



Software Verification

# Introduction to

Eclipse, JDK, MAVEN, JENKINS, JUNIT

POWERED BY

201210908 윤성일

201311265 김상원

201214150 정성철





# Presentation Index

## 기본 개발 환경

Eclipse  
-  
JDK

## 빌드 환경 및 단위 테스트 툴

MAVEN  
-  
JENKINS  
-  
JUnit





# CTIP for JAVA

자바에서의 CTIP 환경을 구축



## IDE - Eclipse

그동안 자바 개발을 해온 익숙한 IDE인 Eclipse를 선택.

Maven과 연동 및 사용 경험이 있음.

## Build - Maven

저장소를 이용한 뛰어난 의존성 관리.

사용 경험이 있는 MAVEN을 선택.



# 기본 개발 환경

## Eclipse & JDK 란?

### ECLIPSE

이클립스 (Eclipse)는 다양한 플랫폼에서 사용가능.

자바를 비롯한 다양한 언어를 지원하는 프로그래밍  
통합 개발환경을 목적으로 시작

현재는 OSGi를 도입하여, 범용 응용 소프트웨어  
플랫폼으로 진화

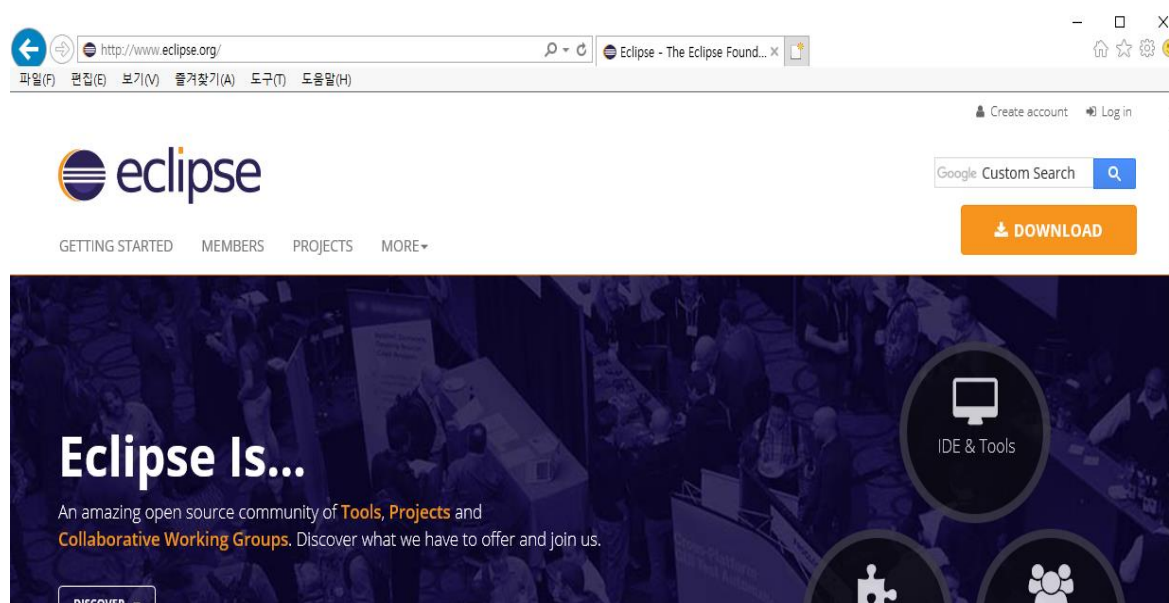
### JDK (Java Development Kit)

Java 환경에서 돌아가는 프로그램을 개발하는 데  
필요한 툴들을 모아놓은 소프트웨어 패키지



# ECLIPSE 설치

## 이클립스 설치 단계 1

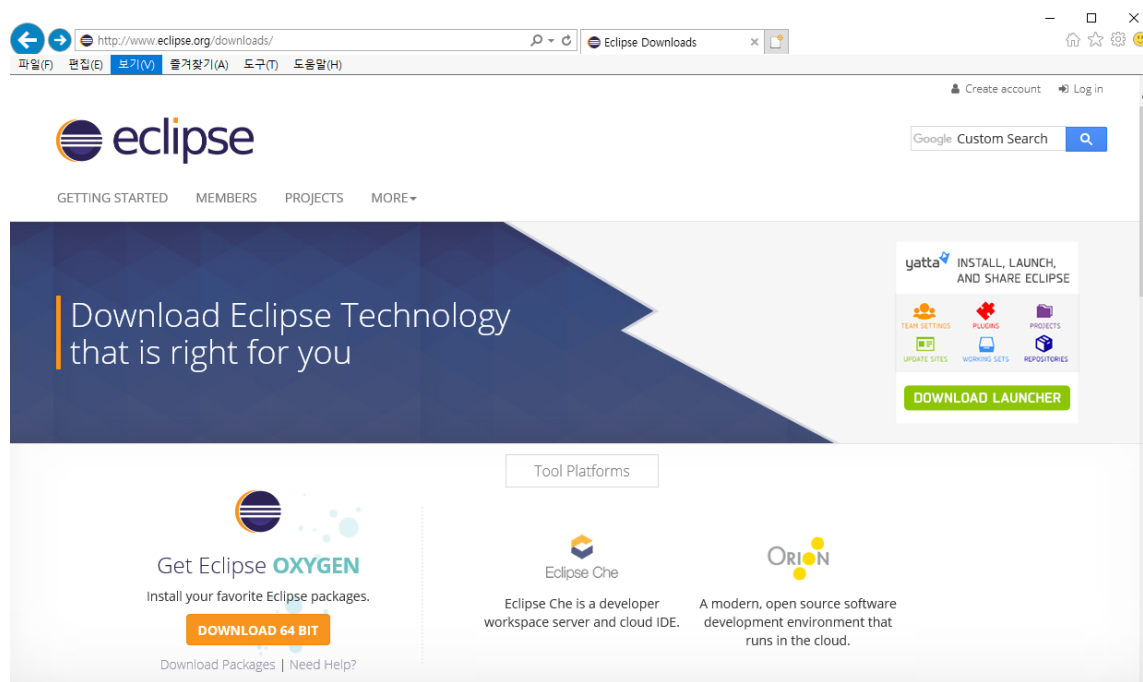


이클립스  
홈페이지([www.eclipse.org](http://www.eclipse.org))  
접속



# ECLIPSE 설치

## 이클립스 설치 단계 2



운영체제에 맞는 버전을  
설치



# ECLIPSE 설치

이클립스 설치 단계 3

Eclipse Installer



The required 64-bit Java 1.7.0 virtual machine could not be found.  
Do you want to browse your system for it?

예(Y)

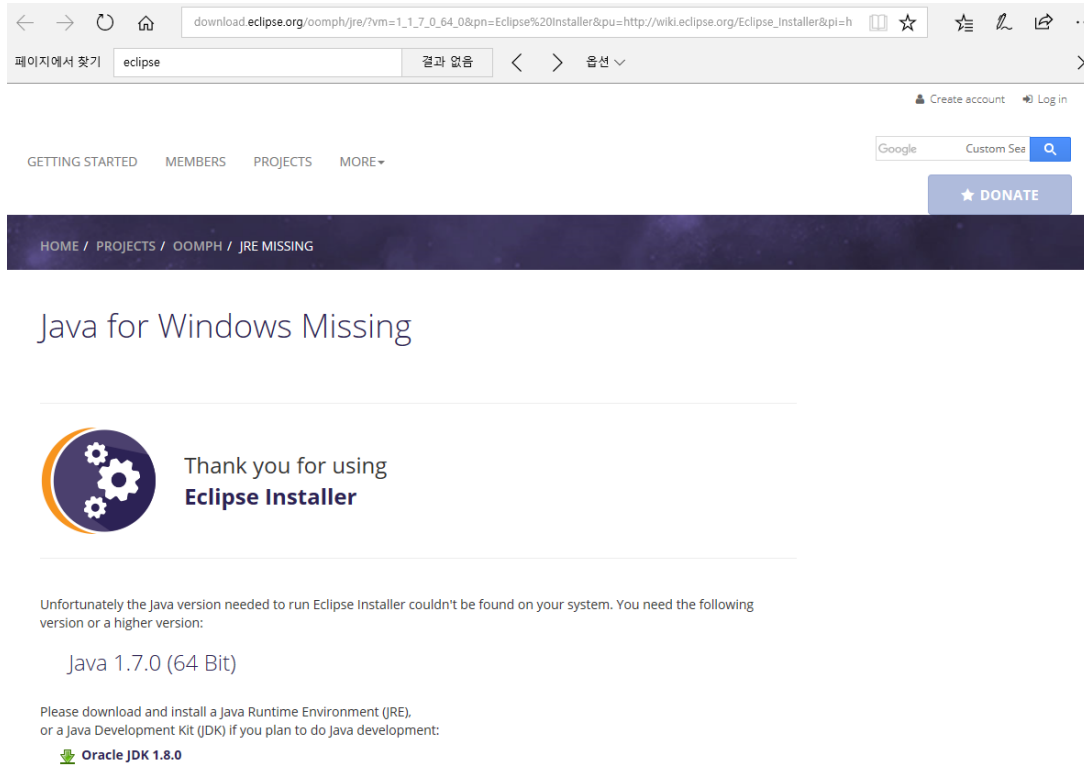
아니요(N)

예(Y) 클릭



# JDK 설치

## JDK 설치 단계 1



<http://www.oracle.com/technetwork/java/javase/downloads/jdk9-downloads-3848520.html>





# JDK 설치

## JDK 설치 단계 2

The screenshot shows the Oracle Technology Network page for Java SE Development Kit 8 Downloads. The page includes a navigation menu, a search bar, and a list of download links for various operating systems and architectures. A license agreement section is visible, with radio buttons for 'Accept License Agreement' and 'Decline License Agreement'. Below the license agreement is a table of download links with columns for Product / File Description, File Size, and Download.

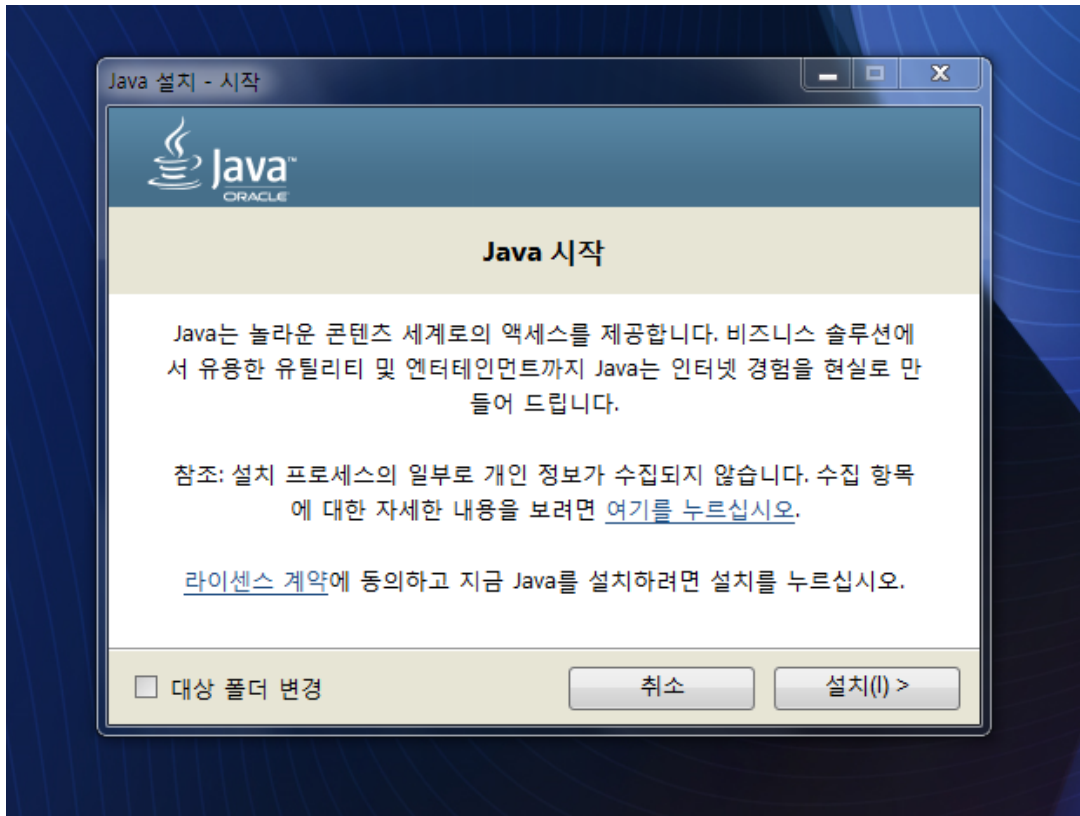
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	<a href="#">jdk-8u161-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.88 MB	<a href="#">jdk-8u161-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	168.96 MB	<a href="#">jdk-8u161-linux-i586.rpm</a>
Linux x86	183.76 MB	<a href="#">jdk-8u161-linux-i586.tar.gz</a>
Linux x64	166.09 MB	<a href="#">jdk-8u161-linux-x64.rpm</a>
Linux x64	180.97 MB	<a href="#">jdk-8u161-linux-x64.tar.gz</a>
macOS	247.12 MB	<a href="#">jdk-8u161-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	<a href="#">jdk-8u161-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.29 MB	<a href="#">jdk-8u161-solaris-sparcv9.tar.gz</a>
Solaris x64	140.57 MB	<a href="#">jdk-8u161-solaris-x64.tar.Z</a>
Solaris x64	97.02 MB	<a href="#">jdk-8u161-solaris-x64.tar.gz</a>
Windows x86	198.54 MB	<a href="#">jdk-8u161-windows-i586.exe</a>
Windows x64	206.51 MB	<a href="#">jdk-8u161-windows-x64.exe</a>

자신의 운영체제 맞는 버전  
설치



# JDK 설치

## JDK 설치 단계 3

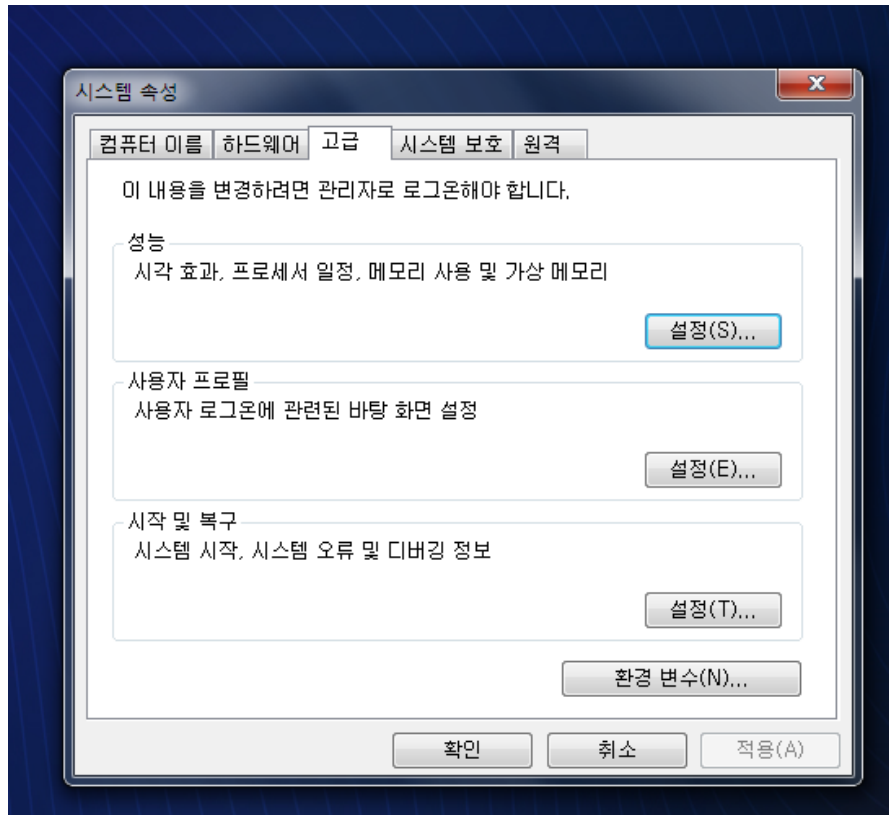


설치 버튼을 누르고  
컴퓨터에 자바를  
설치합니다.



# 환경 변수 설정

## JDK 설치 단계 4

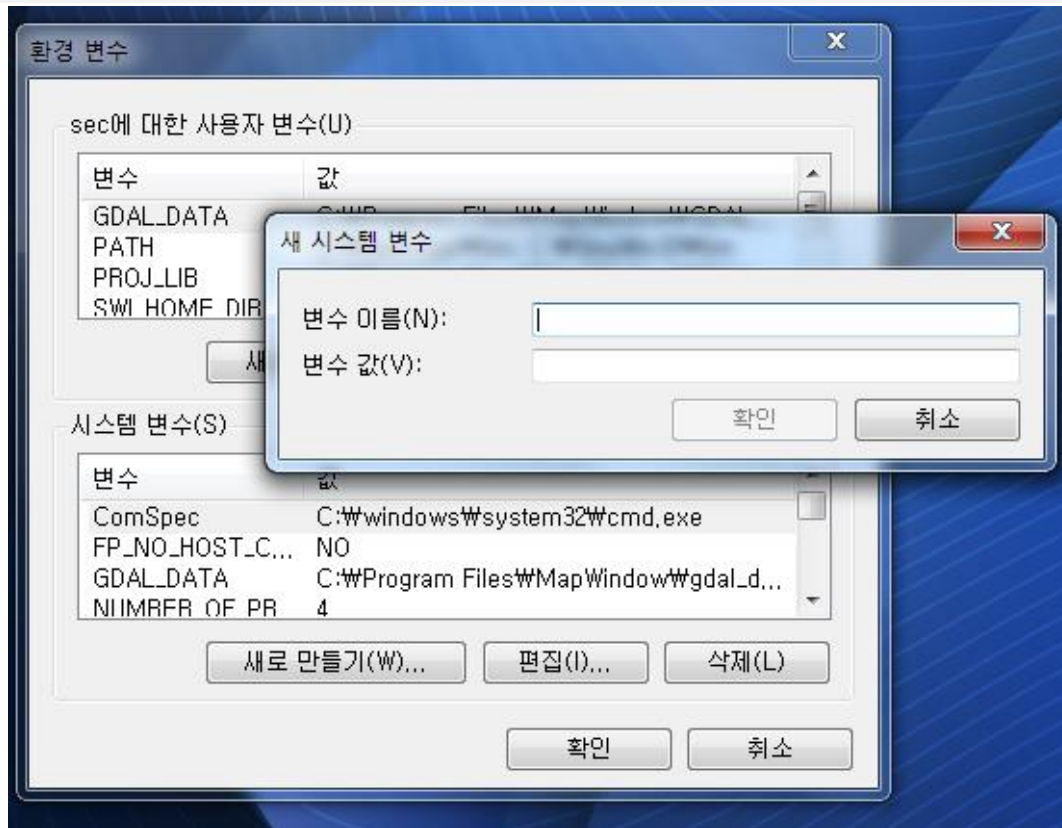


시스템 속성의 고급 탭에서  
환경변수 선택



# 환경 변수 설정

## JDK 설치 단계 5



변수 이름: JAVA\_HOME  
변수 값: (JRE가 설치된 경로)



# 프로그램 빌드 환경

## MAVEN 이란?

### MAVEN

아파치 앤트의 대안, 아파치 라이선스로 배포되는 오픈소스 소프트웨어

표준화된 관리방법을 제공하는 '프로젝트 관리 프레임워크'(Project Management Framework)

프로젝트의 전체적인 라이프 사이클을 관리하는 도구

저장소와 네트워크를 통한 라이브러리 관리

플러그인 기반 기능 수행

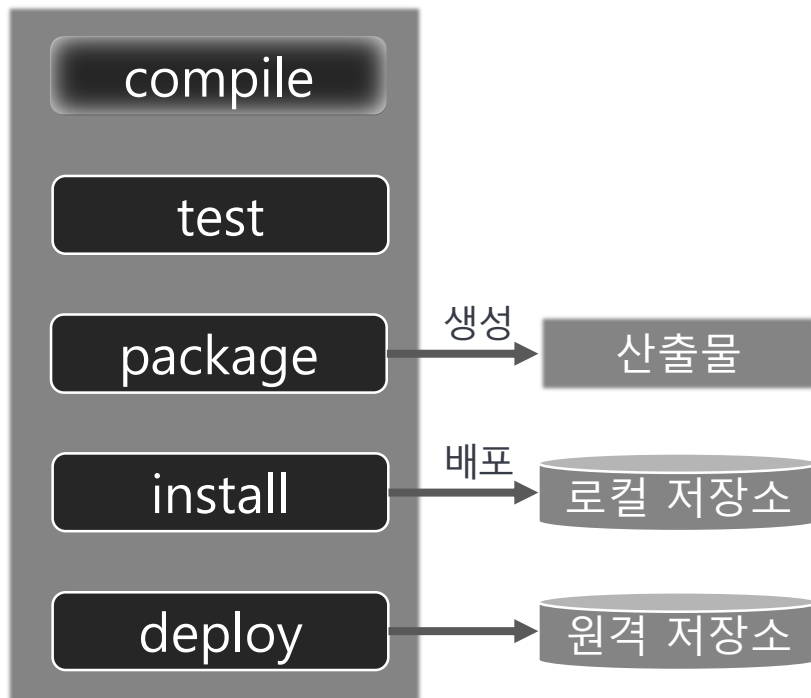




# 프로그램 빌드 환경

MAVEN LIFECYCLE (메이븐 라이프사이클)

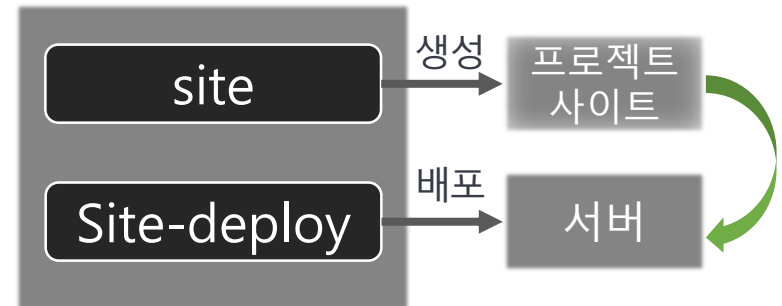
## 기본 라이프 사이클



## Clean 라이프 사이클



## Site 라이프 사이클





# 프로그램 빌드 환경

MAVEN LIFECYCLE (메이븐 라이프사이클)

이전 빌드에서 생성된  
파일들을 삭제하는 단계.

Clean

프로젝트의 소스코드를  
컴파일 하는 단계

Compile

Validate

프로젝트가 올바른지  
확인하고 필요한 모든  
정보를 사용할 수 있는지  
확인하는 단계

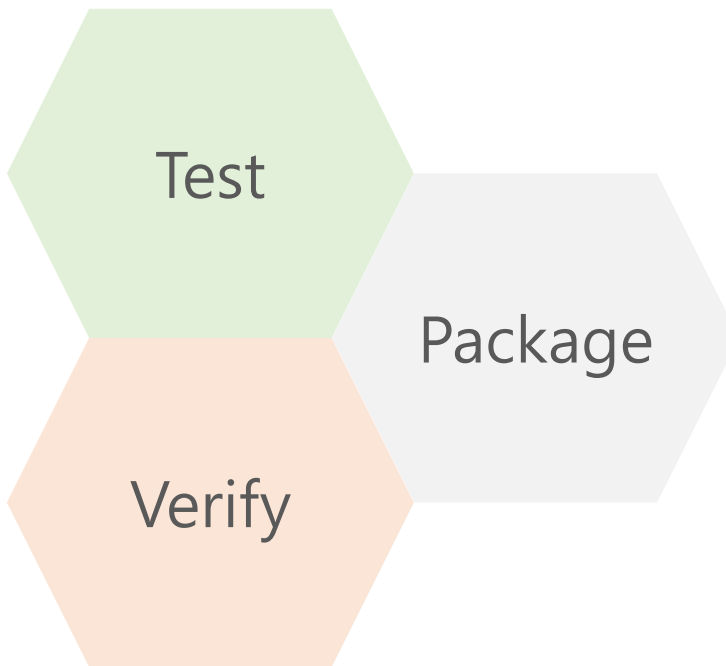


# 프로그램 빌드 환경

MAVEN LIFECYCLE (메이븐 라이프사이클)

유닛(단위) 테스트를 수행하는 단계(테스트 실패 시 빌드 실패로 처리, skip 가능)

통합 테스트 결과에 대한 검사를 실행하여 품질 기준을 충족하는지 확인하는 단계



실제 컴파일된 소스 코드와 리소스들을 jar 등의 배포를 위한 패키지로 만드는 단계





# 프로그램 빌드 환경

MAVEN LIFECYCLE (메이븐 라이프사이클)

패키지를 로컬 저장소에  
설치하는 단계

Install

만들어진 package를 원격  
저장소에 release 하는 단계

Deploy

Site

프로젝트 문서를  
생성하는 단계



# 프로그램 빌드 환경

Pom.xml 이란?

## POM (Project Object Model)

Project Object Model의 정보를 담고 있는 파일

프로젝트 정보, 빌드 설정, 빌드 환경, 의존 모듈, 상위 프로젝트 등 POM연관 정보

build 시, pom.xml을 읽어서, dependency에 정의된 jar 파일을 메이븐 저장소에서 다운로드

Group Id, Artifact Id, version 정보로 구별

Scope에서는 두가지 옵션: compile, provided

Compile: Web 패키지에 포함되는 jar

Provided: Web 패키지에 포함되지 않는 jar





# MAVEN 설치

## MAVEN 설치 단계 1

### Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bu

	Link	Checksum
Binary tar.gz archive	<a href="#">apache-maven-3.5.3-bin.tar.gz</a>	<a href="#">apache-maven-3.5.3-bin.tar.gz.md5</a>
Binary zip archive	<a href="#">apache-maven-3.5.3-bin.zip</a>	<a href="#">apache-maven-3.5.3-bin.zip.md5</a>
Source tar.gz archive	<a href="#">apache-maven-3.5.3-src.tar.gz</a>	<a href="#">apache-maven-3.5.3-src.tar.gz.md5</a>
Source zip archive	<a href="#">apache-maven-3.5.3-src.zip</a>	<a href="#">apache-maven-3.5.3-src.zip.md5</a>

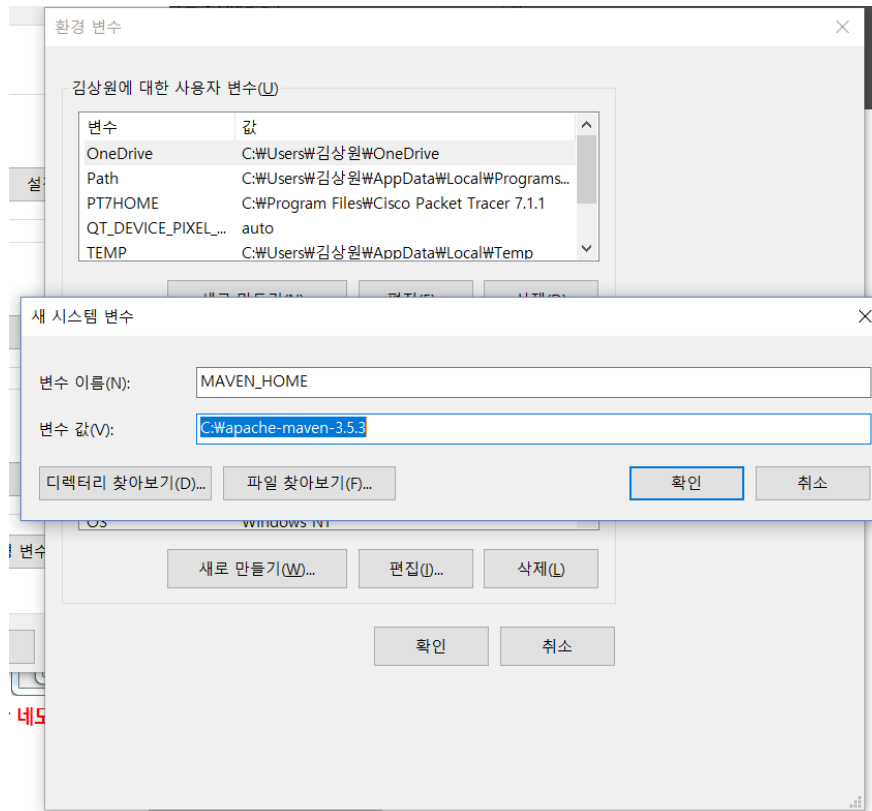
- [Release Notes](#)
- [Reference Documentation](#)
- [Apache Maven Website As Documentation Archive](#)
- All current release sources (plugins, shared libraries,...) available at <https://www.apache.org/dist/maven/>
- latest source code from source repository
- Distributed under the [Apache License, version 2.0](#)

<http://maven.apache.org/download.cgi>



# MAVEN 설치

## MAVEN 설치 단계 2



환경 변수 설정

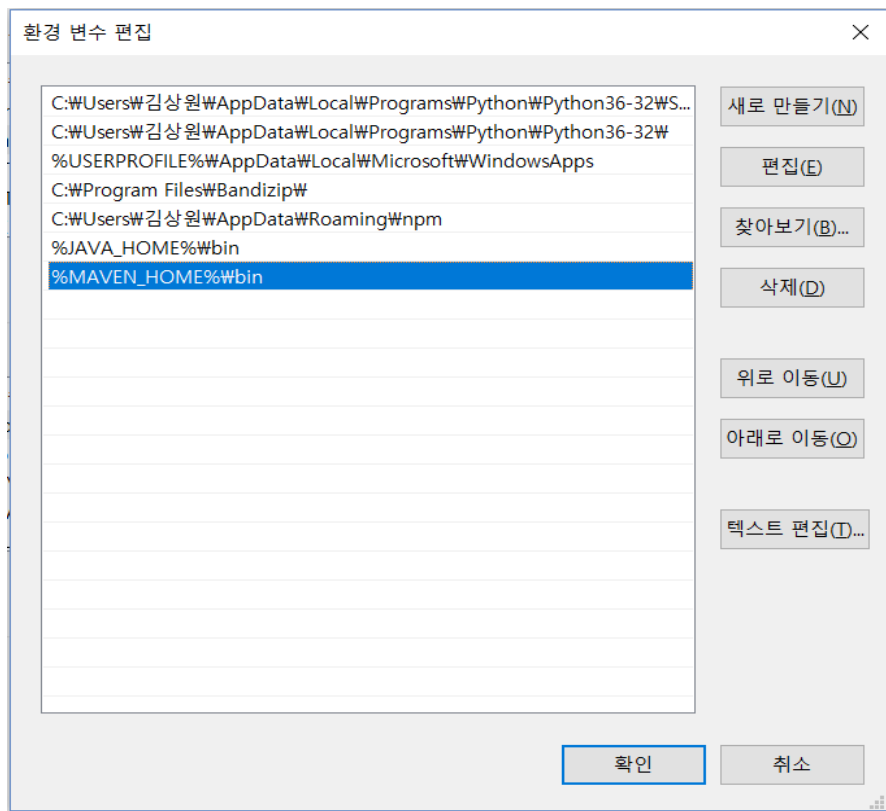
변수 이름: MAVEN\_HOME

변수 값: 설치된 경로



# MAVEN 설치

## MAVEN 설치 단계 3



PATH에 해당 경로 추가

%MAVEN\_HOME%\bin



# MAVEN 설치

## MAVEN 설치 단계 4

```
명령 프롬프트
Microsoft Windows [Version 10.0.16299.251]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\김상원>mvn --version
Apache Maven 3.5.3 (3383c37e1f9e9b3bc3df5050c29c8aff9f295297; 2018-02-25T04:49:05+09:00)
Maven home: C:\apache-maven-3.5.3\bin\..
Java version: 9.0.4, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk-9.0.4
Default locale: ko_KR, platform encoding: MS949
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

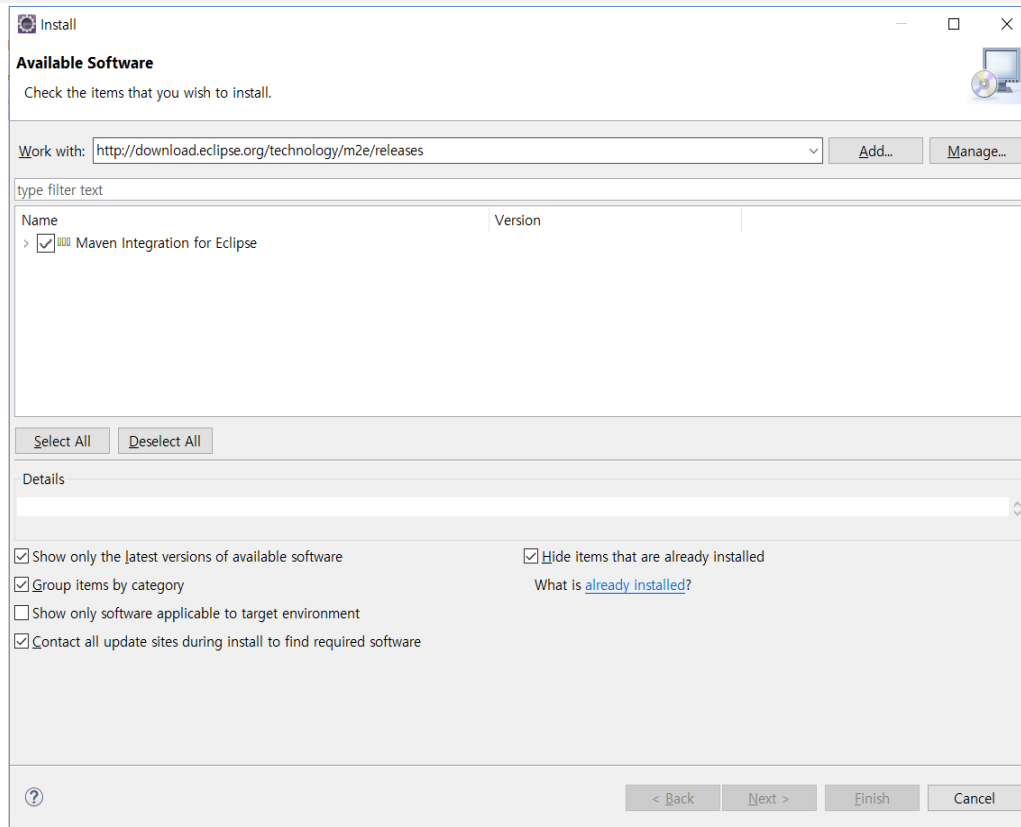
C:\Users\김상원>
```

cmd를 통해 정상 설치 확인



# Eclipse Maven 연동

## MAVEN 연동 단계 1



<http://www.eclipse.org/m2e/>

에서 최신 m2e 업데이트 url  
확인

이클립스 EE Oxygen (최신  
버전) 설치 시 Maven  
플러그인이 자동 설치되는  
것으로 확인되었음.



# Eclipse Maven 연동

## MAVEN 연동 단계 2

- File > New > Maven Project 선택

Group Id: artifact 제작을 책임지는 개체, 조직  
Artifact Id: 실제 Artifact 이름

Vesion: Artifact의 버전  
<mmm.nnn.bbb-qqqqqqq-dd>의 형태

M: 메이저 버전

N: 마이너 버전

B: 버그 수정 레벨

Q:(수식어)와 d(빌드) 추가 가능





# 프로그램 빌드 환경

## JENKINS이란?

### JENKINS

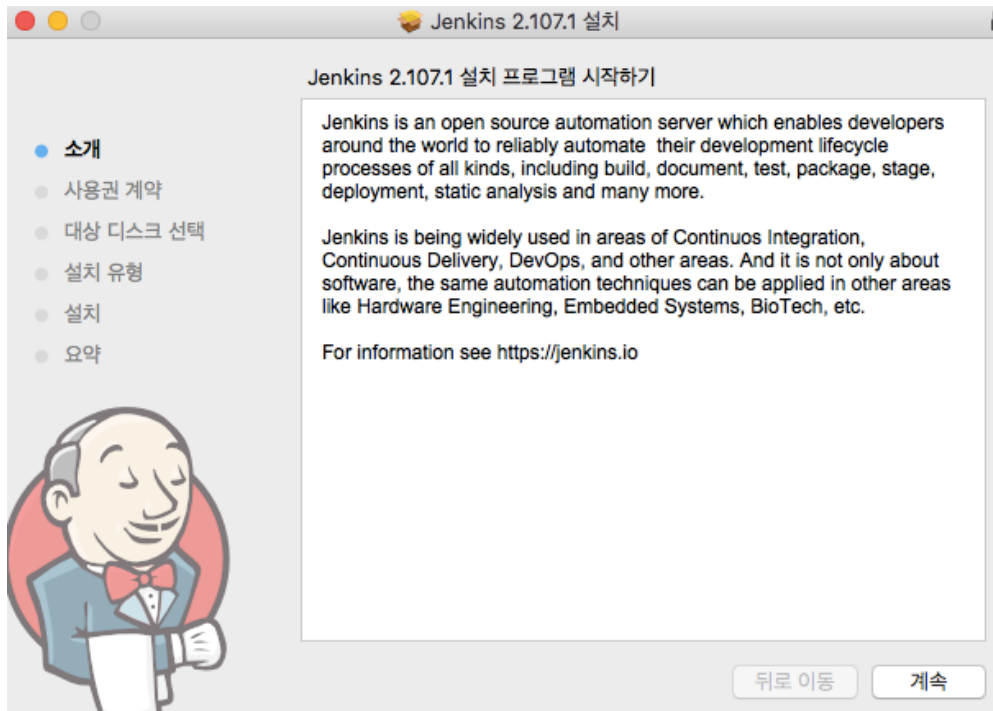
모든 언어의 조합과 소스코드 Repository에 대한 지속적인 통합과 전달 환경을 구축하기 위한 간단한 방법을 제공  
사용자가 구축하는 것 보다 빠르고 강력하게 빌드, 테스트, 배포 도구 등 체인 전체를 통합할 수 있는 방법을 제공  
젠킨스가 각각의 단계에 대한 스크립트 작성의 필요성을 없애 주지는 못함  
가장 유명한 CI tool이며, 많은 플러그인을 제공





# JENKINS 설치

## JENKINS 설치 단계 1



Jenkins 2.107.1 설치  
프로그램 실행



# JENKINS 설정

## JENKINS 설정 단계 1

### Getting Started

## Getting Started

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	** Script Security ** Command Agent Launcher
Timestampers	Workspace Cleanup	Ant	Gradle	<b>Folders</b> ** bouncycastle API ** Structs ** Pipeline: Step API
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	
Git	Subversion	SSH Slaves	Matrix Authorization Strategy	
PAM Authentication	LDAP	Email Extension	Mailer	

\*\* - required dependency

Jenkins 2.107.1

<http://localhost:8080>

접속(default)



# JENKINS 설정

## JENKINS 설정 단계 2

Getting Started

### Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 사용을 위한  
계정생성

Jenkins 2.107.1

Continue as admin

Save and Finish





# JENKINS 설정

## JENKINS 설정 단계 3

The screenshot shows the Jenkins web interface. At the top, there's a search bar and navigation links. The main content area is titled 'Jenkins 관리' and contains several configuration options, each with an icon and a brief description:

- 시스템 설정**: 환경변수 및 경로 정보등을 설정합니다.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials**: Configure the credential providers and types
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- 플러그인 관리**: Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.
- 시스템 정보**: 문제 해결을 돕기위한 다양한 환경 정보를 보여줍니다.
- System Log**: System log captures output from `java.util.logging` output related to Jenkins.

Jenkins 관리 탭에서  
Configure Global Security 설정



# JENKINS 설정

## JENKINS 설정 단계 4

### Configure Global Security

Enable security ?

Disable remember me  ?

Access Control

**Security Realm**

Delegate to servlet container ?

Jenkins' own user database ?

사용자의 가입 허용 ?

LDAP ?

Unix user/group database ?

**Authorization**

Anyone can do anything ?

Legacy mode ?

Logged-in users can do anything ?

Allow anonymous read access ?

Matrix-based security ?

Project-based Matrix Authorization Strategy ?

**Markup Formatter**

Markup Formatter  ?

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

사용자 가입을 허용하려면  
Security Realm의 Jenkins' own user  
database를 선택

아래의 사용자의 가입 허용  
체크박스에 체크

사용자별로 권한을 관리하려면  
Authorization의 Matrix-based  
security를 선택.

아래 User/group to add에 계정을  
입력하고 Add 버튼을 클릭



# JENKINS 설정

## JENKINS 설정 단계 5

### JDK

JDK installations

JDK

Name

M2

JAVA\_HOME

Install automatically

Delete JDK

Add JDK

List of JDK installations on this system

Jenkins Server의 JDK 설정

정보 입력

\* install automatically를

uncheck하면 정보를

입력가능

예) Name : jdk1.8.0\_111

JAVA\_HOME :

/opt/java/jdk1.8.0\_111



# JENKINS 설정

## JENKINS 설정 단계 6

List of JJK installations on this system

### Git

#### Git installations

	<b>Git</b>
Name	<input type="text" value="Default"/>
Path to Git executable	<input type="text" value="git"/>
<input type="checkbox"/> Install automatically	
<input type="button" value="Delete Git"/>	
<input type="button" value="Add Git"/>	

Jenkins Server의 Git 설정

정보를 입력.

예) Name : git

Path to Git executable :

/usr/bin/gi





# JENKINS 설정

## JENKINS 설정 단계 7

List of Ant installations on this system

### Maven

Maven installations

Maven

Name

M3

MAVEN\_HOME

Install automatically



Delete Maven

Add Maven

List of Maven installations on this system

Maven 설정 정보를  
입력한다.

예) Name : mvn

MAVEN\_HMLE :

/usr/share/maven





# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 1

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

Credentials

New View

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
		<a href="#">Maven</a>	-	-	-
		<a href="#">Test</a>	-	-	-

아이콘: [S](#) [M](#) [L](#)

[Legend](#) [RSS 모두](#) [RSS 실패](#) [RSS 최근 빌](#)

Jenkins 관리 탭에서 시스템  
설정



# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 2

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

Credentials

New View

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

All + 상세 내역

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
		<a href="#">Maven</a>	-	-	-
		<a href="#">Test</a>	-	-	-

아이콘: [S](#) [M](#) [L](#)

[Legend](#) [RSS 모두](#) [RSS 실패](#) [RSS 최근 빌](#)

새로운 Item 선택









# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 3

**Enter an item name**

» A job already exists with the name 'Test'

- 
**Freestyle project**  
 이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.
- 
**Pipeline**  
 Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- 
**Multi-configuration project**  
 다양한 환경에서의 테스트, 플레폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.
- 
**Folder**  
 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- 
**GitHub Organization**  
 Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- 
**Multibranch Pipeline**  
 Creates a set of Pipeline projects according to detected branches in one SCM repository.

프로젝트명 입력 후 아래  
Freestyle project를 선택



# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 4

**General**   소스 코드 관리   빌드 유발   빌드 환경   Build   빌드 후 조치

이름: Test1

설명: [Plain text] [미리보기](#)

GitHub project

Project url:  [?](#) 고급...

Throttle builds [?](#)

오래된 빌드 삭제 [?](#)

이 빌드는 매개변수가 있습니다 [?](#)

빌드 안함 [?](#)

필요한 경우 concurrent 빌드 실행 [?](#)

고급...

해당 프로젝트의 이름과 설명을 입력한후 GitHub project 에 체크 후 Project url 입력



# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 5

소스 코드 관리

None  
 Git

Repositories

Repository URL

**Please enter Git repository.**

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours

소스 코드 관리에서 Git  
설정 후 Repository URL 입력



# JENKINS 실행 시 설정

## JENKINS 실행 설정 단계 6

The screenshot shows the Jenkins 'Build' configuration page. The 'Invoke top-level Maven targets' section is expanded, showing the following fields:

- Maven Version: mvn
- Goals: clean package
- POM: pom.xml
- Properties: (empty text area)
- JVM Options: (empty text area)
- Inject build variables:
- Use private Maven repository:
- Settings file: Use default maven settings
- Global Settings file: Use default maven global settings

Buld 에서 invoke top-level  
Maven targets 선택 후  
입력창에 bulid 정보를 입력



# 단위 테스트 도구

## JUnit이란?

### JUnit

Java 프로그래밍 언어의 단위 테스트 프레임 워크

JUnit은 컴파일시, JAR로 링크

단정문으로 테스트 케이스의 수행 결과를 판별함

결과는 성공(녹색), 실패(붉은색) 중 하나로 표시







# JUnit

## JUnit의 대표적인 단정문

### 대표적인 단정문

`assertArrayEquals(a,b)` : 배열 a와b가 일치함을 확인

`assertEquals(a,b)` : 객체 a와b의 값이 같은지 확인

`assertTrue(a)` : a가 참인지 확인

`assertSame(a,b)` : 객체 a와b가 같은 객체임을 확인

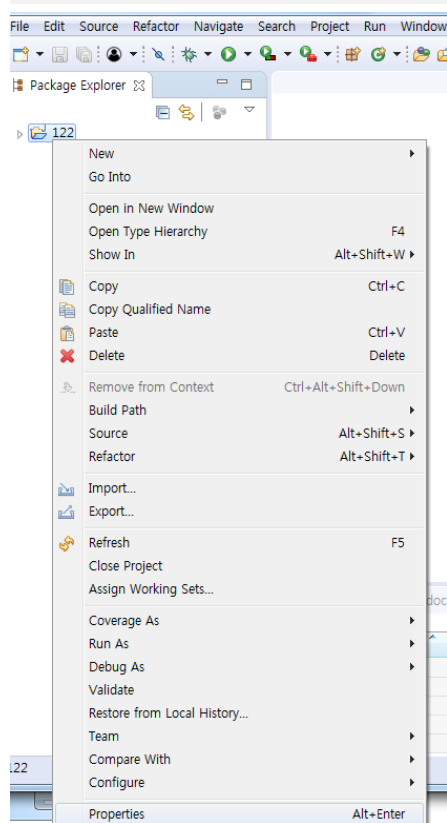
`assertNotNull(a)` : a객체가 null이 아님을 확인



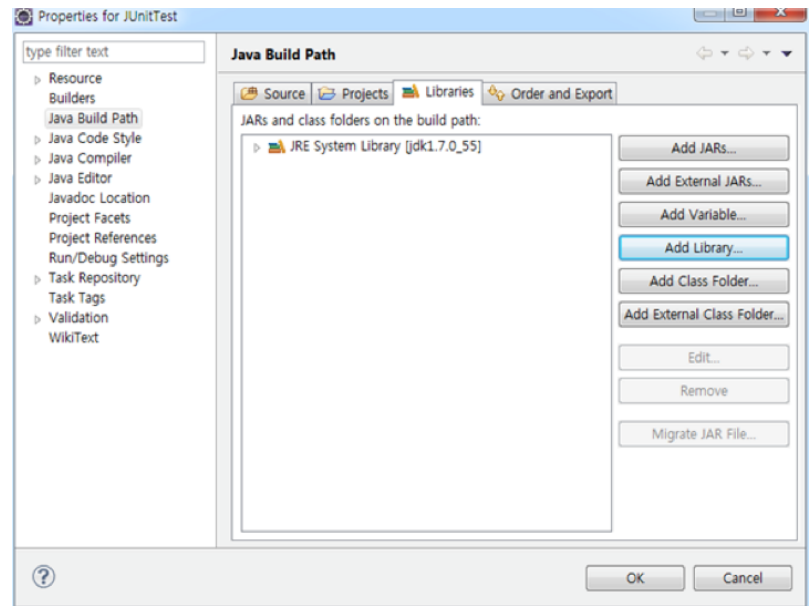


# Junit 실행 예시

## 실행 예시 -1



Test 코드에서 Properties  
선택

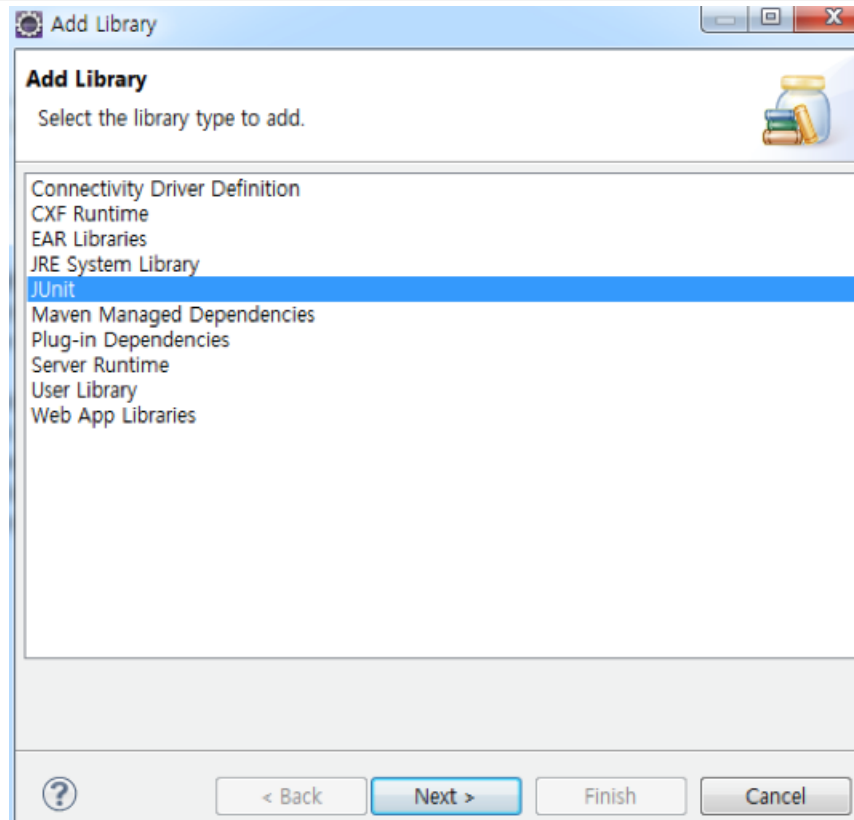


Java Build Path 선택  
Libraries 탭을 선택 후 Add Library 선택



# Junit 실행 예시

## 실행 예시 -2

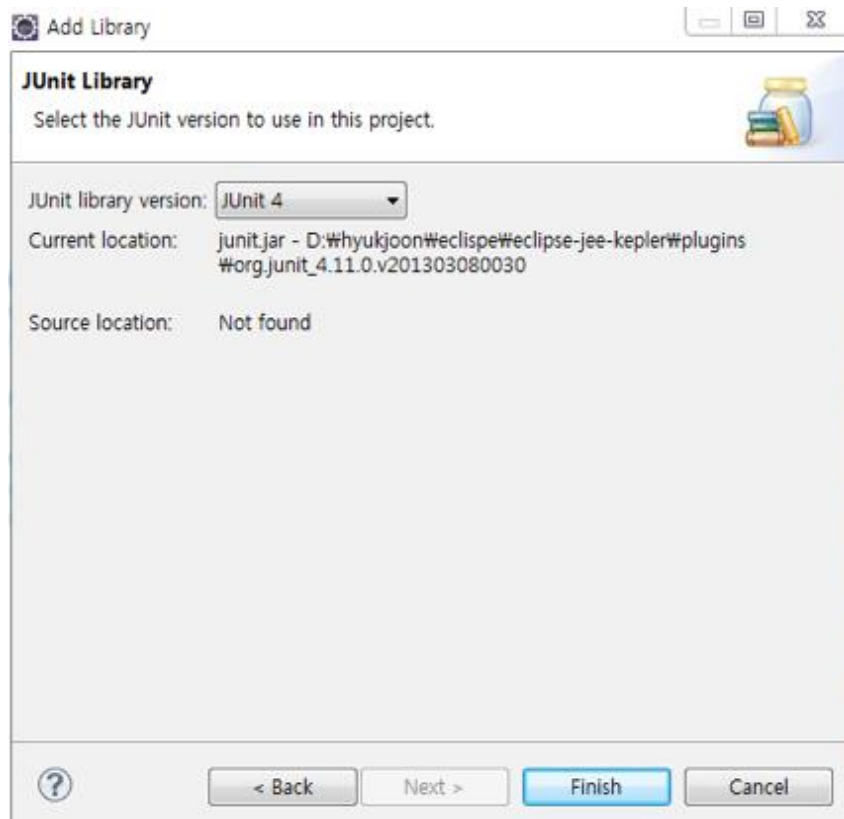


JUnit 선택 후 Next



# Junit 실행 예시

## 실행 예시 -3

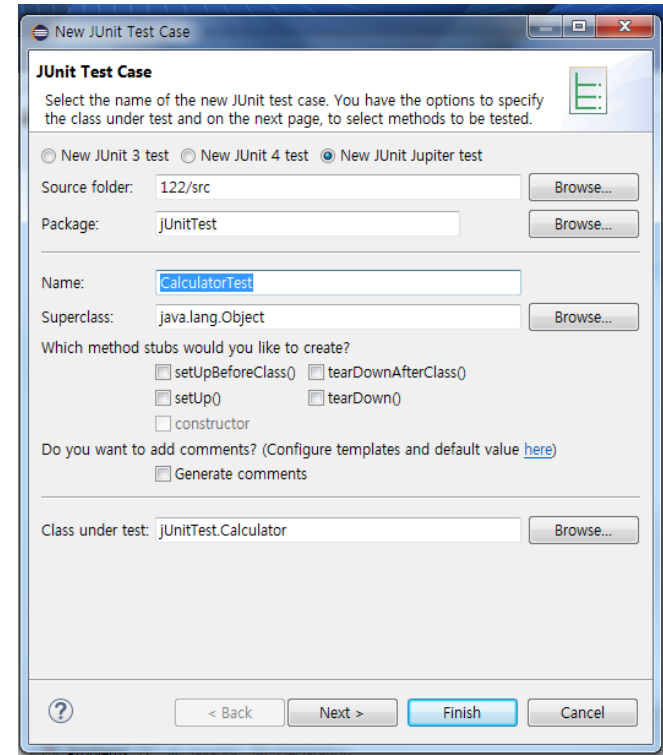
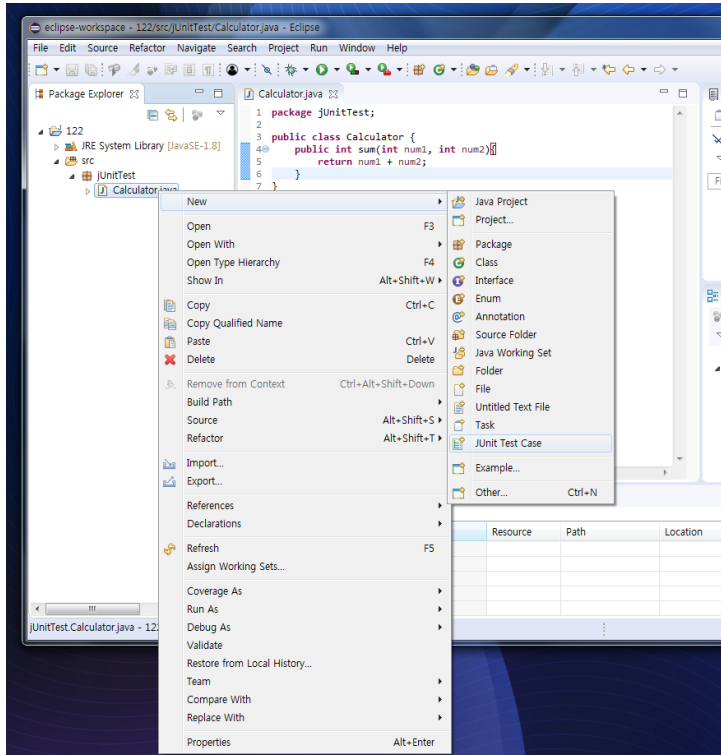


Version 선택 후 Finish



# Junit 실행 예시

## 실행 예시 -4

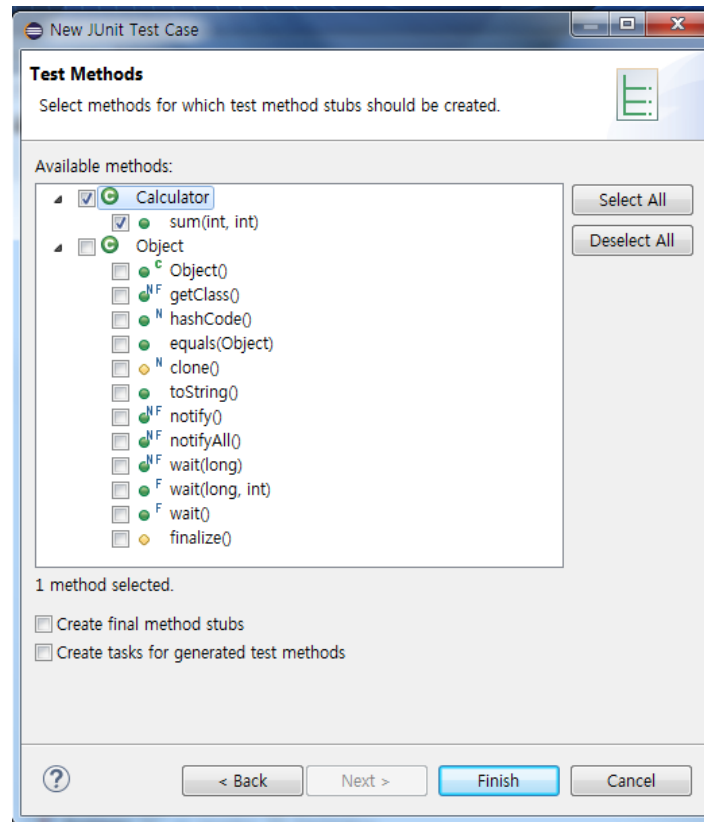


테스트할 소스 선택 후 Junit Test Case 선택



# Junit 실행 예시

## 실행 예시 -5



테스트 코드 선택 후 Finish  
선택



# Junit 실행 예시

## 실행 예시-6

```
1 package junitTest;
2
3+ import static org.junit.jupiter.api.Assertions.*;
6
7 class CalculatorTest2 {
8
9-  @Test
10 void testSum() {
11     fail("Not yet implemented");
12 }
13
14 }
15
```

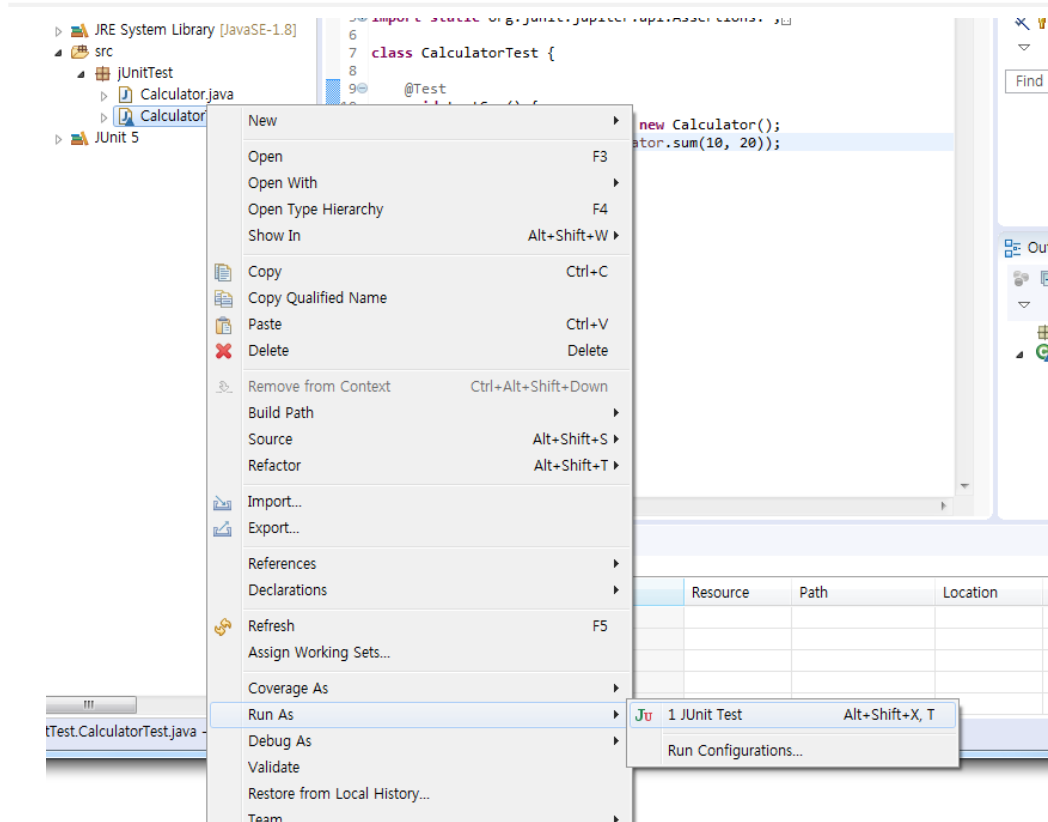
```
1 package junitTest;
2
3+ import static org.junit.jupiter.api.Assertions.*;
6
7 class CalculatorTest {
8
9-  @Test
10 void testSum() {
11     Calculator calculator = new Calculator();
12     assertEquals(30, calculator.sum(10, 20));
13 }
14
15 }
16
```

테스트할 소스 선택 후 Junit Test Case 선택



# Junit 실행 예시

## 실행 예시 -7



테스트할 소스 선택 후 Run  
As Junit 선택





# Junit 실행 예시

실행 예시-8

Finished after 0.058 seconds

Runs: 1/1   Errors: 0   Failures: 0

CalculatorTest [Runner: JUnit 5] (0.00)

```

1 package
2
3 import
6
7 class
8
9
10
11
12
13
14
15 }
16

```

Junit이 실행되는 것을 확인



SOFTWARE

I N D E X

# THANK YOU

ECLIPSE | JDK | MAVEN | JENKINS | JUNIT

201210908 윤성일  
201311265 김상원  
201214150 정성철

김 상 원

정 성 철

윤 성 일



Powered by dslab