Lab 3.1 Using Splint for C Static Analysis

安装splint软件

网址上给出的链接可以下载软件的源代码，但是在依照网址上步骤安装时发现我不能用成功的链接出可执行文件。于是决定自己去寻找资源。

```
Makefile:673: recipe for target 'splint' failed
make[3]: *** [splint] Error 1
make[3]: Leaving directory '/home/huangjiongrui/splint-3.1.2/src'
Makefile:1104: recipe for target 'Headers/flag_codes.gen' failed
make[2]: *** [Headers/flag_codes.gen] Error 2
make[2]: Leaving directory '/home/huangjiongrui/splint-3.1.2/src'
Makefile:175: recipe for target 'all-recursive' failed
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory '/home/huangjiongrui/splint-3.1.2'
Makefile:130: recipe for target 'all' failed
make: *** [all] Error 2
```

通过课程网址上splint的官网，我在 (http://splint.org/linux.html)上下载了linux上的二进制文件直接进行解压和安装。

直接解压之后，设置以下的环境变量：

```
export PATH=$PATH:~/splint-3.1.1/bin

export LCLIMPORTDIR=~/splint-3.1.1/imports

export LARCH_PATH=~/splint-3.1.1/lib
```

splint的使用

1. 写了一个缓冲区溢出漏洞的.c 文件，使用splint进行漏洞查询。以下为.c文件的源代码：

```
#include <stdio.h>
#include <string.h>
void func ( char* src){
    char buffer[12];
    strcpy(buffer, src);
}

int main(){
    char hello[100];
    gets(hello);
    func(hello);
    printf("hello world!");
}
```

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ splint vul1.c
Splint 3.1.1 --- 28 Apr 2003

vul1.c: (in function main)
vul1.c:10:2: Use of gets leads to a buffer overflow vulnerability.  Use fgets
                instead: gets
  Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
  inhibit warning)
vul1.c:10:2: Return value (type char *) ignored: gets(hello)
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalother to inhibit warning)
vul1.c:13:2: Path with no return in function declared to return int
  There is a path through a function declared to return a value on which there
  is no return statement. This means the execution may fall through without
  returning a meaningful result to the caller. (Use -noret to inhibit warning)
vul1.c:3:6: Function exported but not used outside vul1: func
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   vul1.c:6:1: Definition of func

Finished checking --- 4 code warnings
```

观察到除了我故意写的func函数被splint发现存在着缓冲区溢出的漏洞，splint还发现了gets函数有缓冲区溢出的漏洞。除此之外，还找出了两个warning，分别是返回值的缺失和函数内动态变量定义后未在函数外使用。这些都是编程中经常会犯的错误。这个检查程序很有用。

2. 写了一个含有格式化字符串漏洞的.c 文件，使用splint进行漏洞查询。以下为.c文件的源代码：

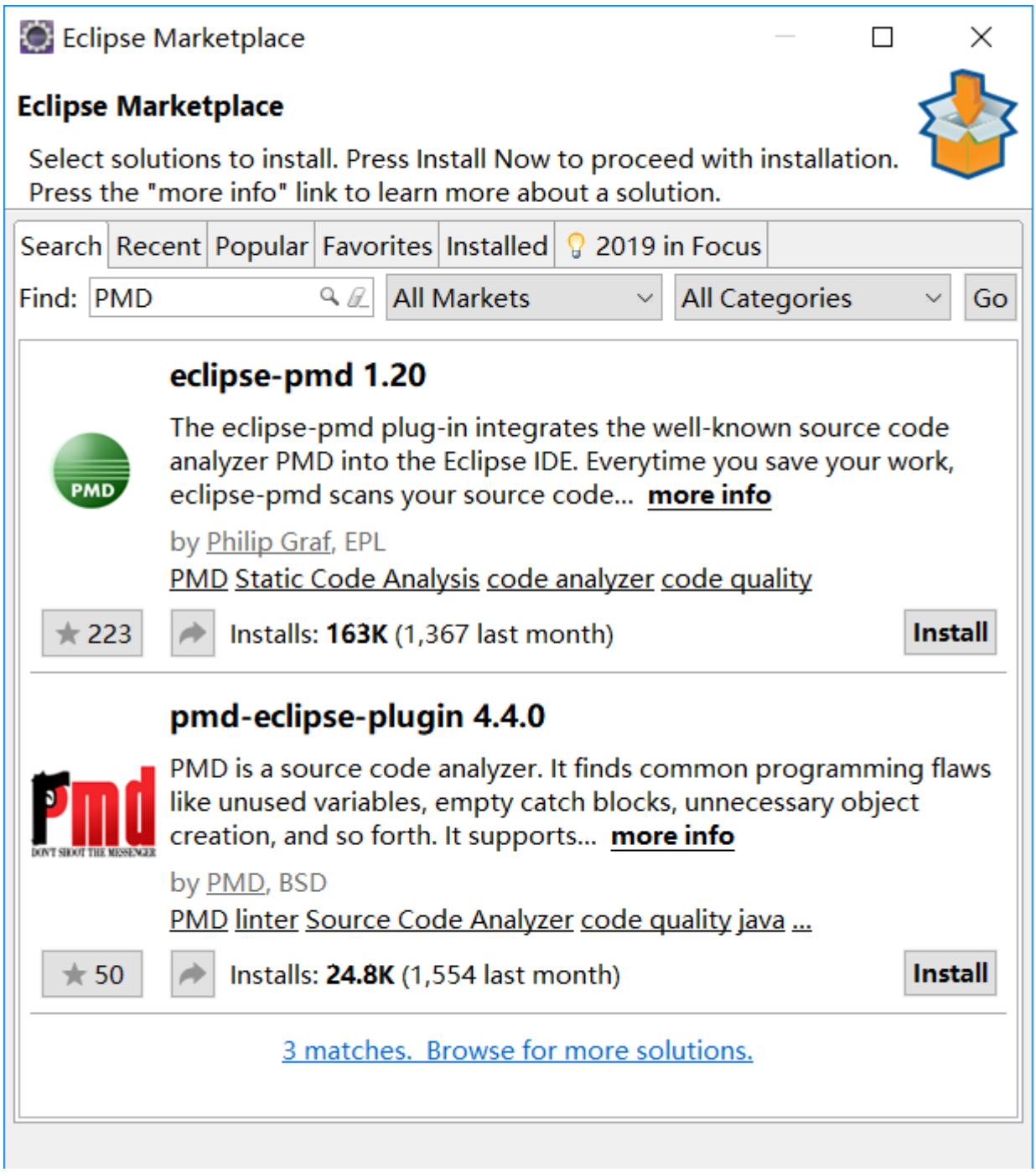
```c
#include <stdio.h>
#include <string.h>
int main(){
    char in[100];
    gets(in);
    printf(in);
}
```

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ splint vul2.c
Splint 3.1.1 --- 28 Apr 2003

vul2.c: (in function main)
vul2.c:5:2: Use of gets leads to a buffer overflow vulnerability.  Use fgets
               instead: gets
  Use of function that may lead to buffer overflow. (Use -bufferoverflowhigh to
  inhibit warning)
vul2.c:5:2: Return value (type char *) ignored: gets(in)
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalother to inhibit warning)
vul2.c:6:2: Format string parameter to printf is not a compile-time constant:
               in
  Format parameter is not known at compile-time.  This can lead to security
  vulnerabilities because the arguments cannot be type checked. (Use
  -formatconst to inhibit warning)
vul2.c:7:2: Path with no return in function declared to return int
  There is a path through a function declared to return a value on which there
  is no return statement. This means the execution may fall through without
  returning a meaningful result to the caller. (Use -noret to inhibit warning)

Finished checking --- 4 code warnings
```

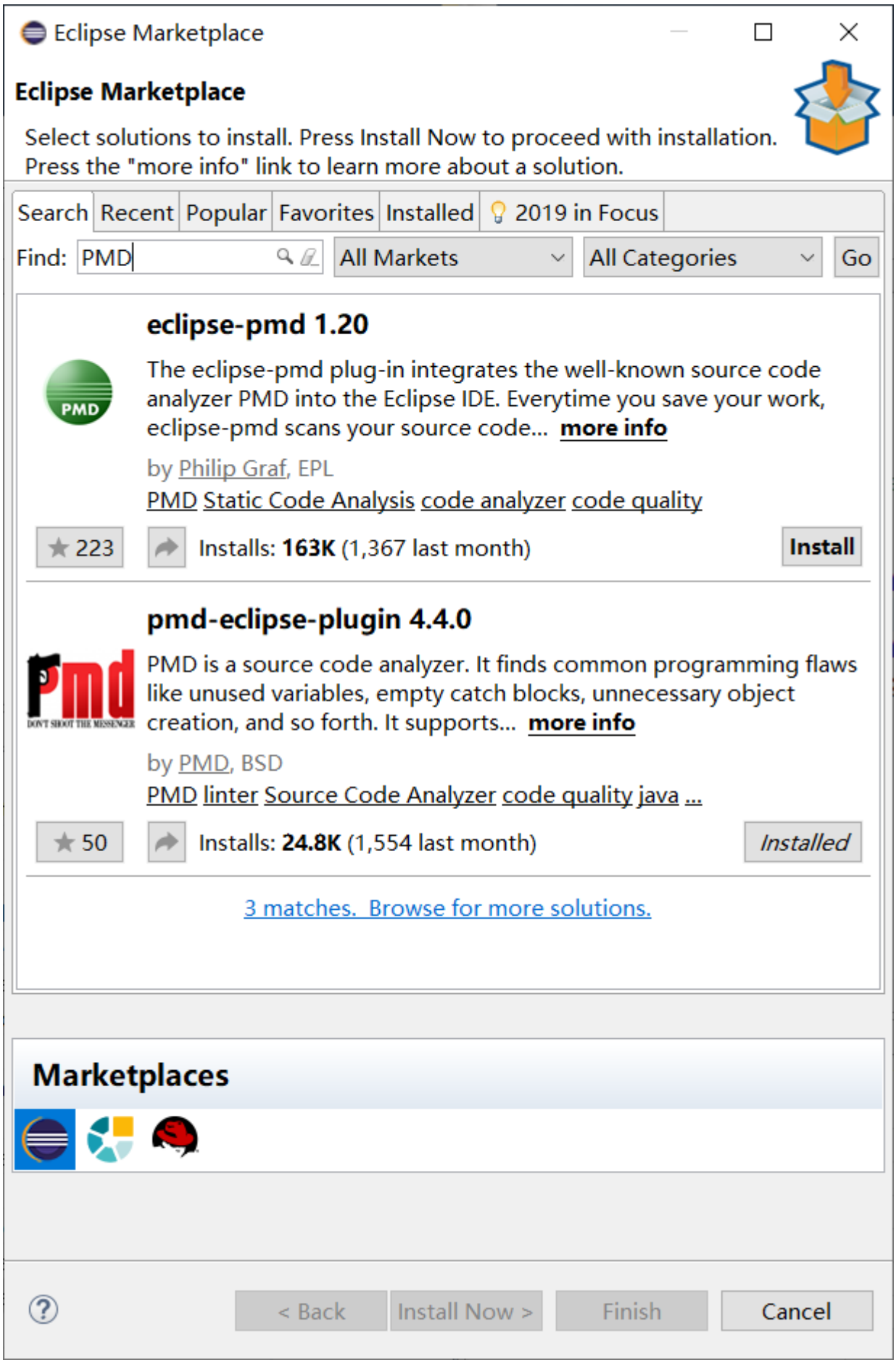
可以看到splint发现了在源文件中第6行的格式化字符串错误。除此之外，它也像之前那个源文件一样找到了gets的缓冲区溢出漏洞以及返回值的缺失等问题。

Lab 3.2 Using Eclipse for Java Static Analysis

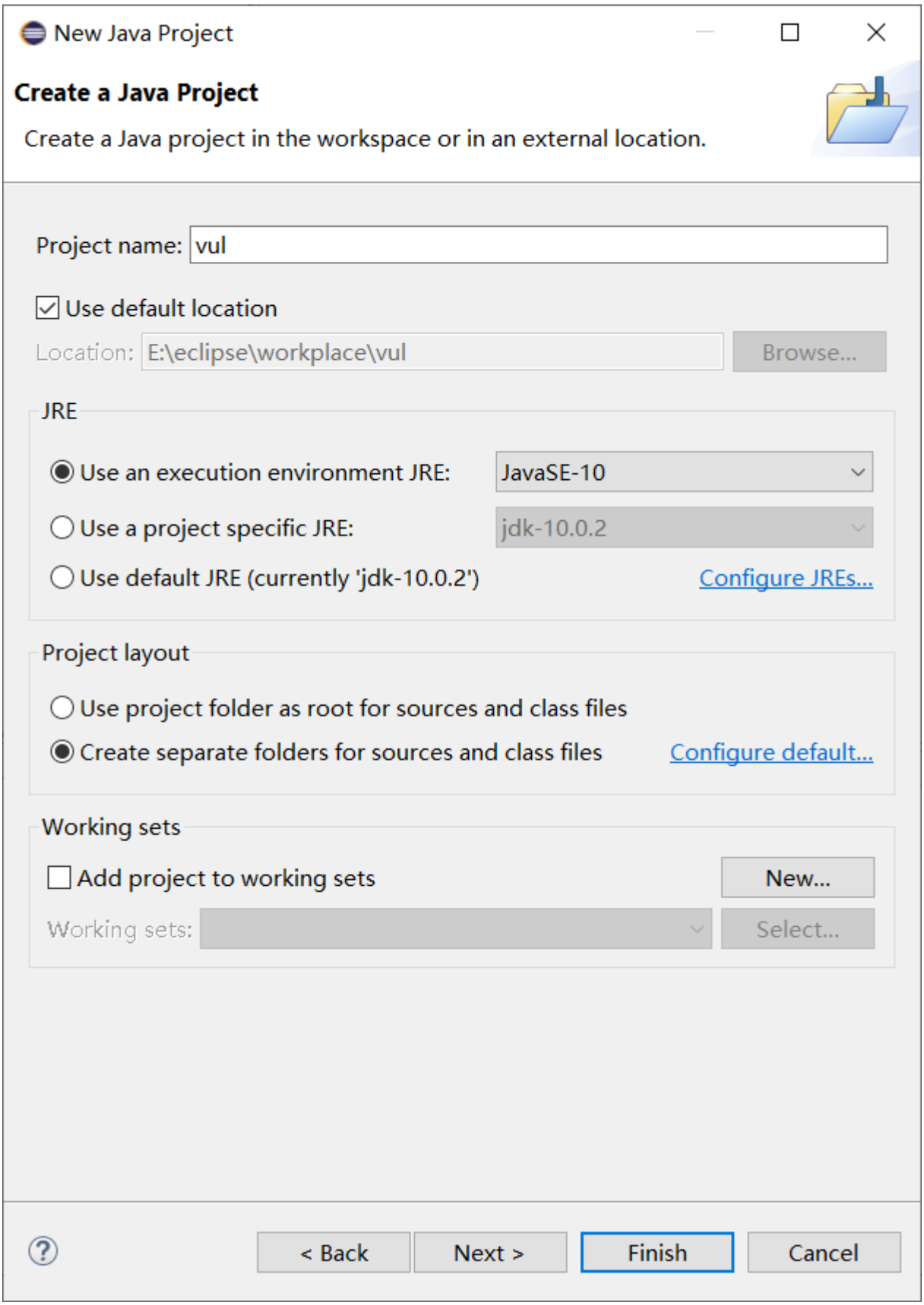
1. 因为感觉课程网站上的方法很麻烦，所以直接在eclipse market上寻找PMD并安装。



安装完成后，再次搜索PMD就显示已安装：



2. 创建一个Java project，并且编写一段含有问题的代码。

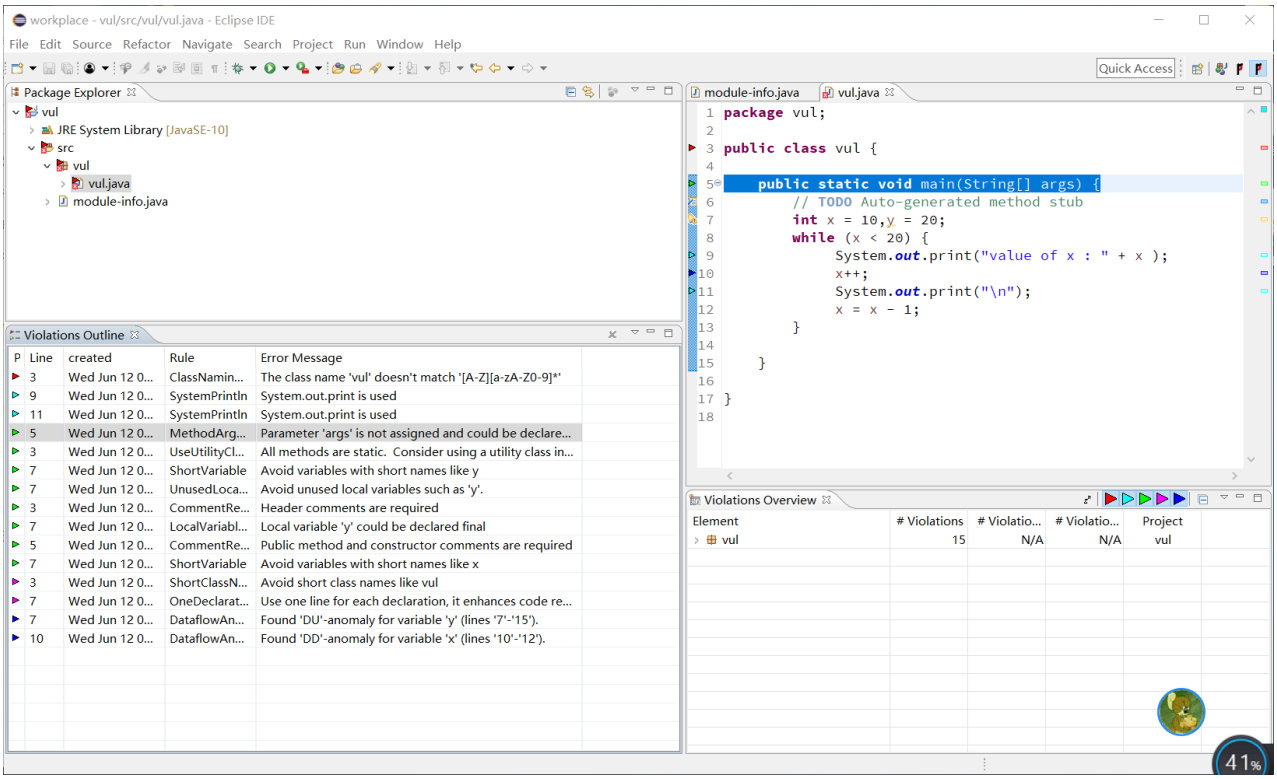


代码如下：

```
package vul;
```

```java
public class vul {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int x = 10,y = 20;
        while (x < 20) {
            System.out.print("value of x : " + x );
            x++;
            System.out.print("\n");
            x = x - 1;
        }
    }
}
```

3. 之后右键代码，使用PMD进行静态分析：



可以看到，PMD分析出了我代码中的问题。class name 中的vul没有大写，以及定义了没有用的变量y，甚至还贴心的建议程序员避免使用太短的变量名x和类名。但是这个软件也不是万能的，比如我代码中明显的死循环插件就没有检查到。