

Introduction to Static Analysis

Dependable Software Laboratory

Static Analysis

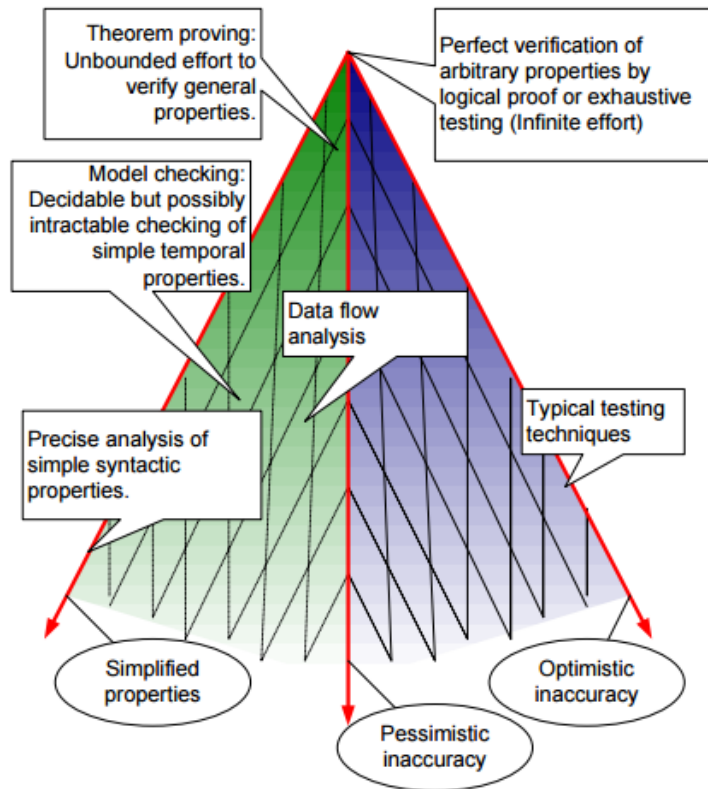
- Static analysis is the process of examining source code prior to compilation
 - Without executing
- Static analysis can diagnose for:
 - **Quality aspects** such as maintainability, reliability, understandability and complexity
 - **Testing** issues
 - **Coding standard** compliance issues
 - Best programming practices and **unsafe programming** constructs and **coding defects**

Static Analysis

- Analyze the program without executing it
 - Doesn't depend on having good test cases or even any test cases
 - Generally, doesn't know what your software is *supposed to do*
 - Looks for violations of reasonable programming
 - Not a replacement for testing
 - Very good at finding problems on **untested paths**
 - But many defects can't be found with static analysis
 - False alarm occurs
 - Generally 30%

A position of the Static Analysis in Verification

- Verification Trade-off Dimensions



- Optimistic inaccuracy
 - We may accept some programs that do not possess the property.
 - It may not detect all violations.
 - Example: Testing
- Pessimistic inaccuracy
 - It is not guaranteed to accept a program even if the program does possess the property being analyzed, because of false alarms.
 - Example: Automated program analysis
- Simplified properties
 - It reduces the degree of freedom by simplifying the property to check.
 - Example: Model Checking

Static Analysis

- Static analysis can be divided 3 levels
- **Level 1: syntax checking of the source code**
 - Rule checking, coding style checking
- **Level 2: quality analysis with translated source code to CFG/DFG form**
 - Semantic analysis, complexity analysis
- **Level 3: static analysis, analyzing critical errors which can be issued during execution**
 - Divided by zero, NULL pointer, Etc.

Level 1

- Rule checking, coding style checking
- Syntax checking by IDE (e.g. *eclipse*, *visual studio*) is a kind of static analysis
- Several kinds of rules
 - Simple rule checking
 - *E.g. Brace location, tab, Etc.*
 - Safe coding rule checking
 - T. A., R.W. WITTY, "SAFE PROGRAMMING," 1978
 - Safe specification and programming(coding) is the simplest way to improve software reliability
 - Proposing several rules for safe software (safe programming)
 - *E.g. infinite loop checking with counter, protecting buffer overflow code*
- It is useful for coding style checking when working as a team
 - Readability
 - Maintainability

```
public void test(){  
  
}  
  
public void test1()  
{  
  
}
```

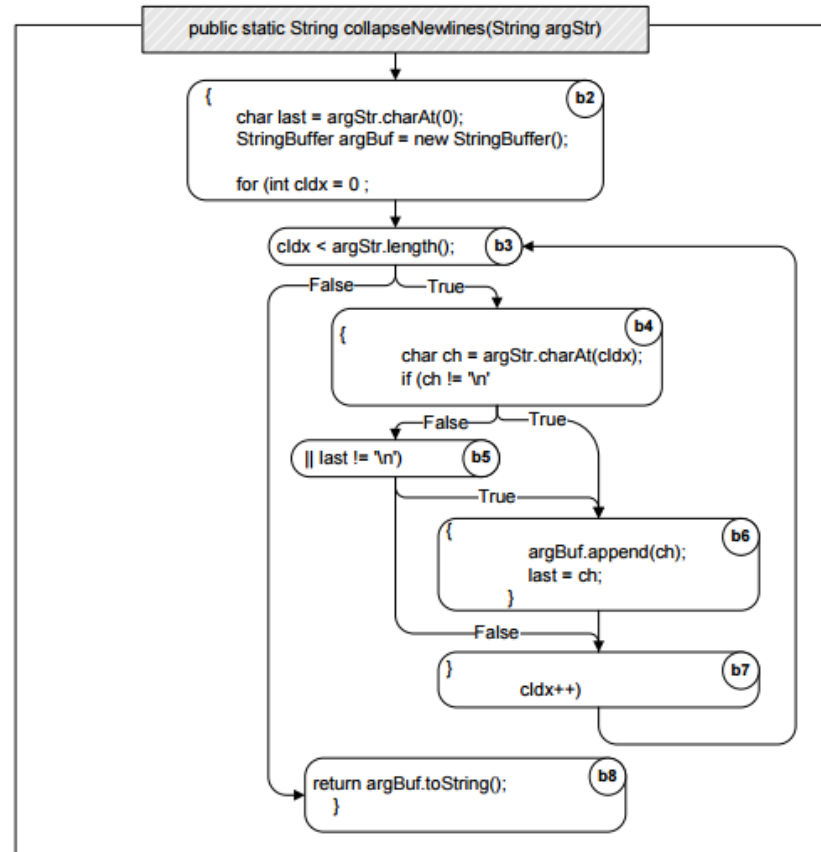
Level 2

- Kinds of complexity, coverage, depend metrics
 - Using CFG (Control Flow Graph), DFD (Data Flow Graph), Etc.

```
public static String collapseNewlines(String argStr)
{
    char last = argStr.charAt(0);
    StringBuffer argBuf = new StringBuffer();

    for (int cldx = 0 ; cldx < argStr.length(); cldx++)
    {
        char ch = argStr.charAt(cldx);
        if (ch != '\n' || last != '\n')
        {
            argBuf.append(ch);
            last = ch;
        }
    }

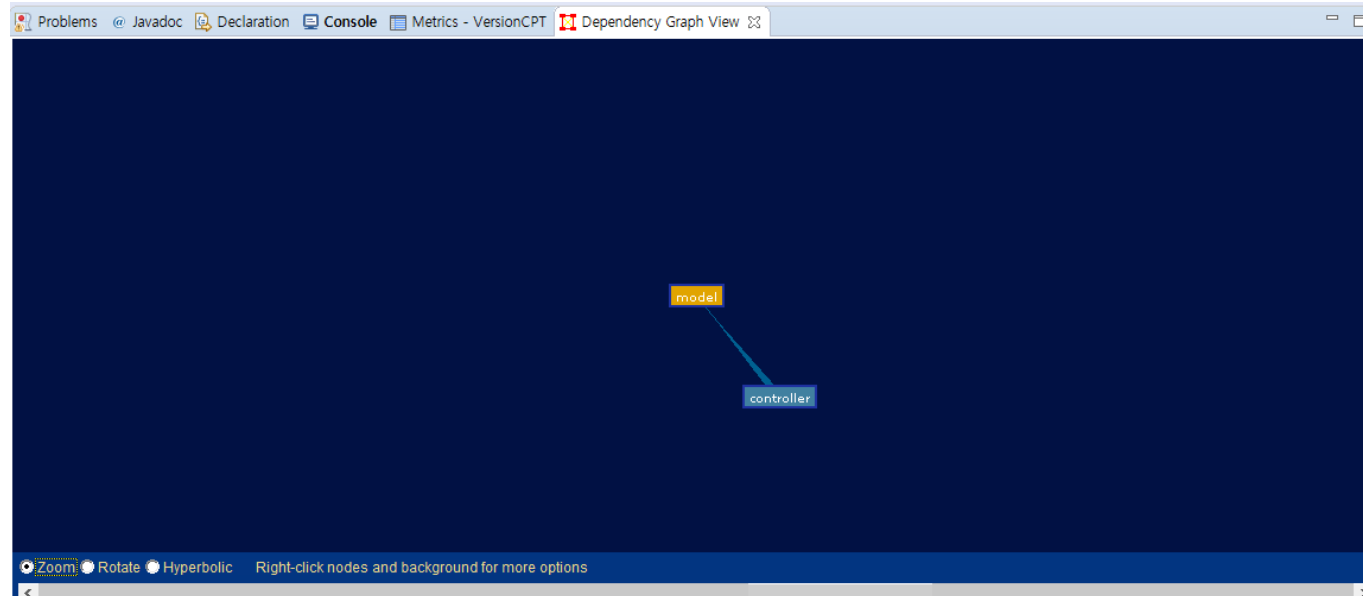
    return argBuf.toString();
}
```



Level 2

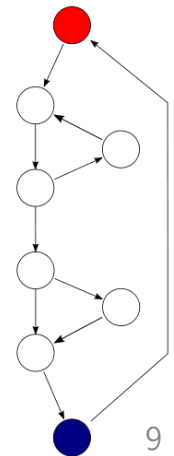
- Source code information and dependency graph
 - *E.g.* Cyclomatic Complexity

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
> McCabe Cyclomatic Complexity (avg/max per method)	1.7	2.407	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG	
> Number of Parameters (avg/max per method)	0.655	0.878	5	/VersionCPT/src/controller/AdminVer.java	chkAuth	
> Nested Block Depth (avg/max per method)	1.355	0.746	5	/VersionCPT/src/controller/AdminVer.java	chkAuth	
> Afferent Coupling (avg/max per packageFragment)	1.5	1.5	3	/VersionCPT/src/model		
> Efferent Coupling (avg/max per packageFragment)	1.5	1.5	3	/VersionCPT/src/controller		
> Instability (avg/max per packageFragment)	0.5	0.5	1	/VersionCPT/src/controller		
> Abstractness (avg/max per packageFragment)	0	0	0	/VersionCPT/src/controller		
> Normalized Distance (avg/max per packageFragment)	0.5	0.5	1	/VersionCPT/src/model		
> Depth of Inheritance Tree (avg/max per type)	1.091	0.287	2	/VersionCPT/src/controller/ServerThread.java		
> Weighted methods per Class (avg/max per type)	187	17	7.828	28	/VersionCPT/src/controller/ServerThread.java	
> Number of Children (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Overridden Methods (avg/max per type)	1	0.091	0.287	1	/VersionCPT/src/controller/ServerThread.java	
> Lack of Cohesion of Methods (avg/max per type)	0.596	0.261	0.894	/VersionCPT/src/model/Version.java		
> Number of Attributes (avg/max per type)	54	4.909	2.678	11	/VersionCPT/src/model/Choice.java	
> Number of Static Attributes (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Methods (avg/max per type)	108	9.818	6.422	24	/VersionCPT/src/model/Version.java	
> Number of Static Methods (avg/max per type)	2	0.182	0.575	2	/VersionCPT/src/controller/MainServer.java	
> Specialization Index (avg/max per type)	0.061	0.192	0.667	/VersionCPT/src/controller/ServerThread.java		
> Number of Classes (avg/max per packageFragment)	11	5.5	0.5	6	/VersionCPT/src/controller	
> Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/VersionCPT/src/controller	
> Number of Packages	2					
> Total Lines of Code	917					
> Method Lines of Code (avg/max per method)	549	4.991	12.482	119	/VersionCPT/src/controller/ServerThread.java	analyzeMSG



Cyclomatic Complexity

- Cyclomatic complexity is a software metric (measurement), used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. It was developed by Thomas J. McCabe, Sr. in 1976.
- Cyclomatic complexity is computed using the control flow graph of the program: the nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command. Cyclomatic complexity may also be applied to individual functions, modules, methods or classes within a program.
- One testing strategy, called basis path testing by McCabe who first proposed it, is to test each linearly independent path through the program; in this case, the number of test cases will equal the cyclomatic complexity of the program.



Cyclomatic Complexity

- **Lower is better.** A McCabe complexity under 5 is good, from 5-10 is OK, and over 10 is too complex. A high flow complexity may be a symptom of a function which does too much or has low cohesion (does to many different things). But don't take these numbers too seriously -- you may have comprehensible control flow despite high numbers. For example, one large *switch* statement can be clear to understand, but can dramatically increase the count.
- **23 is too high**

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
▼ McCabe Cyclomatic Complexity (avg/max per method)		1.7	2.407	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
▼ src		1.7	2.407	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
▼ controller		2.614	3.582	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
▼ ServerThread.java		9.333	9.741	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
▼ ServerThread		9.333	9.741	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
analyzeMSG	23					

Level 3

- Static analysis
 - analyzing critical errors which can be issued during execution
 - Without execution(compile)

The screenshot displays the Eclipse IDE interface with the following components:

- Bug Explorer:** A tree view showing a hierarchy of bugs. The selected bug is "There is an apparent infinite loop in new controller.MainServer() [Scary(8), High confidence]".
- Main Editor:** Shows the source code of `MainServer.java`. A `while (isConnected) {` loop is highlighted in blue, corresponding to the selected bug.
- Bug Info:** A panel on the right providing details for the selected bug:
 - Navigation:** MainServer.java: 46
 - Description:** There is an apparent infinite loop in new controller.MainServer() Local variable named isConnected At MainServer.java[line 29]
 - Bug:** There is an apparent infinite loop in new controller.MainServer()
 - Text:** This loop doesn't seem to have a way to terminate (other than by perhaps throwing an exception).
 - Rank:** Scary (8), confidence: High
 - Pattern:** IL_INFINITE_LOOP
 - Type:** IL, Category: CORRECTNESS (Correctness)
 - XML output:** A snippet of XML representing the bug data.

Automated Static Analysis

- There are several tools for static analysis of source codes
 - Commercial
 - Powerful tool is too expensive
 - Open source
 - Several open source tools exist also

Tools

Level 1 : Rule Checking , Coding Style Checking - 코드의 syntax 를 분석

checkstyle (java)	: 코딩 표준 준수 검사 도구
StyleCop (C#)	: 코딩 표준 준수 검사 도구
N'SIQ CppStyle (c/c++)	: NHN 의 표준 코딩 스타일 체크 도구
cpplint (c++)	: 구글의 코딩 스타일 가이드를 위한 코딩 규칙 검사 도구
PMD (java)	: Dead code, 빈 조건문, 반복되는 코드 검사 등을 지원
sonar (c/c++, java 등)	: 코딩 규칙, 중복 검사 등을 지원 (다양한 기능이 있는 플랫폼)
QA-C (c)	: MISRA C 를 포함한 코딩 표준과 코딩 룰 준수 검사 도구

Tools

Level 2: OOO analysis - 코드를 CFG/DFG 등의 형태로 변환하여 각종 품질 관련 분석을 수행

eclipsemetrics : 소스코드 복잡도 분석 도구 in Eclipse, 그래프로 결과 확인

javancss(java) : 소스코드 복잡도 분석 도구

cobertura(java) : 소스코드 복잡도 분석 및 커버리지 측정

MALPAS(c, ada, PLM) : 분석을 위해 directed graphs 와 regular algebra 를 사용하여 프로그램을 표현. 자동화 도구를 사용해 프로그램의 구조를 묘사하는 것이 가능. Code 가 specification 을 만족하는 것을 보여주기 위한 formal proof 지원. Nuclear, aerospace, defence 분야의 safety critical application 의 correctness 를 보여주기 위해 사용되고 있다.

ECLAIR(c/c++) : formal methods-based static analysis

BLAST(c/c++) : lazy abstraction 기반 C 프로그램 model checker

Sotoarc/Sotograph(c/c++, java) : 구조를 시각화하여 의존성과 참조 문제 등을 확인하는 도구

Tools

Level 3 : static analysis - 코드에 대한 정적분석을 통해 실제 execution 時발행할 수 있는 중요한 오류(Divide by Zero, Null Pointer 등)를 예측

(대부분의 도구가 비슷한 정적 분석을 수행한다.)

Findbugs(java) : 소스코드에 문제가 될 수 있는 부분을 찾는 정적 분석 도구.

(예로 integer 를 나누기 할 경우 double 이나 float 로 cast 되는 부분을 알려준다.)

cppcheck(c/c++) : regular expression 분석을 통해 소스코드의 문제점을 분석

Fluctuat(c, ada) : 부동소수점에 관한 검사에 특화된 분석 도구

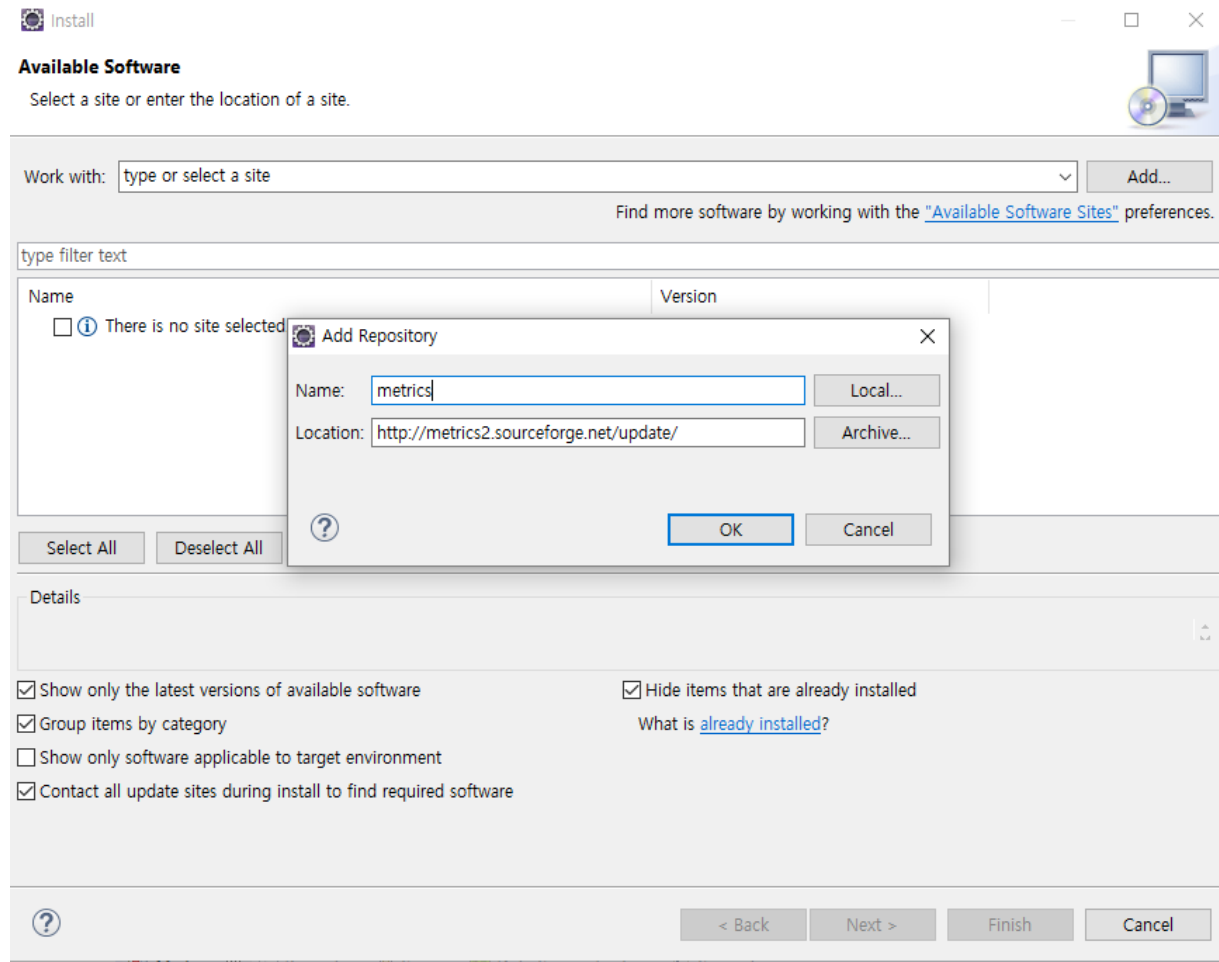
DevPartner(java, .NET) : 통합도구(code coverage, dead lock detection, code quality 등)

Coverity(c/c++,c#,java) : 정적 분석 도구 리더 중 하나로 open source에서 제공하는

기본적인 기능을 대부분 가지고 있으며, coverage 까지 확인 가능한 도구

Eclipse Metrics Plugin

- Level 2
- Install
 - Help -> Install New Software -> Add
 - > input the location (http://metrics2.sourceforge.net/update/)



Eclipse Metrics Plugin

Install

Available Software
Check the items that you wish to install.

Work with:

Find more software by working with the ["Available Software Sites"](#) preferences.

type filter text

Name	Version
> <input checked="" type="checkbox"/> Metrics	

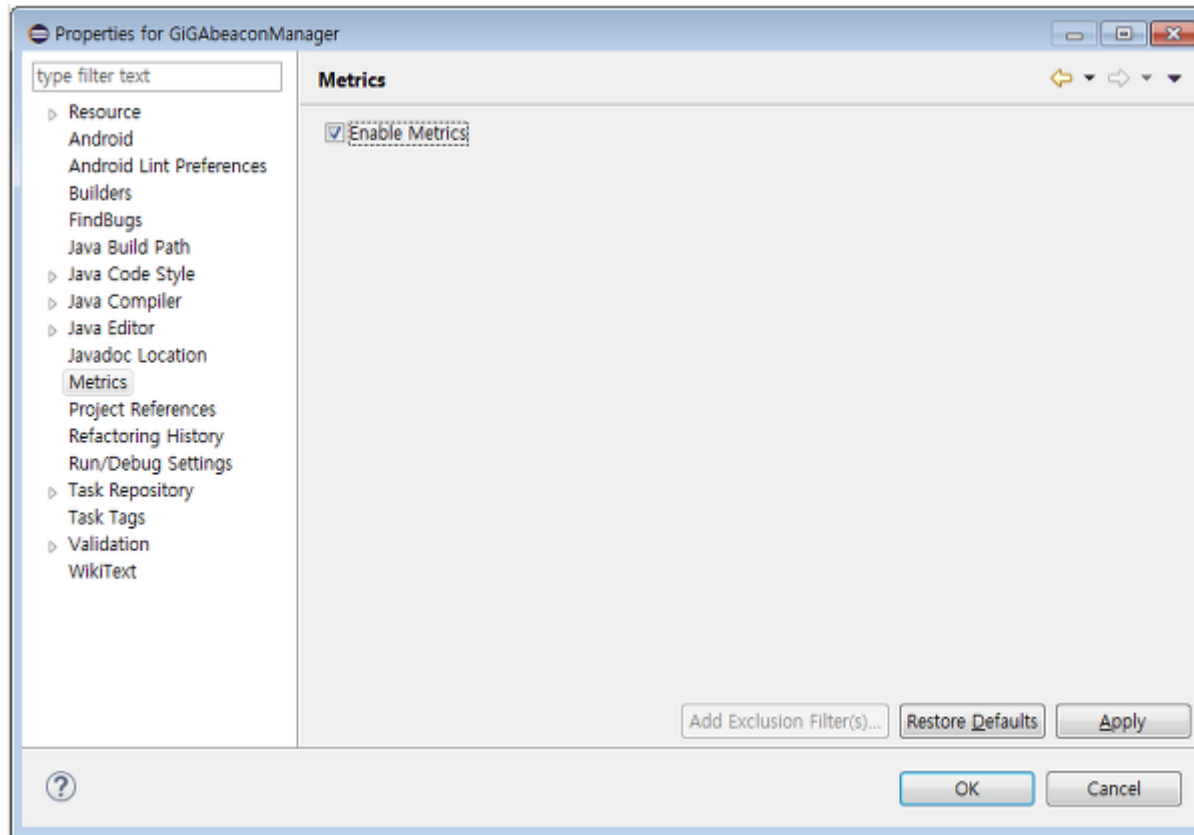
Details
Metrics plugin for eclipse [More...](#)

Show only the latest versions of available software
 Group items by category
 Show only software applicable to target environment
 Contact all update sites during install to find required software

Hide items that are already installed
What is [already installed?](#)

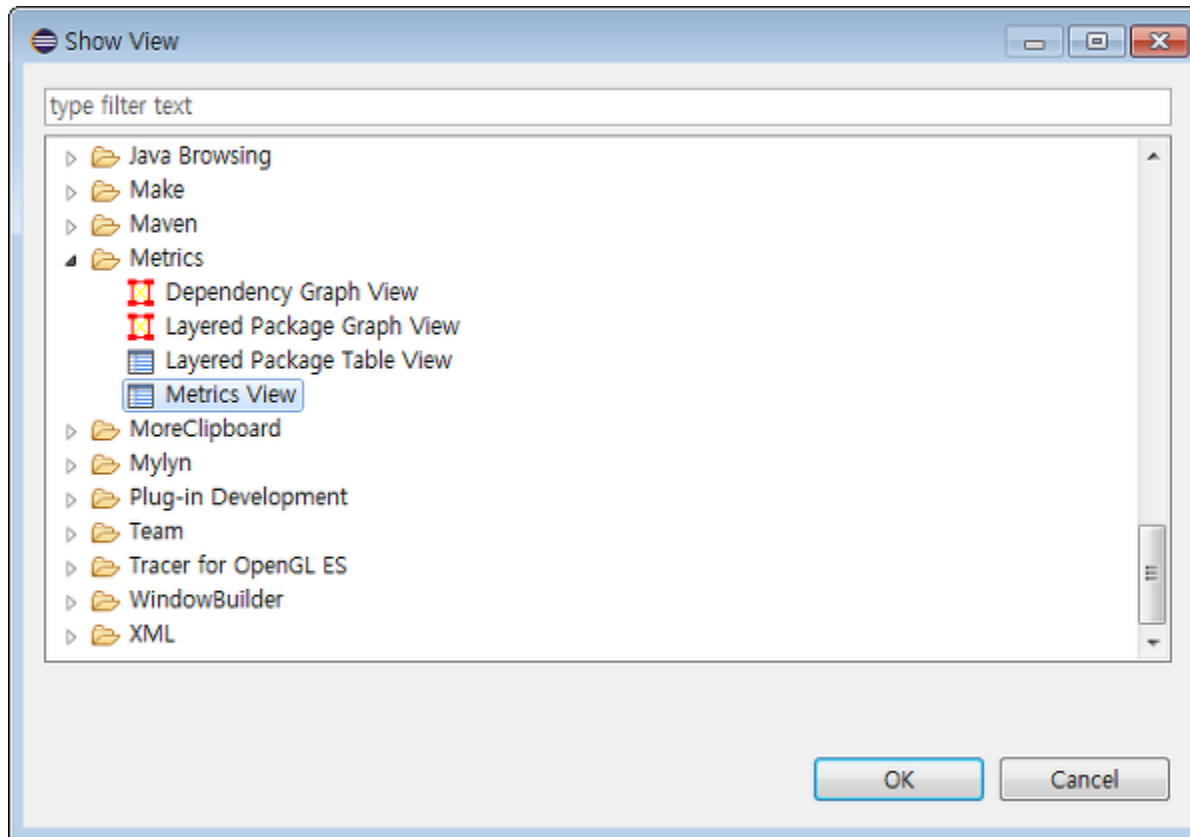
Eclipse Metrics Plugin

- 각 project -> properties -> Metrics -> enable



Eclipse Metrics Plugin

- Window -> Show View -> Metrics -> Metrics View



Eclipse Metrics Plugin

The screenshot shows the Eclipse IDE interface. The top part displays the source code for `MainServer.java` in the `controller` package. The code includes imports for `java.io.FileInputStream` and defines a `MainServer` class with several attributes and a `main` method.

The bottom part of the screenshot shows the 'Metrics' view, which displays a table of metrics for the project. The table is titled 'Metrics - VersionCPT - Weighted methods per Class (avg/max per type)'. The table has columns for Metric, Total, Mean, Std. Dev., Maximum, Resource causing Maximum, and Method.

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
> McCabe Cyclomatic Complexity (avg/max per method)		1.7	2.407	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
> Number of Parameters (avg/max per method)		0.655	0.878	5	/VersionCPT/src/controller/AdminVer.java	chkAuth
> Nested Block Depth (avg/max per method)		1.355	0.746	5	/VersionCPT/src/controller/AdminVer.java	chkAuth
> Afferent Coupling (avg/max per packageFragment)		1.5	1.5	3	/VersionCPT/src/model	
> Efferent Coupling (avg/max per packageFragment)		1.5	1.5	3	/VersionCPT/src/controller	
> Instability (avg/max per packageFragment)		0.5	0.5	1	/VersionCPT/src/controller	
> Abstractness (avg/max per packageFragment)		0	0	0	/VersionCPT/src/controller	
> Normalized Distance (avg/max per packageFragment)		0.5	0.5	1	/VersionCPT/src/model	
> Depth of Inheritance Tree (avg/max per type)		1.091	0.287	2	/VersionCPT/src/controller/ServerThread.java	
> Weighted methods per Class (avg/max per type)	187	17	7.828	28	/VersionCPT/src/controller/ServerThread.java	
> Number of Children (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Overridden Methods (avg/max per type)	1	0.091	0.287	1	/VersionCPT/src/controller/ServerThread.java	
> Lack of Cohesion of Methods (avg/max per type)		0.596	0.261	0.894	/VersionCPT/src/model/Version.java	
> Number of Attributes (avg/max per type)	54	4.909	2.678	11	/VersionCPT/src/model/Choice.java	
> Number of Static Attributes (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Methods (avg/max per type)	108	9.818	6.422	24	/VersionCPT/src/model/Version.java	
> Number of Static Methods (avg/max per type)	2	0.182	0.575	2	/VersionCPT/src/controller/MainServer.java	
> Specialization Index (avg/max per type)		0.061	0.192	0.667	/VersionCPT/src/controller/ServerThread.java	
> Number of Classes (avg/max per packageFragment)	11	5.5	0.5	6	/VersionCPT/src/controller	
> Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/VersionCPT/src/controller	
> Number of Packages	2					
> Total Lines of Code	917					
> Method Lines of Code (avg/max per method)	549	4.991	12.482	119	/VersionCPT/src/controller/ServerThread.java	analyzeMSG

Eclipse Metrics Plugin

- Complexity, code line, 상속 관계 등

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
> McCabe Cyclomatic Complexity (avg/max per method)		1.7	2.407	23	/VersionCPT/src/controller/ServerThread.java	analyzeMSG
> Number of Parameters (avg/max per method)		0.655	0.878	5	/VersionCPT/src/controller/AdminVer.java	chkAuth
> Nested Block Depth (avg/max per method)		1.355	0.746	5	/VersionCPT/src/controller/AdminVer.java	chkAuth
> Afferent Coupling (avg/max per packageFragment)		1.5	1.5	3	/VersionCPT/src/model	
> Efferent Coupling (avg/max per packageFragment)		1.5	1.5	3	/VersionCPT/src/controller	
> Instability (avg/max per packageFragment)		0.5	0.5	1	/VersionCPT/src/controller	
> Abstractness (avg/max per packageFragment)		0	0	0	/VersionCPT/src/controller	
> Normalized Distance (avg/max per packageFragment)		0.5	0.5	1	/VersionCPT/src/model	
> Depth of Inheritance Tree (avg/max per type)		1.091	0.287	2	/VersionCPT/src/controller/ServerThread.java	
> Weighted methods per Class (avg/max per type)	187	17	7.828	28	/VersionCPT/src/controller/ServerThread.java	
> Number of Children (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Overridden Methods (avg/max per type)	1	0.091	0.287	1	/VersionCPT/src/controller/ServerThread.java	
> Lack of Cohesion of Methods (avg/max per type)		0.596	0.261	0.894	/VersionCPT/src/model/Version.java	
> Number of Attributes (avg/max per type)	54	4.909	2.678	11	/VersionCPT/src/model/Choice.java	
> Number of Static Attributes (avg/max per type)	0	0	0	0	/VersionCPT/src/controller/AdminProject.java	
> Number of Methods (avg/max per type)	108	9.818	6.422	24	/VersionCPT/src/model/Version.java	
> Number of Static Methods (avg/max per type)	2	0.182	0.575	2	/VersionCPT/src/controller/MainServer.java	
> Specialization Index (avg/max per type)		0.061	0.192	0.667	/VersionCPT/src/controller/ServerThread.java	
> Number of Classes (avg/max per packageFragment)	11	5.5	0.5	6	/VersionCPT/src/controller	
> Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/VersionCPT/src/controller	
> Number of Packages	2					
> Total Lines of Code	917					
> Method Lines of Code (avg/max per method)	549	4.991	12.482	119	/VersionCPT/src/controller/ServerThread.java	analyzeMSG

Eclipse Metrics Plugin

- Dependency Graph

The screenshot shows the Eclipse IDE interface with the 'Dependency Graph View' open. The toolbar at the top has a red box around the Dependency Graph icon. The main window displays a dependency graph with the following nodes and connections:

- model** (orange node) is connected to **controller** (blue node).
- com.hanamicon.rs.collector** (orange node) is the central node, connected to many other nodes:
- com.hanamicon.beacon.bluetooth.Hanabee** (blue node)
- R.id.tvDisplay** (blue node)
- android.widget** (blue node)
- R.id.btnRun** (blue node)
- R.id.tvElapsed** (blue node)
- R.id.btnEnd** (blue node)
- android.support.v4.util** (blue node)
- android.app** (blue node)
- android.view.View** (blue node)
- R.layout.activity_main** (blue node)
- android.view** (blue node)
- android.os** (blue node)
- com.hanamicon.beacon.bluetooth** (blue node)
- R.string.app_name** (blue node)
- com.hanamicon.beacon.model** (blue node)

At the bottom of the window, there is a toolbar with options: Zoom, Rotate, Hyperbolic, and a note: 'Right-click nodes and background for more options'.

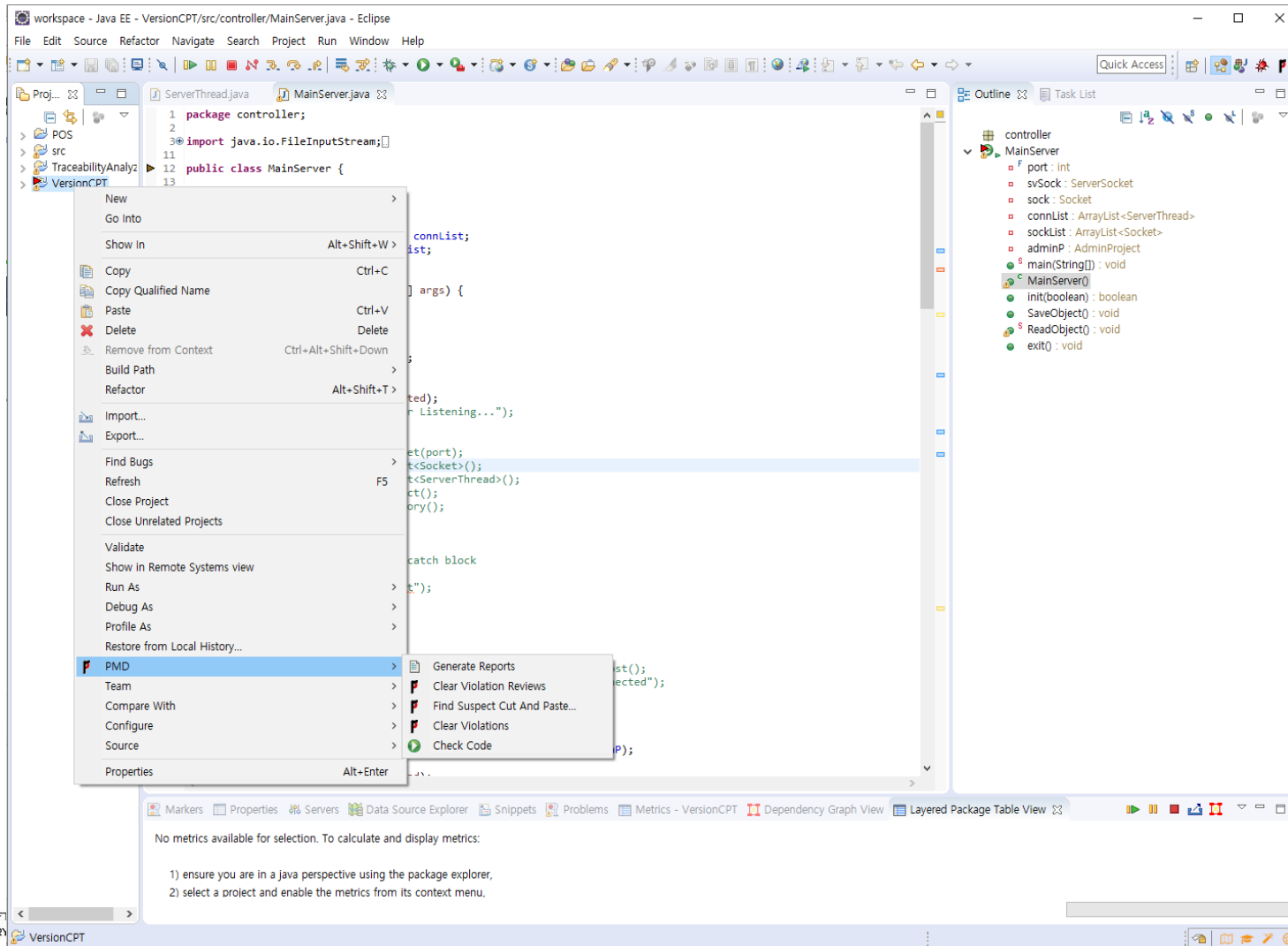
PMD

- Level 1+Level 3 (part of)
- PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, Salesforce.com Apex and Visualforce, PLSQL, Apache Velocity, XML, XSL.
- Additionally it includes CPD, the copy-paste-detector. CPD finds duplicated code in Java, C, C++, C#, Groovy, PHP, Ruby, Fortran, JavaScript, PLSQL, Apache Velocity, Scala, Objective C, Matlab, Python, Go, Swift and Salesforce.com Apex and Visualforce.

PMD

- Install

- Install new software -> <https://dl.bintray.com/pmd/pmd-eclipse-plugin/updates/>



PMD

workspace - PMD - VersionCPT/src/controller/MainServer.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

ServerThread.java MainServer.java`73 public boolean init(boolean isConnected) {
74 // TODO implement here
75 // client, adminU, adminP, adminV 객체 선언
76
77 svSock = null;
78 sock = null;
79
80 System.out.println("Server Listening...");
81
82 try {
83 svSock = new ServerSocket(port);
84 sockList = new ArrayList<Socket>();
85 connList = new ArrayList<ServerThread>();
86 adminP = new AdminProject();
87 isConnected = true;
88 }
89 }`

Violations

Violations Overview

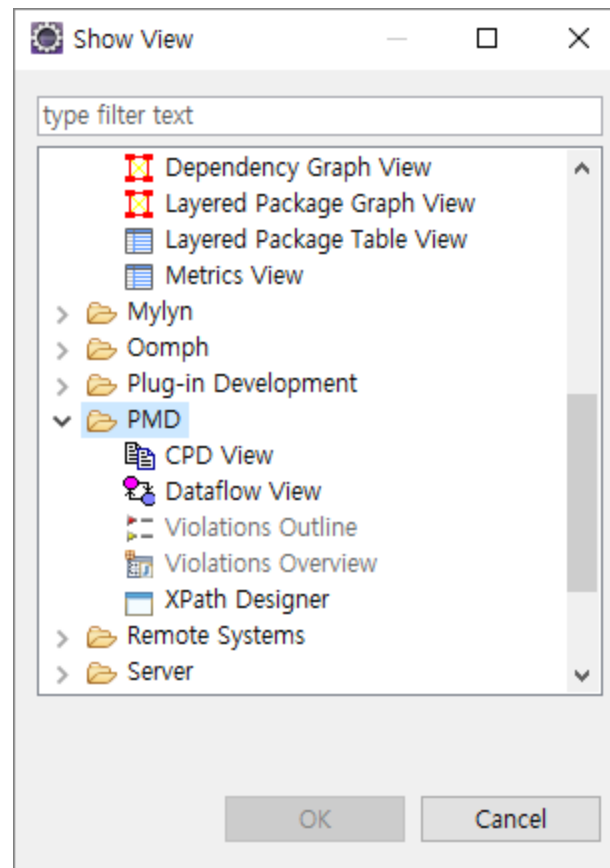
Priority filtering

Priority	Line	created	Rule	Error Message
High	137	Thu ...	M...	Method na...
High	101	Thu ...	M...	Method na...
High	80	Thu ...	Sy...	System.out...
Medium	73	Thu ...	Av...	Avoid reassi...
Medium	113	Thu ...	Sy...	System.out...
Medium	91	Thu ...	Sy...	System.out...
Medium	53	Thu ...	Sy...	System.out...
Medium	63	Thu ...	Sy...	System.out...
Medium	12	Thu ...	C...	headerCom...
Medium	16	Thu ...	Be...	Found non...
Medium	130	Thu ...	E...	Avoid empt...
Medium	90	Thu ...	Av...	Avoid print...
Medium	122	Thu ...	ifS...	Avoid using...
Medium	62	Thu ...	Av...	Avoid print...
Medium	14	Thu ...	Be...	Found non...
Medium	15	Thu ...	Be...	Found non...
Medium	158	Thu ...	E...	Avoid empt...
Medium	55	Thu ...	Av...	Avoid insta...
Medium	25	Thu ...	C...	publicMeth...
Medium	117	Thu ...	Av...	Avoid print...
Medium	14	Thu ...	Fi...	This final fie...
Medium	17	Thu ...	Be...	Found non...
Medium	127	Thu ...	ifS...	Avoid using...
Medium	150	Thu ...	Av...	Avoid catch...
Medium	18	Thu ...	C...	fieldComme...
Medium	15	Thu ...	C...	fieldComme...
Medium	16	Thu ...	C...	fieldComme...
Medium	160	Thu ...	ifS...	Avoid using...
Medium	73	Thu ...	C...	publicMeth...
Medium	17	Thu ...	Lo...	Avoid using...
Medium	168	Thu ...	C...	publicMeth...

Element	Violations	Violations/req.	Violations/line	Project
controller	415	N/A	N/A	VersionCPT
> AdminProject.java	38	N/A	N/A	VersionCPT
> Server.java	55	N/A	N/A	VersionCPT
> MainServer.java	56	N/A	N/A	VersionCPT
> AdminUser.java	59	N/A	N/A	VersionCPT
> ServerThread.java	57	N/A	N/A	VersionCPT
> AdminVer.java	150	N/A	N/A	VersionCPT
model	187	N/A	N/A	VersionCPT
> Category.java	36	N/A	N/A	VersionCPT
> Constraints.java	8	N/A	N/A	VersionCPT
> Version.java	55	N/A	N/A	VersionCPT
> Opinion.java	24	N/A	N/A	VersionCPT
> Choice.java	64	N/A	N/A	VersionCPT

PMD

- 이외에도 dataflow, CPD (Finding duplicated code) 가 가능



PMD

- Method 별 data flow

The screenshot displays the Eclipse IDE interface with PMD analysis results. The Package Explorer on the left shows the project structure. The central code editor shows the `ReadObject()` method in `MainServer.java`. The Dataflow View at the bottom right provides a detailed analysis of the method's data flow, including a control flow graph and a table of dataflow events.

Line	Method	Dataflow types	Codeline
137	SaveObject exit() init(boolean) : boolean main(String)	fis, u(ois)	public static void ReadObject() throws IOException {
137			public static void ReadObject() throws IOException {
138		d(fis)	FileInputStream fis = null;
139		d(ois)	ObjectInputStream ois = null;
143		d(fis)	fis = new FileInputStream("object.dat");
144		r(fis), d(ois)	ois = new ObjectInputStream(fis);
146		r(ois)	ois.readObject();
151			e.printStackTrace();
155		r(fis)	if (fis != null)
157		r(fis)	fis.close();
160		r(ois)	if (ois != null)
162		r(ois)	ois.close();
166		u(fis), u(ois)	}

PMD

- Report example
 - User can select the form of the report

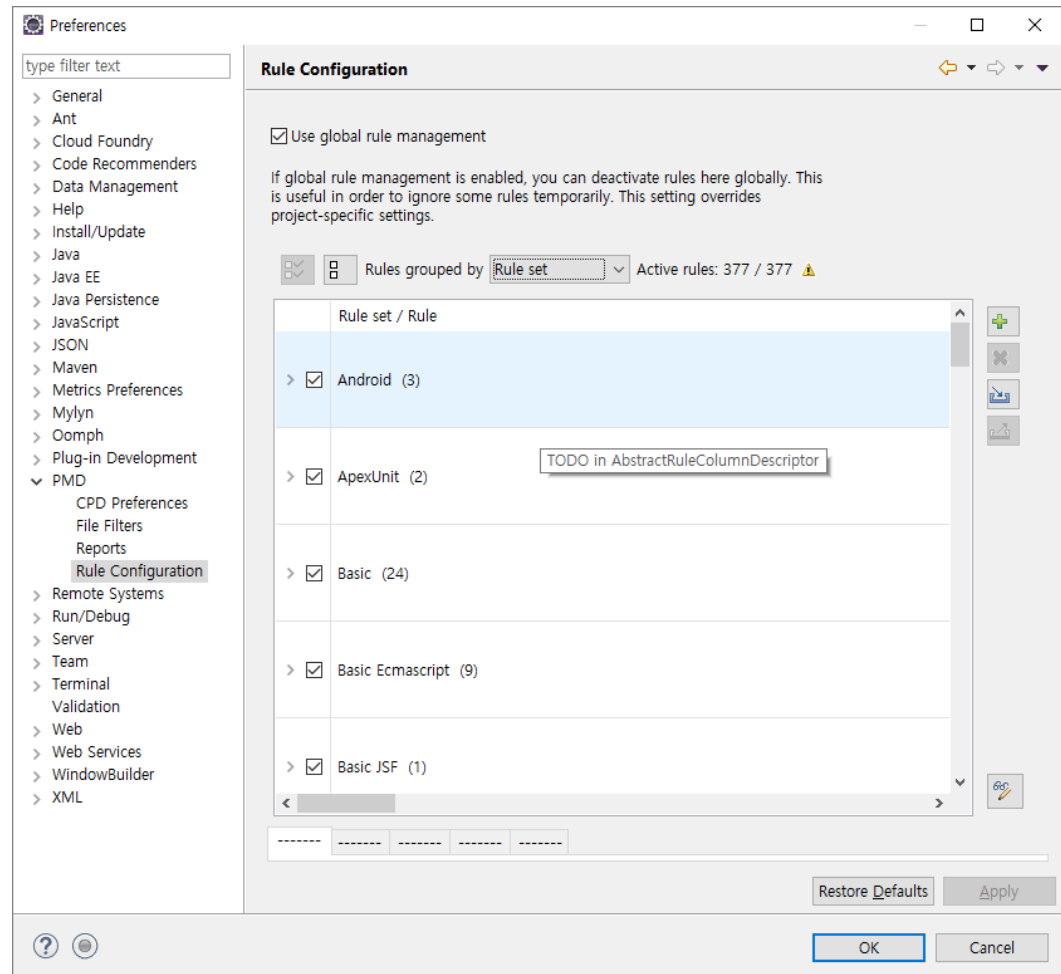
The screenshot shows the PMD Preferences dialog box with the 'Reports' tab selected. The 'Name' column lists various report formats, with 'text' checked. The 'Properties' column is empty. Below the list, there are input fields for 'Name:' and 'Description:', and a 'Properties' section.

Overlaid on the dialog is a preview of the generated XML report. The report is an XML document with the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmd timestamp="2017-05-25T19:15:36.482" version="5.6.1">
  <file name="src/controller/AdminProject.java">
    <violation priority="4" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/imports.html#UnusedImports"
      class="AdminProject" package="controller" ruleset="Import Statements" rule="UnusedImports" endcolumn="0" begincolumn="0"
      endline="3" beginline="3"> Avoid unused imports such as 'java.io.File' </violation>
    <violation priority="4" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/imports.html#UnusedImports"
      class="AdminProject" package="controller" ruleset="Import Statements" rule="UnusedImports" endcolumn="0" begincolumn="0"
      endline="4" beginline="4"> Avoid unused imports such as 'java.text.DateFormat' </violation>
    <violation priority="4" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/imports.html#UnusedImports"
      class="AdminProject" package="controller" ruleset="Import Statements" rule="UnusedImports" endcolumn="0" begincolumn="0"
      endline="5" beginline="5"> Avoid unused imports such as 'java.text.SimpleDateFormat' </violation>
    <violation priority="4" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/imports.html#UnusedImports"
      class="AdminProject" package="controller" ruleset="Import Statements" rule="UnusedImports" endcolumn="0" begincolumn="0"
      endline="6" beginline="6"> Avoid unused imports such as 'java.util.Date' </violation>
    <violation priority="4" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/imports.html#UnusedImports"
      class="AdminProject" package="controller" ruleset="Import Statements" rule="UnusedImports" endcolumn="0" begincolumn="0"
      endline="11" beginline="11"> Avoid unused imports such as 'model.Opinion' </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentRequired"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentRequired" endcolumn="0" begincolumn="0"
      endline="122" beginline="14"> headerCommentRequirement Required </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/unusedcode.html#UnusedPrivateField"
      class="AdminProject" package="controller" ruleset="Unused Code" rule="UnusedPrivateField" endcolumn="0" begincolumn="0"
      endline="17" beginline="17"> Avoid unused private fields such as 'projectName'. </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentRequired"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentRequired" endcolumn="0" begincolumn="0"
      endline="17" beginline="17"> fieldCommentRequirement Required </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/coupling.html#LooseCoupling"
      class="AdminProject" package="controller" ruleset="Coupling" rule="LooseCoupling" endcolumn="0" begincolumn="0"
      endline="18" beginline="18"> Avoid using implementation types like 'HashMap'; use the interface instead </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentRequired"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentRequired" endcolumn="0" begincolumn="0"
      endline="18" beginline="18"> publicMethodCommentRequirement Required </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentRequired"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentRequired" endcolumn="0" begincolumn="0"
      endline="82" beginline="21"> Comment is too large: Too many lines </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentSize"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentSize" endcolumn="0" begincolumn="0"
      endline="81" beginline="26"> Comment is too large: Too many lines </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/coupling.html#LooseCoupling"
      class="AdminProject" package="controller" ruleset="Coupling" rule="LooseCoupling" endcolumn="0" begincolumn="0"
      endline="84" beginline="3" > Avoid using implementation types like 'HashMap'; use the interface instead </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/optimizations.html#MethodArgumentCouldBeFinal"
      class="AdminProject" package="controller" ruleset="Optimization" rule="MethodArgumentCouldBeFinal" endcolumn="0"
      begincolumn="0" endline="84" beginline="84"> Parameter 'projectName' is not assigned and could be declared final </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/design.html#AvoidSynchronizedAtMethodLevel"
      class="AdminProject" package="controller" ruleset="Design" rule="AvoidSynchronizedAtMethodLevel" endcolumn="0"
      begincolumn="0" endline="88" beginline="88"> Use block level rather than method level synchronization </violation>
    <violation priority="3" externalInfoUrl="https://pmd.github.io/pmd-5.6.1/pmd-java/rules/java/comments.html#CommentRequired"
      class="AdminProject" package="controller" ruleset="Comments" rule="CommentRequired" endcolumn="0" begincolumn="0"
      endline="88" beginline="84"> publicMethodCommentRequirement Required </violation>
```

PMD

- Window -> Preference -> PMD -> Rule Configuration
 - 사용할 rule set 설정
 - 새로운 rule 추가 가능



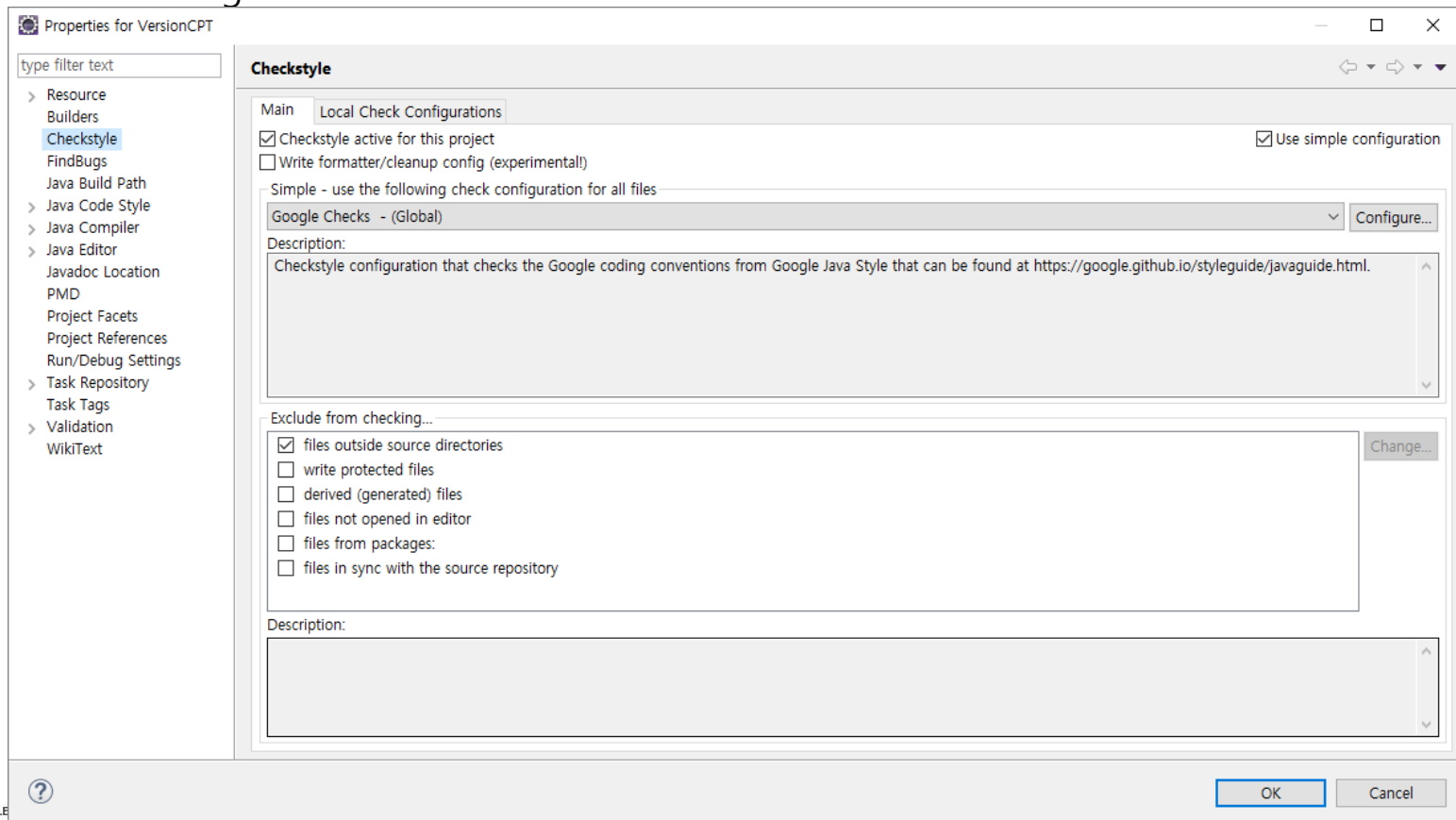
Checkstyle



- Level 1
- Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard.
- Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration files are supplied supporting the [Sun Code Conventions](#), [Google Java Style](#).

Checkstyle

- Install
 - Install New Software -> <http://eclipse-cs.sf.net/update/>
- Properties
 - Activation
 - Rule configuration



Checkstyle

- Naming Convention example

The screenshot shows the Eclipse IDE interface with several Checkstyle configuration windows open. The main window is the 'Built-In Configuration "Google Checks"' dialog, which displays a tree view of modules under the 'Naming Conventions' group. The 'Catch Parameter Name' module is selected. A secondary dialog, 'Configuration of checkstyle module "Catch Parameter Name"', is open, showing the 'Advanced' tab. This dialog includes a 'Severity' dropdown set to 'inherit', a 'format' property with the value '^([a-z]([a-z0-9]|[a-zA-Z0-9])*)?\$', and a text input field labeled 'Input a test string here'. The dialog also has checkboxes for 'Translate tokens' and 'Sort tokens', and 'OK' and 'Cancel' buttons. In the background, the Eclipse IDE shows a project structure with a 'controller' package containing 'MainServer.java'. The IDE's status bar at the bottom indicates the current file is 'VersionCPT'.

Checkstyle

- Other rule example

항목	원인	회피방법
JavadocPackage	모든 method, class에는 help가 존재해야 한다.	시간상 힘들고, 관리되지 않는 주석은 더욱 큰 혼란을 가지고 온다. method의 이름 규칙으로 대신하기로 한다.
NewlineAtEndOfFile	java code의 가장 마지막 줄은 빈공백열로 마쳐져야 한다.	마지막 line에는 항상 빈공백을 넣는다.
Translation	Properties file을 이용한 경우, 국가별 번역이 모두 존재해야 한다.	국가별 번역 파일을 따로 만들거나 default 문자열만을 이용한다.
FileLength	java file의 length는 2000 line을 넘지 않도록 작성한다.	2000 line이 넘어가는 경우, 설계상의 문제가 있기 때문에 class를 재정의한다.
FileTabCharacter	java file 내부에 tab 문자열이 있으면 안된다.	tab을 모두 space로 치환해서 사용하도록 한다.
RegexpSingleline	1 line에는 한개의 method만이 존재해야 한다.	1 line에 대한 설정을 명확하게 해서 사용하도록 한다.

- <http://netframework.tistory.com/363>

Checkstyle

- Activate example

The screenshot shows the Eclipse IDE with a Java file open. The code is as follows:

```
16 private Socket sock;
17 private ArrayList<ServerThread> connList;
18 private ArrayList<Socket> sockList;
19 private AdminProject adminP;
20
21 public static void main(String[] args) {
22     new MainServer();
23 }
24
25 public MainServer() {
26     boolean isConnected = false;
27
28
29     isConnected = init(isConnected);
30     /*System.out.println("Server Listening...");
31
32     try {
33         svSock = new ServerSocket(port);
34         sockList = new ArrayList<Socket>();
35         connList = new ArrayList<ServerThread>();
36         adminP = new AdminProject();
37         //shMem = new SharedMemory();
38
39         isConnected = true;
40     } catch (IOException e) {
41         // TODO Auto-generated catch block
42         e.printStackTrace();
43         System.out.println("init");
44     }
45 */
46     while (isConnected) {
47         try {
```

A blue box highlights the code from line 21 to 47. A text box in the center says "Violation 부분을 표시" (Mark violation parts).

The bottom panel shows a list of violations:

description	Resource	Path	Location	Type
⚠ 'catch rcurlly' have incorrect indentation level 16, expected level should be 4.	MainServer.ja...	/VersionCPT/src/c...	line 119	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 16, expected level should be 4.	MainServer.ja...	/VersionCPT/src/c...	line 152	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 16, expected level should be 4.	Server.java	/VersionCPT/src/c...	line 47	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 16, expected level should be 4.	Server.java	/VersionCPT/src/c...	line 73	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 16, expected level should be 4.	Server.java	/VersionCPT/src/c...	line 86	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 4.	Server.java	/VersionCPT/src/c...	line 172	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 6.	MainServer.ja...	/VersionCPT/src/c...	line 64	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 6.	Server.java	/VersionCPT/src/c...	line 44	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 6.	ServerThread...	/VersionCPT/src/c...	line 216	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 8.	ServerThread...	/VersionCPT/src/c...	line 96	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 24, expected level should be 8.	ServerThread...	/VersionCPT/src/c...	line 130	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 6.	Server.java	/VersionCPT/src/c...	line 167	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 6.	Server.java	/VersionCPT/src/c...	line 195	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 8.	MainServer.ja...	/VersionCPT/src/c...	line 126	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 8.	MainServer.ja...	/VersionCPT/src/c...	line 131	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 8.	MainServer.ja...	/VersionCPT/src/c...	line 159	Checkstyle Pr...
⚠ 'catch rcurlly' have incorrect indentation level 32, expected level should be 8.	MainServer.ja...	/VersionCPT/src/c...	line 164	Checkstyle Pr...
⚠ 'catch' child have incorrect indentation level 17, expected level should be 6.	Server.java	/VersionCPT/src/c...	line 98	Checkstyle Pr...
⚠ 'catch' child have incorrect indentation level 17, expected level should be 6.	Server.java	/VersionCPT/src/c...	line 101	Checkstyle Pr...
⚠ 'catch' child have incorrect indentation level 24, expected level should be 6.	MainServer.ja...	/VersionCPT/src/c...	line 90	Checkstyle Pr...
⚠ 'catch' child have incorrect indentation level 24, expected level should be 6.	MainServer.ja...	/VersionCPT/src/c...	line 91	Checkstyle Pr...
⚠ 'catch' child have incorrect indentation level 24, expected level should be 6.	MainServer.ja...	/VersionCPT/src/c...	line 117	Checkstyle Pr...

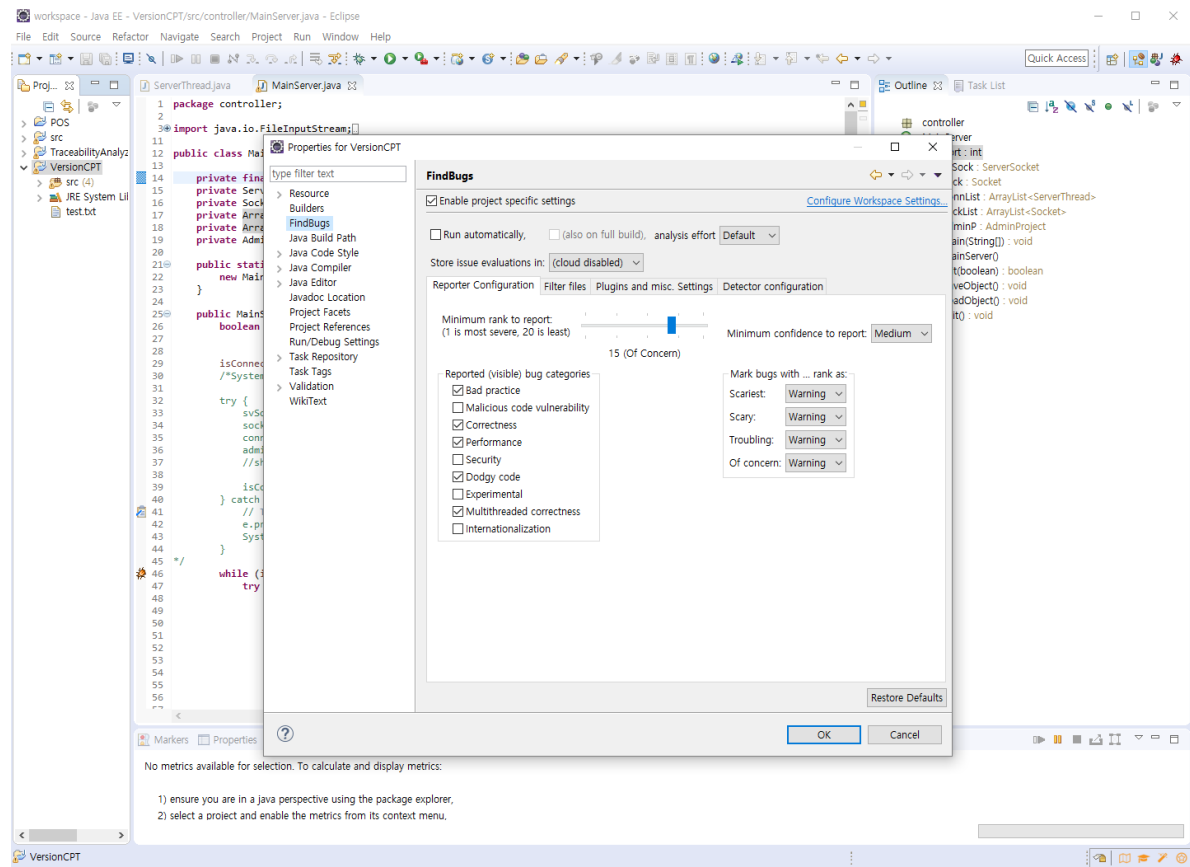
FindBugs

- Level 3
- It is an open source program which looks for bugs in Java code
 - Operates on Java bytecode, rather than source code
 - Source code also available



FindBugs

- Install
 - Install new software -> <http://findbugs.cs.umd.edu/eclipse>
- project -> properties -> Findbugs -> enable



FindBugs

- Detector configuration
 - Searching **rule setting**
 - Enable/disable

Properties for VersionCPT

type filter text

- > Resource Builders
- FindBugs
- > Java Build Path
- > Java Code Style
- > Java Compiler
- > Java Editor
 - Javadoc Location
 - Project Facets
 - Project References
 - Run/Debug Settings
- > Task Repository
 - Task Tags
- > Validation
 - WikiText

FindBugs

Enable project specific settings [Configure Workspace Settings...](#)

Run automatically, (also on full build), analysis effort: Default

Store issue evaluations in: (cloud disabled)

Reporter Configuration | Filter files | Plugins and misc. Settings | **Detector configuration**

Disabled detectors will not participate in FindBugs analysis.
'Grayed out' detectors will run, however they will not report any results to the UI.

Show hidden detectors

Detector id	Pattern(s)	Speed	Provider	Category
<input checked="" type="checkbox"/> AppendingToAnObjectOutputStream	IO	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> AtomicityProblem	AT	fast	FindBugs	Multithreaded cor...
<input type="checkbox"/> BadAppletConstructor	BAC	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> BadResultSetAccess	SQL	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> BadSyntaxForRegularExpression	RE	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> BadUseOfReturnValue	RV	fast	FindBugs	Dodgy code
<input checked="" type="checkbox"/> BadlyOverriddenAdapter	BOA	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> BooleanReturnNull	NP	fast	FindBugs	Bad practice
<input type="checkbox"/> CallToUnsupportedMethod	Dm	fast	FindBugs	Dodgy code
<input type="checkbox"/> CheckExpectedWarnings	FB	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> CheckImmutableAnnotation	JCIP	fast	FindBugs	Bad practice
<input checked="" type="checkbox"/> CheckRelaxingNullnessAnnotation	NP	fast	FindBugs	Dodgy code
<input checked="" type="checkbox"/> CheckTypeQualifiers	TO	slow	FindBugs	Correctness/Dodgy

Detector details

edu.umd.cs.findbugs.detect.FindSpinLoop
This detector looks for loops that spin reading from a field.

Reported patterns:

Restore Defaults

OK Cancel

FindBugs

- Example - InfiniteLoop

Properties for VersionCPT

type filter text

- > Resource
- Builders
- FindBugs**
- Java Build Path
- > Java Code Style
- > Java Compiler
- > Java Editor
- Javadoc Location
- Project Facets
- Project References
- Run/Debug Settings
- > Task Repository
- Task Tags
- > Validation
- WikiText

FindBugs

Enable project specific settings [Configure Workspace Settings...](#)

Run automatically, (also on full build), analysis effort **Default** ▾

Store issue evaluations in: **(cloud disabled)** ▾

Reporter Configuration | Filter files | Plugins and misc. Settings | **Detector configuration**

Disabled detectors will not participate in FindBugs analysis.
'Grayed out' detectors will run, however they will not report any results to the UI.

Show hidden detectors

Detector id	Pattern(s)	Speed	Provider	Category
<input checked="" type="checkbox"/> InconsistentAnnotations	NP	fast	FindBugs	Dodgy code
<input type="checkbox"/> InefficientIndexOf	IIO	fast	FindBugs	Performance
<input type="checkbox"/> InefficientInitializationInsideLoop	IIL	fast	FindBugs	Performance
<input type="checkbox"/> InefficientMemberAccess	IMA	fast	FindBugs	Performance
<input type="checkbox"/> InefficientToArray	ITA	fast	FindBugs	Performance
<input checked="" type="checkbox"/> InfiniteLoop	IL	fast	FindBugs	Correctness
<input checked="" type="checkbox"/> InfiniteRecursiveLoop	IL	fast	FindBugs	Correctness

Detector details

edu.umd.cs.findbugs.detect.InfiniteLoop
Looks for an infinite loop.

Reported patterns:
IL_INFINITE_LOOP (IL, CORRECTNESS): An apparent infinite loop

Plugin: edu.umd.cs.findbugs.plugins.core
Version: 3.0.1-dev-20170525-UNKNOWN
Provider: FindBugs project (<http://findbugs.sourceforge.net>)

Restore Defaults

OK Cancel

FindBugs

- Report setting
 - Report로 생성할 항목들 설정 등

Properties for VersionCPT

type filter text

- > Resource
- Builders
- FindBugs
- Java Build Path
- > Java Code Style
- > Java Compiler
- > Java Editor
- Javadoc Location
- Project Facets
- Project References
- Run/Debug Settings
- > Task Repository
- Task Tags
- > Validation
- WikiText

FindBugs

Enable project specific settings [Configure Workspace Settings...](#)

Run automatically, (also on full build), analysis effort **Default** ▾

Store issue evaluations in: **(cloud disabled)** ▾

Reporter Configuration | Filter files | Plugins and misc. Settings | Detector configuration

Minimum rank to report:
(1 is most severe, 20 is least) Minimum confidence to report: **Medium** ▾

15 (Of Concern)

Reported (visible) bug categories

- Bad practice
- Malicious code vulnerability
- Correctness
- Performance
- Security
- Dodgy code
- Experimental
- Multithreaded correctness
- Internationalization

Mark bugs with ... rank as:

Scariest: **Warning** ▾

Scary: **Warning** ▾

Troubling: **Warning** ▾

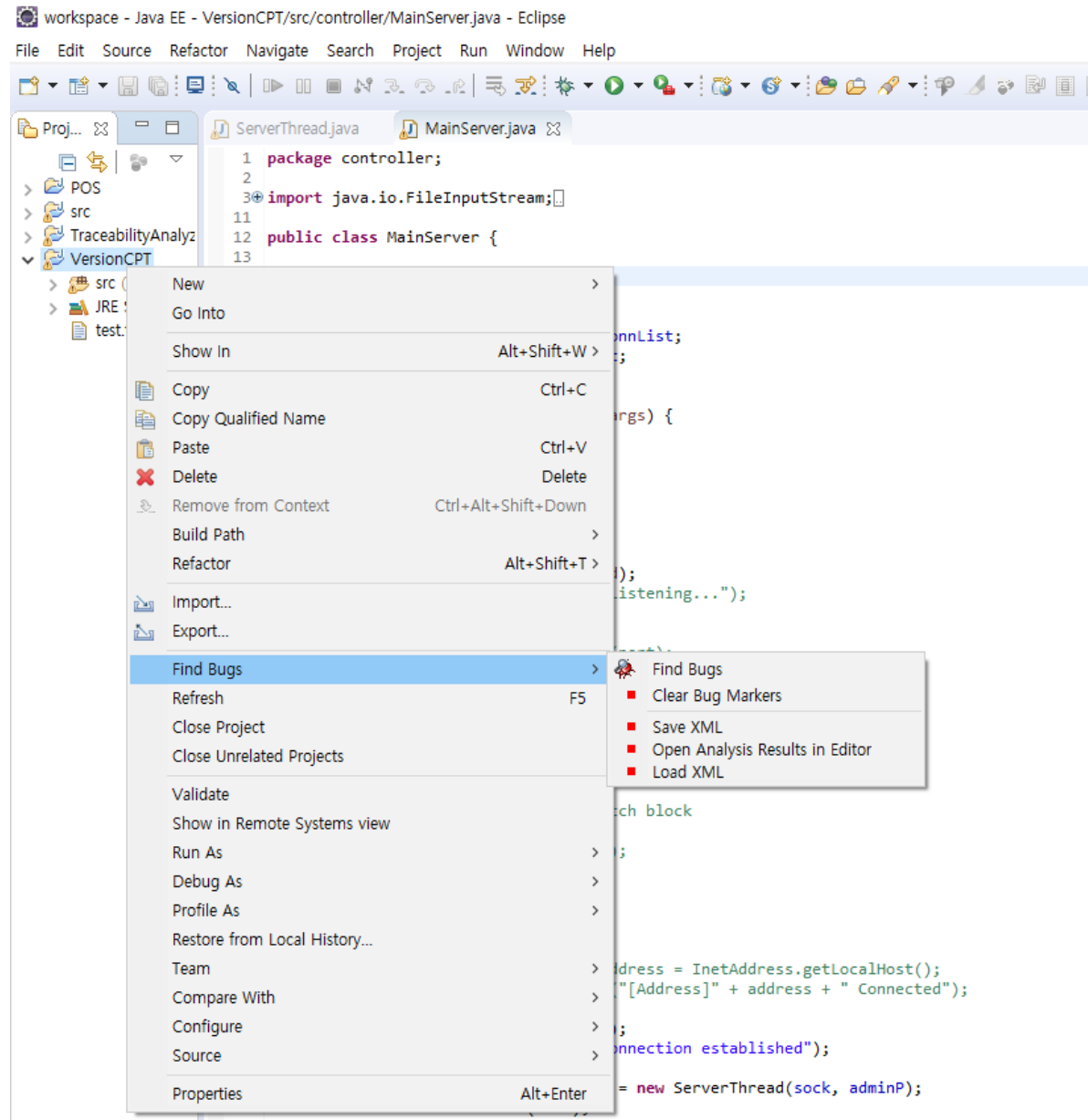
Of concern: **Warning** ▾

[Restore Defaults](#)

OK **Cancel**

FindBugs

- Execution
 - Find Bugs click
- XML generation is possible



FindBugs

The screenshot shows the Eclipse IDE interface with the following components:

- Workspace:** workspace - FindBugs - VersionCPT/src/controller/MainServer.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse development tools.
- Bug Explorer:** A tree view on the left showing the project structure. A red box highlights the following items:
 - VersionCPT (4)
 - Scary (1)
 - High confidence (1)
 - An apparent infinite loop (1)
 - There is an apparent infinite**
 - Troubling (3)
 - Normal confidence (3)

- Code Editor:** Displays the source code for `MainServer.java`. The bug is located at line 29, where a local variable `isConnected` is assigned a value that depends on a condition that is never updated, leading to an infinite loop.
- Bug Reviews:** A panel below the code editor showing a list of bugs. It currently displays "(no cloud selected) No signin required".
- Bug Info:** A panel on the right providing details for the selected bug:
- Navigation:** There is an apparent infinite loop in new controller.MainServer()
Local variable named isConnected
At MainServer.java:[line 29]
- Bug:** There is an apparent infinite loop in new controller.MainServer()
This loop doesn't seem to have a way to terminate (other than by perhaps throwing an exception).
- Rank:** Scary (8), **confidence:** High
- Pattern:** IL_INFINITE_LOOP
- Type:** IL, **Category:** CORRECTNESS (Correctness)
- XML output:**

```
<BugInstance type="IL_INFINITE_LOOP" priority="1" rank="8" abbrev="IL" cate
<Class classname="controller.MainServer">
  <SourceLine classname="controller.MainServer" sourcefile="MainServer.ja
  </Class>
  <Method classname="controller.MainServer" name="<init>" signature="()V" i
  <SourceLine classname="controller.MainServer" start="25" end="71" start
  </Method>
  <SourceLine classname="controller.MainServer" start="46" end="46" startBy
  <LocalVariable name="isConnected" register="1" pc="98" role="LOCAL_VARIAB
  <SourceLine classname="controller.MainServer" start="29" end="29" startBy
</BugInstance>
```
- Footer:** A status bar at the bottom shows the bug summary: "There is an apparent infinite loop in new controller.MainServer() [Scary(8), High confidence]"


JDepend

- Level 2

- Help -> Eclipse Marketplace -> jdepend 입력후 검색

The screenshot shows the Eclipse IDE interface. The main editor displays the source code for `MainServer.java`. The Eclipse Marketplace window is open, showing search results for `jdepend`. The top result is `JDepend4Eclipse 1.2.4` by `Andrey Loskutov, EPL`. The description states: "JDepend4Eclipse plugin for Eclipse is a wrapper for running JDepend from within Eclipse. - What is JDepend? (from www.clarkware.com): JDepend traverses a set of... more info". The window shows 60 stars and 17.1K installs (278 last month). Below the search results, there is a section for "Marketplaces" with icons for various sources. At the bottom of the IDE, the "Markers" view shows a list of errors and warnings, including "catch rcurlly have incorrect indentation level 16, expected" and "catch' child have incorrect indentation level 17, expected".

JDepend

- Source code가 있는 폴더 선택 후 JDepend run

The screenshot shows the Eclipse IDE interface with the JDepend plugin. The 'Packages' view on the left shows a project structure with two main packages: 'controller' and 'model'. The 'Dependencies' view on the right displays several tables:

- Selected object(s):** Shows dependencies for the 'controller' and 'model' packages.
- Packages with cycle:** Shows packages involved in dependency cycles.
- Depends upon - efferent dependencies:** Shows packages that the selected package depends on.
- Used by - afferent dependencies:** Shows packages that use the selected package.

The 'Selected object(s)' table is as follows:

Package	CC(concr....	AC(abstr....	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
controller	6	0	0	1	0.00	1.00	0.00	
model	5	0	1	0	0.00	0.00	1.00	

The 'Depends upon - efferent dependencies' table is as follows:

Package	CC(concr....	AC(abstr....	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
model	5	0	1	0	0.00	0.00	1.00	

The 'Used by - afferent dependencies' table is as follows:

Package	CC(concr....	AC(abstr....	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
controller	6	0	0	1	0.00	1.00	0.00	

The 'Metrics' view at the bottom left shows a graph with a red arrow pointing to a dashed box labeled 'Instability ->' and a red triangle labeled 'Abs'.

JDepend

- CC :: Concrete Class 인터페이스나 추상 클래스가 아닌 Concrete Class 의 수를 나타냄
- AC :: Abstract Class 추상 클래스나 인터페이스의 수를 나타내며 확장성의 척도가 됨
- Ca :: Afferent Couplings 현재 패키지의 클래스에 의존하고 있는 패키지의 수를 나타내며 책임의 척도가 됨
- Ce :: Efferent Couplings 현재 패키지의 클래스들이 의존하고 있는 패키지의 수를 나타내며 독립성의 척도가 됨
- A :: Abstractness ($A = AC/CC+AC$) 추상화 정도를 나타내며, 0 은 구체적인 패키지를, 1 은 추상적인 패키지를 나타냄
- I :: Instability ($I = Ce/(Ce+Ca)$) 변화에 대한 안정성을 나타내며 0 부터 1 사이의 값을 가짐, 0 은 외부 변화에도 끄떡 없는 패키지이며 1 은 작은 변화에도 쉽게 흔들릴 수 있는 패키지를 나타냄
- D :: Distance to Main Sequence Main Sequence 로부터의 거리를 나타내며, 0 은 Main Sequence 와 완전 가깝고 1 은 완전 먼 상태임, Main Sequence란 이상적인 패키지로 완전 추상적이면서 안정적이거나 완전 구체적이면서 불안정한 패키지를 나타냄
- Cycle :: Package dependency cycles 패키지들 상호 간에 의존성을 가지고 있을 때 발생함, 안 좋은 상황이기 때문에 경고 아이콘으로 보여짐

SonarQube

- 이것도 한번 사용해 보세요

발표 – Static Analysis

- 각자 서로 team의 source code를 대상으로 static analysis를 수행, 결과 발표
 - 3개의 도구 선택 (복잡도 or 의존성 분석 도구 1개 반드시 포함)
 - 설명한 도구를 포함해 많은 도구들 중 자유롭게 선택
 - 분석 결과 중 critical 한 부분들에 대한 분석발표

1	4, 3
2	4, 1
3	1, 2
4	2, 3
5	8, 7
6	8, 5
7	5, 6
8	6, 7