

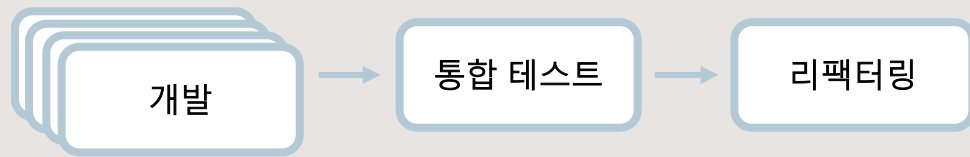


# TestNG를 활용한 TTD 가이드

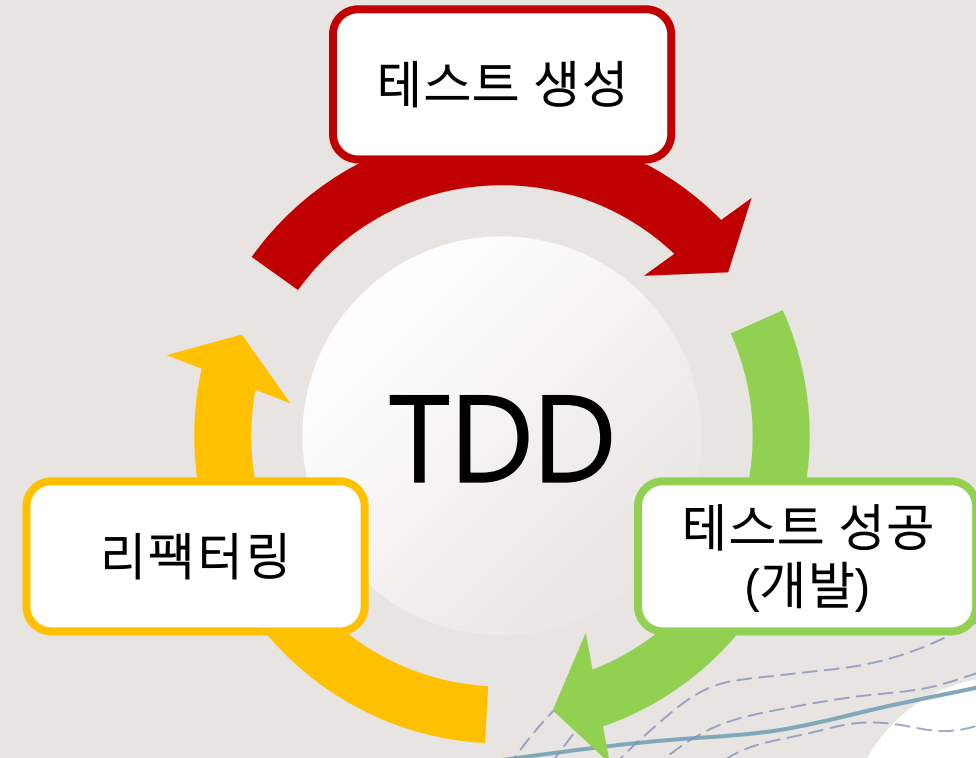
by. 김우빈

# WHAT: 테스트 주도 개발 (TDD)

TDD(Test Driven Development, 테스트 주도 개발)는 짧은 개발 주기의 반복에 의존하는 개발 프로세스로서, 개발을 우선하고 테스트를 하는 전형적인 개발 프로세스와 반대로 **테스트를 우선하여 개발을 하는 개발 프로세스**입니다.



일반 개발 프로세스



# WHY: 테스트 주도 개발 (TDD)

## 1. 객체지향적인 코드 개발

테스트 코드를 먼저 작성한다면 미리 함수의 기능과 입출력 구조를 정의하기 때문에 좀 더 명확한 기능과 구조를 설계할 수 있습니다.  
일반 개발 프로세스에서는 복잡한 기능을 구현하기 위해 여러 함수가 섞여 각 함수의 기능이 불분명해지며(개발자만 알게되며) 재사용성이 떨어집니다.  
이와 달리 TDD는 각 함수의 정의를 명확하게 하기 때문에 각 함수의 재사용성이 높아짐으로, 이를 객체지향적이라 볼 수 있습니다.

## 2. 설계 수정 시간의 단축

TDD를 함으로서 테스트 코드를 먼저 작성하기 때문에 최초 설계안을 만족시키며,  
입출력 구조와 기능의 정의를 명확히 하게 되므로 설계의 구조적 문제를 바로 찾아내게 됩니다.  
개발을 하는 도중 설계의 결함을 찾아 설계 수정은 물론 이미 개발했던 코드마저 수정해야 하는 경우를 방지함으로,  
설계 수정, 코드 수정의 시간을 단축하게 됩니다 (따라서 개발 비용도 적게 듭니다).

## 3. 디버깅 시간의 단축

기본적으로 단위 테스트 기반의 테스트 코드를 작성하기 때문에  
추후 문제가 발생하였을 시 각각의 모듈 별로 테스트를 진행해보면 문제의 지점을 쉽게 찾아낼 수 있습니다.

## 4. 유지보수/협력의 용이성

함수의 기능과 입출력 구조를 확인할 수 있기 때문에 재사용할 수 있는 함수를 쉽게 찾을 수 있으며, 똑같은 함수를 다시 개발하는 경우를 줄여줍니다.  
코드 추가/수정한 부분이 다른 기능에 영향을 주었는지 테스트를 통해 쉽게 식별할 수 있습니다.  
Jenkins와 같은 도구를 사용하여 코드 배포 후 테스트를 자동화 할 수 있습니다.

# Java 코드 테스트 도구

Java 코드 테스트 도구는 여러 개 있습니다.

- + Jbehave
- + JUnit
- + Serenity
- + TestNG
- + Selenide
- + Gauge
- + Geb
- + Spock
- + HttpUnit
- + JWebUnit

JUnit과 TestNG가 일반적으로 많이 사용되는 것으로 알려져 있으며, 해당 가이드는 TestNG를 위주로 설명 드립니다.

각 도구의 장단점은 <https://www.lambdatest.com/blog/top-10-java-testing-frameworks/> 에서 확인해 볼 수 있습니다.

# TestNG 설치

## 1. 자바 설치

대부분 버전과 호환되지만 일단 1.8 버전을 기준으로 합니다.  
JAVA\_HOME 환경변수가 제대로 설정되어 있는지 확인해야 됩니다.

## 2. 메이븐 설치

대부분 버전과 호환되지만 일단 3.2.3 버전을 기준으로 합니다.

## 3. 에클립스 설치

대부분 버전과 호환되지만 일단 Eclipse 2019-03-R 패키지 기준으로 합니다.

## 4. TestNG 플러그인 설치

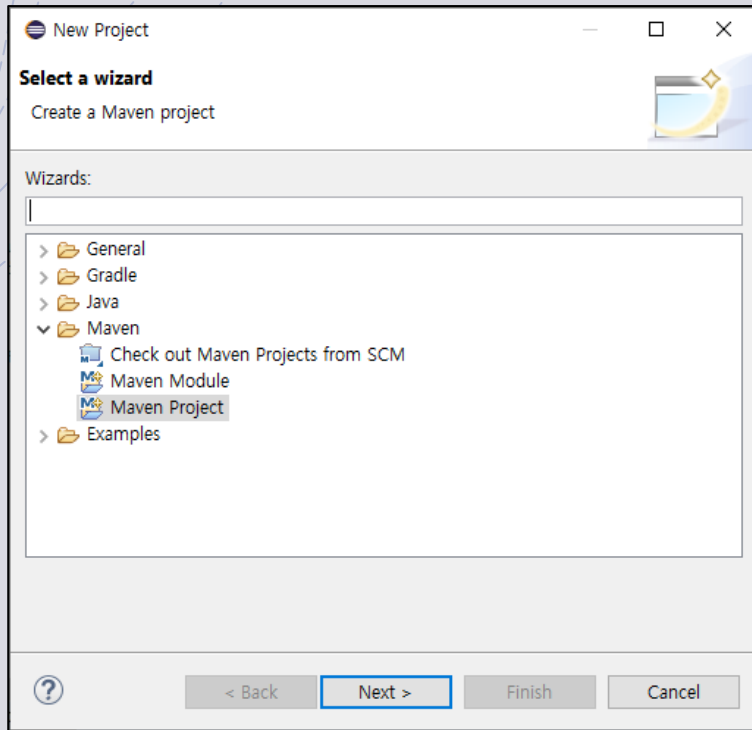
에클립스 > Help > Install New Software > 리포지토리 추가 (work with 섹션에 <https://testng.org/testng-eclipse-update-site> 입력하고 add 누르고 TestNG 체크하고 Next) > Next > Finish > Install anyway > Restart

## 5. (프로젝트 생성 후) 메이븐 설정 pom.xml에 TestNG 추가

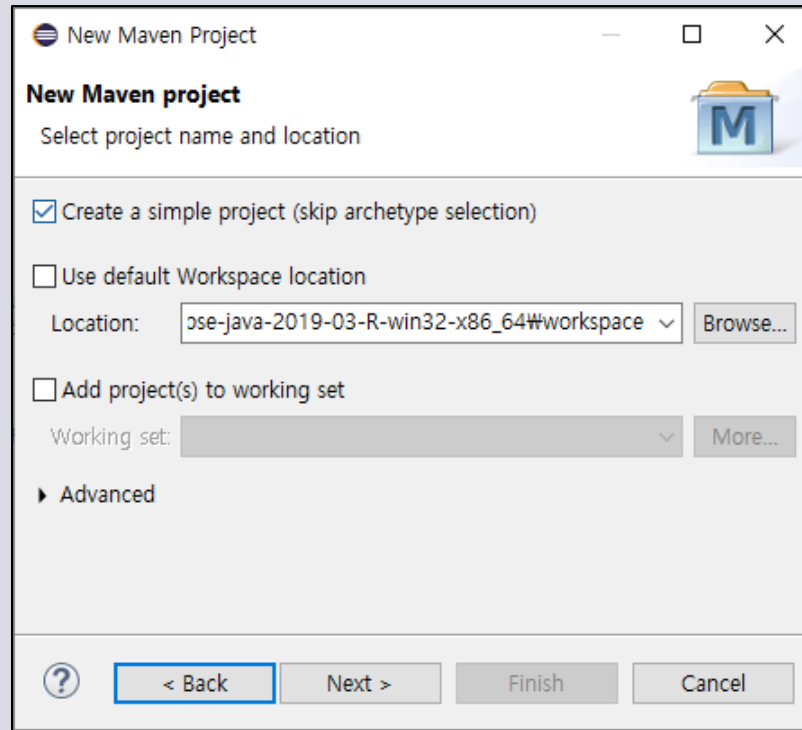
# TestNG 사용 방법 / TDD 프로세스

1. 프로젝트 생성
2. maven 설정에 TestNG dependency 추가
3. TestNG 클래스 생성
4. 테스트 코드 작성
5. 메인 코드 작성
6. TestNG 실행
7. 4~6번 반복

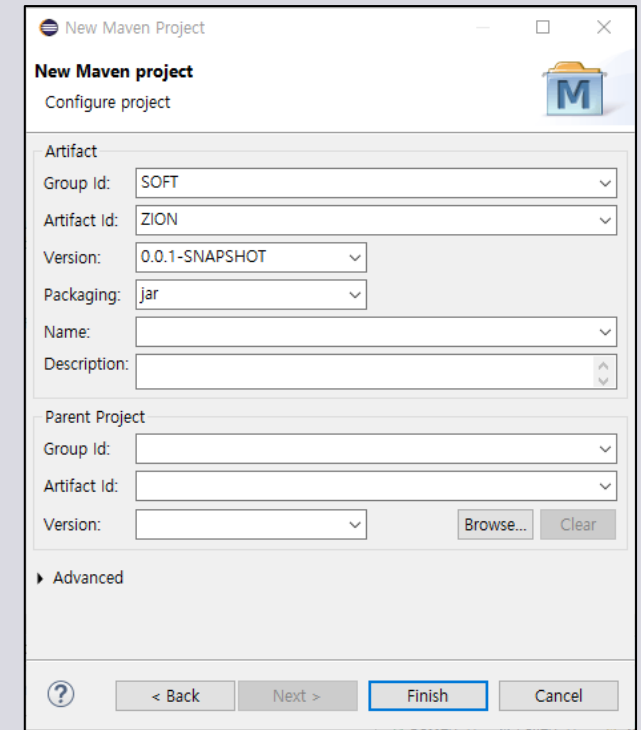
# 1. 프로젝트 생성



1. File > New Project > Maven Project



2. simple project 선택



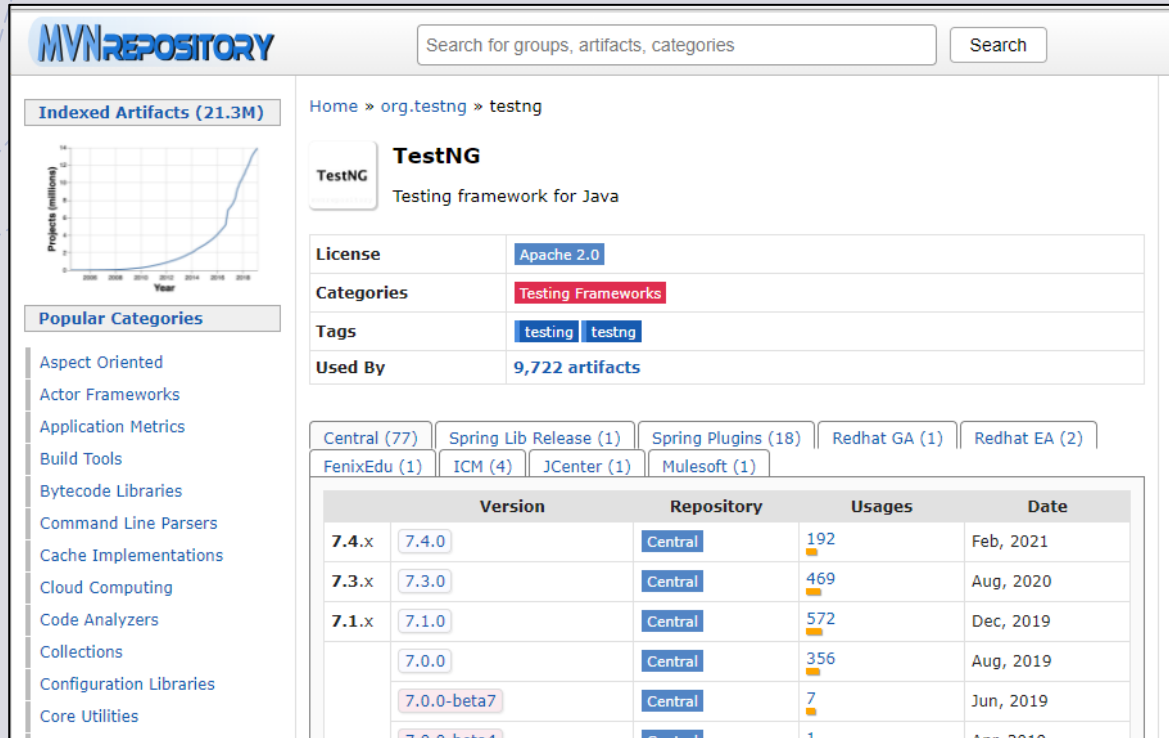
3. Group, Artifact ID 입력 후 finish

## 참고

- 가이드를 위해 아주 간단한 maven simple project를 생성합니다.
- JRE System Library 버전 에러가 나오는 경우, 파일 목록 > JRE System Library 우클릭 > 1.8로 변경



## 2. maven 설정에 TestNG dependency 추가



The screenshot shows the Maven Repository website for the TestNG artifact. The page includes a search bar, a graph of indexed artifacts, and a list of popular categories. The main content area displays the TestNG artifact details, including its license (Apache 2.0), categories (Testing Frameworks), tags (testing, testng), and a table of versions and their usages.

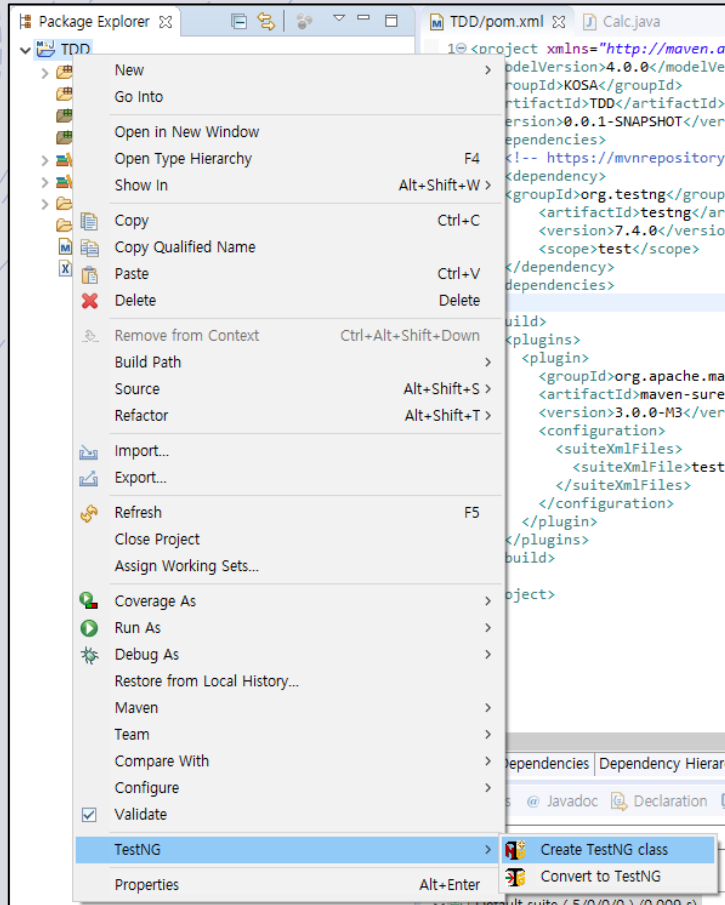
Version	Repository	Usages	Date	
7.4.x	7.4.0	Central	192	Feb, 2021
7.3.x	7.3.0	Central	469	Aug, 2020
7.1.x	7.1.0	Central	572	Dec, 2019
	7.0.0	Central	356	Aug, 2019
	7.0.0-beta7	Central	7	Jun, 2019
	7.0.0-beta4	Central	1	Apr, 2019

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>KOSA</groupId>
4   <artifactId>TDD</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <dependencies>
7     <!-- https://mvnrepository.com/artifact/org.testng/testng -->
8     <dependency>
9       <groupId>org.testng</groupId>
10      <artifactId>testng</artifactId>
11      <version>7.4.0</version>
12      <scope>test</scope>
13    </dependency>
14  </dependencies>
```

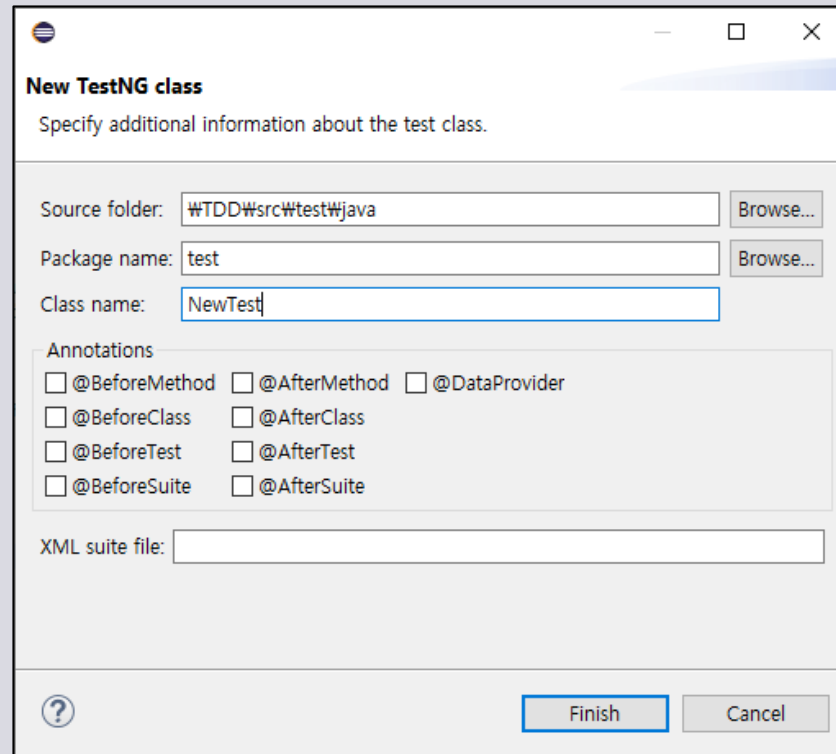
<https://mvnrepository.com> 에서 TestNG 검색하여 7.4.0 버전 클릭, 메이븐 dependency 코드 복사해서 pom.xml 에 dependencies 태그 내에 붙여주세요.



# 3. TestNG 클래스 생성



1. 프로젝트 폴더 우클릭 > TestNG > Create TestNG class



2. package, class name 입력

```
1 package test;
2
3 import org.testng.annotations.Test;
4
5 public class NewTest {
6     @Test
7     public void f() {
8     }
9 }
```

3. 생성된 클래스 확인

## 4. 테스트 케이스 작성

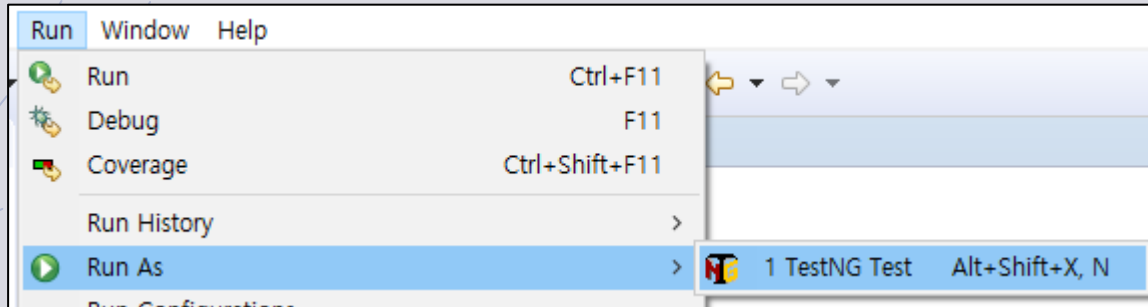
```
1 package test;
2
3 import static org.testng.Assert.assertEquals;
4
5 import org.testng.annotations.Test;
6
7 public class NewTest {
8     @Test
9     public void ng01_add() {
10         Calc calc = new Calc();
11         assertEquals(calc.add(2, 3), 5);
12     }
13
14     @Test
15     public void ng02_add() {
16         Calc calc = new Calc();
17         assertEquals(calc.add(85, 33), 118);
18     }
19 }
```

계산기의 덧셈 기능을 개발한다는 가정하에 테스트 케이스를 작성합니다.

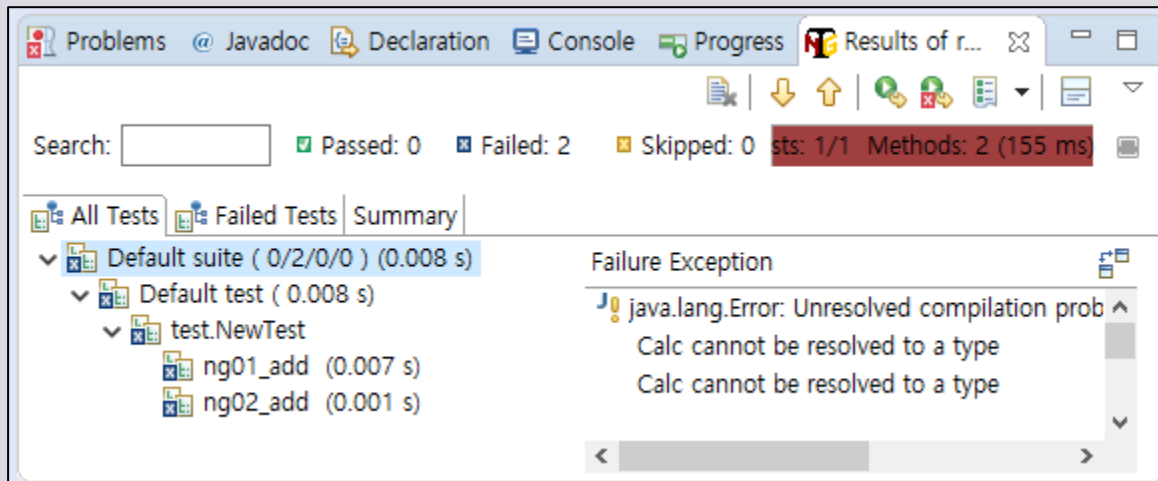
**Calc** 클래스에 **add** 함수를 미리 설계하며, 입출력 구조를 정의합니다 (당연히 메인 코드에는 **Calc** 클래스, **add** 함수가 정의되어 있지 않기 때문에 에러 메시지가 나옵니다).

**assertEquals** 함수는 함수의 결과값과 실제 결과값을 비교하여 매칭되지 않는 경우 **exception**을 발생시킵니다.

## 4.5 테스트 실행 (실패)



1. 상단 메뉴 Run > Run As > TestNG Test



2. 테스트 실행 결과

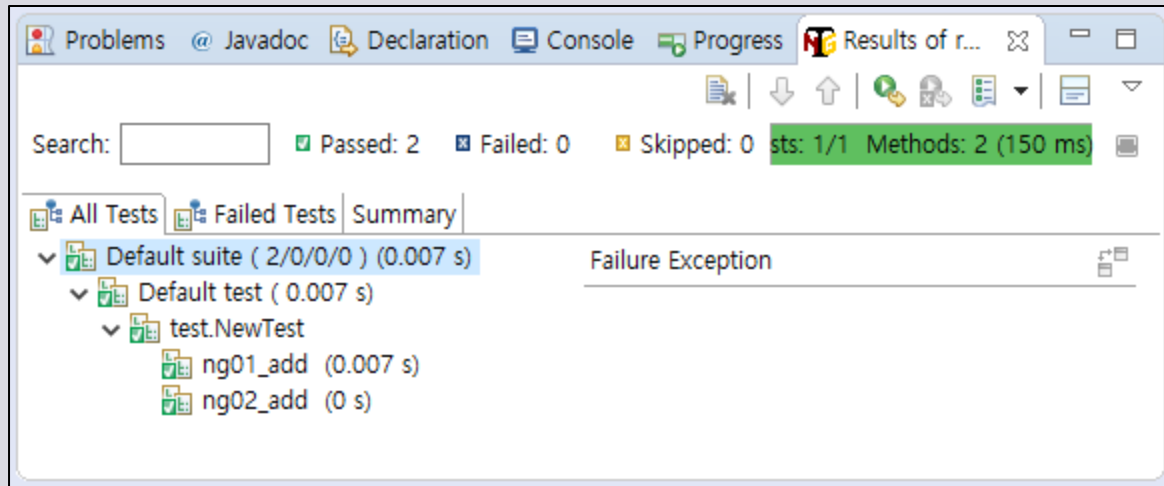
메인 코드가 작성되지 않은 상태에서 테스트를 실행하면 당연히 테스트 실패가 나옵니다.

TDD의 프로세스 첫 단계는 테스트 실패로 시작하며, 테스트를 성공시키기 위해 최소한의 코딩을 하는게 TDD의 주된 목적입니다.

# 5 & 6 메인 코드 작성 후 테스트 (성공)

```
1 package softzion;  
2  
3 public class Calc {  
4     public int add(int x, int y) {  
5         return x + y;  
6     }  
7 }  
8
```

1. 함수 추가



2. 테스트 실행

Calc 클래스를 생성하여 add 함수를 작성 했습니다.  
테스트를 실행해본 결과 무난히 성공했습니다.

# 7. 반복

```
22 @Test
23 public void ng03_subt() {
24     Calc calc = new Calc();
25     assertEquals(calc.subt(5, 2), 3);
26 }
27
28 @Test
29 public void ng04_subt() {
30     Calc calc = new Calc();
31     assertEquals(calc.subt(111, 35), 76);
32 }
```

1. 테스트 케이스 추가

```
1 package softzion;
2
3 public class Calc {
4     public int add(int x, int y) {
5         return x + y;
6     }
7
8     public int subt(int x, int y) {
9         return x + y;
10    }
11 }
```

2. 함수 추가

```
▼ Default suite ( 2/2/0/0 ) (0.009 s)
  ▼ Default test ( 0.009 s)
    ▼ test.NewTest
      ng01_add (0.007 s)
      ng02_add (0.001 s)
      ng03_subt (0 s)
      ng04_subt (0.001 s)
```

3. 테스트 결과 확인 (실패)

```
▼ Default suite ( 4/0/0/0 ) (0.01 s)
  ▼ Default test ( 0.01 s)
    ▼ test.NewTest
      ng01_add (0.008 s)
      ng02_add (0 s)
      ng03_subt (0.001 s)
      ng04_subt (0.001 s)
```

4. 테스트가 성공될 때 까지 메인 코드 수정

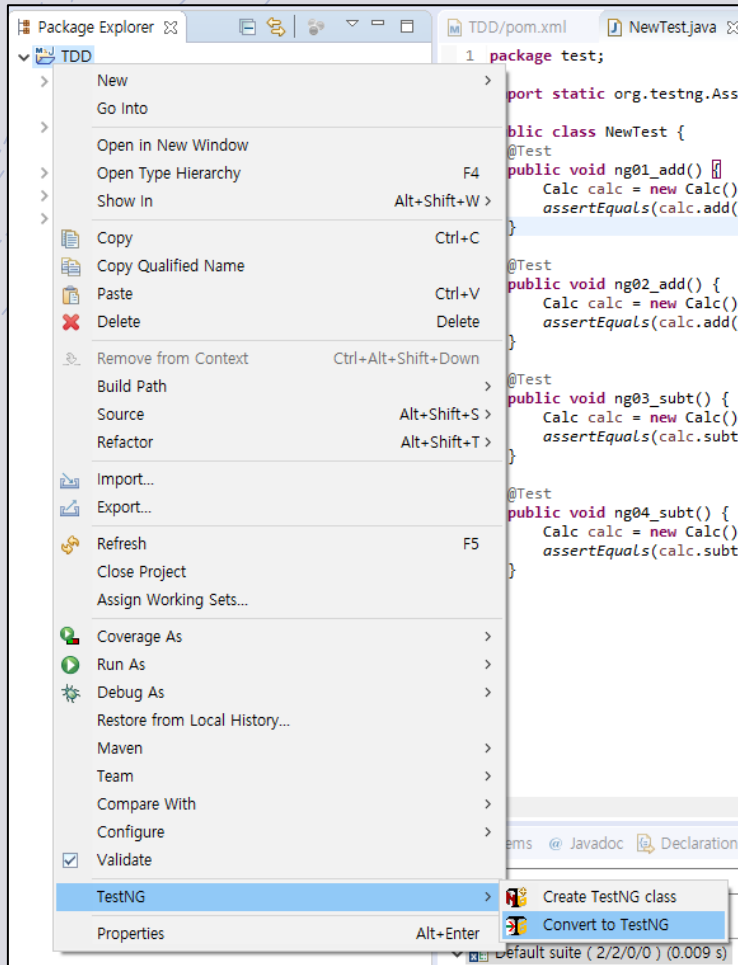
subt 테스트 케이스를 생성하고,  
메인코드에 subt 함수를 추가  
했습니다.

테스트를 실행해본 결과 subt  
함수에 에러가 있는 것을 확인하  
였으며, 메인 코드를 수정하여  
테스트한 결과 모든 테스트가 성  
공이 되었습니다.

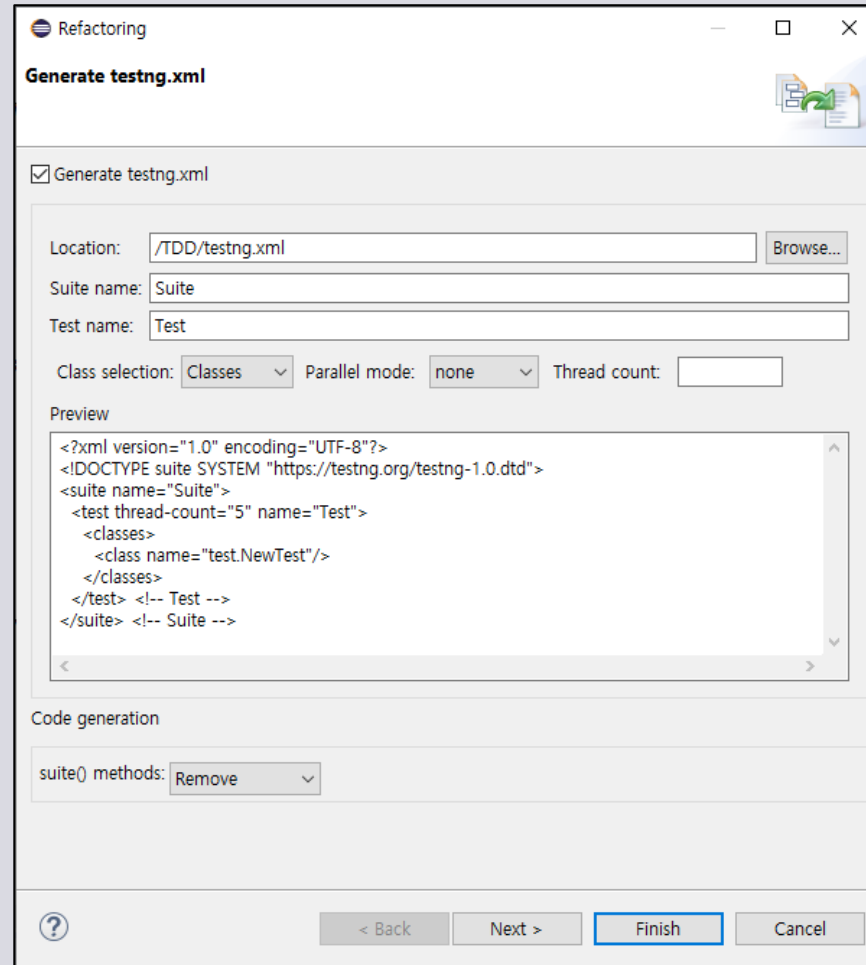
# 참고사항

- + 위 가이드에서는 함수 한 개씩 테스트 케이스 작성하고 개발을 하는 방식으로 반복하며 개발을 했지만, 실제로 TDD를 진행할 경우 모듈 단위로 테스트 케이스를 작성하거나 또는 미리 모든 테스트 케이스를 작성하고 개발을 진행합니다.
- + 테스트를 기준으로 개발 진척도를 확인할 수 있습니다. 예를 들어, 테스트 케이스가 10개 있는 경우, 그 중 7개 테스트가 성공되어 있으면 개발 진척도 70%로 여길 수 있습니다.
- + 테스트 실행하기 위해 굳이 eclipse를 사용할 필요는 없습니다. testng.xml 을 우선 생성한 후, command prompt로 프로젝트 폴더 경로로 이동하여 **mvn clean test** 명령어를 입력하면 테스트가 실행됩니다 (다음 슬라이드 참고).

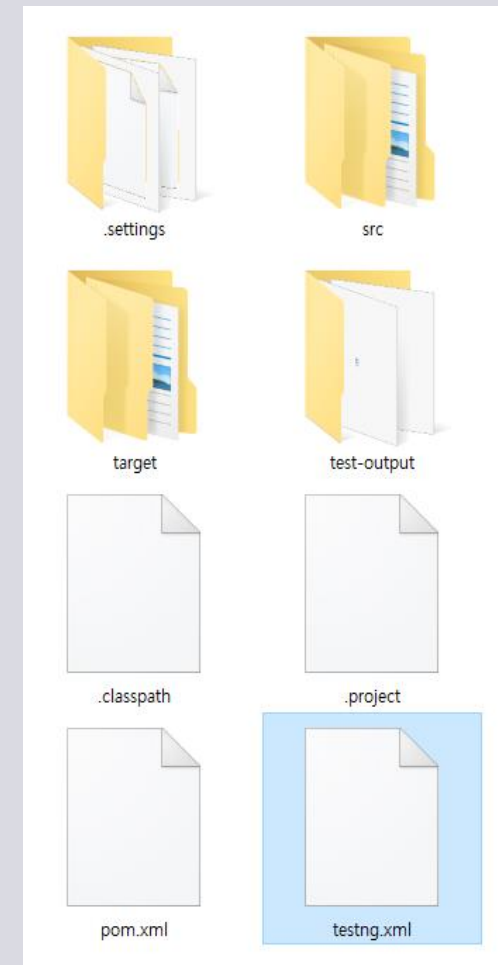
# 메이븐으로 테스트 실행 방법



1. 프로젝트 폴더 우클릭 > TestNG > Convert to TestNG



2. 따로 설정할 필요 없이 Finish



3. 코맨드창으로 프로젝트 폴더 경로에 이동하여 에 `mvn clean test` 입력