

# 1강 JUNIT



# 개요

# Test

## ❖ 이해하기

- 응용 프로그램의 기능을 확인하여 요구 사항에 따라 실행되는지 확인하는 프로세스
- 단위 테스트는 클래스 또는 메소드 단위로 이루어 지는 프로세스

## ❖ Junit

- Java 프로그래밍 언어에서 사용되는 단위 테스트 프레임워크

## ❖ 특징

- 오픈 소스 프레임 워크이며 테스트 작성 및 실행에 사용됨
- 테스트 방법을 식별하기위한 주석을 제공
- 예상 결과를 테스트하기위한 어설션을 제공
- 테스트 실행을 위한 테스트 러너를 제공
- 테스트를 통해 코드를 더 빨리 작성할 수 있으므로 품질이 향상

# 환경설정

# Junit 생성

▼ Test

> JRE System Library [JavaSE-13]

src

New

Open in New Window

Open Type Hierarchy

F4

Show In

Alt+Shift+W >



Copy

Ctrl+C



Copy Qualified Name



Paste

Ctrl+V



Delete

Delete



Remove from Context

Ctrl+Alt+Shift+Down

Build Path



Source

Alt+Shift+S >

Refactor

Alt+Shift+T >



Import...



Java Project



Project...



Package



Class



Interface



Enum



Annotation



Source Folder



Java Working Set



Folder



File



Untitled Text File



Task




JUnit Test Case

# JAVA 설치

## New JUnit Test Case

### JUnit Test Case

 The use of the default package is discouraged.

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit JUnit5 test

Source folder:

Package:

Name:


Superclass:

## New JUnit Test Case



JUnit 5 is not on the build path. Do you want to add it?

- ☐ Not now  
☐ Open the build path property page  
☒ Perform the following action:

 Add JUnit 5 library to the build path

OK

Cancel

## 기본 코드 확인

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class JUnitTest {
    @Test
    void testAdd() {
        String str = "Hello JUnit";
        assertEquals("Hello JUnit", str);
    }
}
```

# 실행 및 결과 확인

tmp - Test/src/JUnitTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Test
  - JRE System Library [Java]
  - src
    - (default package)
      - JUnitTest.java
    - JUnit 5

Run As > JUnit 1 JUnit Test Alt+Shift+X, T

Run Configurations...

Organize Favorites...

Finished after 0.183 seconds

Runs: 1/1 Errors: 0 Failures: 0

> JUnitTest [Runner: JUnit 5] (0.000 s)



# Junit 특징

# JUnit 테스트 프레임워크의 특징

- ❖ Fixtures
  - 테스트 실행 전 또는 실행 후 처리할 내용
- ❖ Test suites
  - 여러 클래스를 하나로 묶어 테스트
- ❖ Test runners
  - 단위 테스트를 하기 위한 메인(이클립스 자체 처리)
- ❖ JUnit classes
  - JUnit에서 지원해 주는 클래스
  - Assert - 어설트 메소드 지원(AssertEquals, AssertTrue 등)
  - TestCase - 여러 클래스를 묶어 처리할 경우 사용
  - TestResult- 테스트 케이스 실행 결과를 수집하는 메소드

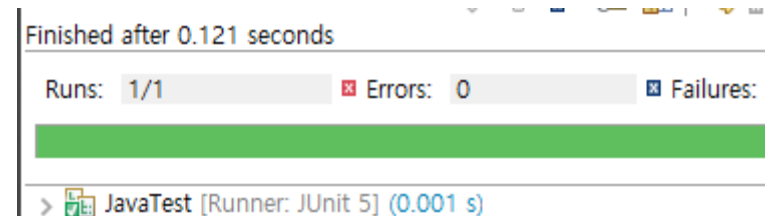
# Fixtures

## ❖ 이해하기

- 테스트를 위해 객체의 초기화와 마무리 정리를 위한 기능 지원
- setUp() : 초기값 설정, DB 연결 등 초기화 처리
- tearDown() : DB 연결 종료와 같이 마무리 정리를 위해 사용

## ❖ 사용 예

```
public class JavaTest extends TestCase{  
    protected int n1, n2;  
    @Override  
    protected void setUp() {  
        n1 = 2; n2 = 3;  
    }  
    public void testAdd() {  
        assertTrue(5==n1+n2);  
    }  
}
```



# Test suites

- ❖ 이해하기
  - 메소드 활용
  - 어노테이션 활용
- ❖ 사용 예

```
public static Test suite() {  
    TestSuite suite = new TestSuite();  
    suite.addTestSuite(TestJUnit1.class);  
    suite.addTestSuite(TestJUnit2.class);  
    return suite;  
}
```

```
@RunWith(Suite.class)  
@SuiteClasses({TestJUnit1.class, TestJUnit2.class})
```

# 실습

# 실습

- ❖ `Java.Junit01.FeaturesEx01.Fixtures`
- ❖ `Java.Junit01.FeaturesEx02.Suite`
- ❖ `Java.Junit01.FeaturesEx03.Runner`

# Junit classes

# Assert

## ❖ 이해하기

- Test 결과를 확인

## ❖ API

- `void assertEquals(boolean expected, boolean actual)`
  - 두 인자의 같음을 비교
- `Void assertFalse(Boolean condition)`
  - 인자의 결과가 거짓인지 비교
- `void assertTrue(boolean condition)`
  - 인자의 결과가 참인지 비교
- `Void assertNotNull(Object object)`
  - 인자의 결과가 null이 아닌지 비교
- `void assertNull(Object object)`
  - 인자의 결과가 null인지 비교
- `void fail()`
  - 예외가 잘 동작되는지 확인하기 위해 강제적 예외 발생



# 실습

- ❖ Java.Junit02.Assert01
- ❖ Java.Junit02.Assert02.fail

# TestCase

## ❖ 이해하기

- 원활한 테스트를 위해 여러 메소드 지원

## ❖ 지원 메소드

- `int countTestCases ()`
  - `run (TestResult result)`에 의해 실행 된 테스트 케이스 수를 계산합니다.
- `String getName ()`
  - TestCase의 이름을 가져옵니다.
- `void setName (String name)`
  - TestCase의 이름을 설정합니다.
- `void setUp ()`
  - 테스트 초기 데이터 설정
- `void tearDown ()`
  - 테스트 종료 시 정리 설정

# 실습

❖ Java.Junit02.Assert03.TestCase

# Quiz

# Quiz

## ❖ Employee

- name(String), age(int), monthlySalary(double)

## ❖ EmpBusinessLogic

- calculateYearlySalary : 연간 판매 수익 반환
- calculateAppraisal
  - 월 판매 수익이 10000 미만일 경우 500 point
  - 월 판매 수익이 10000 이상일 경우 1000 point

## ❖ Quiz

- “25살의 jin 사원이 이번 달 8000만원의 실적을 올렸다”는 가정하에 EmpBusinessLogic의 두 메소드가 정상적으로 동작하는지 테스트 하시오

❖ Java.Junit03.Quiz

# 어노테이션

# 어노테이션

## ❖ 이해하기

- 메소드에 대한 처리 방법 지정

## ❖ 종류

@Test	Method 테스트 시 사용
@Before	Method 실행 전 처리, 메소드 마다 실행
@After	Method 실행 후 처리, 메소드 마다 실행
@BeforeClass	Class 실행 전 처리, 최초 1회 실행
@AfterClass	Class 실행 후 처리, 최초 1회 실행
@Ignore	Method 무시



# Quiz

# Quiz

## ❖ Employee

- name(String), age(int), monthlySalary(double)

## ❖ EmpBusinessLogic

- calculateYearlySalary : 연간 판매 수익 반환
- calculateAppraisal
  - 월 판매 수익이 10000 미만일 경우 500 point
  - 월 판매 수익이 10000 이상일 경우 1000 point

## ❖ Quiz

- “25살의 jin 사원이 이번 달 8000만원의 실적을 올렸다”는 가정하에 EmpBusinessLogic의 두 메소드가 정상적으로 동작하는지 테스트 하시오



어노테이션 적용

## 참조 사이트

- ❖ [https://www.tutorialspoint.com/junit/junit\\_basic\\_usage.htm](https://www.tutorialspoint.com/junit/junit_basic_usage.htm)
- ❖ <https://coding-start.tistory.com/259?category=814944>