    public String getName() {  
        return name;  
    }  
}
```

테스트 주도 개발(TDD)

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a 'main' package with 'java' and 'resources' sub-packages, and a 'test' package with 'java' and 'resources' sub-packages. The 'test/java' package contains 'UserServiceTest'. The code editor shows the following Java code:

```
12
13 @Test
14 public void createUser_whenUserDetailsProvided_returnUserObject(){
15
16     String name = "홍길동";
17     String email = "gildong@naver.com";
18     String password = "qwerty123";
19
20     User user = userService.createUser(name, email, password);
21
22     assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
23     assertEquals(name, user.getName());
24 }
25
26
```

At the bottom, the 'Run' window shows the test execution results:

```
Run: UserServiceTest.createUser_whenUserDetailsProvided... x
>> Tests passed: 1 of 1 test - 10 ms
Test Results 10 ms
  UserServiceTest 10 ms
    createUser_whenUserDe 10 ms
```

The console output shows the path to the Java runtime and the exit code:

```
/Users/jameskim/Library/Java/JavaVirtualMachines/corretto-17.0.7/Contents/Home/bin/java ...
Process finished with exit code 0
```

테스트 주도 개발(TDD)

```
@Test
```

```
public void createUser_whenUserDetailsProvided_returnUserObject(){
```

```
    String name = "홍길동";
```

```
    String email = "gildong@naver.com";
```

```
    String password = "qwerty123";
```

```
    User user = userService.createUser(name, email, password);
```

```
    assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
```

```
    assertEquals(name, user.getName());
```

```
    assertEquals(email, user.getEmail());
```

```
    assertNotNull(user.getId(), "사용자 id가 존재하지 않습니다.");
```

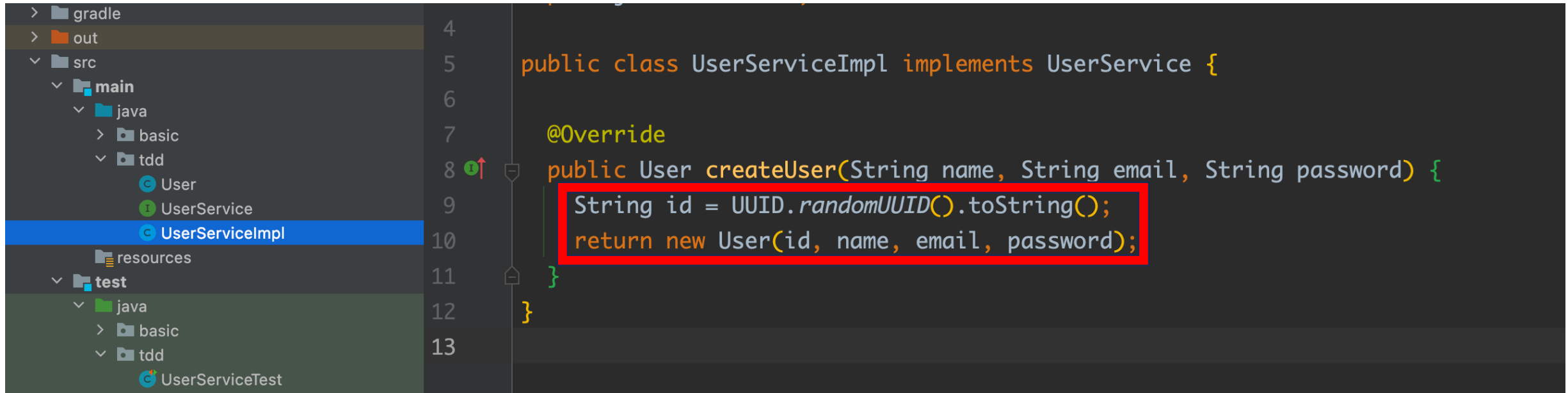
```
}
```

테스트 주도 개발(TDD)

```
JUnit강의 [Unit testPractice] ~/Documents/샘머코
> .gradle
> .idea
> build
> gradle
> out
> src
  > main
    > java
      > basic
      > tdd
        > User
        > UserService
        > UserServiceImpl
    > resources
  > test
    > java
      > basic
      > tdd
        > UserServiceTest
        > img.png
        > resources
          > integerDivision.csv
          > junit-platform.properties
    > build.gradle
    > gradlew
    > gradlew.bat
    > settings.gradle
  > External Libraries
  > Scratches and Consoles

1 package tdd;
2
3 public class User {
4
5     private String id;
6
7     private String name;
8
9     private String email;
10
11    private String password;
12
13    public User(String id, String name, String email, String password) {
14        this.id = id;
15        this.name = name;
16        this.email = email;
17        this.password = password;
18    }
19
20    public String getId() {
21        return id;
22    }
23
24    public String getName() {
25        return name;
26    }
27
28    public String getEmail() {
29        return email;
30    }
31 }
```

테스트 주도 개발(TDD)



```
4  
5 public class UserServiceImpl implements UserService {  
6  
7     @Override  
8     public User createUser(String name, String email, String password) {  
9         String id = UUID.randomUUID().toString();  
10        return new User(id, name, email, password);  
11    }  
12 }  
13
```

테스트 주도 개발(TDD)

The screenshot displays an IDE with a project structure on the left and a code editor on the right. The project structure includes a 'test' directory with sub-directories 'basic' and 'tdd'. The 'UserServiceTest' class is selected in the 'tdd' directory. The code editor shows the following Java code:

```
11
12
13 @Test
14 public void createUser_whenUserDetailsProvided_returnUserObject(){
15
16     String name = "홍길동";
17     String email = "gildong@naver.com";
18     String password = "qwerty123";
19
20     User user = userService.createUser(name, email, password);
21
22     assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
23     assertEquals(name, user.getName());
24     assertEquals(email, user.getEmail());
25     assertNotNull(user.getId(), message: "사용자 id가 존재하지 않습니다.");
26 }
27
```

At the bottom, the IDE shows the execution results for the test. The test passed successfully in 17 ms. The output window shows the process finished with exit code 0.

Test Results: 17 ms

- UserServiceTest: 17 ms
 - createUser_whenUserDe: 17 ms

Process finished with exit code 0

테스트 주도 개발(TDD)

직접 해보세요!

```
@DisplayName("사용자 생성시 빈 이름 추가")
@Test
void createUser_whenNameIsEmpty_throwsIllegalArgumentException() {
}
}
```

테스트 주도 개발(TDD)

```
public class UserServiceTest {  
  
    UserService userService = new UserServiceImpl();  
  
    String name = "홍길동";  
    String email = "gildong@naver.com";  
    String password = "qwerty123";  
  
    @DisplayName("사용자 생성")  
    @Test  
    public void createUser_whenUserDetailsProvided_returnUserObject() {  
  
        User user = userService.createUser(name, email, password);  
  
        assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");  
    }  
}
```


테스트 주도 개발(TDD)

```
@DisplayName("사용자 생성시 빈 이름 추가")
@Test
void createUser_whenNameIsEmpty_throwsIllegalArgumentException() {

    name = "";

    Assertions.assertThrows(IllegalArgumentException.class, () -> {
        userService.createUser(name, email, password);
    });
}
```

테스트 주도 개발(TDD)

```
public class UserServiceImpl implements UserService {  
  
    @Override  
    public User createUser(String name, String email, String password) {  
  
        if(name == null || name.trim().length() == 0){  
            throw new IllegalArgumentException("이름이 존재하지 않습니다.");  
        }  
  
        String id = UUID.randomUUID().toString();  
        return new User(id, name, email, password);  
    }  
}
```

테스트 주도 개발(TDD)

The screenshot shows an IDE with a project structure on the left and a code editor in the center. The project structure includes a 'basic' directory and a 'tdd' directory containing 'UserServiceTest', 'img.png', 'resources' (with 'integerDivision.csv' and 'junit-platform.properties'), 'build.gradle', 'gradlew', 'gradlew.bat', and 'settings.gradle'. The code editor displays the following Java code for 'UserServiceTest':

```
28 }
29
30 @DisplayName("사용자 생성시 빈 이름 추가")
31 @Test
32 void createUser_whenNameIsEmpty_throwsIllegalArgumentException() {
33
34     name = "";
35
36     Assertions.assertThrows(IllegalArgumentException.class, () -> {
37         userService.createUser(name, email, password);
38     });
39
40 }
41
42 }
```

Below the code editor, the 'Run' window shows the execution of 'UserServiceTest.createUser_whenNameIsEmpty_throw...'. The test results indicate that the test passed successfully in 9 ms. The console output shows the path to the Java runtime and the message 'Process finished with exit code 0'.

테스트 주도 개발(TDD)

```
@DisplayName("사용자 생성시 빈 이름 추가")
@Test
void createUser_whenNameIsEmpty_throwsIllegalArgumentException() {
    name = "";
    String expectedExceptionMessage = "이름이 존재하지 않습니다.";

    IllegalArgumentException thrown = Assertions.assertThrows(
        IllegalArgumentException.class, () -> {
            userService.createUser(name, email, password);
        }, message: "이름이 존재하지 않을 때는 IllegalArgumentException을 반환해야 합니다.");

    assertEquals(expectedExceptionMessage, thrown.getMessage());
}
```

테스트 주도 개발(TDD)

05

Mockito

Test Double

Mock, Fake, Spy, Stub, Dummy ...

Mockito

```
public User createUser ( UserInfo userInfo, DAO mySqlDao ){
```

```
    // 코드
```

```
    String userId = mySqlDao.save(user);
```

```
    // 코드
```

```
}
```


Mockito

MVN REPOSITORY

Repository

- Central 2.7k
- Sonatype 744
- Spring Plugins 369
- Spring Lib M 333
- JCenter 155
- IBiblio 85
- Spring Lib Relea... 71
- Geomajas 64

Group





- com.github 331
- io.github 129
- org.apache 97
- org.eclipse 87
- com.javiersc 64
- org.powermock 59
- org.junit 45
- org.testifyproject 34

Category

- Maven Archetype 58
- Android Package 42

Found 3228 results

Sort: **relevance** | popular | newest

- **1. Mockito JUnit Jupiter**
[org.mockito » mockito-junit-jupiter](#)
Mockito JUnit 5 support
Last Release on Jun 18, 2023
- **2. JUnit**
[junit » junit](#)
JUnit is a unit testing framework to write and run repeatable automated tests on Java.
Last Release on Feb 13, 2021
- **3. Mockito Core**
[org.mockito » mockito-core](#)
Mockito mock objects library core API and implementation
Last Release on Jun 18, 2023
- **4. JUnit Jupiter API**
[org.junit.jupiter » junit-jupiter-api](#)
JUnit Jupiter is the API for writing tests using JUnit 5.
Last Release on Jul 6, 2023

Mockito



Mockito JUnit Jupiter » 5.4.0

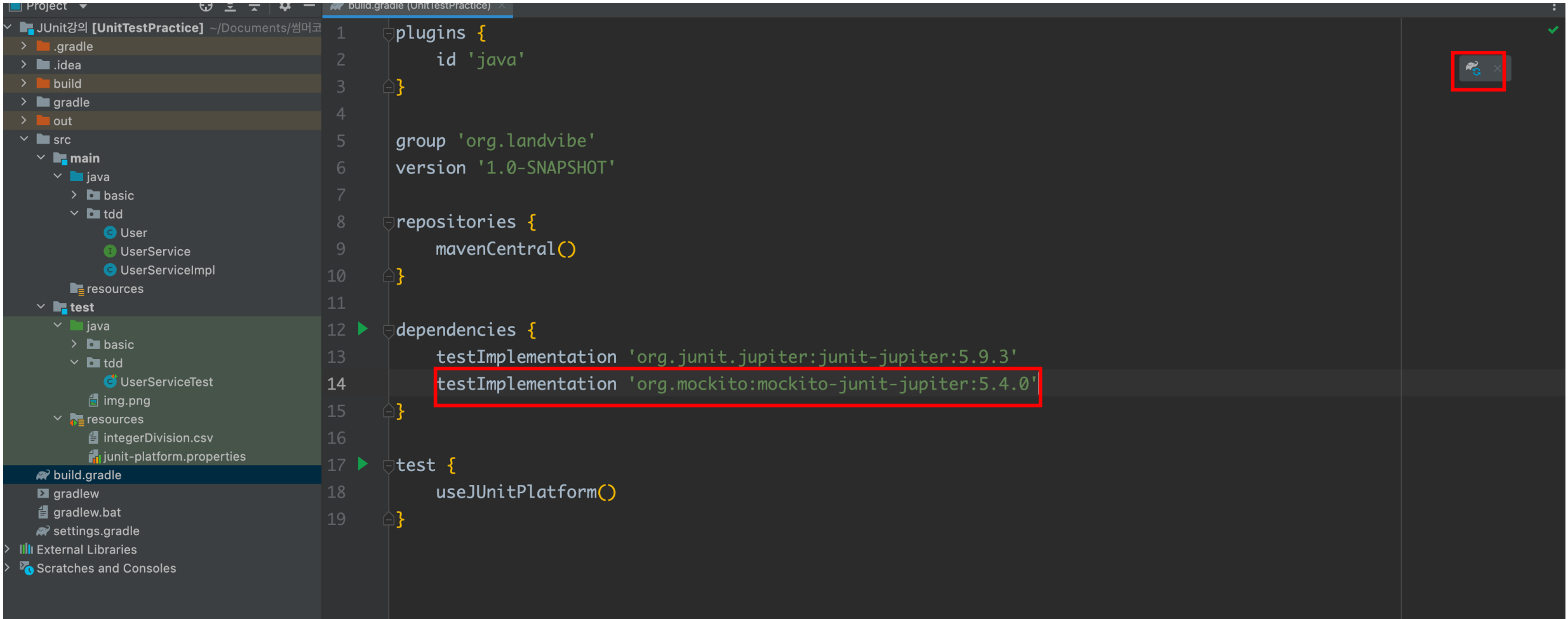
Mockito JUnit 5 support

License	MIT
Categories	Mocking
Tags	mock mocking junit testing mockito
HomePage	https://github.com/mockito/mockito
Date	Jun 18, 2023
Files	pom (2 KB) jar (5 KB) View All
Repositories	Central
Ranking	#141 in MvnRepository (See Top Artifacts) #5 in Mocking
Used By	3,237 artifacts

Maven Gradle **Gradle (Short)** Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
// https://mvnrepository.com/artifact/org.mockito/mockito-junit-jupiter
testImplementation 'org.mockito:mockito-junit-jupiter:5.4.0'
```

Mockito



The screenshot shows an IDE interface with a project explorer on the left and a code editor in the center. The project explorer shows a project named 'JUnit강의 [UnitTestPractice]' with a 'test' directory containing 'UserServiceTest' and 'junit-platform.properties'. The code editor displays the 'build.gradle' file for the 'UnitTestPractice' project. The dependencies section is highlighted with a red box, showing the following configuration:

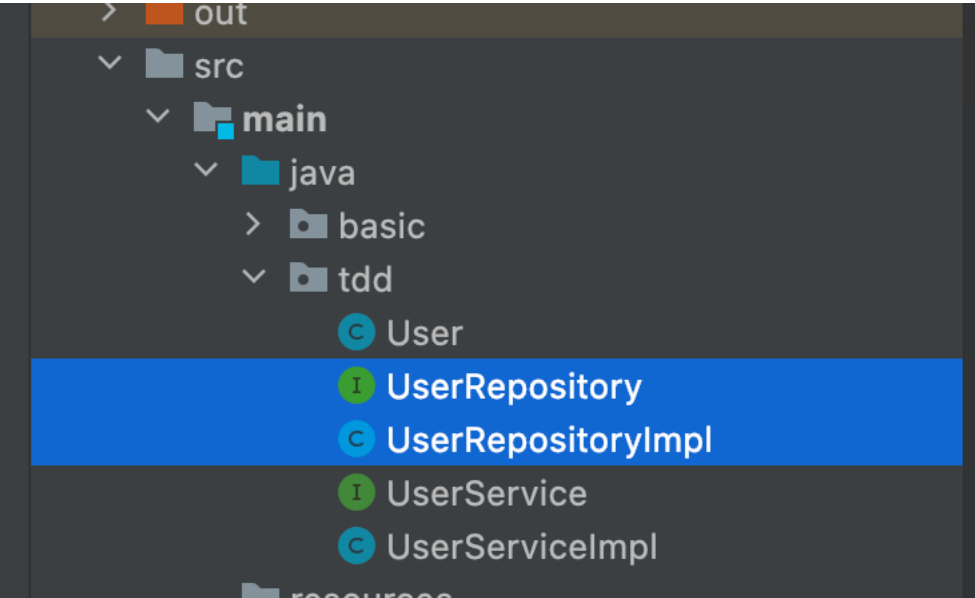
```
1 plugins {  
2     id 'java'  
3 }  
4  
5 group 'org.landvibe'  
6 version '1.0-SNAPSHOT'  
7  
8 repositories {  
9     mavenCentral()  
10 }  
11  
12 dependencies {  
13     testImplementation 'org.junit.jupiter:junit-jupiter:5.9.3'  
14     testImplementation 'org.mockito:mockito-junit-jupiter:5.4.0'  
15 }  
16  
17 test {  
18     useJUnitPlatform()  
19 }
```

In the top right corner of the IDE, a small floating window with a refresh icon is highlighted with a red box.

Mockito

```
public interface UserRepository {  
  
    boolean save(User user);  
  
}
```

```
public class UserRepositoryImpl implements UserRepository {  
  
    Map<String, User> users = new HashMap<>();  
  
    @Override  
    public boolean save(User user) {  
  
        boolean returnValue = false;  
  
        if (!users.containsKey(user.getId())) {  
            users.put(user.getId(), user);  
            returnValue = true;  
        }  
  
        return returnValue;  
    }  
}
```

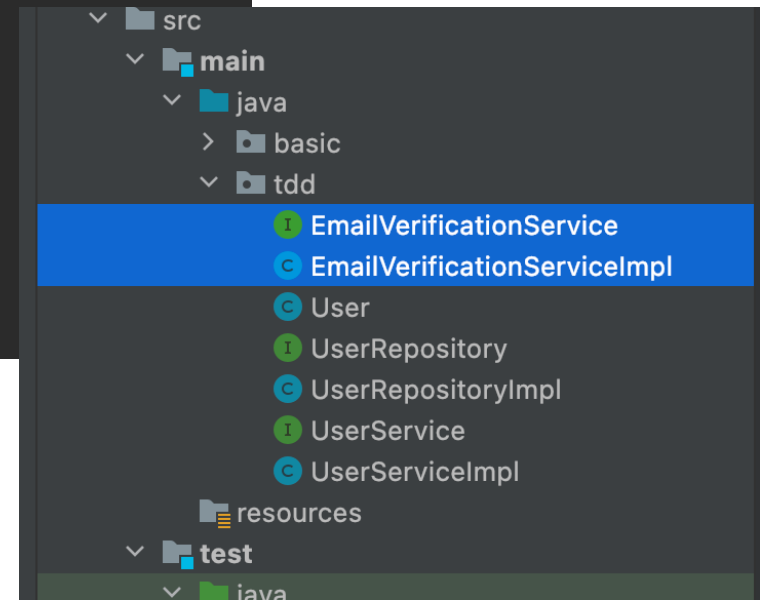


```
> out  
v src  
  v main  
    v java  
      > basic  
      v tdd  
        User  
        UserRepository  
        UserRepositoryImpl  
        UserService  
        UserServiceImpl
```

Mockito

```
public interface EmailVerificationService {  
  
    void scheduleEmailConfirmation(User user);  
}
```

```
public class EmailVerificationServiceImpl implements EmailVerificationService {  
  
    @Override  
    public void scheduleEmailConfirmation(User user) {  
        System.out.println("scheduleEmailConfirmation is called");  
    }  
}
```



Mockito

```
@ExtendWith(MockitoExtension.class)
public class MockitoTest {

    UserService userService = new UserServiceImpl();
```

Mockito

1 related problem

```
public class UserServiceImpl implements UserService {
```

```
    private UserRepository userRepository;
```

```
    private EmailVerificationService emailVerificationService;
```

1 related problem

```
public UserServiceImpl(UserRepository userRepository,
    EmailVerificationService emailVerificationService) {
    this.userRepository = userRepository;
    this.emailVerificationService = emailVerificationService;
}
```

```
@ExtendWith(MockitoExtension.class)
```

```
public class UserServiceTest {
```

```
    @InjectMocks
```

```
    UserServiceImpl userService;
```

```
    @Mock
```

```
    EmailVerificationServiceImpl emailVerificationService;
```

```
    @Mock
```

```
    UserRepositoryImpl userRepository;
```

Mockito

The image shows an IDE interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with the following folders and files:

- JUnit강의 [Unit testPractice] ~/Documents/샘머코
 - .gradle
 - .idea
 - build
 - gradle
 - out
 - src
 - main
 - java
 - basic
 - tdd
 - EmailVerificationService
 - EmailVerificationServiceImpl
 - User
 - UserRepository
 - UserRepositoryImpl
 - UserService
 - UserServiceException**
 - UserServiceImpl
 - resources

The code editor shows the following code:

```
1 package tdd;
2
3 public class UserServiceException extends RuntimeException{
4
5     public UserServiceException(String message) {
6         super(message);
7     }
8 }
9
```


Mockito

의 [UnitTestPractice] ~/Documents/샘머코
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

- main
 - java
 - basic
 - tdd
 - EmailVerificationService
 - EmailVerificationServiceImpl
 - User
 - UserRepository
 - UserRepositoryImpl
 - UserService
 - UserServiceException
 - UserServiceImpl
- resources
- test
 - java
 - basic
 - tdd
 - UserServiceTest
- resources
 - integerDivision.csv
 - junit-platform.properties
- build.gradle
- build
- build.bat
- build.gradle
- External Libraries
- Run and Consoles

```
@Override
public User createUser(String name, String email, String password) {

    if (name == null || name.trim().length() == 0) {
        throw new IllegalArgumentException("이름이 존재하지 않습니다.");
    }

    String id = UUID.randomUUID().toString();
    User user = new User(id, name, email, password);
    boolean isUserCreated = false;

    try {
        isUserCreated = userRepository.save(user);
    } catch (RuntimeException ex) {
        throw new UserServiceException(ex.getMessage());
    }

    if (!isUserCreated) {
        throw new UserServiceException("사용자 생성 실패");
    }

    try {
        emailVerificationService.scheduleEmailConfirmation(user);
    } catch (RuntimeException ex) {
        throw new UserServiceException(ex.getMessage());
    }

    return user;
}
```

Mockito

```
@DisplayName("사용자 생성")
@Test
public void createUser_whenUserDetailsProvided_returnUserObject() {
    when(userRepository.save(any(User.class))).thenReturn(true);

    User user = userService.createUser(name, email, password);

    assertNotNull(user, message: "회원가입한 사용자는 null이면 안됩니다.");
    assertEquals(name, user.getName());
    assertEquals(email, user.getEmail());
    assertNotNull(user.getId(), message: "사용자 id가 존재하지 않습니다.");
    verify(userRepository)
        .save(any(User.class));
    verify(userRepository, Mockito.times(wantedNumberOfInvocations: 1))
        .save(any(User.class));
}
```

Mockito



```
1 package tdd;
2
3 public class EmailNotificationServiceException extends RuntimeException{
4
5     public EmailNotificationServiceException(String message) {
6         super(message);
7     }
8 }
9
```

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a 'main' directory with 'java', 'basic', and 'tdd' sub-directories. The 'tdd' directory contains several classes, with 'EmailNotificationServiceException' selected. The code editor displays the following Java code, which is highlighted with a red box:

Mockito

```
@DisplayName("사용자 생성시 이메일 예외 발생")
@Test
void createUser_whenEmailNotificationExceptionThrown_throwsUserServiceException() {
    // Arrange
    when(userRepository.save(any(User.class))).thenReturn(true);

    doThrow(EmailNotificationServiceException.class) Stubber
        .when(emailVerificationService) EmailVerificationServiceImpl
        .scheduleEmailConfirmation(any(User.class));

    // Act & Assert
    Assertions.assertThrows(UserServiceException.class, () -> {
        userService.createUser(name, email, password);
    }, message: "UserServiceException이 발생해야 합니다.");

    // Assert
    verify(emailVerificationService, times(wantedNumberOfInvocations: 1)).
        scheduleEmailConfirmation(any(User.class));
}
```

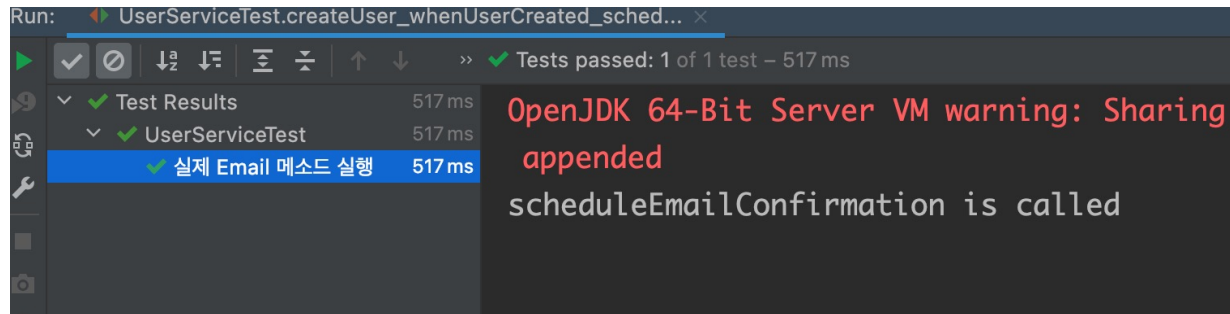
Mockito

```
@DisplayName("실제 Email 메소드 실행")
@Test
void createUser_whenUserCreated_schedulesEmailConfirmation() {
    // Arrange
    when(userRepository.save(any(User.class))).thenReturn(true);

    doCallRealMethod().when(emailVerificationService).scheduleEmailConfirmation(any(User.class));

    // Act
    userService.createUser(name, email, password);

    // Assert
    verify(emailVerificationService, times(wantedNumberOfInvocations: 1))
        .scheduleEmailConfirmation(any(User.class));
}
```



Run: UserServiceTest.createUser_whenUserCreated_sched... x

Tests passed: 1 of 1 test - 517 ms

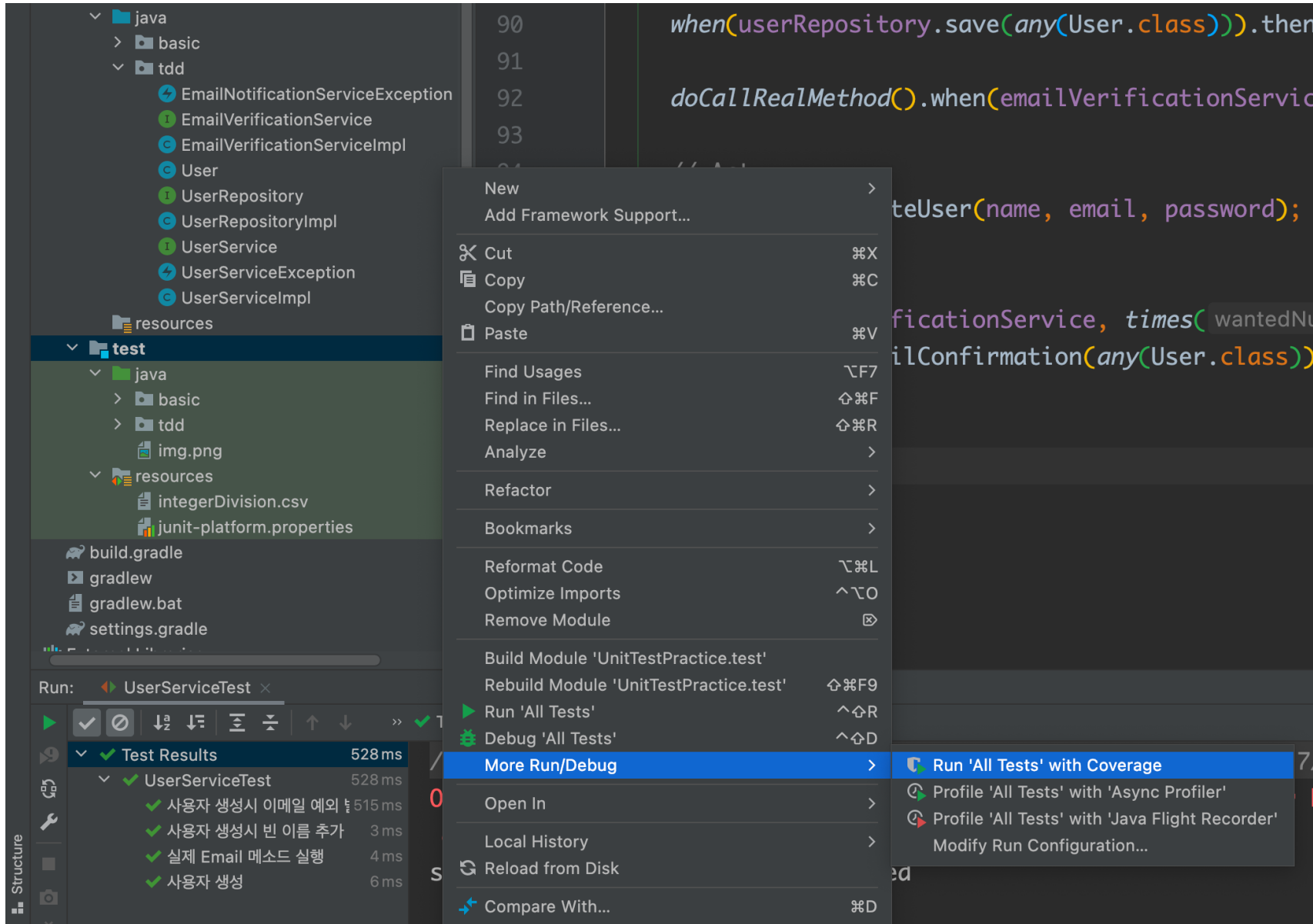
Test Results	517 ms
UserServiceTest	517 ms
실제 Email 메소드 실행	517 ms

OpenJDK 64-Bit Server VM warning: Sharing appended
scheduleEmailConfirmation is called

06

테스트 커버리지와 Jacoco

테스트 커버리지와 Jacoco



테스트 커버리지와 Jacoco

- src
 - main
 - java 71% classes, 76% lines covered
 - basic 100% classes, 100% lines covered
 - Calculator 100% methods, 100% lines covered
 - tdd 66% classes, 74% lines covered
 - EmailNotificationServiceException 0% methods, 0% lines covered
 - EmailVerificationService
 - EmailVerificationServiceImpl 100% methods, 100% lines covered
 - User 100% methods, 100% lines covered
 - UserRepository
 - UserRepositoryImpl 0% methods, 0% lines covered
 - UserService
 - UserServiceException 100% methods, 100% lines covered
 - UserServiceImpl 100% methods, 84% lines covered
 - resources

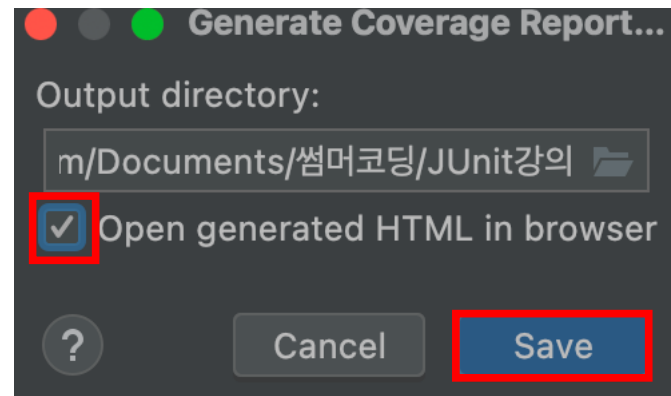
66% classes, 74% lines covered in package 'tdd'

Element	Class, %	Method, %	Line, %
EmailNotificationServiceException	0% (0/1)	0% (0/1)	0% (0/1)
EmailVerificationService	100% (0/0)	100% (0/0)	100% (0/0)
EmailVerificationServiceImpl	100% (1/1)	100% (1/1)	100% (1/1)
User	100% (1/1)	100% (4/4)	100% (8/8)
UserRepository	100% (0/0)	100% (0/0)	100% (0/0)
UserRepositoryImpl	0% (0/1)	0% (0/1)	0% (0/5)
UserService	100% (0/0)	100% (0/0)	100% (0/0)
UserServiceException	100% (1/1)	100% (1/1)	100% (1/1)
UserServiceImpl	100% (1/1)	100% (2/2)	84% (16/19)

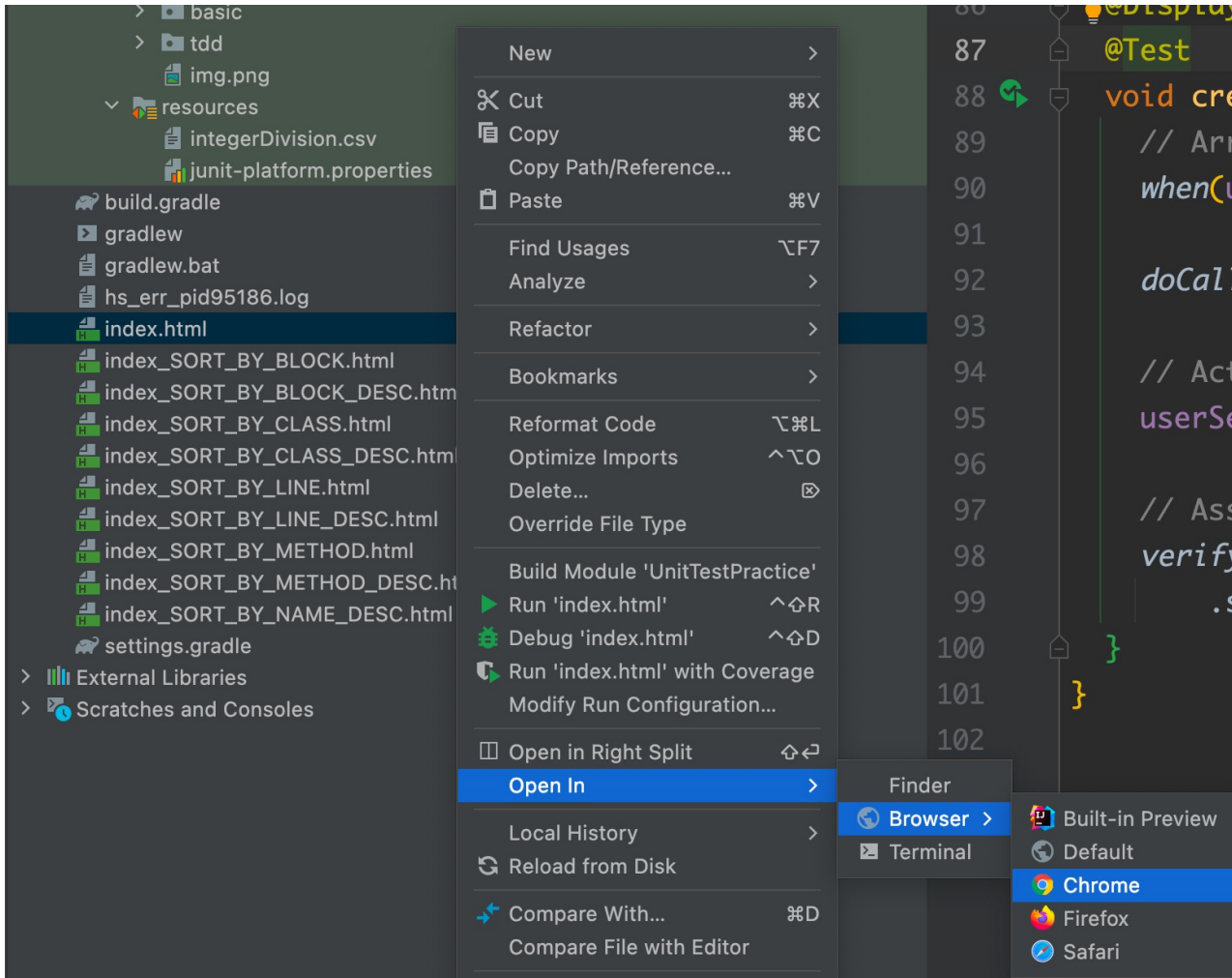
테스트 커버리지와 Jacoco

71% classes, 76% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
apple			
basic	100% (1/1)	100% (2/2)	100% (3/3)
com			
images			
java			
Generate Coverage Report...			
jdk			
META-INF			
net			
netscape			
org			
sun			
tdd	66% (4/6)	80% (8/10)	74% (26/35)
toolbarButtonGraphics			
win32-x86			
win32-x86-64			



테스트 커버리지와 Jacoco



Overall Coverage Summary			
Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	73.3% (11/15)	70.7% (29/41)
Coverage Breakdown			
Package	Class, %	Method, %	Line, %
basic	100% (1/1)	100% (3/3)	100% (3/3)
tdd	66.7% (4/6)	66.7% (8/12)	68.4% (26/38)

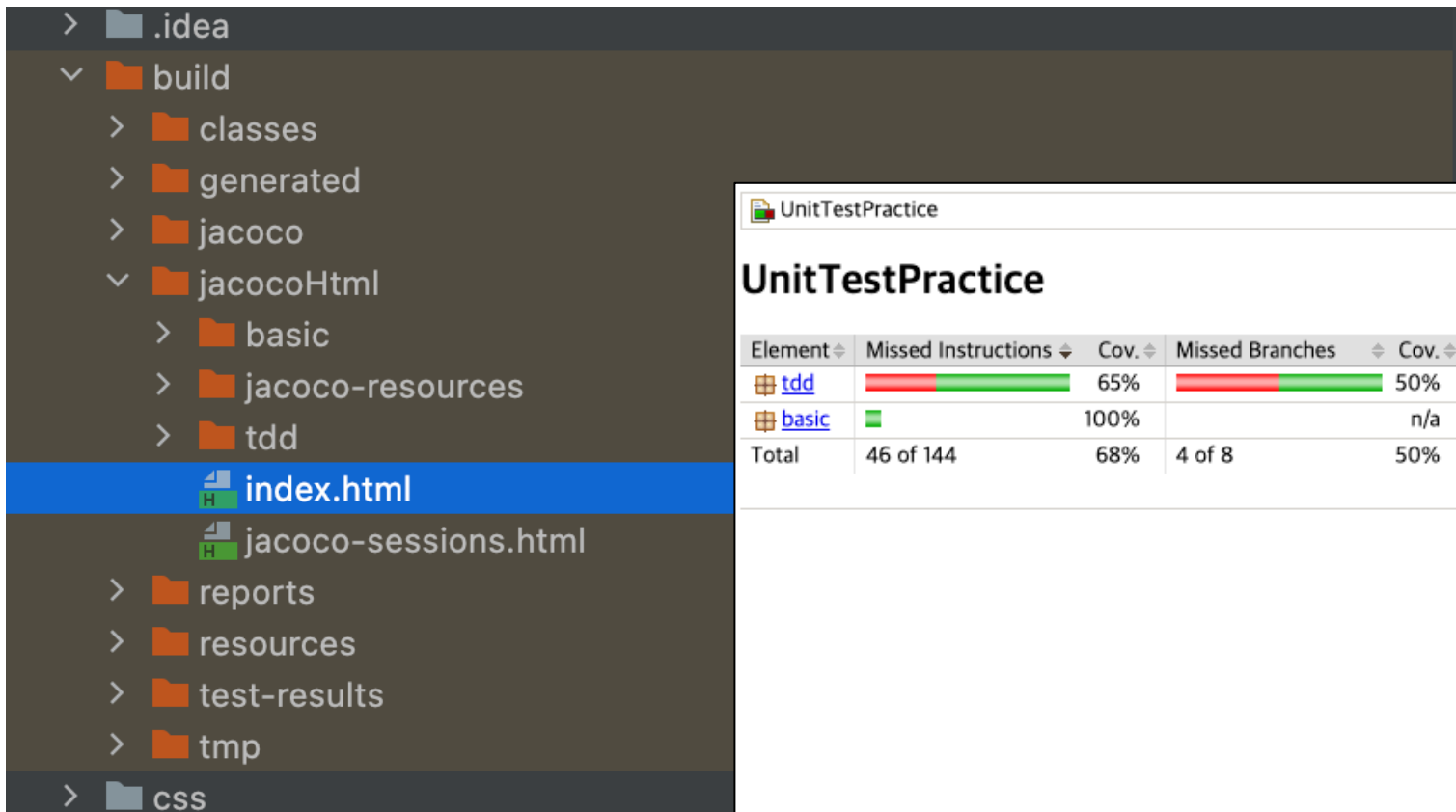
generated on 2023-07-11 10:11

테스트 커버리지와 Jacoco

```
plugins {  
    id 'java'  
    id 'jacoco'  
}  
  
jacocoTestReport {  
    reports {  
        html.enabled true  
        xml.enabled false  
        csv.enabled false  
        html.destination file("${buildDir}/jacocoHtml")  
    }  
}
```

```
JUnit7 | 100 - ./gradlew test jacocoTestReport
```

테스트 커버리지와 Jacoco



UnitTestPractice

UnitTestPractice

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
tdd		65%		50%	7	16	13	43	4	12	2	6
basic		100%		n/a	0	3	0	3	0	3	0	1
Total	46 of 144	68%	4 of 8	50%	7	19	13	46	4	15	2	7