Feesual CPT Tool

Final Presentation

TEAM 8
박성근 (201211347)
임제현 (201211376)
김태홍 (201411270)

Table of Contents

PMD - Static Analysis Comment

FeedbackController.java

P	Line	created	Rule	Error Message
▶	34	Wed May 31 19:30:10 KST 2017	CollapsibleIfStatements	These nested if statements could be combined
▶	42	Wed May 31 19:30:10 KST 2017	CollapsibleIfStatements	These nested if statements could be combined
▶	29	Wed May 31 19:30:10 KST 2017	CollapsibleIfStatements	These nested if statements could be combined
▶	55	Wed May 31 19:30:10 KST 2017	CollapsibleIfStatements	These nested if statements could be combined
▶	29	Wed May 31 19:30:10 KST 2017	SimplifyBooleanExpressions	Avoid unnecessary comparisons in boolean expressions
▶	34	Wed May 31 19:30:10 KST 2017	SimplifyBooleanExpressions	Avoid unnecessary comparisons in boolean expressions

*CollapseibleIfStatements(합칠 수 있는 if문 지적)

➔ 모두 이런 식으로 되어있는 부분을 지적하였다.

```
if(table.getTable()[i][4]!=null){
    if(table.getTable()[i][4].equals(property)){
        f=1;
    }
}
```

&&를 사용하여 if문을 줄일 수 있음을 확인하였다.

PMD - Static Analysis Comment

*SimplifyBooleanExpressions(축약 가능한 Boolean형 지적)

➔ 개발자의 습관을 확인.

```
if(table.getTable()[i][6]!=null){
    if(table.getTable()[i][6].equals("single")==false){
        return feedbackList[2];
    }
}
```

PMD - Static Analysis Comment

Table.java

P	Line	created	Rule	Error Message
▶	46	Wed May 31 18:53:48 KST 2017	SimplifyBooleanExpressions	Avoid unnecessary comparisons in boolean expressions
▶	50	Wed May 31 18:53:48 KST 2017	SimplifyBooleanExpressions	Avoid unnecessary comparisons in boolean expressions
▶	1	Wed May 31 18:53:48 KST 2017	UnusedImports	Avoid unused imports such as 'javax.swing.JTable'

*UnusedImports(사용하지 않는 import로 지적)

➔ 실제로 Table.java에서 JTable을 사용하지 않았기에 삭제할 수 있음을 확인하였다.

*SimplifyBooleanExpressions(축약 가능한 Boolean형 지적)

➔ 개발자의 습관을 확인.

PMD - Static Analysis Comment

TestCaseController.java

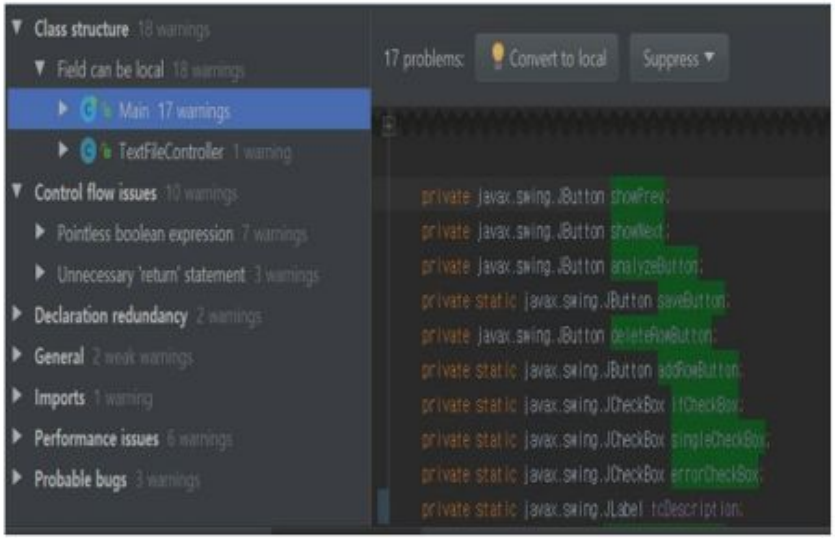
P	Line	created	Rule	Error Message
▶	3	Wed May 31 18:53:48 KST 2017	UnusedImports	Avoid unused imports such as 'java.util.ArrayList'

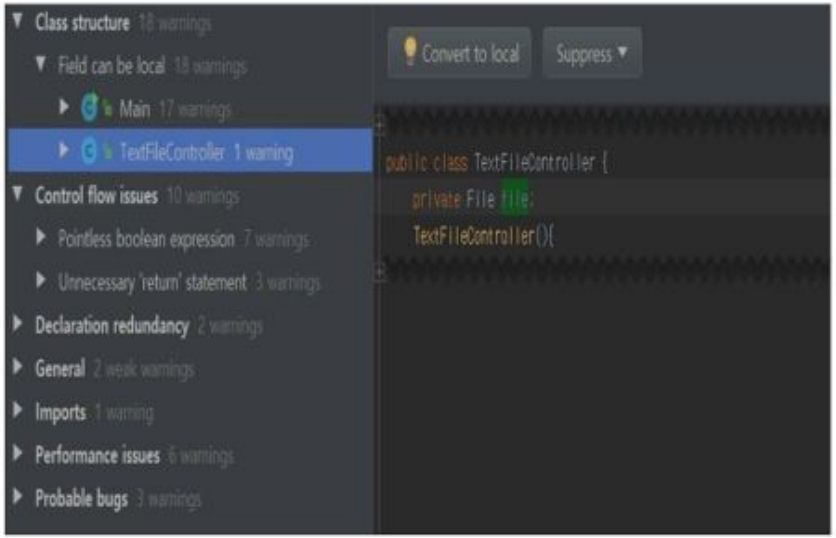
*UnusedImports(사용하지 않는 import로 지적)

➔ TestCaseController.java에서 사용하므로 import해야 한다.

```
public class TestCaseController {
   private ArrayList<TestCase> tcList;
   private ArrayList<TestCase> propertySub;
   private ArrayList<TestCase> nonPropertySub;
   private ArrayList<TestCase> propertyTc;
   private ArrayList<TestCase> nonPropertyTc;
```

IntelliJ - Static Analysis Comment

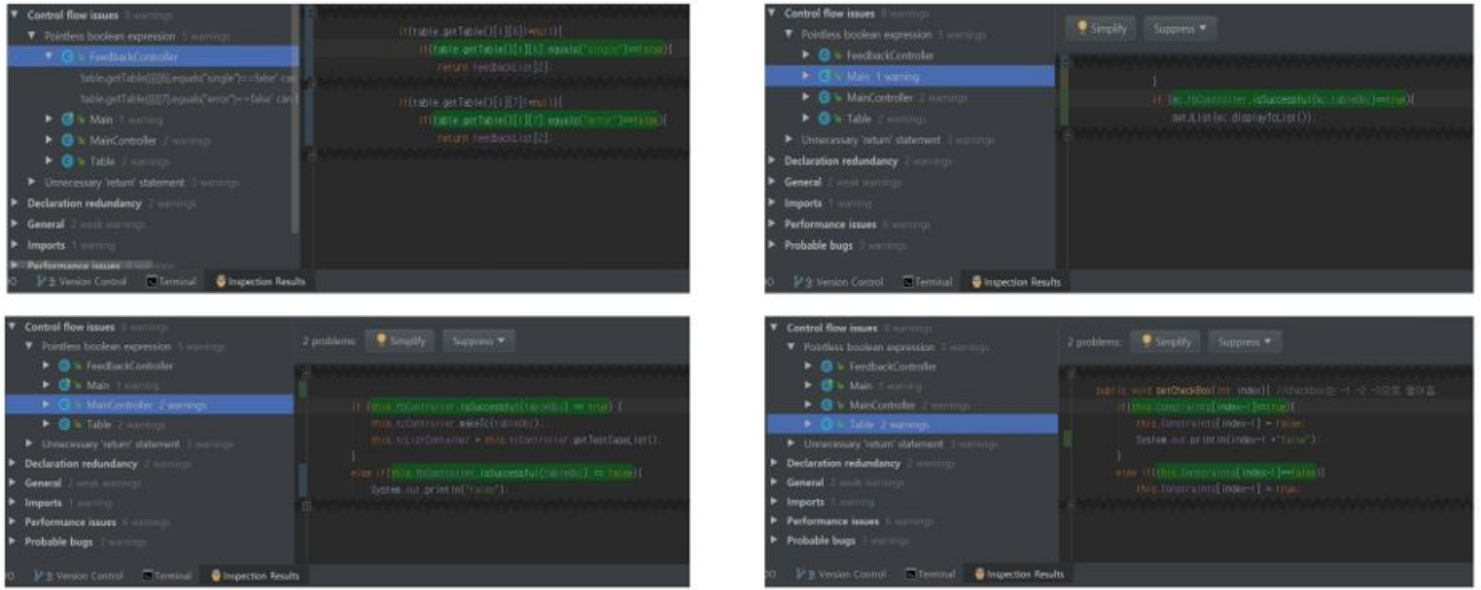




변수가 하나의 메소드에서만 사용 (지역변수 추천)

➔ 지적받은 부분은 수정 가능한 부분임을 확인하였다.

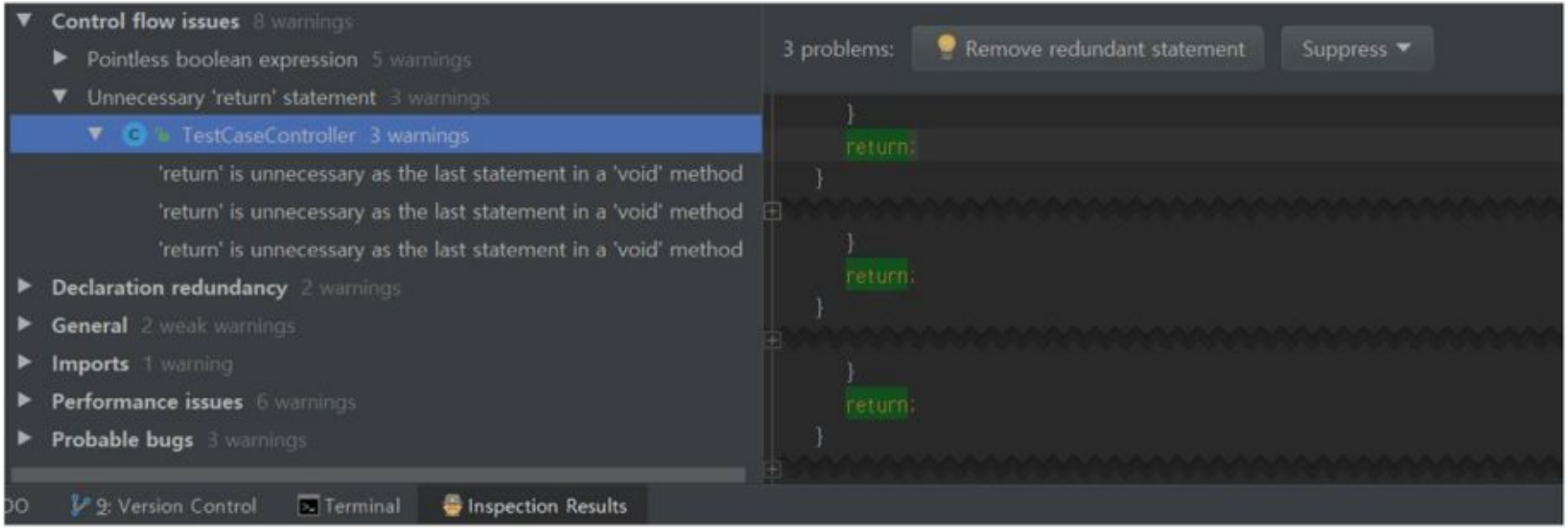
IntelliJ - Static Analysis Comment



‘ == true / false ‘ 사용 (단순히 expression 추천)

➔ 개발자의 습관임을 확인하였다.

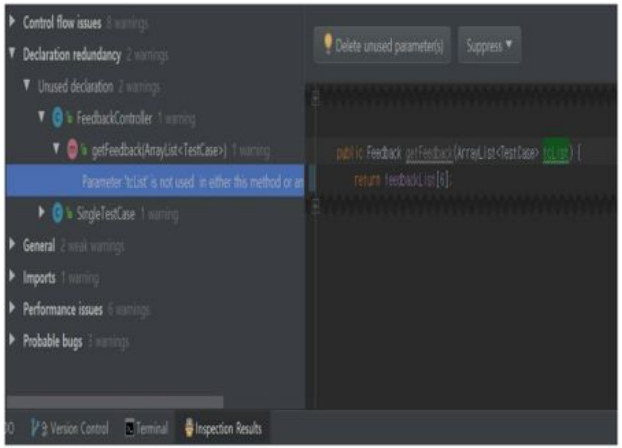
IntelliJ - Static Analysis Comment

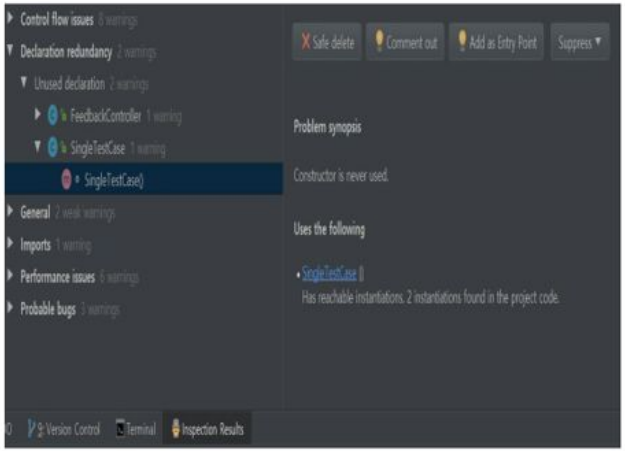


불필요한 return문 사용

➔ 끝을 명시적으로 표현하고 싶어서 사용했다. 고칠 필요는 없다고 생각하였다.

IntelliJ - Static Analysis Comment



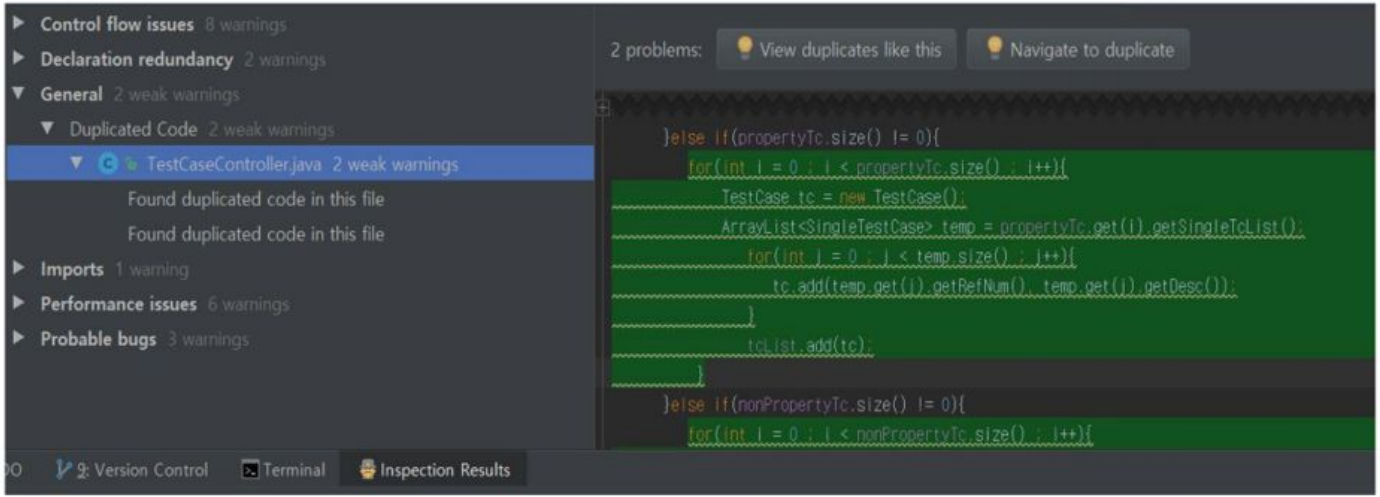


메소드 / 생성자 사용 x

➔ 왼쪽의 경우는 개발 중 확인 용도로 생성된 메소드이며 사용하지 않기 때문에 삭제가 필요함을 확인하였다.

오른쪽의 경우는 개발자의 습관으로 인한 발생한 생성자이며 사용하지 않기 때문에 삭제가 필요함을 확인하였다.

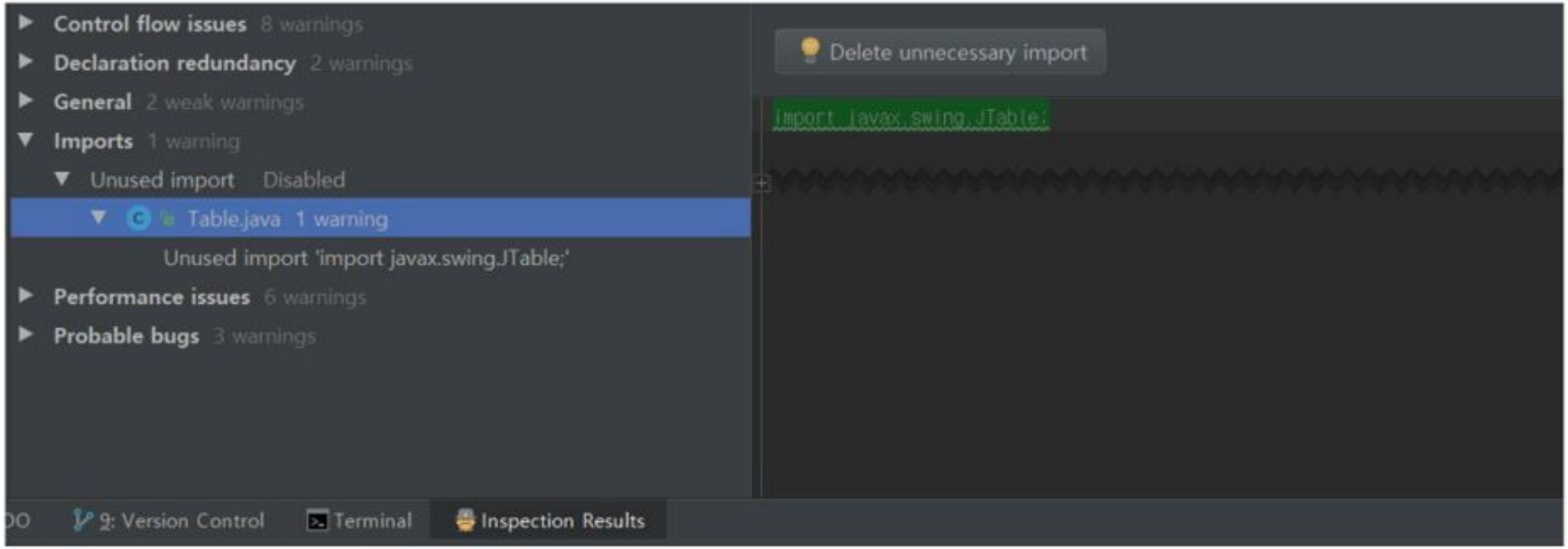
IntelliJ - Static Analysis Comment



동일한 코드 중복

➔ 코드가 중복된다고 했는데 nonPropertyTc을 사용하냐 propertyTc를 사용하냐 이것만 차이나기 때문에 그렇다고 생각하였다. 코드가 중첩되어도 사용하는 ArrayList가 다르므로 분리해줘야한다고 생각해서 수정하지 않았다.

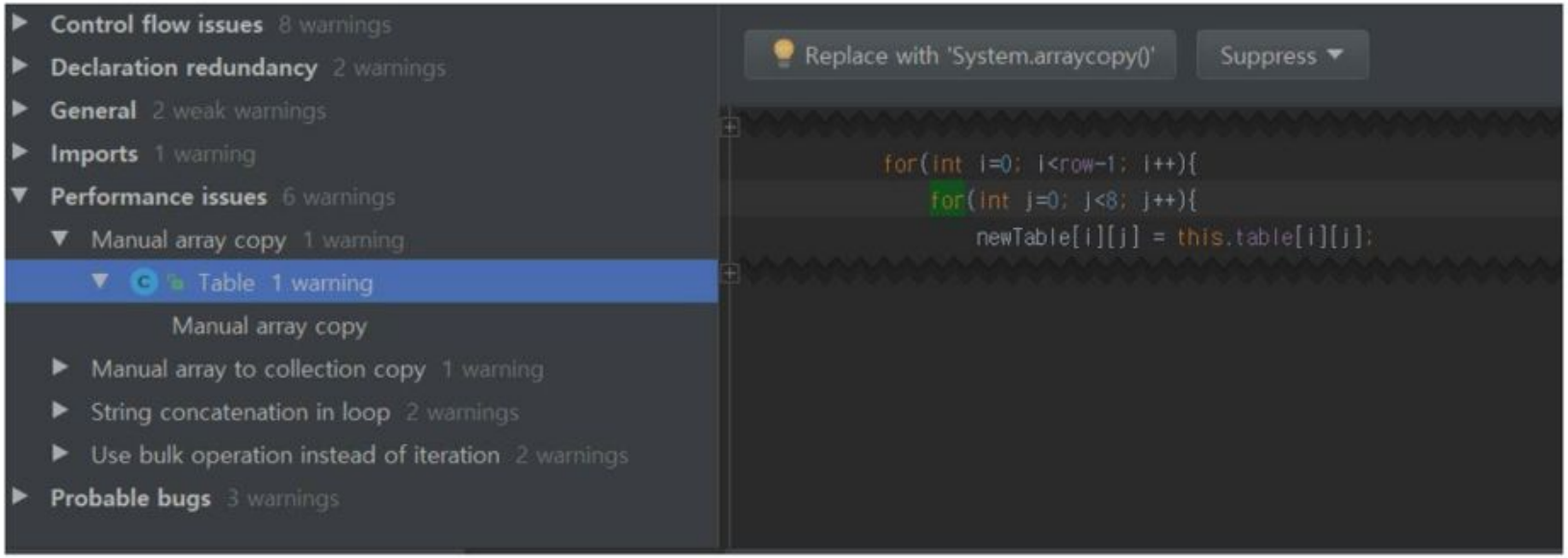
IntelliJ - Static Analysis Comment



Import한 내용 사용 x

➔ 실제로 사용하지 않기 때문에 삭제가 필요함을 확인하였다.

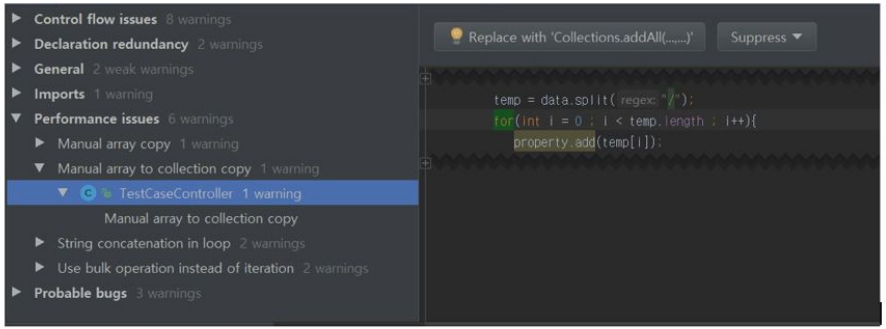
IntelliJ - Static Analysis Comment



단순 array 복사 (arraycopy 추천)

➔ arraycopy의 존재를 알려주었다.

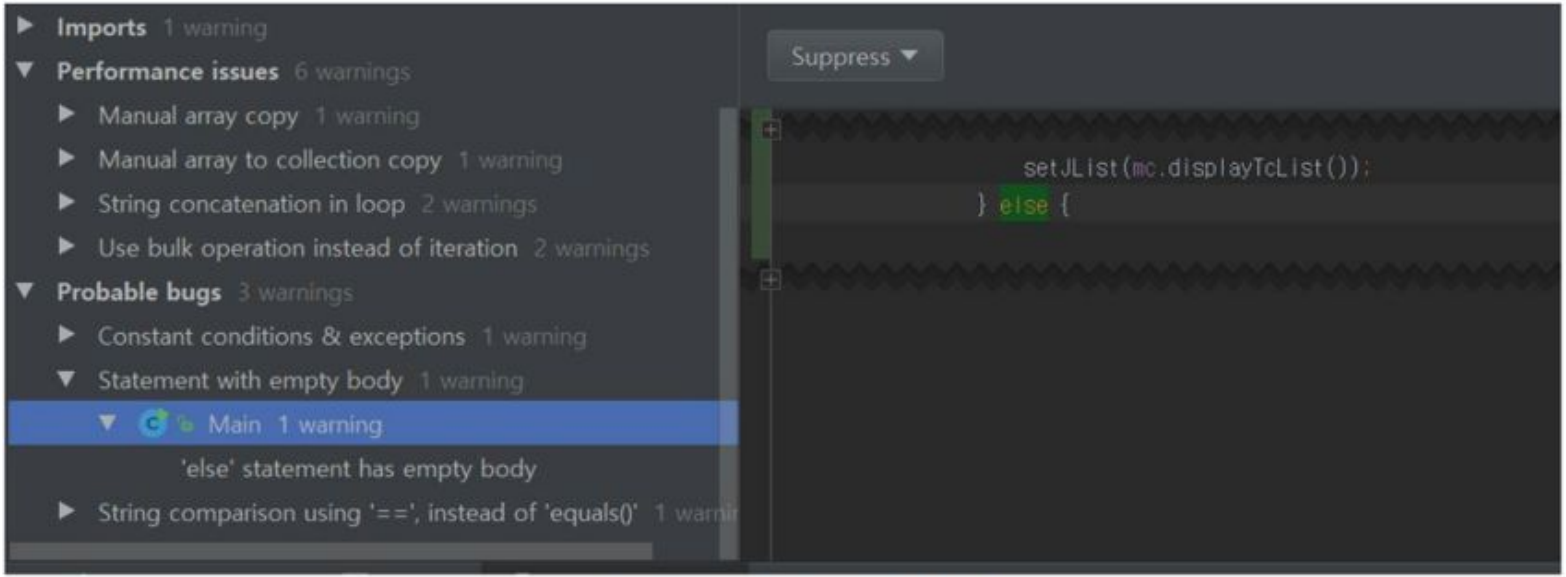
IntelliJ - Static Analysis Comment




```
public void saveProperty(ArrayList<String> property, String data){
   if(data.contains("/")){
      String[] temp;
      temp = data.split("/");
      for(int i = 0 ; i < temp.length ; i++){
         property.add(temp[i]);
      }
   }else{
      property.add(data);
   }
   return;
}
```

➔ 배열을 통째로 array에 넣는데는 collection.addall 메소드를 추천하는데 굳이 바꿔야할 필요성을 느끼지 못하였다.

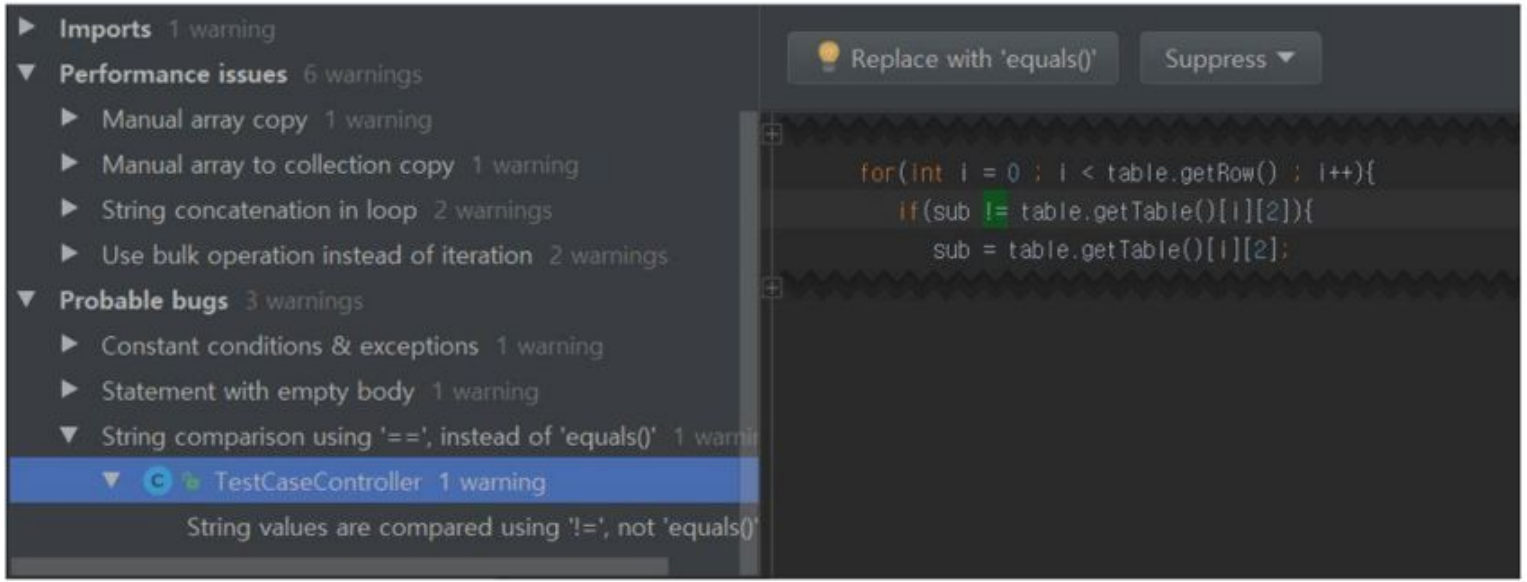
IntelliJ - Static Analysis Comment



else문 body가 비어있음

➔ 실제 코드에서 삭제가 필요함을 확인하였다.

IntelliJ - Static Analysis Comment



Stirng 비교문에 != / == 사용 (equals()함수 사용)

➔ 수정이 필요함을 확인하였다.

Level 1 - Static Analysis Comment

개발자의 코딩 습관을 확인할 수 있었다.

치명적으로 작용할 수 있는 에러 또한 확인할 수 있었다.

그리고 코드가 좀 더 군더더기 없이 간결하게 발전할 수 있음을 확인하였다.

EMP - Static Analysis Comment

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per method)		3.259	6.435	40	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
src		3.259	6.435	40	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
(default package)		3.259	6.435	40	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
TestCaseController.java		10.571	13.468	40	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
TestCaseController		10.571	13.468	40	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
makeTc	40					
recursive2	20					
checkProperty	5					
recursive1	4					
saveProperty	3					
TestCaseController	1					
getTestCaseList	1					
FeedbackController.java		6.25	8.526	21	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
FeedbackController		6.25	8.526	21	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
getFeedback	21					
isSuccessful	2					
FeedbackController	1					
getFeedback	1					
TextFileController.java		7	6	13	/SMA2017_FeesualCPT/src/TextFileController.java	saveRequest
TextFileController		7	6	13	/SMA2017_FeesualCPT/src/TextFileController.java	saveRequest
saveRequest	13					
TextFileController	1					
MainController.java		1.7	1.269	5	/SMA2017_FeesualCPT/src/MainController.java	displayTcList
Main.java		2.167	1.344	4	/SMA2017_FeesualCPT/src/Main.java	getJTableData

TestCaseController.makeTc 메소드
TestCaseController.recursive2 메소드
FeedbackController.getFeedback 메소드
TextFileController.saveRequest 메소드

Cyclomatic Complexity가 높게 나옴

➔ TestCase를 생성함에 있어서 table을 탐색하고 모든 Testcase를 돌기때문에 다중for문과 재귀함수가 필요하다고 생각했다. 또 계산하는데 조건들이 많이 필요해서 조건문도 많이 사용하였다. 그러다 보니 Complexity가 높았다고 생각하였다.

EMP - Static Analysis Comment

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Nested Block Depth (avg/max per method)		1.907	1.531	7	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
src		1.907	1.531	7	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
(default package)		1.907	1.531	7	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
FeedbackController.java		2.75	2.487	7	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
FeedbackController		2.75	2.487	7	/SMA2017_FeesualCPT/src/FeedbackController.java	getFeedback
getFeedback	7					
isSuccessful	2					
FeedbackController	1					
getFeedback	1					
TestCaseController.java		3.571	2.129	7	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
TestCaseController		3.571	2.129	7	/SMA2017_FeesualCPT/src/TestCaseController.java	makeTc
makeTc	7					
recursive2	6					
checkProperty	4					
recursive1	3					
saveProperty	3					
TestCaseController	1					
getTestCaseList	1					
TextFileController.java		3	2	5	/SMA2017_FeesualCPT/src/TextFileController.java	saveRequest
MainController.java		1.5	0.922	4	/SMA2017_FeesualCPT/src/MainController.java	displayTcList

FeedbackController.getFeedback 메소드
TestCaseController.makeTc 메소드
TestCaseController.recursive2 메소드

중첩된 Block이 많음

➔ 여러 조건에 따라 수행되는 결과가 달라야 했기 때문에, if문 안에 if문, 또는 for문이 들어가는 경우가 많았다. 이 때문에 중첩된 Block이 많다고 생각하였다.

EMP - Static Analysis Comment

- 작성된 코드의 Cyclomatic Complexity와 {}의 중첩도를 확인할 수 있었고, 우리 팀이 작성한 코드가 얼마나 복잡한지 확인할 수 있었다. 하지만 최선을 다해 짜낸 알고리즘이었기 때문에 현재 코드 이상으로 개선할 수는 없다고 생각하였다.

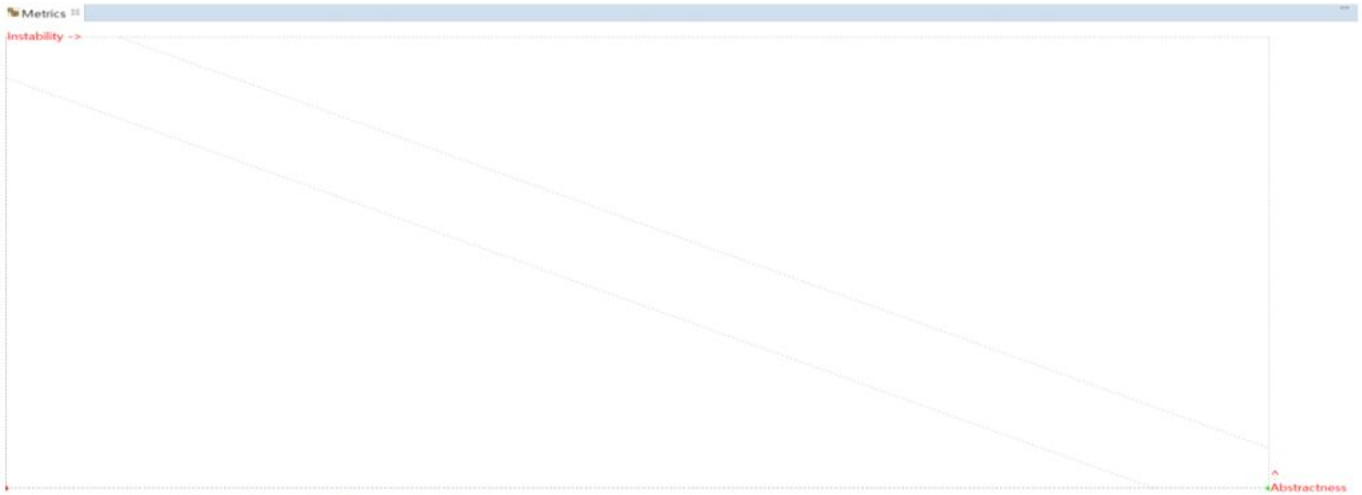
JDpend - Static Analysis Comment

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
Default	25	0	0	1	0.00	1.00	0.00	
org.junit	0	0	1	0	0.00	0.00	1.00	

JDepend

```
---------------------------
- Package: Default
---------------------------

Stats:
    Total Classes: 25
    Concrete Classes: 25
    Abstract Classes: 0

    Ca: 0
    Ce: 1

    A: 0
    I: 1
    D: 0

Depends Upon:
    org.junit

Used By:
    Not used by any packages.
```

JDpend - Static Analysis Comment



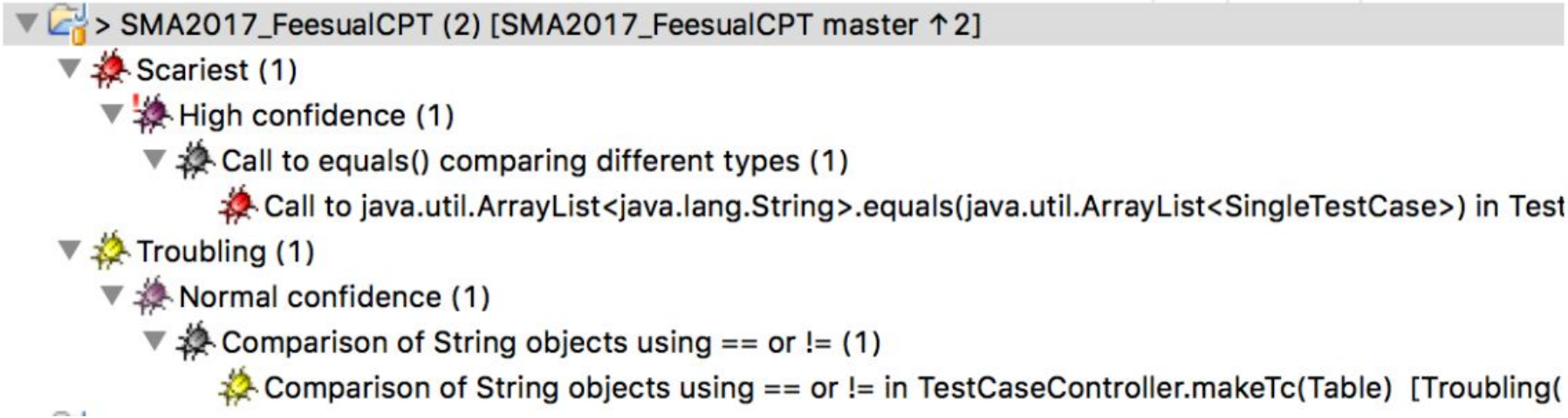
안정적이면서 추상화가 덜 되었음.

➔ 패키지가 1개 밖에 없어서 안정적이며 추상화가 덜 되었다고 평가받은 것 같다. OOPT 2040을 수행할 때, Feesual CPT Tool은 1개의 패키지로 충분하다고 생각했기 때문에 당연한 결과이다.

FindBugs - Static Analysis Comment

8조 분석결과

- FindBug 를 사용하여 분석한 결과, 총 2 개의 warning 이 발견 되었다.



FindBugs - Static Analysis Comment

T8. Feesual CPT

1. String 파라미터에 == , != 연산자 사용

2. 서로 다른 성격의 자료형간 비교

➔ 1번의 경우는 equals로 수정해야함을 확인하였다.

2번의 경우는 유닛테스트를 수행하면서, 고의적으로 에러 상황을 발생시키고 확인하려는 코드였기 때문에 저러한 지적이 나온것 같다. 완성된 코드를 보낼 때, 유닛 테스트 코드는 제외하고 보내야함을 확인하였다.

FindBugs - Static Analysis Comment

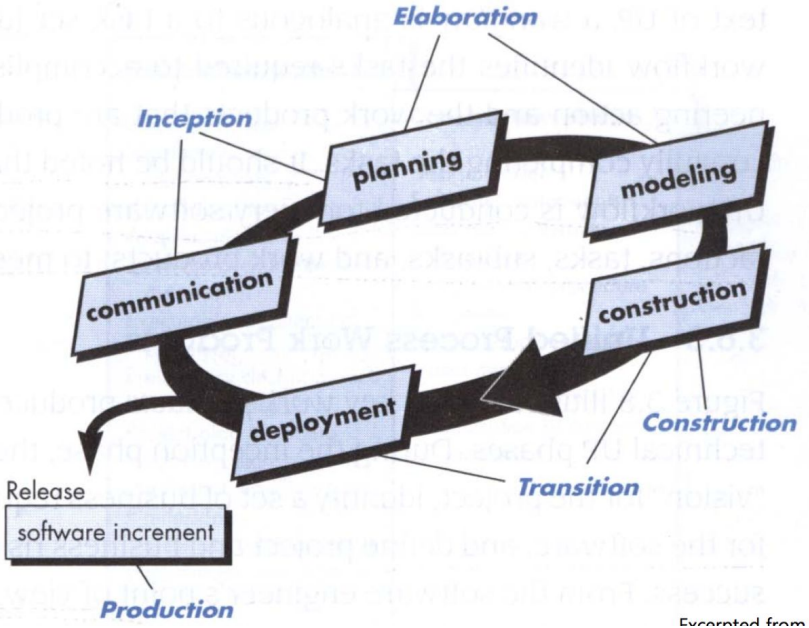
실제로 코드가 돌아가면서 발생할 수 있는 치명적인 상황을
지적해주었다.

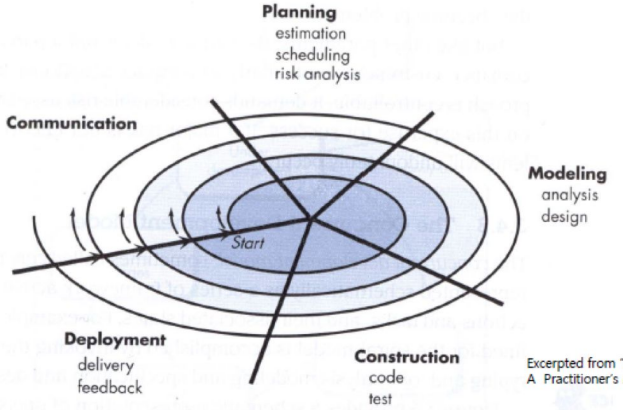
유닛 테스트코드에서 고의적으로 발생시킨 에러 코드를 제외하면,
상당히 적은 수의 지적이 있었다고 생각한다.

좀 더 완벽한 코드를 작성하도록 꼼꼼히 확인해야한다.

느낀 점

느낀 점





Excerpted from "Software Engineering: A Practitioner's Approach" by Roger S. Pressman

느낀 점

장점

- 구현하기 전에 design하는 단계 등을 통해 기능에 대해 확실하게 생각해볼 수 있다.
- Class diagram과 Sequence Diagram등을 통해 프로그램의 흐름을 알고, 체계적으로 구현할 수 있다.
- 프로그램 구현 시에 큰 어려움 없이 구현할 수 있다.

단점

- 구현하기 전까지 많은 시간과 비용이 소모된다.
- 문서 작성에 많은 시간이 투자 되었다.

Demonstration Video