소프트웨어검증 발표

——#1 junit , Eclipse , 정적분석도구

T5 201013759 근 량
201013760 기세파

contents

JUnit이란?

JUnit

Developer(s)	Kent Beck, Erich Gamma, David Saff, Mike Clark (University of Calgary)
Stable release	4.11[1] / November 14, 2012
Written in	Java
Operating system	Cross-platform
Type	Unit testing tool
License	Eclipse Public License[2]
Website	junit.org

wikipedia

Junit는 가장 많이 사용되는 Java 단위테스트 프레임워크.

콘솔 환경에서 명령행으로도 실행 가능.

Eclipse에는 기본 도구로 되어 있음.

Junit의 필요한 예

【예1】

제가 만든 각종 연산하는 프로그램이 있습니다. 문제는 이상은 없는것 같은데 결과가 이상합니다. 그래서 테스트를 시작합니다.
덧셈부분이 잘못 됐다고 치고, 일단 차례대로 메소드들은 테스트 할 것입니다. 그런데 잘못된 덧셈을 하기 전 각각의 메소드 들을 모두 테스트 해야하고 잘못된 선택으로 취소가 됐다면 다시 처음부터 테스트를 해야합니다.

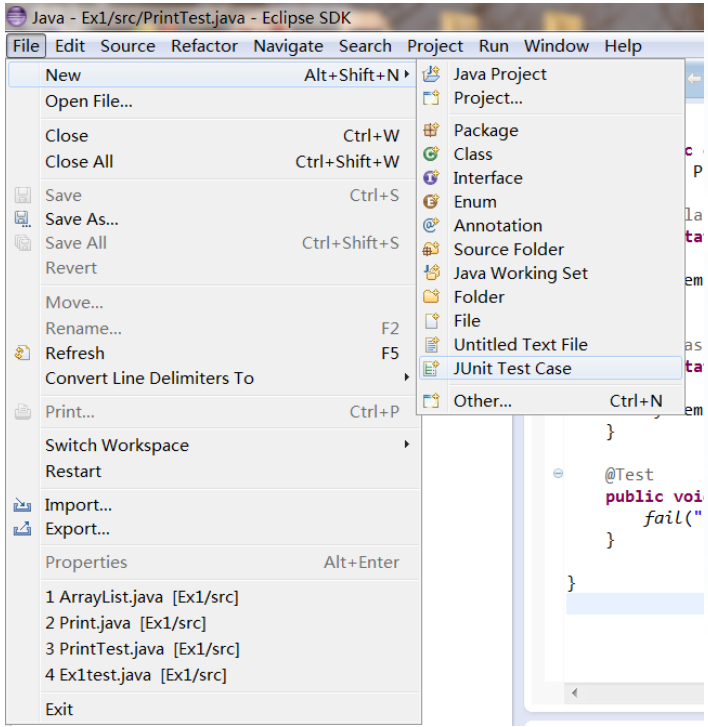
【예2】

DB와 관련된 클래스입니다. 임시적으로 테스트를 위해 DB 접속을 위한 암호를 넣고 테스트 후 남겨뒀다고 하면, 이 정보는 심각한 보안상 문제를 일으킬수 있습니다.

JUnit의 장점

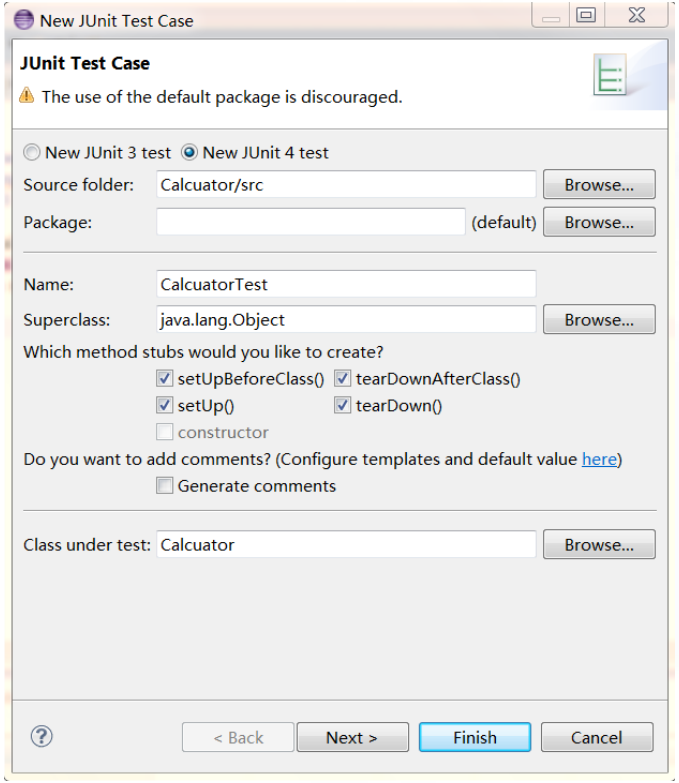
- 메소드 정확히 구현되었는지를 확인
- 원하는 모듈만 순서대로 테스트 가능
- 단위 테스팅으로 통합 테스트시의 회귀결함을 감소
- 단위 테스팅으로 통해 코드품질을 보장

Junit 사용 예



Eclipse에서는**JUnit**을기본적으로사용.

Test building



JUnit 3와4를선택

Source code 경로

Test Fixture

Main class

```
public class Calcuator {
    public double
add(double n1, double n2)
{
        return n1 + n2;     //  Integer Return Method
    }
}
```

더하기 연산자

test1

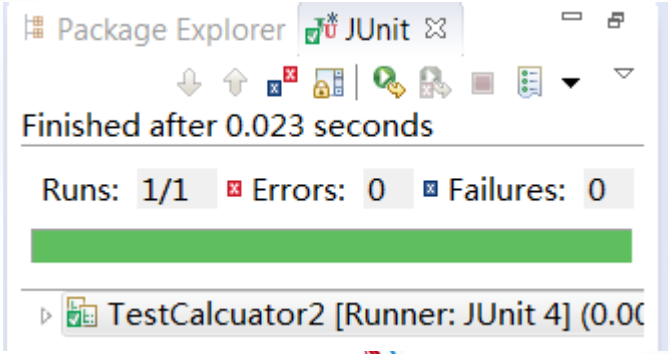
```
import junit.framework.TestCase;

public class TestCalcuator
extends TestCase {
    public void testAdd(){
        Calcuator calcuator=new
Calcuator();
        double
result=calcuator.add(1,2);
        assertEquals(3,result,0);
//기대값(expected) 과실제값(actual) 이같은지비교

    }

}
```

assertEquals를이용한 테스팅 성공시 초록색으로 표시

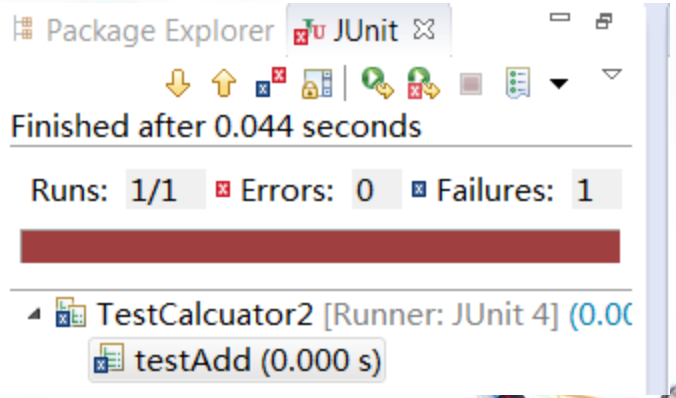


test2

```
import junit.framework.TestCase;

public class TestCalcuator2
extends TestCase  {
    public void testAdd(){
        Calcuator calcuator=new
Calcuator();
        double
result=calcuator.add(1,2);
        assertEquals(5,result,0);
// 기대값(expected) 과실제값(actual) 이같은지비교
    }
}
```

assertEquals로 테스팅
실패시 빨간색으로
나타내고해당Unit표시.



All test

```
import junit.framework.TestSuite;
import junit.framework.Test;
import junit.textui.TestRunner;
public class TestAll extends TestSuite
{
    public static Test suite() {
        TestSuite suite = new
TestSuite("TestSuite Test");

suite.addTestSuite(TestCalcuator.class);

suite.addTestSuite(TestCalcuator2.class);
        return suite;
    }
    public static void main(String
args[]){
        TestRunner.run(suite());
    }
```

addTestSuite테스트 클래스 추가(클래스명.class) 여러개의 Test클래스들을 모아 실행할 때 쓰인다.

테스트 실행 주석

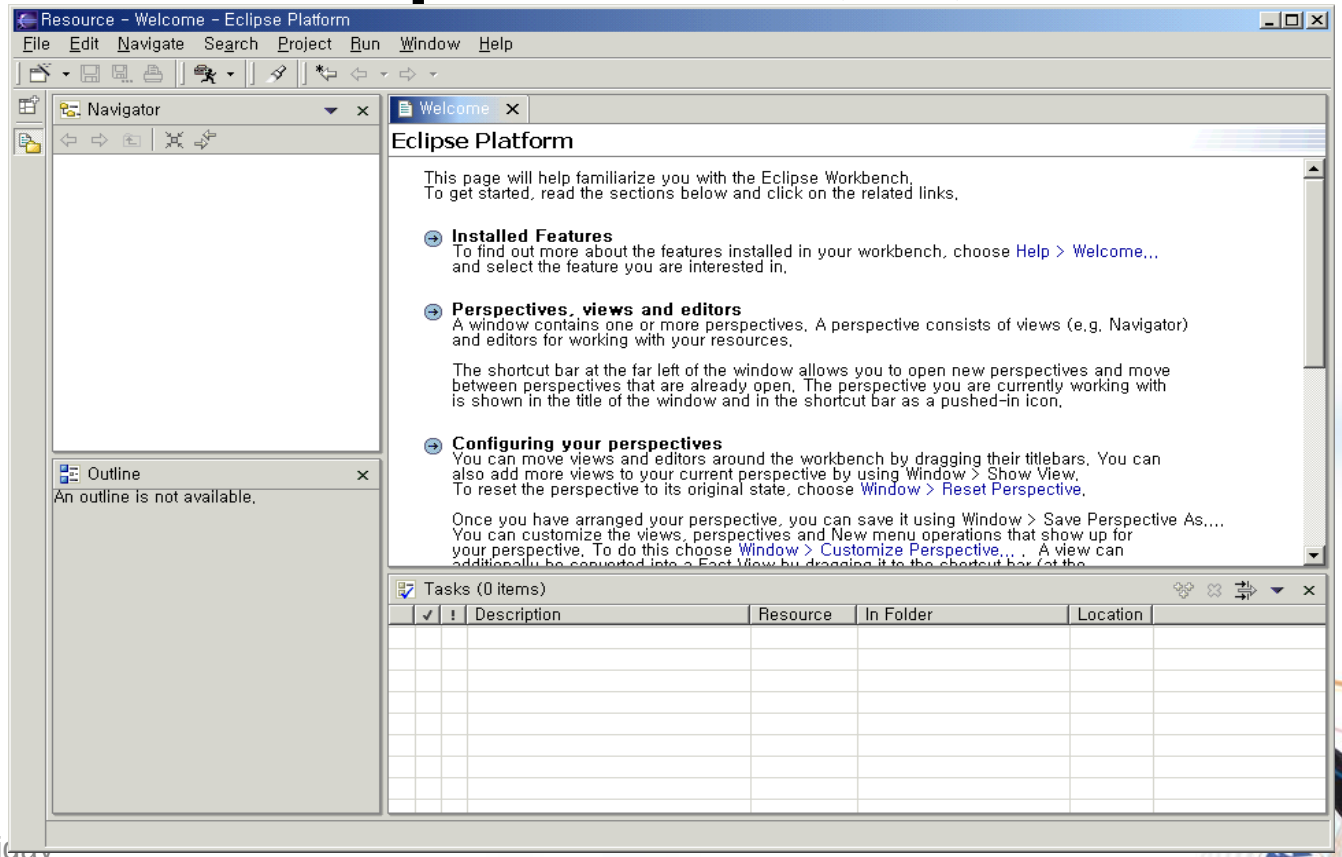
- @Before Test method가 실행되기 전에 실행되는 method
- @Before Class
- - Test를 시작하기전에 한번만 실행한다.
- @After Test method가 실행되고 난 후에 실행되는 method
- @After Class
- - Test를 마친후에 한번만 실행한다
- @Test
- - Test를 실행할 method 앞에 붙임
- - expected
- 발생할 것으로 예상되는 예외를 지정. 예외가 생기지 않으면 실패.
- - timeout
- 테스트가 끝나는 시간을 예측. 시간보다 길게 끝나면 실패.
- @Ignore
- - 테스트를 하지 않을 method 앞에 붙임

Eclipse

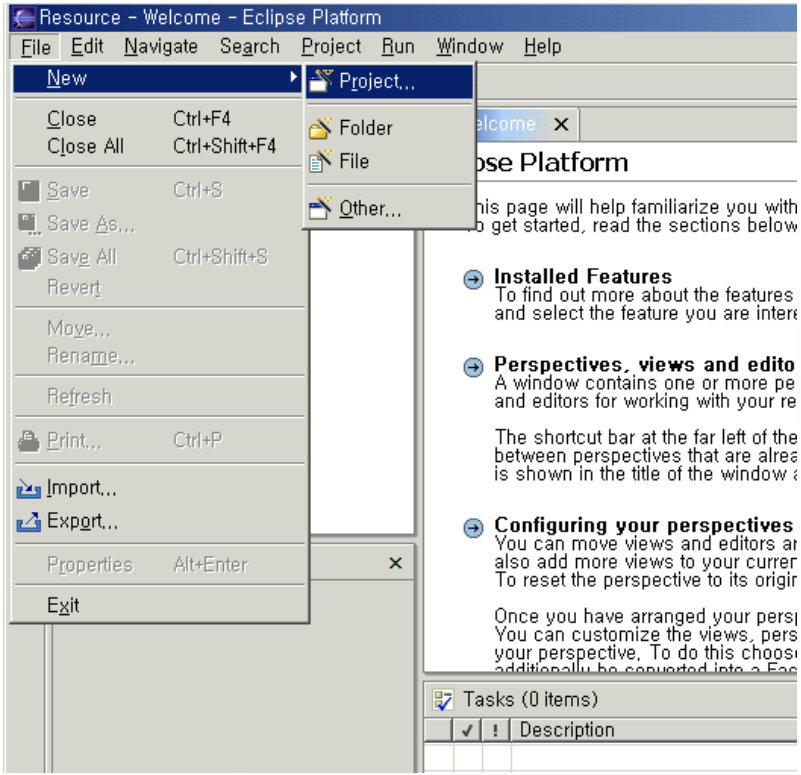
- 자바로 작성된 자유 소프트웨어 (EPL License)
- 다양한 언어를 지원하는 통합

- 다운로드
  - http://www.eclipse.org/downloads/index.php 에서 다운받은 후 설치

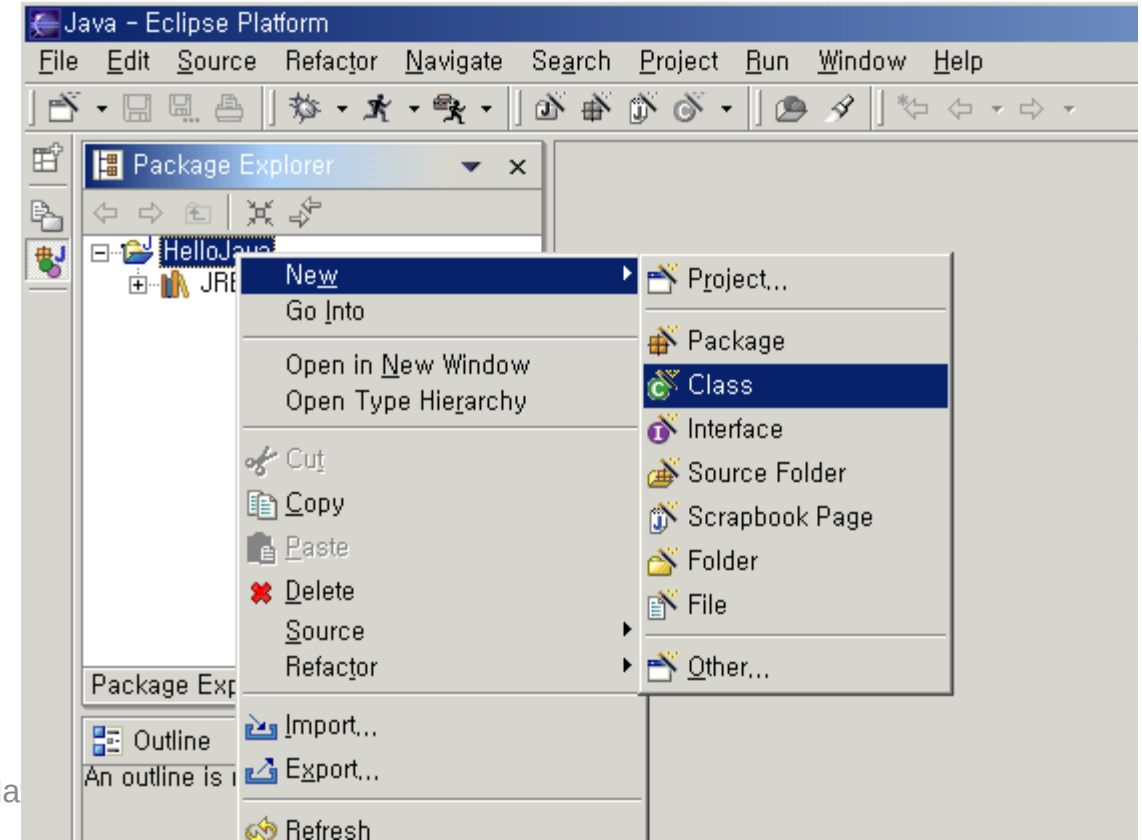
Eclipse 초기화면



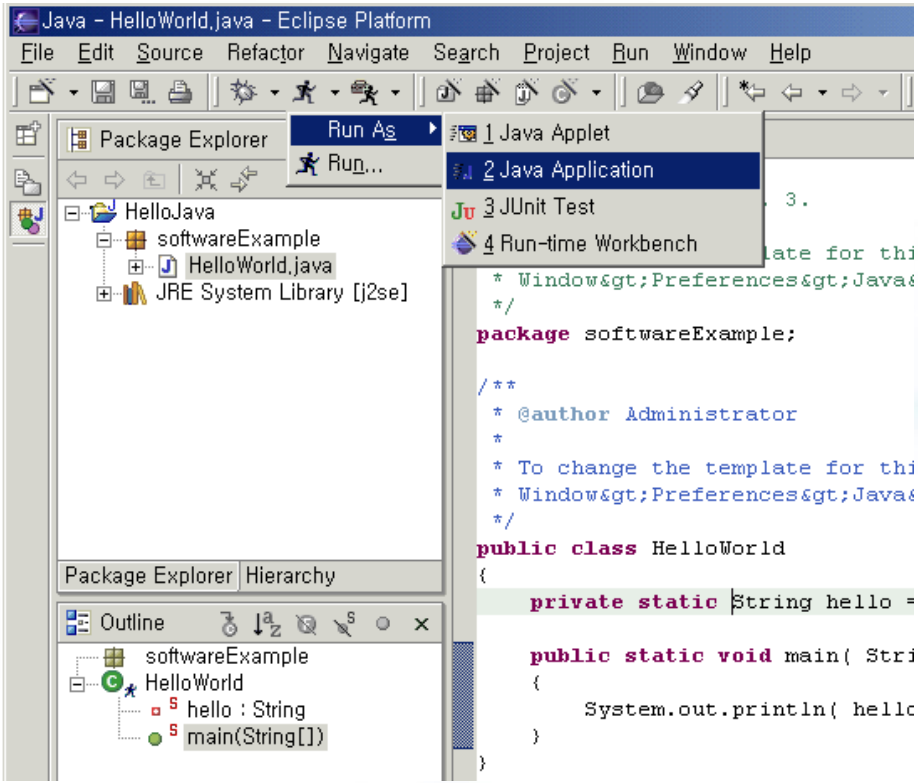
Project 생성


Resource - Welcome - Eclipse Platform
File Edit Navigate Search Project Run Window Help
New
Close Ctrl+F4
Close All Ctrl+Shift+F4
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename...
Refresh
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Exit
Project...
Folder
File
Other...
Installed Features
Perspectives, views and edito
Configuring your perspectives
Tasks (0 items)
Description

Class 생성


Java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
HelloJava
JRE
New
Go Into
Open in New Window
Open Type Hierarchy
Cut
Copy
Paste
Delete
Source
Refactor
Import...
Export...
Refresh
Project...
Package
Class
Interface
Source Folder
Scrapbook Page
Folder
File
Other...
Outline
An outline is

Eclipse - 실행

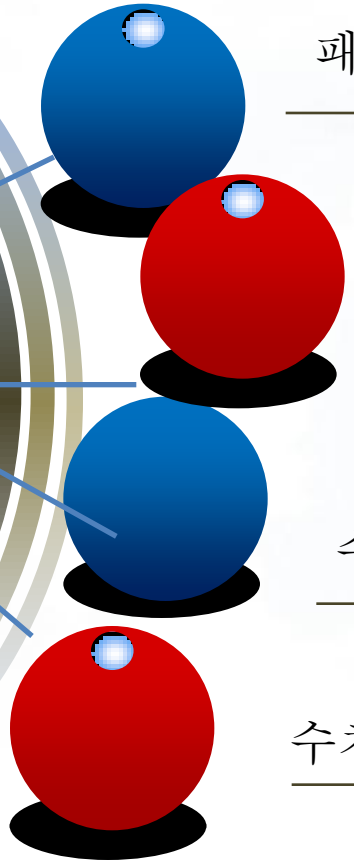

Java - HelloWorld.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Run As
Run...
1 Java Applet
2 Java Application
3 JUnit Test
4 Run-time Workbench
Package Explorer
HelloJava
softwareExample
HelloWorld.java
JRE System Library [j2se]
Package Explorer Hierarchy
Outline
softwareExample
HelloWorld
hello : String
main(String[])
* Window>Preferences>Java
*/
package softwareExample;
/**
* @author Administrator
*
* To change the template for thi
* Window>Preferences>Java
*/
public class HelloWorld
{
private static String hello =
public static void main(Stri
{
System.out.println(hello
}
}

JDepend

패키지 의존성과 설계 품질의 객체 지향식 측정을 통해, 패키지를 분석하고 관리할 수 있도록 지원해주는도구.

특징

jdepend



패키지 별로 의존성 측정가능

패키지 의존성과 관련된 데이터 품질을 수치화하여 표현

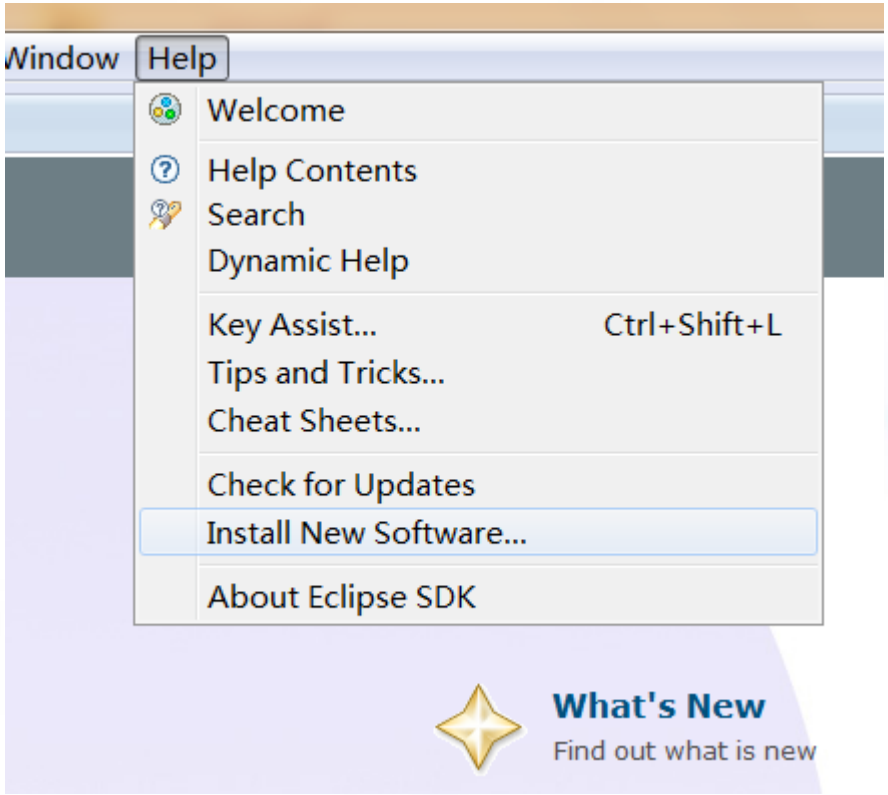
수치화된 의존성 정보를 텍스트 형태로 제공

수치화된 데이터 품질을 그래프로 표현

설치방법step1:

Help -> Install New Software

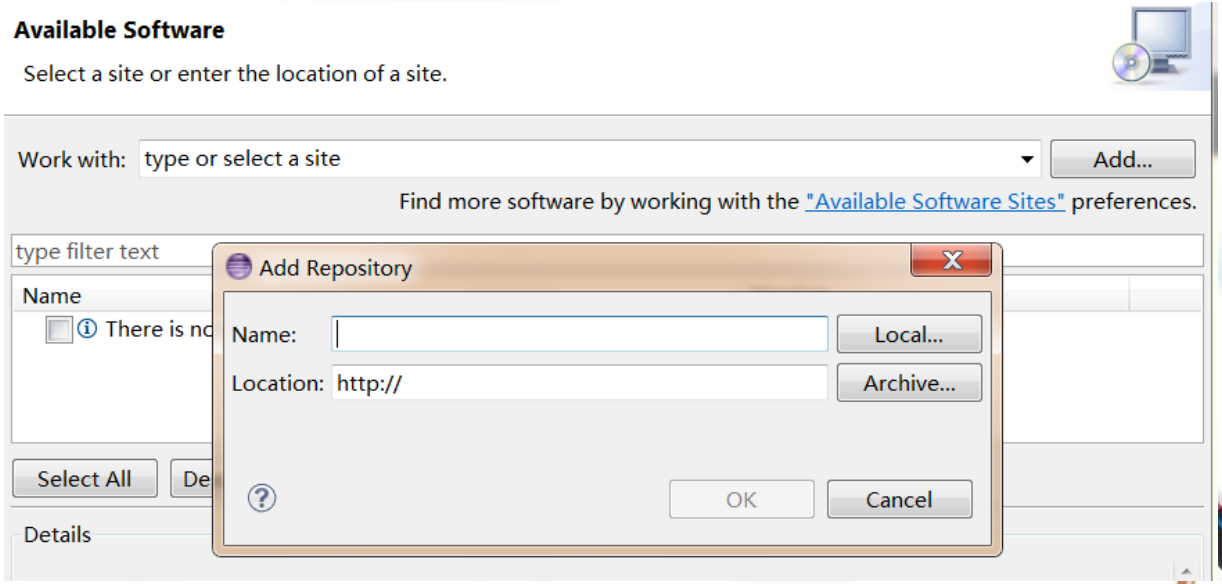
44



설치방법step2 :

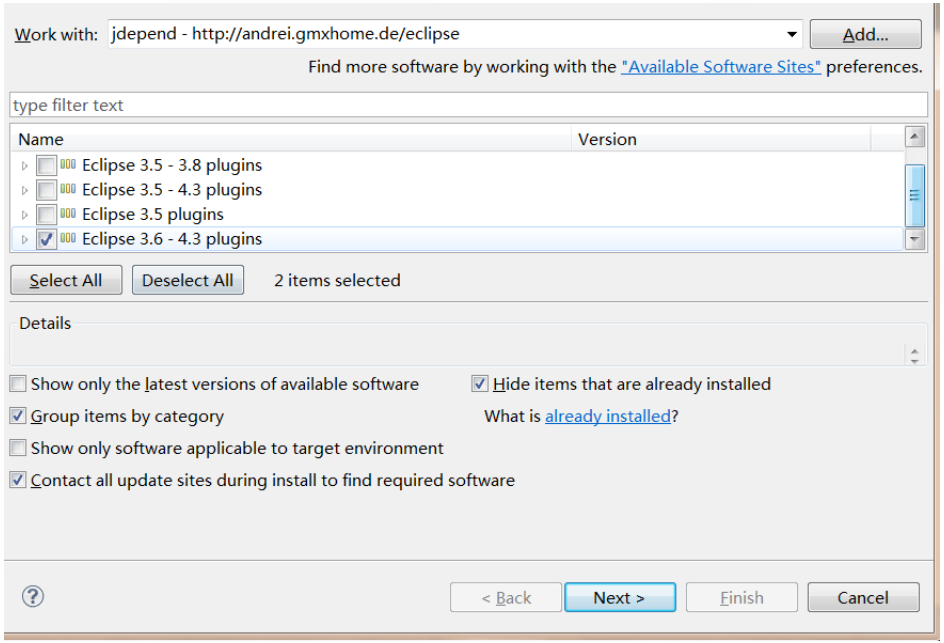
Add 클릭 Location :

http://andrei.gmxhome.de/eclipse/



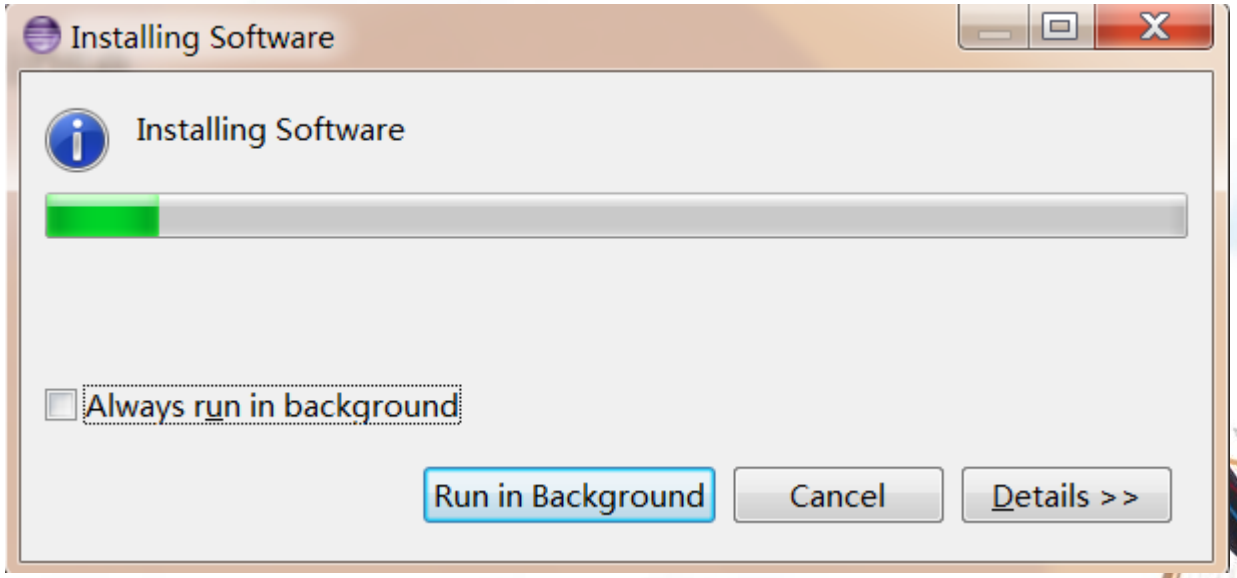
설치방법step3:

name중의 ecilpse 3.6-4.3 plugins 체크 ‘Next’ 버튼을클릭.



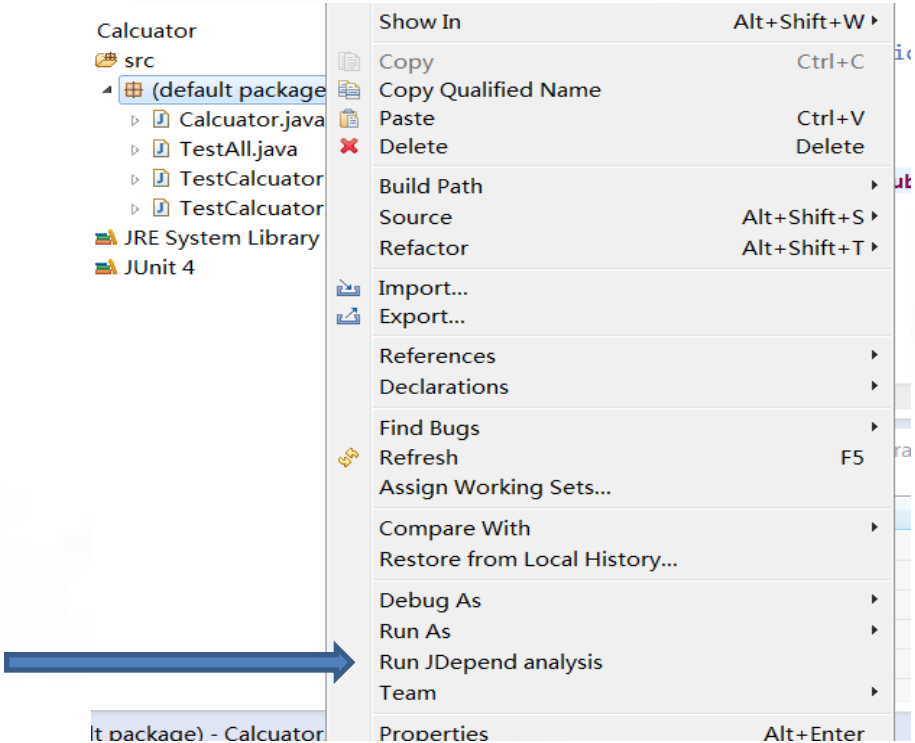
설치방법step4:

이어서 “next” ” finish” 버튼을 클릭 아래 화면이 나옵니다.

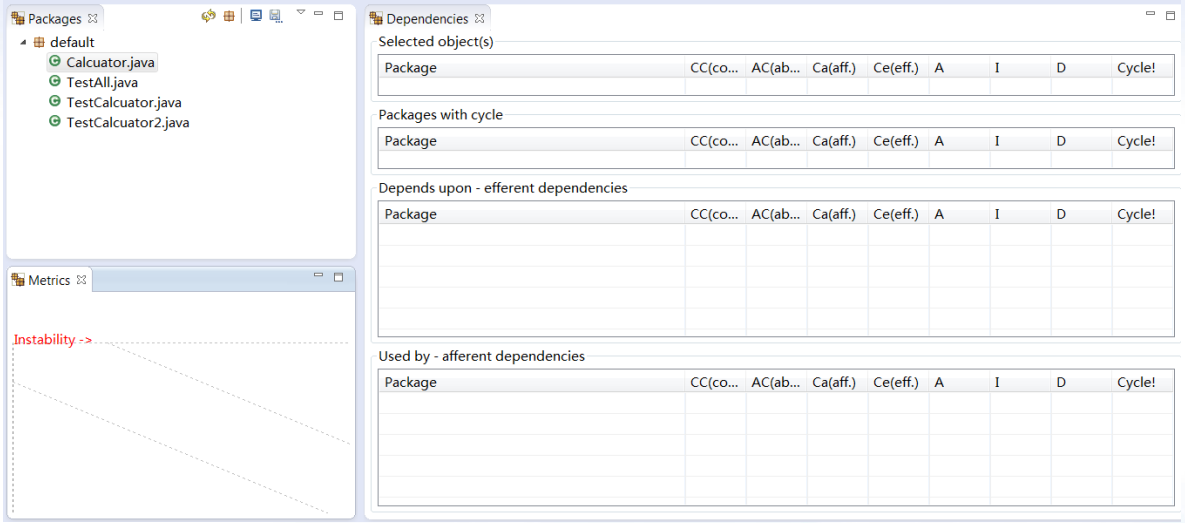


설치방법step5:

원하는 패키지에다가 Run Jdepend analysis 를클릭.



Packages

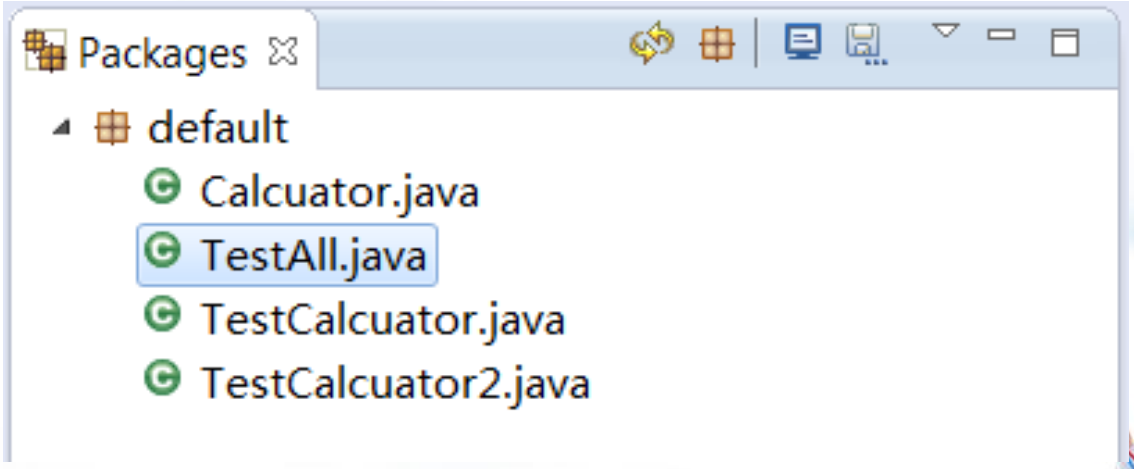


Metrics

Dependencies

Packages

패키지 의존성 정보
를 확인할 수 있음



Dependencies View

선택한 패키지의 의
존성 정보를 수치화
하여 표현

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
junit.framework	0	0	1	0	0.00	0.00	1.00	
junit.textui	0	0	1	0	0.00	0.00	1.00	

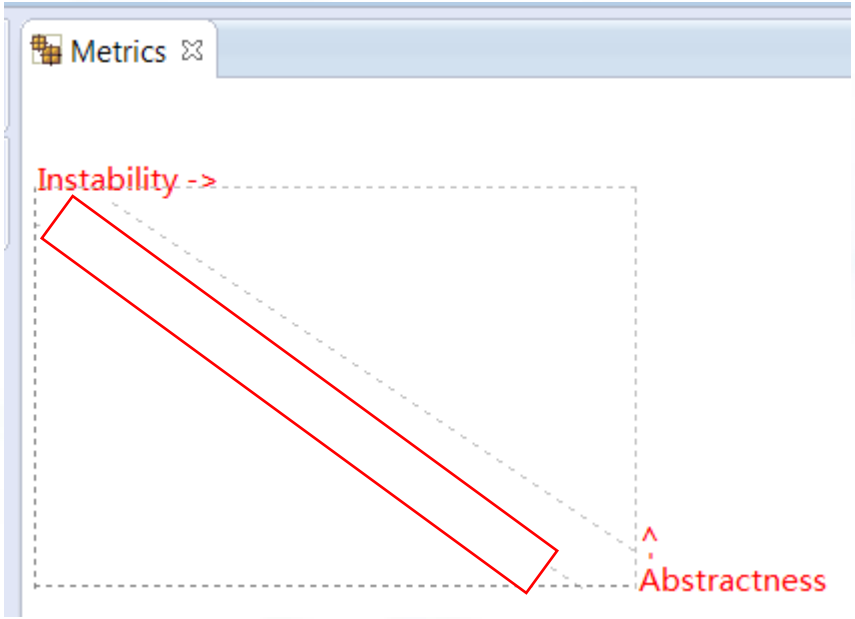
Dependencies View

수치에 대한 정의

- CC : Interface, Abstract Class를 제외한 Concrete class의 갯수.
- AC :추상클래스나 Interface의 갯수. 확장성의 척도.
- Ca :현재 패키지의 클래스에 의존하고 있는 패키지의 수. 책임의 척도가 됨.
- Ce:현재 패키지의 클래스들이 의존하고 있는 패키지의 수. 독립성의 척도가됨.
- A : 추상화 정도를 나타내며, 0~1 사이의 값을 가짐. 0은 완전 구체적인 패키지,1은 추상적인 패키지를 나타낸다.
- I: 변화에 대한 안정성을 나타내며, 0~1 사이의 값을 가짐. 0은 외부 변화에도 끄떡없는 패키지이며, 0은 작은변화에도 쉽게 흔들릴 수 있는 패키지를 나타낸다.
- D: Main Sequence로부터의 거리를 나타내며, 0은 Main Sequence와 가깝고, 1은 먼 상태.
- Cycle: 패키지들 상호 간에 의존성을 가지고 있을 때 발생 안 좋은 상황이기 때문에 경고 아이콘으로 보여짐

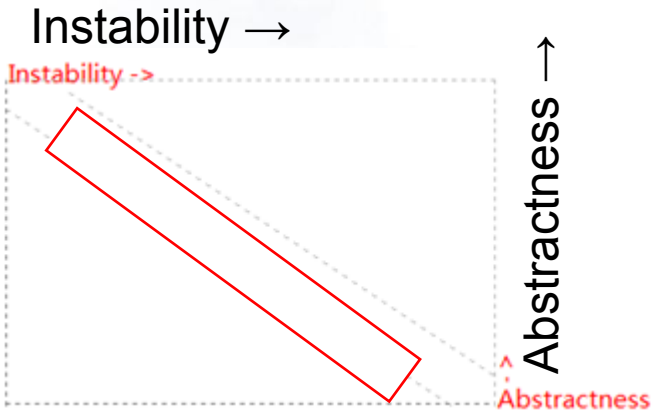
Metrics View

Dependencies 에서
수치화된 정보를 그
래프로 보여줌



Very High Stability Interface Packages

이상적인Package로 추상화가 잘 이루어져있으며 안정적인 Package를 의미한다.



Dependencies에서 수치화된 정보를 Console에서 텍스트로 확인 가능

Dependencies Console

JDepend

```
---------------------------------------------------
- Package: junit.textui
---------------------------------------------------
No stats available: package referenced, but not analyzed.


---------------------------------------------------
- Package Dependency Cycles:
---------------------------------------------------


---------------------------------------------------
- Summary:
---------------------------------------------------

Name, Class Count, Abstract Class Count, Ca, Ce, A, I, D, V:

Default,4,0,0,2,0,1,0,1
junit.framework,0,0,1,0,0,0,1,1
junit.textui,0,0,1,0,0,0,1,1
```

Depend 알게된 것

현재 패키지의 의존성 문제점을 찾기 쉽다.

개발중에도 문제점을 체크할 수 있다.

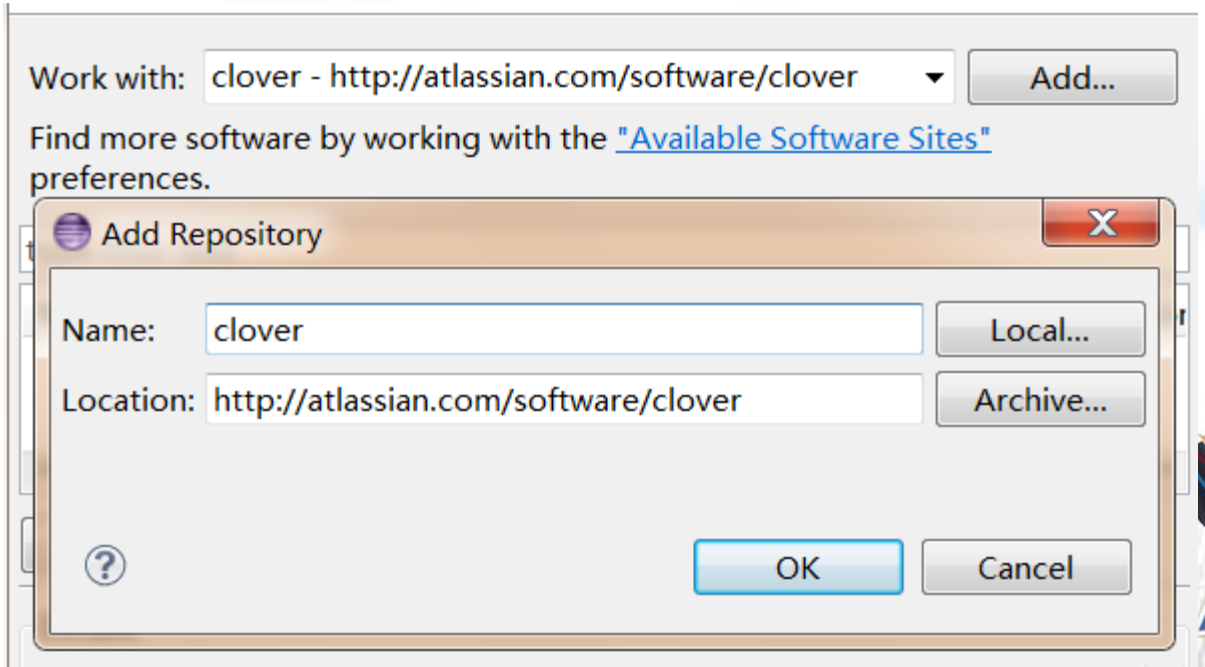
Clover

Clover필요하는 이유

- 테스트에 의해 프로그램이 어떻게 작동하는지 확인
- 전체 테스트 부분 중 얼만큼 완료됐는지 판단
- 결과에 따라 테스트 기능 개량 가능
- 검사 항목, 기능 추가 여부 결정

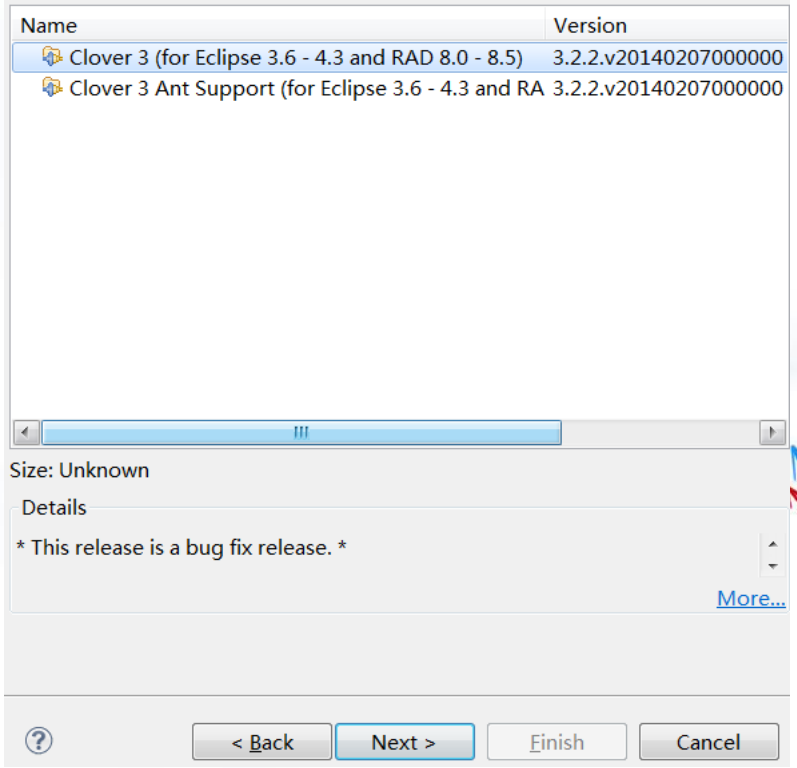
Clover 설치 방법 step1

Eclipse에서help->Install new software선택
주소:
http://update.atlassian.com/eclipse/clover

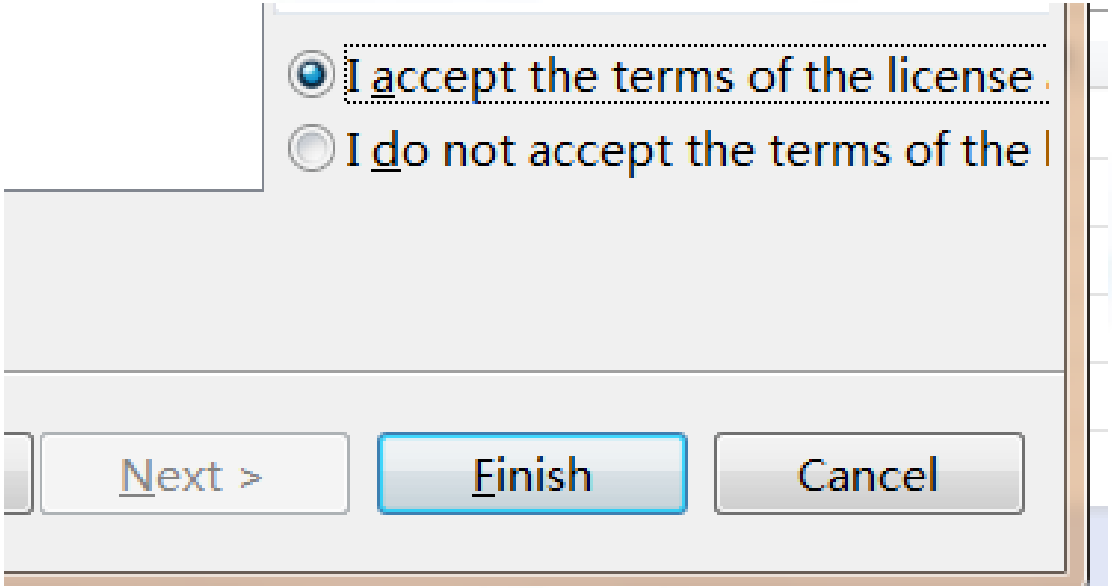


Clover 설치 방법 step2

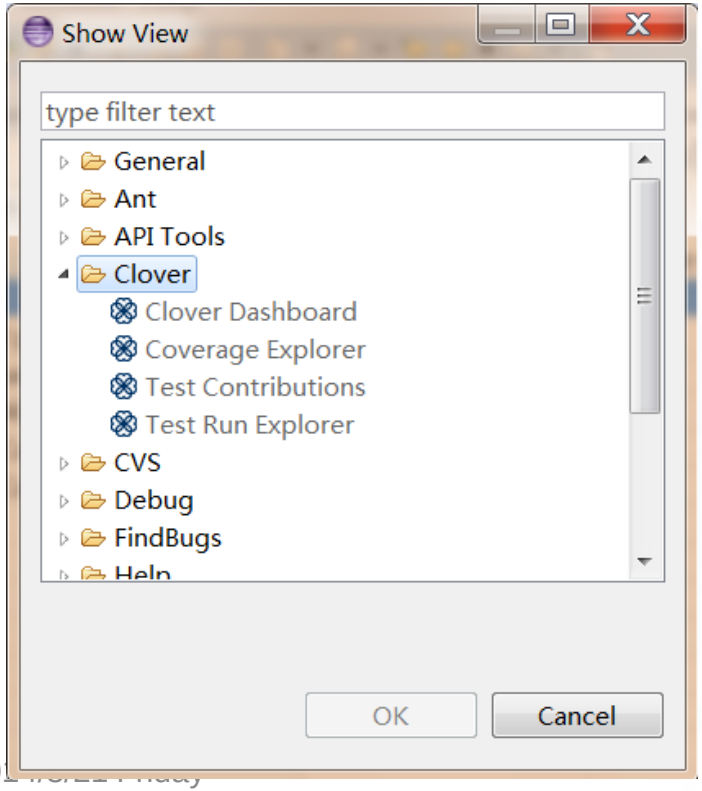
next…



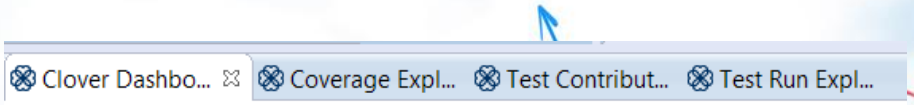
Clover 설치 방법 step3
finish Click



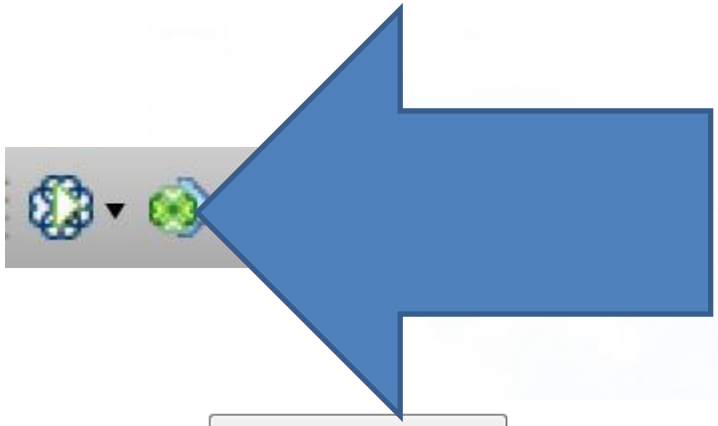
Eclipse menu에서Window->Show View->Other



Clover 폴더에 있는모든Clover Interface를 선택하고 아래와 같이 생기면OK..!



Coverage explorer



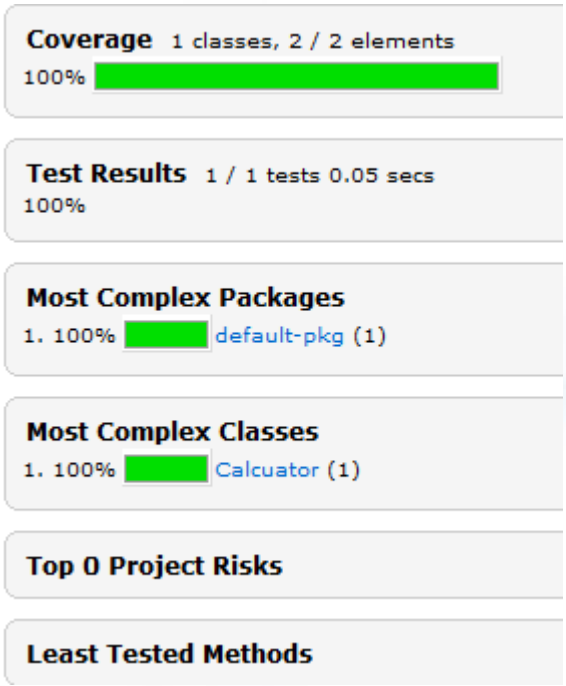
UnitTest를시작한다. 그럼 아래와 같이 테스트 코드가 얼마나Cover됐는지 Percentage로 나타난다.

Show: All classes

Elem	Cov%	Av Me Cpx	Cpx
Calcuator	35.3%	1.0	5.0
(default package)	35.3%	1.0	5.0
Calcuator.java	100.0%	1.0	1.0
TestAll.java	0.0%	1.0	2.0
TestCalcuator.java	0.0%	1.0	1.0
TestCalcuator2.java	100.0%	1.0	1.0

Coverage Dashboard

Coverage
Test Coverage의 비율에 대한 정보
Test Results
테스트 결과에 대한 정보
Most Complex Packages
가장 복잡성을 가진 Package를 보여준다
Most Complex Classes
가장 복잡성을 가진 Class를 보여준다
Top Project Risks
가장 복잡하고 적은 Test Coverage를 가진 Class를 보여준다
Least Tested Methods
가장 낮은 Test Coverage의 Methods을 보여준다



Java Editor

초록색
Test를 합격하거나 Main Method처럼 외부 Test를 하는 라인
노란색
실패한 Test Coverage
빨간색
Coverage가 없는 Code
회색
제외된 코드

```
public class Calcuator {
    public double add(double n1, double n2) {
        return n1 + n2;
    }
}
```

```
public static Test suite() {
    TestSuite suite = new TestSuite("TestSuite T
    suite.addTestSuite(TestCalcuator.class);
    suite.addTestSuite(TestCalcuator2.class);
    return suite;
```

```
import junit.framework.TestSuite;
```

Coverage Treemap Report

- Package별로 출력을 해준다.
- 각사각형의 크기는complexity함을 보여줌
- 클래스의label을 출력하지 않는다.

Coverage Treemap Report

◆복잡성과Coverage된 Project 또는 Package를 쉽게 볼수있다.

◆Package(labeled)로 나누고 그안에class(unlabeled)로 보여준다.

◆Package 또는Class의 크기는 복잡성을 나타낸다.

◆색깔로codecoverage의 정도를 나타낸다.

Thank you!

The end