2016 소프트웨어 모델링&분석

그 놈! Clone Checker

Team4

Contents

System Test Report - stage 1000_ver6

#	1.
문서 분석 결과	3페이지, (전략) 가중치를 두어 프로그램을 실행하면 유사도를 비율(%)로 사용자에게 알려주는 것을 목표로 한다 : STR에서는 백분율을 점수로 수정하였다고 하였으나, 실제로는 이전 버전(Stage1000_T4_ver5)에서 변하지 않았음.
대응	최종 유사도는 백분율(%)로 계산하기로 이미 그 전에 구현을 해 놓았었고 실제로 이전의 코드를 실행시켰을 때, Show Detail 버튼을 열어보면 최종 유사도가 백분율(%)로 나옴을 알 수 있다. 실제로 T4_5_System_Test.pdf의 아래쪽 Pairwise 부분에 첨부되어 있는 캡쳐를 보면 백분율이 쓰여 있음이 보인다. 따라서 수정하지 않음.

System Test Report - stage 1000_ver6

#	2.
문서 분석 결과	4페이지, 이전 버전과 비교하였을 때 내용이 변경된 점이 보임, 수정 내역이 없어 쉽게 알기 어려움.
대응	수정 내역을 Stage 1000이 아닌 2030에서 적는 것이라 생각하여 수정하지 않음.

#	3.
문서 분석 결과	4페이지, 실제 프로그램 사용 시 7개 파일 분석에 4분 소요.
대응	50개 파일 분석에 5분 소요로 수정.

System Test Report - stage 1000_ver6

#	4.
문서 분석 결과	8페이지, 실행 환경에 IDE를 넣는 것은 적절하지 않아 보임.
대응	OS : Windows 7, 개발 언어 : JAVA 1.8.0_77 로 수정.

#	5.
문서 분석 결과	10페이지, 이전에 언급된 적 없는 Use-Case 인 Draw Result가 등장함.
대응	Draw Result(UseCase)를 삭제.

#	6.
문서 분석 결과	11페이지, 다이어그램에 Draw Result가 존재 하지 않음.
대응	다이어그램 수정.

System Test Report - stage 2030_ver4

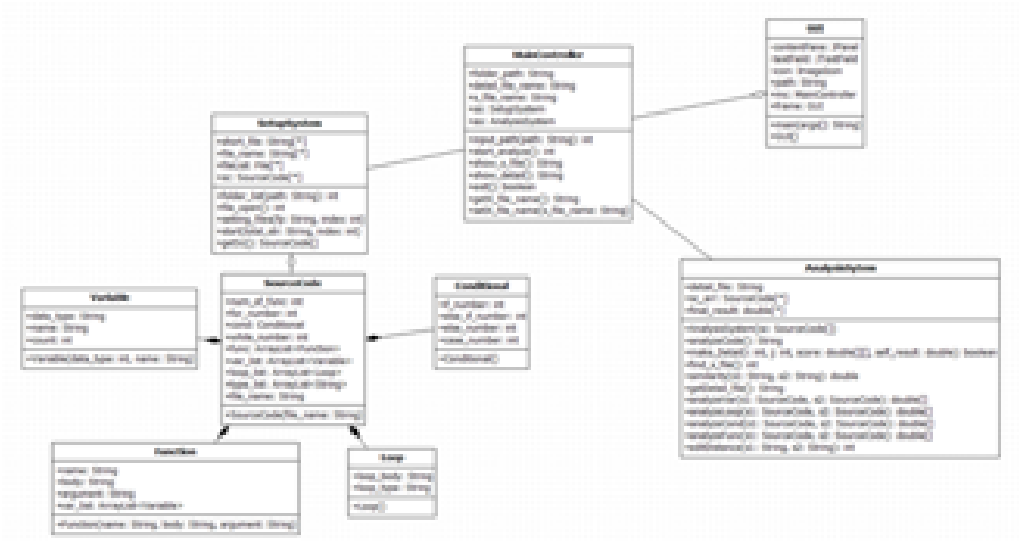
#	1.
문서 분석 결과	5페이지, Ref. # 표가 이전 버전 문서를 반영하고 있지 않음.
대응	반영하고 있으므로 수정하지 않음.

#	2.
문서 분석 결과	7페이지, (A)에서 정의한 Use-Case를 완전하게 표현하고 있지 않음.
대응	수정된 Use-Case를 바탕으로 수정함.

System Test Report - stage 2030_ver4

#	3.
문서 분석 결과	21,22페이지, Show Detail 항목의 Typical Courses of Events의 설명 중, 3번과 4번의 순서가 잘못 되어 있음. 실제로는 txt 파일을 닫는 것과 버튼이 비활성화 되는 것에 상관 관계가 없음.
대응	1. (A) Actor가 Show Detail 버튼을 누른다. 2. (S) Show Detail 버튼을 비활성화 시킨다. 3. (S) 유사도 검사의 세부 내용이 담긴 .txt 파일의 내용을 보여준다. 로 수정함.

System Test Report - stage 2030_ver4

#	4.
문서 분석 결과	23페이지.. A. MainController에 실제 코드 상에 존재하는 start_analyze():int가 존재하지 않음. B. AnalysisSystem의 make_Detail()의 반환형은 boolean이나, 이것이 적절히 반영되어 있지 않음.. C. GUI 클래스가 반영되어 있지 않음.
대응	클래스 다이어그램 수정. 

System Test Report - stage 2030_ver4

#	5.
문서 분석 결과	25페이지, Variable 항목이 UML과 일치하지 않음. A. count는 UML 상에서 attribute이나, 표에서는 operation으로 표현됨. B. Variable() 메서드가 표현되어 있지 않음.
대응	수정함.

#	6.
문서 분석 결과	26~28페이지, 내부 시스템 동작에 대해서는 작성하지 않았음.
대응	System Sequence Diagram은 System 자체를 블랙박스로 보고 시퀀스를 쓰는 것이라 생각하여 내부 시스템의 동작에 대해서는 쓰지 않기로 하였음.즉 수정하지 않음.

System Test Report - stage 2030_ver4

#	7.
문서 분석 결과	26페이지, Input Path 이후 Setting Files이 실행되어야 하나, 그것에 관한 언급이 없음.
대응	6번과 같은 이유.

#	8.
문서 분석 결과	CK.STC.321 Test Case와는 달리, 실제 프로그램은 창을 닫기 전에 버튼이 비활성화 됨.
대응	경로 입력을 성공하고 Start 버튼을 누르고, 분석 완료 알림창이 뜬 뒤에 Show Detail 버튼을 누르면 Show Detail 버튼이 비활성화 되는지 확인한다. 로 수정함.

System Test Report - stage 2030_ver4

#	9.
문서 분석 결과	Actor가 user인 것에 대해서만 use-case를 operation에 대응시켰으나, 이 과정에서 system이 actor인 use-case에 대해서는 operation의 이름을 어떻게 바꾸는지 명세하지 않았음.
대응	operation의 이름을 바꾼다는 것이 이해가 안되어서 수정 못함.

#	10.
문서 분석 결과	System이 actor인 use-case에 대해, operation name을 표기하지 아니하고 서술함.
대응	이해가 안되어서 수정 못함.

System Test Report - stage 2030_ver4

#	11.
문서 분석 결과	클래스 다이어그램보다 도메인 모델이 거대하고 상세하게 표현되어 있음.
대응	수정함.

System Test Report - stage 2040_ver3

#	1.
문서 분석 결과	3~11 페이지, A. Input Path는 실질적으로 경로 유효성을 판단하고, Setting Files을 호출하는 역할을 하나, 표에는 그러한 서술이 존재하지 않음. B. Setting Files : 함수의 지역변수 저장은 SourceCode 내부의 변수가 아닌, Function 내부의 변수에 저장하는 것으로 보이나, 표의 서술은 SourceCode 내부의 변수에 저장한다고 되어 있음. C. Start Analyze : OSP Stage 1000 ver6에서 Start Analyze Code로 변경된 Function임.

System Test Report - stage 2040_ver3

대응	A, B : 수정함. C : 2041은 Define Real Use Cases로 UseCase에 대한 내용을 적는 것이라 판단. 본 프로그램상 UseCase에는 Start Analyze, Functional Requirement에는 Start Analyze Code라 되어 있음 따라서 이름이 Start Analyze라 하는 것이 맞다고 판단하여 수정 안함.

#	2.
문서 분석 결과	16 ~ 19페이지, 전체적으로 실제 작성된 코드를 적절히 반영하고 있지 않음.
대응	반영하고 있다고 생각하여 수정하지 않았음.

System Test Report - stage 2040_ver3

#	3.
문서 분석 결과	18페이지, Start Analyze 그림에서 코드 상에는 조건문이 존재하나, 그림에선 표현되지 아니함.
대응	이해가 안되어서 수정 못함.

#	4.
문서 분석 결과	19페이지, Show Detail 그림에서, 메모장을 호출하여 txt 파일의 내용을 보여주는 부분이 명확하게 나와있지 않음.
대응	System함수를 이용하는 것이라서 따로 표현하지는 않았음.

System Test Report - stage 2050_ver2

#	1.
문서 분석 결과	3~5페이지, Cross Reference의 Start Analyze를 Start Analyze Code로 변경이 필요함.
대응	Start Analyze Code는 Functional Requirements의 이름이고, Start Analyze는 Usecase이름이 맞으므로 수정 안함.

#	3.
문서 분석 결과	19~20페이지, Pre-Conditions와 Post-Conditions가 반대로 적혀 있음.
대응	수정함.

System Test Report - stage 2050_ver2

#	2.
문서 분석 결과	6~19페이지, . A. Input, Output의 형태가 메서드로 나와 있지만, 실제 표의 내용은 변수 형임. B. analyzeCode : 배열의 사이즈를 선정한 이유가 명시되지 않음. C. SourceCode, Variable, Loop, Conditional, Function 항목의 메서드는 모두 동일한 기능(클래스 변수 초기화)을 수행하나, 실제 코드 상에서는 클래스 내부의 변수들의 접근 제어자가 모두 public, 또는 default이고, Getter, Setter 또한 존재하지 않음.
대응	A, B : 수정함. C : 변수 초기화를 하는 점은 맞지만, 코드 상에서 public으로 접근하기에 편하다 생각하여 제어자를 private으로 지정하지 않았음.

System Test Report - stage 2050_ver2

#	4.
문서 분석 결과	20~24페이지, 절대 경로를 사용하여, 특정 컴퓨터에서만 테스트 할 수 있도록 해두었고, 따라서 리눅스 기반의 CTIP 환경에서는 에러를 발생시킨다. 상대 경로를 이용하는 코드로 변경하여, CTIP 환경에서 테스트 가능하도록 하는 것이 적절하다 판단함.
대응	수정하지 않음.

#	5.
문서 분석 결과	29페이지, 바디 모음을 비교한다고 서술하였으나, 구체적인 비교 방법이 서술되어 있지 않음.
대응	소스 코드의 전체 함수 body를 하나로 모으고 모든 변수의 이름을 "attr"로, 모든 함수의 이름을 "oper"로 바꾼 상태에서 두 함수 body 모음을 Levenshtein distance 알고리즘을 통하여 비교한다. 유사도에 따라서 점수를 부여한다. 라고 수정함.

System Test Report - stage 2060_ver2

문서 분석 결과 문제점은 없었으나, 2050을 수정하면서 내용을 수정함

총 33개의 부분에서 문서를 수정함

System Test Report

– Category Partition Testing

#	1
Testing 결과	Single 법칙에 의해 생성된 테스트 케이스는 모두 통과 하였으나, Struct 또는 typedef에 의해서 생성된 변수 테스트에서는 정확하지 않은 결과가 발생하였다. ■ 이는, 변수에 대한 명세만 있고, 구조체에 대해서는 어떻게 검사한다는 specification이 존재하지 않았기 때문.

대응	구조체 또한 하나의 자료형으로 Parsing을 진행하는데, 변수 검사와 Specification이 동일하므로 따로 적을 내용이 없다고 판단하여 따로 적지 않음. 변수 테스트에서 정확하지 않은 결과가 발생한 부분에 대해서는 코드를 수정함으로써 옳게 고침

System Test Report

– Category Partition Testing

[기존 코드]

```
if(exist_flag==0){
    sc[index].type_list.add(temp_str);
    sc[index].type_list.add(temp_str+"*");
    sc[index].type_list.add(temp_str+"**");
    sc[index].type_list.add(temp_str+"[]");
    sc[index].type_list.add(temp_str+"[][]");
}
```

[수정 코드]

```
if(exist_flag==0){
    temp_str = temp_str.trim();
    sc[index].type_list.add(temp_str);
    sc[index].type_list.add(temp_str+"*");
    sc[index].type_list.add(temp_str+"**");
    sc[index].type_list.add(temp_str+"[]");
    sc[index].type_list.add(temp_str+"[][]");
}
```

System Test Report

– Pairwise Testing (Testing 결과)

테스트에 앞서

본 프로그램은 변수, 함수, 조건문, 반복문에 대한 개수 확인을 제대로 진행하지 못하므로, 이를 고려하여 이번 Testing에서는 유사도 검사에서 정확한 동작을 수행하는지에 대한 확인을 목적으로 하였다.

Pairwise testing 테스트 중 변수나 함수, 조건문 등의 컴포넌트를 정확히 카운트 하지 못하는 현상을 발견하여, 대안으로 유사도 검사를 정확히 수행하는지에 대한 case를 만들었다. 그러나 모든 case에 대해 유사도 검사를 정확하게 실행하는 case가 존재하지 않았다.

위 결과 보고서에서, 함수 내용 유사도 항목이 존재하지 않는 것을 확인할 수 있다.

System Test Report

– Pairwise Testing (대응)

1. 변수 개수 확인 : Category-partitioning Testing 에서 수정한 코드를 바탕으로 총 개수를 정확히 셀 수 있게 수정

System Test Report

– Pairwise Testing

```
//지역변수에서 동일한 변수의 개수 덧셈
for(Function f1: s1.func){
    for(Function f2: s1.func){
        for(Variable v1 : f1.var_list){
            for(Variable v2 : f2.var_list){
                if(v1.name.equals(v2.name) && v1.data_type.equals(v2.data_type) && f1.name.equals(f2.name)){
                    //System.out.println("지역변수"+f1.name+":"+v1.data_type+" "+v1.name+"==="+f2.name+":"+v2.data_type+" "+v2.name);
                    sum++;
                }
            }
        }
    }
}
```

[2 – 기존 코드]

System Test Report

– Pairwise Testing

```
//지역변수에서 농일한 변수의 개수 덧셈
for(Function f1: s1.func){
    for(Function f2: s2.func){
        for(Variable v1 : f1.var_list){
            for(Variable v2 : f2.var_list){
                if(v1.name.equals(v2.name) && v1.data_type.equals(v2.data_type) && f1.name.equals(f2.name)){
                    //System.out.println("지역변수"+f1.name+":"+v1.data_type+" "+v1.name+"==="+f2.name+":"+v2.data_type+" "+v2.name);
                    sum++;
                }
            }
        }
    }
}
```

[2 – 수정 코드]

System Test Report

– Pairwise Testing

```
result = "2) 함수 이름 유사 검사 => 동일한 이름의 함수 개수 : " + Double.toString(score[1][3]) + "개";
temp = " ∴ 검사 결과 => " + Double.toString(score[1][4]) + "점";
out.write(result);
out.newLine();
out.write(temp);
out.newLine();
result = "3) 함수 Body 유사 검사 => " + Double.toString(score[1][5]) + "%";
temp = " ∴ 검사 결과 => " + Double.toString(score[1][6]) + "점";
result = " ∴ 함수 유사도 검사 최종 점수 => " + Double.toString(score[1][7]) + "점";
out.write(result);
out.newLine();
```

[3 – 기존 코드]

System Test Report

– Pairwise Testing

```
result = "2) 함수 이름 유사 검사 => 동일한 이름의 함수 개수 : " + Double.toString(score[1][3]) + "개";
temp = " ∴ 검사 결과 => " + Double.toString(score[1][4]) + "점";
out.write(result);
out.newLine();
out.write(temp);
out.newLine();
result = "3) 함수 Body 유사 검사 => " + Double.toString(score[1][5]) + "%";
temp = " ∴ 검사 결과 => " + Double.toString(score[1][6]) + "점";
out.write(result);
out.newLine();
out.write(temp);
out.newLine();
result = " ∴ 함수 유사도 검사 최종 점수 => " + Double.toString(score[1][7]) + "점";
out.write(result);
out.newLine();
```

[3 – 수정 코드]

System Test Report

– Brute Force Testing

Test Case No#	1
문제	경로 입력을 하지 아니하고 Input 버튼 클릭, '경로 입력 실패' 팝업 표시를 해야하는데, '.C 파일 말고 다른 파일이 존재' 팝업 표시
원인	GUI 부분과 Setup System 부분을 수정하는 과정에서 문제 발생
대응	수정 완료

Test Case No#	2
문제	존재하지 않는 경로를 입력하고 Input 버튼 클릭, '경로 입력 실패' 팝업 표시를 해야하는데, '.C 파일 말고 다른 파일이 존재' 팝업 표시
원인	GUI 부분과 Setup System 부분을 수정하는 과정에서 문제 발생
대응	수정 완료

System Test Report

– Brute Force Testing

Test Case No#	3
문제	C 파일이 하나만 존재하는 경로를 입력하고 Input 버튼 클릭, '경로 입력 실패' 팝업 표시를 해야하는데 '유효하지 않은 경로' 팝업 표시
원인	GUI 부분과 Setup System 부분을 수정하는 과정에서 문제 발생
대응	수정 완료

Test Case No#	4
문제	프로그램을 1회 정상적으로 동 작 후 존재하지 않는 경로를 입력하고 Input 버튼 클릭, '경로 입력 실패' 팝업 표시 후 Start, Show X_File, Show Detail 버튼 비활성화 되어야 하는데. 모든 버튼 활성화 상태, 이전 분석 결과 출력
원인	경로 입력 실패 시 모든 버튼의 비활성화를 해두지 않음
대응	수정 완료

System Test Report

– Brute Force Testing

Test Case No#	5
문제	Show X_File 버튼을 연속 클릭, Show X_File 버튼 1회 클릭 후 비 활성화, 팝업 화면의 확인 버튼 클릭 전까지 활성화 상태 유지되며 클릭 시, 팝업 화면이 무제한으로 생성됨.
원인	버튼이 누름과 동시에 비활성화 해두지 않아서 발생
대응	수정 완료

Test Case No#	6
문제	C 파일만 존재하는 올바른 경 로를 입력, Input 버튼 클릭 후 Start 버튼 클릭, 소스 코드 분석 후 Show X_File, Show Detail 버튼 활성화 되어야 하는데, 에러 발생
원인	분석하는 과정에서의 문제
대응	수정 완료

System Test Report

– Brute Force Testing

Test Case No#	7
문제	빈 경로 입력 후 Input 버튼 클릭, '경로 입력 실패' 팝업 표시되어야 하는데, '경로 탐색 성공' 팝업 표시
원인	GUI 부분과 Setup System 부분을 수정하는 과정에서 문제 발생
대응	수정 완료

Test Case No#	8
문제	7개의 C 파일이 있는 경로 입력 후 Input, Start 버튼 클릭, 분석에 약 7.7초 소요(Activity 2064)되어야 하는데, Input에 5.18초, Start에 약 4분30초 소요
원인	분석 과정의 내용들이 늘어나서 발생
대응	수정 완료

System Test Report

– Brute Force Testing

Test Case No#	9
문제	입력된 C 파일에 변수의 개수 94개, for문의 개수 9개, do - while 문 존재, 변수의 개수 94개, for문의 개수 9개 검출, do-while문 검출 되어야 하는데, 변수의 개수 93개, for문의 개수 8개로 검출, do-while문 검출하지 못함.
원인	do-while문은 애초에 분석을 하지 않음
대응	do-while문 분석은 분석한다는 언급이 애초에 없었으므로 수정 안함. 나머지 부분은 코드 수정

System Test Report

– Brute Force Testing

Test Case No#	10
문제	동일한 변수의 개수를 파악할 때, (전체 변수 개수 > 동일한 변수 개수) 이어야 하는데, (전체 변수 개수 < 동일한 변수 개수) 로 분석
원인	동일한 소스코드의 변수끼리 비교하여 항상 큰 개수로 나옴

```
//지역변수에서 동일한 변수의 개수 덧셈
for(Function f1: s1.func){
    for(Function f2: s1.func){
        for(Variable v1 : f1.var_list){
            for(Variable v2 : f2.var_list){
                if(v1.name.equals(v2.name) && v1.data_type.equals(v2.data_type) && f1.name.equals(f2.name)){
                    //System.out.println("지역변수"+f1.name+":"+v1.data_type+" "+v1.name+"=
                    sum++;
                }
            }
        }
    }
}
```

```
//지역변수에서 동일한 변수의 개수 덧셈
for(Function f1: s1.func){
    for(Function f2: s2.func){
        for(Variable v1 : f1.var_list){
            for(Variable v2 : f2.var_list){
                if(v1.name.equals(v2.name) && v1.data_type.equals(v2.data_type) && f1.name.equals(f2.name)){
                    //System.out.println("지역변수"+f1.name+":"+v1.data_type+" "+v1.name+"==="+f2.name+":"+v2.data_type+" "+v2.name);
                    sum++;
                }
            }
        }
    }
}
```

System Test Report

– Brute Force Testing

Test Case No#	11
문제	원본 파일 A와 원본과 유사한 파일 B를 입력함, X_File로 A가 검출 되어야 하는데, X_File로 B가 검출
원인	X_File을 찾는 데에 평균치로 찾기 때문에 1:1 로는 한계치가 있음
대응	수정 불가

Test Case No#	12
문제	main() 함수만 존재하는 C 파일 입력, 정상적으로 분석해야 하는데, 에러 발생
원인	수정 과정에서 에러가 발생하지 않아서 잘 모르겠음
대응	수정 불가

System Test Report

– Brute Force Testing

Test Case No#	13
문제	내용이 존재하는 C 파일 2개와 내용이 없는 C 파일 2 개를 입력, 정상적으로 분석해야 하는데, 에러 발생
원인	수정 과정에서 에러가 발생하지 않아서 잘 모르겠음
대응	수정 불가

Test Case No#	14
문제	분석에 시간이 많이 소요, 분석 진행중을 안내하는 팝업 표시가 되고, 프로그램의 정상적인 동작이 되어야 하는데, 분석 완료 팝업이 뜨기 이전까지 프로그램이 정지, 모든 버튼 입력 불가능
원인	GUI Frame을 띄우는 과정에서 발생하는 한계
대응	수정 불가

Static Analysis Report

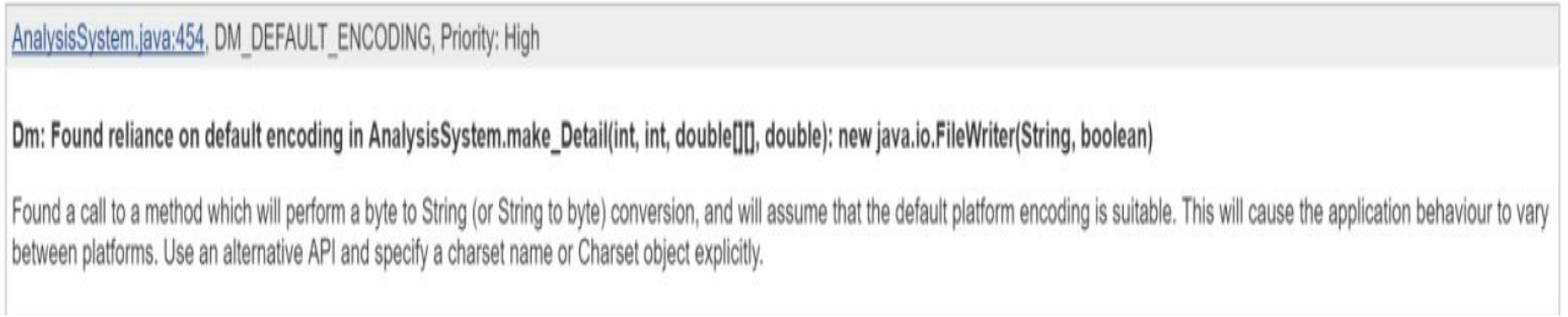
- SonarQube

분석 결과	가장 많이 위반한 규칙 : Duplicated lines A. AnalysisSytem.java 파일의 전체 659라인 중 238라인이 중복 B. SetupSystem.java 파일의 전체 466라인 중 52라인이 중복
대응	어느 부분이 중복인지 정확히 보이지 않아 대응하기에 모호하였고, 중복의 원인이 함수 호출이나, 유사도 검사를 위한 반복문의 잦은 사용으로 보임.

Static Analysis Report

- FindBugs (High 등급 수정)

AnalysisSystem.java : 454 DM_DEFAULT_ENCODING

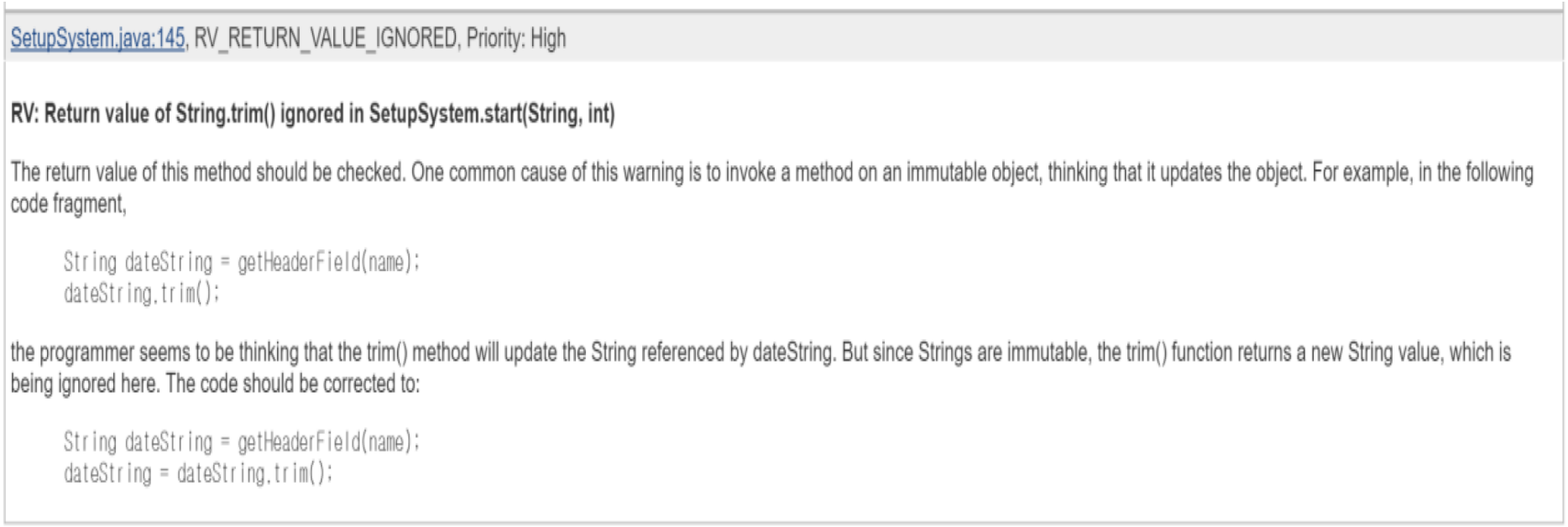


원인 : 플랫폼 마다 동작이 다를 수 있다고 Warning 하는 부분
대응 : 본 프로그램은 구현 환경과, 동작 환경 OS가 Windows이므로 딱히 문제가 없다고 판단

Static Analysis Report

- FindBugs (High 등급 수정)

SetupSystem.java:145 RV_RETURN_VALUE_IGNORED



SetupSystem.java:145, RV_RETURN_VALUE_IGNORED, Priority: High

RV: Return value of String.trim() ignored in SetupSystem.start(String, int)

The return value of this method should be checked. One common cause of this warning is to invoke a method on an immutable object, thinking that it updates the object. For example, in the following code fragment,

```
String dateString = getHeaderField(name);
dateString.trim();
```

the programmer seems to be thinking that the trim() method will update the String referenced by dateString. But since Strings are immutable, the trim() function returns a new String value, which is being ignored here. The code should be corrected to:

```
String dateString = getHeaderField(name);
dateString = dateString.trim();
```

원인 : String 변수를 trim 한 후에 String 변수에 다시 배정을 안해서 발생한 Warning
대응 : trim 한 후에 다시 String 변수에 배정

Static Analysis Report

- FindBugs (High 등급 수정)

[기존 코드]

```
144                 temp_str = temp.replaceAll(";", "");
145                 temp_str.trim();
146                 temp_str = temp_str.substring(temp_str.lastIndexOf(" ")+1);
147                 int exist_flag=0;//없으면0 있으면1
148                 for(String s: sc[index].type_list){
149                     if(s.equals(temp_str)){
```

Static Analysis Report

- FindBugs (High 등급 수정)

[수정 코드]

```
144             temp_str = temp.replaceAll(";", "");
145             temp_str = temp_str.trim();
146             temp_str = temp_str.substring(temp_str.lastIndexOf(" ")+1);
147             int exist_flag=0;//없으면0 있으면1
148             for(String s: sc[index].type_list){
```

Static Analysis Report

– FindBugs (Normal)

AnalysisSystem.java:85, SF_SWITCH_NO_DEFAULT, Priority: Normal

SF: Switch statement found in AnalysisSystem.analyzeVar(SourceCode, SourceCode) where default case is missing

This method contains a switch statement where default case is missing. Usually you need to provide a default case.

Because the analysis only looks at the generated bytecode, this warning can be incorrect triggered if the default case is at the end of the switch statement and the switch statement doesn't contain break statements for other cases.

AnalysisSystem.java:225, SBSC_USE_STRINGBUFFER_CONCATENATION, Priority: Normal

SBSC: AnalysisSystem.analyzeFunc(SourceCode, SourceCode) concatenates strings using + in a loop

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.

Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.

For example:

```
// This is bad
String s = "";
for (int i = 0; i < field.length; ++i) {
  s = s + field[i];
}

// This is better
StringBuffer buf = new StringBuffer();
for (int i = 0; i < field.length; ++i) {
  buf.append(field[i]);
}
String s = buf.toString();
```

GUI.java:346, DLS_DEAD_LOCAL_STORE, Priority: Normal

DLS: Dead store to e1 in GUI$6.actionPerformed(ActionEvent)

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because FindBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

Static Analysis Report

– FindBugs (Normal)

GUI.java:392, DM_EXIT, Priority: Normal

Dm: GUI$7$1.actionPerformed(ActionEvent) invokes System.exit(...), which shuts down the entire virtual machine

Invoking System.exit shuts down the entire Java virtual machine. This should only been done when it is appropriate. Such calls make it hard or impossible for your code to be invoked by other code. Consider throwing a RuntimeException instead.

Loop.java:6, URF_UNREAD_FIELD, Priority: Normal

UrF: Unread field: Loop.loop_type

This field is never read. Consider removing it from the class.

Loop.java:7, URF_UNREAD_FIELD, Priority: Normal

UrF: Unread field: Loop.loop_body

This field is never read. Consider removing it from the class.

MainController.java:13, URF_UNREAD_FIELD, Priority: Normal

UrF: Unread field: MainController.folder_path

This field is never read. Consider removing it from the class.

SetupSystem.java:30, NP_NULL_ON_SOME_PATH_FROM_RETURN_VALUE, Priority: Normal

NP: Possible null pointer dereference in SetupSystem.folder_list(String) due to return value of called method

The return value from a method is dereferenced without a null check, and the return value of that method is one that should generally be checked for null. This may lead to a `NullPointerException` when the code is executed.

Static Analysis Report

- FindBugs (Normal)

SetupSystem.java:83, SBSC_USE_STRINGBUFFER_CONCATENATION, Priority: Normal

SBSC: SetupSystem.setting_files(String, int) concatenates strings using + in a loop

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.

Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.

For example:

```
// This is bad
String s = "";
for (int i = 0; i < field.length; ++i) {
  s = s + field[i];
}

// This is better
StringBuffer buf = new StringBuffer();
for (int i = 0; i < field.length; ++i) {
  buf.append(field[i]);
}
String s = buf.toString();
```

SetupSystem.java:271, SBSC_USE_STRINGBUFFER_CONCATENATION, Priority: Normal

SBSC: SetupSystem.start(String, int) concatenates strings using + in a loop

The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration.

Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.

For example:

```
// This is bad
String s = "";
for (int i = 0; i < field.length; ++i) {
  s = s + field[i];
}

// This is better
StringBuffer buf = new StringBuffer();
for (int i = 0; i < field.length; ++i) {
  buf.append(field[i]);
}
String s = buf.toString();
```

Static Analysis Report

– FindBugs (Normal)

SetupSystem.java:463, EI_EXPOSE_REP, Priority: Normal

EI: SetupSystem.getSc() may expose internal representation by returning sc

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

Variable.java:8, URF_UNREAD_PUBLIC_OR_PROTECTED_FIELD, Priority: Normal

UrF: Unread public/protected field: Variable.count

This field is never read. The field is public or protected, so perhaps it is intended to be used with classes not seen as part of the analysis. If not, consider removing it from the class.

Static Analysis Report

- FindBugs (대응 방안)

대응 방안(예정)

1) Switch 문에서 Default의 경우가 없음
   => case default : break; 추가
2) String에 + 연산을 반복문을 통하여 진행하여 발생하는 Warning
   => StringBuffer의 append()함수를 이용하여 변수를 계산한 뒤에 최종적인 값을 String에 대입
3) Unread 되었던 부분
   => 삭제를 하거나 read 하는 부분이 있도록 수정

Static Analysis Report

- PMD

AnalysisSystem.java:4, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.util.ArrayList'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:3, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.awt.FileDialog'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:7, UnusedImports, Priority: Normal

Avoid unused imports such as 'javax.swing.JScrollBar'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

Static Analysis Report

- PMD

GUI.java:8, UnusedImports, Priority: Normal

Avoid unused imports such as 'javax.swing.JScrollPane'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:9, UnusedImports, Priority: Normal

Avoid unused imports such as 'javax.swing.JTextArea'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:15, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.io.BufferedWriter'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

Static Analysis Report

- PMD

GUI.java:16, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.io.FileWriter'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:24, UnusedImports, Priority: Normal

Avoid unused imports such as 'javax.swing.ScrollPaneConstants'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

GUI.java:26, DontImportJavaLang, Priority: Normal

Avoid importing anything from the package java.lang.

Avoid importing anything from the package 'java.lang'. These classes are automatically imported (JLS 7.5.3).

```
import java.lang.String;        // this is unnecessary

public class Foo {}

// --- in another source code file...

import java.lang.*;     // this is bad

public class Foo {}
```

Static Analysis Report

- PMD

SetupSystem.java:7, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.io.StringReader'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

SetupSystem.java:8, DontImportJavaLang, Priority: Normal

Avoid importing anything from the package java.lang.

Avoid importing anything from the package 'java.lang'. These classes are automatically imported (JLS 7.5.3).

```
import java.lang.String;        // this is unnecessary

public class Foo {}

// --- in another source code file...

import java.lang.*;     // this is bad

public class Foo {}
```

SetupSystem.java:9, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.util.LinkedList'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

Static Analysis Report

- PMD

SourceCode.java:2, UnusedImports, Priority: Normal

Avoid unused imports such as 'java.util.LinkedList'.

Avoid the use of unused import statements to prevent unwanted dependencies.

```
// this is bad
import java.io.File;
public class Foo {}
```

Static Analysis Report

- PMD 대응 : AnalysisSystem.java

Line Number : 4

```
1 import java.io.BufferedWriter;
2 import java.io.FileWriter;
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.Calendar;
```

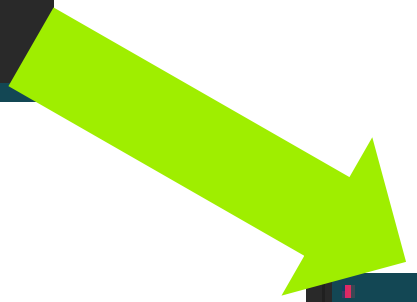
```
1 import java.io.BufferedWriter;
2 import java.io.FileWriter;
3 import java.io.IOException;
4 import java.util.Calendar;
5
```

Static Analysis Report

- PMD 대응 : SourceCode.java

Line Number : 2

```
import java.util.ArrayList;
import java.util.LinkedList;
```



```
import java.util.ArrayList;
```

Static Analysis Report

- PMD 대응 : GUI.java

Line Number : 3, 7, 8, 9, 15, 16, 24

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.FileDialog;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollBar;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.awt.event.ActionEvent;
import javax.swing.SwingConstants;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;
```

Static Analysis Report

- PMD 대응 : GUI.java

```
1  import java.awt.Color;
2  import java.awt.EventQueue;
3
4  import javax.swing.JFrame;
5  import javax.swing.JPanel;
6  import javax.swing.border.EmptyBorder;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9  import javax.swing.JButton;
10 import java.awt.event.ActionListener;
11 import java.io.IOException;
12 import java.awt.event.ActionEvent;
13 import javax.swing.SwingConstants;
14 import java.awt.Font;
15 import java.awt.Graphics;
16
17 import javax.swing.JTextField;
18
19 import java.lang.Runtime;
```

Static Analysis Report

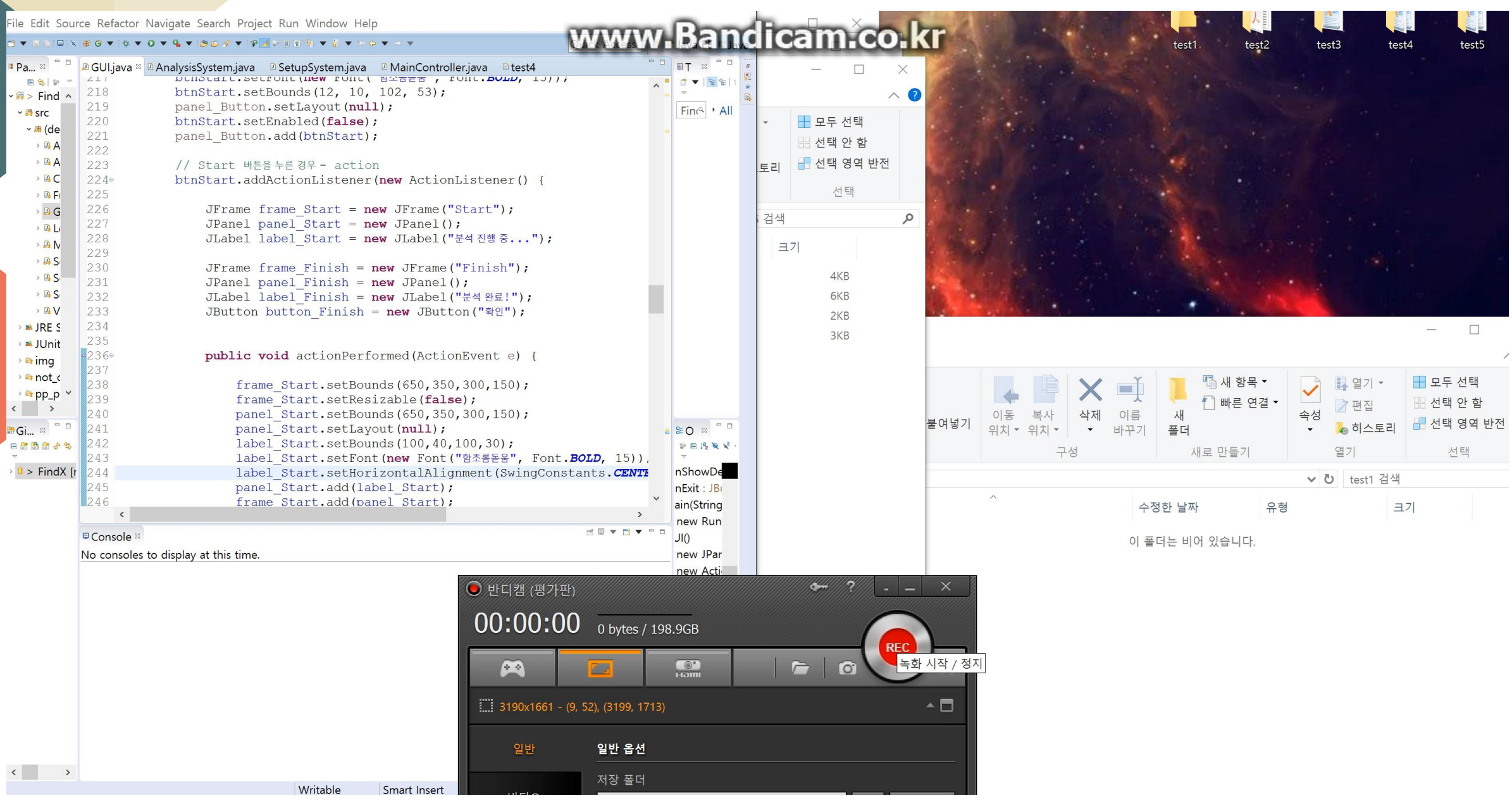
- PMD 대응 : SetupSystem.java

Line Number : 7, 9

```
import java.io.StringReader;
import java.lang.String;
import java.util.LinkedList;
```

```
import java.lang.String;
import java.util.regex.Matcher;
```

Demo


File Edit Source Refactor Navigate Search Project Run Window Help
www.Bandicam.co.kr
GUI.java
AnalysisSystem.java
SetupSystem.java
MainController.java
test4
btnStart.setBounds(12, 10, 102, 53);
panel_Button.setLayout(null);
btnStart.setEnabled(false);
panel_Button.add(btnStart);
// Start 버튼을 누른 경우 - action
btnStart.addActionListener(new ActionListener() {
JFrame frame_Start = new JFrame("Start");
JPanel panel_Start = new JPanel();
JLabel label_Start = new JLabel("분석 진행 중...");
JFrame frame_Finish = new JFrame("Finish");
JPanel panel_Finish = new JPanel();
JLabel label_Finish = new JLabel("분석 완료!");
JButton button_Finish = new JButton("확인");
public void actionPerformed(ActionEvent e) {
frame_Start.setBounds(650,350,300,150);
frame_Start.setResizable(false);
panel_Start.setBounds(650,350,300,150);
panel_Start.setLayout(null);
label_Start.setBounds(100,40,100,30);
label_Start.setFont(new Font("함초롬돋움", Font.BOLD, 15));
label_Start.setHorizontalAlignment(SwingConstants.CENTE
panel_Start.add(label_Start);
frame_Start.add(panel_Start);
Console
No consoles to display at this time.
Writable
Smart Insert
모두 선택
선택 안 함
선택 영역 반전
선택
검색
크기
4KB
6KB
2KB
3KB
test1
test2
test3
test4
test5
새 항목
빠른 연결
이동 위치
복사 위치
삭제
이름 바꾸기
새 폴더
구성
새로 만들기
열기
편집
히스토리
속성
열기
선택
test1 검색
수정한 날짜
유형
크기
이 폴더는 비어 있습니다.
반디캠 (평가판)
00:00:00
0 bytes / 198.9GB
REC
녹화 시작 / 정지
3190x1661 - (9, 52), (3199, 1713)
일반
일반 옵션
저장 폴더

Review

Review

- OOPT 개발 프로세스 사용

Review

- 4학년과의 co-working 소감 및 장단점

THANK YOU