Lab 3.1 Using splint for C static analysis

Overview

这个实验主要是教我们如何使用 c 静态分析工具splint。

Step 1 下载Splint

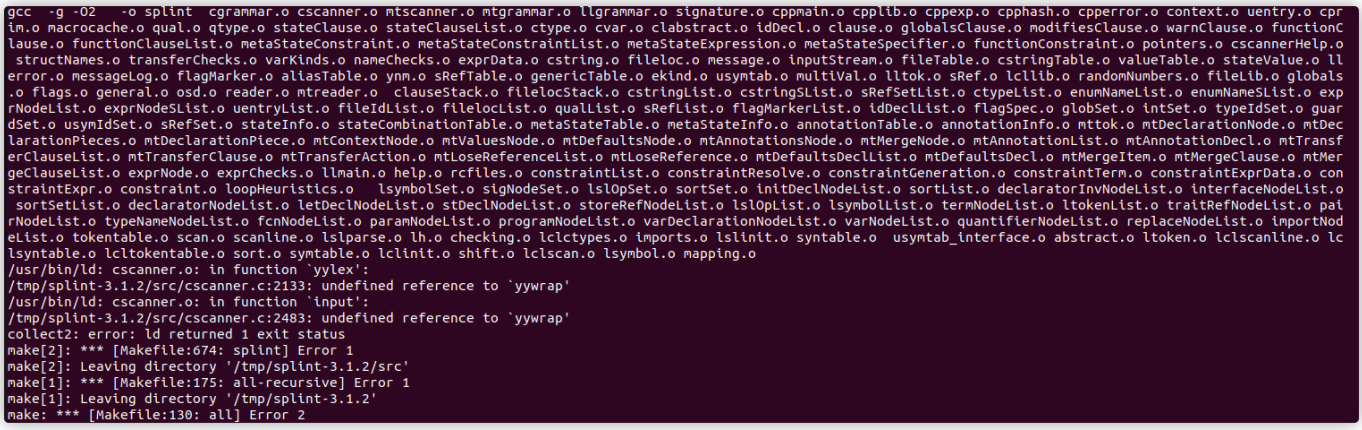
这里按照给出的链接下载就行。

Step 2 Extract and setup Splint.

这一步主要是在虚拟机中编译splint，生成可执行的splint程序，运行的命令如下。

```
tar -zxvf splint-3.1.2.linux.tgz
sudo mkdir /usr/local/splint
cd spint-3.1.2
sudo apt-get install flex
make -j4
sudo make install
```

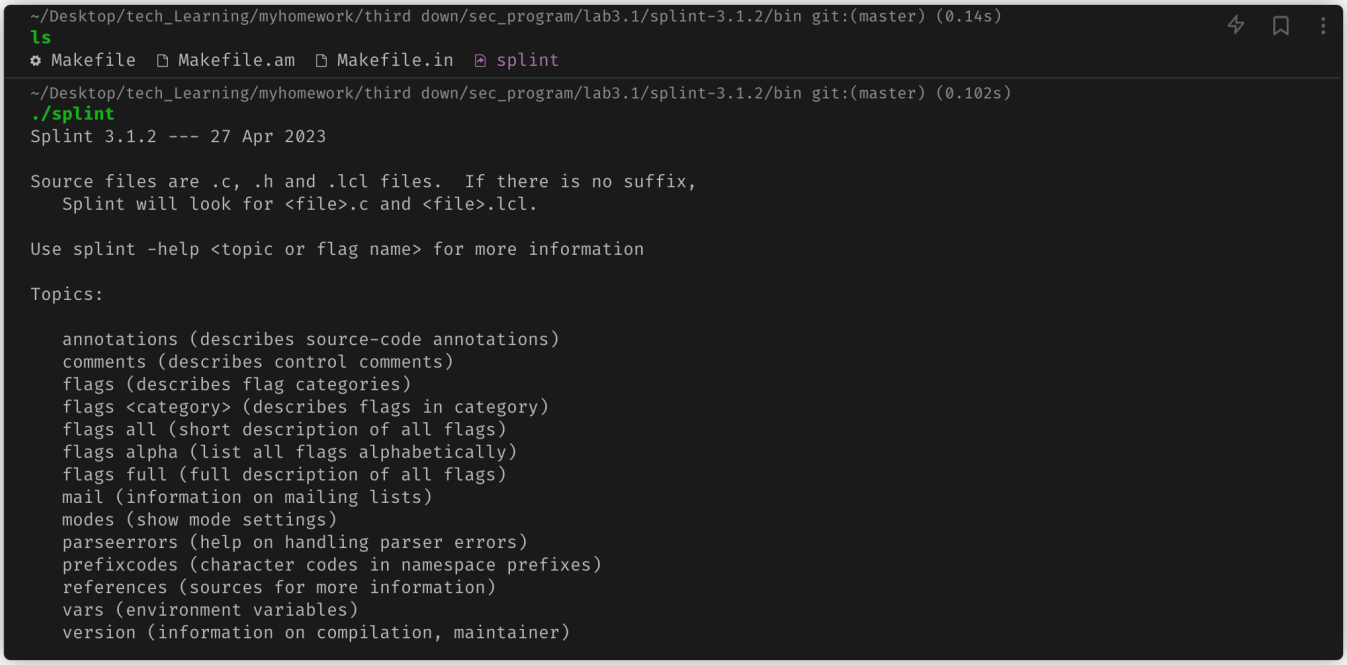
需要注意的是，在执行 `make -j4` 的时候编译器输出了错误信息，如图所示。



```
gcc  -g -O2   -o splint  cgrammar.o cscanner.o mtscanner.o mtgrammar.o llgrammar.o signature.o cppmain.o cpplib.o cppexp.o cpphash.o cpperror.o context.o uentry.o cpr
im.o macrocache.o qual.o qtype.o stateClause.o stateClauseList.o ctype.o cvar.o clabstract.o idDecl.o clause.o globalsClause.o modifiesClause.o warnClause.o functionC
lause.o functionClauseList.o metaStateConstraint.o metaStateConstraintList.o metaStateExpression.o metaStateSpecifier.o functionConstraint.o pointers.o cscannerHelp.o
 structNames.o transferChecks.o varKinds.o nameChecks.o exprData.o cstring.o fileloc.o message.o inputStream.o fileTable.o cstringTable.o valueTable.o stateValue.o ll
error.o messageLog.o flagMarker.o aliasTable.o ynm.o sRefTable.o genericTable.o ekind.o usymtab.o multiVal.o lltok.o sRef.o lcllib.o randomNumbers.o fileLib.o globals
.o flags.o general.o osd.o reader.o mtreader.o  clauseStack.o filelocStack.o cstringList.o cstringSList.o sRefSetList.o ctypeList.o enumNameList.o enumNameSList.o exp
rNodeList.o exprNodeSList.o uentryList.o fileIdList.o filelocList.o qualList.o sRefList.o flagMarkerList.o idDeclList.o flagSpec.o globSet.o intSet.o typeIdSet.o guar
dSet.o usymIdSet.o sRefSet.o stateInfo.o stateCombinationTable.o metaStateTable.o metaStateInfo.o annotationTable.o annotationInfo.o mttok.o mtDeclarationNode.o mtDec
larationPieces.o mtDeclarationPiece.o mtContextNode.o mtValuesNode.o mtDefaultsNode.o mtAnnotationsNode.o mtMergeNode.o mtAnnotationList.o mtAnnotationDecl.o mtTransf
erClauseList.o mtTransferClause.o mtTransferAction.o mtLoseReferenceList.o mtLoseReference.o mtDefaultsDeclList.o mtDefaultsDecl.o mtMergeItem.o mtMergeClause.o mtMer
geClauseList.o exprNode.o exprChecks.o llmain.o help.o rcfiles.o constraintList.o constraintResolve.o constraintGeneration.o constraintTerm.o constraintExprData.o con
straintExpr.o constraint.o loopHeuristics.o   lsymbolSet.o sigNodeSet.o lslOpSet.o sortSet.o initDeclNodeList.o sortList.o declaratorInvNodeList.o interfaceNodeList.o
 sortSetList.o declaratorNodeList.o letDeclNodeList.o stDeclNodeList.o storeRefNodeList.o lslOpList.o lsymbolList.o termNodeList.o ltokenList.o traitRefNodeList.o pai
rNodeList.o typeNameNodeList.o fcnNodeList.o paramNodeList.o programNodeList.o varDeclarationNodeList.o varNodeList.o quantifierNodeList.o replaceNodeList.o importNod
eList.o tokentable.o scan.o scanline.o lslparse.o lh.o checking.o lclctypes.o imports.o lslinit.o syntable.o  usymtab_interface.o abstract.o ltoken.o lclscanline.o lc
lsyntable.o lcltokentable.o sort.o symtable.o lclinit.o shift.o lclscan.o lsymbol.o mapping.o
/usr/bin/ld: cscanner.o: in function `yylex':
/tmp/splint-3.1.2/src/cscanner.c:2133: undefined reference to `yywrap'
/usr/bin/ld: cscanner.o: in function `input':
/tmp/splint-3.1.2/src/cscanner.c:2483: undefined reference to `yywrap'
collect2: error: ld returned 1 exit status
make[2]: *** [Makefile:674: splint] Error 1
make[2]: Leaving directory '/tmp/splint-3.1.2/src'
make[1]: *** [Makefile:175: all-recursive] Error 1
make[1]: Leaving directory '/tmp/splint-3.1.2'
make: *** [Makefile:130: all] Error 2
```

Google 之后发现这好像是 flex 的 bug，调试之后展示无法解决，所以在接下来的部分我迁移到了本机（maxOS 上完成），大部分命令都相同，只是需要使用`brew install flex`安装一下flex套件即可。

在 macOS 中编译完之后可以在bin文件夹下找到splint文件，如图所示。



Step 3 编写有问题的代码

```c
#include<stdio.h>

int main(){
    // unused vars;
    int a;
    int b;
    // dereferencing null pointer
    int *pointer = NULL;
    a = *pointer;
    return 0;
}

```

Step 4 使用 splint 嗅探

这里在命令行执行 `./splint sample.c` 就行，可以看到splint输出了我们故意留下的问题，分别是存在未使用的变量，以及解引用空指针。

```
~/Desktop/tech_Learning/myhomework/third down/sec_program/lab3.1/splint-3.1.2/bin git:(master) (0.112s)
./splint test.c
Splint 3.1.2 --- 27 Apr 2023

test.c: (in function main)
test.c:9:10: Dereference of null pointer pointer: *pointer
  A possibly null pointer is dereferenced.  Value is either the result of a
  function which may return null (in which case, code should check it is not
  null), or a global, parameter or structure field declared with the null
  qualifier. (Use -nullderef to inhibit warning)
   test.c:8:20: Storage pointer becomes null
test.c:6:6: 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)

Finished checking --- 2 code warnings
```

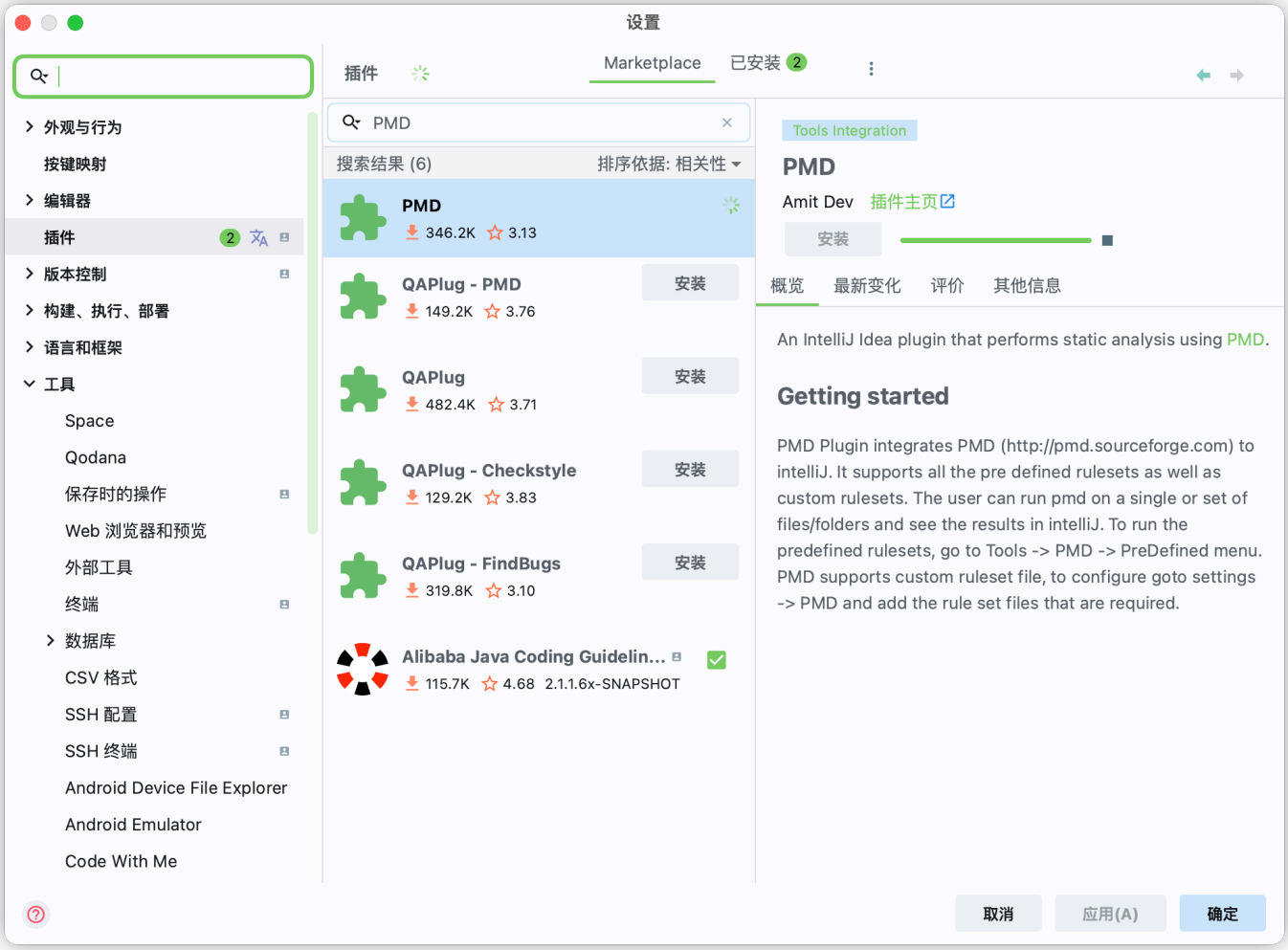
Lab 3.2 Using eclipse for java static analysis

Overview

这个实验主要是教授我们使用 eclipse 进行 java 语言的静态分析。

Step 1 Install plugins

由于我的电脑之前没有安装过eclipse，所以这里使用功能类似的 Intellij IDEA 完成实验。
我们打开 IDEA 的插件市场，这里我选择 PMD 进行安装。



Step 2 编辑问题代码

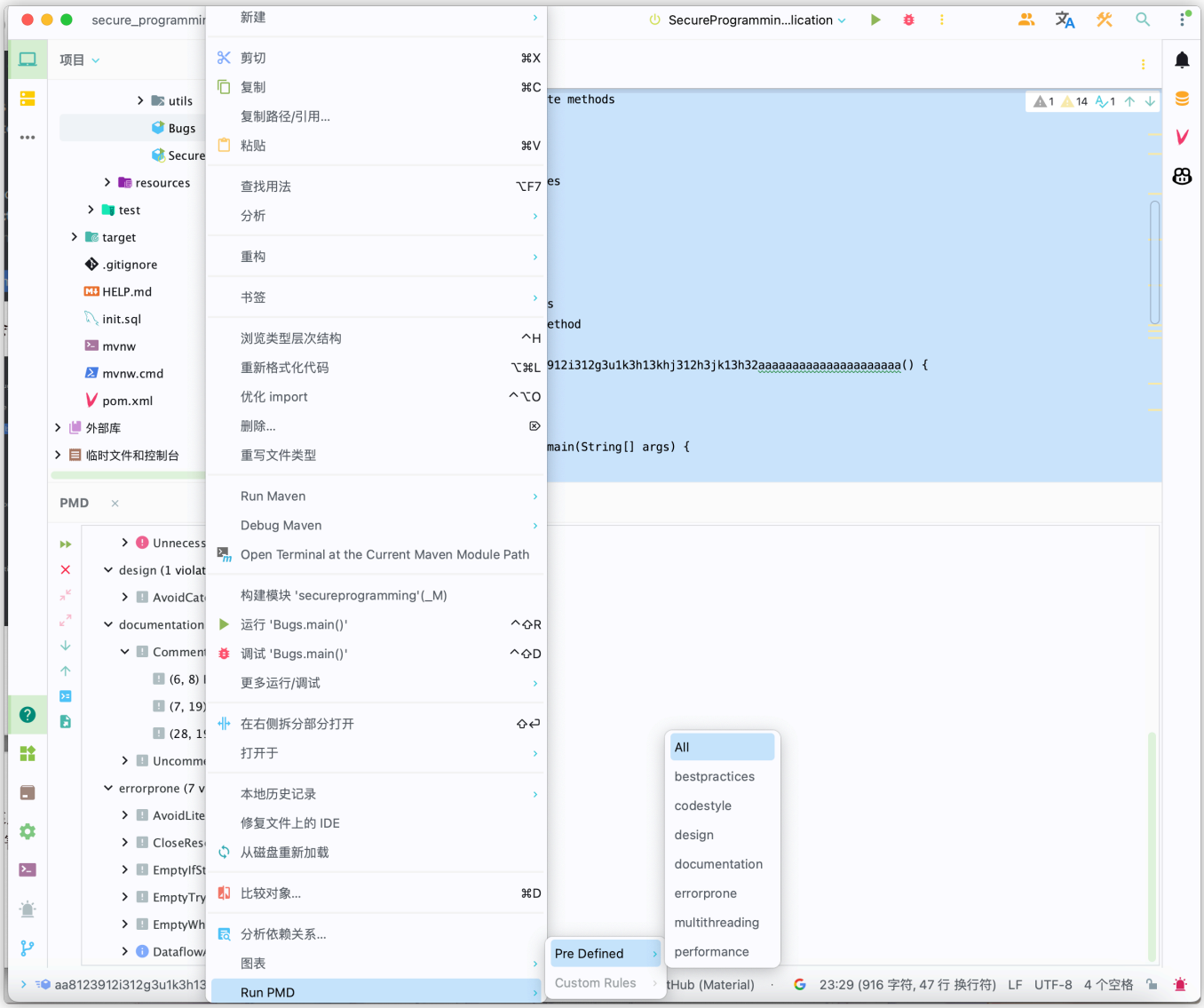
这里使用一个存在许多 bug 的 java 程序为例。

```java
package com.htl.secure_programming;

import java.util.Scanner;
import java.util.Scanner; // duplicate import

public class Bugs {
    public static void def(int a) {
        // unused method parameters
        return;
    }

    private void unusedMethod() {
        // unused private methods
        return;
    }

    // short method names
    private void a() {

    }

    // long method names
    // unused private method
    private void aa8123912i312g3u1k3h13khj312h3jk13h32aaaaaaaaaaaaaaaaaaaaa() {

    }

    public static void main(String[] args) {
        int a = 0;
        int b; // unused local vars
        Scanner in = new Scanner(System.in);
        try {

        } catch (Exception e) {
            // empty catch blocks
            def(a);
        }
        // always true
        if (1 == 1) {
            // empty block
        }
```

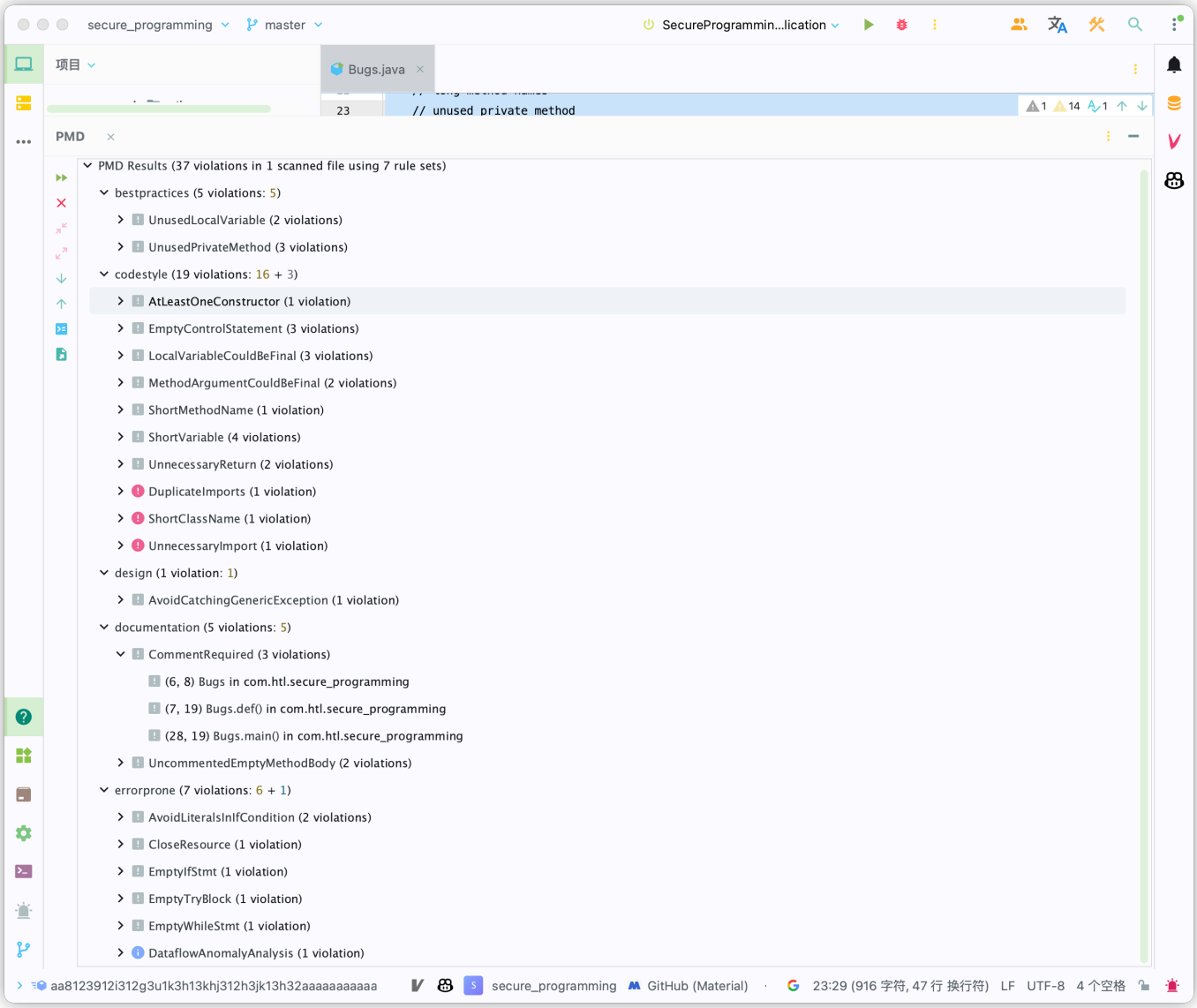
```
42          // dead loop
43          while (true) {
44
45          }
46      }
47  }
48
```

Step 3 运行 PMD

这里右击文件，选择runpmd→predefined→all，就可以运行扫描。



可以看到 PMD 插件扫描出了众多代码中存在的问题。


secure_programming
master
SecureProgrammin...lication
项目
Bugs.java
23
// unused private method
PMD
PMD Results (37 violations in 1 scanned file using 7 rule sets)
bestpractices (5 violations: 5)
UnusedLocalVariable (2 violations)
UnusedPrivateMethod (3 violations)
codestyle (19 violations: 16 + 3)
AtLeastOneConstructor (1 violation)
EmptyControlStatement (3 violations)
LocalVariableCouldBeFinal (3 violations)
MethodArgumentCouldBeFinal (2 violations)
ShortMethodName (1 violation)
ShortVariable (4 violations)
UnnecessaryReturn (2 violations)
DuplicateImports (1 violation)
ShortClassName (1 violation)
UnnecessaryImport (1 violation)
design (1 violation: 1)
AvoidCatchingGenericException (1 violation)
documentation (5 violations: 5)
CommentRequired (3 violations)
(6, 8) Bugs in com.htl.secure_programming
(7, 19) Bugs.def() in com.htl.secure_programming
(28, 19) Bugs.main() in com.htl.secure_programming
UncommentedEmptyMethodBody (2 violations)
errorprone (7 violations: 6 + 1)
AvoidLiteralsInIfCondition (2 violations)
CloseResource (1 violation)
EmptyIfStmt (1 violation)
EmptyTryBlock (1 violation)
EmptyWhileStmt (1 violation)
DataflowAnomalyAnalysis (1 violation)
aa8123912i312g3u1k3h13khj312h3jk13h32aaaaaaaaaa
secure_programming
GitHub (Material)
23:29 (916 字符, 47 行 换行符)
LF
UTF-8
4 个空格