SP Lab 3.1&3.2

Name: 吴欣倍

StuId: 3190103044

Lab 3.1

Lab: Using splint for C static analysis

Overview

Objective

Gain the first-hand experience on using static code analysis tools to check c program for security vulnerabilities and coding mistakes.

Goal

- 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.

Procedure

1. Download Splint source code distribution and setup Splint.

```
$ tar -zxvf splint-3.1.2.linux.tgz
$ sudo mkdir /usr/local/splint
  [sudo]password for ***: (enter password)
$ cd splint-3.1.2
$ ./configure -prefix=/usr/local/splint
$ sudo make install
```

Note: the command given by TA has some mistakes. "--prefix=" should be modified as "-prefix=".

When execute `sudo make install`, an error occurs. We can deal with the error by directly adding the `yyswap` function into `cscanner.c`. Save file and execute `sudo make install` again.

```
int yywrap() {
    return 1;
}
```

Use vi Text Editor to add the following to the environment variable:

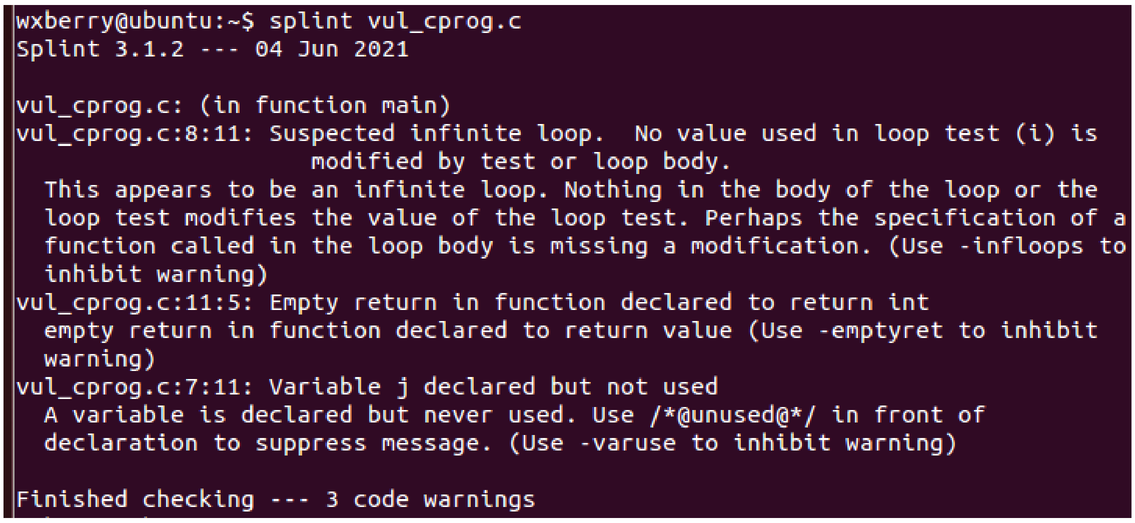
```
$ vi ~/.bashrc
export LARCH_PATH=/usr/local/splint/share/splint/lib
export LCLIMPORTDIR=/usr/splint/share/splint/imports
export PATH=$PATH:/usr/local/splint/bin
$ source ~/.bashrc
```

2. Write a program named `vul_cprog.c` in c within 3 problems.

```
#include <stdio.h>
#include <stdlib.h>
// problem 1: variable j isn's used.
// problem 2: infinite loop.
// problem 3: empty return.
int main() {
    int i=2, j;
    while(i==2) {
        printf("hellp, world\n");
    }
    return ;
}
```

3. Use Splint to detect problems in code sample `vul_cprog.c`

   In the figure shown below, we can find that all the 3 problems are detected.



```
wxberry@ubuntu:~$ splint vul_cprog.c
Splint 3.1.2 --- 04 Jun 2021

vul_cprog.c: (in function main)
vul_cprog.c:8:11: Suspected infinite loop.  No value used in loop test (i) is
                     modified by test or loop body.
  This appears to be an infinite loop. Nothing in the body of the loop or the
  loop test modifies the value of the loop test. Perhaps the specification of a
  function called in the loop body is missing a modification. (Use -infloops to
  inhibit warning)
vul_cprog.c:11:5: Empty return in function declared to return int
  empty return in function declared to return value (Use -emptyret to inhibit
  warning)
vul_cprog.c:7:11: Variable j 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)

Finished checking --- 3 code warnings
```

   Succeed.

Lab 3.2

Lab: Using eclipse for java static analysis

Overview

Objective

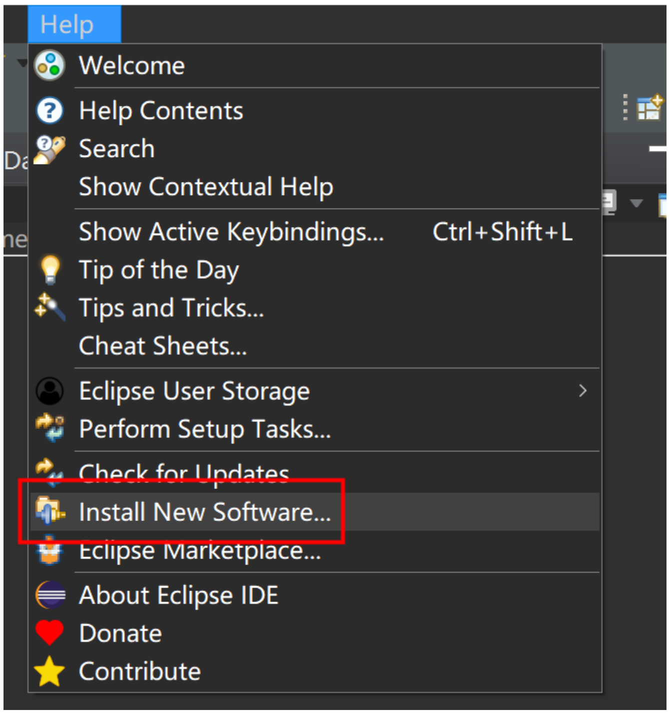
Gain the first-hand experience on using static code analyzers in Eclipse to check Java program for security vulnerabilities and coding mistakes.

Goal

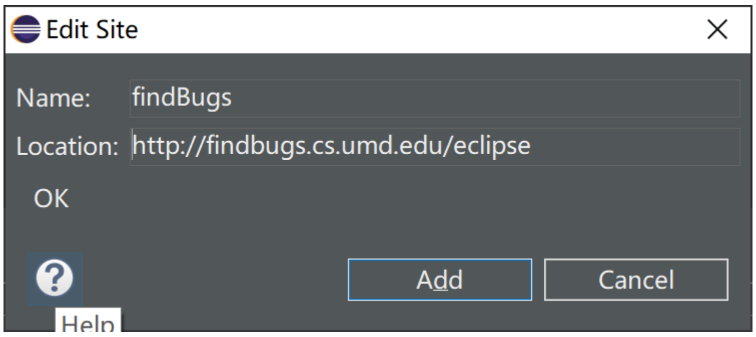
- Install plugins in Java.
- Learn to check Java code by using static code analyzers in Eclipse. Descibe your observations in your report.

Procedure

1. I chose `PMD` as my code analyzer plugin in Eclipse. Find the installation instructions in URL(https://pmd.github.io/pmd-eclipse-plugin-p2-site/4.24.0.v20210529-0600/). Follow the instructions to install it.
   - Download and decompress the file.
   - Click "Help->Install New Software". Add the file.



   - Click "Add", enter Name and Location as below. Click "Add" to search appropriate plugin.

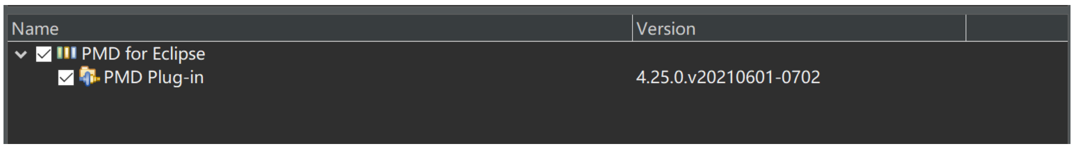

Edit Site

Name: findBugs

Location: http://findbugs.cs.umd.edu/eclipse

OK

   - Select appropriate plugin and click "Next->Next->Finish".



Name	Version
☑ PMD for Eclipse	
☑ PMD Plug-in	4.25.0.v20210601-0702

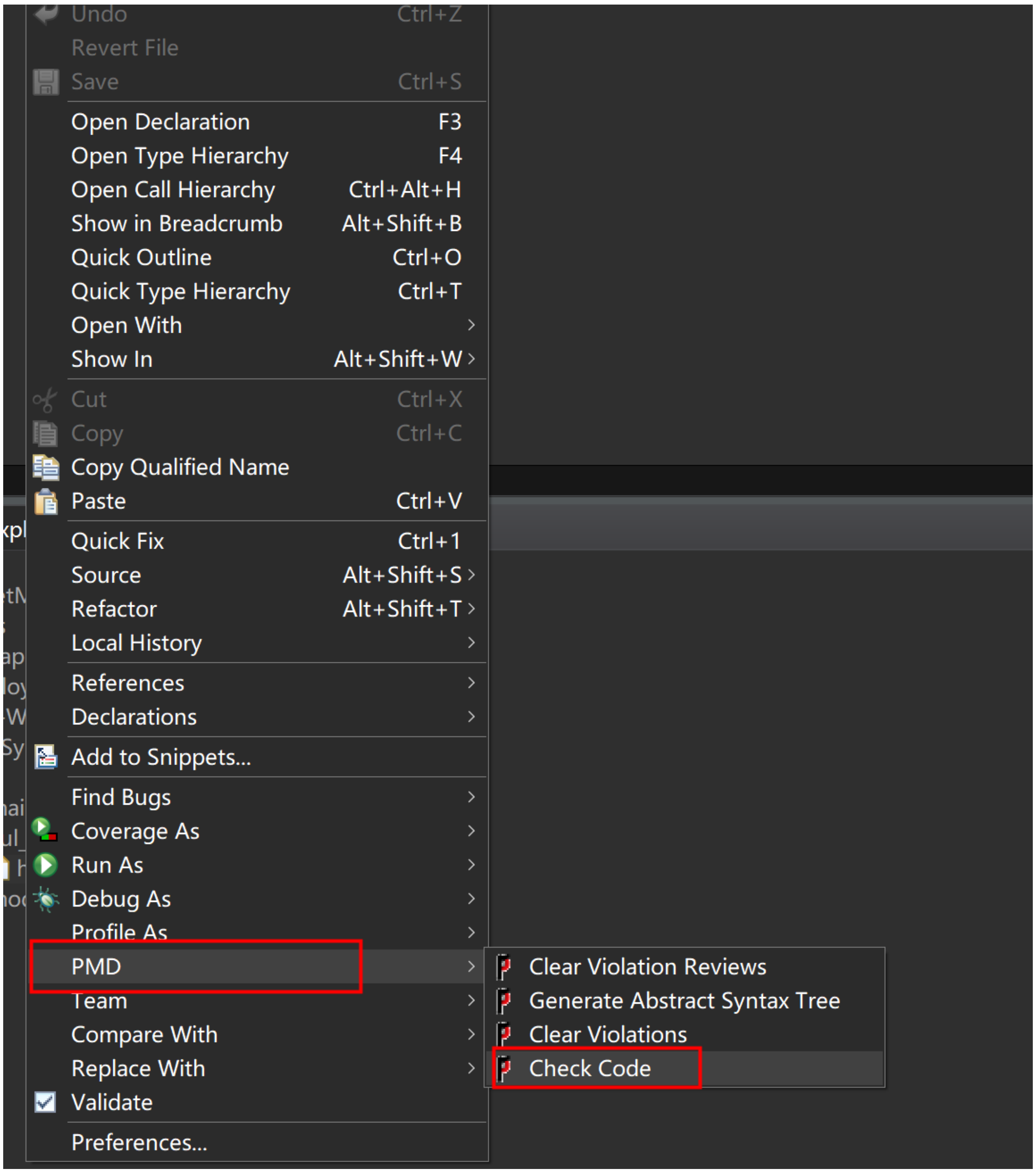
   Waiting for installation. After installation finished, restart Eclipse.

2. I write a java application as testing sample.

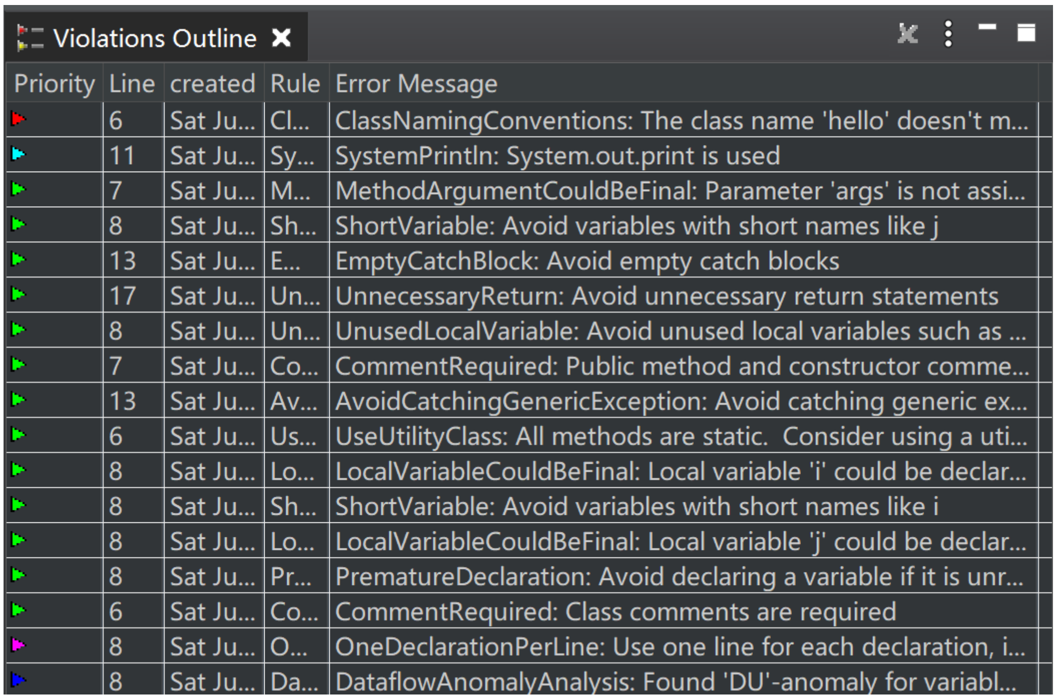
```
package vul_javaprog;
```

```java
// problem 1: unused variable j.
// problem 2: infinite loop.
// problem 3: empty return.
// problem 4: empty catch.
public class hello {
    public static void main(String args[]) {
        int i=2, j;
        try {
            while(i==2) {
                System.out.print("hello, world!\n");
            }
        } catch (Exception e) {

        }
        return ;
    }
}
```



3. The check report is shown below:


Violations Outline

Priority	Line	created	Rule	Error Message
▶	6	Sat Ju...	Cl...	ClassNamingConventions: The class name 'hello' doesn't m...
▶	11	Sat Ju...	Sy...	SystemPrintln: System.out.print is used
▶	7	Sat Ju...	M...	MethodArgumentCouldBeFinal: Parameter 'args' is not assi...
▶	8	Sat Ju...	Sh...	ShortVariable: Avoid variables with short names like j
▶	13	Sat Ju...	E...	EmptyCatchBlock: Avoid empty catch blocks
▶	17	Sat Ju...	Un...	UnnecessaryReturn: Avoid unnecessary return statements
▶	8	Sat Ju...	Un...	UnusedLocalVariable: Avoid unused local variables such as ...
▶	7	Sat Ju...	Co...	CommentRequired: Public method and constructor comme...
▶	13	Sat Ju...	Av...	AvoidCatchingGenericException: Avoid catching generic ex...
▶	6	Sat Ju...	Us...	UseUtilityClass: All methods are static. Consider using a uti...
▶	8	Sat Ju...	Lo...	LocalVariableCouldBeFinal: Local variable 'i' could be declar...
▶	8	Sat Ju...	Sh...	ShortVariable: Avoid variables with short names like i
▶	8	Sat Ju...	Lo...	LocalVariableCouldBeFinal: Local variable 'j' could be declar...
▶	8	Sat Ju...	Pr...	PrematureDeclaration: Avoid declaring a variable if it is unr...
▶	6	Sat Ju...	Co...	CommentRequired: Class comments are required
▶	8	Sat Ju...	O...	OneDeclarationPerLine: Use one line for each declaration, i...
▶	8	Sat Ju...	Da...	DataflowAnomalyAnalysis: Found 'DU'-anomaly for variabl...

In this report, we can find that `EmptyCatchBlock`, `UnnecessaryReturn`, `UnusedLocalVaraible` warnings are reported.

But PMD didn't report infinite loop error. Maybe because that is a logic error rather than syntax error.

Succeed.