Software Modeling & Analysis

Introduction to

SVN, Mantis, JFeature, Junit

사용법 및 CTIP 개론

Team No	Team2
과목	Software Modeling and Analysis
담당교수	JUNBEOM YOO Associate Professor / Ph.D
팀 구성원	201014184 김도윤
	201111367 여승훈
	201111347 김태호
제출일자	2016-04-06

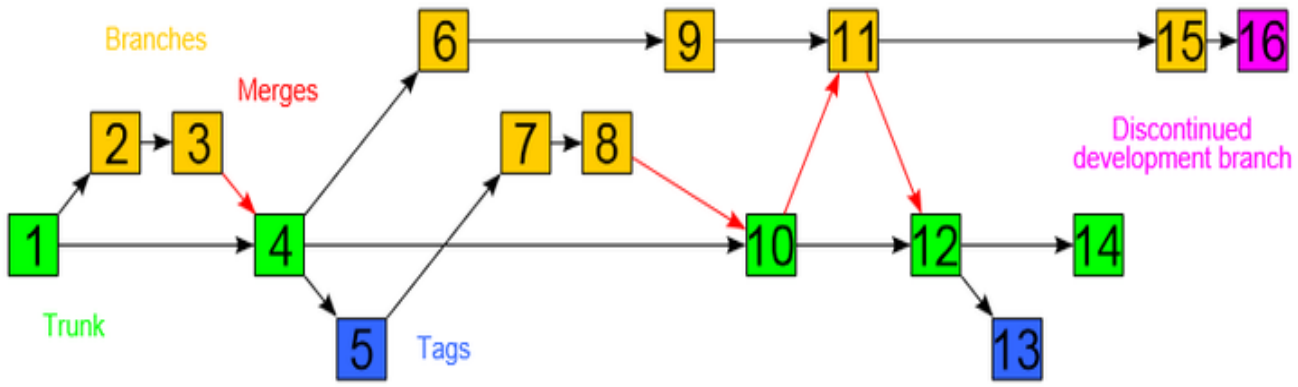
Index

1 SVN 3
1.1 SVN 관련 용어 3
1.2 SVN 명령어 4
1.3 Eclipse 플러그인 설치 5
1.4 SVN 사용하기 9
1.5 Check Out 11
1.6 Commit 12
1.7 Update 14
2 Mantis 15
2.1 Mantis 장점 15
2.2 Mantis 사용 방법 15
3 JFeature 19
3.1 주요 특징 19
3.2 JFeature 설치 19
3.3 JFeature 사용법 20
4 Junit 24
4.1 Eclipse 에서 Junit 라이브러리 추가하기 24
4.2 JUnit 으로 단위테스트를 하는 방법 26
4.3 사용자가 직접 테스트 클래스를 만드는 방법 → TestCase Class 를 상속받는 방법 26
4.4 JUnit 프레임워크의 유용한 클래스들 ➔ Assert Class 27
4.5 Assert 클래스의 Method 를 사용하는 방법 28
5 CTIP (Continuous integration + Continuous Test , Continuous Test & Integration Platform) 29
5.1 CTIP 구성 요소 30
5.2 CTIP Tool 30
5.3 성공적인 CI 수행 조건 30
5.4 CTIP 의 장점 31
5.5 CTIP 흐름도 31

1 SVN

- SubVersioN 의 약자로 자유 소프트웨어 버전 관리 시스템
- 개발자들의 소스 관리를 편리하게 하기 위한 툴
- 아파치 라이선스
- 처음 파일을 저장시킬 때는 파일 원본 그대로 저장을 시키고 그 다음부터는 실제 파일을 저장하는 형식이 아닌 그 파일과의 차이점을 저장시킨다.
- 언제든 예전의 형태로 돌아갈 수 있다.
- 전체적으로 서버/클라이언트 모델을 따른다.

1.1 SVN 관련 용어



1.1.1 Repository(저장소)

- 모든 프로젝트의 프로그램 소스들이 저장되는 저장소
- 소스뿐 아니라 소스의 변경 사항도 모두 저장된다.
- 네트워크를 통해 여러 사람이 접근 가능하다.

1.1.2 Revision

- 프로젝트 진행 상황 파악
- 소스 파일을 수정하여 commit 하면 일정 규칙에 의해 숫자가 증가
- SubVersioN 의 경우 파일 별로 revision 이 매겨지지 않고 한번의 commit 으로 전체 revision 이 매겨진다.

1.1.3 Trunk

- 중간 작업이 완료된 파일이 존재

1.1.4 Branch

- 현재 작업중인 파일들이 존재

1.1.5 Tags

- 중간중간 백업 중간 완성본 혹은 최종 완성본이 존재

1.2 SVN 명령어

1.2.1 Import

- 파일이나 폴더를 저장소에 최초로 추가하는 명령어
- 새로운 파일이나 폴더를 추가할 때 사용하는 용어
- commit 을 하지 않으면 적용되지 않는다.

1.2.2 Commit

- 파일을 저장소에 업로드 하는 명령어
- commit 을 할 때마다 revision number 가 올라간다.
- 웹 하드에 commit 을 할 때마다 자동적으로 백업이 된다.

1.2.3 Commit Log

- commit 을 할 때마다의 log 를 작성
- log 란 현재까지 작업의 진행상태를 글로 적는 명령어

1.2.4 CheckOut

- 웹 하드로부터 데이터를 다운 받을 때에 하는 명령어

1.2.5 Revert

- 최근 revision 으로 파일을 복구하는 명령어

1.2.6 Export

- CheckOut 과는 달리 version 관리 파일을 뺀 순수 소스만을 가져오는 명령어

1.2.7 Update

- CheckOut 해서 받은 소스를 최신의 소스로 업데이트 하는 명령어

1.2.8 Diff

- 예전 소스 파일과 지금의 소스 파일을 비교하는 명령어

1.2.9 Lock

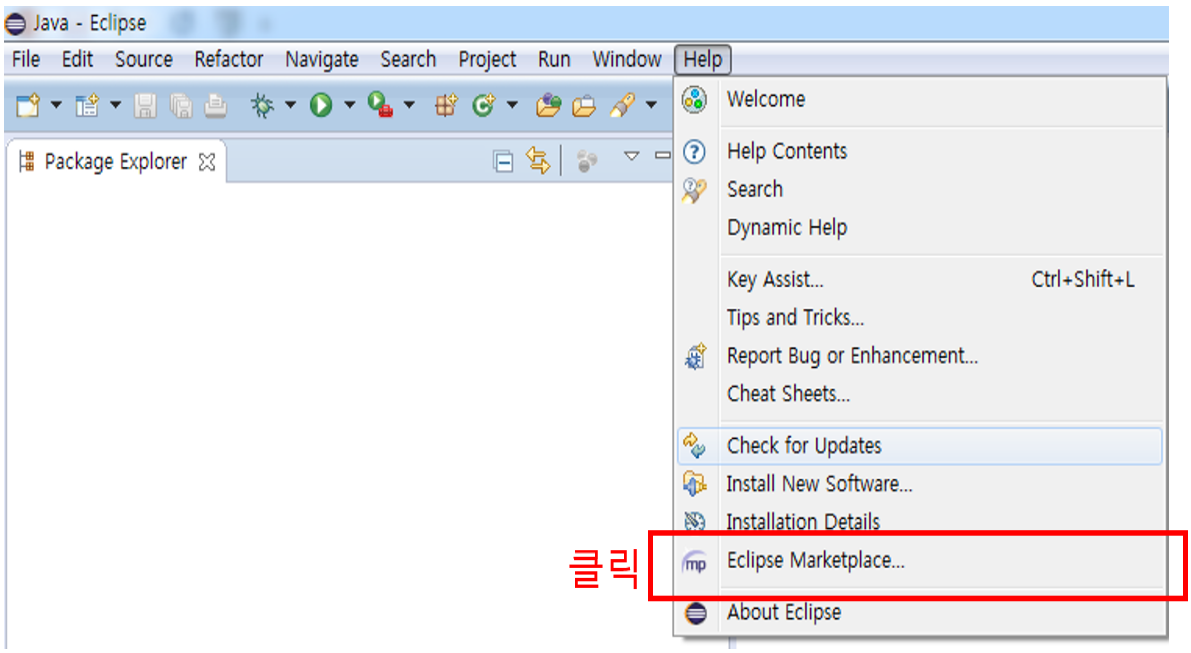
- 파일에 Lock 을 걸어 Lock 을 건 사용자만이 수정 가능하게 하는 명령어

1.2.10 Add

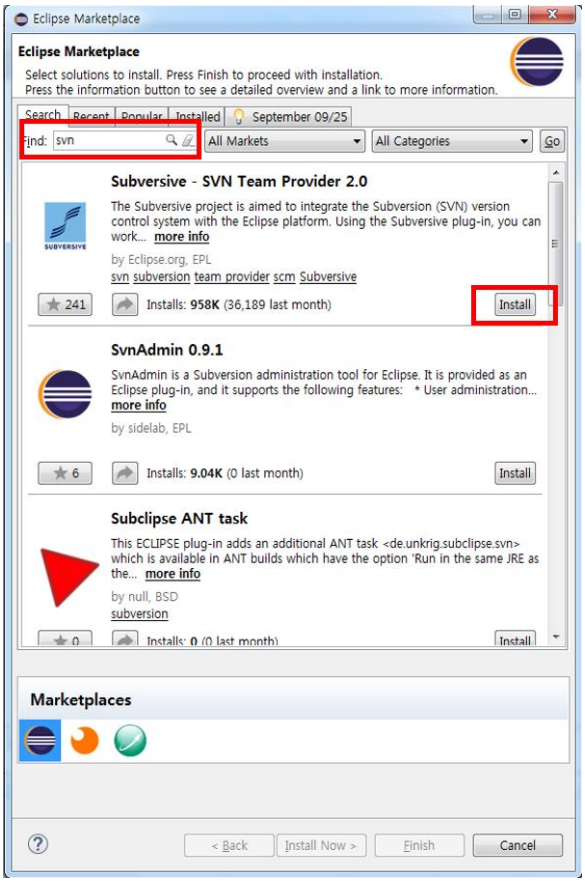
- 새 파일을 만들었을 경우 파일을 추가하는 명령어

1.3 Eclipse 플러그인 설치

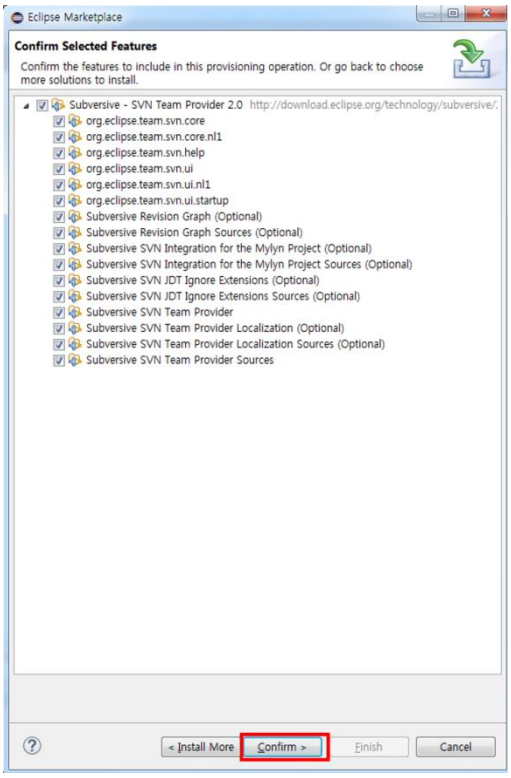
1.3.1 Eclipse 에서 Help > Eclipse > Marketplace 를 선택한다.

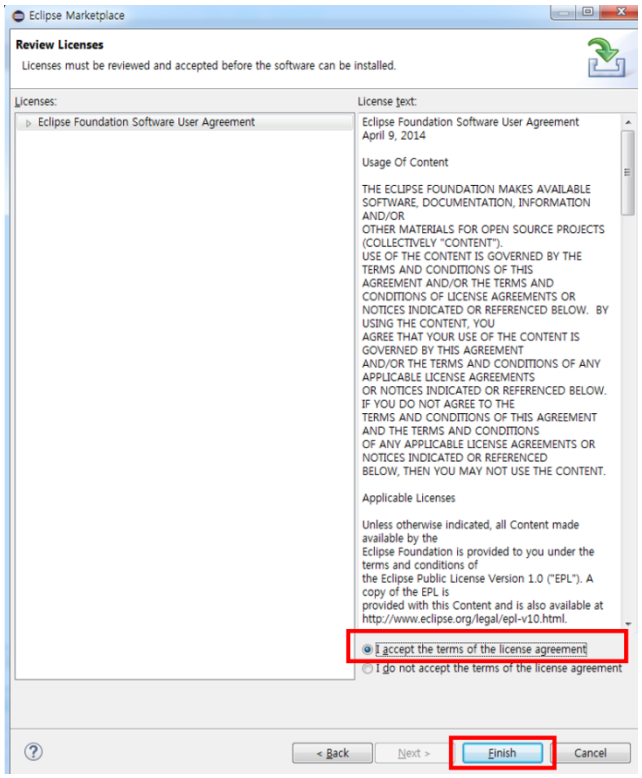


1.3.2 Eclipse Marketplace 에서 "SVN"을 검색하여, "Subversive" 플러그인을 설치한다.

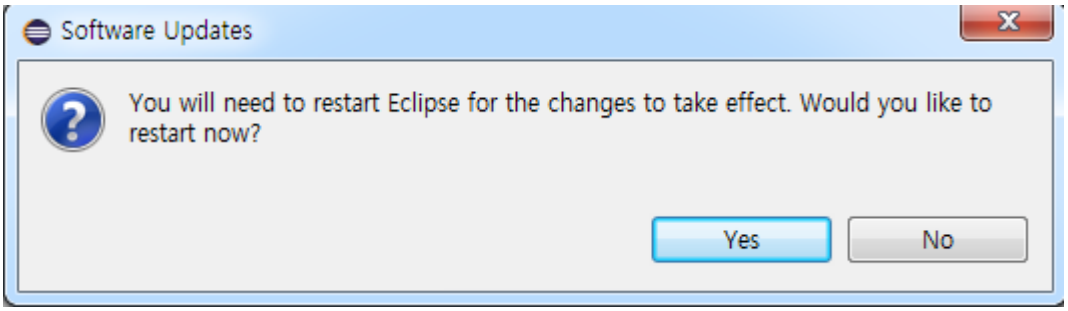


1.3.3 아래의 흐름을 따라 한다.

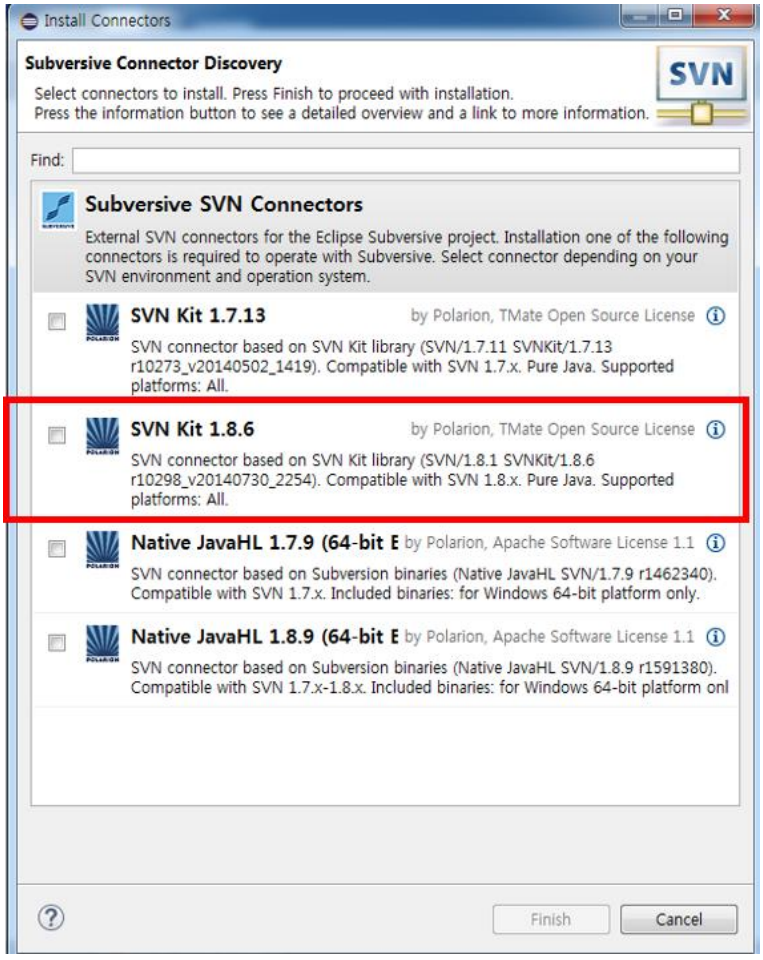




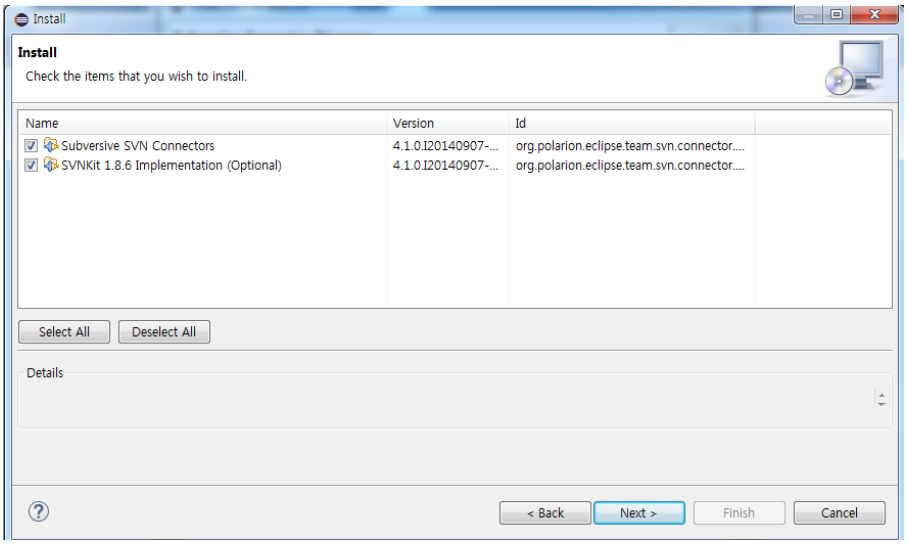
1.3.4 플러그인 설치 완료 시, 변경사항을 적용하기 위해 Eclipse 를 재 시작한다.

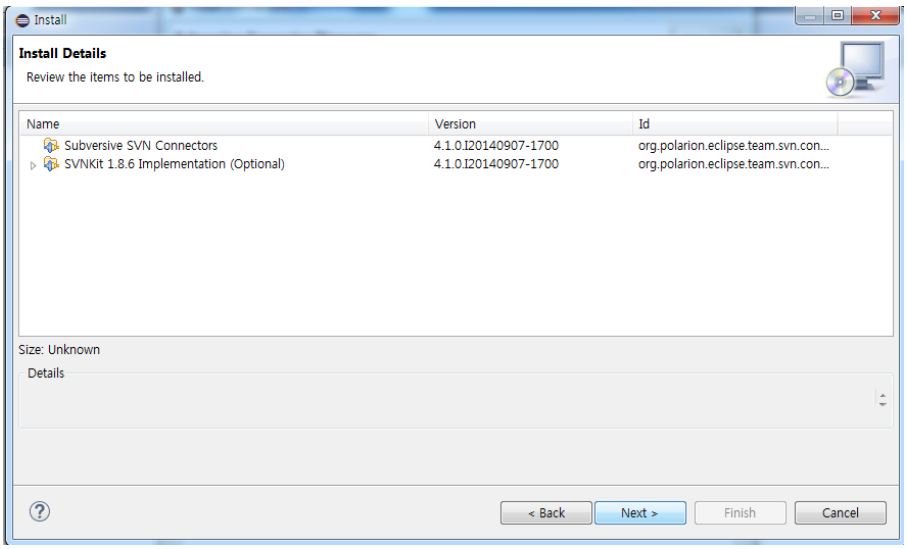


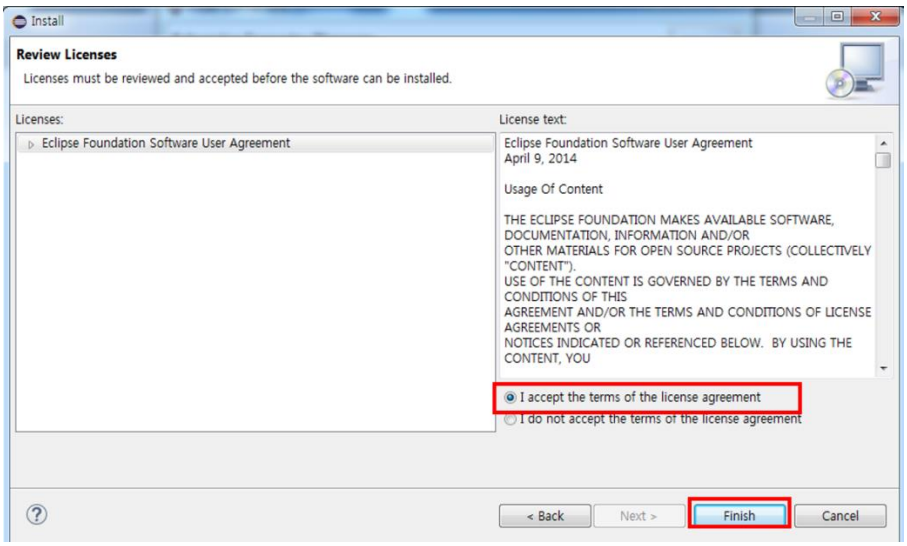
1.3.5 Eclipse 를 재 시작 한 후 아래와 같은 창이 뜨며 "SVN Kit 1.8.6"을 설치한다.



1.3.6 아래의 흐름을 따라 한다.





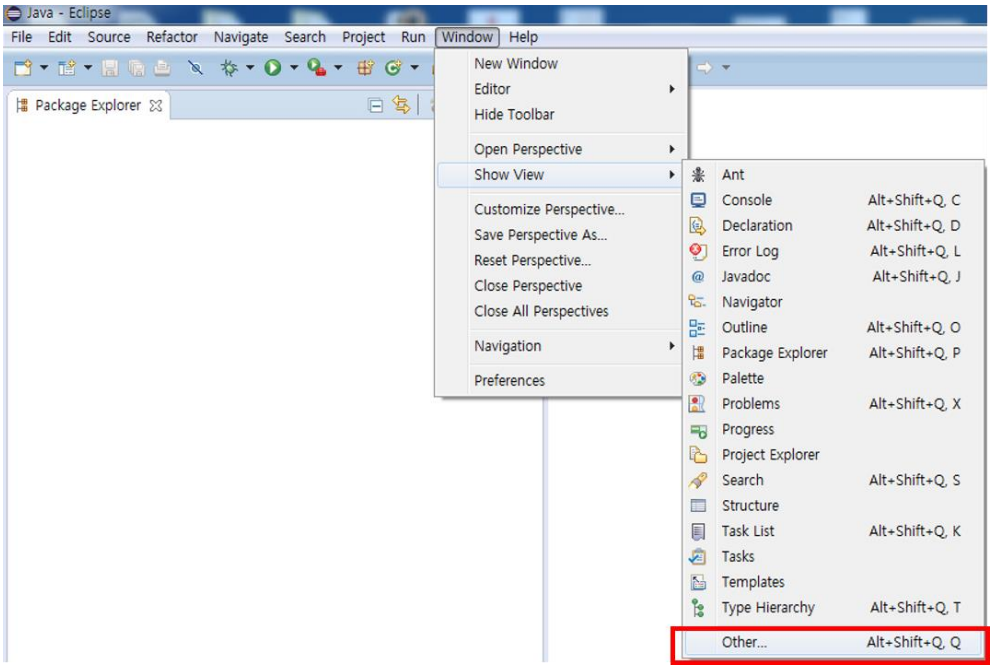


1.3.7 Eclipse 를 재 시작한다.

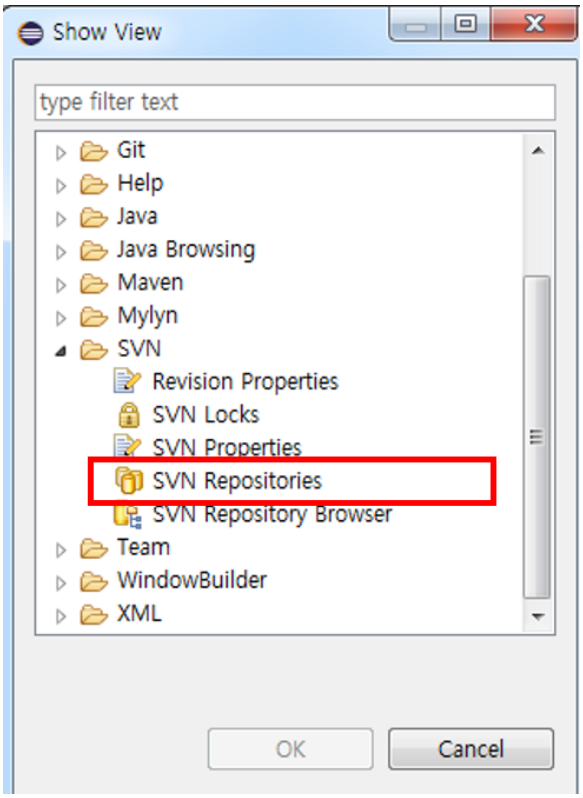
1.4 SVN 사용하기

1.4.1 SVN Repository 를 볼 수 있도록 창을 등록한다.

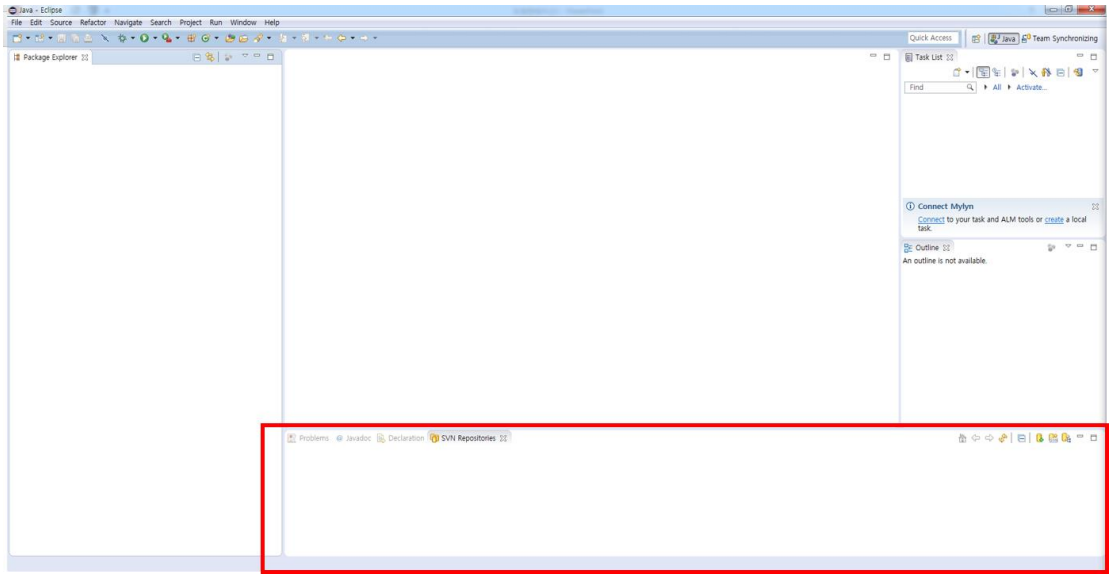
1.4.2 Window > Show View > Other 를 클릭한다.



1.4.3 SVN > SVN Repositories 를 선택한다.

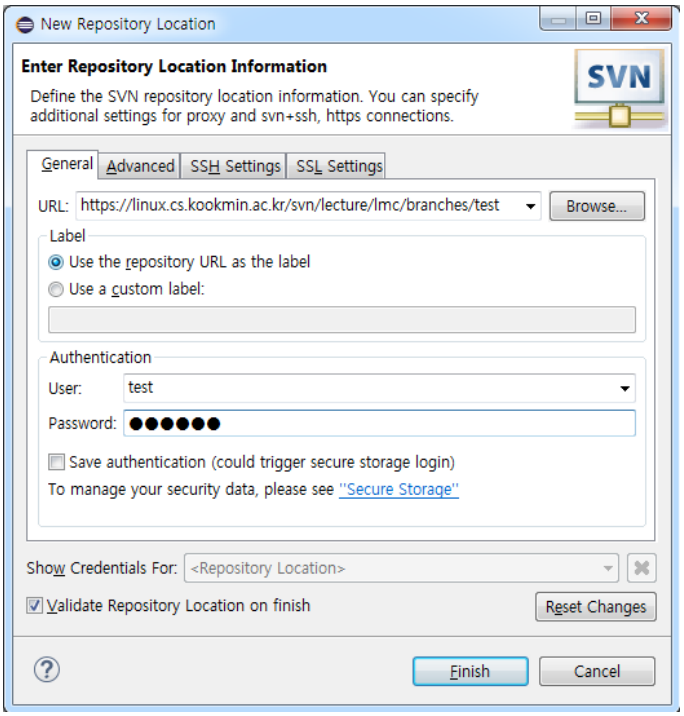


1.4.4 Eclipse 에서 "SVN Repositories" 창이 등록된 것을 확인한다.



1.4.5 "SVN Repositories" 창에서 오른쪽 버튼을 눌러 New > Repository 를 통해 새로운 저장소를 등록하거나, 아래와 같이 버튼을 클릭하여 새로운 저장소를 등록한다.

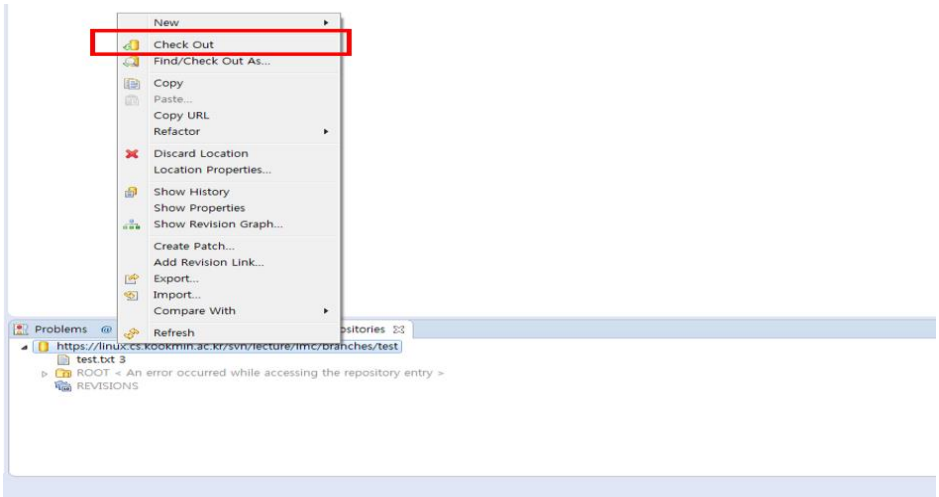
1.4.6 새로운 저장소를 등록하기 위해 저장소의 RUL 과 본인의 계정 및 비밀번호를 입력한다.



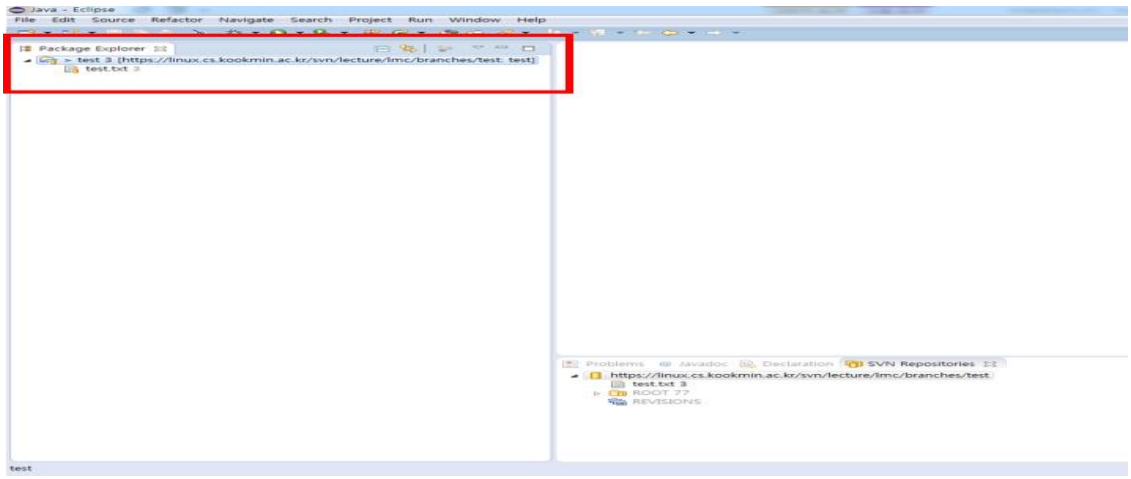
1.4.7 새로운 저장소가 등록되면 아래와 같은 화면을 확인할 수 있다.

1.5 Check Out

1.5.1 등록된 저장소에서 오른쪽 클릭을 통해 Check Out 을 하여 소스를 다운받는다.

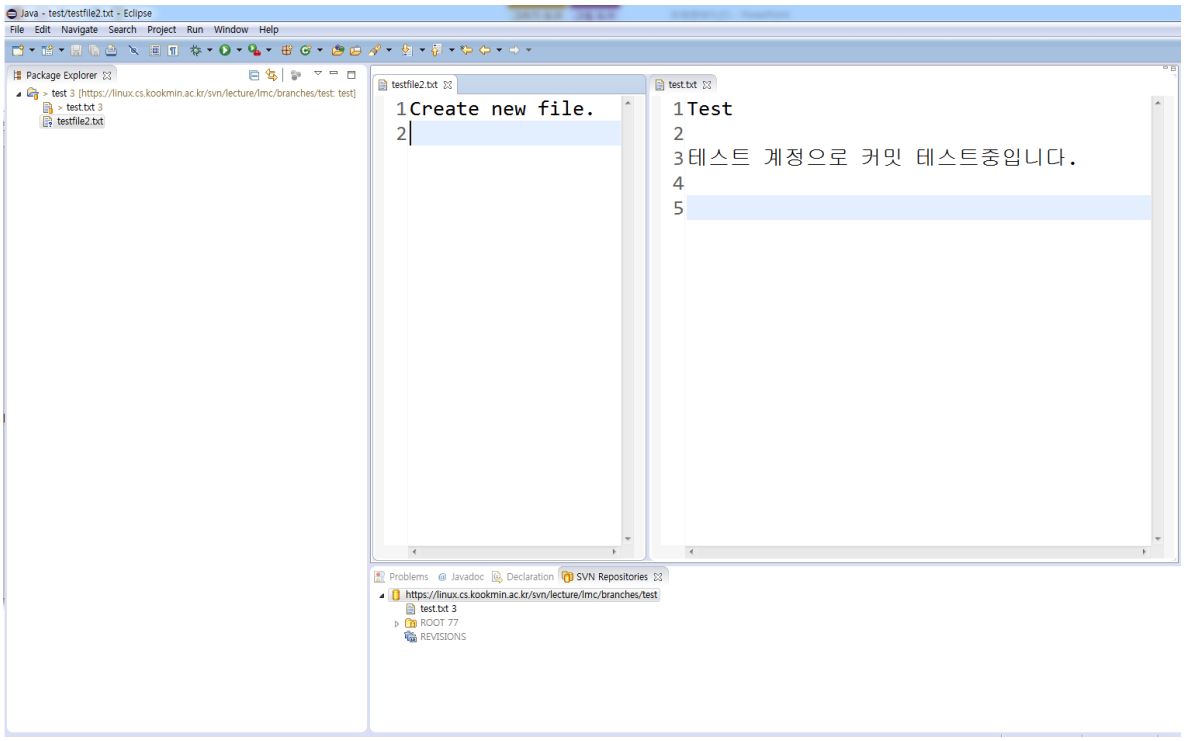


1.5.2 Check Out 이 완료되면 아래 그림과 같이 소스코드를 확인할 수 있으며, 로컬에서 소스 편집이 가능하다.

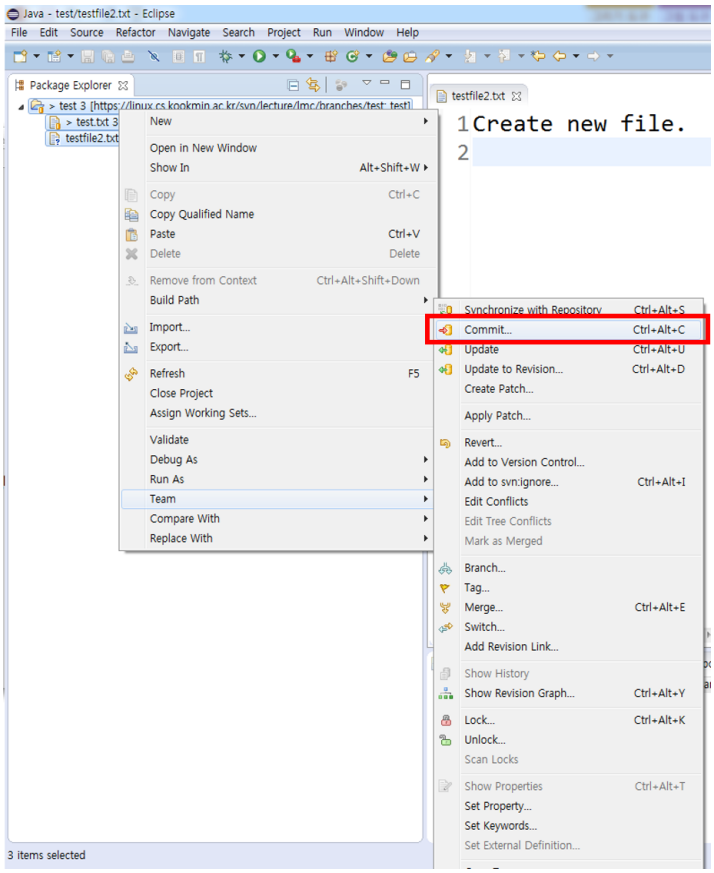


1.6 Commit

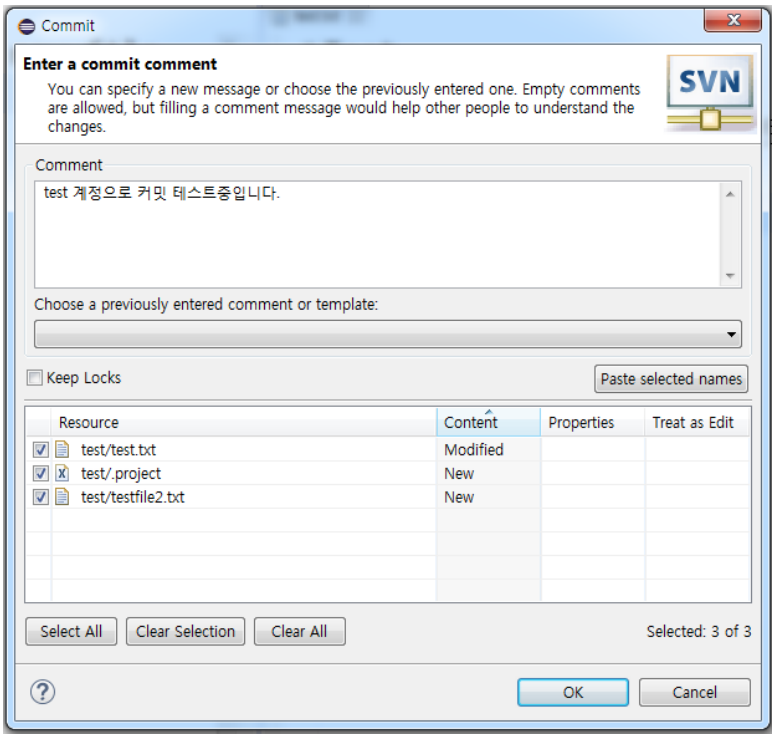
1.6.1 내려 받은 소스에서 파일을 생성 및 편집을 한다.



1.6.2 프로젝트를 오른쪽 클릭하여 아래와 같이 Team > Commit 메뉴를 선택한다.

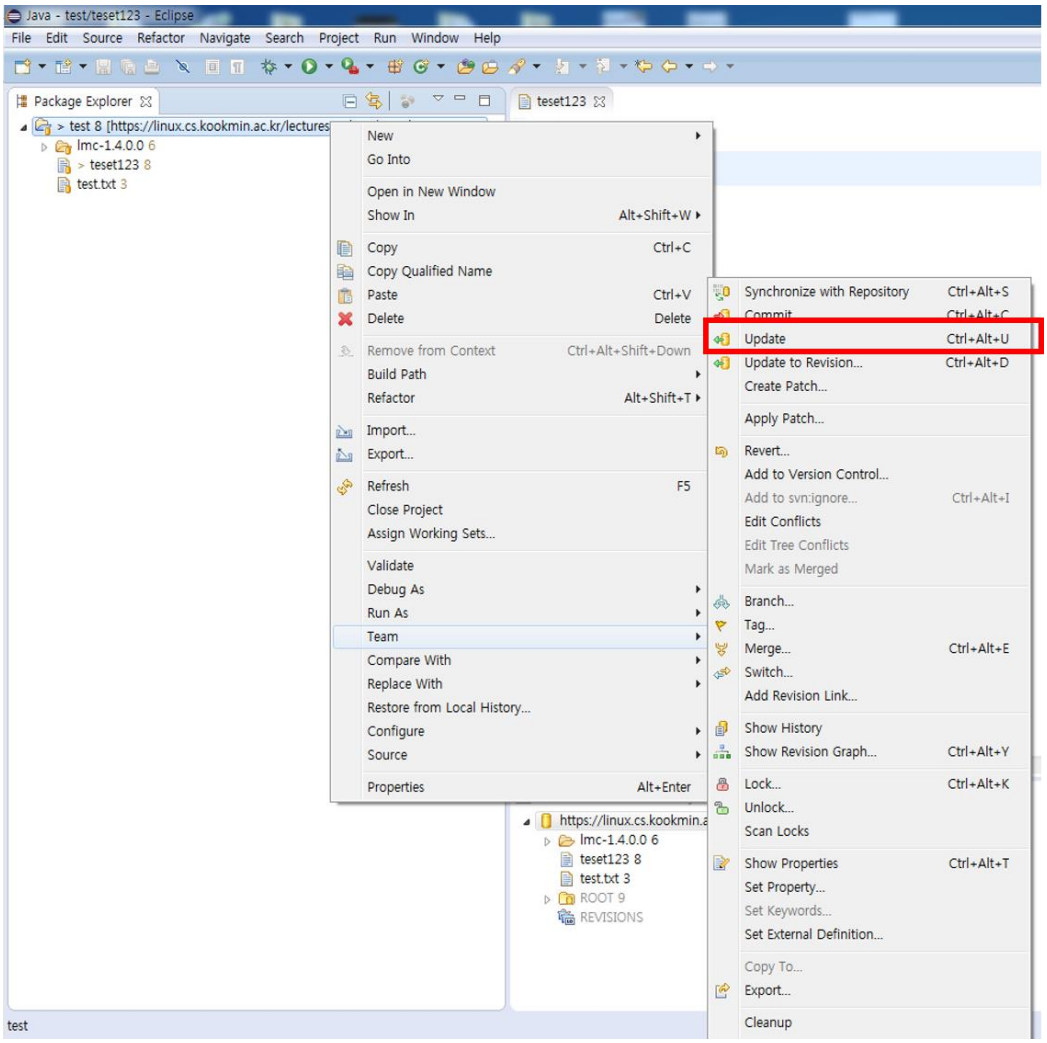


1.6.3 Commit 을 할 때 반드시 Comment 를 작성하도록 되어 있다. 버전마다 구별할 수 있도록 Comment 를 작성한다.



1.7 Update

- Update 는 현재 작업 디렉토리의 내용을 저장소의 최신 revision 으로, 또는 지정 revision 으로 갱신하는 행위이다.
- Commit 을 하기 전 또는 작업을 하기 전에 원하는 revision 으로 갱신을 한 후 소스를 편집하고 Commit 을 해야 한다.
- 따라서 Update 한 후 Commit 을 버릇처럼 수행해야 한다.
- 아래 그림과 같이 Update 메뉴를 선택하면 최신의 revision 으로 갱신된다. 만약 특정 revision 으로 갱신하고 싶다면 Update to Revision 메뉴를 선택한다.



2 Mantis

- Bug Tracking System 소프트웨어의 품질 보증 및 프로그래머가 소프트웨어 버그를 tracking 할 수 있도록 도와주는 시스템이다.
- Open Source r 기반의 Bug Tracking System
- PHP 로 만들어졌으며, Mysql DB 와 웹 서버를 필요로 한다.
- 윈도우, OS/2, UNIX 기반의 다양한 시스템에 실행할 수 있다.
- GNU(General Public License) 기반이다

2.1 Mantis 장점

- 무료로 사용 가능하다.
- 설치가 간단하다.
- Bug 에 대한 History 가 저장된다.
- E-mail 로 알림이 가능하다.
- SVN 등 Source Version 관리 시스템과 연동이 가능하다.

2.2 Mantis 사용 방법

2.2.1 로그인

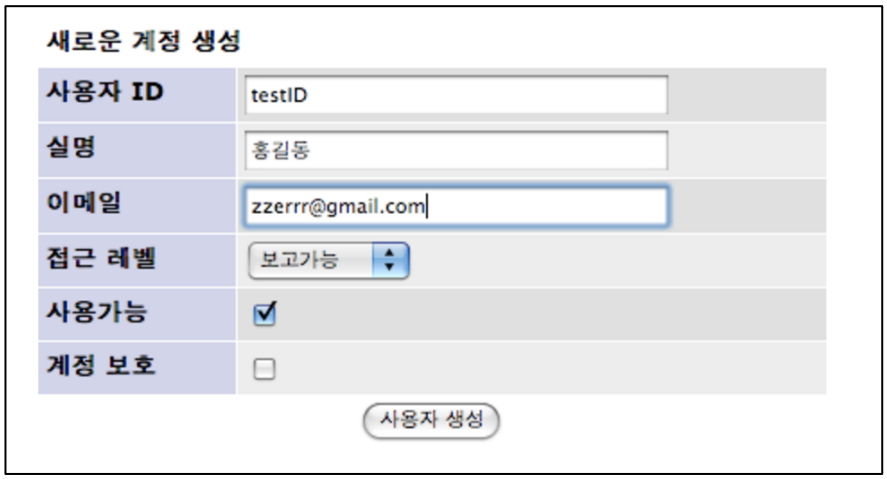
최초 시작 시 다음과 같은 두 개의 에러 메시지가 나타난다.

첫 번째는 administer 비밀 번호를 변경하라는 뜻이며 두 번째는 Mantis 설치 경로에서 Admin 폴더를 삭제하라는 메시지이다. 해당 에러는 해결하기 위해서 설치 폴더로 돌아가 Admin 폴더를 삭제한다. 그리고 Administrator/root 를 입력하고 로그인을 한다. 그리고 Administrator 계정의 패스워드를 수정하면 Mantis 의 모든 설치과정을 마치게 된다.

2.2.2 계정 만들기

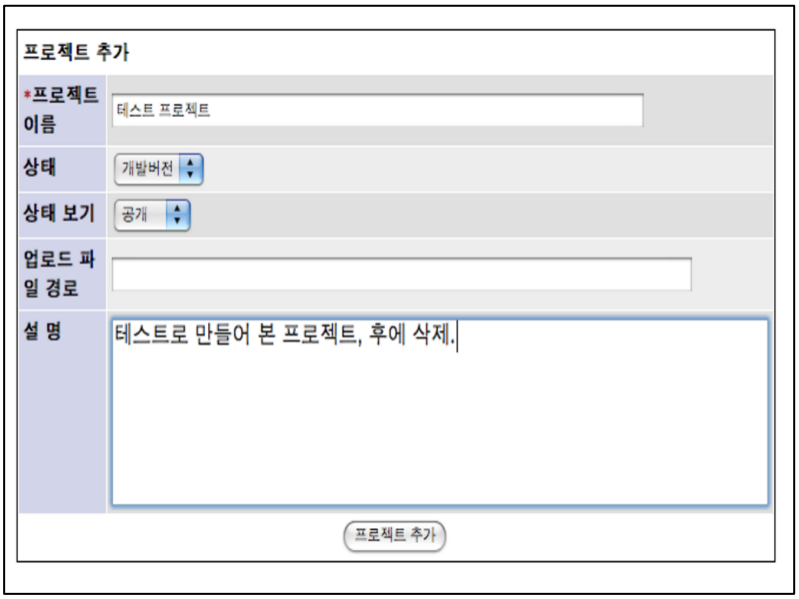
'관리/사용자 관리' 메뉴로 들어가 '계정 생성' 버튼을 클릭하여 사용자를 추가한다. 계정 생성 후에 상세한 옵션이나 권한을 설정할 수 있다.

e-mail 로 계정 등록이 발생되기 때문에 e-mail 주소를 정확히 입력해야 한다.



2.2.3 프로젝트 생성

'관리/프로젝트 관리' 메뉴로 들어가 새로운 프로젝트를 생성한다.

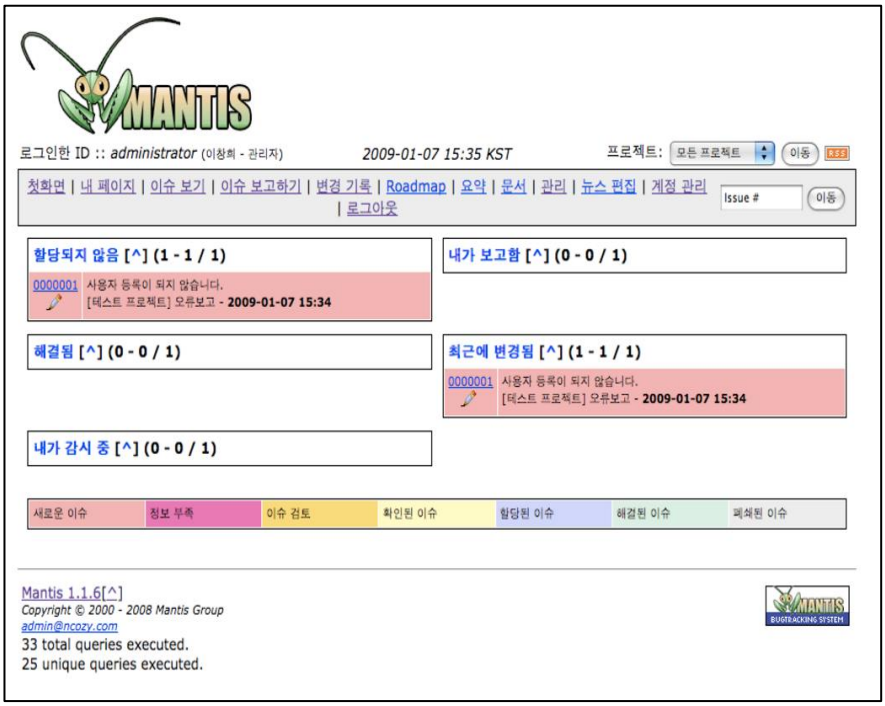


2.2.4 이슈 등록 및 확인

사용자 계정(testID)로 로그인하여 '이슈 보고하기' 메뉴를 클릭한다. 해당 프로젝트를 선택하고 버튼을 클릭하여 필드를 입력하고 등록한다.

2.2.5 관리자 확인

다시 관리자로 로그인하면 메인 화면에서 새로운 이벤트를 확인할 수 있다. 할당되지 않은 이슈에는 담당자를 할당하거나 바로 처리할 수 있다.

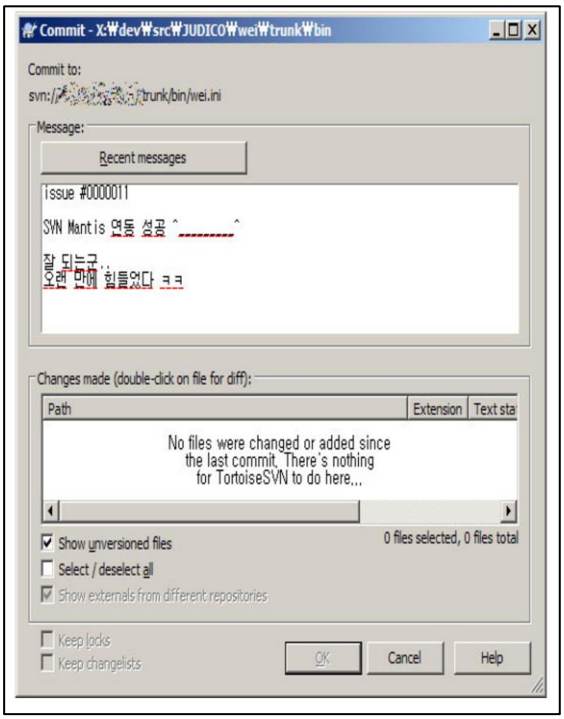


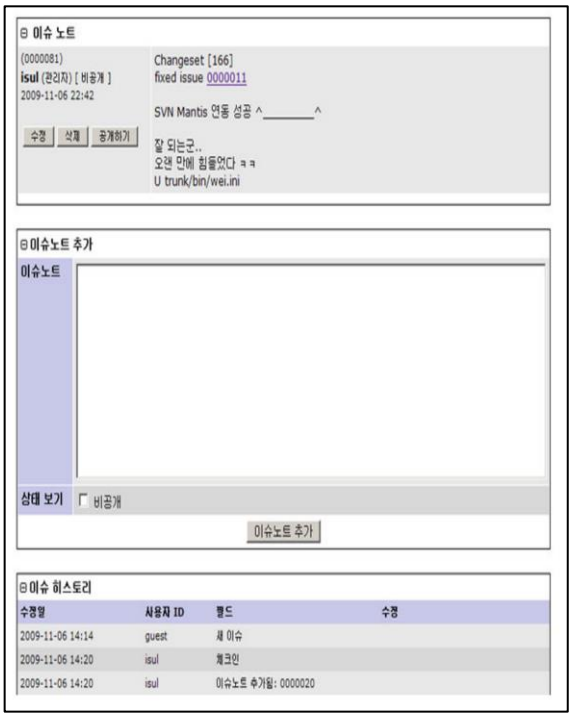
2.2.6 Mantis Access-Level

- Viewer : 버그와 버그리스트를 볼 수 있다. (ex. guest)
- Reporter : 새로운 버그를 보고할 수 있다.
- Updater : 리포터와 권한이 같지만, 버그를 업데이트 할 수 있다.
- Developer : Update 보다 더 많은 권한을 가지고 있고, 실제 프로젝트에 대한 개발이 가능하다.
- Manager : Developer 보다 더 많은 권한을 가지고 있고, 관리하고 있는 프로젝트들에 대해서 모든 권한을 가지고 있다.
- Administrator : 모든 권한을 가지고 있다. 계정 생성, 비밀번호 초기화, 유저 추가 삭제 등

2.2.7 SVN + Mantis 연동하기

SVN 클라이언트에서 Commit Log 를 입력하여 Commit 하면 Mantis 의 이슈 노트와 history 에 그 내용이 자동으로 기록된다.





3 JFeature

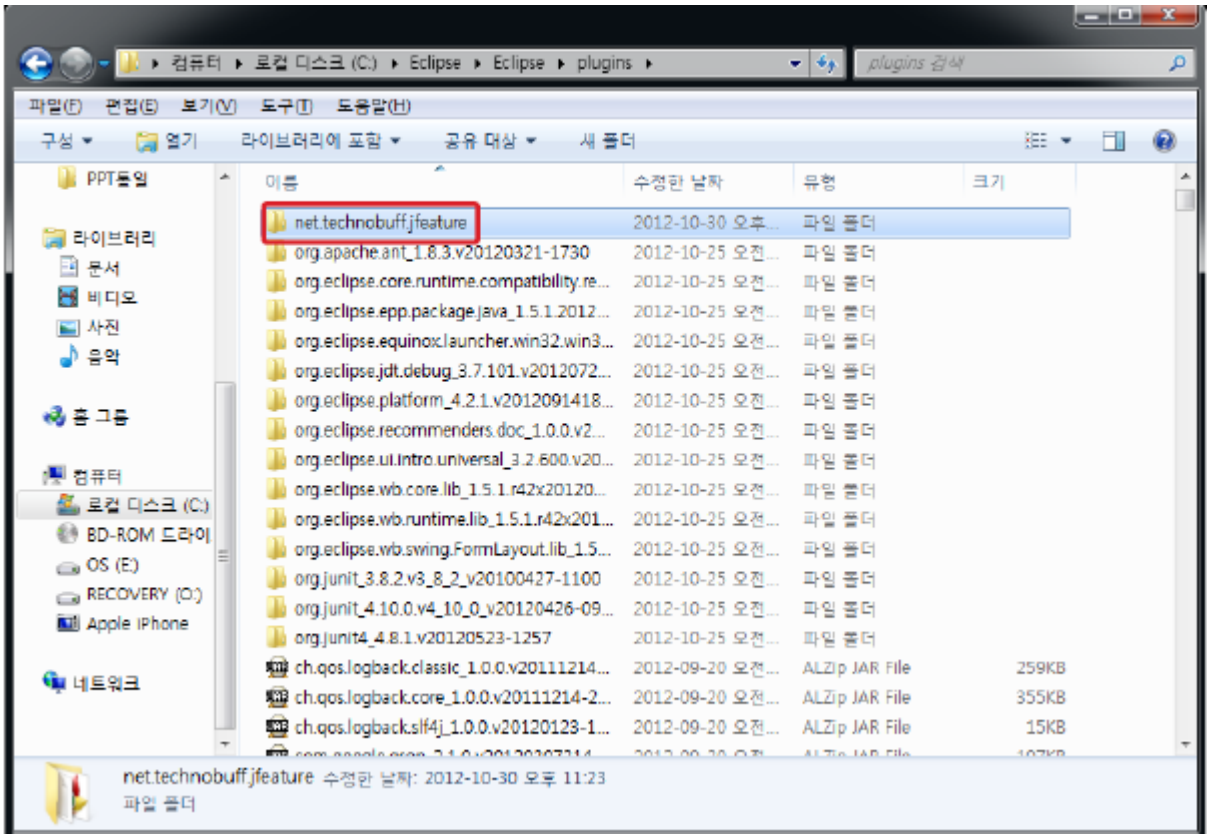
- 개발하는 코드에 해당하는 요구사항에 초점
- IDE 환경에 요구사항을 작성하거나 불러온다
- 개발된 테스트 코드와 요구사항을 매칭시킨다.

3.1 주요 특징

- Junit 의 Testcase Method 와 통합할 수 있다.
- 요구사항의 수정사항이 생겼을 때, 즉시 view 를 제공하므로 빠른 편집이 가능하다.
- 요구사항의 반영 여부를 Coverage 형태로 쉽게 확인할 수 있다.
- 각 요구사항의 priority, dependency 등을 연관하여 Unit Test 를 진행할 수 있다.

3.2 JFeature 설치

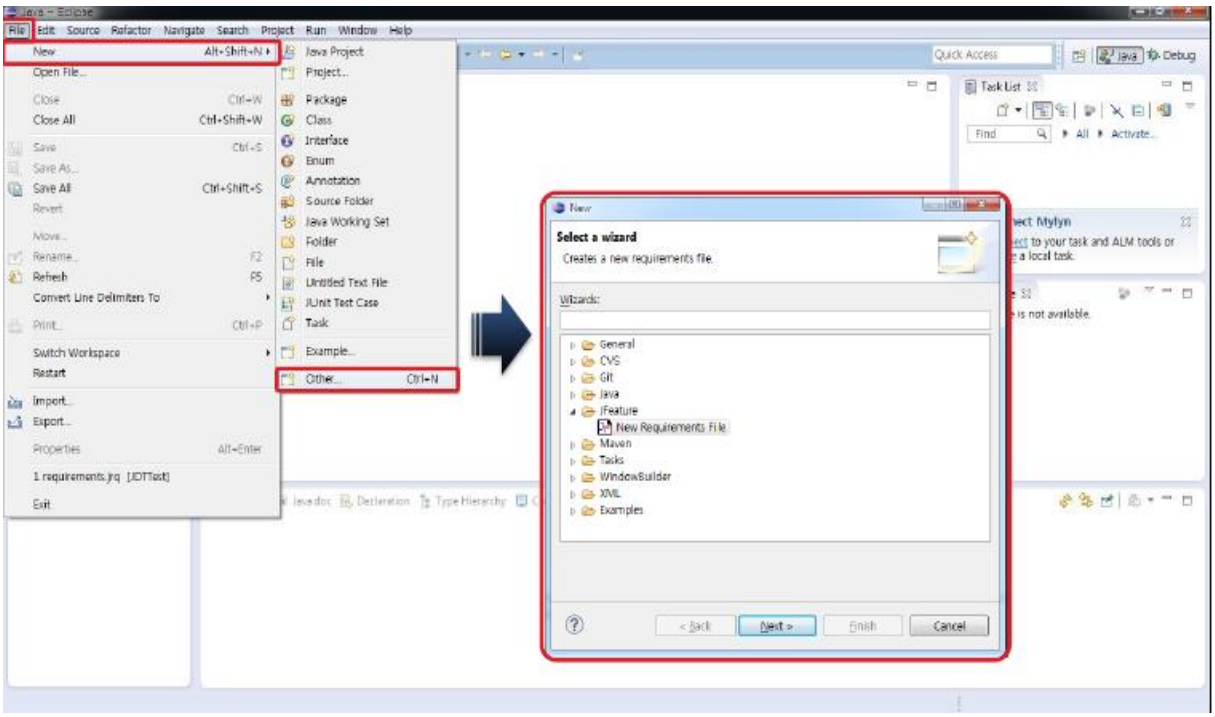
3.2.1 JFeature 를 다운로드 받은 후 Eclipse 폴더에서 Plugins 디렉토리에 압축을 해제한다.



3.2.2 Eclipse 를 실행한 뒤 Window > Show View > Other 에서 JFeature 를 선택한다.

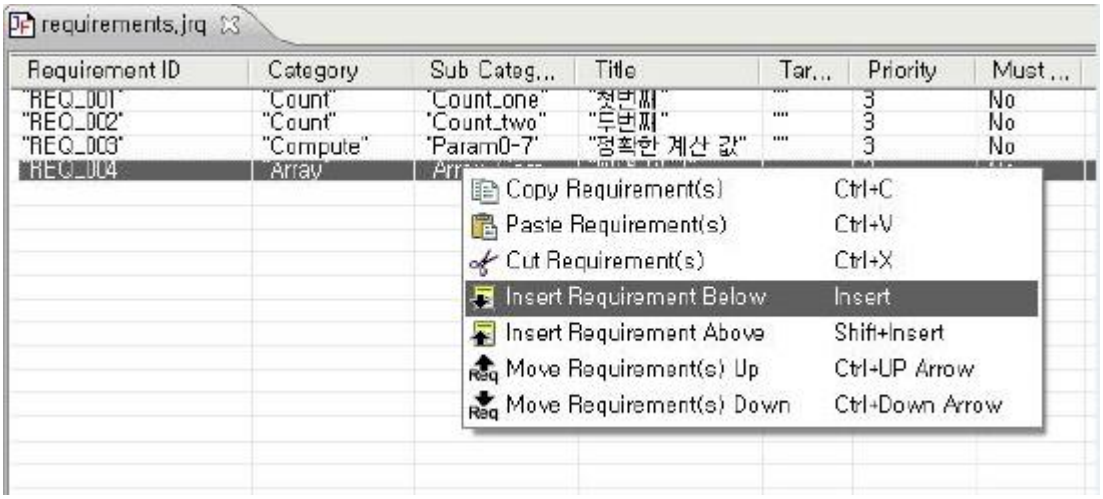
3.3 JFeature 사용법

3.3.1 Eclipse 에서 New > Other > JFeature > JFeature Requirements File 을 선택한다.

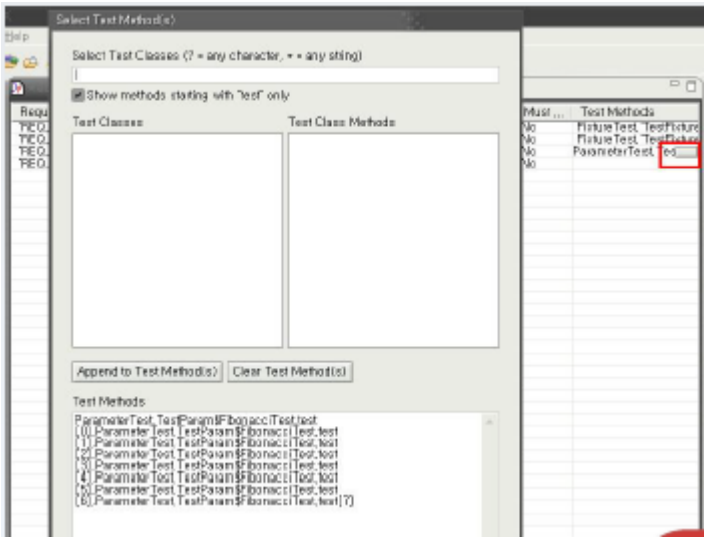


3.3.2 요구사항이 적용될 프로젝트를 선택하고, 요구사항 파일명을 입력한다.

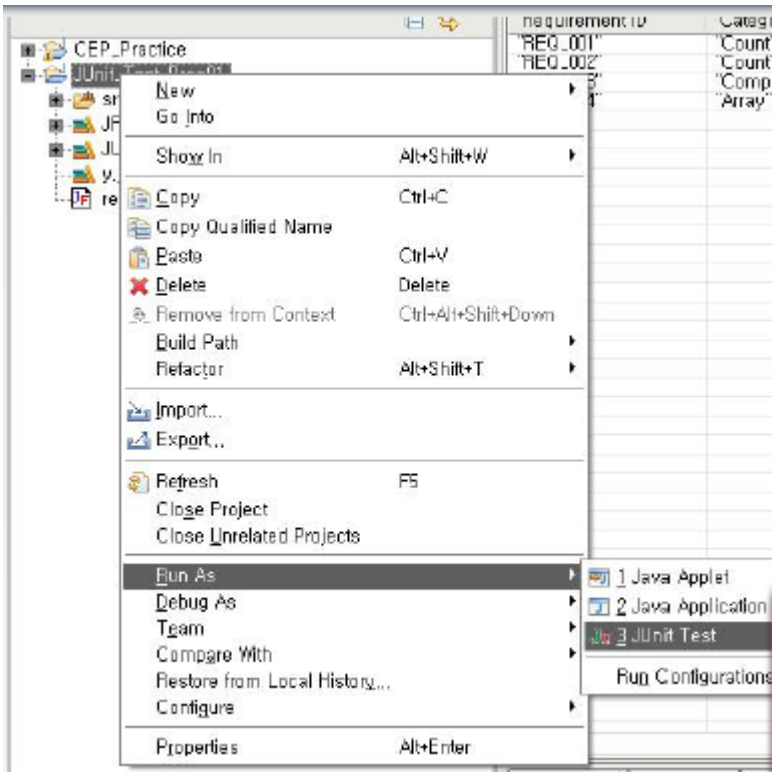
3.3.3 상단에 추가된 메뉴를 이용해 요구사항을 입력한다.



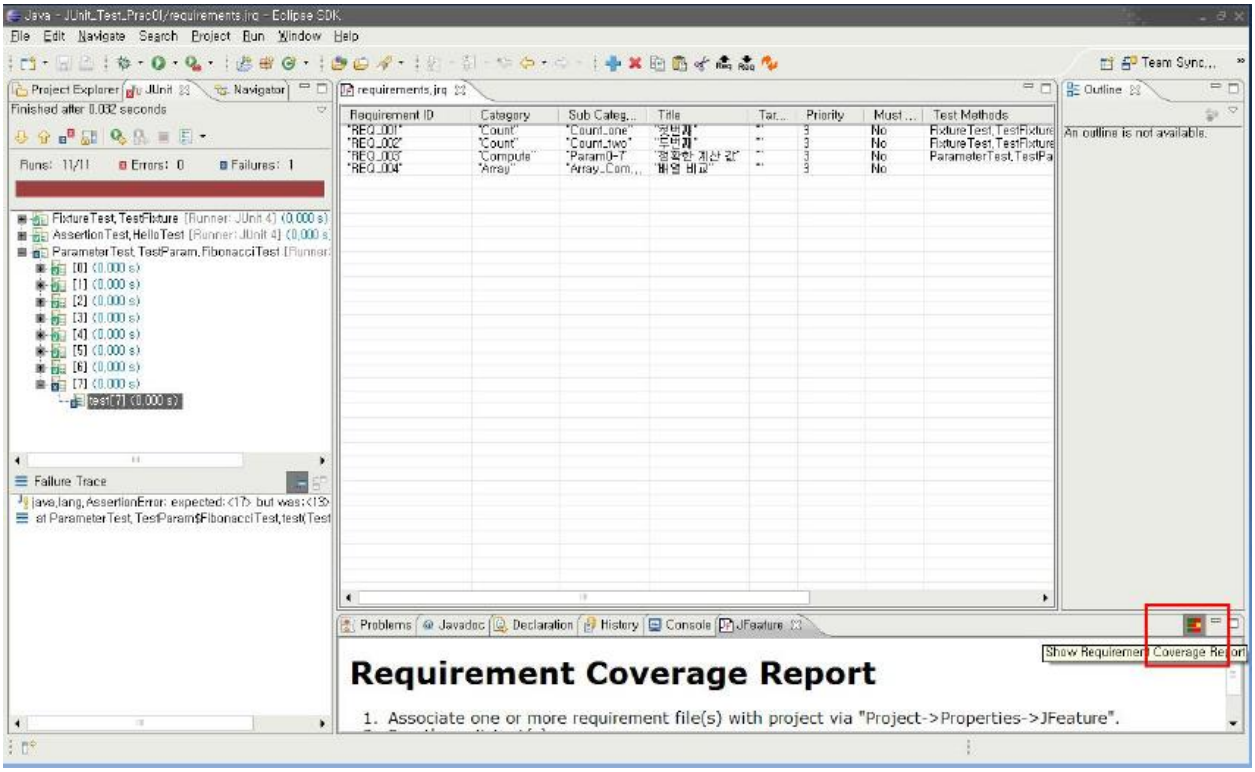
3.3.4 해당 Requirements 에 맞는 TestMethod 를 지정한다.



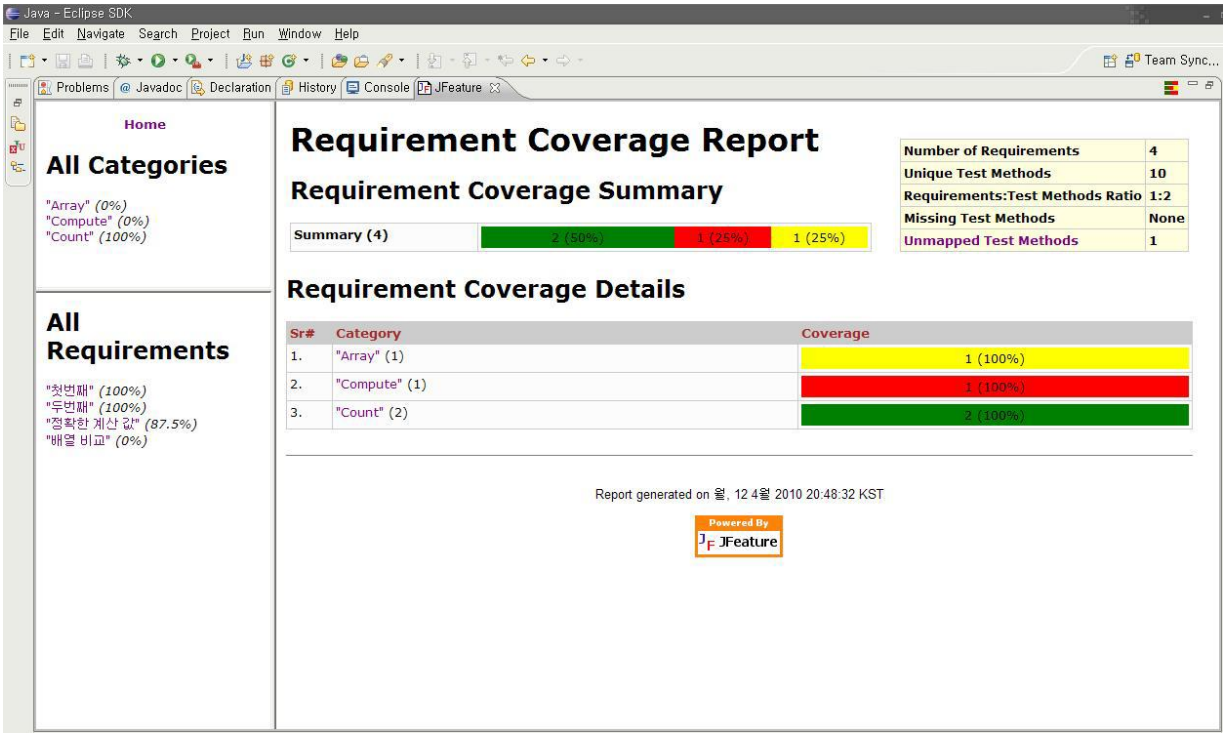
3.3.5 Junit Test 실행을 합니다.



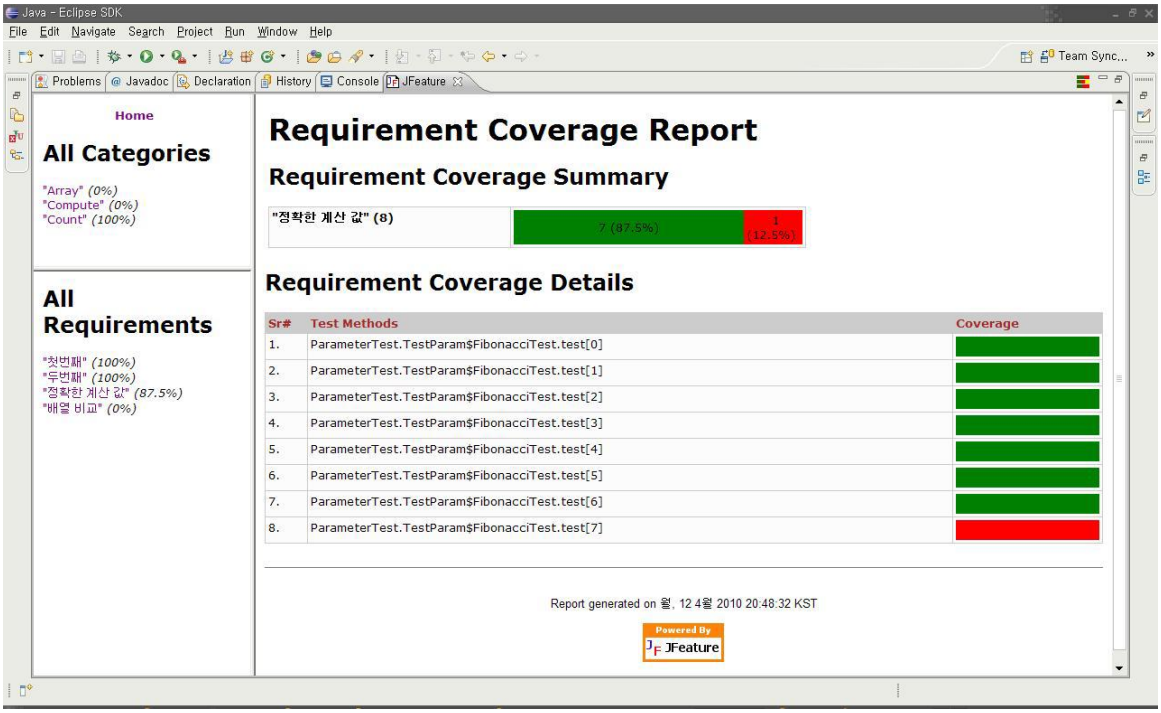
3.3.6 UnitTest Result 가 보이고, 하단의 JFeature 탭에 Requirement Coverate Report 가 생긴 것을 확인한다. 탭 우측의 버튼을 클릭하여 Detail 창을 볼 수 있다.



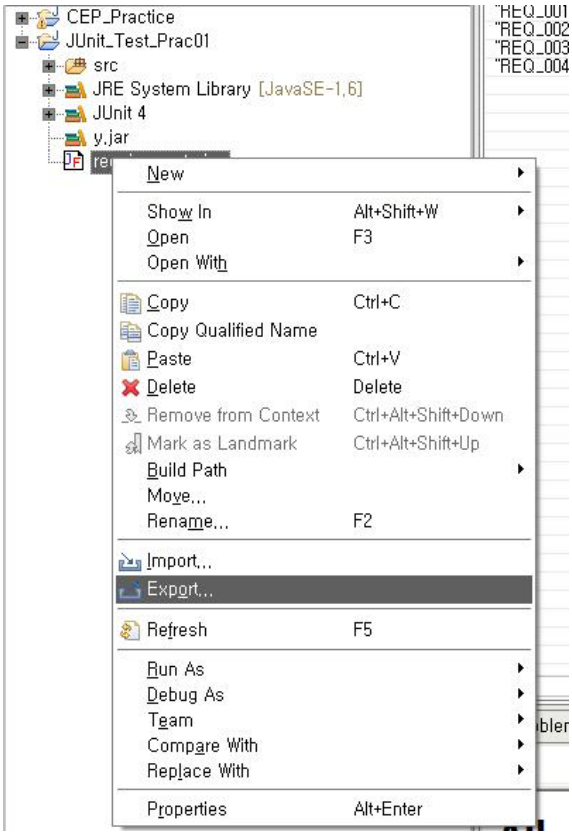
3.3.7 아래의 그림은 Requirement Coverage Report Summary 이다. 노란색은 Requirements 는 있으나, TestMethod 가 없는 경우를 나타낸다. 또한 빨간색은 TestMethod 가 실패한 경우, 녹색은 TestMethod 가 성공한 경우이다.

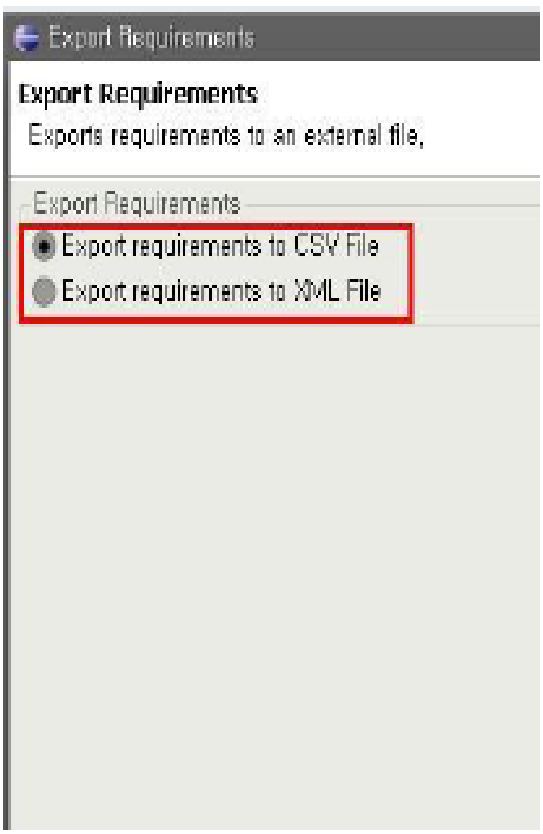


3.3.8 아래의 그림은 Requirement Coverage Details 창이다. 에러 내용에 대한 Detail 내용을 확인할 수 있다.



3.3.9 Requirements 파일을 저장한다. CSV 파일 혹은 XML 으로 저장할 수 있다. CSV 파일은 Excel 에서 불러와 편집이 가능하다.





4 Junit

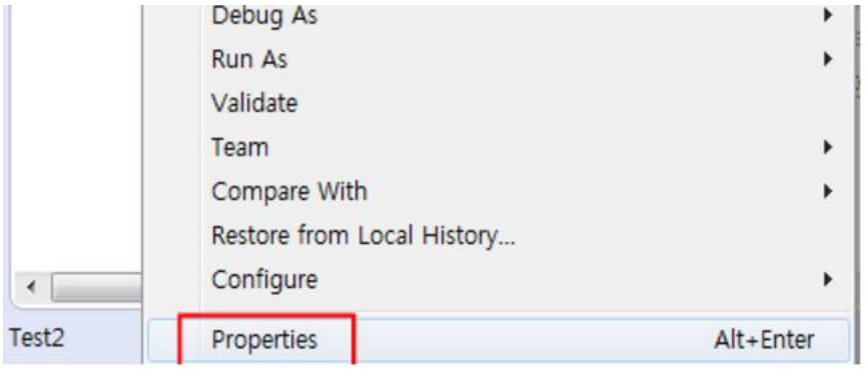
- 단위 테스트를 위한 프레임 워크
- J Unit은 테스트 주도 개발(TDD, Test-Driven Development, 테스트를 먼저 한 뒤 코드를 작성하는 방법)에서 많이 사용하는 프레임워크이며 자동화된 테스트가 가능

  ⇨ 단위 테스트 : 전체 프로그램을 구성하고 있는 기본 단위(unit) 프로그램이 정상적으로 동작하는지 테스트하는것

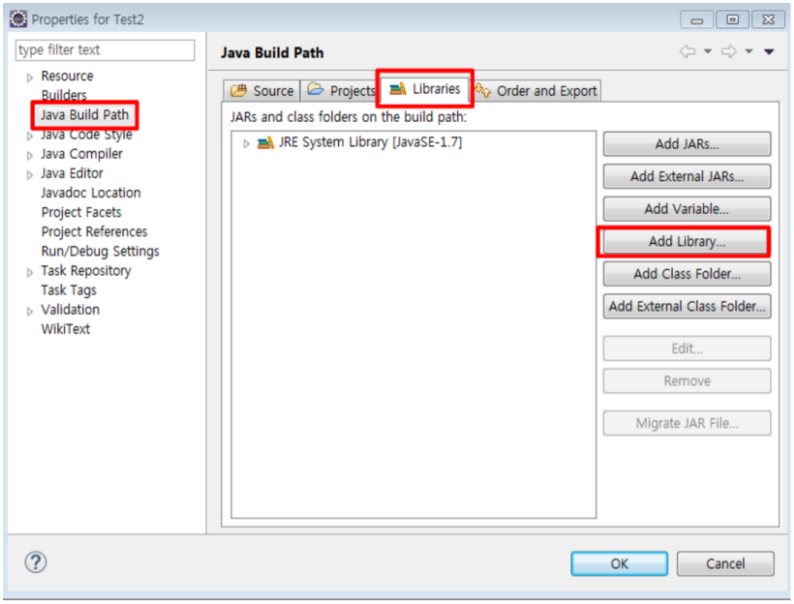
  ⇨ 프레임워크 : 여러 애플리케이션에서 재활용 가능하고, 공유 가능한 부분을 미리 만들어 놓은 기반 구조를 의미

4.1 Eclipse 에서 Junit 라이브러리 추가하기

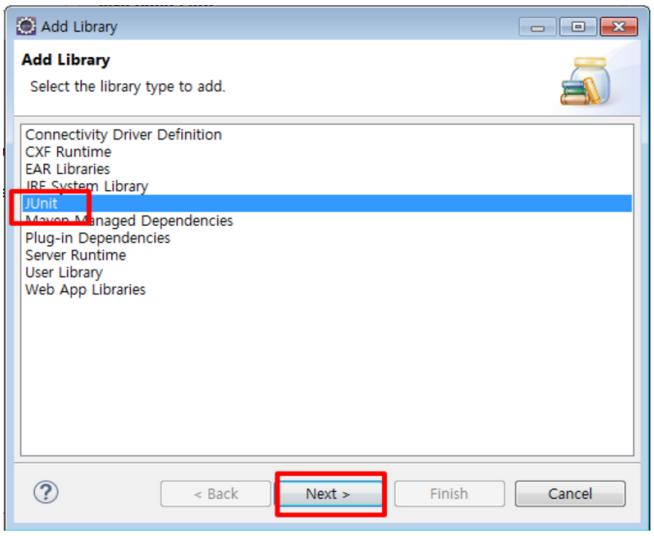
4.1.1 Java Project 의 마우스 오른쪽 버튼을 클릭, 메뉴의 가장 아래에 위치한 Properties 클릭한다.



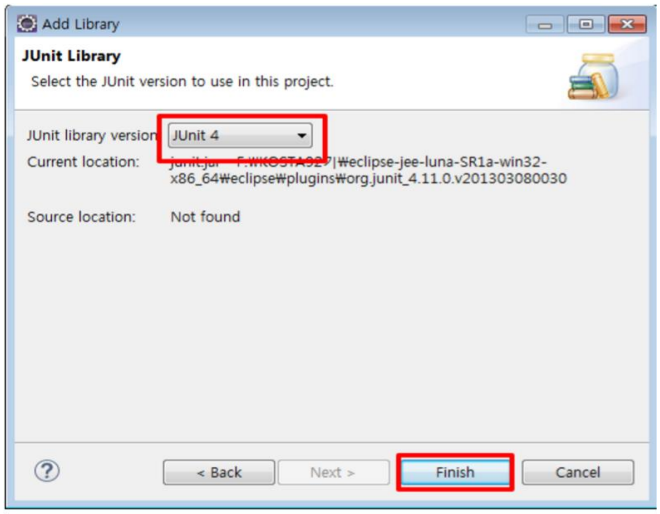
4.1.2 Java Build Path > Library > "Add Library"를 클릭한다.



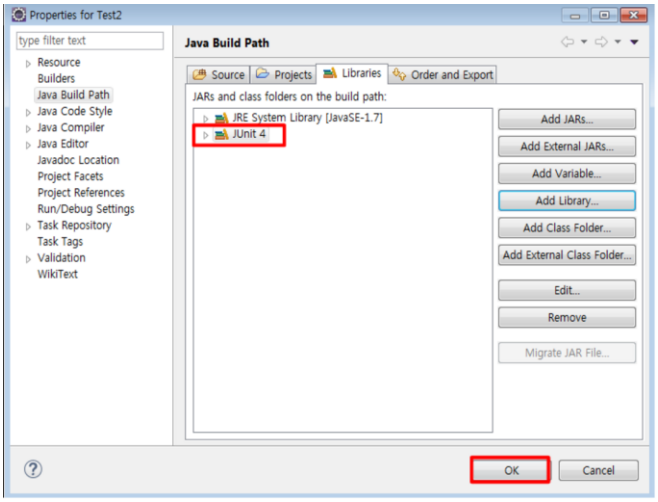
4.1.3 Junit 을 선택하고 Next 버튼을 클릭한다.



4.1.4 Junit Version 을 설정할 수 있는 화면이 나타나면 Junit 4 를 선택한다.



4.1.5 Properties 창에 Junit Library 가 추가된 것을 확인한다.



4.2 JUnit 으로 단위테스트를 하는 방법

- JUnit 라이브러리의 코드를 사용해서 테스트 클래스를 만들고 테스트를 진행하는 것
- 테스트를 진행할 때 필요한 클래스

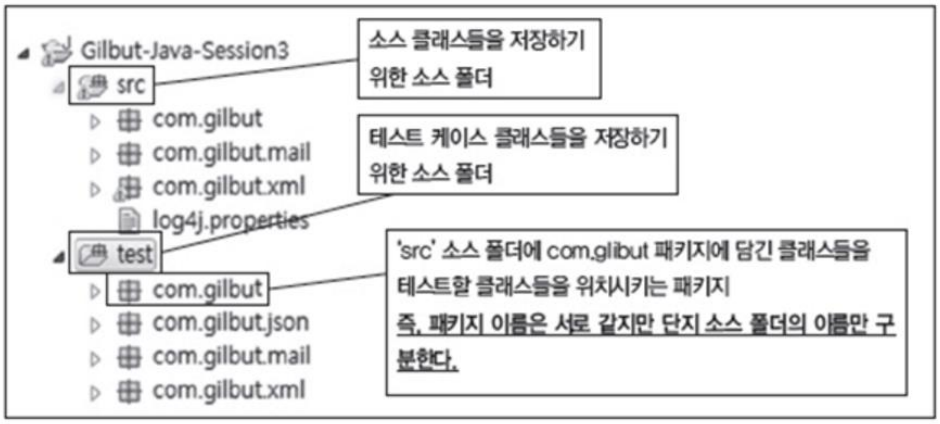
  ⇨ 테스트하고자 하는 대상 클래스(TargetClass)

  ⇨ 대상 클래스를 테스트하는 테스트 클래스(TestTargetClass)

  ⇨ J Unit 프레임워크를 제공하는 TestCase 클래스
- 소스 클래스(Source Class) : 기능이 구현된 클래스

  테스트 클래스(TestClass), 테스트 케이스 클래스(TestCaseClas) : 테스트 구문을 포함한 클래스

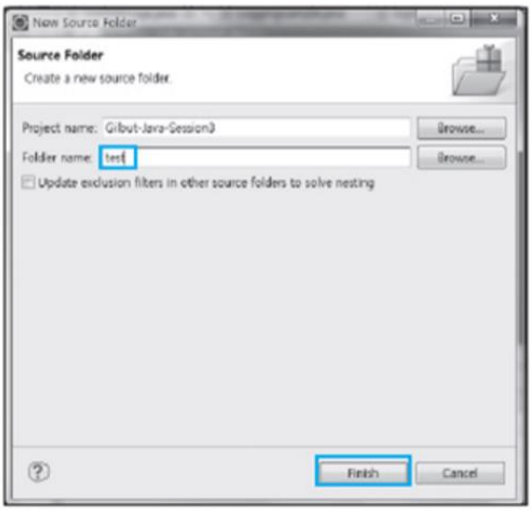
  ⇨ 보통 이 두개의 클래스는 각기 다른 소스폴더(Source Folder)에 따로 저장해서 구분



4.3 사용자가 직접 테스트 클래스를 만드는 방법 → TestCase Class 를 상속받는 방법

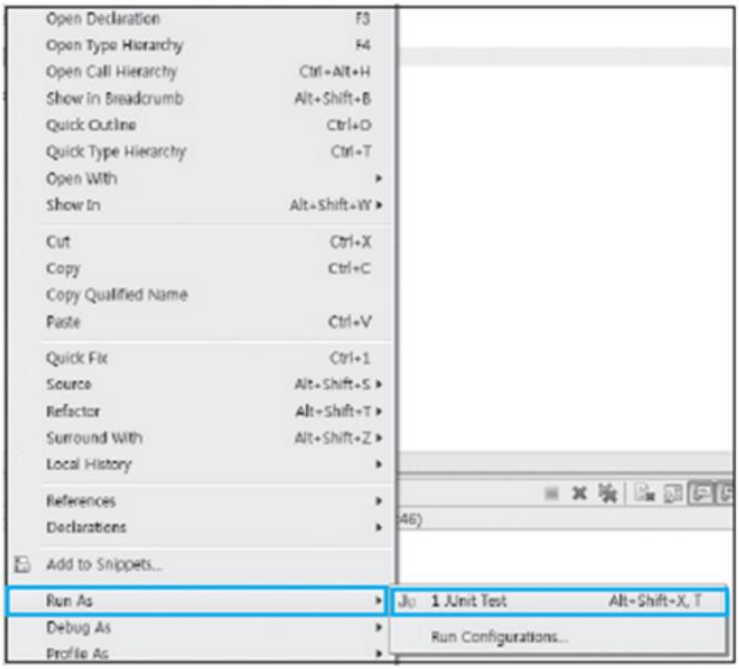
4.3.1 DateUtil 클래스가 속한 Java Project 에서 마우스 오른쪽 버튼 클릭,

4.3.2 [new] → [source folder]를 클릭, folder name에 "test"라고 입력한 뒤 [finish] 클릭



4.3.3 Test() 메소드 영역을 마우스로 더블 클릭을 하면 음영 표시,

4.3.4 이 영역에 마우스 오른쪽 버튼 클릭 → [Run As] → [JUnit Test] 클릭



4.4 JUnit 프레임워크의 유용한 클래스들 ➔ Assert Class

- Assert Class

⇨ 데이터 검증을 하기 위한 method들로 구성

⇨ Test method의 런타임 중간에 발생하는 데이터에 대해서 검증 가능

⇨ 개발자가 예상한 값과 정확히 일치하면 success로 처리

⇨ 예상한 값과 클래스에서 받은 값이 일치하지 않는다면 AssertionError가 발생하게 되며

JUnit에서는 failure라고 처리

- Overriding 된 assert() method들

Method Name	Description
assertArraEquals(Object[ ]expected, Object[ ]actual),	두 개의 배열이 같은지 확인한다. Expected 매개변수에는 예상되는 갑을 넣고 actual은 실행 결과 값을 넣는다.
assertEquals(Object expected, Object actual)	두 개의 매개변수가 같은지 확인한다. 메소드 이름에서 추축할 수 있듯이 내부적으로 equals() 메소드를 사용하기 때문에 값이 같은 같은 확인한다.
assertSame(Object expected, Object actual)	두 개의 매개변수가 같은지 확인한다. assertEquals()와 달리 JVM 메모리 주소까지 같

	같은 확인한다.
assertNotSame(Object expected, Object actual)	두 개의 매개변수가 다른지 확인한다. assertSame()메소드와 반대되는 기능이다.
assertNull(Object target)	매개변수 target이 null 인지 확인한다.
assertNotNull(Object target)	매개변수 target이 null이 아닌지 확인한다.
assertTrue(Boolean condition)	매개변수 condition이 true인지 확인한다.
assertFalse(Boolean condition)	매개변수 condition이 false인지 확인한다.

- JUnit Annotation

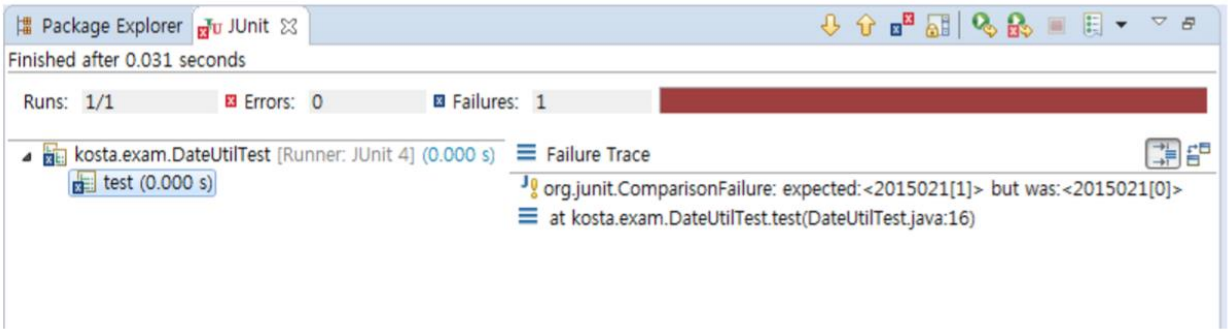
Annotation	Explanation
@Test	Unit Test를 수행할 Method
@Ignore	Test를 수행하지 않을 Method
@After	Test Method가 실행 전,후 초기화 및 자원 정리
@Before	
@AfterClass	모든 Method에 대한 Test 수행 전 후에 한번 초기화 및 자원 정리
@BeforeClass	

4.5 Assert 클래스의 Method를 사용하는 방법

```
@Test
public void test() {
    String today = DateUtil.getCurrentDate();

    //getCurrentDate() 메소드가 정상적인 결과값을 반환하는지
    //테스트하기 위해서 사용된 assertNotNull()과 assertEquals()메소드
    assertNotNull(today);
    assertEquals("20150211", today);
}
```

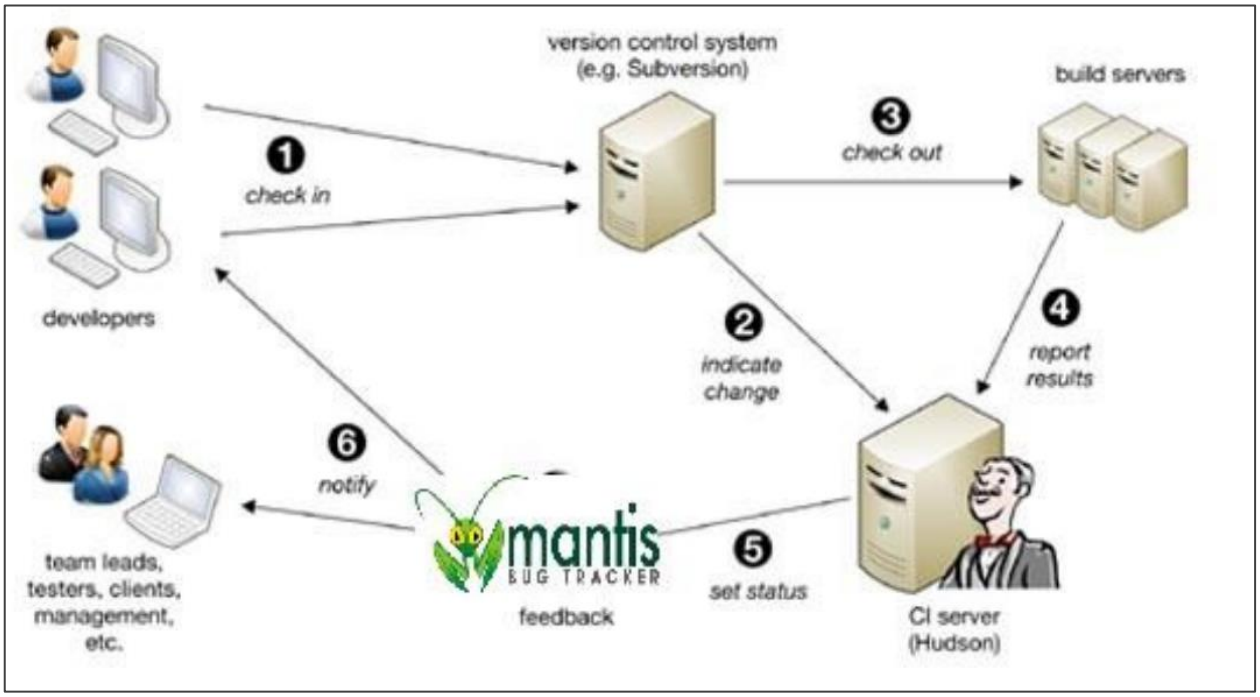
- assert 관련 method 실행한 결과



⇨ Failure Trace를 보면 어떤 값이 같지 않은지, 몇 번째 라인이 잘못되었는지 등 실행 결과를 정확하게 알 수 있다.

5 CTIP (Continuous integration + Continuous Test , Continuous Test & Integration Platform)

- 코드 품질 관리 영역, 빌드 및 배포 관리 영역, 소스코드 버전 관리 영역, 대상 서버군으로 구성
- 코드 품질 관리 영역
  - CVS, SVN 등의 소스코드 버전 관리 시스템을 사용하여 프로젝트 전체 소스코드의 일관성을 유지한다. 개발자는 버전 관리 시스템의 repository로부터 최신 소스를 check out하고, 작업내용을 check in 한다.
- 빌드 및 배포 관리
  - CI 서버를 통해 지속적으로 빌드를 수행하고, 대상 서버에 대한 배포작업을 수행한다. 빌드 주기는 정해진 시점에 수행하거나, 버전관리 시스템의 repository에 변경이 있을 경우 즉시 수행될 수 있다.
- 코드 품질 관리
  - 오픈소스 코드 검토 도구를 활용하여 코드 품질을 확인하고, 결과를 개발자에게 통보한다. 다양한 오픈소스 및 상용 품질 관리 도구가 개발되어 있으므로 필요에 따라 선택하여 적용할 수 있다.
- 서버군
  - 운영 서버 배포를 위한 스테이징 서버, 테스트 실행을 위한 테스트 서버 등을 운영한다



5.1 CTIP 구성 요소

- CI Server
- 소스코드 버전 관리 시스템
- 빌드 시스템
- 품질 관리 도구

1. Glen : 다양한 품질 관리 도구를 쉽게 적용하고, 통합된 보고서를 생성하는 품질관리 도구의 관리 도구입니다.
2. JUnit : 대표적인 Java 단위 테스트 프레임워크 입니다.
3. Cobertura : JUnit 등의 단위 테스트에 대한 커버리지 측정 도구입니다.
4. Checkstyle : 각 소스코드에 대하여 표준 소스코드 스타일 가이드의 준수 여부를 검토합니다.
5. PMD : copy & paste 된 코드와 소스코드의 잠재적인 오류 사항을 검토합니다.
6. JavaNCSS : Java 프로젝트의 size를 측정합니다.
7. JDepend : Java 패키지의 설계 품질을 검토합니다.

5.2 CTIP Tool

Category	Tool
CI Server	Hudson
Unit Testing	Junit
Build	Hudson
Version Control	Git
Bug Tracking & Community	Mantis
Static Analysis	Eclipse TPTP, Sonar, cppcheclipse

5.3 성공적인 CI 수행 조건

- 단일 소스 저장소(Source Repository) 관리

  ⇨ 소스코드 버전 관리 시스템을 도입하여 소스코드를 일관성 있게 관리한다.

- 빌드 자동화

  ⇨ CI 서버와 Ant 빌드 스크립트를 통해 빌드를 자동화 한다.

- 자체적으로 테스트 가능한 빌드

  ⇨ 코드 품질 관리 도구들을 통한 단위 테스트 등의 테스트 성공 여부와 서버의 deploy 성공 여부를 통해 빌드 성공 여부를 확인한다.

- 빠른 빌드 수행

  ⇨ CI 서버와 Ant 빌드 스크립트를 통해 단계적 빌드를 구성한다

- 운영환경과 유사한 환경 구성

  ⇨ CI 서버의 환경을 운영환경과 유사하도록 구성한다.

- 최신 결과물에 대한 쉬운 접근

  ⇨ CI 서버를 통해 최신 빌드 결과물 및 빌드 리포트를 쉽게 내려받을 수 있다.

- 손쉬운 빌드 상태 모니터링

  ⇨ CI 서버가 제공하는 RSS feed 혹은 e-mail 전송기능을 통해 빌드 상태를 모니터링 할 수 있다

5.4 CTIP의 장점

- 코드 품질에 대한 더 높은 신뢰성 제공
- 프로젝트에 대한 더 나은 가시성 제공
- 수동으로 수행해야 하는 반복 작업을 줄일 수 있음
- 시간과 장소에 구애 받지 않고 배포 할 수 있는 소프트웨어를 만들 수 있고, 위험을 줄일 수 있음

5.5 CTIP 흐름도