'12. 3. 16, Presentation #1

Junit & Eclipse

T2

200511318 김희재

200511326 박현진

200711436 서영주

200913987 이승효

Contents

<< Unit Testing >>

- Test Driven Development (TDD)
- Junit
- Hamcrest
- Mockito

<< Eclipse >>

- Subclipse
- TPTP
- ANT

Unit Testing

- Test Driven Development(TDD)
- Junit
- Hamcrest
- Mockito

TDD

◆ *테스트코드 실행순서*

- 실제코드를 작성하기전 테스트 코드를 거쳐 실제 필요한 API를 개발
- 테스트 코드의 통과여부로 개발 진행과정을 통제

◆ *TDD 개발 과정*

- 기존 개발 과정 (디자인 → 개발 → 테스트)
- TDD 개발 과정 (디자인 → 테스트 스크립트 개발 → 개발 → 리펙토링)
  - 코드작성 이전에 테스트 스크립트를 먼저 작성
  - 실제 코드 작성, 테스트 스크립트를 통과(PASS) 될 수 있도록 코딩.
  - 코드작성 후 코드의 가독성, 유지 보수성 향상을 위해 리펙토링

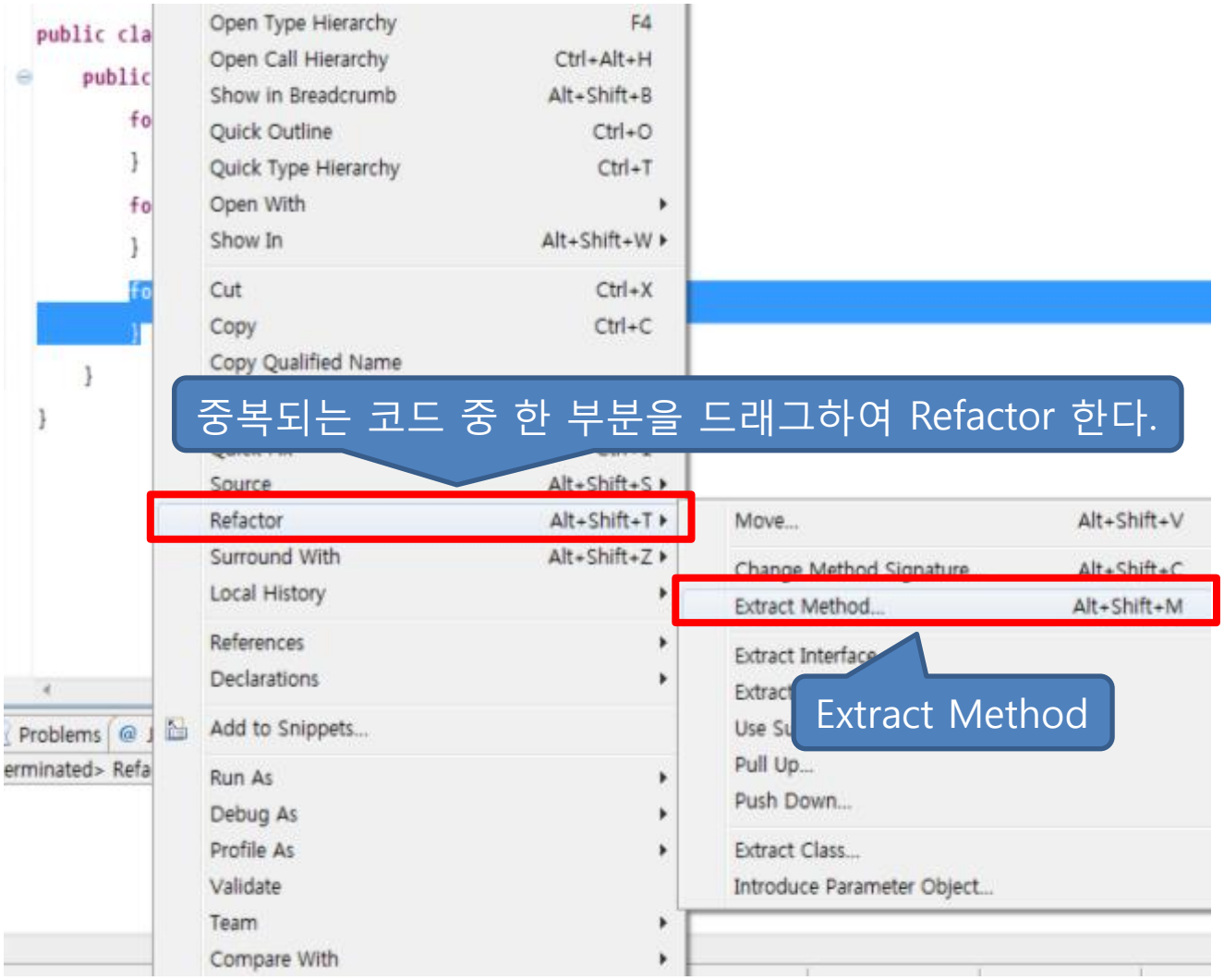
  ※ 리펙토링(Refactoring) : 코드의 기능은 변하지 않으면서,
  구조적 변화를 통해 성능을 끌어올리는 기술

Refactoring 사용 예

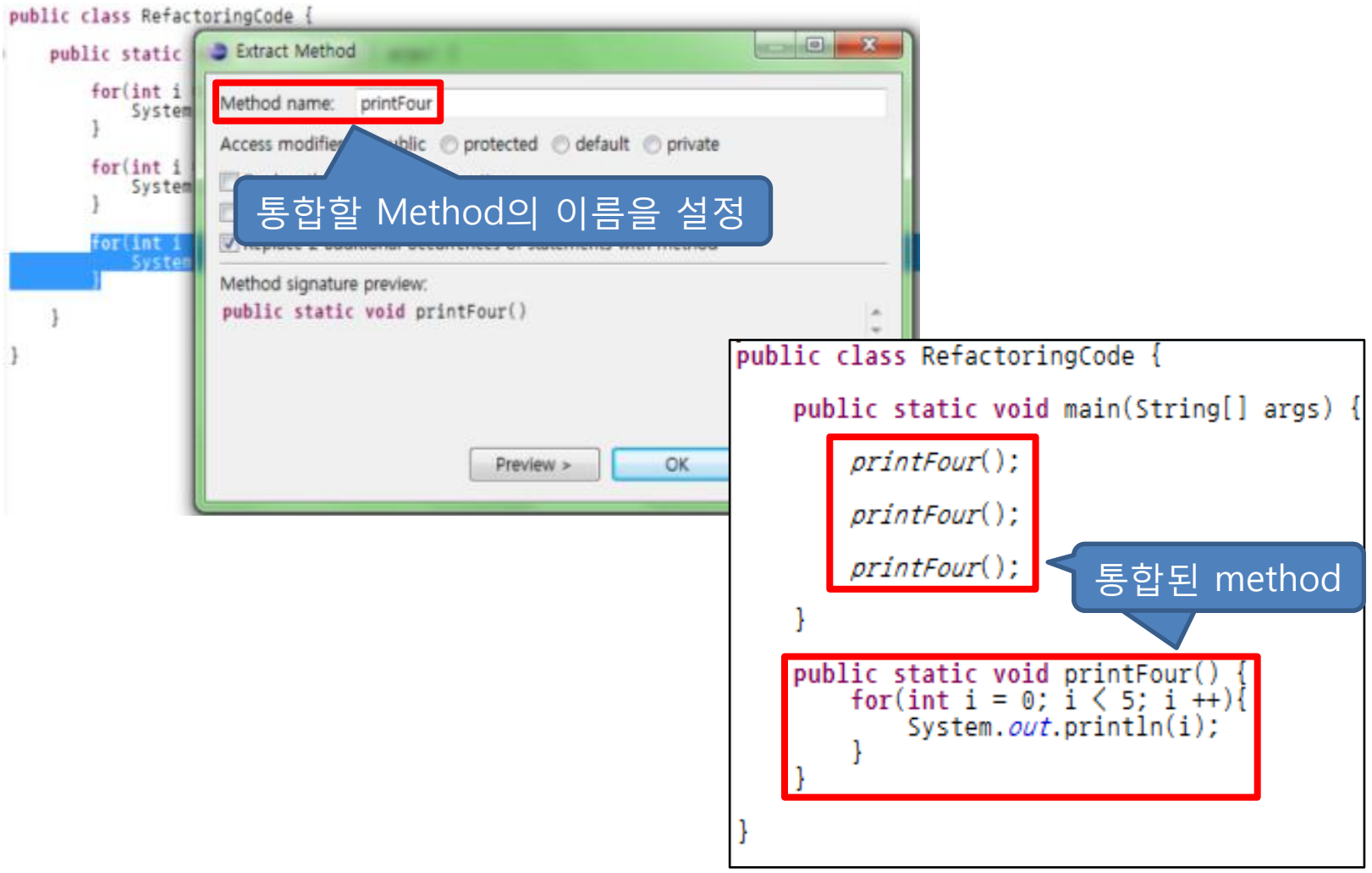
```
public class RefactoringCode {

    public static void main(String[] args) {

        for(int i = 0; i < 5; i ++){
            System.out.println(i);
        }

        for(int i = 0; i < 5; i ++){
            System.out.println(i);
        }

        for(int i = 0; i < 5; i ++){
            System.out.println(i);
        }

    }

}
```

중복되는 3개의 for문을 Refactoring 한다.

Refactoring 사용 예



Refactoring 사용 예



Unit Test

◆ *Unit*

- 어플리케이션을 테스트 하기 위한 최소 단위
- Procedural Programming 에서는 Function, Procedure
- OOP 에서는 Method가 Unit

◆ *Unit Test*

- 소프트웨어 모듈을 테스트 할때 수행하는 최소 단위(Unit)의 테스트
- 자동화된 테스트를 통해 테스트 결과를 문서화
- Unit Test의 결과를 통해 개발 진행과정을 결정
- 모든 Unit을 테스트 하여도 모든 Error을 검출하는 것은 불가능
- Unit자체의 결함(중복, 비적절)시 테스트 코드 작성 비용이 증가

JUnit

◆ *Annotation*

@Test	테스트시 실행할 메소드 앞에 붙여줌
@Test(expected)	테스트 메소드에서 발생할 예외를 지정 정의된 예외는 메소드 안에서 try, catch를 해줄 필요가 없음
@Test(timedout)	테스트시 메소드에 시간제한을 두어 일정 시간동안 실행 후 종료
@Ignore	테스트 수행시 해당 메소드를 테스트에서 제외 제외되는 이유를 적어서 직관적으로 확인가능
@Before, @After	모든 테스트 메소드가 실행되기 전, 후에 실행되는 메소드를 지정
@BeforeClass, @AfterClass	테스트 메소드의 개수에 상관없이 테스트 전, 후에 한번만 실행
@RunWith	기본 TestRunner대신 지정된 클래스를 통해 테스트를 수행

JUnit

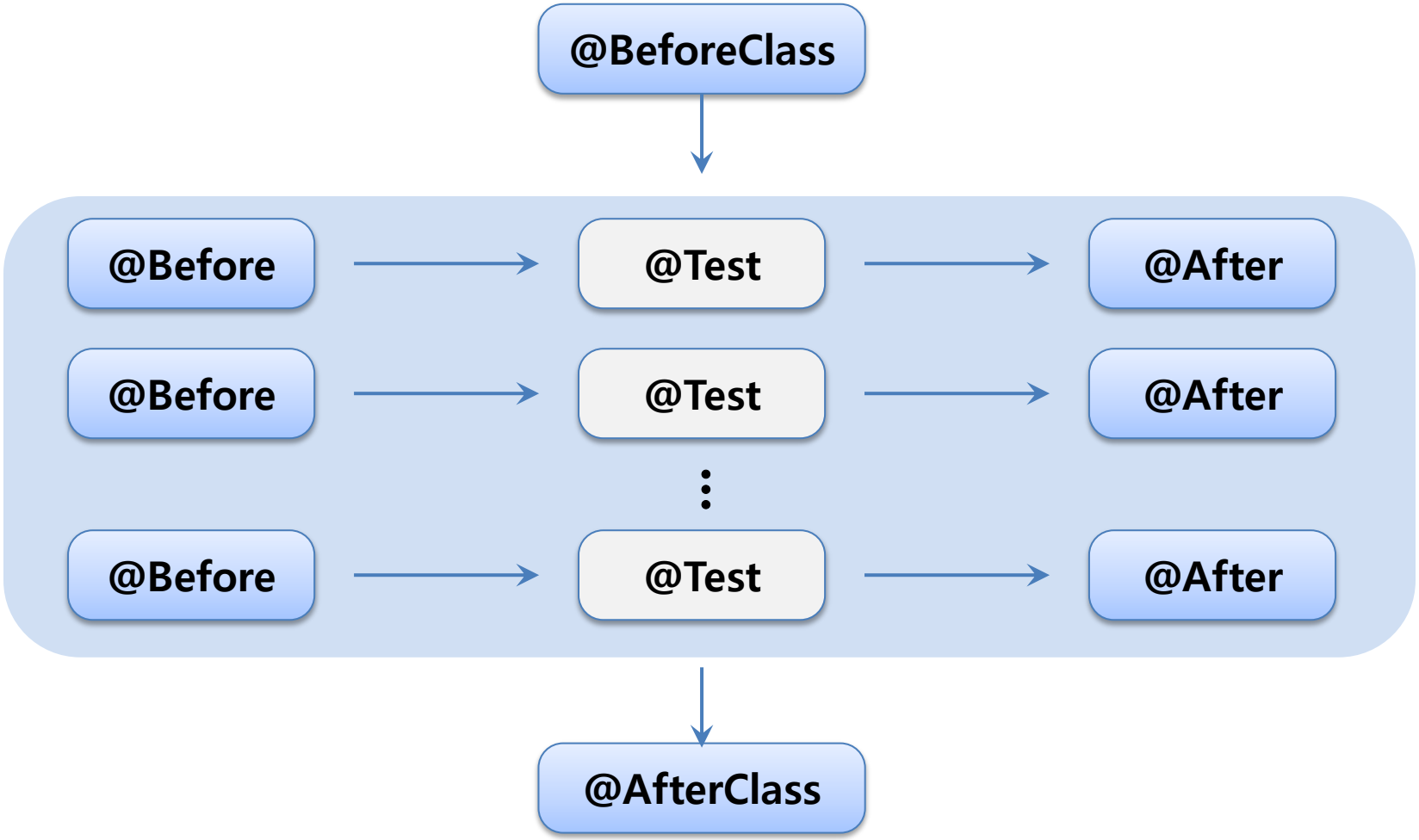
◆ *Annotation*

@SuiteClasses	테스트를 하려는 여러 개의 클래스들을 지정
@Category	테스트 케이스에 Tag값을 지정하여 해당 Test Case들만 실행
@Parameters	여러 개의 파라메터 값을 테스트할때 자동으로 테스트를 수행

◆ *Method*

assertEquals	기본 TestRunner대신 지정된 클래스를 통해 테스트를 수행
assertSame, assertNotSame	두 객체가 동일한 객체인지 아닌지 검사 내부적으로 두 객체의 메모리 주소가 같은지 검사
assertNull, assertNotNull	기대값(객체)의 Null 유/무를 판단
assertTrue, assertFalse	기대값의 참/거짓 을 판단
fail	해당 메소드 호출시 즉시 해당 테스트 실패 미완료 테스트 케이스, 예외처리 테스트에 사용

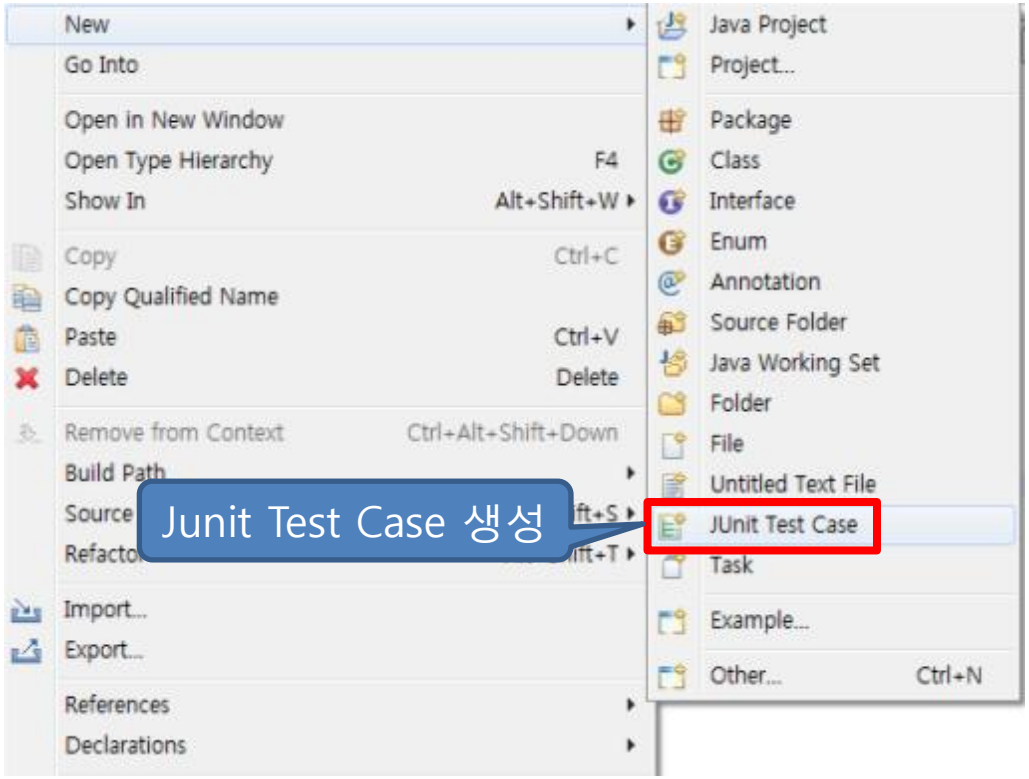
JUnit의 특징


@BeforeClass
@Before
@Test
@After
@Before
@Test
@After
@Before
@Test
@After
@AfterClass

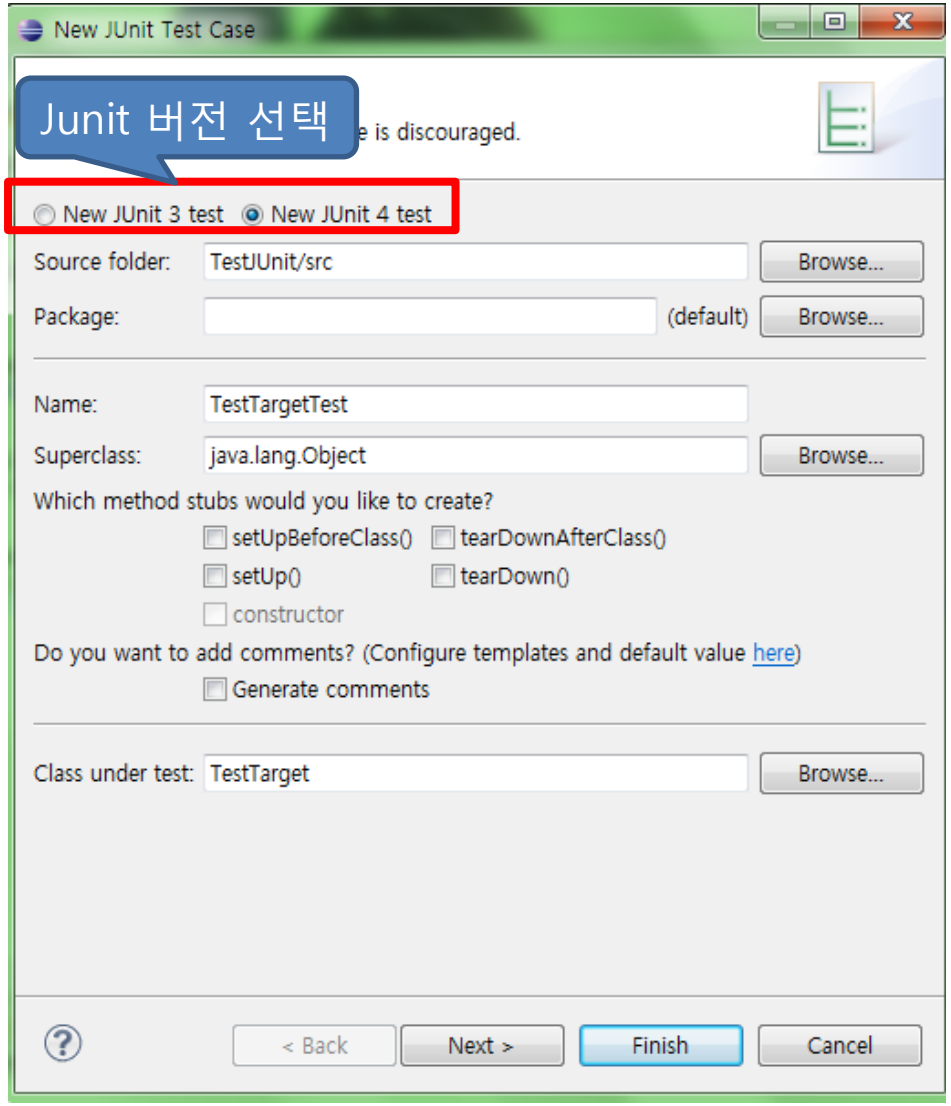
JUnit4 사용 예

테스트하고자 하는 코드

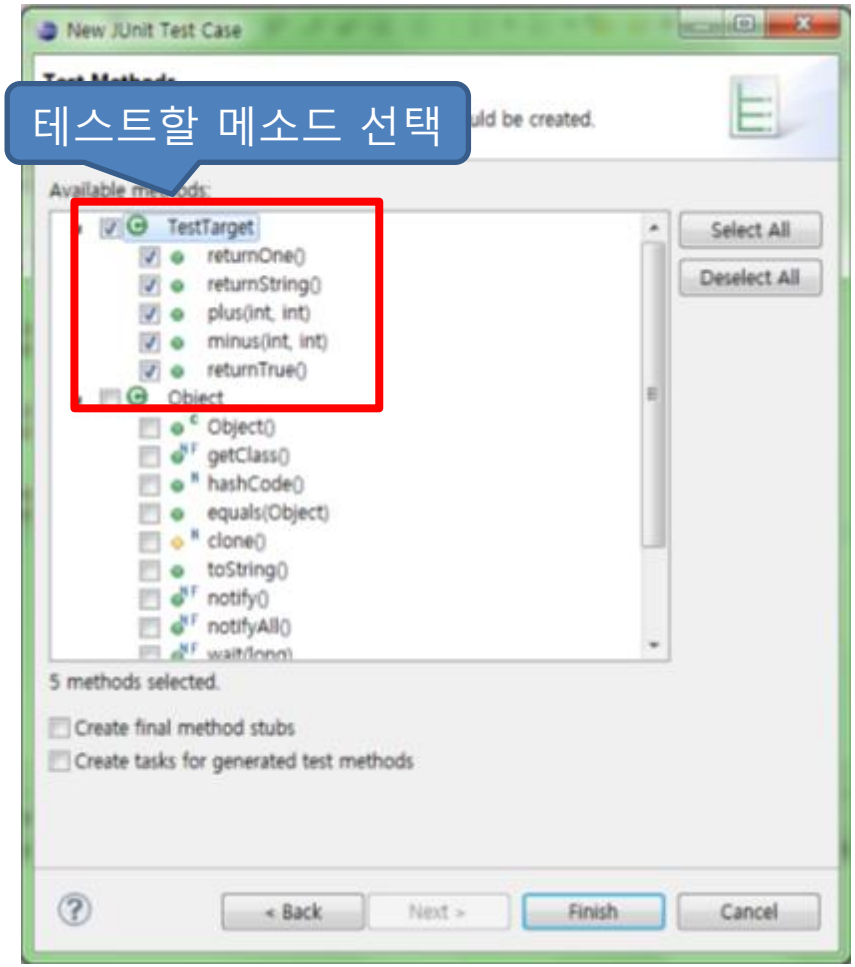
```
public class TestTarget {

    public int returnOne(){
        return 1;
    }

    public String returnString(){
        return "String";
    }

    public int plus(int x, int y){
        return x + y;
    }

    public int minus(int x, int y){
        return x - y;
    }

    public boolean returnTrue(){
        return true;
    }

}
```



JUnit4 사용 예


New JUnit Test Case
Junit 버전 선택
e is discouraged.
New JUnit 3 test
New JUnit 4 test
Source folder:
TestJUnit/src
Browse...
Package:
(default)
Browse...
Name:
TestTargetTest
Superclass:
java.lang.Object
Browse...
Which method stubs would you like to create?
setUpBeforeClass()
tearDownAfterClass()
setUp()
tearDown()
constructor
Do you want to add comments? (Configure templates and default value here)
Generate comments
Class under test:
TestTarget
Browse...
< Back
Next >
Finish
Cancel

JUnit4 사용 예


New JUnit Test Case
테스트할 메소드 선택
TestTarget
returnOne()
returnString()
plus(int, int)
minus(int, int)
returnTrue()
Object
Object()
getClass()
hashCode()
equals(Object)
clone()
toString()
notify()
notifyAll()
Select All
Deselect All
5 methods selected.
Create final method stubs
Create tasks for generated test methods
< Back
Next >
Finish
Cancel

JUnit4 사용 예

```
import static org.junit.Assert.*;

public class TestTargetTest {

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
    }

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testReturnOne() {
        fail("Not yet implemented");
    }

    @Test
    public void testReturnString() {
        fail("Not yet implemented");
    }
```

생성된 테스트 결과

JUnit4 사용 예

```
import static org.junit.Assert.*;

public class TestTargetTest {

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
    }

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testReturnOne() {
        fail("Not yet implemented");
    }

    @Test
    public void testReturnString() {
        fail("Not yet implemented");
    }
```

생성된 테스트 결과

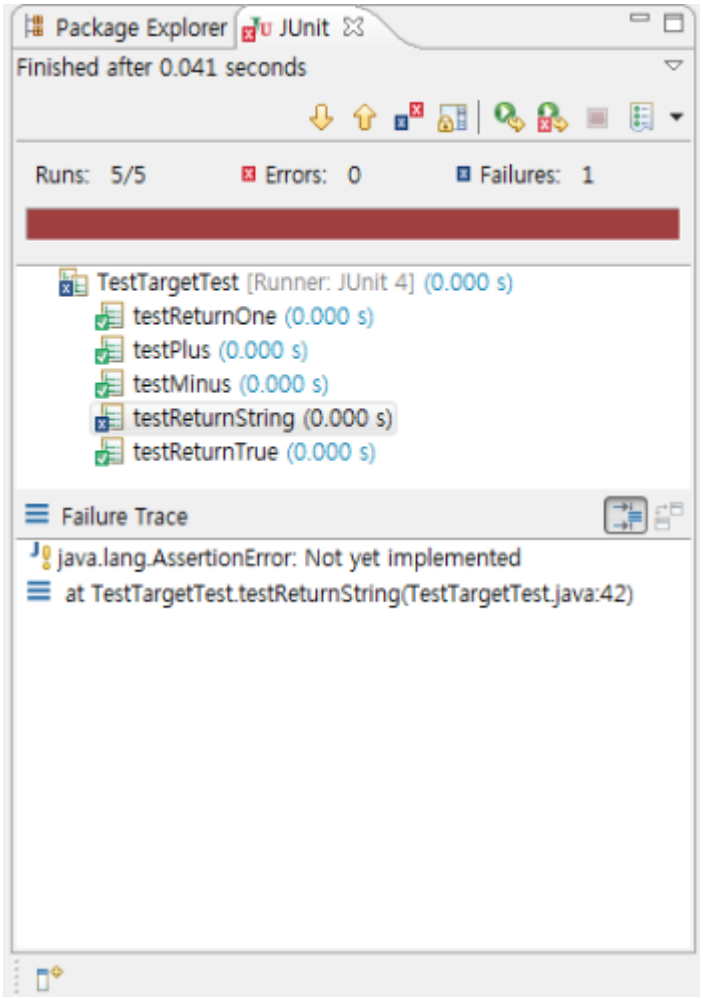
JUnit4 사용 예

```
@Test
public void testReturnOne() {
    assertEquals(1, testTarget.returnOne());
}

@Test
public void testReturnString() {
    fail("Not yet implemented");
    assertEquals(testTarget.returnString(), "String");
}

@Test
public void testPlus() {
    assertEquals(testTarget.plus(1, 2), 3);
}
```

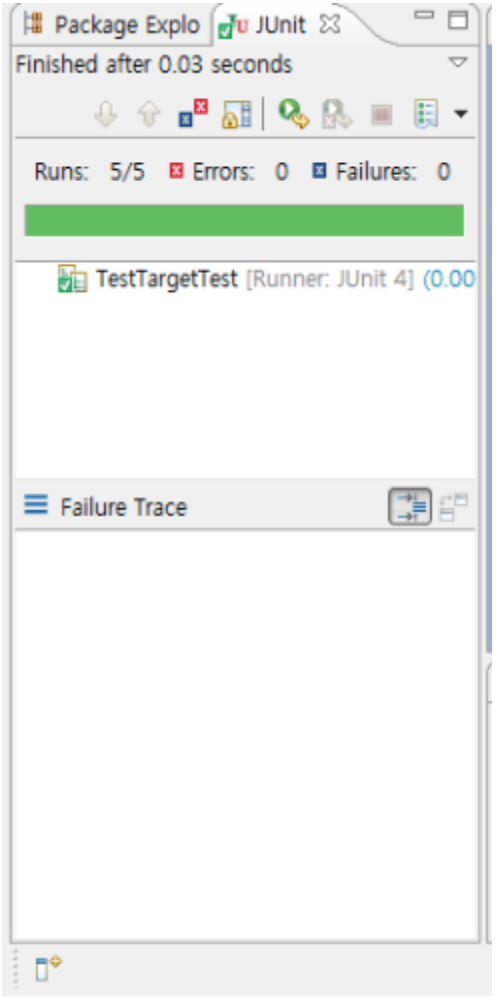
Assert 메소드들을 이용한 테스트 케이스들

JUnit4 사용 예

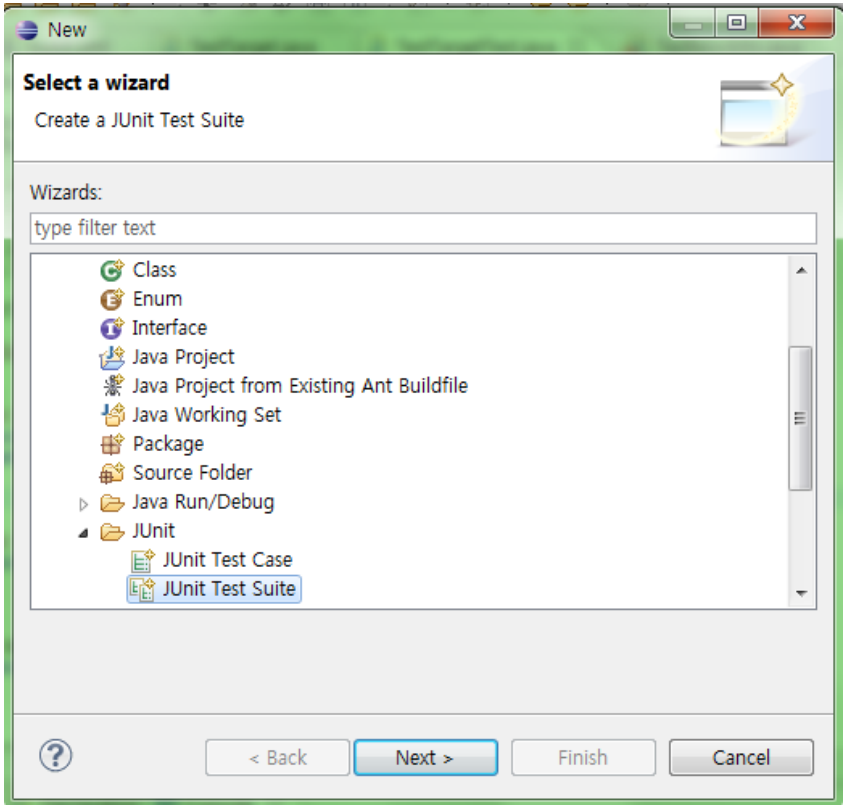
테스트 실패시

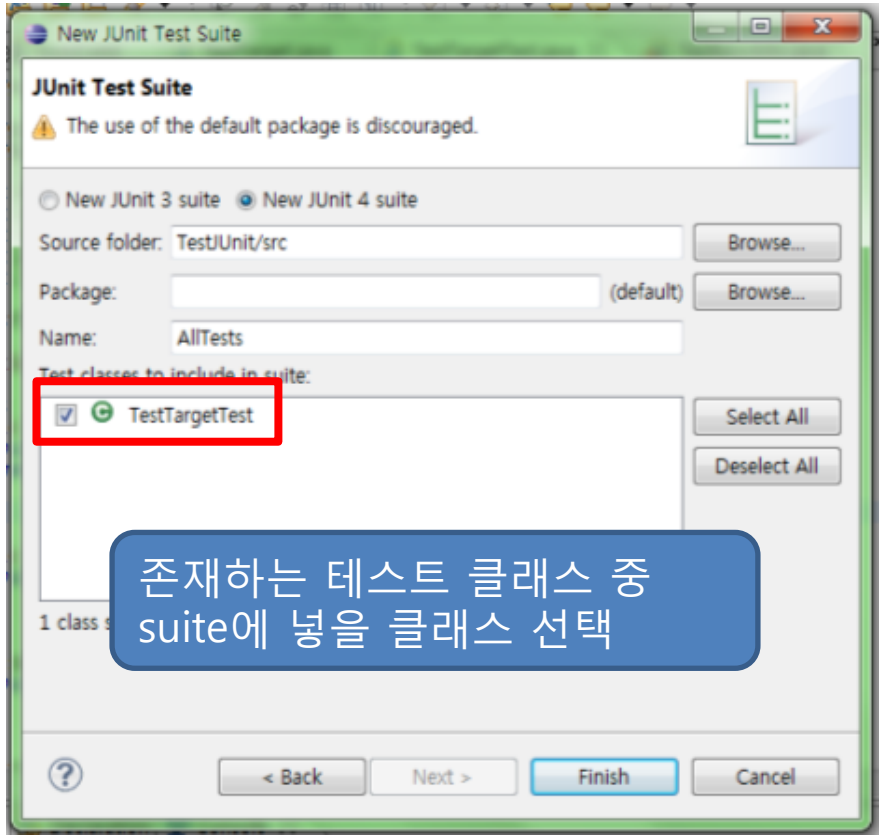


테스트 성공시



JUnit4 사용 예

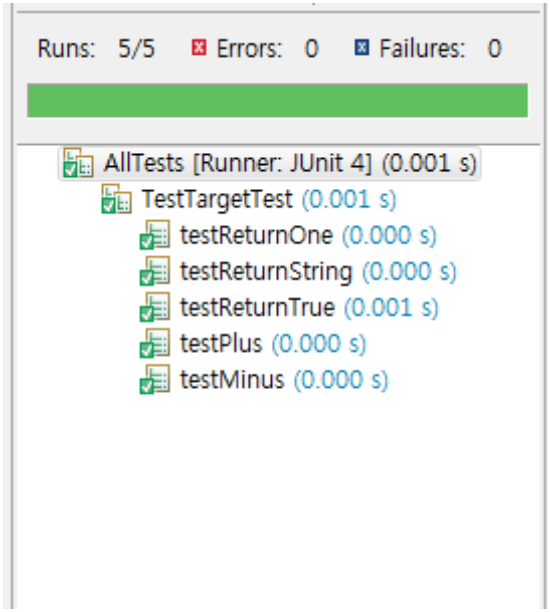

New
Select a wizard
Create a JUnit Test Suite
Wizards:
type filter text
Class
Enum
Interface
Java Project
Java Project from Existing Ant Buildfile
Java Working Set
Package
Source Folder
Java Run/Debug
JUnit
JUnit Test Case
JUnit Test Suite
< Back
Next >
Finish
Cancel


New JUnit Test Suite
JUnit Test Suite
The use of the default package is discouraged.
New JUnit 3 suite
New JUnit 4 suite
Source folder:
TestJUnit/src
Browse...
Package:
(default)
Browse...
Name:
AllTests
TestTargetTest
Select All
Deselect All
존재하는 테스트 클래스 중
suite에 넣을 클래스 선택
< Back
Next >
Finish
Cancel

JUnit4 사용 예

```
@RunWith(Suite.class)
@SuiteClasses({ TestTargetTest.class })
public class AllTests {

    private static Test suite() {
        // TODO Auto-generated method stub
        TestSuite suite = new TestSuite(AllTests.class.getName());
        return suite;
    }
}
```

생성시 선택된 테스트 클래스들이 전부 들어감



테스트 실행 결과가 AllTests밑에 각 테스트별로 나타나게 됨

JUnit의 문제점

- GUI나 모듈 이상 크기의 테스트 등 **의존성이 있는 코드**나 **테스트의 범위가 큰 경우** 테스트 코드를 작성이 어려움이 있다.
- **반환 결과가 없는 기능**의 테스트에 어려움이 있다

```
public void printText(){
    System.out.println("Test");
}

@Test
public void testPrintText(){
    assertEquals(printText(), ???);
    assertThat(printText(), is(???));
}
```

콘솔에 출력하는 문장

printText의 호출 여부,
printText 실행시 출력되는 문자열의 결과를 테스트할 수 없다.

Hamcrest

◆ *Hamcrest*

- jMock 라이브러리에 포함되어 있는 비교표현API에서
  리팩토링을 통해 독립테스팅에 있어서 기능이나 성능의 향상이 아닌 표현식의 가독성을 높여줌
- 테스트 케이스 작성시 문맥적으로 자연스러운 **생활언어(English)에 가깝게 표현**

◆ *Hamcrest 적용*

적용 전

```
assertEquals(100, account.getBalance());
assertNotNull(resource.newConnection());
assertTrue(account.getBalance() > 0)
assertTrue(user.getLoginName().indexOf("GG")>-1);
```

적용 후

```
assertThat(account.getBalance(), is(eaualTo(10000)));
assertThat(resource.newConnection(), is(notNullValue());
assertThat(account.getBalance(), isGreaterThan(0));
assertThat(user.getLoginName(), containsString("GG"));
```

Hamcrest 라이브러리

◆ *Matcher*

패키지	설 명
Org.hamcrest.**core**	오브젝트나 값들에 대한 기본적인 Matcher들
Org.hamcrest.**beans**	Java 빈(Bean)과 그 값 비교에 사용되는 Matcher들
Org.hamcrest.**collection**	배열과 컬렉션 Matcher들
Org.hamcrest.**number**	수 비교를 하기 위한 Matcher들
Org.hamcrest.**object**	오브젝트와 클래스를 비교하는 Matcher들
Org.hamcrest.**text**	문자열 비교
Org.hamcrest.**xml**	XML 문서 비교

◆ *Core*

패키지	설 명	클래스명
Anything	어떤 오브젝트가 사용되든 일치한다고 판별	IsAnything
describedAs	테스트 실패시 보여줄 추가적 메시지를 만들어주는 메시지 데코레이터	DescribedAs
equalTo	두 오브젝트가 동일한지 판별	IsEqual
Is	내부적으로 equalTo와 동일.	Is

Hamcrest 사용 예

- ***import***
  - Import시 static이 쓰인다.

```
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.CoreMatchers.equalTo;
```

- ***함수의 사용***

```
@Test
public void testReturnString() {
    //fail("Not yet implemented");
    assertEquals(testTarget.returnString(), "String");
    assertThat(testTarget.returnString(), is("String"));
    assertThat(testTarget.returnString(), is(equalTo("String")));
}
```

Mockito

◆ *사용목적*

실제 객체를 만들기엔 비용과 시간이 많이 들거나 의존성이 길게 걸쳐져 있어 제대로 구현하기 어려울 경우 **가짜 객체(Mock 객체)**를 만들어서 사용

◆ *Mockito??*

테스트용 Mock객체를 만들고 API를 이용한 검증을 지원해주는 라이브러리

◆ *주요기능*

- 검증 – Mock객체의 특정 메소드가 호출됐는지 확인한다.
- Argument Matcher – 인자에 상관 없이 메소드의 호출 검증 가능
- 순서 검증 – Mock객체 메소드의 호출 순서도 검증 가능

Mock객체가 필요한 경우

- *구현해야 할 클래스의 스펙의 인터페이스*

```
public interface Cipher {

    public String encrypt(String source);
    public String decrypt(String source);

}
```

- *테스트를 해야 하는데 cipher가 미구현일 경우*

```
@Test
public void testSavePassword() throws Exception {

    UserRegiser register = new UserRegister(); //
    Cipher cipher = ????;

    String userId = "test";
    String password = "temp";

    register.savePassword(userId, cipher.encrypt(password));
    String decryptedPassword = cipher.decrypt(register.getPassword(userId));
    assertEquals(password, decryptedPassword);

}
```

구현이 되어있는데

미구현의 경우

Mockito 메소드

◆ *Mock() 메소드*

Mock(interface.class)	인터페이스나 클래스 지정시 구현 클래스로 객체가 생성된 것처럼 동작

◆ *Verify() 메소드*

verify(T mock)	mock작업이 한 번 수행 되었는지 검증
verify(T mock, VMode mode)	mock작업이 mode에 지정된 만큼 수행 되었는지를 검증

◆ *수행회수 검증 메소드*

thenAnswer(Answer<?> answer)	Answer라는 인터페이스를 구현, 원하는 작업을 수행
thenCallRealMethod()	해당 메소드가 구현되어 있다면, 실제 메소드를 호출
thenReturn(T value)	지정한 값을 리턴
thenReturn(T value, T... values)	지정되어 있는 값을 순차적으로 리턴
thenThrow **(java.lang.Throwable... throwables)**	예외를 야기시키는 Throwable 객체를 지정

Mockito 사용 예

```
import static org.mockito.Mockito.*;

import org.mockito.InOrder;

import java.awt.List;
import java.util.LinkedList;

import org.junit.BeforeClass;
import org.junit.Test;

public class TestMockito {

    @Test
    public void makeMockInterface(){

        List mockedList = mock(List.class);

        // mock 객체 사용
        mockedList.add("one");
        mockedList.add("two");

        // 검증 하기
        verify(mockedList).add("one");
    }
    }
```

Mockito 사용 예

```
@Test
public void makeMock(){

    // interface 뿐 아니라 구체 클래스도 mock으로 만들 수 있다.
    LinkedList listMockedList = mock(LinkedList.class);

    // stubbing
    when(listMockedList.get(0)).thenReturn("first");
    //when(listMockedList.get(1)).thenThrow(new RuntimeException());

    // 첫 번째 element를 출력한다.
    System.out.println(listMockedList.get(0));

    // runtime exception이 발생한다.
    System.out.println(listMockedList.get(1));

    // 999번째 element 얻어오는 부분은 stub되지 않았으므로 null이 출력
    System.out.println(listMockedList.get(999));

    // stubbing 된 부분이 호출되는지 확인할 수 있긴 하지만 불필요한 일입니다.
    // 만일 코드에서 get(0)의 리턴값을 확인하려고 하면 테스트가 깨집니다.
    verify(listMockedList).get(0);
}
```

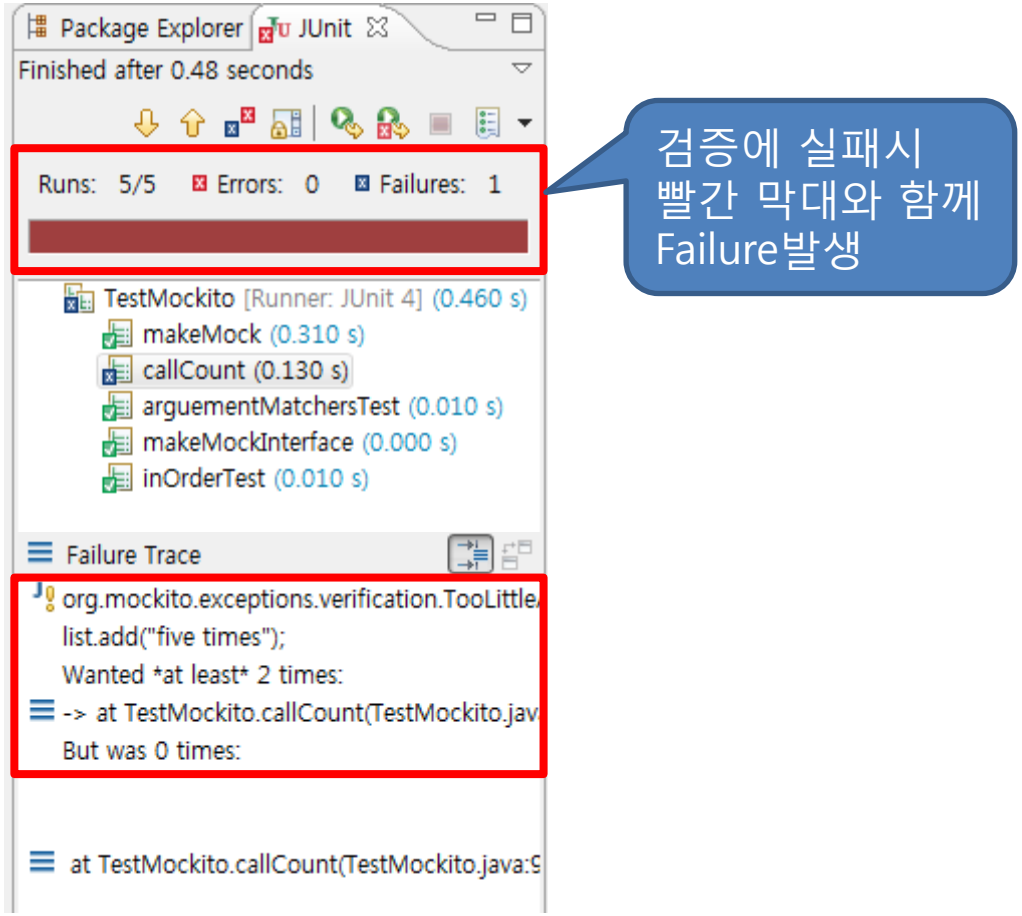
Mockito 사용 예

```
@Test
public void arguementMatchersTest(){

    LinkedList mockedList = mock(LinkedList.class);

    // 내장된 argument matcher인 anyInt()를 이용한 stubbing
    when(mockedList.get(anyInt())).thenReturn("element");

    // 다음 코드는 "element"를 출력한다.
    System.out.println(mockedList.get(999));

    // argument matcher를 이용해 검증할 수도 있다.
    verify(mockedList).get(anyInt());
}
```

Mockito 사용 예

```
public void callCount(){

    List mockedList = mock(List.class);
    // mock 설정
    mockedList.add("once");

    mockedList.add("twice");
    mockedList.add("twice");

    mockedList.add("three times");
    mockedList.add("three times");
    mockedList.add("three times");

    // 아래의 두 가지 검증 방법은 동일하다. times(1)은 기본값이라 생략되도 상관없다.
    verify(mockedList).add("once");
    verify(mockedList, times(1)).add("once");

    // 정확히 지정된 횟수만큼만 호출되는지 검사한다.
    verify(mockedList, times(2)).add("twice");
    verify(mockedList, times(3)).add("three times");

    // never()를 이용하여 검증한다. never()는 times(0)과 같은 의미이다.
    verify(mockedList, never()).add("never happened");

    // atLeast()와 atMost()를 이용해 검증한다.
    verify(mockedList, atLeastOnce()).add("three times");
    verify(mockedList, atLeast(2)).add("five times");
    verify(mockedList, atMost(5)).add("three times");
}
```

Mockito 사용 예


Package Explorer
JUnit
Finished after 0.48 seconds
Runs: 5/5
Errors: 0
Failures: 1
TestMockito [Runner: JUnit 4] (0.460 s)
makeMock (0.310 s)
callCount (0.130 s)
arguementMatchersTest (0.010 s)
makeMockInterface (0.000 s)
inOrderTest (0.010 s)
Failure Trace
org.mockito.exceptions.verification.TooLittle
list.add("five times");
Wanted *at least* 2 times:
-> at TestMockito.callCount(TestMockito.jav
But was 0 times:
at TestMockito.callCount(TestMockito.java:9
검증에 실패시
빨간 막대와 함께
Failure발생

Eclipse Plugins

- Subversion(SVN)
- TPTP
- ANT

Subversion(SVN)

◆ *SVN??*

- 형상관리(SCM: Software Configuration Management) 도구
- 팀 프로젝트를 진행 시 공동의 소스를 관리할 수 있도록 도와주는 프로그램

◆ *특징*

- 개발/수정 단계의 각 버전이 섞이지 않아 쉽게 관리 가능.
- 소스를 잘못 수정했더라도 기록이 남아 되돌리기 쉽다.
- 추가/수정/삭제 등의 기록이 모두 남고 변경사항을 추적하기 쉽다.
- 개발자들이 따로 백업을 하지 않아도 된다.

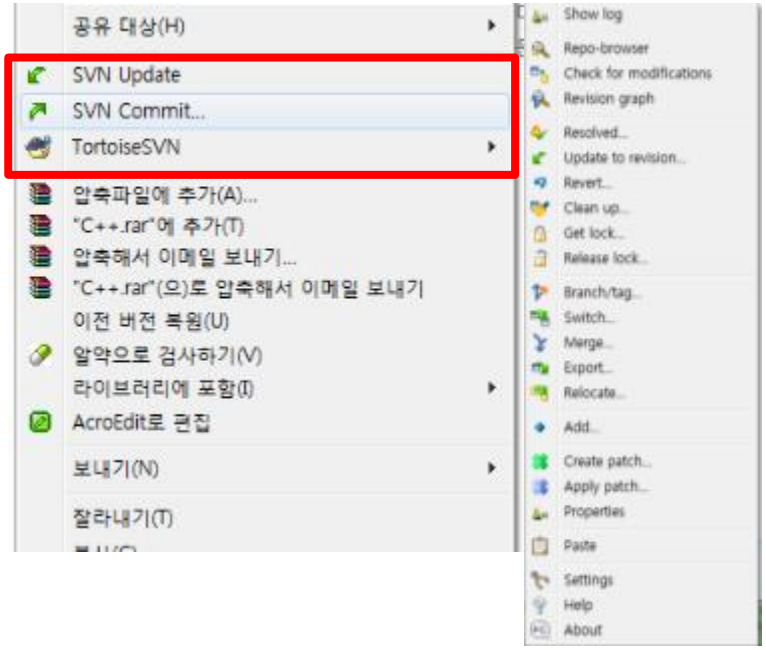
Subversion(SVN)

- *기능*

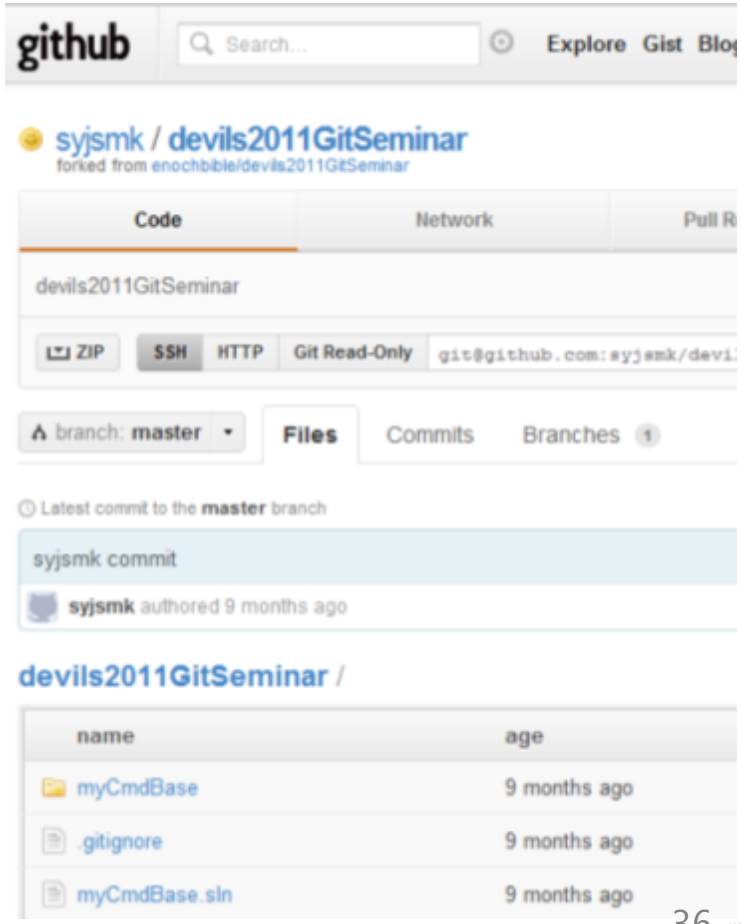
SVN Server	소스코드가 담기는 서버, Client Tool을 사용하여 접속
SVN Client	서버 접근을 위한 개발자용 도구
Repository	SVN Server가 관리하는 소스의 정보가 담긴 시스템
Share	맨 처음 Repository에 프로젝트를 올리는 작업
Check Out	Repository의 최종소스를 처음으로 Client에 내려 받는 작업
Update	서버에 있는 최신소스를 받아오는 작업
Commit (Export)	서버로 자신의 소스를 업데이트 하는 작업

Subversion(SVN)

- ***SVN 프로그램***
  - Tortoise SVN

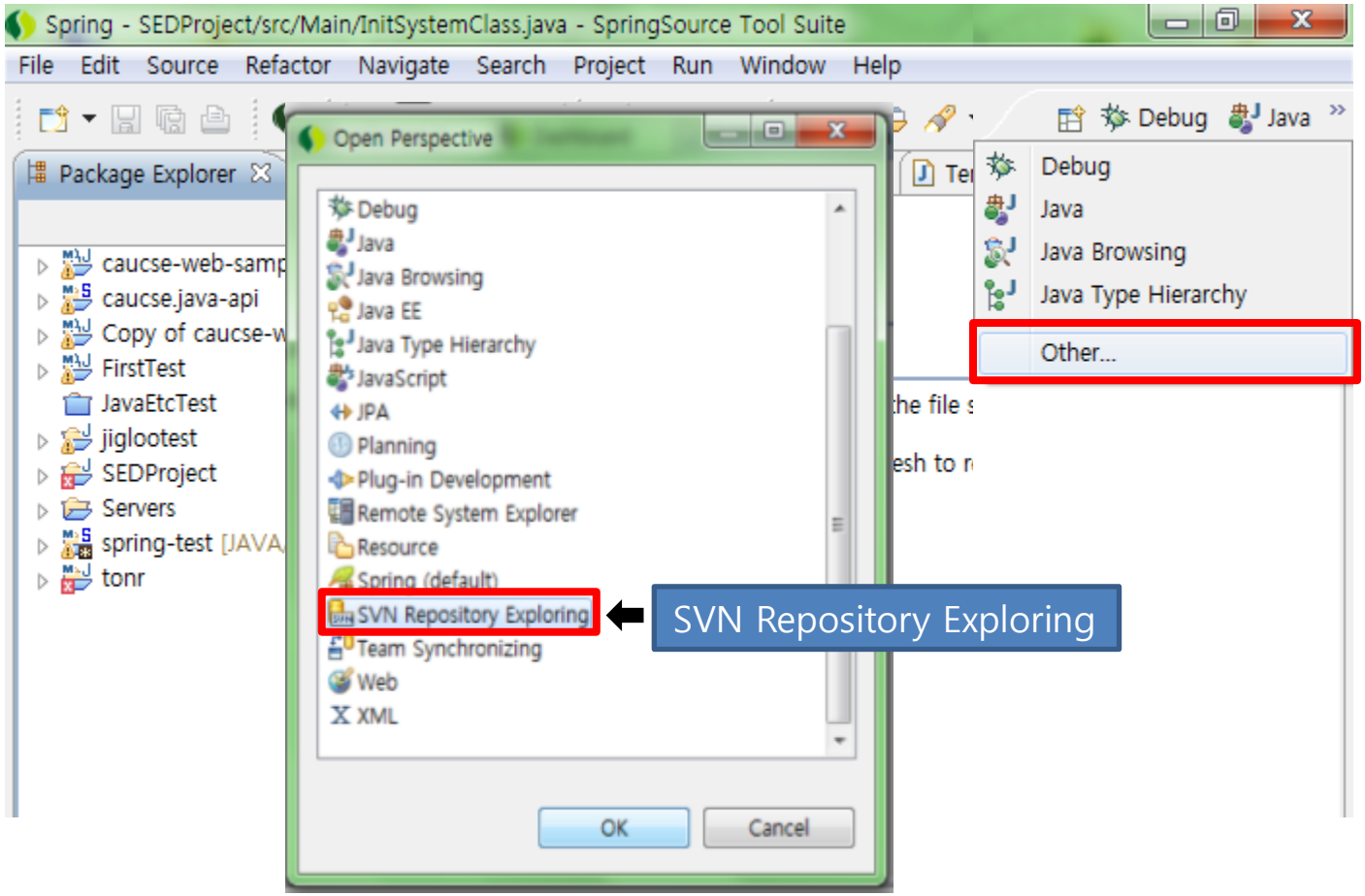


  - github



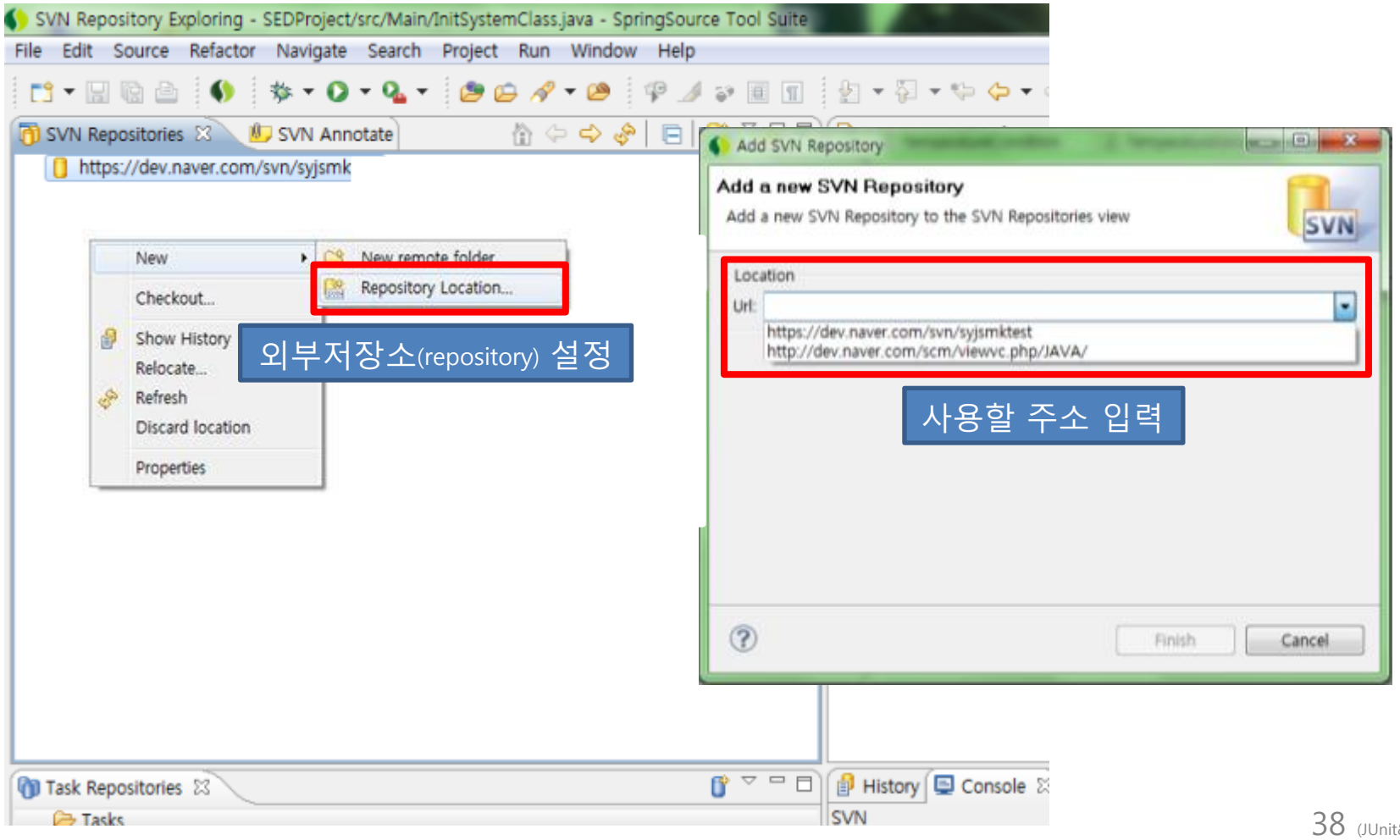
Subclipse

- ***Subclipse - Eclipse 내장 플러그인***
  - Eclipse 내에 Subversion 과 같은 역할을 하는 플러그인



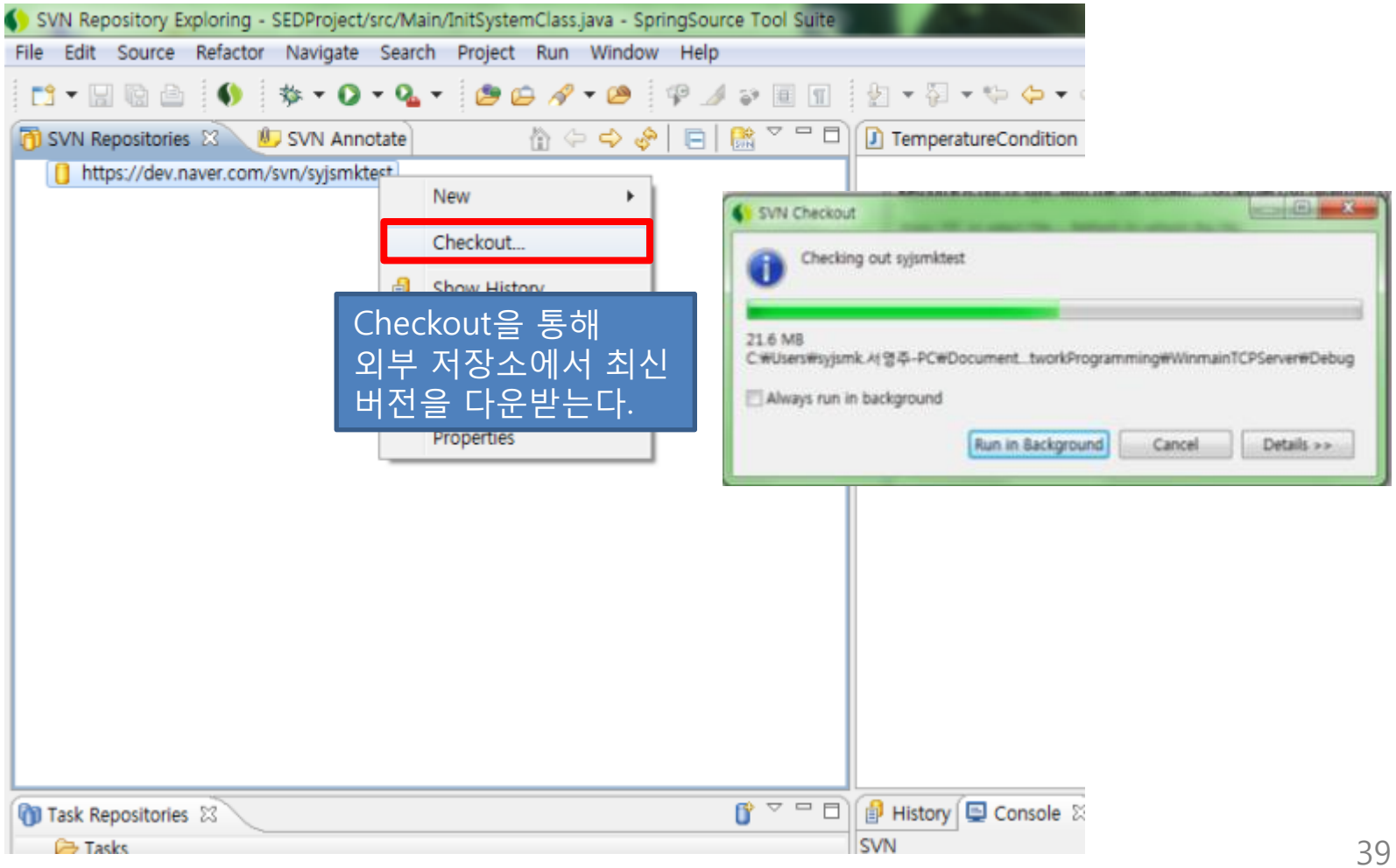
Subclipse

- *Eclipse 내장 플러그인*



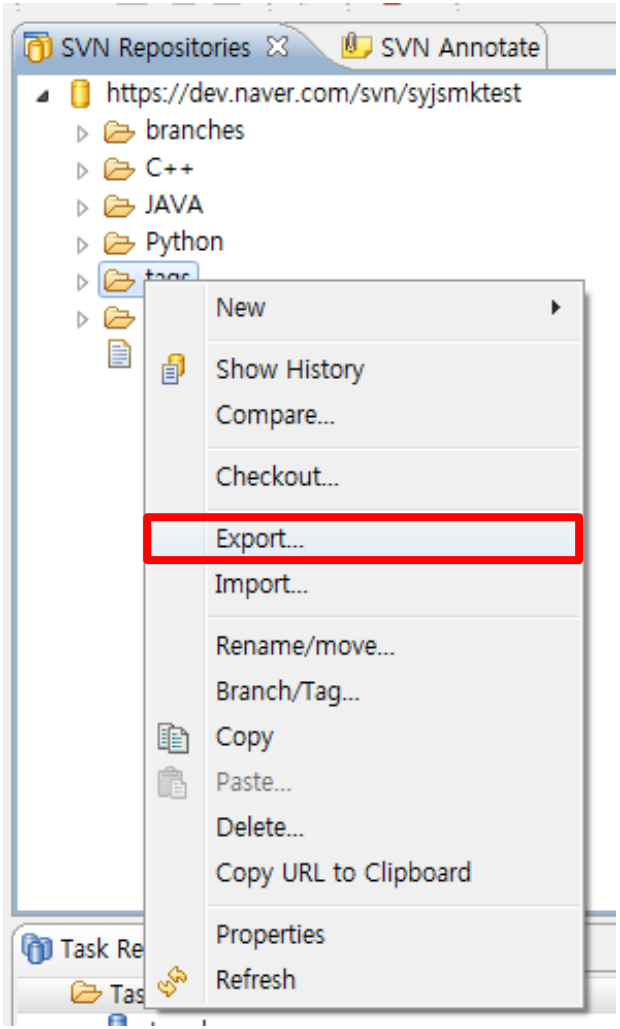
Subclipse

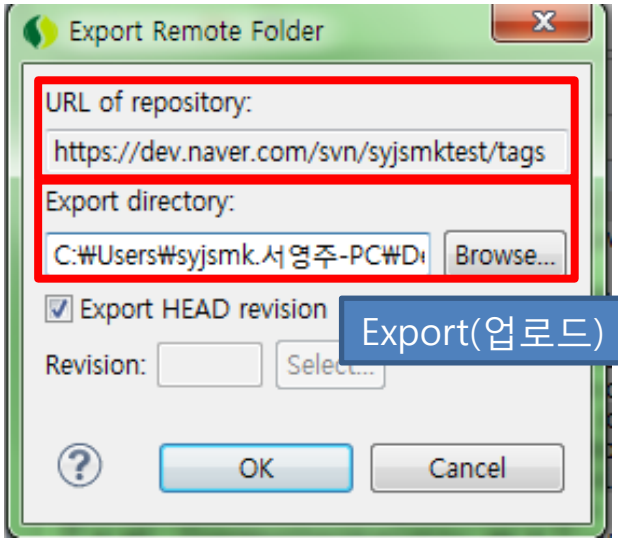
- *Eclipse 내장 플러그인*



Subclipse

- *Eclipse 내장 플러그인*

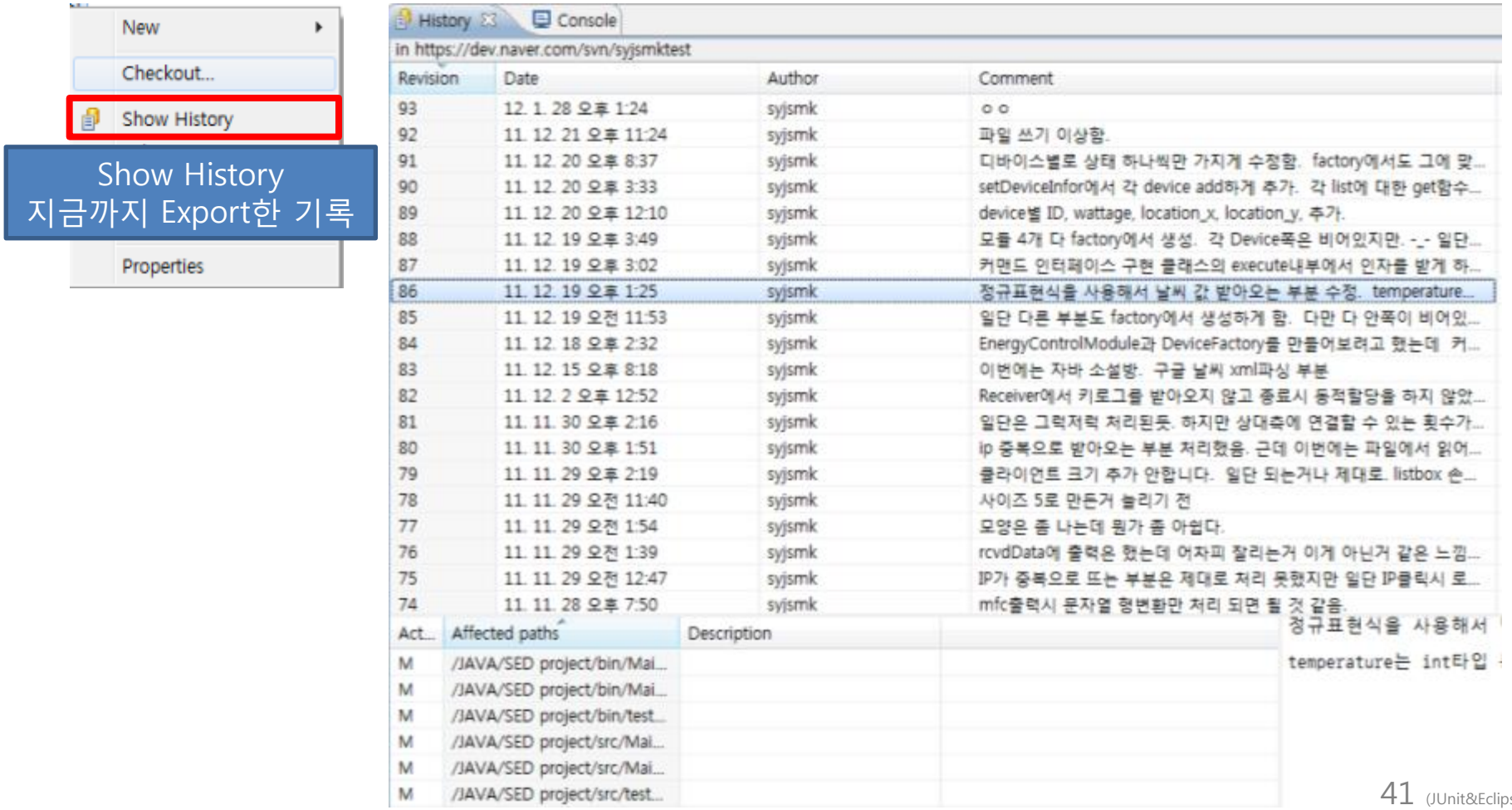




Export(업로드) 할 폴더 설정

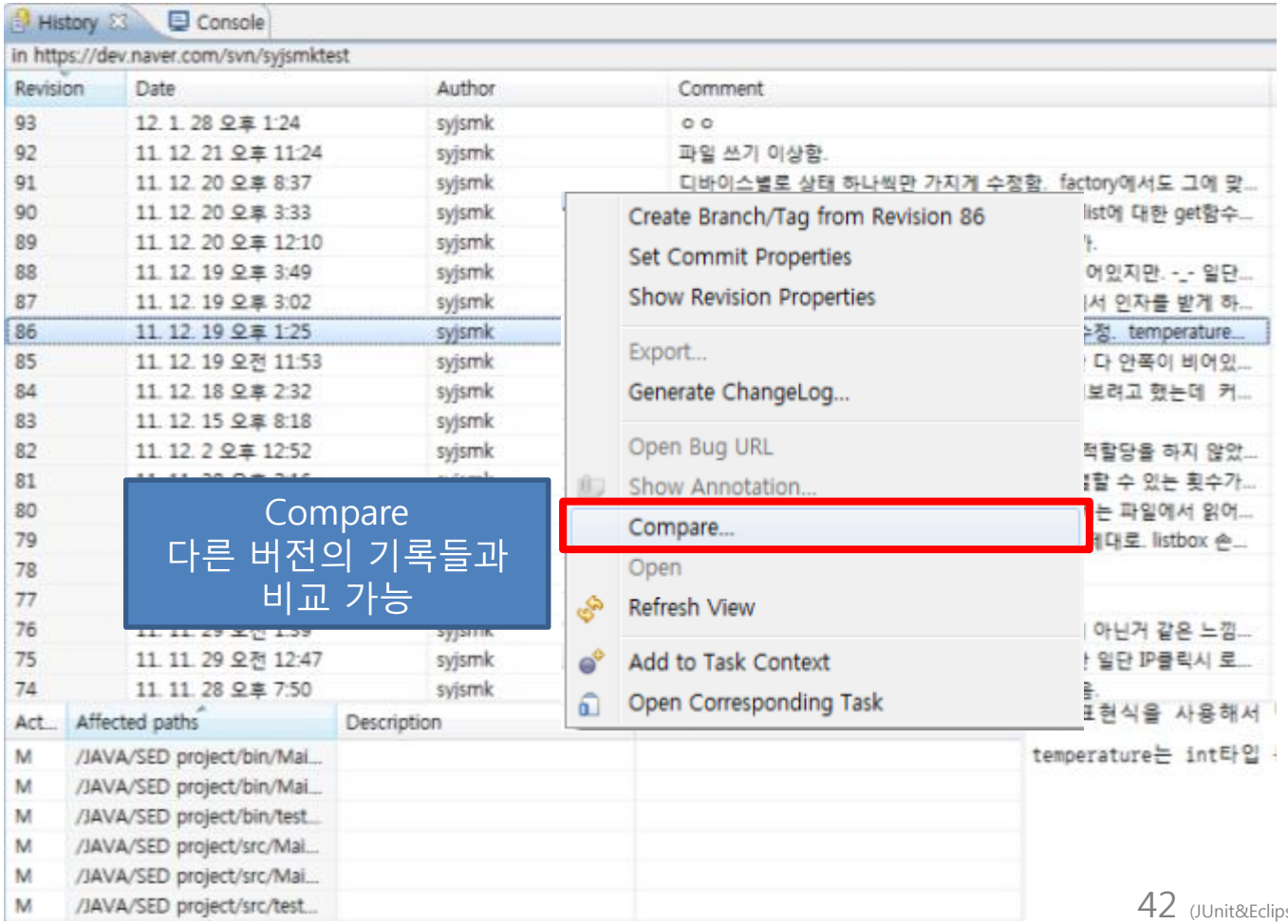
Subclipse

- ***Eclipse 내장 플러그인***

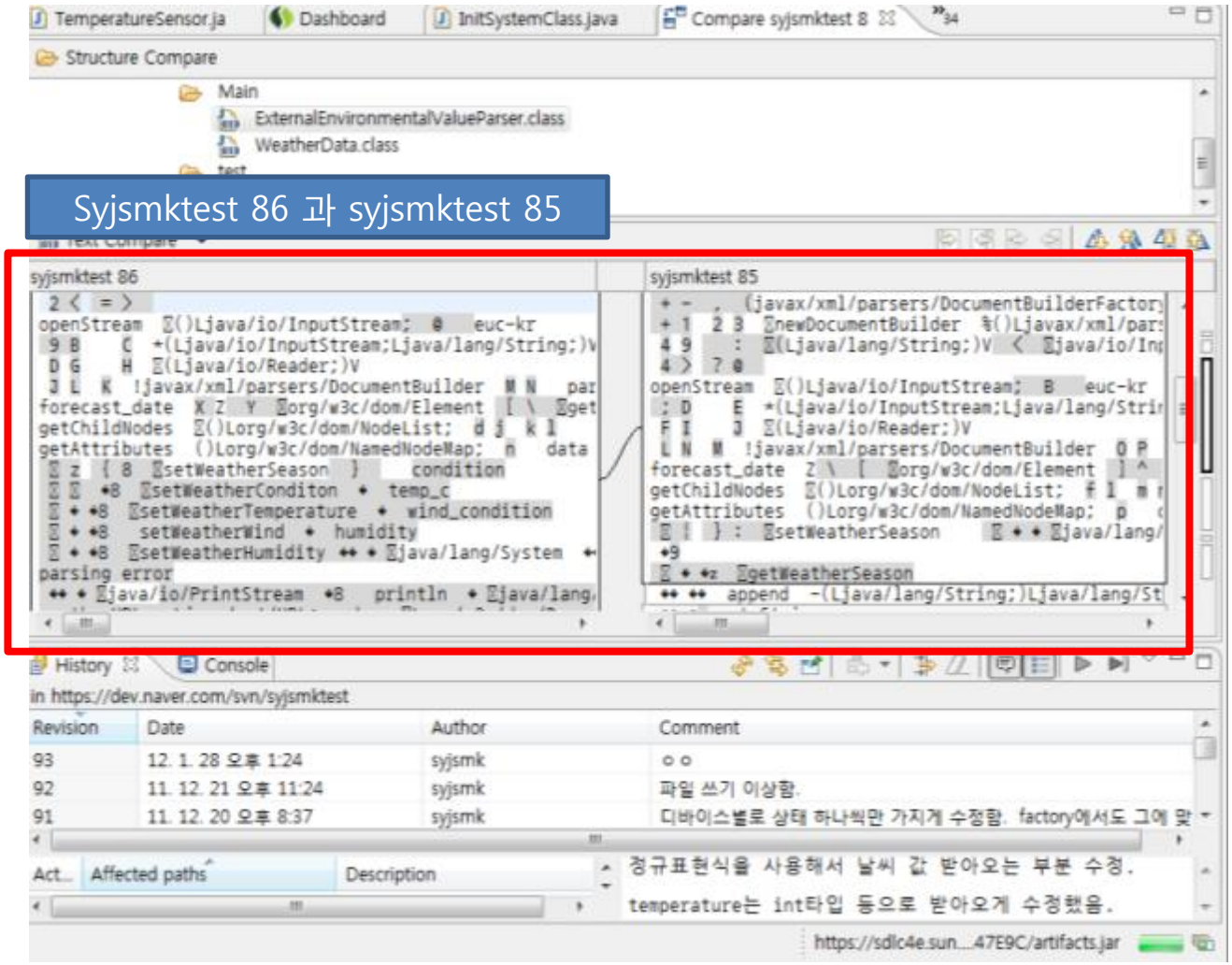


Subclipse

- *Eclipse 내장 플러그인*



Subclipse


TemperatureSensor.ja
Dashboard
InitSystemClass.java
Compare syjsmktest 8
Structure Compare
Main
ExternalEnvironmentalValueParser.class
WeatherData.class
Syjsmktest 86 과 syjsmktest 85
syjsmktest 86
syjsmktest 85
History
Console
in https://dev.naver.com/svn/syjsmktest
Revision
Date
Author
Comment
93
12. 1. 28 오후 1:24
syjsmk
ㅇㅇ
92
11. 12. 21 오후 11:24
syjsmk
파일 쓰기 이상함.
91
11. 12. 20 오후 8:37
syjsmk
디바이스별로 상태 하나씩만 가지게 수정함. factory에서도 그에 맞
Act...
Affected paths
Description
정규표현식을 사용해서 날씨 값 받아오는 부분 수정.
temperature는 int타입 등으로 받아오게 수정했음.

TPTP

◆ *What is TPTP??*

- Test & Performance Tools Platform 프로젝트의 약자로 이클립스에서 오픈소스로 진행중인 테스트 및 성능 관련 툴을 위한 플랫폼

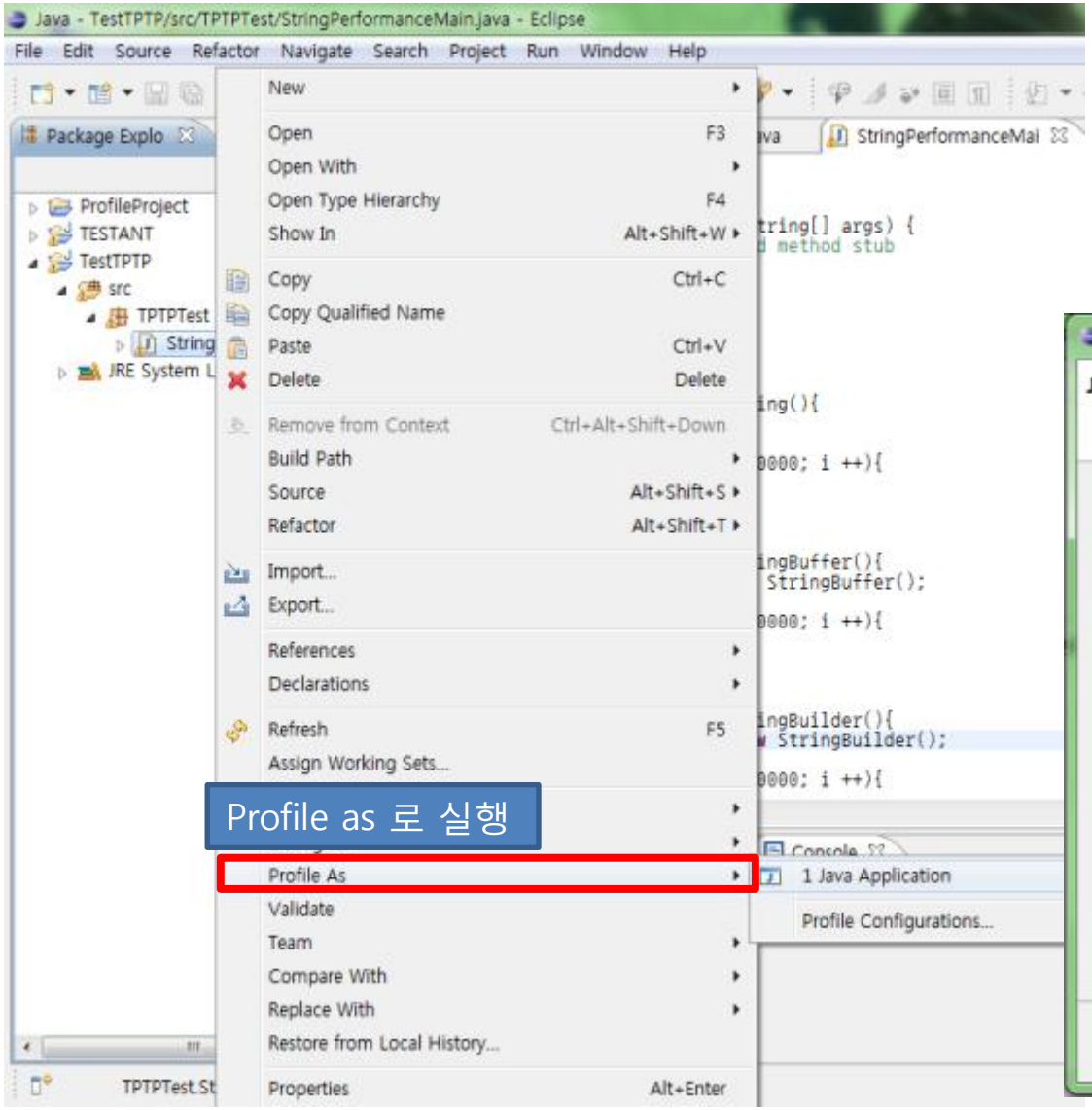
◆ *사용목적*

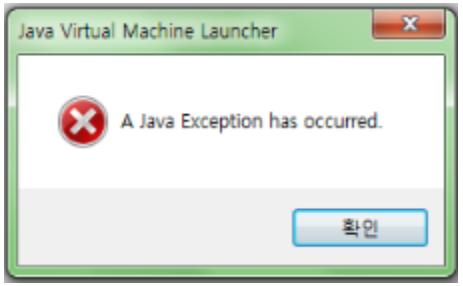
- 소스레벨의 분석을 위한 툴. 느린 메소드, 느린 클래스를 찾기 위함.

◆ *주요기능*

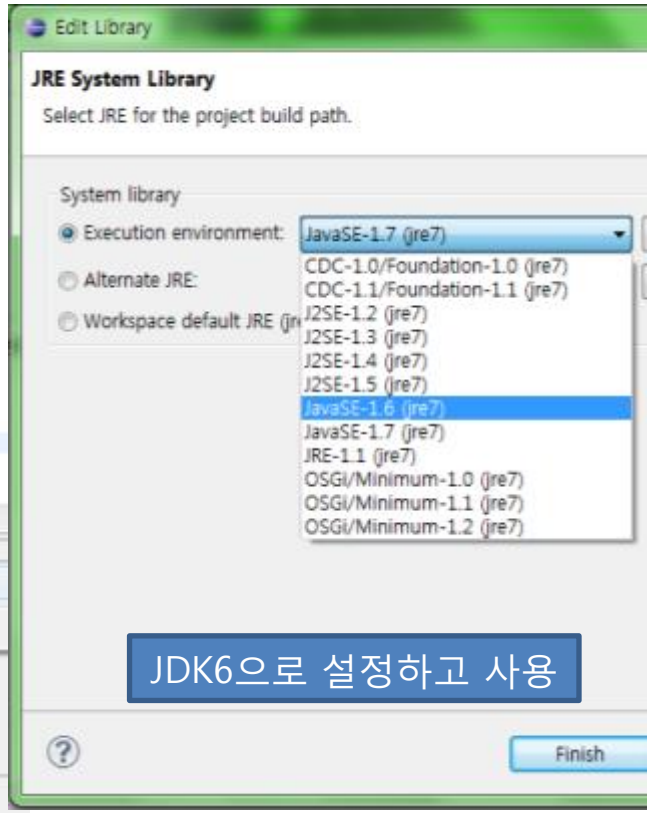
- 메모리 사용량을 개체, 클래스, 소스의 라인단위까지 분석
- 모니터링, 테스팅 자동화, 프로파일 등 애플리케이션의 문제점을 찾고 해결하는데 도움이 되는 기능을 제공.

TPTP



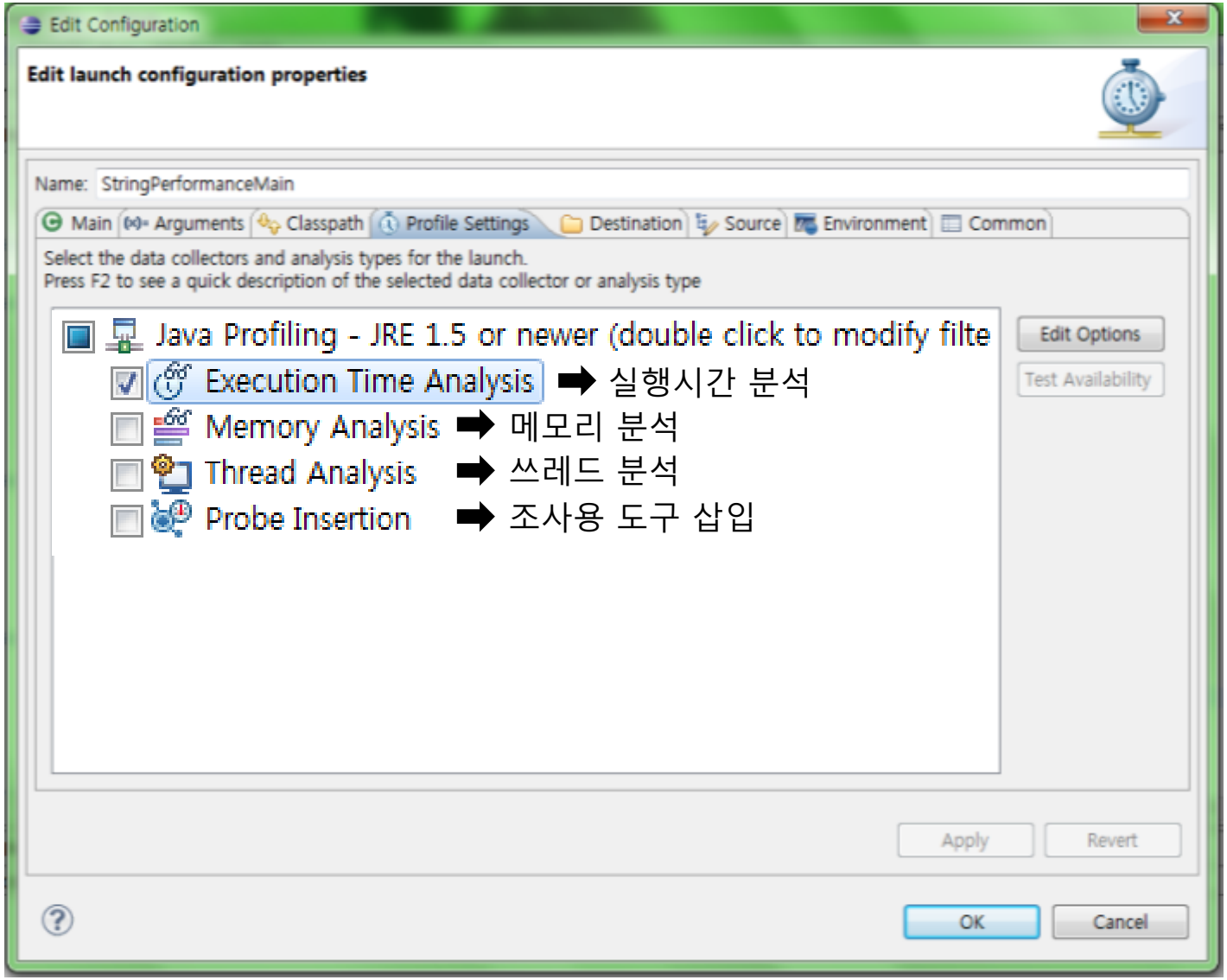


JDK7 에서는 에러 발생

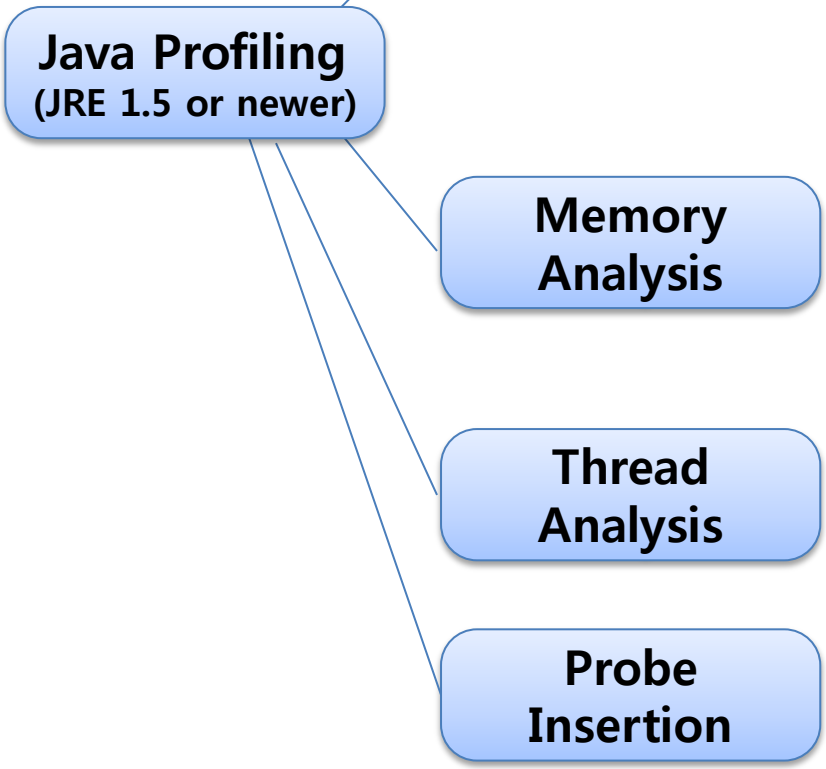


JDK6으로 설정하고 사용

TPTP


Edit Configuration
Edit launch configuration properties
Name: StringPerformanceMain
Main
Arguments
Classpath
Profile Settings
Destination
Source
Environment
Common
Select the data collectors and analysis types for the launch.
Press F2 to see a quick description of the selected data collector or analysis type
Java Profiling - JRE 1.5 or newer (double click to modify filte
Execution Time Analysis ➡ 실행시간 분석
Memory Analysis ➡ 메모리 분석
Thread Analysis ➡ 쓰레드 분석
Probe Insertion ➡ 조사용 도구 삽입
Edit Options
Test Availability
Apply
Revert
OK
Cancel

TPTP



- **Execution Flow**
- **Execution Statistics**
  - Session Summary / Execution Statistics / Call Tree / Method Invocation Details / Method Invocation
- **UML 2 Class Interaction**
- **UML 2 thread Interaction**

- **Object Allocation**

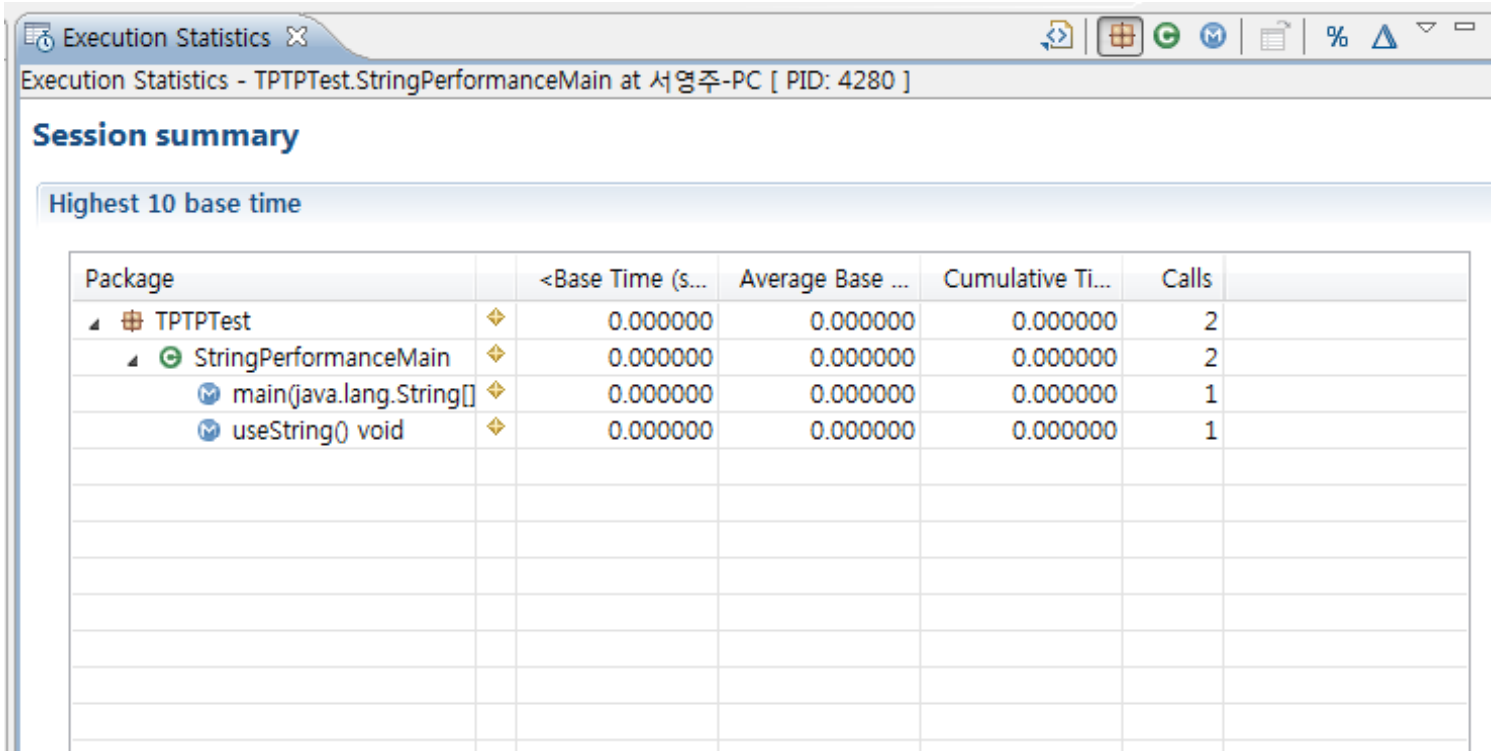
- **Thread Statistics**
- **Monitor Statistics**
- **Threads Visualizer**

TPTP

◆ *Session Summary(실행시간 분석결과)*

- 메소드 단위까지 수행시간과 호출된 횟수까지 표시
  → 프로그램의 성능에 영향을 많이 주는 메소드를 찾을수 있음.

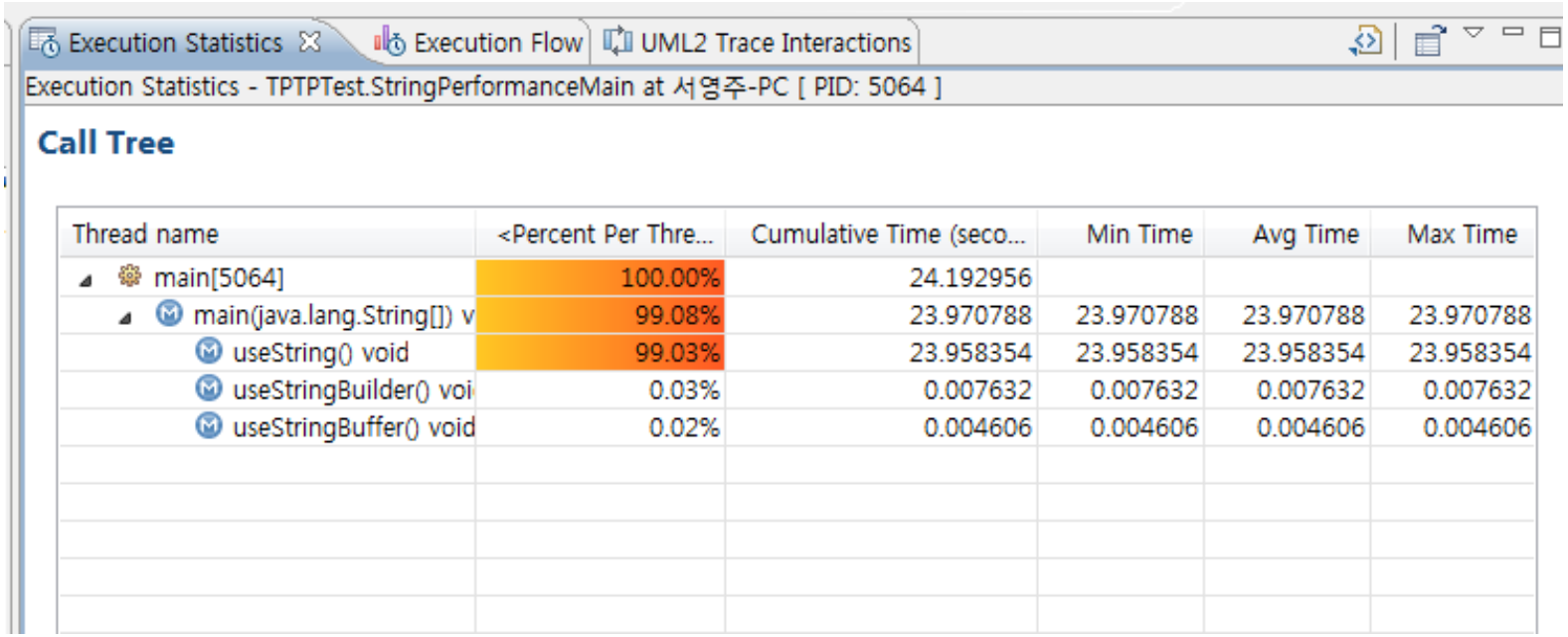
- 실행시간의 점유율과 호출 실행시간을 모니터링할 수 있게 해줌



TPTP 사용법

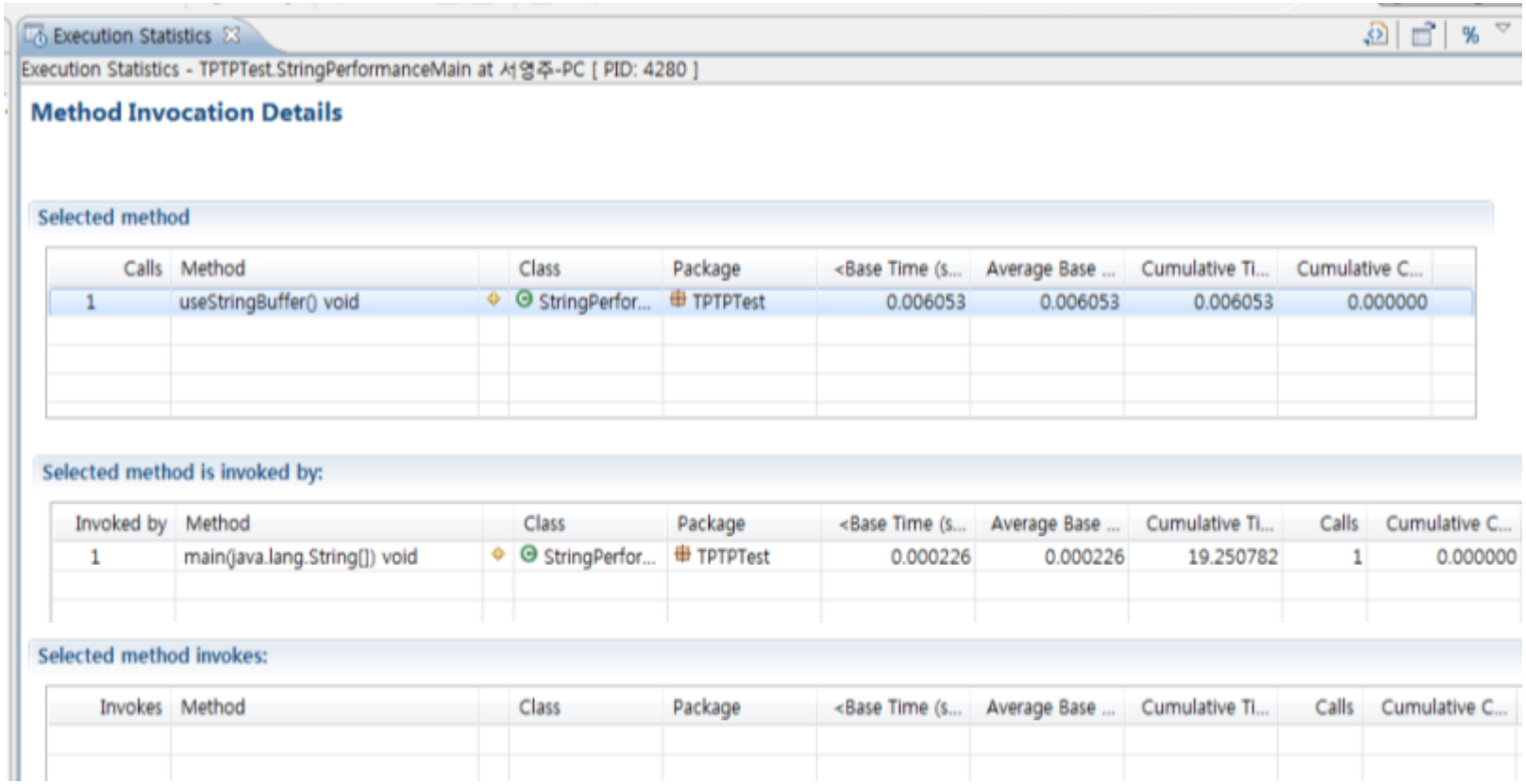
◆ *Call Tree*

- 메소드가 호출된 정보를 그래프로 보여준다.



TPTP 사용법

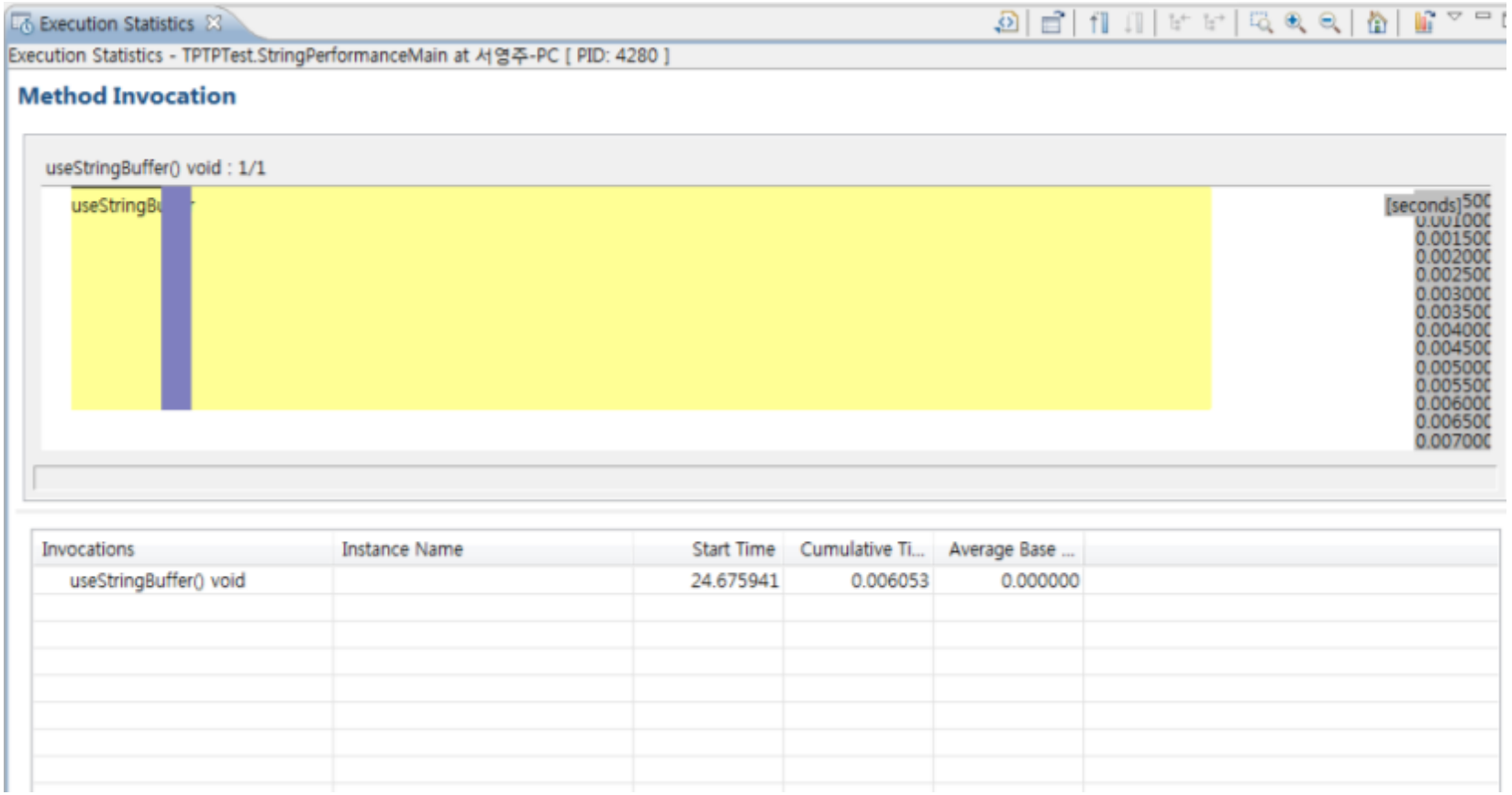
- ***Method Invocation details*** *(메소드 호출의 상세 정보)*
  - Session Summary나 Execution Static에서 특정 메소드를 클릭시 표시
  - 실행시간, 호출시간 및 선택된 메소드와 연관된 메소드들의 정보까지 표시



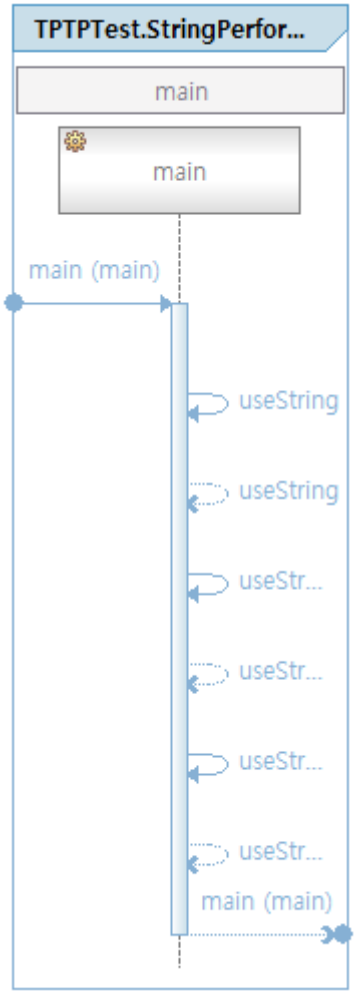
TPTP 사용법

◆ *Method Invocation*

- 해당 메소드의 실행 점유율을 보여줌



TPTP 사용법



- ***Trace Interaction*** *(함수호출 시퀀스 다이어그램)*

- ***Memory Statistics****(메모리 사용 통계)*

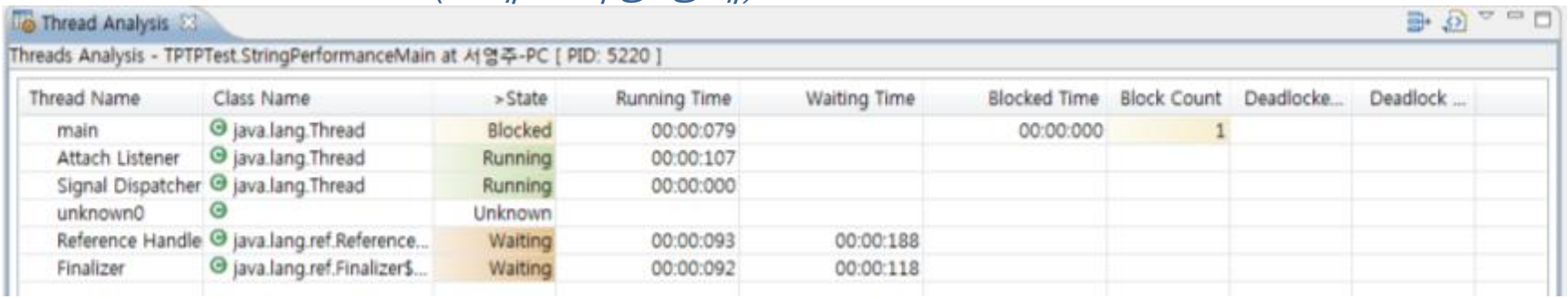
Memory Statistics

Filter: No filter. Click here to set filter

>Class Name	Package	Live Instances	Active Size (bytes)	Total Instances	Total Size (bytes)	Avg. Age
main	main	1	8	1	8	0

TPTP 사용법

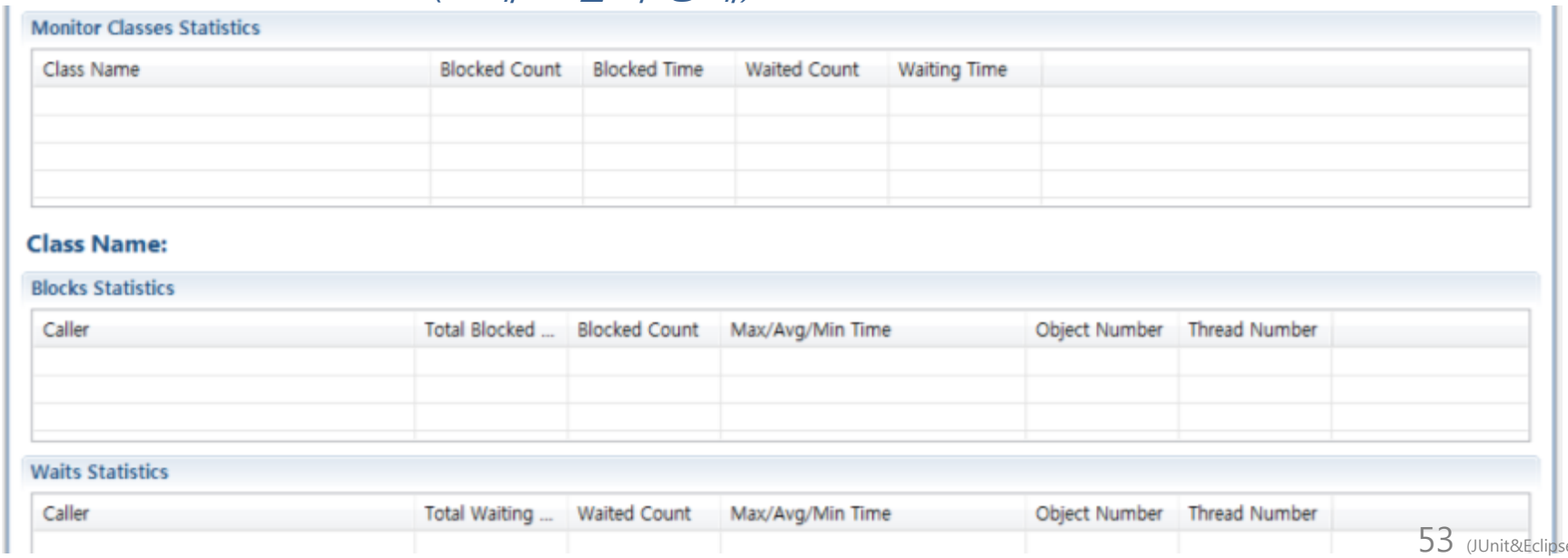
- ***Thread statistics*** *(쓰레드 사용 통계)*


Thread Analysis

Threads Analysis - TPTPTest.StringPerformanceMain at 서영주-PC [PID: 5220]

Thread Name	Class Name	>State	Running Time	Waiting Time	Blocked Time	Block Count	Deadlocke...	Deadlock ...
main	java.lang.Thread	Blocked	00:00:079		00:00:000	1		
Attach Listener	java.lang.Thread	Running	00:00:107					
Signal Dispatcher	java.lang.Thread	Running	00:00:000					
unknown0		Unknown						
Reference Handle	java.lang.ref.Reference...	Waiting	00:00:093	00:00:188				
Finalizer	java.lang.ref.Finalizer$...	Waiting	00:00:092	00:00:118				

- ***Thread statistics*** *(스레드 감시 통계)*


Monitor Classes Statistics

Class Name	Blocked Count	Blocked Time	Waited Count	Waiting Time

Class Name:

Blocks Statistics

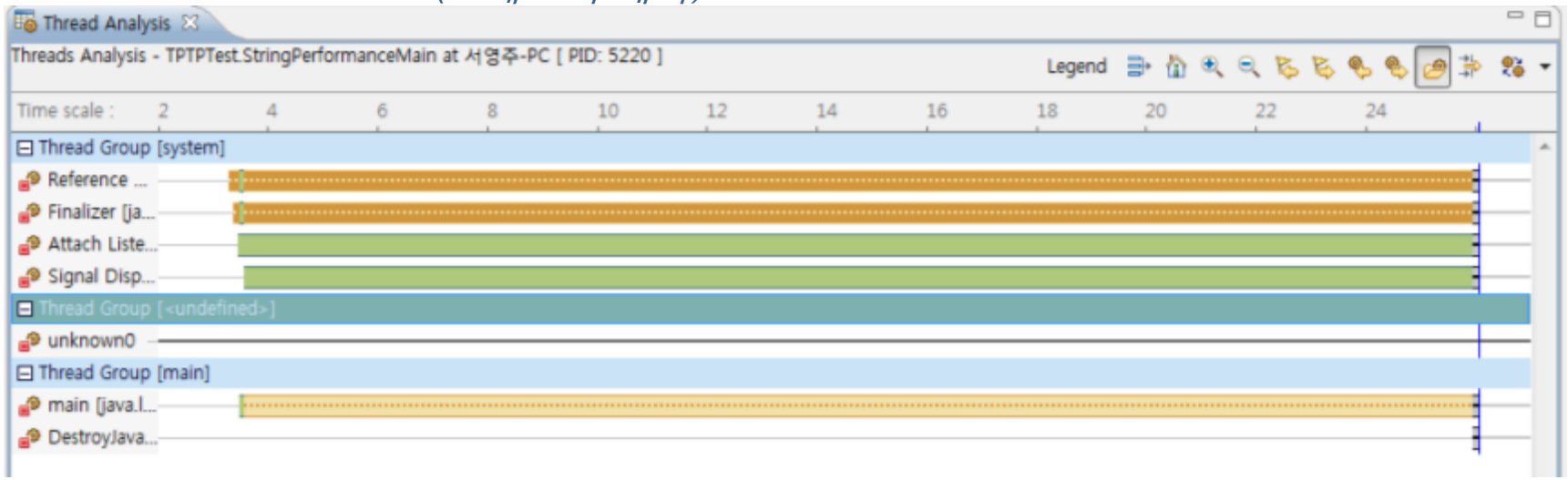
Caller	Total Blocked ...	Blocked Count	Max/Avg/Min Time	Object Number	Thread Number

Waits Statistics

Caller	Total Waiting ...	Waited Count	Max/Avg/Min Time	Object Number	Thread Number

TPTP 사용법

- ***Thread statistics*** *(쓰레드 구체화)*



Ant

◆ *Ant??*

- 빌드 도구 소프트웨어
- 태스크라는 xml요소를 빌드파일을 작성하여 빌드 규칙 생성

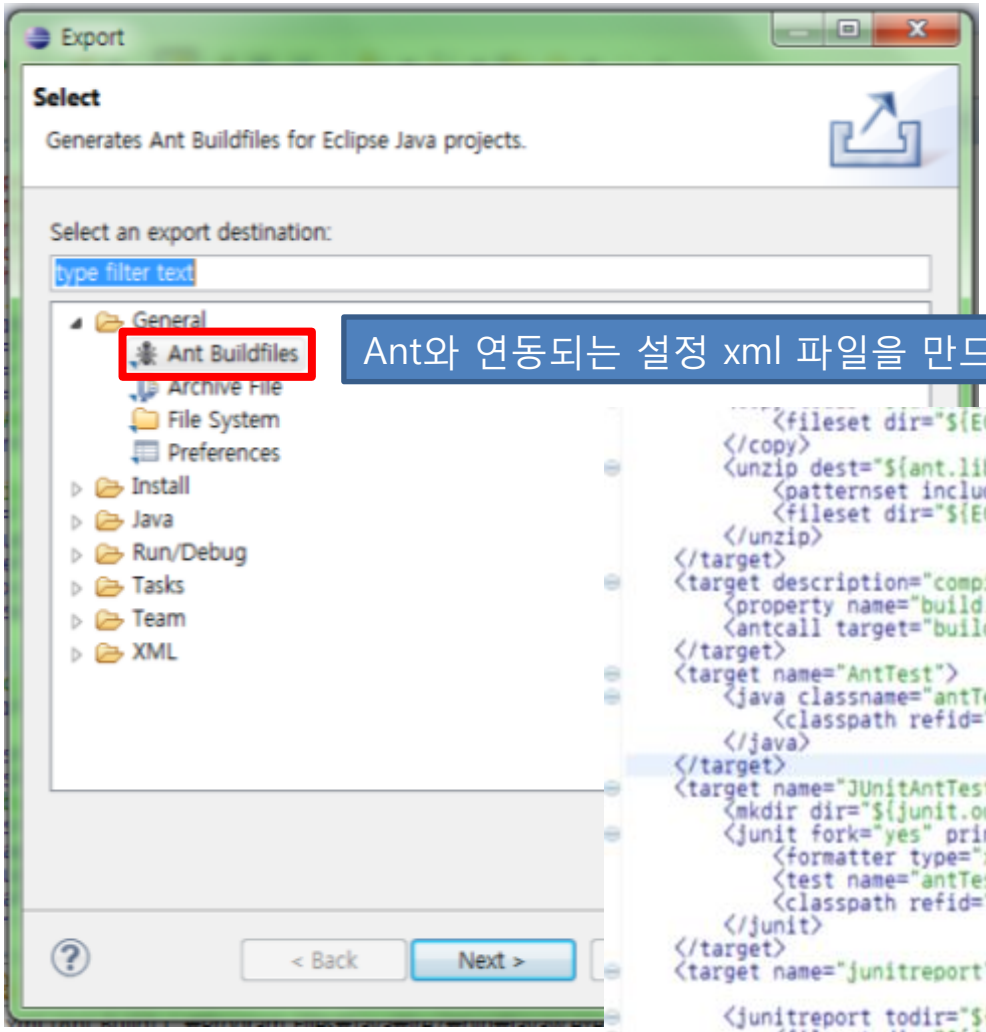
◆ *사용목적*

- 원래는 Apache Tomcat을 빌드하기 위해 개발됨.
  Xml요소를 이용해 만들어진 태스크를 실행하고, Hudson과 연동하여
  CTIP환경 구축 가능.

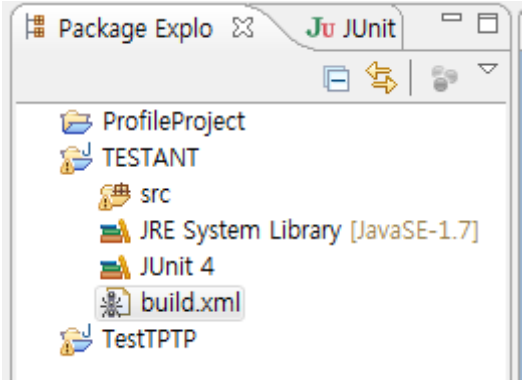
◆ *주요기능*

- Javac : java 소스코드를 컴파일
- Junit : 테스트 프레임워크 Junit을 사용하여 java프로그램을 테스트
- CVS : CVS연결을 시작하여 CVS저장소에서 체크아웃, 커밋, 업데이트

Ant

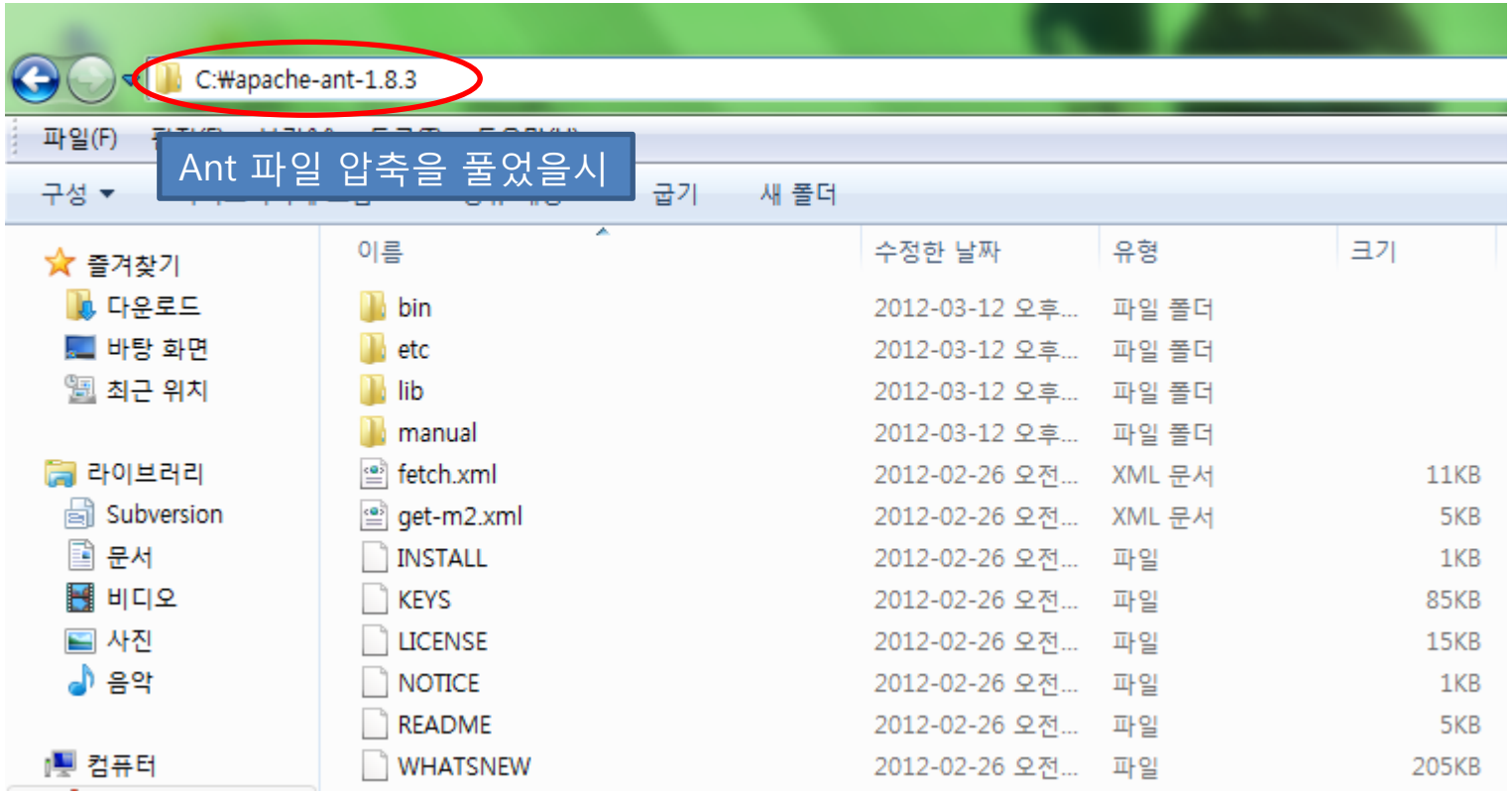


Ant와 연동되는 설정 xml 파일을 만드는 기능이 있음.

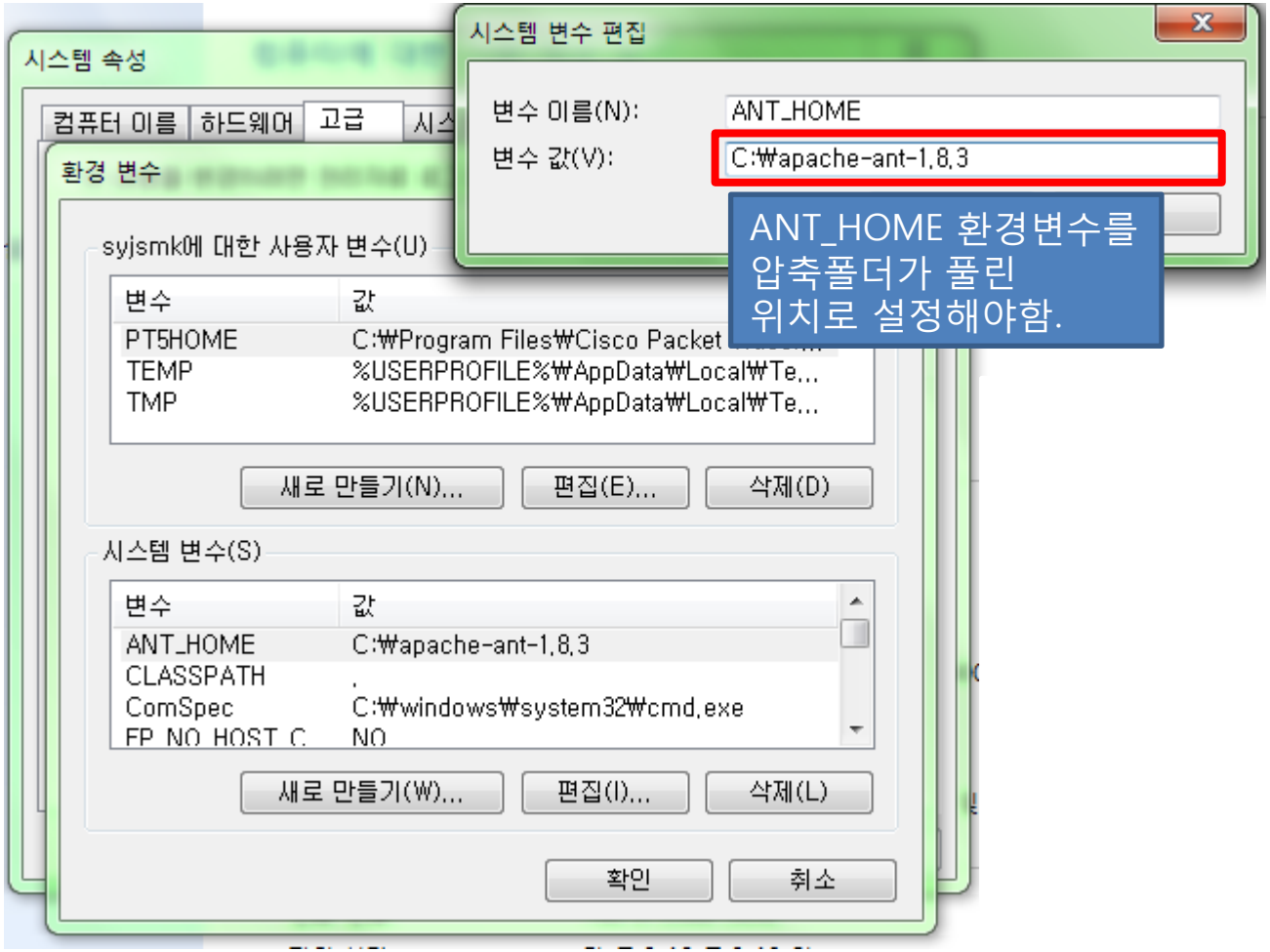



```
            <fileset dir="${ECLIPSE_HOME}/plugins" includes="org.eclipse.jdt.core_*.jar"/>
        </copy>
        <unzip dest="${ant.library.dir}">
            <patternset includes="jdtCompilerAdapter.jar"/>
            <fileset dir="${ECLIPSE_HOME}/plugins" includes="org.eclipse.jdt.core_*.jar"/>
        </unzip>
    </target>
    <target description="compile project with Eclipse compiler" name="build-eclipse-compiler">
        <property name="build.compiler" value="org.eclipse.jdt.core.JDTCompilerAdapter"/>
        <antcall target="build"/>
    </target>
    <target name="AntTest">
        <java classname="antTest.AntTest" failonerror="true" fork="yes">
            <classpath refid="TESTANT.classpath"/>
        </java>
    </target>
    <target name="JUnitAntTest">
        <mkdir dir="${junit.output.dir}"/>
        <junit fork="yes" printsummary="withOutAndErr">
            <formatter type="xml"/>
            <test name="antTest.JUnitAntTest" todir="${junit.output.dir}"/>
            <classpath refid="TESTANT.classpath"/>
        </junit>
    </target>
    <target name="junitreport">

        <junitreport todir="${junit.output.dir}">
            <fileset dir="${junit.output.dir}">
                <include name="TEST-*.xml"/>
            </fileset>
            <report format="frames" todir="${junit.output.dir}"/>
        </junitreport>
```

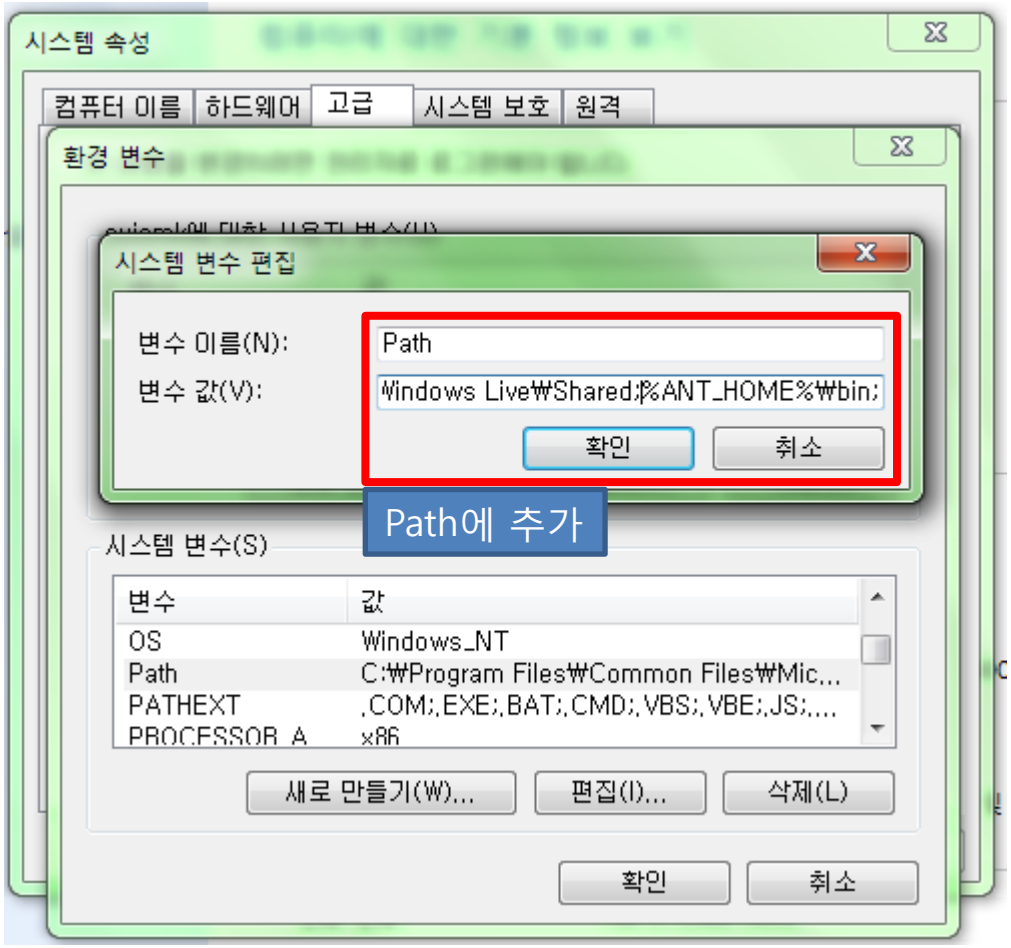
Ant


C:₩apache-ant-1.8.3
파일(F)
Ant 파일 압축을 풀었을시
구성
굽기
새 폴더
즐겨찾기
다운로드
바탕 화면
최근 위치
라이브러리
Subversion
문서
비디오
사진
음악
컴퓨터
이름
수정한 날짜
유형
크기
bin 2012-03-12 오후... 파일 폴더
etc 2012-03-12 오후... 파일 폴더
lib 2012-03-12 오후... 파일 폴더
manual 2012-03-12 오후... 파일 폴더
fetch.xml 2012-02-26 오전... XML 문서 11KB
get-m2.xml 2012-02-26 오전... XML 문서 5KB
INSTALL 2012-02-26 오전... 파일 1KB
KEYS 2012-02-26 오전... 파일 85KB
LICENSE 2012-02-26 오전... 파일 15KB
NOTICE 2012-02-26 오전... 파일 1KB
README 2012-02-26 오전... 파일 5KB
WHATSNEW 2012-02-26 오전... 파일 205KB

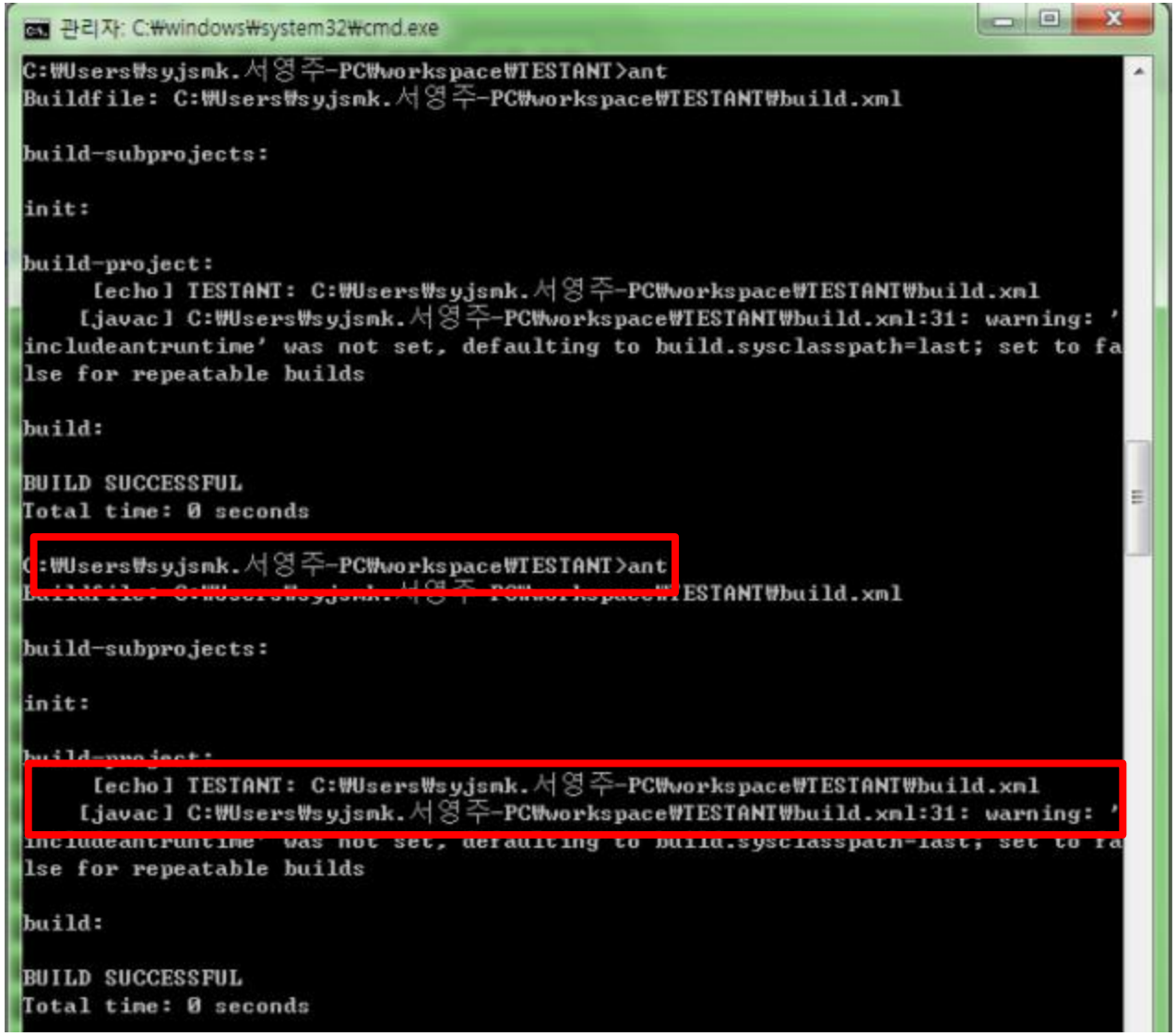
Ant


시스템 속성
컴퓨터 이름
하드웨어
고급
환경 변수
syjsmk에 대한 사용자 변수(U)
변수
값
PT5HOME
C:₩Program Files₩Cisco Packet
TEMP
%USERPROFILE%₩AppData₩Local₩Te...
TMP
%USERPROFILE%₩AppData₩Local₩Te...
새로 만들기(N)...
편집(E)...
삭제(D)
시스템 변수(S)
변수
값
ANT_HOME
C:₩apache-ant-1.8.3
CLASSPATH
.
ComSpec
C:₩windows₩system32₩cmd.exe
FP_NO_HOST_C...
NO
새로 만들기(W)...
편집(I)...
삭제(L)
확인
취소
시스템 변수 편집
변수 이름(N):
ANT_HOME
변수 값(V):
C:₩apache-ant-1.8.3
ANT_HOME 환경변수를
압축폴더가 풀린
위치로 설정해야함.

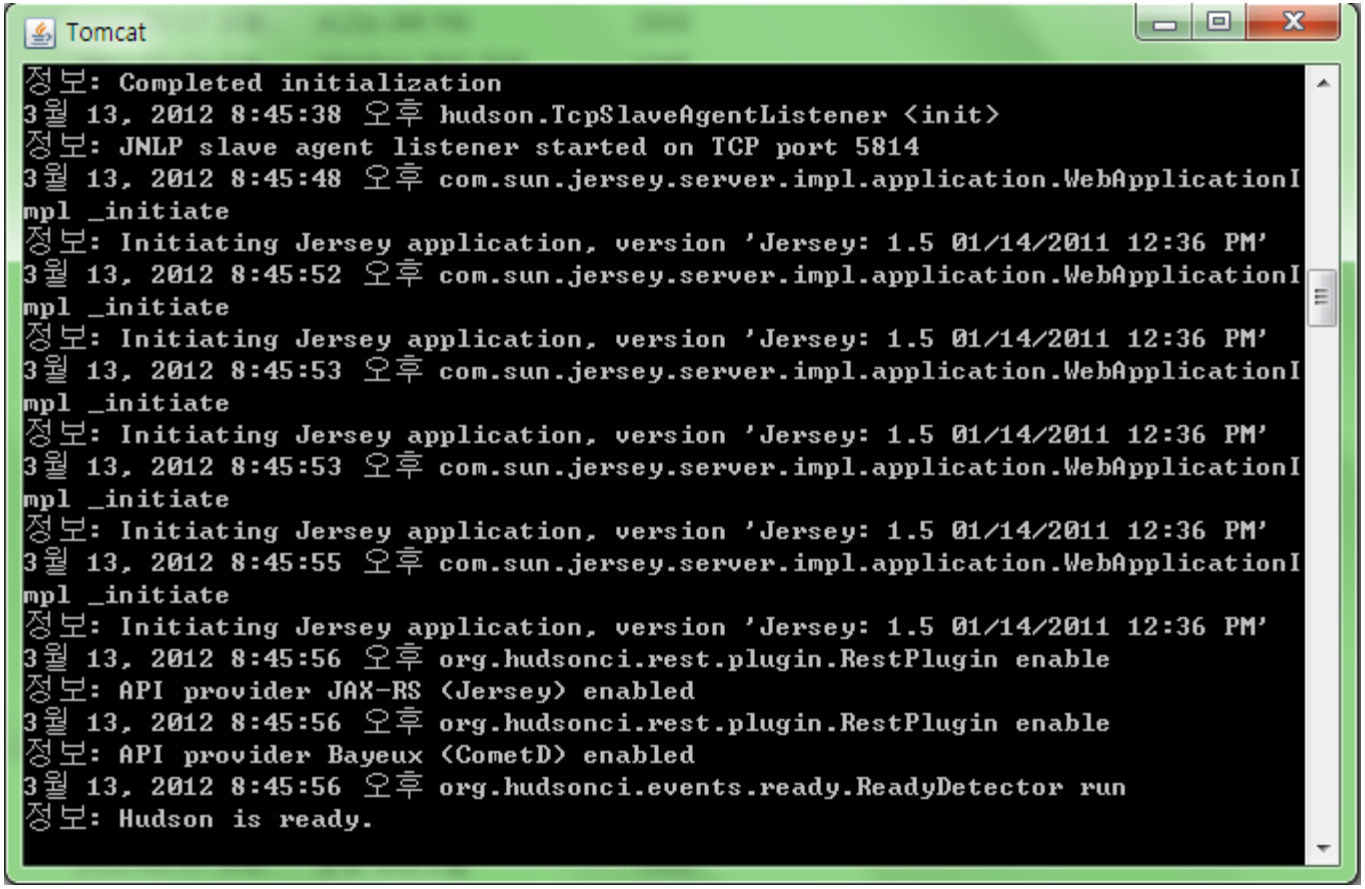
Ant


시스템 속성
컴퓨터 이름
하드웨어
고급
시스템 보호
원격
환경 변수
시스템 변수 편집
변수 이름(N):
Path
변수 값(V):
Windows Live₩Shared;%ANT_HOME%₩bin;
확인
취소
Path에 추가
시스템 변수(S)
변수
값
OS
Windows_NT
Path
C:₩Program Files₩Common Files₩Mic...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;...
PROCESSOR_A
x86
새로 만들기(W)...
편집(I)...
삭제(L)
확인
취소

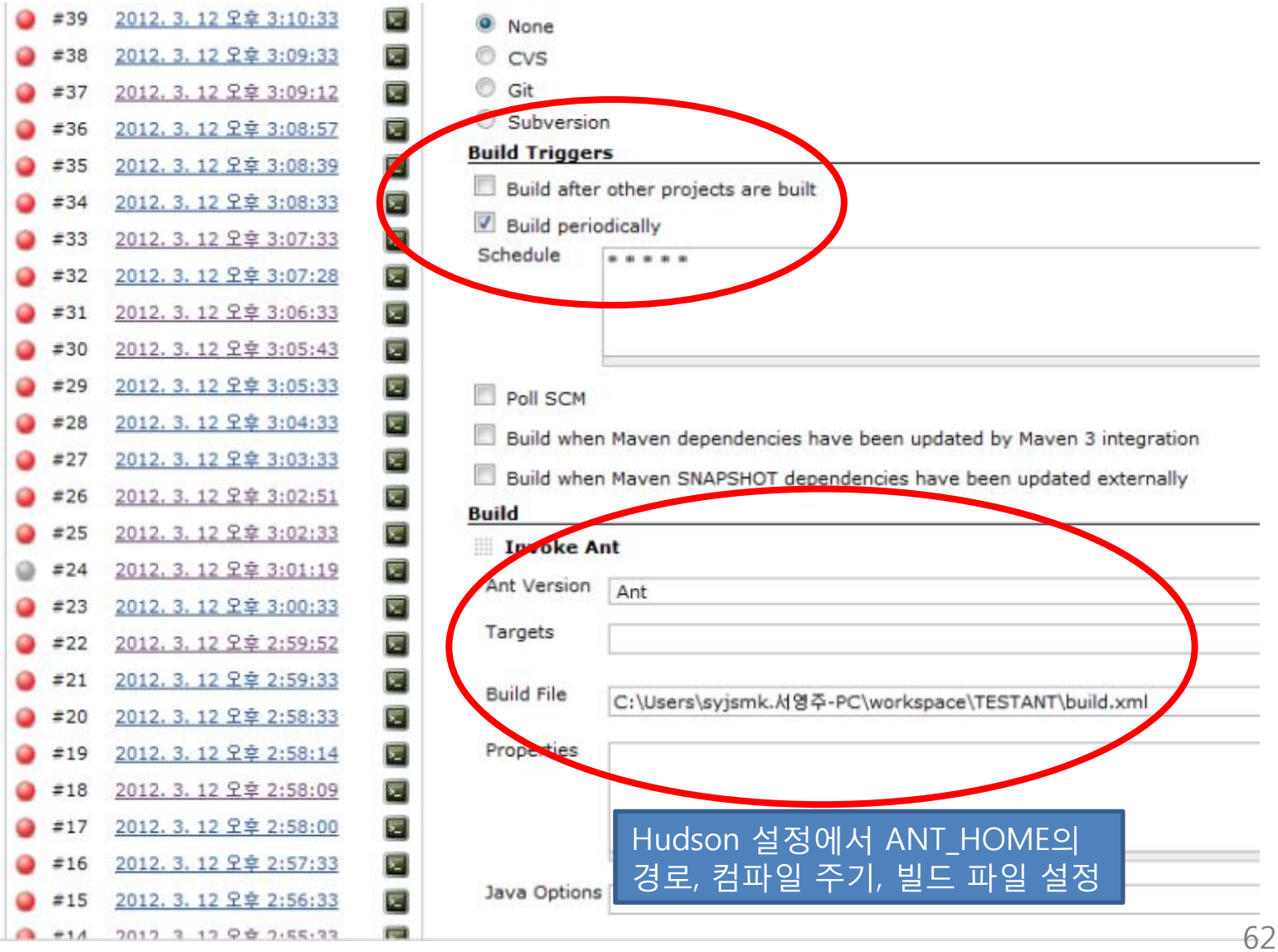
Ant


관리자: C:₩windows₩system32₩cmd.exe
C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT>ant
Buildfile: C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml
build-subprojects:
init:
build-project:
[echo] TESTANT: C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml
[javac] C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml:31: warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds
build:
BUILD SUCCESSFUL
Total time: 0 seconds
C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT>ant
₩TESTANT₩build.xml
build-subprojects:
init:
[echo] TESTANT: C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml
[javac] C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml:31: warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds
build:
BUILD SUCCESSFUL
Total time: 0 seconds

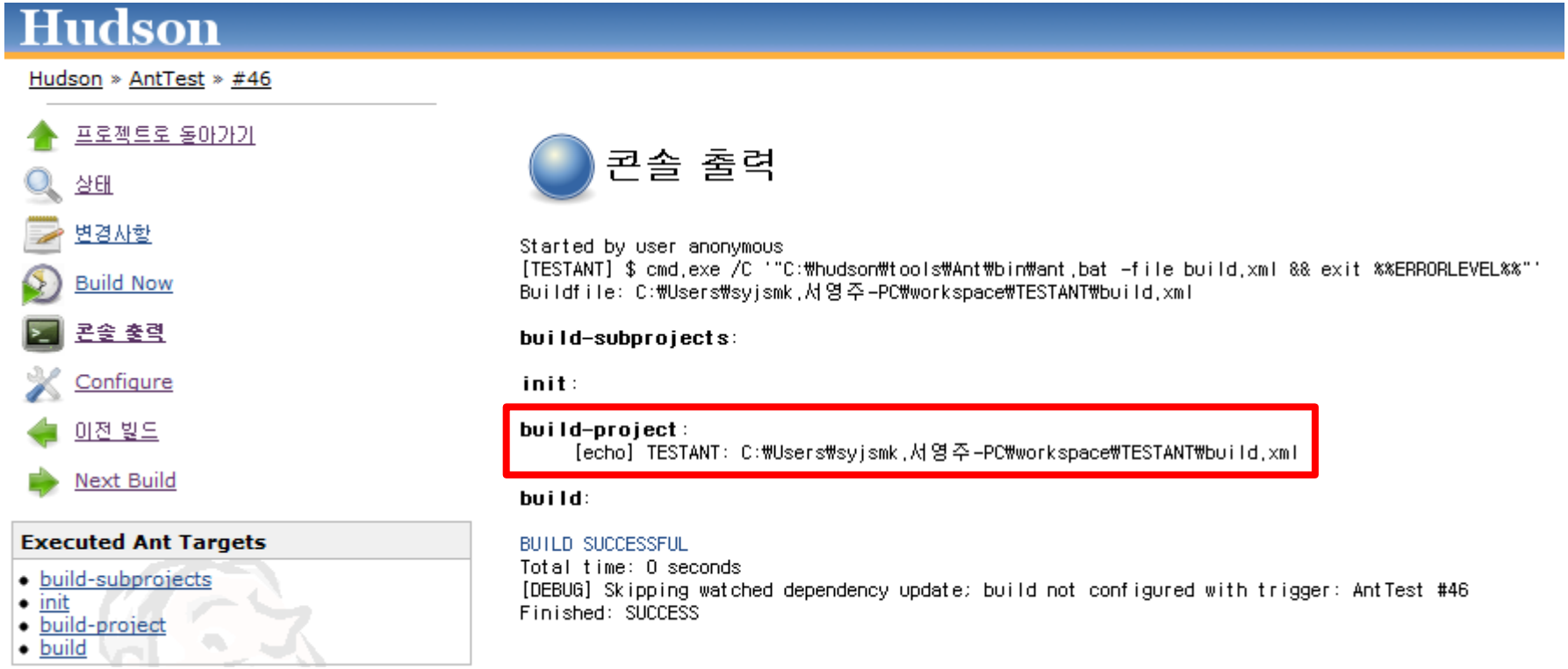
Ant


Tomcat
정보: Completed initialization
3월 13, 2012 8:45:38 오후 hudson.TcpSlaveAgentListener <init>
정보: JNLP slave agent listener started on TCP port 5814
3월 13, 2012 8:45:48 오후 com.sun.jersey.server.impl.application.WebApplicationI
mpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.5 01/14/2011 12:36 PM'
3월 13, 2012 8:45:52 오후 com.sun.jersey.server.impl.application.WebApplicationI
mpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.5 01/14/2011 12:36 PM'
3월 13, 2012 8:45:53 오후 com.sun.jersey.server.impl.application.WebApplicationI
mpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.5 01/14/2011 12:36 PM'
3월 13, 2012 8:45:53 오후 com.sun.jersey.server.impl.application.WebApplicationI
mpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.5 01/14/2011 12:36 PM'
3월 13, 2012 8:45:55 오후 com.sun.jersey.server.impl.application.WebApplicationI
mpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.5 01/14/2011 12:36 PM'
3월 13, 2012 8:45:56 오후 org.hudsonci.rest.plugin.RestPlugin enable
정보: API provider JAX-RS (Jersey) enabled
3월 13, 2012 8:45:56 오후 org.hudsonci.rest.plugin.RestPlugin enable
정보: API provider Bayeux (CometD) enabled
3월 13, 2012 8:45:56 오후 org.hudsonci.events.ready.ReadyDetector run
정보: Hudson is ready.

Ant


#39 2012. 3. 12 오후 3:10:33
#38 2012. 3. 12 오후 3:09:33
#37 2012. 3. 12 오후 3:09:12
#36 2012. 3. 12 오후 3:08:57
#35 2012. 3. 12 오후 3:08:39
#34 2012. 3. 12 오후 3:08:33
#33 2012. 3. 12 오후 3:07:33
#32 2012. 3. 12 오후 3:07:28
#31 2012. 3. 12 오후 3:06:33
#30 2012. 3. 12 오후 3:05:43
#29 2012. 3. 12 오후 3:05:33
#28 2012. 3. 12 오후 3:04:33
#27 2012. 3. 12 오후 3:03:33
#26 2012. 3. 12 오후 3:02:51
#25 2012. 3. 12 오후 3:02:33
#24 2012. 3. 12 오후 3:01:19
#23 2012. 3. 12 오후 3:00:33
#22 2012. 3. 12 오후 2:59:52
#21 2012. 3. 12 오후 2:59:33
#20 2012. 3. 12 오후 2:58:33
#19 2012. 3. 12 오후 2:58:14
#18 2012. 3. 12 오후 2:58:09
#17 2012. 3. 12 오후 2:58:00
#16 2012. 3. 12 오후 2:57:33
#15 2012. 3. 12 오후 2:56:33
None
CVS
Git
Subversion
Build Triggers
Build after other projects are built
Build periodically
Schedule
* * * * *
Poll SCM
Build when Maven dependencies have been updated by Maven 3 integration
Build when Maven SNAPSHOT dependencies have been updated externally
Build
Invoke Ant
Ant Version
Ant
Targets
Build File
C:\Users\syjsmk.서영주-PC\workspace\TESTANT\build.xml
Properties
Java Options
Hudson 설정에서 ANT_HOME의
경로, 컴파일 주기, 빌드 파일 설정

Ant


Hudson
Hudson » AntTest » #46
프로젝트로 돌아가기
상태
변경사항
Build Now
콘솔 출력
Configure
이전 빌드
Next Build
Executed Ant Targets
build-subprojects
init
build-project
build
콘솔 출력
Started by user anonymous
[TESTANT] $ cmd.exe /C '"C:₩hudson₩tools₩Ant₩bin₩ant.bat -file build.xml && exit %%ERRORLEVEL%%"'
Buildfile: C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml
build-subprojects:
init:
build-project:
[echo] TESTANT: C:₩Users₩syjsmk.서영주-PC₩workspace₩TESTANT₩build.xml
build:
BUILD SUCCESSFUL
Total time: 0 seconds
[DEBUG] Skipping watched dependency update; build not configured with trigger: AntTest #46
Finished: SUCCESS

References.

☆ **Book**

- 테스트 주도 개발 : 고품질 쾌속개발을 위한 TDD 실천법과 도구
- 자바 개발자도 쉽고 즐겁게 배우는 테스팅 이야기

☆ **Mockito**

http://code.google.com/p/mockito/wiki/MockitoFeaturesInKorean

☆ **TPTP**

http://antop.tistory.com/135
http://ksjjang.tistory.com/63
http://cafe.naver.com/junes81.cafe?iframe_url=/ArticleRead.nhn%3Farticleid=3173&
http://www.okjsp.pe.kr/seq/181503

☆ **Ant**

http://nigh.tistory.com/entry/tomcat-%EC%84%A4%EC%A0%95
http://idkbj.tistory.com/17
http://groups.google.com/group/ksug/browse_thread/thread/108d876b51d44795
http://blog.naver.com/PostView.nhn?blogId=kittenjun&logNo=10125029239
http://cafe.naver.com/junes81.cafe?iframe_url=/ArticleRead.nhn%3Farticleid=3248&
http://blog.naver.com/rlaaudtnr8?Redirect=Log&logNo=50036439808&from=postView
http://linuxism.tistory.com/371
http://pmguda.com/479