2016 소프트웨어 모델링&분석

그 놈! Clone Checker

Team4

Contents

Modified Part

1) Functional Requirements

OSP Stage 1000 ver5	OSP Stage 1000 ver6
- Display Main	- Display Main
- Input Path	- Input Path
- Setting Files	- Setting Files
- Start Analyze	- Start Analyze Code
- Analyze Change Name	- Analyze Variable
- Analyze Loop	- Analyze Loop
- Analyze Conditional	- Analyze Conditional
- Analyze Function	- Analyze Function
- Calculate Similarity	- Make Detail
- Show X_File	- Find X_File
- Show Detail	- Show X_File
- Exit	- Show Detail
	- Exit

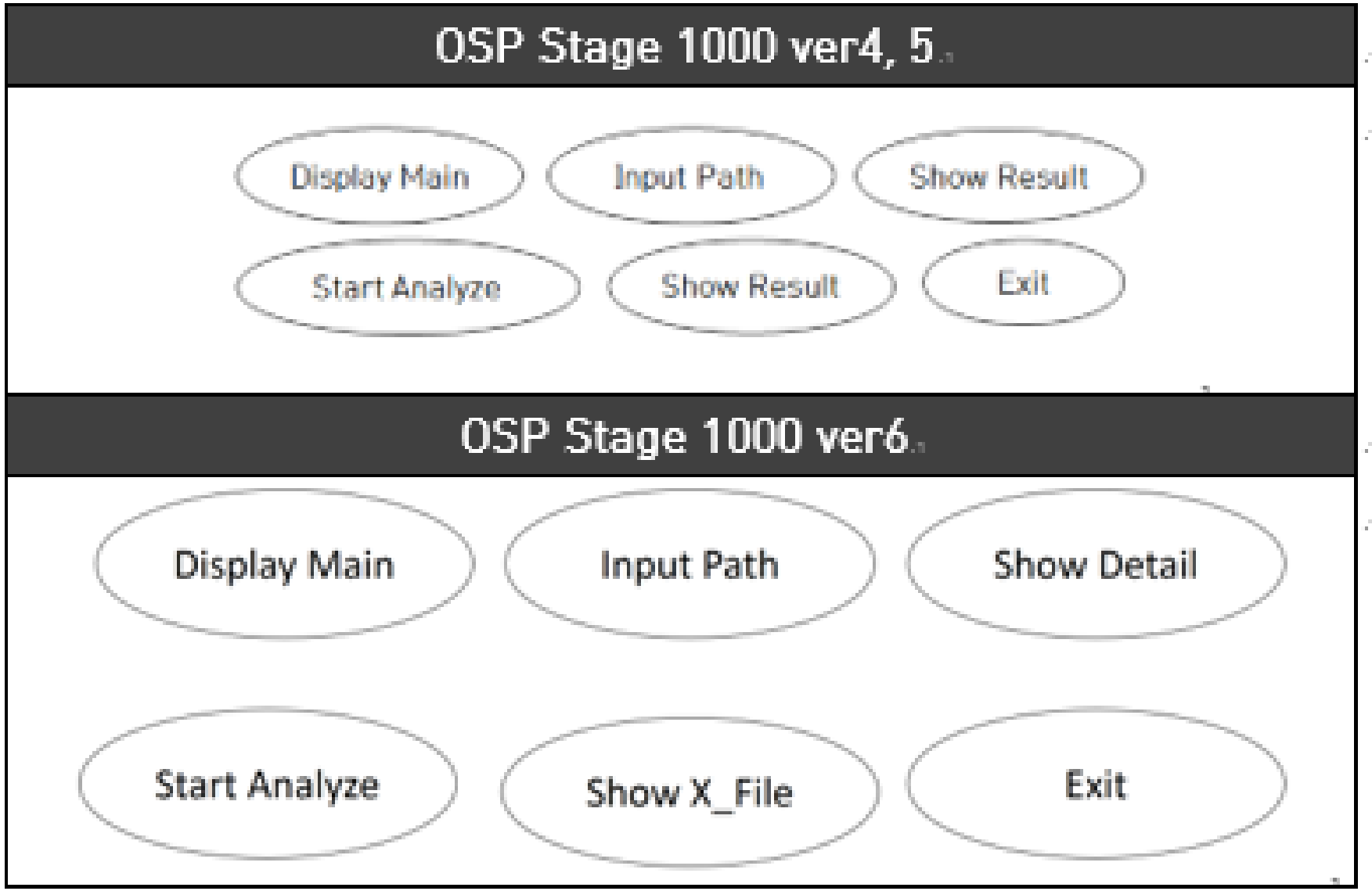
Modified Part

OSP Stage 1000 ver5	
Function	Description
Display Main	UI 를 포함한 실행 초기 화면
Input Path	검사하는 파일들이 저장되어 있는 폴더의 경로를 입력해준다.
Setting Files	분석에 앞서 파일을 분석하기 쉽게 정리한다
Start Analyze Code	분석을 시작한다.
Analyze Variable	변수에 대한 검사를 한다.
Analyze Loop	반복문을 검사한다.
Analyze Conditional	조건문을 검사한다.
Analyze Function	함수에 대한 검사를 한다.
Calculate Similarity	분석 결과를 계산한다.
Show X_File	원본으로 추정되는 파일명을 보여준다.
Show Detail	자세한 분석 결과를 보여준다.
Exit	유사도 검사 프로그램을 종료한다.

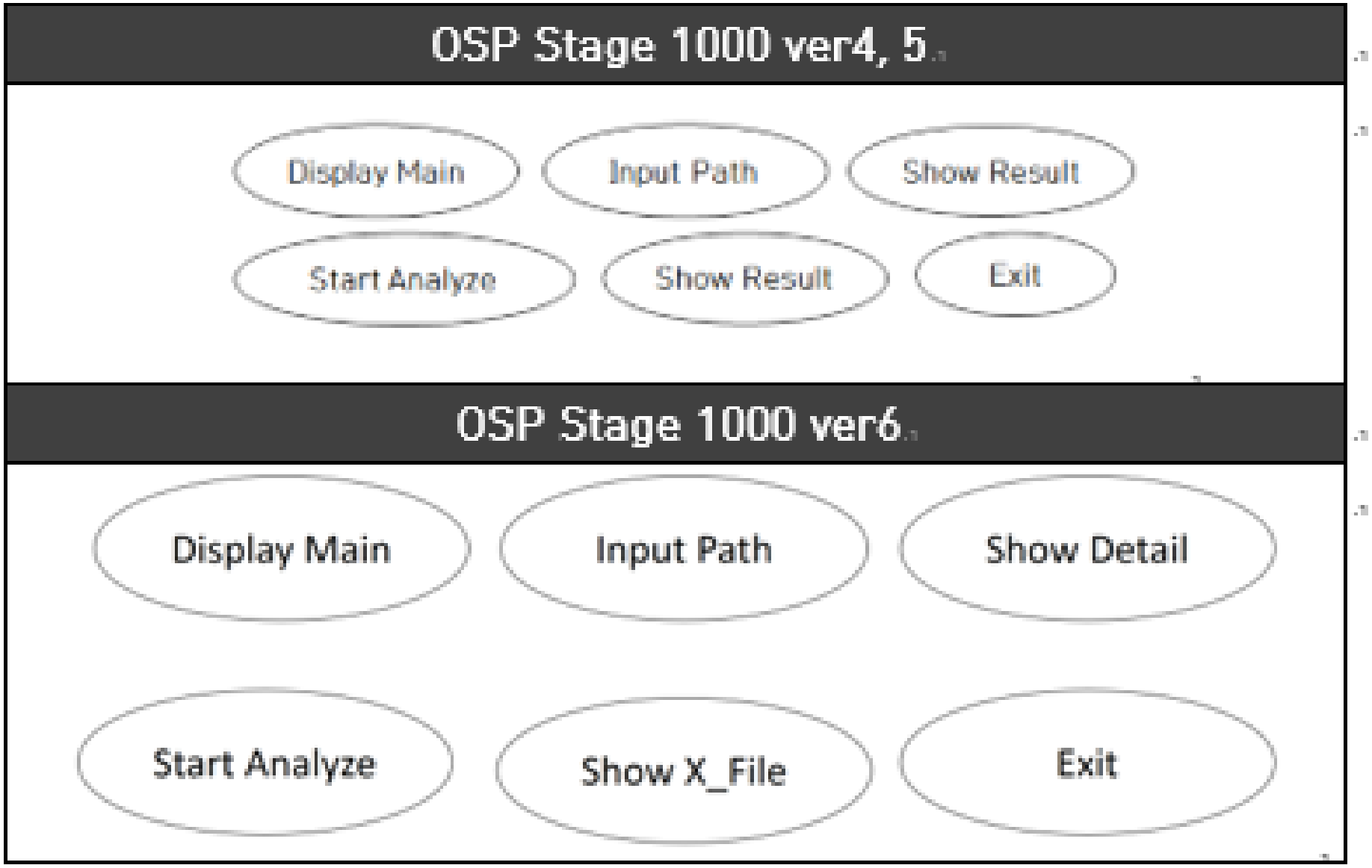
Modified Part

OSP Stage 1000 ver6	
Function	Description
Display Main	UI를 포함한 실행 초기 화면
Input Path	검사하는 파일들이 저장되어 있는 폴더의 경로를 입력해준다.
Setting Files	분석에 앞서 파일을 분석하기 쉽게 정리한다
Start Analyze Code	분석을 시작한다.
Analyze Variable	변수에 대한 검사를 한다.
Analyze Loop	반복문을 검사한다.
Analyze Conditional	조건문을 검사한다.
Analyze Function	함수에 대한 검사를 한다.
Make Detail	최종 유사도 점수를 계산하고, 검사의 세부 내용을 .txt 파일에 저장한다.
Find X_File	X_File을 찾는다.
Show X_File	X_File의 파일명을 보여준다.
Show Detail	검사의 세부 내용을 보여준다.
Exit	유사도 검사 프로그램을 종료한다.

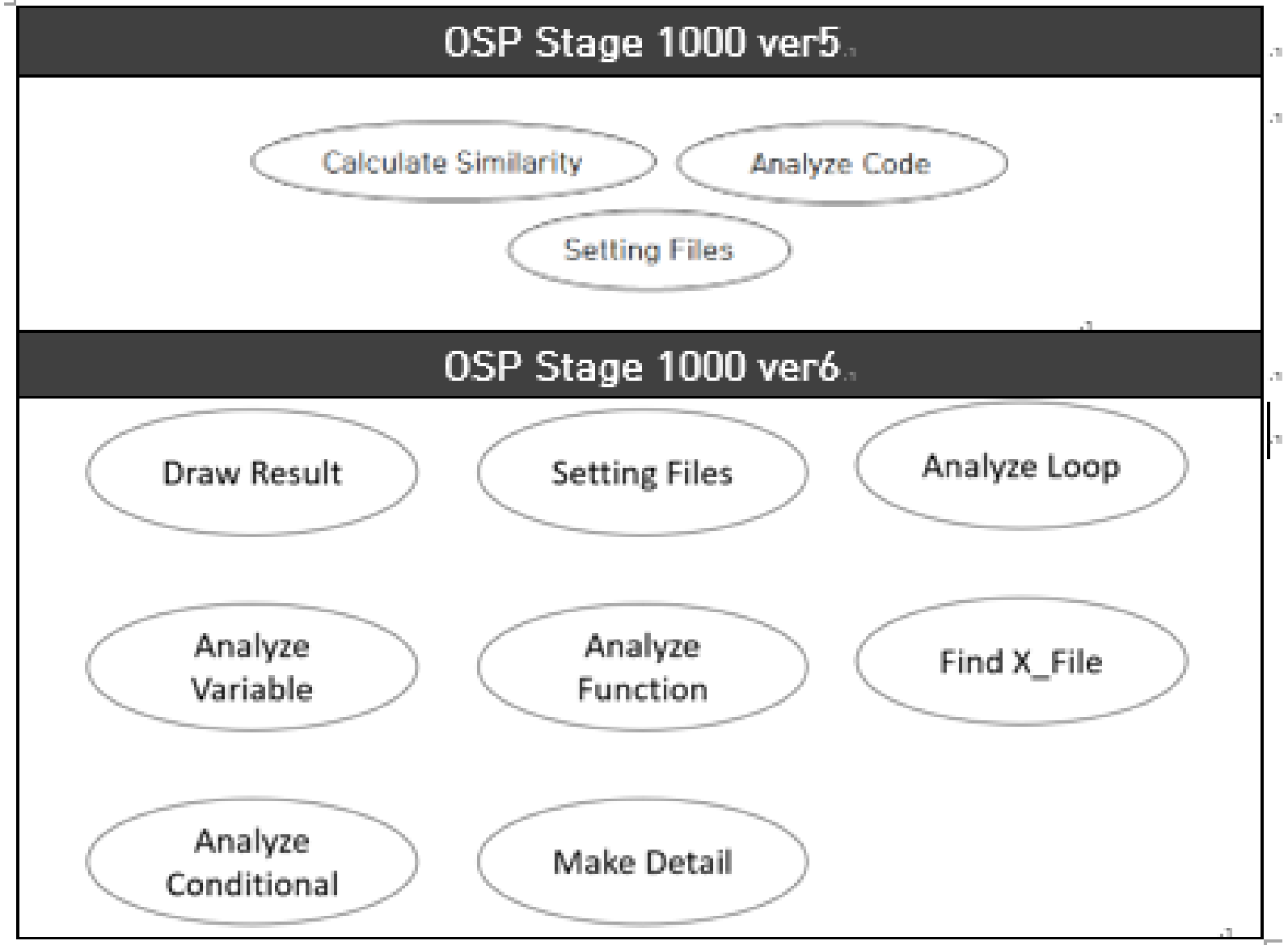
Modified Part



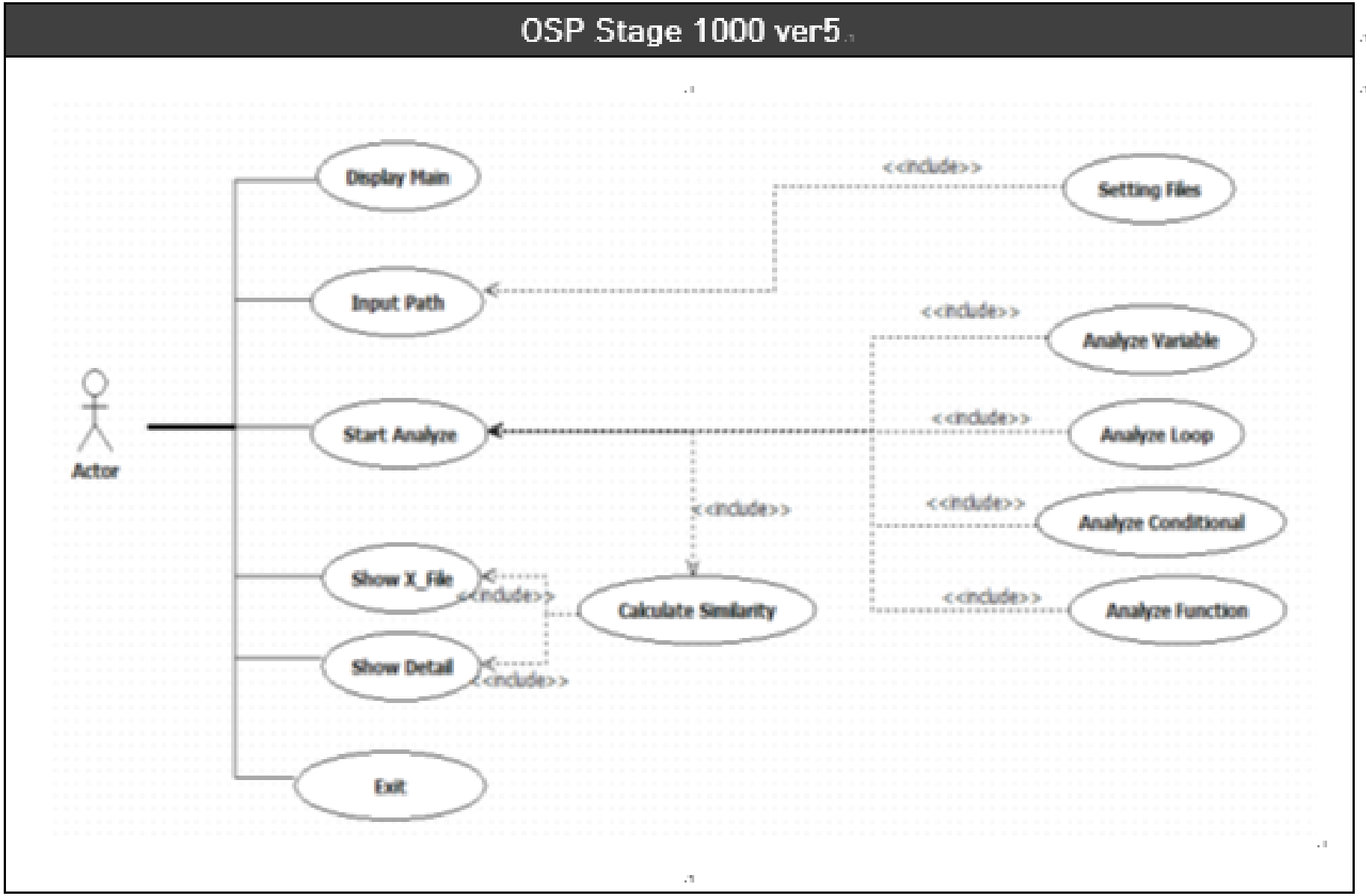
Modified Part



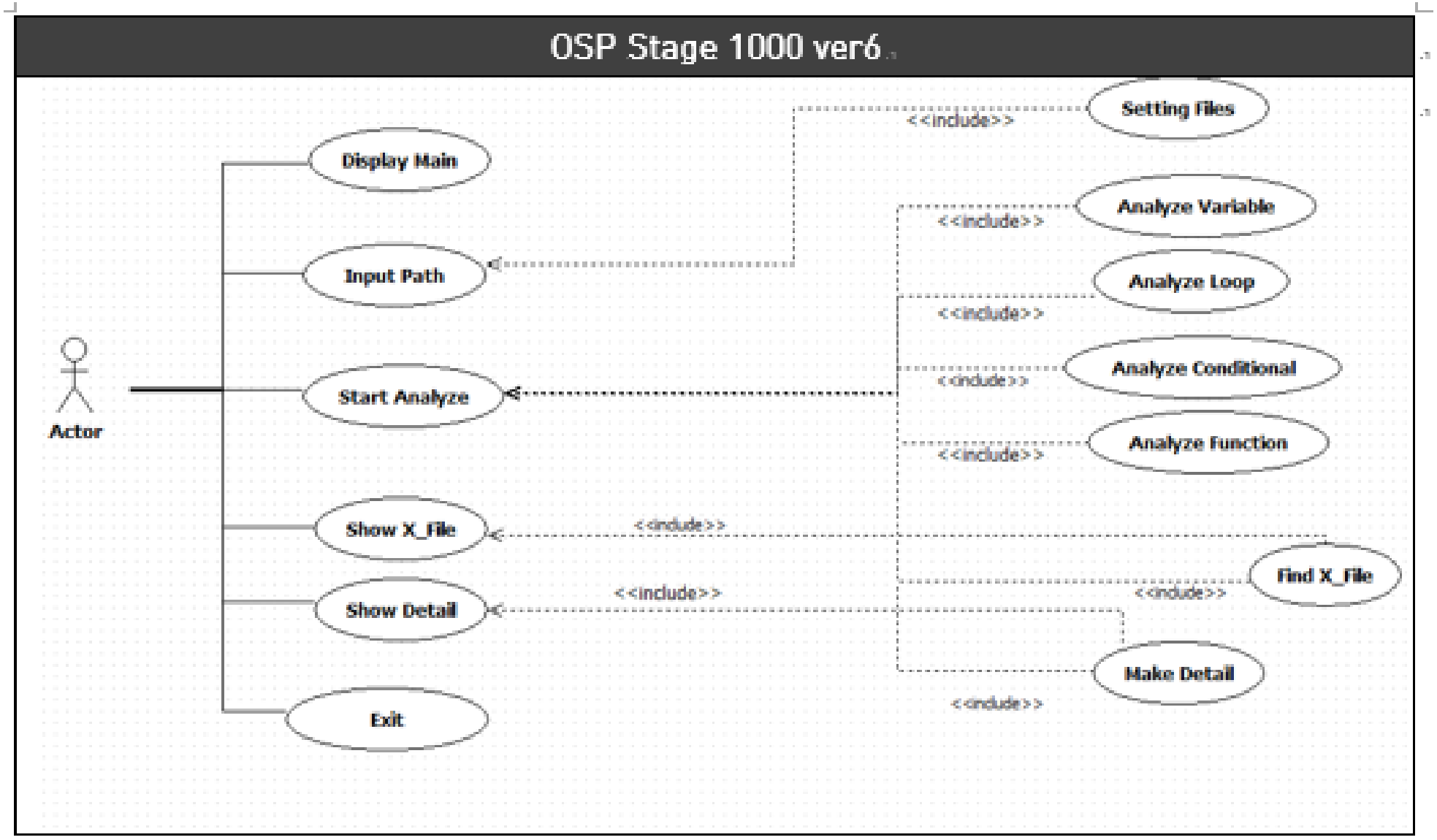
Modified Part



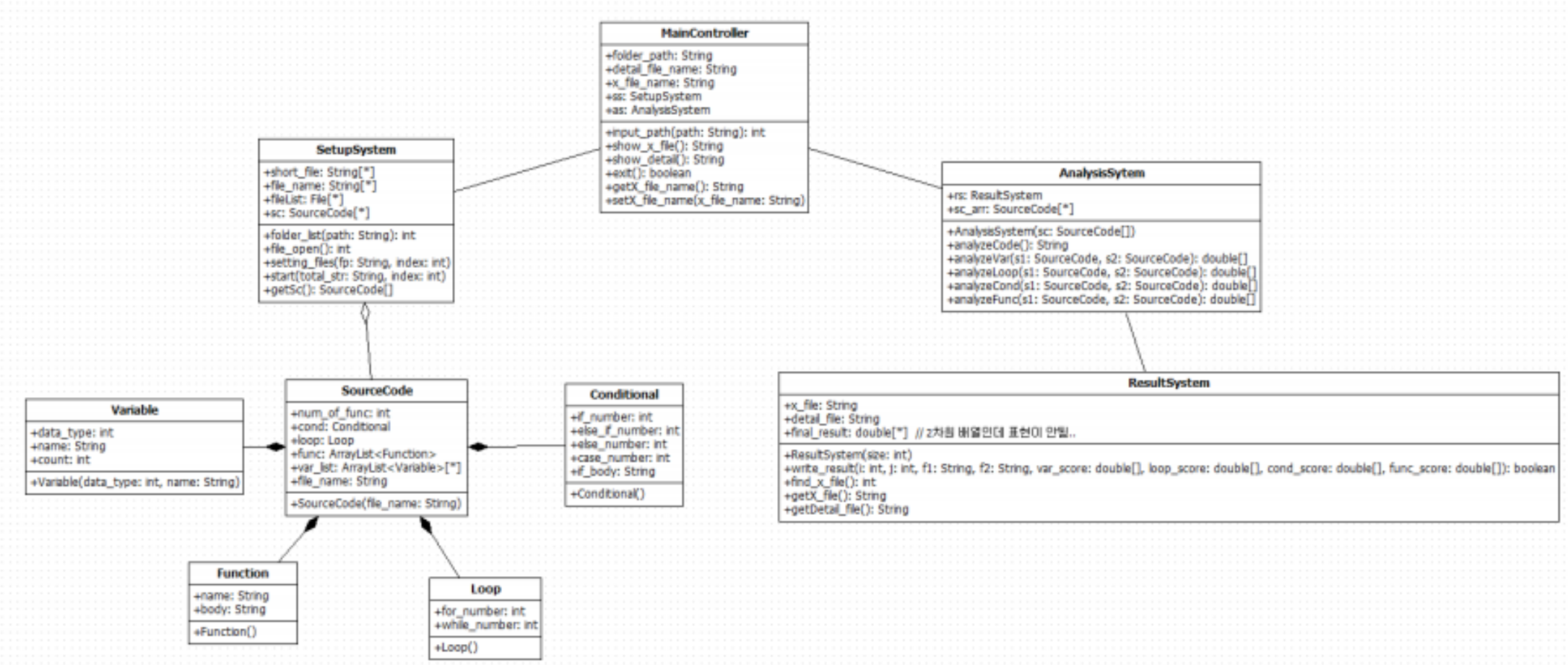
Modified Part



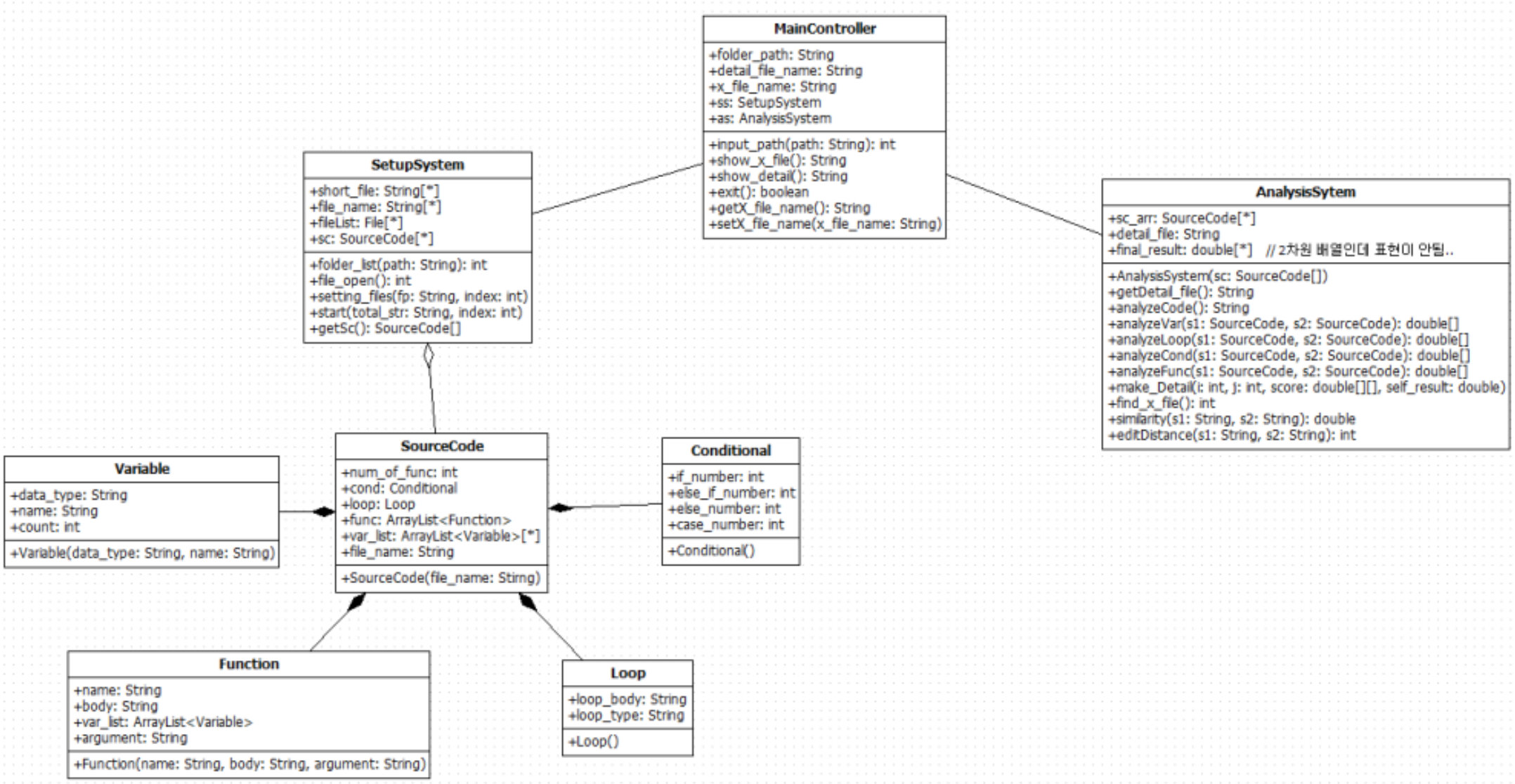
Modified Part



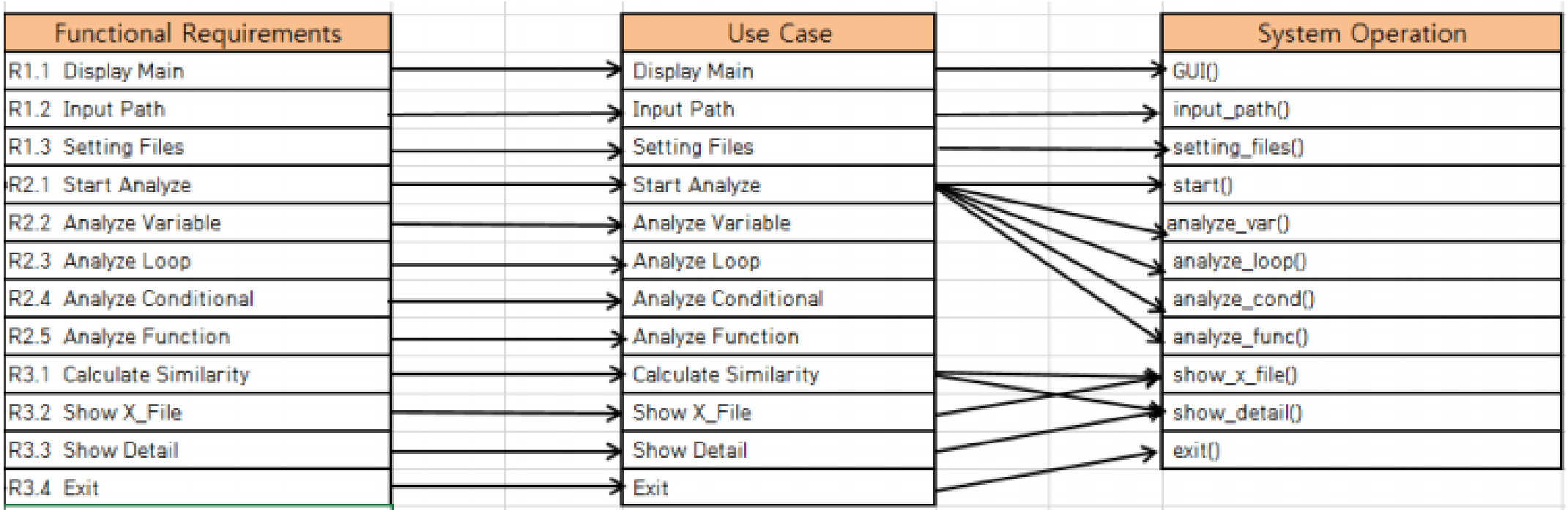
Modified Part



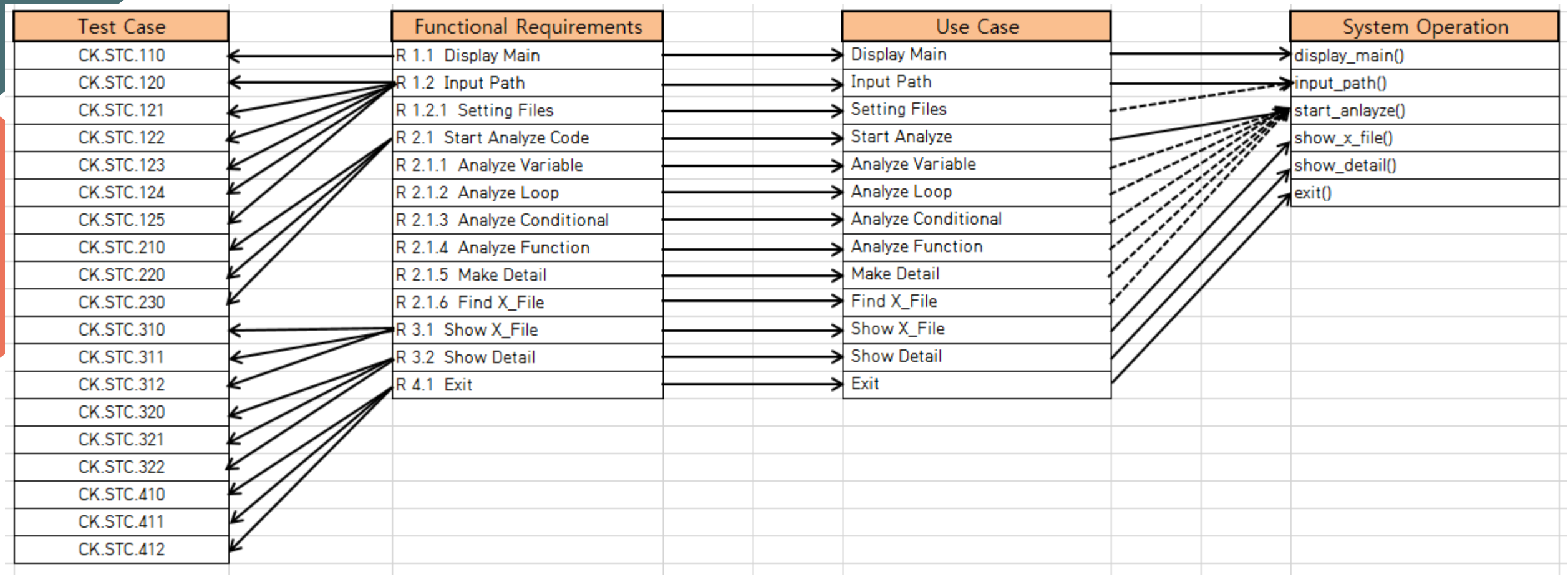
Modified Part



Modified Part



Modified Part



Unit Test - Analysis System

```java
import static org.junit.Assert.*;

public class AnalysisSystemTest {

    @Test
    public void testAnalyzeCode() {
        MainController mc = new MainController();
        mc.input_path("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good");
        mc.start_analyze();

        //그놈 파일(X-File)이 .c로 끝나는지 확인
        assertTrue(mc.as.analyzeCode().substring(mc.as.analyzeCode().length()-2).equals(".c"));
    }

}

```

Unit Test - Setup System

```java
import static org.junit.Assert.*;

public class SetupSystemTest {
    @Test
    public void testFolder_list() {
        SetupSystem ss = new SetupSystem();

        //1-1 ".c" 파일만 들어있는 폴더의 경로를 입력했을때, 2가 반환되는지 확인
        assertEquals(2,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good"));
        //1-2 ".c" 파일만 들어있지 않은 폴더의 경로를 입력했을때, 1이 반환되는지 확인
        assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MServer\\asd"));
        //1-3 경로형태가 아닌 이상한 문자를 입력했을때, 1이 반환되는지 확인
        assertEquals(1,ss.folder_list("is_not_directory"));
        //1-4 없는 경로를 입력했을때, 1이 반환되는지 확인
        assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\IS_NOT_DIRECTORY"));
        //1-5 file_name에 경로 디렉토리에 있는 파일이 제대로 들어가는지 확인
        ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good");
        assertEquals("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good\\AbsoCompare - 복사본 - 복사본 - 복사본.c",ss.file_name[0]);
    }

}
```

Unit Test - Setup System

```
22
23      @Test
24      public void testFile_open() {
25
26          SetupSystem ss = new SetupSystem();
27          //2-1 디렉토리내에 ".c"파일들만 존재할 때, flag가 2를 반환하는지 확인
28          ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good");
29          assertEquals(2,ss.file_open());
30          //2-2 디렉토리내에 ".c"말고 다른파일이 존재할때, flag가 1을 반환하는지 확인
31          ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd");
32          assertEquals(1,ss.file_open());
33      }
34
35      @Test
36      public void testStart() {
37          SetupSystem ss = new SetupSystem();
38          //3-1 Start() 실행시 Variable객체가 제대로 리스트에 들어가있는지 확인
39          ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good");
40          assertNotNull(ss.sc[0].func.get(0).var_list.get(0));
41          //3-2 Start() 실행시 i번째 소스코드의 함수 개수와 sc[i].num_of_func 같은지 확인
42          assertEquals(3,ss.sc[0].num_of_func); //첫번째 소스코드,즉,"AbsoCompare - 복사본 - 복사본 - 복사본.c"안에는 3개의 함수가있음.
43      }
44
```

Unit Test - Setup System

```
44
45      @Test
46      public void testGetSc() {
47          SetupSystem ss = new SetupSystem();
48          //4-1 GetSc() 실행시 Sc[]가 제대로 반환되는지 확인
49          ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\good");
50          assertSame(ss.sc,ss.getSc());
51      }
52
53  }
```

System Testing

– Category Partition Testing & Pairwise Testing

Test Case No#	100.204.300.310.402.410.630.701.711.721.731.741.751.801.811.82 1.831.901.911.1002.1102.1112.1202.1212.1302.1312.1322.1332.14 02.500.600.510.610.511.620.512.513.
문제	경로탐색성공 팝업 노출, 분석완료 팝업 노출, 분석결과파일 생성, X_File 명 노출, 분석결과내용 노출, 종료 순서로 실행되지 않음.
원인	SetupSystem.start() 함수 부분의 구현 누락.
대응	기능 구현.

Test Case No#	100.204.300.310.402.410.630.701.711.721.731.741.751.801.811.82 1.831.901.911.1002.1102.1112.1202.1212.1302.1312.1322.1332.14 02.500.600.510.610.511.620.513.
문제	경로탐색성공 팝업 노출, 분석완료 팝업 노출, 분석결과파일 생성, X_File 명 노출, 종료 순서로 실행되지 않음.
원인	SetupSystem.start() 함수 부분의 구현 누락.
대응	기능 구현.

System Testing

– Category Partition Testing & Pairwise Testing

Test Case No#	100.204.300.310.402.410.630.701.711.721.731.741.751.801.811.82 1.831.901.911.1002.1102.1112.1202.1212.1302.1312.1322.1332.14 02.500.600.510.610.512.511.620.513.
문제	경로탐색성공 팝업 노출, 분석완료 팝업 노출, 분석결과파일 생성, 분석결과내용 노출, X_File 명 노출, 종료 순서로 실행되지 않음.
원인	SetupSystem.start() 함수 부분의 구현 누락.
대응	기능 구현.

Test Case No#	100.204.300.310.402.410.630.701.711.721.731.741.751.801.811.82 1.831.901.911.1002.1102.1112.1202.1212.1302.1312.1322.1332.14 02.500.600.510.610.512.513.
문제	경로탐색성공 팝업 노출, 분석완료 팝업 노출, 분석결과파일 생성, 분석결과내용 노출, 종료 순서로 실행되지 않음.
원인	SetupSystem.start() 함수 부분의 구현 누락.
대응	기능 구현.

System Testing

– Category Partition Testing & Pairwise Testing

Test Case No#	100.204.300.310.402.410.630.701.711.721.731.741.751.801.811.821.831.901.911.1002.1102.1112.1202.1212.1302.1312.1322.1332.1402.500.600.510.610.513
문제	경로탐색성공 팝업 노출, 분석완료 팝업 노출, 분석결과파일 생성, 종료 순서로 실행되지 않음
원인	SetupSystem.start() 함수 부분의 구현 누락
대응	기능 구현

Test Case No#	100.204.300.310.402.410.500.600.511.620, 513
문제	경로탐색성공 팝업 노출, X_File 명 노출 안 함, 종료 순서로 실행되지 않음
원인	SetupSystem.start() 함수 부분의 구현 누락
대응	기능 구현

System Testing

– Category Partition Testing & Pairwise Testing

Test Case No#	100.204.300.310.402.410.500.600.512, 513.
문제	경로탐색성공 팝업 노출, 분석결과내용 노출 안 함, 종료 순서로 실행되지 않음.
원인	SetupSystem.start() 함수 부분의 구현 누락.
대응	기능 구현.

System Testing

– Category Partition Testing & Pairwise Testing

```
95      public void start(String total_str,int index){
96
97          sc[index].type_list.add("int");
98          sc[index].type_list.add("int*");
99          sc[index].type_list.add("int**");
100         sc[index].type_list.add("int[]");
101         sc[index].type_list.add("int[][]");
102         sc[index].type_list.add("unsigned int");
103         sc[index].type_list.add("short");
104         sc[index].type_list.add("long");
105         sc[index].type_list.add("double");
106         sc[index].type_list.add("double*");
107         sc[index].type_list.add("double**");
108         sc[index].type_list.add("double[]");
109         sc[index].type_list.add("double[][]");
110         sc[index].type_list.add("float");
111         sc[index].type_list.add("char");
112         sc[index].type_list.add("char*");
113         sc[index].type_list.add("char**");
114         sc[index].type_list.add("char[]");
115         sc[index].type_list.add("char[][]");
116         sc[index].type_list.add("void");
117         sc[index].type_list.add("void*");
118         sc[index].type_list.add("bool");
119         sc[index].type_list.add("file*");
120         sc[index].type_list.add("size_t");
121         //기본적으로 파싱해야하는 자료형들 type_list에 추가
```

```
124         String str;
125         str = total_str;
126         //1중포인터,2중포인터 모두 앞의 자료형에 붙이고 뒤의 변수이름엔 띄워지도록 수정
127         total_str = total_str.replaceAll("(\\s)*\\*(\\s)*", "* ");
128         total_str = total_str.replaceAll("\\* \\*", "**");
129
130             Pattern p_struc = Pattern.compile("[ |\\t]*typedef[ |\\t]*.*;");
131             Matcher m_struc = p_struc.matcher(str);
132             while(m_struc.find()){
133                 String temp = m_struc.group();
134                 String temp2 = temp;
135                 temp2 = temp2.replaceAll("\\*", "\\\\*");
136                 temp2 = temp2.replaceAll("\\[", "\\\\[");
137                 temp2 = temp2.replaceAll("\\]", "\\\\]");
138                 total_str = total_str.replaceAll(temp2, "");
139                 String temp_str;
140                 temp_str = temp.replaceAll(";", "");
141                 temp_str.trim();
142                 temp_str = temp_str.substring(temp_str.lastIndexOf(" ")+1);
143                 int exist_flag=0;//없으면0 있으면 1
144                 for(String s: sc[index].type_list){
145                     if(s.equals(temp_str)){
146                         exist_flag=1;
147                         break;
148                     }
149                 }
```

System Testing

– Category Partition Testing & Pairwise Testing

```
150                 if(exist_flag==0){
151                     sc[index].type_list.add(temp_str);
152                     sc[index].type_list.add(temp_str+"*");
153                     sc[index].type_list.add(temp_str+"**");
154                     sc[index].type_list.add(temp_str+"[]");
155                     sc[index].type_list.add(temp_str+"[][]");
156                 }
157                 str = str.replaceAll(temp, "");
158                 //typedef a b;같이 선언된것들에서 b를 자료형으로 type_list에 넣어줌.
159 
160             }
161             //System.out.println("구조체 검사 1 완료");
162 
163 
164         Pattern pattern = Pattern.compile(".*struct(\\w*[^}]|\\s[^}]|\\n[^}])*}.*"); // 구조체 검색을 위한 정규식
165          Matcher matcher = pattern.matcher(str);
166          while(matcher.find()){                        //구조체 끝부분의 }바로전까지찾음 따라서 이 와일문 안에서는 }부터 ;까지찾으면 그게 구조체의 이름.
167             String str2 = matcher.group();
168             if(str2.indexOf("typedef") > -1){       // 앞에 struct뿐만아니라 typedef까지 있는경우로 {부터 ;까지 찾음으로써 typedef로 바꾼 별명까지 파싱
169                 Pattern p = Pattern.compile("}\\s*.*\\s*;");
170                 Matcher m = p.matcher(str2);
171                 if(m.find()){
172                     String temp_str = m.group();
173                     temp_str = temp_str.replaceAll(" ", "");                    //공백제거부분
174                     temp_str = temp_str.substring(1, temp_str.length()-1); //variable에 최종적으로 구조체에서 파싱한 이름이 들어감.
175                     int exist_flag=0;//없으면0 있으면 1
176                     for(String s: sc[index].type_list){
177                         if(s.equals(temp_str)){
178                             exist_flag=1;
179                             break;
```

System Testing

– Category Partition Testing & Pairwise Testing

```
180                         }
181                     }
182                     if(exist_flag==0){
183                         sc[index].type_list.add(temp_str);
184                         sc[index].type_list.add(temp_str+"*");
185                         sc[index].type_list.add(temp_str+"**");
186                         sc[index].type_list.add(temp_str+"[]");
187                         sc[index].type_list.add(temp_str+"[][]");
188                     }
189                 }
190             }
191             else{                                   //typedef없이 그냥 struct만 쓴경우 구조체 자료형으로 인
192                 String temp_str = str2;
193                 if(temp_str.indexOf("=")==-1 && temp_str.indexOf(")")==-1){
194                     temp_str = temp_str.substring(temp_str.indexOf("struct")+7,temp_str.indexOf("{"));
195                     temp_str = temp_str.replaceAll(" ", "");
196                     int exist_flag=0;//없으면0 있으면 1
197                     for(String s: sc[index].type_list){
198                         if(s.equals(temp_str)){
199                             exist_flag=1;
200                             break;
201                         }
202                     }
203                     if(exist_flag==0){
204                         sc[index].type_list.add(temp_str);
205                         sc[index].type_list.add(temp_str+"*");
206                         sc[index].type_list.add(temp_str+"**");
207                         sc[index].type_list.add(temp_str+"[]");
208                         sc[index].type_list.add(temp_str+"[][]");
209                     }
```

System Testing

– Category Partition Testing & Pairwise Testing

```
                    //System.out.println("==================:"+temp_str);
                }
            }
        }

        //System.out.println("구조체 검사 4 완료");



        System.out.println("함수 검사 시작");

    // 함수 파싱
        String type_str="(int";
        StringBuffer strbuf = new StringBuffer(total_str);

        for(String s: sc[index].type_list){
            type_str = type_str+"|"+s;
        }
        type_str += ")";
        type_str = type_str.replaceAll("\\*", "\\\\*");
        type_str = type_str.replaceAll("\\[", "\\\\[");
        type_str = type_str.replaceAll("\\]", "\\\\]");
        pattern = Pattern.compile(type_str+".*(\\s)*\\(.*\\)(\\s)*\\{");
        matcher = pattern.matcher(total_str);
           while(matcher.find() ){
              if((int)(total_str.charAt(matcher.start()-1)) < 33 || (int)(total_str.charAt(matcher.start()-1)) > 122 ){
              int count=0;
             int index_func=matcher.end()-1;
             do{
```

System Testing

– Category Partition Testing & Pairwise Testing

```
240                 if((total_str.charAt(index_func)) == '{'){
241                     count++;
242                 }
243                 else if((total_str.charAt(index_func)=='}')){
244                     count--;
245                 }
246                 index_func++;
247             }
248             while(count!=0 && total_str.length() != index_func);
249             String function_all = total_str.substring(matcher.start(), index_func);
250               //System.out.println(function_all);//함수전체
251               //System.out.println("================");
252               String function_name;
253               String function_body;
254               String function_argument;
255               String str_func = total_str.substring(matcher.start(),matcher.end());
256               //System.out.println(str_func);//함수 선언부
257               function_argument = str_func.substring(str_func.indexOf('(')+1, str_func.indexOf(')'));
258               function_name = str_func.substring(str_func.indexOf(' '),str_func.indexOf('('));//함수 선언부중 매개변수나오는 괄호부분 앞까지
259               function_body = function_all.substring(function_all.indexOf('{')+1);
260               //System.out.println("func_name: "+function_name);
261               //System.out.println("func_body: "+function_body);
262               sc[index].num_of_func ++;
263               sc[index].func.add(new Function(function_name,function_body,function_argument));    //함수 이름과body를 소스코드객체의 function리스트에 추가
264               //함수 리스트에 추가 후 total_str에서 지우기.
265               String ttemp ="";
266               for(int k=0;k<function_all.length();k++)
267                   ttemp+=" ";
268               strbuf.replace(matcher.start(), index_func, ttemp);
269             }
```

System Testing

– Category Partition Testing & Pairwise Testing

```
270             }
271         System.out.println("함수개수: "+sc[index].func.size());
272         System.out.println("변수 검사 시작");
273       //변수 파싱
274         type_str="(int";
275         for(String s: sc[index].type_list){
276             type_str = type_str+"|"+s;
277         }
278         type_str += ")";
279         type_str = type_str.replaceAll("\\*", "\\\\*");
280         type_str = type_str.replaceAll("\\[", "\\\\[");
281         type_str = type_str.replaceAll("\\]", "\\\\]");
282 
283         //전역변수 먼저 파싱.
284         pattern = Pattern.compile(type_str+" ([^{;])*;"); // 변수 검색을 위한 정규식
285         matcher = pattern.matcher(strbuf);
286         while(matcher.find()){
287             String str_temp = matcher.group();
288             str_temp = str_temp.replaceAll("=.*[\\,\\;]", "");
289             if(str_temp.indexOf(')')==-1){  //변수로 인식되는부분
290                 String variable_type = str_temp.substring(0, str_temp.indexOf(' '));
291                 str_temp = str_temp.substring(str_temp.indexOf(' ')+1);
292                 String variable_name ;
293                 //System.out.println(str_temp);
294                 if(str_temp.indexOf("[")!=-1){
295                     variable_type +="[]";
296                 }
297                 str_temp = str_temp.replaceAll("\\[([^\\[\\n])*\\]", "");
298                 str_temp = str_temp.replaceAll(";", "");
299                 str_temp = str_temp.replaceAll("\\*", "");
```

System Testing

– Category Partition Testing & Pairwise Testing

```
300                 str_temp = str_temp.replaceAll("[ | ]+", "");
301                 String strArr[] = str_temp.split(",");
302                 for(String s: strArr){
303                     sc[index].var_list.add(new Variable(variable_type,s));
304                 }
305                 //System.out.println(variable_type);
306             }
307         }
308         System.out.println("전역변수개수: "+sc[index].var_list.size());
309         //함수안의 지역변수 파싱
310         for(Function func : sc[index].func){
311             String str_func_body = func.body;
312         pattern = Pattern.compile(type_str+" ([^{;])*;"); // 함수안의 지역 변수 검색을 위한 정규식
313         matcher = pattern.matcher(str_func_body);
314         //System.out.println("========func: "+func.name);
315         while(matcher.find()){                              //함수안의 지역변수를 찾아서 리스트에 넣어주는부분.
316             String str_temp = matcher.group();
317             str_temp = str_temp.replaceAll("=.*[\\,\\;]", "");
318             if(str_temp.indexOf(')')==-1){  //변수로 인식되는부분
319                 String variable_type = str_temp.substring(0, str_temp.indexOf(' '));
320                 str_temp = str_temp.substring(str_temp.indexOf(' ')+1);
321                 String variable_name ;
322                 //System.out.println(str_temp);
323                 if(str_temp.indexOf("[")!=-1){
324                     variable_type +="[]";
325                 }
326                 str_temp = str_temp.replaceAll("\\[([^\\[\\n])*\\]", "");
327                 str_temp = str_temp.replaceAll(";", "");
328                 str_temp = str_temp.replaceAll("\\*", "");
329                 str_temp = str_temp.replaceAll("[ | ]+", "");
```

System Testing

– Category Partition Testing & Pairwise Testing

```
330                 String strArr[] = str_temp.split(",");
331                 for(String s: strArr){
332                     func.var_list.add(new Variable(variable_type,s));
333                     //System.out.println("variable_name: "+s);
334                 }
335                 //System.out.println("variable_type: "+variable_type);
336             }
337         }
338
339
340
341         //함수안의 매개변수 지역변수로 파싱
342         String strArr_f[] = func.argument.split(",");
343         for(String s_f : strArr_f){
344         pattern = Pattern.compile(type_str+" ([^{;])*"); // 함수의 매개변수안의 변수 검색을 위한 정규식
345         matcher = pattern.matcher(s_f);
346         //System.out.println("========func_argument: "+s_f);
347         while(matcher.find()){
348             String str_temp = matcher.group();
349             str_temp = str_temp.replaceAll("=.*[\\,\\;]", "");
350             if(str_temp.indexOf(')')==-1){  //변수로 인식되는부분
351                 String variable_type = str_temp.substring(0, str_temp.indexOf(' '));
352                 str_temp = str_temp.substring(str_temp.indexOf(' ')+1);
353                 String variable_name ;
354                 //System.out.println(str_temp);
355                 if(str_temp.indexOf("[")!=-1){
356                     variable_type +="[]";
357                 }
358                 str_temp = str_temp.replaceAll("\\[([^\\[\\n])*\\]", "");
359                 str_temp = str_temp.replaceAll(";", "");
```

System Testing

– Category Partition Testing & Pairwise Testing

```
360                 str_temp = str_temp.replaceAll("\\*", "");
361                 str_temp = str_temp.replaceAll("[ | ]+", "");
362                 String strArr[] = str_temp.split(",");
363                 for(String s: strArr){
364                     func.var_list.add(new Variable(variable_type,s));
365                     //System.out.println("argument_variable_name: "+s);
366                 }
367                 //System.out.println("argument_variable_type: "+variable_type);
368             }
369         }
370         }
371
372         System.out.println("각 함수의 지역변수개수: "+func.var_list.size());
373         }
374
375         System.out.println("조건문 검사 시작");
376         pattern = Pattern.compile("[^else][^\\w]if[^\\w]"); //if문 검색을위한 정규식
377         matcher = pattern.matcher(total_str);
378         while(matcher.find()){                  // if문개수 검색
379             //System.out.println("if 문 발견");
380            sc[index].cond.if_number++;
381         }
382         pattern = Pattern.compile("[^\\w]else if[^\\w]"); //else_if 검색을위한 정규식
383         matcher = pattern.matcher(str);
384         while(matcher.find()){                      // else_if개수 검색
385             //System.out.println("else if 문 발견");
386            sc[index].cond.else_if_number++;
387         }
388         pattern = Pattern.compile("[^\\w]else[^\\w][^if]"); //else 검색을위한 정규식
389         matcher = pattern.matcher(str);
```

System Testing

– Category Partition Testing & Pairwise Testing

```
390         while(matcher.find()){                    // else개수 검색
391             //System.out.println("else 문 발견");
392             sc[index].cond.else_number++;
393         }
394         pattern = Pattern.compile("( |\\t)*(case|default)(\\s|\\w)*:"); //case 검색을위한 정규식
395         matcher = pattern.matcher(str);
396         while(matcher.find()){                    // case개수 검색
397             //System.out.println("case 문 발견");
398             sc[index].cond.case_number++;
399         }
400
401         System.out.println("반복문 검사 시작");
402         //for문검사시작
403         pattern = Pattern.compile("for.*(\\s)*\\(.*\\)(\\s)*\\{");
404         matcher = pattern.matcher(total_str);
405             while(matcher.find() ){
406                 if((int)(total_str.charAt(matcher.start()-1)) < 33 || (int)(total_str.charAt(matcher.start()-1)) > 122 ){
407                 int count=0;
408                 int index_for=matcher.end()-1;
409                 do{
410                     if((total_str.charAt(index_for)) == '{'){
411                         count++;
412                     }
413                     else if((total_str.charAt(index_for)=='}')){
414                         count--;
415                     }
416                     index_for++;
417                 }
418                 while(count!=0 && total_str.length() != index_for);
419                 String for_all = total_str.substring(matcher.start(), index_for);
```

System Testing

– Category Partition Testing & Pairwise Testing

```
420             String for_body = for_all.substring(for_all.indexOf('{')+1);
421             String loop_type = "for";
422             sc[index].for_number++;
423             sc[index].loop_list.add(new Loop(loop_type,for_body));
424              }
425           }
426
427
428         //while문 검사시작
429         pattern = Pattern.compile("while.*(\\s)*\\(.*\\)(\\s)*\\{");
430         matcher = pattern.matcher(total_str);
431            while(matcher.find() ){
432               if((int)(total_str.charAt(matcher.start()-1)) < 33 || (int)(total_str.charAt(matcher.start()-1)) > 122 ){
433               int count=0;
434              int index_while=matcher.end()-1;
435              do{
436                 if((total_str.charAt(index_while)) == '{'){
437                    count++;
438                 }
439                 else if((total_str.charAt(index_while)=='}')){
440                    count--;
441                 }
442                 index_while++;
443              }
444              while(count!=0 && total_str.length() != index_while);
445              String while_all = total_str.substring(matcher.start(), index_while);
446              String while_body = while_all.substring(while_all.indexOf('{')+1);
447              String loop_type = "while";
448              sc[index].while_number++;
449              sc[index].loop_list.add(new Loop(loop_type,while_body));
```

System Testing

– Category Partition Testing & Pairwise Testing

```
449             sc[index].loop_list.add(new Loop(loop_type,while_body));
450                 }
451             }
452
453             System.out.println("for문 개수: "+sc[index].for_number);
454             //System.out.println("for문 body: "+sc[index].loop_list.get(sc[index].loop_list.size()-1).loop_body);
455             System.out.println("while문 개수: "+sc[index].while_number);
456     }
457
```

System Testing – Brute Force Testing

Test Case No#	1
문제	Mac OS X에서 프로그램 실행 후 디렉터리 경로를 입력하고 Input 버튼 클릭 할 경우 경로 탐색 성공 팝업이 표시됨, 실제로는 경로를 파싱하지 못하여 Exception 발생함
원인	Window OS에서 개발해서 발생한 문제
대응	거절

Test Case No#	2
문제	1회 분석 후 잘못된 경로를 입력하여 다시 분석 시도 할 경우 모든 버튼 활성화 상태 유지, 클릭 시 비정상적 인 동작 보임
원인	버튼 비활성화 기능 구현 누락
대응	기능 구현

System Testing – Brute Force Testing

Test Case#2 - 활성/비활성 부분 수정 코드

```
btnStart.setEnabled(true);  //  Start버튼 활성화
btnShowXFile.setEnabled(true); // show x_file버튼 활성화
btnShowDetail.setEnabled(true); // show detail버튼 활성화
btnStart.setEnabled(false); //  start버튼 비활성화
btnShowXFile.setEnabled(false); //  show x_file버튼 비활성화
btnShowDetail.setEnabled(false); //  show detail버튼 비활성화
```

System Testing – Brute Force Testing

Test Case No#	3.
문제	main 함수만 존재하는 2개 의 .c 파일을 입력으로 사용 할 경우 Exception 발생 후 프로 그램이 분석 진행 중 상 태에서 정지함..
원인	함수 분석 클래스의 구현 누락.
대응	기능 구현.

Test Case No#	4.
문제	전체 소스가 주석 처리된 2개 의 .c 파일을 입력으로 사용 할 경우 Exception 발생 후 프로 그램이 분석 진행 중 상 태에서 정지함..
원인	코드를 비교하기 전에 주석을 제거하는 작업을 거치기 때문에, 전체 소스 코드가 주석으로 처리되어 있다면, 코드가 모두 지워져, 두 파일 모두 빈 파일이 되기 때문에 발생.
대응	거절.

System Testing – Brute Force Testing

Test Case No#	5.
문제	동일한 변수명 개수 카운트 할 경우 전체 변수의 개수보다 더 많은 수가 집계되어 결과로 저장됨..
원인	변수 분석 클래스의 구현 누락.
대응	기능 구현.

Test Case No#	6.
문제	변수가 선언되지 않은 함수 작성 할 경우 함수로 집계하지 않음..
원인	함수 분석 클래스의 구현 누락.
대응	기능 구현.

System Testing – Brute Force Testing

```
//전역변수에서 동일한 변수의 개수 덧셈
for(Variable v1: s1.var_list){
    for(Variable v2: s2.var_list){
        if(v1.name.equals(v2.name) && v1.data_type.equals(v2.data_type)){
            sum++;
        }
    }
}
//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)){
                    sum++;
                }
            }
        }
    }
}
```

System Testing – Brute Force Testing

Test Case No#	7
문제	각각 대문자와 소문자로 작성 된 동일한 함수를 사용할 경우 이름이 유사한 함수로 집계하지 않음
원인	함수 분석 클래스의 구현 누락
대응	기능 구현

Test Case No#	8
문제	1회 분석 후 새로운 경로를 입력하고 Input 버튼 클릭 할 경우 기존 X_File 정보가 남아 있음
원인	Show X_File 함수의 구현 누락
대응	기능 구현

System Testing – Brute Force Testing

```
88          total_str = total_str.toLowerCase(); // 소문자로 통합
```

```
// Show_X_File 버튼을 누른 경우 - action
btnShowXFile.addActionListener(new ActionListener() {

    JFrame frame_Show ;
    //= new JFrame("Show X_File");
    JPanel panel_Show ;
    //= new JPanel();
    JLabel label_Show;
    JButton btnOk = new JButton("확인");


    public void actionPerformed(ActionEvent e) {
        String name = mc.show_x_file();
        System.out.println(name);
    //  JTextArea text_Show = new JTextArea(name);
        frame_Show = new JFrame("Show X_File");
        panel_Show = new JPanel();
        label_Show = new JLabel(name);
        frame_Show.setBounds(650,350,300,150);
        frame_Show.setResizable(false);
        panel_Show.setBounds(650,350,300,150);
        panel_Show.setLayout(null);
        label_Show.setFont(new Font("함초롬돋움", Font.BOLD, 15));
        label_Show.setBounds(50,20,200,20);
```

System Testing – Brute Force Testing

Test Case No#	9.
문제	1개의 파일이 있는 디렉터리의 경로를 입력 후 Input 버튼 클릭 할 경우 경로 탐색 성공 팝업 출력 후 모든 버튼 활성 화.
원인	1개의 파일이 있는 디렉터리를 입력할 경우의 예외처리 누락.
대응	코드에 예외처리 부분 추가.

Test Case No#	10.
문제	분석 후 Show Detail 버튼 클릭 한 후 세부내용을 적은 텍스트 파일이 내용 수정 가능한 상태 로 열림.
원인	메모장으로 열기 때문에 발생.
대응	거절.

System Testing – Brute Force Testing

```
if(file_name.length==1){   //디렉토리에 c파일이 1개만 존재할경우.
    return 0;
}
```

System Testing – Brute Force Testing

Test Case No#	11.
문제	분석 후 Show Detail 버튼 클릭 시 세부내용이 점수 형태로 표시됨.
원인	기준을 백분율이 아닌 점수로 잡았기 때문에 발생.
대응	문서의 기준을 점수로 수정.

Test Case No#	12.
문제	정상적으로 동작하는 특정 C 코드 파일 두 개가 들어 있는 디렉터리의 경로를 입력 후 Input 버튼을 클릭 할 경우 Exception 발생 후 프로 그램이 정지함.
원인	프로그램을 완벽하게 구현하지못해서 발생.
대응	구현 완료.

THANK YOU