Lab 3.1 Using splint for C static analysis

Overview

The learning objective of this lab is for students to gain the first-hand experience on using static code analysis tools to check c program for security vulnerabilities and coding mistakes.

Splint link is a tool for statically checking C programs for security vulnerabilities and programming mistakes. Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information.

11 kinds of problems detected by Splint include:

- Dereferencing a possibly null pointer;
- Using possibly undefined storage or returning storage that is not properly defined;
- Type mismatches, with greater precision and flexibility than provided by C compilers;
- Violations of information hiding;
- Memory management errors including uses of dangling references and memory leaks;
- Dangerous aliasing;
- Modifications and global variable uses that are inconsistent with specified interfaces;
- Problematic control flow such as likely infinite loops, fall through cases or incomplete switches, and suspicious statements;
- Buffer overflow vulnerabilities;
- Dangerous macro implementations or invocations;
- Violations of customized naming conventions.

More details you can get from Splint User's Manual link.

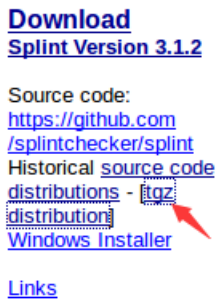
With such knowledge, your goal is to achieve the followings:

- Install splint;
- Finish code samples with 2 different kinds of problems which can be detected by Splint. You can choose any 2 of 11 problems as above.
- Use splint to detect the 2 kinds of problems. Descibe your observations in your report.

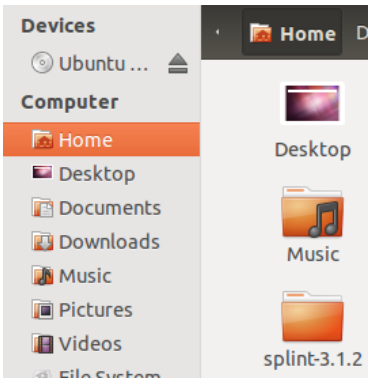
SP2021-LAB3.1&3.2-3190300677-姚熙源

Steps :

1. Download tgz distribution on the splint.org website.



2. Extract the .tgz file and copy the splint-3.1.2 file to home page.



3. Setup Splint.

```
peteryap@peteryap-vm:~$ cd splint-3.1.2
bash: cd: splint-3.1.2: No such file or directory
peteryap@peteryap-vm:~$ cd splint-3.1.2
peteryap@peteryap-vm:~/splint-3.1.2$ ./configure --prefix=/usr/local/splint
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for flex... no
checking for lex... no
checking for yywrap in -lfl... no
checking for yywrap in -ll... no
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets $(MAKE)... (cached) yes
checking whether ln -s works... yes
```

```
checking whether make sets $(MAKE)... (cached) yes
checking whether ln -s works... yes
checking for bison... no
checking for grep... grep
checking for diff... diff
checking for cat... cat
checking for rm... rm
checking for mv... mv
checking for cp... cp
checking for sed... sed
checking whether we need _ALL_SOURCE to expose mode_t... no
checking whether to include support for LCL files... yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating imports/Makefile
config.status: creating lib/Makefile
config.status: creating src/Makefile
config.status: creating test/Makefile
config.status: creating doc/Makefile
config.status: creating bin/Makefile
config.status: creating config.h
config.status: executing depfiles commands
```

Command “sudo apt-get install flex”

```
peteryap@peteryap-vm:~/splint-3.1.2$ sudo apt-get install flex
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libfl-dev m4
Suggested packages:
  bison build-essential
The following NEW packages will be installed:
  flex libfl-dev m4
0 upgraded, 3 newly installed, 0 to remove and 420 not upgraded.
Need to get 451 kB of archives.
After this operation, 1,006 kB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://old-releases.ubuntu.com/ubuntu/ precise/main m4 amd64 1.4.16-2ubuntu
1 [200 kB]
Get:2 http://old-releases.ubuntu.com/ubuntu/ precise/main libfl-dev amd64 2.5.35-
10ubuntu3 [18.9 kB]
Get:3 http://old-releases.ubuntu.com/ubuntu/ precise/main flex amd64 2.5.35-10ubu
ntu3 [232 kB]
Fetched 451 kB in 2s (154 kB/s)
Selecting previously unselected package m4.
(Reading database ... 143882 files and directories currently installed.)
Unpacking m4 (from .../m4_1.4.16-2ubuntu1_amd64.deb) ...
Selecting previously unselected package libfl-dev.
Unpacking libfl-dev (from .../libfl-dev_2.5.35-10ubuntu3_amd64.deb) ...
Selecting previously unselected package flex.
Unpacking flex (from .../flex_2.5.35-10ubuntu3_amd64.deb) ...
Processing triggers for install-info ...
Processing triggers for man-db ...
```

Command “make” configure too much file, so there is no screenshots.

Command “sudo make install” same too.

4. Use vi Text Editor to add the following to the environment variable.
   export LARCH_PATH=/usr/local/splint/share/splint/lib
   export LCLIMPORTDIR=/usr/splint/share/splint/imports
   export PATH=$PATH:/usr/local/splint/bin

```
peteryap@peteryap-vm:~/splint-3.1.2$ vi ~/.bashrc
peteryap@peteryap-vm:~/splint-3.1.2$ cat ~/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
export LARCH_PATH=/usr/local/splint/share/splint/lib
export LCLIMPORTDIR=/usr/splint/share/splint/imports
export PATH=$PATH:/usr/local/splint/bin
[ -z "$PS1" ] && return
```

5. Execute source ~/.bashrc

6. Write a code example with two different kind of problems. “vi test.c”

```
#include<stdio.h>

int main()
{
        int a,b,c;
        a = 100;
        while(1){
                a++;
        }
        return 0;
}
```

Problems exist are dead loop and variable has not been used in program.

7. Use splint to detect the problems in the c program.

```
peteryap@peteryap-vm:~/splint-3.1.2$ splint test.c
Splint 3.1.2 --- 03 May 2009

Cannot find standard library: standard.lcd
     Check LARCH_PATH environment variable.
test.c: (in function main)
test.c:7:8: Test expression for while not boolean, type int: 1
  Test expression type is not boolean or int. (Use -predboolint to inhibit
  warning)
test.c:10:9: Unreachable code: return 0
  This code will never be reached on any possible execution. (Use -unreachable
  to inhibit warning)
test.c:5:8: Variable b declared but not used
  A variable is declared but never used. Use /*@unused@*/ in front of
  declaration to suppress message. (Use -varuse to inhibit warning)
test.c:5:10: Variable c declared but not used

Finished checking --- 4 code warnings
```

8. Splint warning shows that the while expression is not Boolean and exists in dead loop, variable is declared but never used in the program. The problems exist in the c program are found, then the lab is completed.

Lab 3.2 Using eclipse for java static analysis

Overview

The learning objective of this lab is for students to gain the first-hand experience on using static code analyzers in Eclipse to check Java program for security vulnerabilities and coding mistakes.

In this Lab, your goal is to achieve the followings:

- Install plugins in Java;
- Learn to check Java code by using static code analyzers in Eclipse. Descibe your observations in your report.

Open Source Code Analyzers in Java

Here we introduce 3 kinds of open source code analyzers in Java.

FindBugs

FindBugs looks for bugs in Java programs. It can detect a variety of common coding mistakes, including thread synchronization problems, misuse of API methods, etc. Go To FindBugs link.

PMD

PMD scans Java source code and looks for potential problems like:

* Unused local variables
* Empty catch blocks
* Unused parameters
* Empty 'if' statements
* Duplicate import statements
* Unused private methods
* Classes which could be Singletons
* Short/long variable and method names

Go To PMD link.

Checkstyle

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 file is supplied supporting the Sun Code Conventions. As well, other

sample configuration files are supplied for other well known conventions. Can be integrated into CruiseControl and Eclipse. Go To Checkstyle link.

Two Ways of Installing Eclipse Plugin

Here we introduce two normal ways of installing Eclipse Plugins.

Install plugins online

All plugins can be installed with the standard update manager of Eclipse. Following is the procedure:

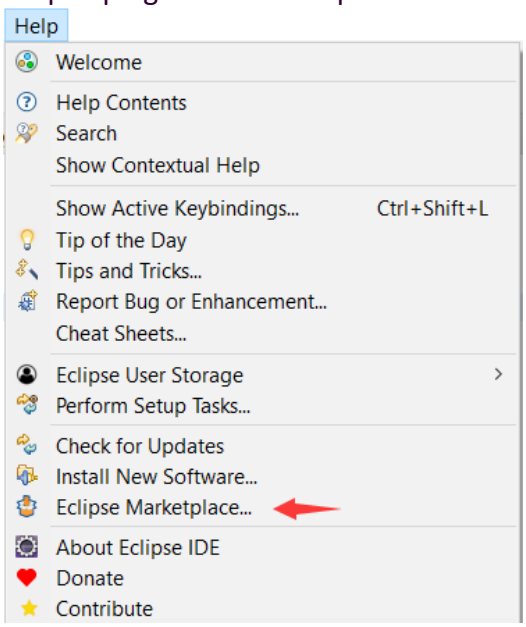
* Choose the menu item "Help->Intall New Software".
* Click on "Add..." and enter a name and the URL of the plugin.
* Now you can select the feature (plugin) you wish to install.
* Confirm the upcoming dialogs and restart Eclipse.

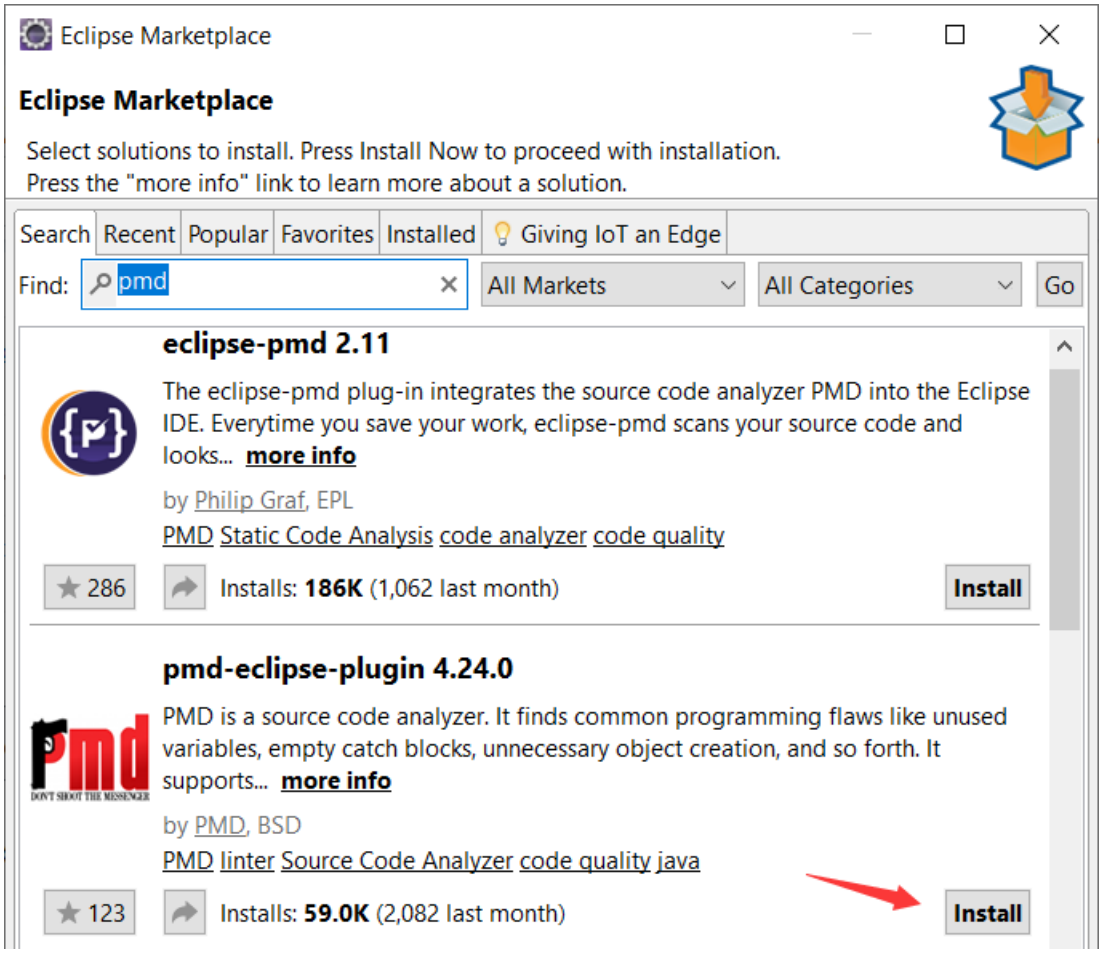
Install plugins offline

For Eclipse 3.4 or later, you can install downloaded plugins by copying the contents of plugins to the folder "dropins". And you can delete the plugins in folder "dropins" to uninstall plugins. After installing or uninstalling, you need to restart Eclipse.

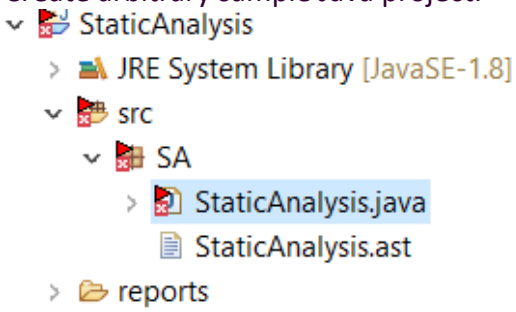
Steps :

1. Install PMD Plugins
   (Because the link http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/ cannot be add to the repository, so we download PMD plugin in others way.)
   Eclipse -> Help -> Eclipse MarketPlace -> Search “PMD” in the box -> Select the PMD-eclipse-plugin but not Eclipse-PMD. Click on install and restart eclipse then its done.





2. Create arbitrary sample Java project.



SP2021-LAB3.1&3.2-3190300677-姚熙源

Java Code

```
package SA;

public class StaticAnalysis {

    public static void main(String[] args) {
        int a,b,c,d,e;
        a = 1;
        b = 0;
        c = cmp_max(a,b);
        while(c + 1 == 2) {
            c = 1;
        }
    }

    public static int cmp_max(int a, int b) {
        if(a>b) {
            return a;
        }
        else {
            return b;
        }
    }

}
```

Check Code Violation

Violations Overview

Element	# Violations	# Violations...	# Violations...	Project
SA	23	N/A	N/A	StaticAnalysis
StaticAnalysis.java	23	N/A	N/A	StaticAnalysis
UseUtilityClass	1	N/A	N/A	StaticAnalysis
LocalVariableCouldBeFinal	2	N/A	N/A	StaticAnalysis
ClassNamingConventions	1	N/A	N/A	StaticAnalysis
MethodArgumentCouldBe	3	N/A	N/A	StaticAnalysis
MethodNamingConventior	1	N/A	N/A	StaticAnalysis
OnlyOneReturn	1	N/A	N/A	StaticAnalysis
CommentRequired	3	N/A	N/A	StaticAnalysis
OneDeclarationPerLine	1	N/A	N/A	StaticAnalysis
PackageCase	1	N/A	N/A	StaticAnalysis
ShortVariable	7	N/A	N/A	StaticAnalysis
UnusedLocalVariable	2	N/A	N/A	StaticAnalysis

Violations Outline

P	Line	created	Rule	Error Message
	1	Wed Jun 16 19:05:05 MYT 2021	PackageCase	PackageCase: Package name contains upper case characters
	3	Wed Jun 16 19:05:05 MYT 2021	ClassNamingConventions	ClassNamingConventions: The utility class name 'StaticAnalysis' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper|Constants)'
	3	Wed Jun 16 19:05:05 MYT 2021	CommentRequired	CommentRequired: Class comments are required
	3	Wed Jun 16 19:05:05 MYT 2021	UseUtilityClass	UseUtilityClass: All methods are static. Consider using a utility class instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
	5	Wed Jun 16 19:05:05 MYT 2021	CommentRequired	CommentRequired: Public method and constructor comments are required
	5	Wed Jun 16 19:05:05 MYT 2021	MethodArgumentCouldB...	MethodArgumentCouldBeFinal: Parameter 'args' is not assigned and could be declared final
	6	Wed Jun 16 19:05:05 MYT 2021	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'e' could be declared final
	6	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like a
	6	Wed Jun 16 19:05:05 MYT 2021	UnusedLocalVariable	UnusedLocalVariable: Avoid unused local variables such as 'd'.
	6	Wed Jun 16 19:05:05 MYT 2021	UnusedLocalVariable	UnusedLocalVariable: Avoid unused local variables such as 'e'.
	6	Wed Jun 16 19:05:05 MYT 2021	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'd' could be declared final
	6	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like d
	6	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like e
	6	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like c
	6	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like b
	6	Wed Jun 16 19:05:05 MYT 2021	OneDeclarationPerLine	OneDeclarationPerLine: Use one line for each declaration, it enhances code readability.
	15	Wed Jun 16 19:05:05 MYT 2021	MethodNamingConventi...	MethodNamingConventions: The static method name 'cmp_max' doesn't match '[a-z][a-zA-Z0-9]*'
	15	Wed Jun 16 19:05:05 MYT 2021	MethodArgumentCouldB...	MethodArgumentCouldBeFinal: Parameter 'a' is not assigned and could be declared final
	15	Wed Jun 16 19:05:05 MYT 2021	MethodArgumentCouldB...	MethodArgumentCouldBeFinal: Parameter 'b' is not assigned and could be declared final
	15	Wed Jun 16 19:05:05 MYT 2021	CommentRequired	CommentRequired: Public method and constructor comments are required
	15	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like a
	15	Wed Jun 16 19:05:05 MYT 2021	ShortVariable	ShortVariable: Avoid variables with short names like b
	17	Wed Jun 16 19:05:05 MYT 2021	OnlyOneReturn	OnlyOneReturn: A method should have only one exit point, and that should be the last statement in the method

There is a pmd-report text file showing the analysis.

```
src/SA/StaticAnalysis.java:1:   PackageCase:    PackageCase: Package name contains upper case characters
src/SA/StaticAnalysis.java:1:   PackageCase:    PackageCase: Package name contains upper case characters
src/SA/StaticAnalysis.java:3:   ClassNamingConventions: ClassNamingConventions: The utility class name 'StaticAnalysis' doesn't
src/SA/StaticAnalysis.java:3:   ClassNamingConventions: ClassNamingConventions: The utility class name 'StaticAnalysis' doesn't
src/SA/StaticAnalysis.java:3:   CommentRequired:    CommentRequired: Class comments are required
src/SA/StaticAnalysis.java:3:   CommentRequired:    CommentRequired: Class comments are required
src/SA/StaticAnalysis.java:3:   UseUtilityClass:    UseUtilityClass: All methods are static.  Consider using a utility class in
src/SA/StaticAnalysis.java:3:   UseUtilityClass:    UseUtilityClass: All methods are static.  Consider using a utility class in
src/SA/StaticAnalysis.java:5:   CommentRequired:    CommentRequired: Public method and constructor comments are required
src/SA/StaticAnalysis.java:5:   CommentRequired:    CommentRequired: Public method and constructor comments are required
src/SA/StaticAnalysis.java:5:   MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'args' is not assigned and co
src/SA/StaticAnalysis.java:5:   MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'args' is not assigned and co
src/SA/StaticAnalysis.java:6:   LocalVariableCouldBeFinal:  LocalVariableCouldBeFinal: Local variable 'd' could be declared fin
src/SA/StaticAnalysis.java:6:   LocalVariableCouldBeFinal:  LocalVariableCouldBeFinal: Local variable 'd' could be declared fin
src/SA/StaticAnalysis.java:6:   LocalVariableCouldBeFinal:  LocalVariableCouldBeFinal: Local variable 'e' could be declared fin
src/SA/StaticAnalysis.java:6:   LocalVariableCouldBeFinal:  LocalVariableCouldBeFinal: Local variable 'e' could be declared fin
src/SA/StaticAnalysis.java:6:   OneDeclarationPerLine:  OneDeclarationPerLine: Use one line for each declaration, it enhances c
src/SA/StaticAnalysis.java:6:   OneDeclarationPerLine:  OneDeclarationPerLine: Use one line for each declaration, it enhances c
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like a
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like a
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like b
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like b
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like c
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like c
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like d
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like d
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like e
src/SA/StaticAnalysis.java:6:   ShortVariable:  ShortVariable: Avoid variables with short names like e
src/SA/StaticAnalysis.java:6:   UnusedLocalVariable:    UnusedLocalVariable: Avoid unused local variables such as 'd'.
src/SA/StaticAnalysis.java:6:   UnusedLocalVariable:    UnusedLocalVariable: Avoid unused local variables such as 'd'.
src/SA/StaticAnalysis.java:6:   UnusedLocalVariable:    UnusedLocalVariable: Avoid unused local variables such as 'e'.
src/SA/StaticAnalysis.java:6:   UnusedLocalVariable:    UnusedLocalVariable: Avoid unused local variables such as 'e'.
src/SA/StaticAnalysis.java:15:  CommentRequired:    CommentRequired: Public method and constructor comments are required
src/SA/StaticAnalysis.java:15:  CommentRequired:    CommentRequired: Public method and constructor comments are required
src/SA/StaticAnalysis.java:15:  MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'a' is not assigned and could
src/SA/StaticAnalysis.java:15:  MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'a' is not assigned and could
src/SA/StaticAnalysis.java:15:  MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'b' is not assigned and could
src/SA/StaticAnalysis.java:15:  MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'b' is not assigned and could
```

3. PMD plugin help us to find those problems in the java program. We can click the violation and see whether which problem exists. This lab is completed.