



2016 소프트웨어 모델링&분석

그놈! Clone Checker

Team4





Contents

Modified Part

Activity2051. Implement Class & Method Definitions

Activity2052. Implement Windows

Activity2053. Implement Reports

Activity2055. Write Unit Test Code



Modified Part

Modified Part

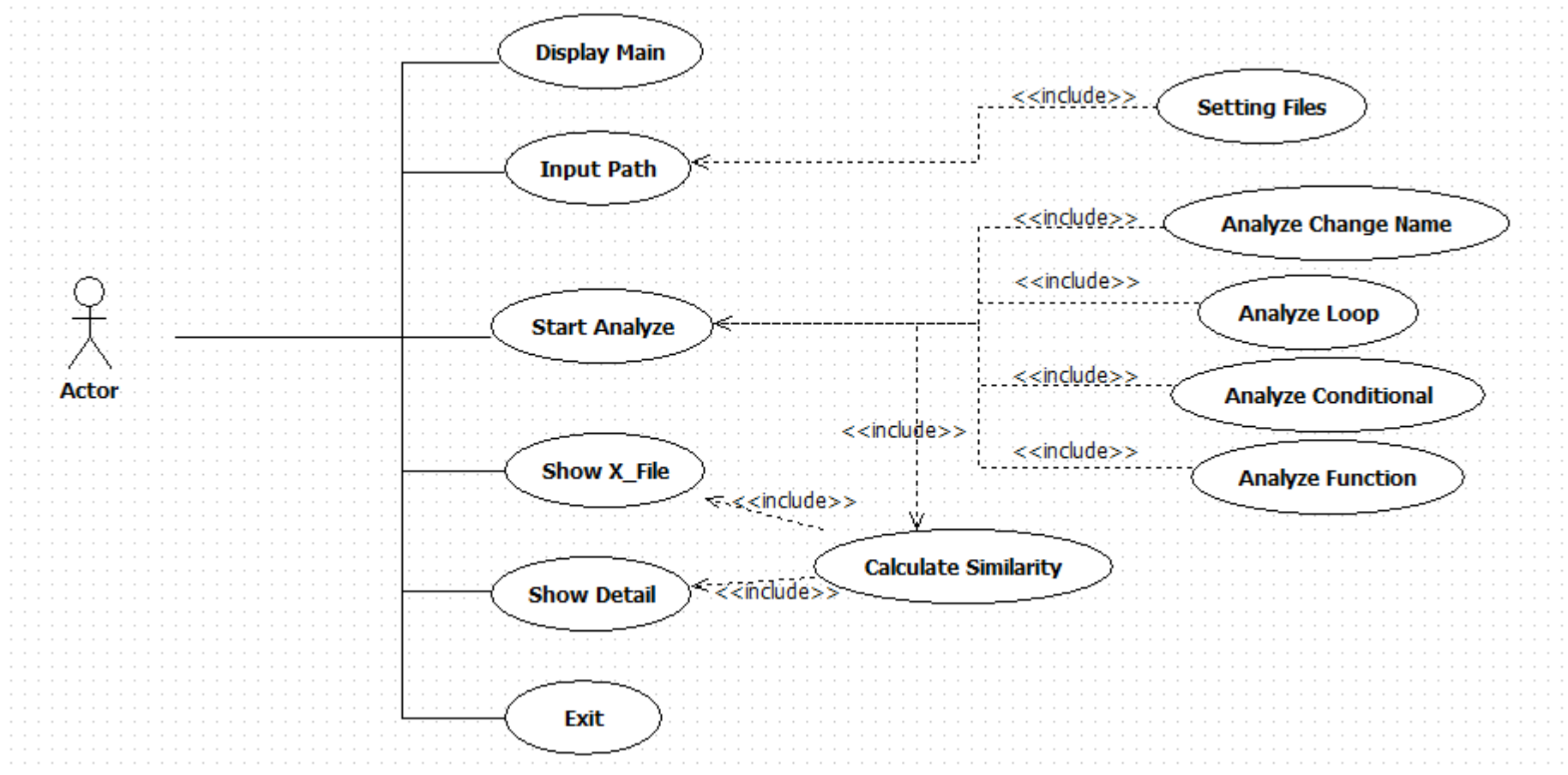
Functional Requirements

OSP Stage 1000 ver4	OSP Stage 1000 ver5
<ul style="list-style-type: none">- Display Main- Input Path- Setting Files- Start Analyze Code- Analyze Change Name- Analyze Loop- Analyze Conditional- Analyze Function- Calculate Similarity- Show X_File- Show Detail- Exit	<ul style="list-style-type: none">- Display Main- Input Path- Setting Files- Start Analyze Code- Analyze Variable- Analyze Loop- Analyze Conditional- Analyze Function- Calculate Similarity- Show X_File- Show Detail- Exit

Change Name 을 Variable로 수정

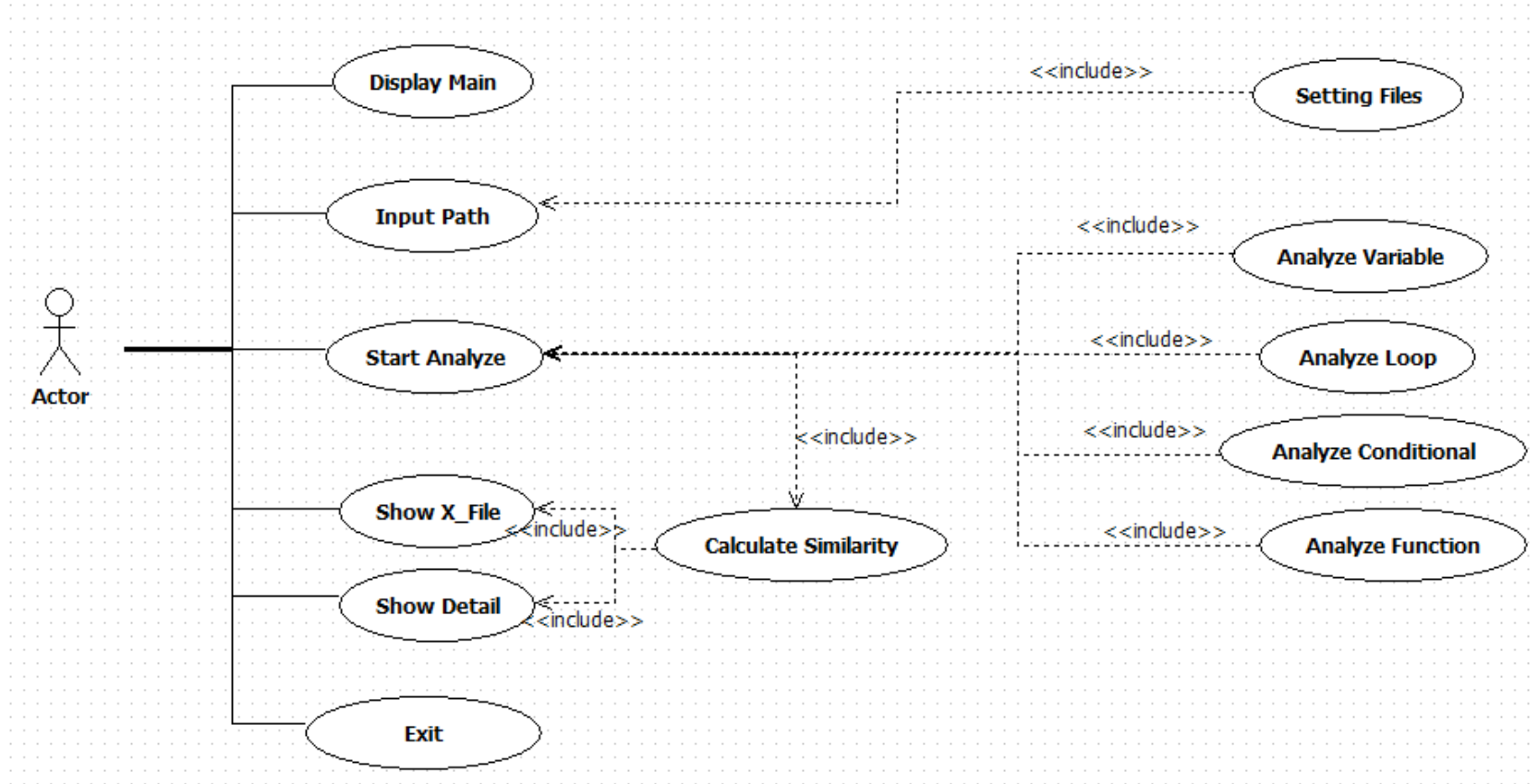
Modified Part

Use Case Diagram - 기존의 다이어그램



Modified Part

Use Case Diagram - 수정된 다이어그램



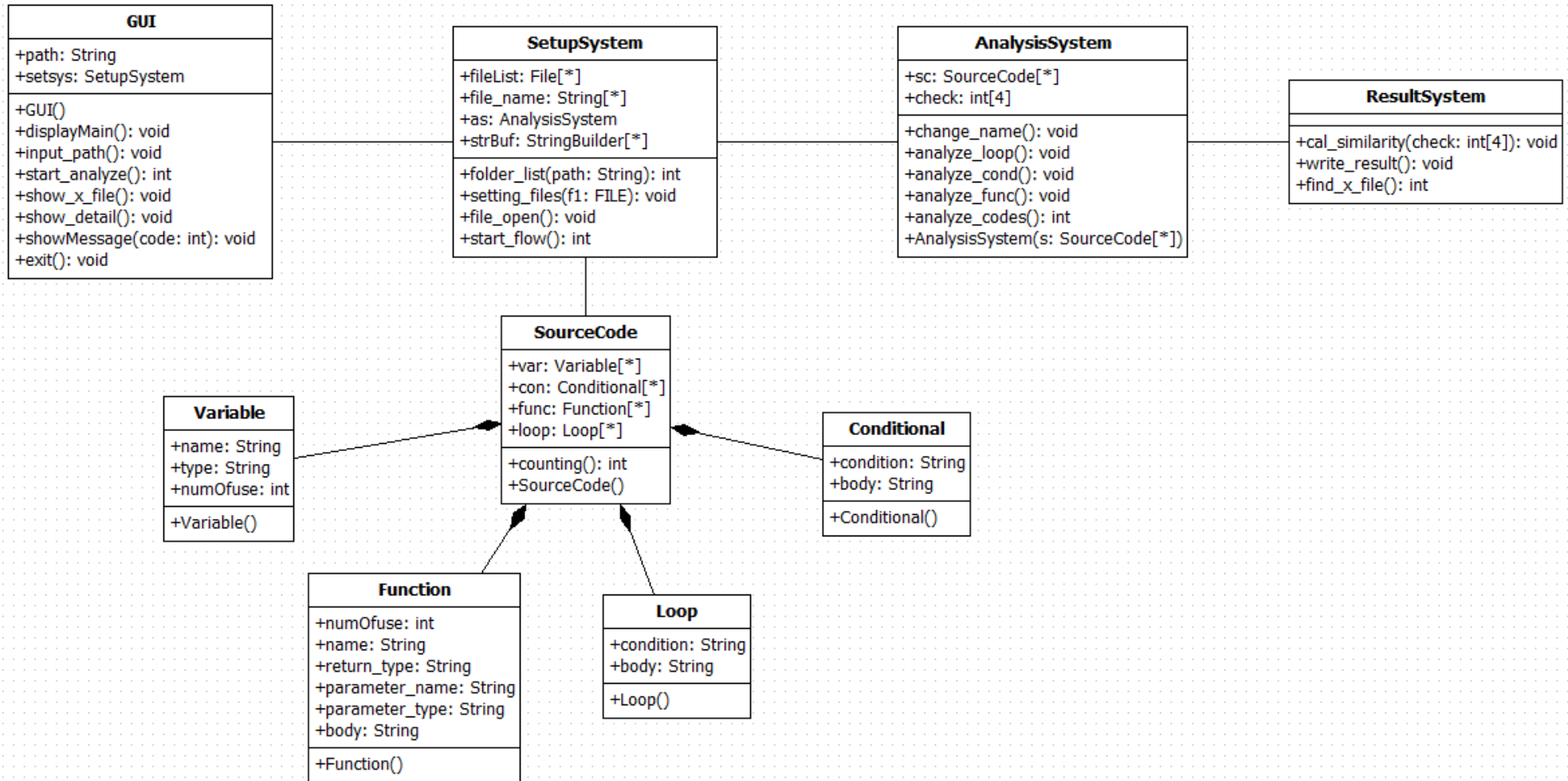
Modified Part

Use Case Diagram - 수정된 다이어그램

OSP Stage 1000 ver5	
Use-Case	Category
Display Main	Primary
Input Path	Primary
Setting Files	Primary
Start Analyze	Primary
Analyze Variable	Primary
Analyze Loop	Primary
Analyze Conditional	Primary
Analyze Function	Primary
Calculate Similarity	Primary
Show X_File	Primary
Show Detail	Primary
Exit	Primary

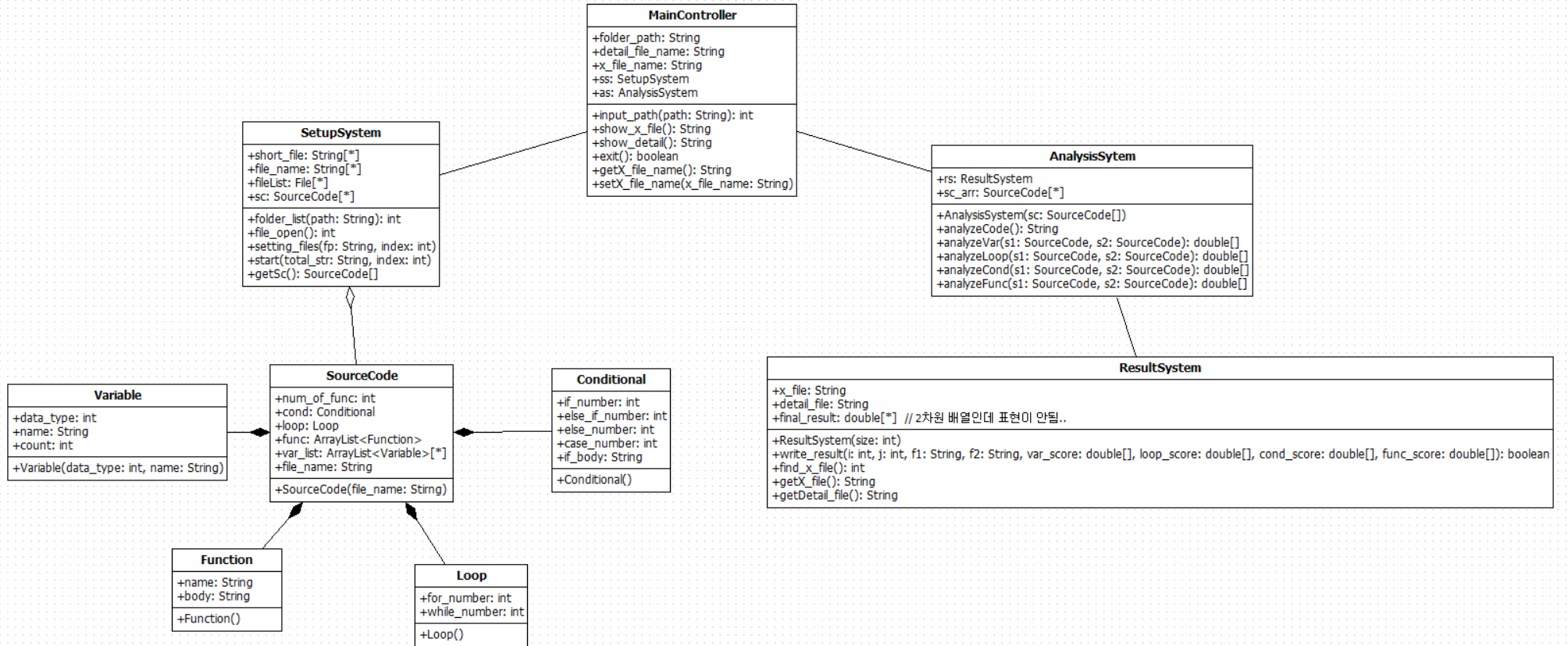
Modified Part

Domain Model (수정 전)



Modified Part

Domain Model (수정 후)



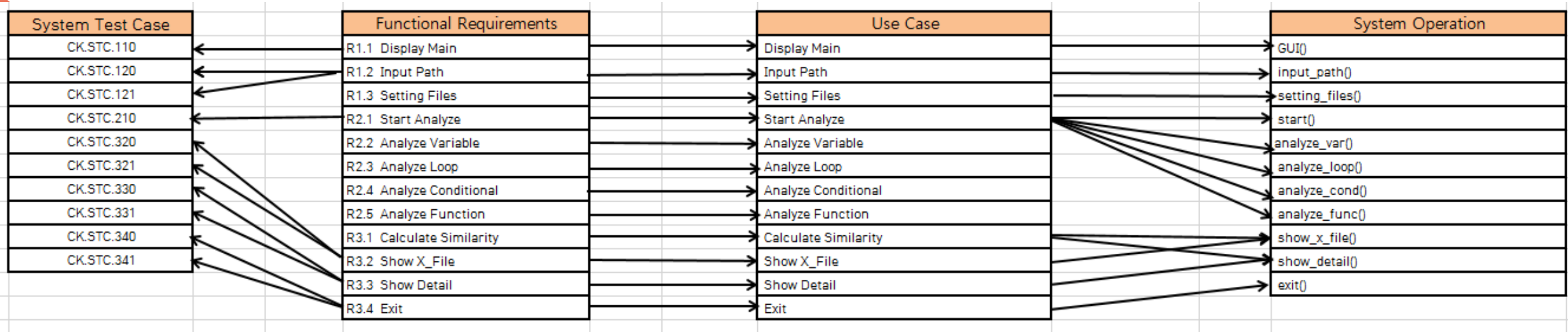
Modified Part

Traceability(수정 전)



Modified Part

Traceability(수정 후)





Activity2051. Implement Class & Method Definitions

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type	Class
Name	GUI
Purpose	사용자가 직접적으로 프로그램과 연결되는 클래스로 사용자에게 UI 를 보여줌
Overview (Class)	N/A
Cross Reference	R1.1
Exceptional Courses of Events	N/A

↵

Type	Class
Name	MainController
Purpose	GUI 와 연결되고, 내부기능과 연결되어서 프로그램을 중간에서 전체적으로 프로그램을 관리하는 클래스
Overview (Class)	N/A
Cross Reference	All
Exceptional Courses of Events	N/A

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type [Ⓜ]	Class [Ⓜ]
Name [Ⓜ]	AnalysisSystem [Ⓜ]
Purpose [Ⓜ]	정리된 소스코드의 데이터를 바탕으로 변수 유사도, 조건문 유사도, 반복문 유사도, 함수 유사도 점수를 측정하는 클래스 [Ⓜ]
Overview (Class) [Ⓜ]	N/A [Ⓜ]
Cross Reference [Ⓜ]	R2.1, R2.2, R2.3, R2.4, R2.5 [Ⓜ]
Exceptional Courses of Events [Ⓜ]	N/A [Ⓜ]

Type [Ⓜ]	Class [Ⓜ]
Name [Ⓜ]	ResultSystem [Ⓜ]
Purpose [Ⓜ]	유사도 분석의 세부 내용을 텍스트 파일로 저장하고, 본 프로그램의 목적인 그놈! 파일, 즉 중심일 것 같은 소스코드를 유추하는 클래스 [Ⓜ]
Overview (Class) [Ⓜ]	N/A [Ⓜ]
Cross Reference [Ⓜ]	R3.1, R3.2, R3.3 [Ⓜ]
Exceptional Courses of Events [Ⓜ]	N/A [Ⓜ]

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type ↴	Class ↴
Name ↴	SetupSystem ↴
Purpose ↴	코드를 비교, 분석하기 전에 코드에 있는 코드를 분석할 수 있도록 주석 제거, 소문자 통일, 소스코드 정보 저장 등의 초기작업을 하는 클래스 ↴
Overview (Class) ↴	N/A ↴
Cross Reference ↴	R1.3 ↴
Exceptional Courses of Events ↴	N/A ↴

↴

Type ↴	Class ↴
Name ↴	SourceCode ↴
Purpose ↴	소스 코드의 정보들을 모아두는 클래스 ↴
Overview (Class) ↴	N/A ↴
Cross Reference ↴	R1.3 ↴
Exceptional Courses of Events ↴	N/A ↴

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type	Class
Name	Conditional
Purpose	소스 코드에 있는 조건문에 관한 정보를 저장하는 클래스
Overview (Class)	N/A
Cross Reference	R1.3, R2.4
Exceptional Courses of Events	N/A

Type	Class
Name	Function
Purpose	소스 코드에 있는 함수에 관한 정보를 저장하는 클래스
Overview (Class)	N/A
Cross Reference	R1.3, R2.5
Exceptional Courses of Events	N/A

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type	Class
Name	Loop
Purpose	소스 코드에 있는 반복문에 관한 정보를 저장하는 클래스
Overview (Class)	N/A
Cross Reference	R1.3, R2.3
Exceptional Courses of Events	N/A

↵

Type	Class
Name	Variable
Purpose	소스 코드에 있는 변수에 관한 정보를 저장하는 클래스
Overview (Class)	N/A
Cross Reference	R1.3, R2.2
Exceptional Courses of Events	N/A

Activity2051. Implement Class & Method Definitions

1. GUI - Method Definitions

Type	Method
Name	main
Purpose	메인 메소드
Cross Reference	Function : R 1.1 Use Case : Display Main
Input (Method)	N/A
Output (Method)	N/A
Abstract operation (Method)	1. 프로그램을 실행하면 제일 먼저 실행되서 GUI 를 호출해서 UI 를 보여준다.
Exceptional Courses of Events	N/A

Type	Method
Name	GUI
Purpose	GUI 구성에 관한 메소드
Cross Reference	Function : R 1.1 Use Case : Display Main
Input (Method)	N/A
Output (Method)	N/A
Abstract operation (Method)	1. GUI 에 필요한 요소들을 나타내어 준다. 2. 버튼을 누르면 버튼에 알맞은 actionListener 가 작동하여 MainController 에 있는 함수들을 호출한다. 3. MainController 에서 반환받은 값에 따라 화면을 사용자에게 보여준다.
Exceptional Courses of Events	N/A

Activity2051. Implement Class & Method Definitions

2. Main Controller - Method Definitions

Type	Method
Name	input_path
Purpose	사용자가 입력한 폴더의 경로를 SetupSystem 으로 넘겨준다.
Cross Reference	Function : R 1.2 Use Case : Input Path
Input (Method)	path : String
Output (Method)	check_result : int
Abstract operation (Method)	1. GUI 에서 입력받은 path 를 SetupSystem 으로 넘겨준다. 2. SetupSystem 으로부터 받은 check_result 를 return 한다.
Exceptional Courses of Events	

Type	Method
Name	start_analyze
Purpose	AnalysisSystem 을 생성하여 유사도 검사를 진행한다.
Cross Reference	Function : R 2.1 Use Case : Start Analyze Code
Input (Method)	N/A
Output (Method)	a_flag : int
Abstract operation (Method)	1. AnalysisSystem 의 analyzeCode()를 실행하고 return 된 String 값을 setX_file_name 에 넘겨준다. 2. AnalysisSystem 을 통하여 ResultSystem 에 접근하여 getDetail_file()을 호출하고, return 된 String 값을 detail_file_name 에 저장한다. 3. a_flag 에 2 를 대입하여 return 함으로써 정상적으로 메소드가 끝났음을 알린다.
Exceptional Courses of Events	

Activity2051. Implement Class & Method Definitions

2. Main Controller - Method Definitions

Type	Method
Name	show_x_file
Purpose	그놈! 파일 이름 반환
Cross Reference	Function : R 3.2 Use Case : Show XFile
Input (Method)	N/A
Output (Method)	temp : String
Abstract operation (Method)	1. MainController 내부의 getX_file_name()을 호출하여 그놈! 파일의 이름을 받아온다. 3. temp 에 그놈! 파일의 이름을 저장하여 return 한다.
Exceptional Courses of Events	

Type	Method
Name	show_detail
Purpose	검사의 세부 내용이 담긴 파일 이름 반환
Cross Reference	Function : R 3.3 Use Case : Show Detail
Input (Method)	N/A
Output (Method)	detail : String
Abstract operation (Method)	1. MainController 의 detail_file_name 을 detail 에 저장한다. 2. detail 을 return 한다.
Exceptional Courses of Events	

Type	Method
Name	exit
Purpose	GUI 를 종료시킨다.
Cross Reference	Function : R 3.4 Use Case : Exit
Input (Method)	N/A
Output (Method)	okay : Boolean
Abstract operation (Method)	1. okay 에 true 를 저장하여 return 한다.
Exceptional Courses of Events	

Activity2051. Implement Class & Method Definitions

3. Analysis System - Method Definitions

Type	Method
Name	AnalysisSystem
Purpose	AnalysisSystem 의 생성자 메소드
Cross Reference	Function : R 2.1 Use Case : Start Analyze
Input (Method)	sc : SourceCode[]
Output (Method)	sc_arr : SourceCode[]
Abstract operation (Method)	1. SourceCode[] sc 를 clone()을 이용하여 SourceCode[] sc_arr 에 복사한다.
Exceptional Courses of Events	N/A

Type	Method
Name	analyzeVar
Purpose	코드에 있는 변수의 유사도를 분석한다.
Cross Reference	Function : R 2.2 Use Case : Analyze Variable
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	1. 변수 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다. 2. s1 의 전체 변수의 개수와 s2 의 전체 변수의 차를 기준으로 점수를 부여하고 이를 score[0]에 저장한다. 또한 s1 과 s2 의 전체 변수의 개수를 각각 score[4]와 score[5]에 저장한다.(참고 자료 표 #1). 3. s1 과 s2 사이에 동일한 자료형 내의 동일한 변수명의 개수를 세어 score[6]에 저장하고 기준에 의하여 점수를 부여하고 이를 2로 나눈 후 score[1]에 저장한다.(참고 자료 표 #2). 4. s1 과 s2 사이에 동일한 자료형 내의 동일한 빈도수를 갖는 변수의 개수를 세어 score[7]에 저장하고 기준에 의하여 점수를 부여하여 이를 score[2]에 저장한다.(참고 자료 표 #3). 5. 변수 유사도 검사의 최종 점수를 score[3]에 저장한다.(참고 자료 표 #4). 6. score 를 return 한다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

3. Analysis System - Method Definitions

Type	Method.
Name	analyzeCode.
Purpose	코드 2 개를 묶어서 분석을 진행한다..
Cross Reference	Function : R 2.1. Use Case : Start Analyze.
Input (Method)	N/A.
Output (Method)	result : String.
Abstract operation (Method)	1. 각각의 유사도 검사 결과 값을 담은 double 형 배열 4 개를 생성한다. . double[] var_score = new double[8]; double[] loop_score = new double[7]; double[] cond_score = new double[13]; double[] func_score = new double[6]; 2. ResultSystem 형 클래스 변수 rs 에 ResultSystem 객체를 new 해준다. 3. 이중 for 문을 통하여 소스코드를 2 개씩 묶어서 유사도 검사를 진행한다..

	4. analyzeVar()를 호출하여 변수 유사도 검사 결과를 var_score 에 저장한다. . analyzeLoop()를 호출하여 반복문 유사도 검사 결과를 loop_score 에 저장한다.. analyzeCond()를 호출하여 조건문 유사도 검사 결과를 cond_score 에 저장한다.. analyzeFunc()를 호출하여 함수 유사도 검사 결과를 func_score 에 저장한다.. 5. rs 의 write_result()를 호출하여 변수, 반복문, 조건문, 함수의 유사도 검사의 결과를 특정 파일에 저장한다.. 6. rs 의 find_x_file()을 호출하여 int 형 변수 value 에 저장한다.. 7. sc_arr 에서 value 인덱스 값의 file_name 을 String 변수 result 에 저장하여 return 한다..
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

3. Analysis System - Method Definitions

Type	Method.
Name	analyzeLoop.
Purpose	코드에 있는 반복문의 유사도를 분석한다.
Cross Reference	Function : R 2.3. Use Case : Analyze Loop.
Input (Method)	s1 : SourceCode , s2 : SourceCode.
Output (Method)	score : double[].
Abstract operation (Method)	1. 반복문 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다. 2. s1 내의 for 문의 개수와 s2 내의 for 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[0]에 저장한다. 또한 각각의 for 문의 개수를 score[3]과 score[4]에 저장한다. (참고 자료 표 #5). 3. s1 내의 while 문의 개수와 s2 내의 while 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[1]에 저장한다. 또한 각각의 for 문의 개수를 score[5]과 score[6]에 저장한다. (참고 자료 표 #6). 4. 반복문 유사도 검사의 최종 점수를 score[2]에 저장한다. (참고 자료 표 #7). 5. score 를 return 한다.
Exceptional Courses of Events	.

Type	Method.
Name	analyzeFunc.
Purpose	코드에 있는 함수의 유사도를 분석한다.
Cross Reference	Function : R 2.5. Use Case : Analyze Function.
Input (Method)	s1 : SourceCode , s2 : SourceCode.
Output (Method)	score : double[].
Abstract operation (Method)	1. 함수 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다. 2. s1 내의 함수의 개수와 s2 내의 함수의 개수의 차를 기준으로 점수를 부여하여 이를 score[0]에 저장한다. 또한 각각의 for 문의 개수를 score[1]과 score[2]에 저장한다. (참고 자료 표 #13). 3. s1 내의 모든 함수와 s2 내의 모든 함수의 이름을 비교하여 이름이 같거나 아주 유사한 함수의 개수를 세어 이를 score[3]에 저장하고, 기준에 의해 점수를 부여하여 score[4]에 저장한다. (참고 자료 표 #14). 4. 반복문 유사도 검사의 최종 점수를 score[5]에 저장한다. (참고 자료 표 #15). 5. score 를 return 한다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

3. Analysis System - Method Definitions

Type	Method
Name	analyzeCond
Purpose	코드에 있는 조건문의 유사도를 분석한다.
Cross Reference	Function : R 2.4. Use Case : Analyze Conditional
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	1. 반복문 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다. 2. s1 내의 if 문의 개수와 s2 내의 if 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[0]에 저장한다. 또한 각각의 if 문의 개수를 score[5]과 score[6]에 저장한다. (참고 자료 표 #8). 3. s1 내의 else 문의 개수와 s2 내의 else 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[1]에 저장한다. 또한 각각의 else 문의 개수를 score[7]과 score[8]에 저장한다. (참고 자료 표 #9).

	4. s1 내의 else if 문의 개수와 s2 내의 else if 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[2]에 저장한다. 또한 각각의 else if 문의 개수를 score[9]과 score[10]에 저장한다. (참고 자료 표 #10). 5. s1 내의 case 문의 개수와 s2 내의 case 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[3]에 저장한다. 또한 각각의 case 문의 개수를 score[11]과 score[12]에 저장한다. (참고 자료 표 #11). 6. 조건문 유사도 검사의 최종 점수를 score[4]에 저장한다. (참고 자료 표 #12). 7. score 를 return 한다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

4. Setup System - Method Definitions

Type	Method
Name	setting_files
Purpose	분석하기에 앞서 분석할 파일을 주석 제거 및 대소문자 통합해준다.
Cross Reference	Function : R1.3. Use Case : Setting Files.
Input (Method)	fp : String , index : int
Output (Method)	N/A.
Abstract operation (Method)	1. fp 를 경로로 가진 파일을 열어 파일의 주석을 삭제한다. 2. 파일의 대소문자를 통합한다. 3. start()를 호출한다.
Exceptional Courses of Events	N/A.

Type	Method
Name	start
Purpose	분석하기에 앞서 필요한 작업을 하고, 객체를 생성한다.
Cross Reference	Function : R1.3. Use Case : Start Analyze Code.
Input (Method)	total_str : String , index : int.
Output (Method)	N/A.
Abstract operation (Method)	1. 먼저 구조체들을 문법에 맞춰서 파싱해준다. 각 구조체들을 자료형으로 인식할 수 있도록 리스트를 통해 저장해준다. 2. 조건문,반복문,함수,변수들을 파싱해 Conditional, Loop, Function, Variable 클래스변수들을 알맞은 값으로 초기화 시켜준다.(이 때,Variable 객체는 자료형에따라 어느 ArrayList 에 들어가는지 알아서 넣어준다). 3. 이 값들을 Conditional, Loop, Function, Variable 객체를 생성해 SourceCode 객체안에 알맞은 멤버변수에 넣어준다.
Exceptional Courses of Events	N/A.

Activity2051. Implement Class & Method Definitions

4. Setup System - Method Definitions

Type	Method
Name	folder_list
Purpose	MainController가 넘겨준 path를 가지고, 입력된 경로에 있는 파일의 리스트를 가져온다.
Cross Reference	Function : R1.3. Use Case : Setting Files.
Input (Method)	path : String.
Output (Method)	flag : int.
Abstract operation (Method)	1. path가 유효한지 확인한다. 2. path 유효한 경우 (flag = 0) path의 파일 리스트를 fileList에 받아온다. 3. fileList의 길이를 토대로 String형 배열 file_name과 short_file을 생성한다. 4. for문을 통하여 short_file에는 파일의 이름만을, file_name에는 절대경로를 포함한 파일의 이름을 저장한다. 5. file_open()을 호출하여 return 값을 flag에 받는다. 6. flag를 return 한다.
Exceptional Courses of Events	폴더 경로가 유효하지 않은 경우 (flag = 1) flag를 return 한다.

Type	Method
Name	file_open
Purpose	파일 리스트에 들어있는 파일을 하나씩 열어준다.
Cross Reference	Function : R1.3. Use Case : Setting Files.
Input (Method)	N/A.
Output (Method)	f_o_flag : int.
Abstract operation (Method)	1. 폴더내에 .c 파일만 있는 경우 (f_o_flag = 2) SourceCode 객체배열 sc를 만든다. 2. for문을 이용하여 sc의 인덱스마다 객체를 생성한 후 setting_files()를 호출한다.
Exceptional Courses of Events	폴더내에 .c 파일만 있지 않은 경우 (f_o_flag = 1) f_o_flag를 return 한다.

Activity2051. Implement Class & Method Definitions

5. Result System - Method Definitions

Type	Method
Name	write_result
Purpose	분석결과를 메모장에 저장한다.
Cross Reference	Function : R3.1, R3.3. Use Case : Calculate Similarity
Input (Method)	N/A
Output (Method)	w_flag : boolean
Abstract operation (Method)	1. boolean 형 변수 w_flag 생성. 2. 생성자에서 미리 정해 놓은 detail_file 을 открывает. 3. detail_file 에 변수 유사도 검사의 결과를 저장한다. 4. detail_file 에 반복문 유사도 검사의 결과를 저장한다.

	5. detail_file 에 조건문 유사도 검사의 결과를 저장한다. 6. detail_file 에 함수 유사도 검사의 결과를 저장한다. 7. 각각의 점수를 바탕으로 최종 점수를 계산하여 final_result 에 저장한다. (참고 자료 표 #16) 그 후 detail_file 에 최종 점수를 저장한다. 8. w_flag에 true를 저장한다. 9. w_flag를 return 한다.
Exceptional Courses of Events	파일 입출력에서 오류가 발생 하면 (w_flag = false) w_flag를 return 한다.

Activity2051. Implement Class & Method Definitions

5. Result System - Method Definitions

Type	Method
Name	find_x_file
Purpose	분석결과를 토대로 가장 핵심이 되는 것 같은 파일을 찾는다.
Cross Reference	Function : R3.1, R3.2. Use Case : Calculate Similarity.
Input (Method)	N/A
Output (Method)	result_index : int
Abstract operation (Method)	1. 전체 파일의 개수를 담은 int 형 변수 length 생성. 2. 기준 점수를 넘는 검사 조합을 체크할 int[] 형 변수 check_point 생성. 3. check_point 의 한 행에서 기준 점수를 넘는 검사의 개수를 저장할 int[] 형 변수 row_check 생성. 4. 몇번 째 파일이 그놈! 파일인지 인덱스를 담은 int 형 변수 result_index 생성.

	5. while 문을 이용하고 for 문을 사용해 기준 점수가 넘는 검사 인덱스의 check_point 에 1 을 저장하고, row_check 에서 같은 행에 1 을 더해준다. 6. row_check 에서 최대 값을 갖는 행을 찾기 위하여 for 문을 사용해서 검사를 한다. 이 때 동일한 값을 가지고 있어 오래 동안 하나를 정할 수가 없다면 찾는 것을 포기한다. 또한 하나로 정해지는 경우에 그 때의 행의 인덱스를 result_index 에 넣어 return 한다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

5. Result System - Method Definitions

Type	Method
Name	getX_file
Purpose	x_file 을 return 한다.
Cross Reference	N/A
Input (Method)	N/A
Output (Method)	x_file : String
Abstract operation (Method)	1. x_file 을 return 한다.
Exceptional Courses of Events	.

↵

Type	Method
Name	getDetail_file
Purpose	detail_file 을 return 한다.
Cross Reference	N/A
Input (Method)	N/A
Output (Method)	detail_file : String
Abstract operation (Method)	1. detail_file 을 return 한다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

6. Source Code - Method Definitions

Type.	Method.	.
Name.	SourceCode.	.
Purpose.	안에 들어있는 변수들을 초기화 한다.	.
Cross Reference.	Function : R1.3, R2.2, R2.3, R2.4, R2.5. Use Case : Start Analyze.	.
Input (Method).	file_name : String.	.
Output (Method).	N/A.	.
Abstract operation (Method).	1.클래스 변수들을 객체화 시켜준다..	.
Exceptional Courses of Events.	.	.

Activity2051. Implement Class & Method Definitions

7. Variable - Method Definitions

Type.	Method.
Name.	Variable.
Purpose.	안에 들어있는 변수들을 초기화 한다.
Cross Reference.	Function : R2.2. Use Case : Analyze Variable.
Input (Method).	data_type : int , name : String.
Output (Method).	N/A.
Abstract operation (Method).	1. 메소드로 들어온 매개변수로 클래스 변수들을 초기화 시켜준다. 2. 클래스 변수들을 객체화 시켜준다.
Exceptional Courses of Events.	.

Activity2051. Implement Class & Method Definitions

8. Loop - Method Definitions

Type.	Method.
Name.	Loop.
Purpose.	안에 들어있는 변수들을 초기화 한다.
Cross Reference.	Function : R2.3. Use Case : Analyze Loop.
Input (Method).	N/A.
Output (Method).	N/A.
Abstract operation (Method).	1. 메소드로 들어온 매개변수로 클래스 변수들을 초기화 시켜준다. 2. 클래스 변수들을 객체화 시켜준다.
Exceptional Courses of Events.	.

Activity2051. Implement Class & Method Definitions

9. Conditional - Method Definitions

Type	Method
Name	Conditional
Purpose	안에 들어있는 변수들을 초기화 한다.
Cross Reference	Function : R2.4. Use Case : Analyze Conditional.
Input (Method)	N/A.
Output (Method)	N/A.
Abstract operation (Method)	1. 메소드로 들어온 매개변수로 클래스 변수들을 초기화 시켜준다. 2. 클래스 변수들을 객체화 시켜준다.
Exceptional Courses of Events	.

Activity2051. Implement Class & Method Definitions

10. Function - Method Definitions

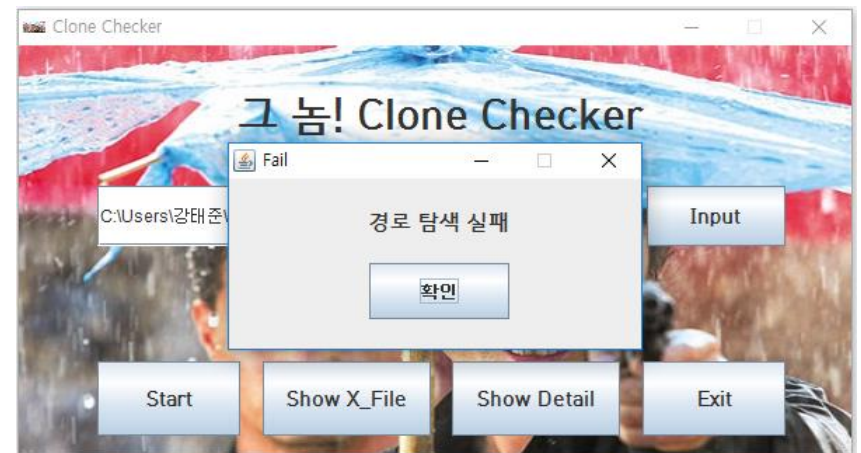
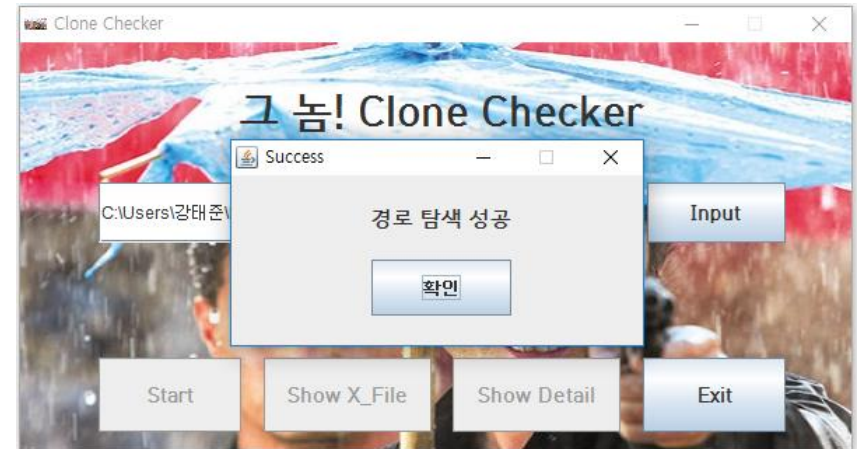
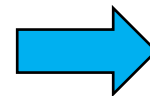
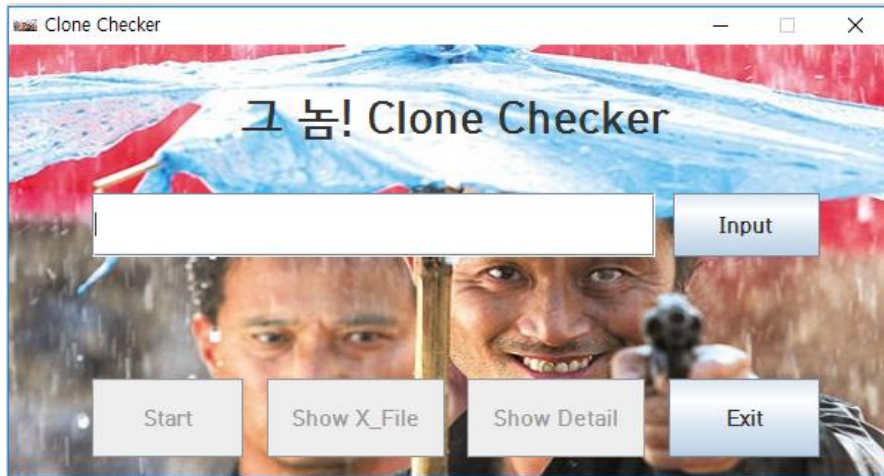
Type	Method
Name	Function
Purpose	안에 들어있는 변수들을 초기화 한다.
Cross Reference	Function : R2.5. Use Case : Analyze Function.
Input (Method)	N/A.
Output (Method)	N/A.
Abstract operation (Method)	1. 메소드로 들어온 매개변수로 클래스 변수들을 초기화 시켜준다. 2. 클래스 변수들을 객체화 시켜준다.
Exceptional Courses of Events	.



Activity2052. Implement Windows

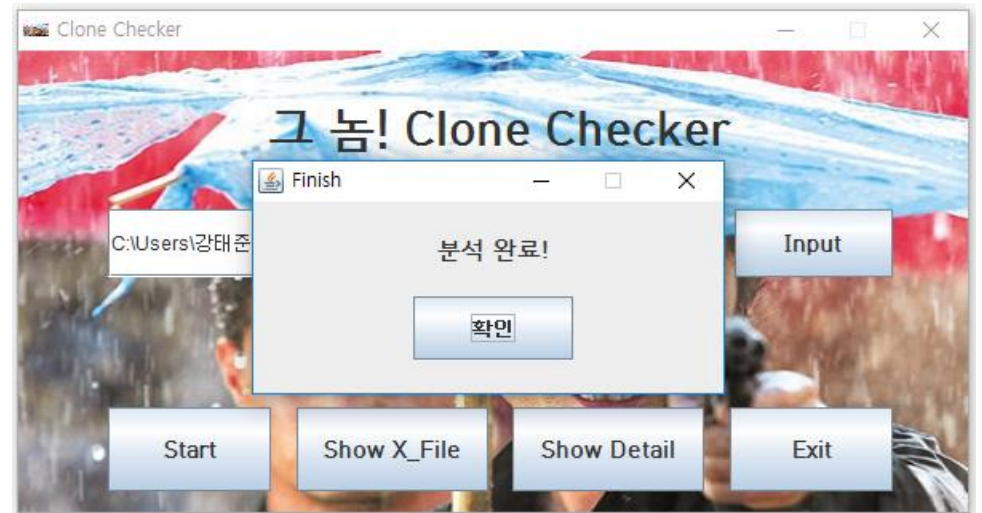
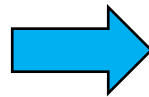
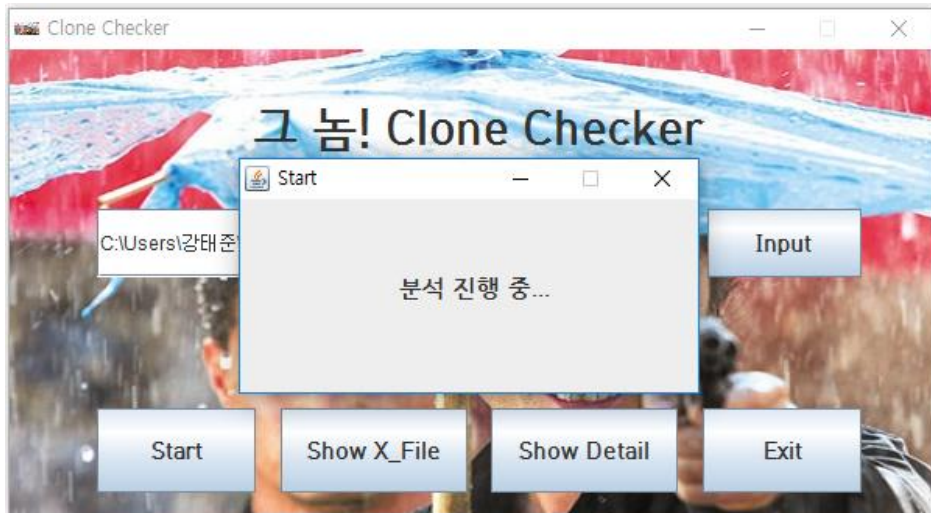
Activity2052. Implement Windows

Name	Input
Responsibilities	Textbox에 분석할 파일이 들어있는 폴더의 경로를 입력하고, 버튼을 누른다.
Type	GUI
Cross References	Function : R1.2. Use Case : Input Path.
Notes	1. 경로를 잘 입력한 경우 - 경로 탐색 성공. 2. 경로를 잘 못 입력한 경우 - 경로 탐색 실패.
Pre-Conditions	UI 초기화면.
Post-Conditions	경로탐색 성공/실패에 대한 팝업창.



Activity2052. Implement Windows

Name	Start
Responsibilities	Start 버튼을 누른다.
Type	GUI
Cross References	Function : R1.1. Use Case : Start Analyze Code
Notes	분석진행중 이라는 팝업창을 띄웠다가, 분석이 완료되면 분석완료라는 팝업창을 띄운다.
Pre-Conditions	UI 초기화면
Post-Conditions	분석진행중/분석완료 팝업창



Activity2052. Implement Windows

Name	Show_XFile
Responsibilities	Show_XFile 버튼을 누른다.
Type	GUI
Cross References	Function : R3.2. Use Case : Show XFile.
Notes	XFile 을 보여준다.
Pre-Conditions	UI 초기화면.
Post-Conditions	XFile 을 보여주는 팝업창.



Activity2052. Implement Windows

Name	Show_Detail.
Responsibilities	Show_Detail 버튼을 누른다..
Type	GUI.
Cross References	Function : R3.3. Use Case : Show Detail.
Notes	저장되어 있는 분석 세부내용을 메모장을 열어서 보여준다..
Pre-Conditions	UI 초기화면.
Post-Conditions	메모장.



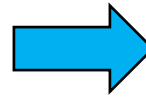
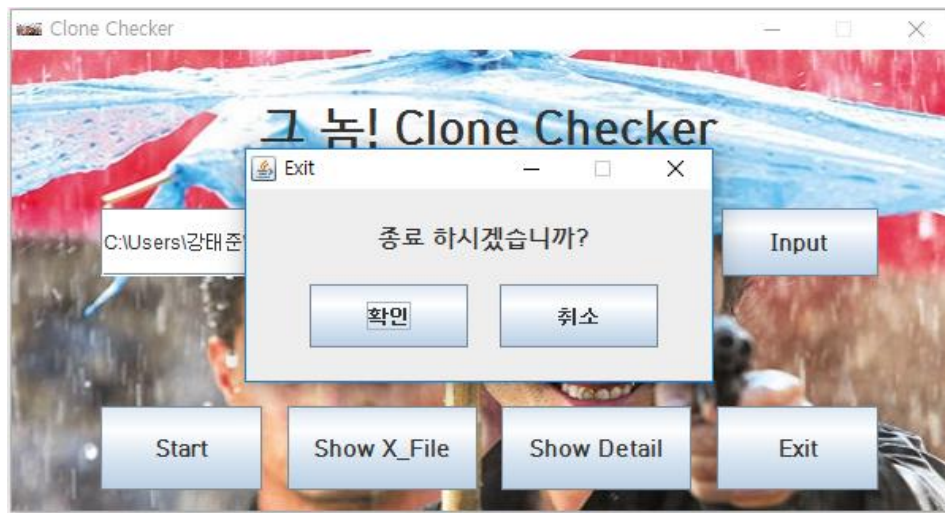
```

20160519040811 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
파일 AbsoCompare.c과 파일 AddToNum.c 유사도 검사 결과
<변수 유사도 검사>
1) 전체 변수 개수 검사 => 파일 AbsoCompare.c의 전체 변수 개수 : 2.0개 파일 AddToNum.c의 전체 변수 개수 : 3.0개
.. 검사 결과 => 90.0점
2) 동일한 변수명 개수 검사 => 동일한 변수명의 개수 : 0.0
.. 검사 결과 => 0.0점
3) 동일한 빈도수 변수 개수 검사 => 동일한 빈도수 변수의 개수 : 4.0
.. 검사 결과 => 400.0점
.. 변수 유사도 검사 최종 점수 => 178.0점
<반복문 유사도 검사>
1) for문 개수 검사 => 파일 AbsoCompare.c의 for문 개수 : 0.0개 파일 AddToNum.c의 for문 개수 : 1.0개
.. 검사 결과 => 90.0점
2) while문 개수 검사 => 파일 AbsoCompare.c의 while문 개수 : 0.0개 파일 AddToNum.c의 while문 개수 : 0.0개
.. 검사 결과 => 100.0점
.. 반복문 유사도 검사 최종 점수 => 95.0점
<조건문 유사도 검사>
1) if문 개수 검사 => 파일 AbsoCompare.c의 if문 개수 : 2.0개 파일 AddToNum.c의 if문 개수 : 0.0개
.. 검사 결과 => 80.0점
2) else문 개수 검사 => 파일 AbsoCompare.c의 else문 개수 : 2.0개 파일 AddToNum.c의 else문 개수 : 0.0개
.. 검사 결과 => 80.0점
3) else if문 개수 검사 => 파일 AbsoCompare.c의 else if문 개수 : 0.0개 파일 AddToNum.c의 else if문 개수 : 0.0개
.. 검사 결과 => 100.0점
4) case문 개수 검사 => 파일 AbsoCompare.c의 case문 개수 : 0.0개 파일 AddToNum.c의 case문 개수 : 0.0개
.. 검사 결과 => 100.0점
.. 조건문 유사도 검사 최종 점수 => 90.0점
<함수 유사도 검사>
1) 전체 함수 개수 검사 => 파일 AbsoCompare.c의 전체 함수 개수 : 3.0개 파일 AddToNum.c의 전체 함수 개수 : 1.0개
.. 검사 결과 => 50.0점
2) 함수 이름 유사 검사 => 유사한 이름의 함수 개수 : 0.0개
.. 검사 결과 => 0.0점
.. 함수 유사도 검사 최종 점수 => 0.6000000000000001점
.. 최종 유사도 검사 결과 => 108.32000000000001점
*****
파일 AbsoCompare.c과 파일 AddToTotal.c 유사도 검사 결과
<변수 유사도 검사>
1) 전체 변수 개수 검사 => 파일 AbsoCompare.c의 전체 변수 개수 : 2.0개 파일 AddToTotal.c의 전체 변수 개수 : 3.0개
.. 검사 결과 => 90.0점
2) 동일한 변수명 개수 검사 => 동일한 변수명의 개수 : 0.0
.. 검사 결과 => 0.0점
3) 동일한 빈도수 변수 개수 검사 => 동일한 빈도수 변수의 개수 : 4.0
.. 검사 결과 => 400.0점
.. 변수 유사도 검사 최종 점수 => 178.0점
<반복문 유사도 검사>

```

Activity2052. Implement Windows

Name	Exit
Responsibilities	Exit 버튼을 누른다.
Type	GUI
Cross References	Function : R3.4. Use Case : Exit.
Notes	종료를 확인하는 팝업창을 띄운다. 1. 확인 - 종료한다. 2. 취소 - 메인화면으로 돌아간다.
Pre-Conditions	UI 초기화면
Post-Conditions	종료를 확인하는 팝업창





Activity2055. Write Unit Test Code

Activity2055. Write Unit Test Code

Analysis System Test

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

import org.junit.Test;

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");
        mc.start_analyze();

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

Activity2055. Write Unit Test Code

Result System Test

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

import org.junit.Test;

public class ResultSystemTest {

    @Test
    //
    public void testWrite_result() {
        MainController mc = new MainController();
        mc.input_path("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd");
        mc.start_analyze();
        double[] var_score = new double[8]; // 1) 2) 3) 최종 1st전체 2nd전체 변수명동일 빈도수동일
        double[] loop_score = new double[7];
        double[] cond_score = new double[13];
        double[] func_score = new double[6];
        for(int i=0; i<mc.as.sc_arr.length; i++) {
            for(int j=0; j<mc.as.sc_arr.length; j++) {
                if(i<j) {
                    var_score = mc.as.analyzeVar(mc.as.sc_arr[i], mc.as.sc_arr[j]);
                    loop_score = mc.as.analyzeLoop(mc.as.sc_arr[i], mc.as.sc_arr[j]);
                    cond_score = mc.as.analyzeCond(mc.as.sc_arr[i], mc.as.sc_arr[j]);
                    func_score = mc.as.analyzeFunc(mc.as.sc_arr[i], mc.as.sc_arr[j]);
                    //write_result에서 성공적으로 전부다 기록이 되면 true를 반환하는지 확인.
                    assertTrue(mc.as.rs.write_result(i, j, mc.as.sc_arr[i].file_name,
                        mc.as.sc_arr[j].file_name, var_score, loop_score, cond_score, func_score));
                }
            }
        }
    }
}
```

```
@Test
// 분석한 결과가 최종 유사도 배열에 저장되지 않았을때, -2가 반환되는지 확인
public void testFind_x_file() {
    ResultSystem rc = new ResultSystem(2);
    assertEquals(-2 ,rc.find_x_file());
}
}
```

Activity2055. Write Unit Test Code

Setup System Test

```
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"));
        //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"));
        //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");
        assertEquals("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\FS_Function.c",ss.file_name[0]);
    }

    @Test
    public void testFile_open() {

        SetupSystem ss = new SetupSystem();
        //2-1 디렉토리내에 ".c"파일들만 존재할 때, flag가 2를 반환하는지 확인
        ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd");
        assertEquals(2,ss.file_open());
        //2-2 디렉토리내에 ".c"말고 다른파일이 존재할때, flag가 1을 반환하는지 확인
        ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient");
        assertEquals(1,ss.file_open());
    }
}
```

Activity2055. Write Unit Test Code

Setup System Test

```
@Test
public void testStart() {
    SetupSystem ss = new SetupSystem();
    //3-1 Start() 실행시 Variable객체가 제대로 리스트에 들어가있는지 확인
    ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd");
    assertNotNull(ss.sc[0].var_list[0].get(0));
    //3-2 Start() 실행시 i번째 소스코드의 함수 개수와 sc[i].num_of_func 같은지 확인
    assertEquals(7,ss.sc[0].num_of_func); //첫번째 소스코드, 즉, "FS_Function.c"안에는 7개의 함수가있음.
}

@Test
public void testGetSc() {
    SetupSystem ss = new SetupSystem();
    //4-1 GetSc() 실행시 Sc[]가 제대로 반환되는지 확인
    ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd");
    assertEquals(ss.sc,ss.getSc());
}
}
```

[참고자료] 검사 항목 별 기준 - 변수

1) 전체 변수 개수 비교 (20%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
5	50

2) 동일한 자료형 내에서 동일한 이름을 갖는 변수 개수 세기 (40%)

$\frac{\text{동일한 이름의 변수 개수의 총합}}{\text{소스 코드의 전체 변수 개수}} * 100$ (%) 를 계산하여 50% 이상일 경우 각각의 소스 코드에 점수를 부여하고 그 둘의 평균을 최종적인 점수로 정한다.

비율	소스코드 1 유사도 점수	소스코드 2 유사도 점수
90 ~ 100	100	100
80 ~ 90	90	90
70 ~ 80	80	80
60 ~ 70	70	70
50 ~ 60	60	60

3) 동일한 자료형 내에서 동일한 빈도수를 갖는 변수 개수 세기(40%)

같은 자료형 내에서 변수가 빈도수가 동일하다면 개수가 1씩 증가

최종적인 개수 * 100 점 으로 점수 결정

4) 최종적인 변수 유사도 계산

(전체 변수 개수 비교 점수) * 0.2 + (동일한 자료형 내에서 동일한 이름을 갖는 변수 개수) * 0.4 + (동일한 자료형 내에서 동일한 이름을 갖는 변수 개수) * 0.4

[참고자료] 검사 항목 별 기준 - 반복문

1) for 문의 개수 비교 (50%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

2) while 문의 개수 비교 (50%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

3) 최종적인 반복문 유사도 계산

(for 문 개수 비교 점수) * 0.5 + (while 문 개수 비교 점수) * 0.5

[참고자료] 검사 항목 별 기준 - 조건문

1) if 문의 개수 비교 (30%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

2) else 문의 개수 비교 (20%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

3) else if 문의 개수 비교 (25%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

4) case 문의 개수 비교 (25%)

차이	유사도 점수
0	100
1	90
2	80
3	70
4	60
그 이하	50

5) 최종적인 조건문 유사도 계산

$$(if문 점수) * 0.3 + (else문 점수) * 0.2 + (else if문 점수) * 0.25 + (case문 점수) * 0.25$$

[참고자료] 검사 항목 별 기준 - 함수

1) 전체 함수 개수 비교 (20%)

차이	유사도 점수
0	100
1	70
2	50
그 이하	30

2) 함수 이름 비교 (50%)

동일한 이름을 갖거나 1글자 차이가 나는 함수의 개수를 센다.

(총 개수) * 100 점

3) 최종적인 함수 유사도 계산

(전체 함수 개수 비교 점수) * 0.2 + (함수 이름 비교 점수) * 0.5

[참고자료] 검사 항목 별 기준 - 전체

(변수 검사) * 0.4 + (반복문 검사) * 0.2 + (조건문 검사) * 0.2 + (함수 검사) * 0.2



Contents

Activity2061. Unit Testing

Activity2063. System Testing

Activity2064. Performance Testing

Activity2067. Testing Traceability Analysis

Activity2061. Unit Testing

Class = AnalysisSystem

1	Name = testAnalyzeCode()	
1-1	assertTrue(mc.as.analyzeCode().substring(mc.as.analyzeCode().length()-2).equals(".c")): 그놈 파일(X-File)이 ".c"로 끝나는지 확인	Pass

Class = ResultSystem

2	Name = testWrite_result()	
2-1	assertTrue(mc.as.rs.write_result(i, j, mc.as.sc_arr[i].file_name, mc.as.sc_arr[j].file_name, var_score, loop_score, cond_score, func_score)): write_result 에서 성공적으로 전부다 기록이 되면 true 를 반환하는지 확인	Pass
3	Name = testFind_x_file()	
3-1	assertEquals(-2 ,rc.find_x_file()): 분석한 결과가 최종 유사도 배열에 저장되지 않았을때, -2 가 반환되는지 확인	

Class = SetupSystem

4.	Name = testFolder_list().	
4-1.	assertEquals(2,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd")): .c" 파일만 들어있는 폴더의 경로를 입력했을때, 2 가 반환되는지 확인.	
4-2.	assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MServer")): .c" 파일만 들어있지 않은 폴더의 경로를 입력했을때, 1 이 반환되는지 확인.	
4-3.	assertEquals(1,ss.folder_list("is_not_directory")): 경로형태가 아닌 이상한 문자를 입력했을때, 1 이 반환되는지 확인.	
4-4.	assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\IS_NOT_DIRECTORY")): 없는 경로를 입력했을때, 1 이 반환되는지 확인.	
4-5.	assertEquals("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd\\FS_Function.c",ss.file_name[0]): file_name 에 경로 디렉토리에 있는 파일이 제대로 들어가는지 확인.	Pass.
5.	Name = testFile_open().	
5-1.	assertEquals(2,ss.file_open()): 디렉토리내에 ".c"파일들만 존재할 때, flag 가 2 를 반환하는지 확인.	
5-2.	assertEquals(1,ss.file_open()): 디렉토리내에 ".c"말고 다른파일이 존재할때, flag 가 1 을 반환하는지 확인.	
6.	Name = testStart().	
6-1.	assertNotNull(ss.sc[0].var_list[0].get(0)): Start() 실행시 Variable 객체가 제대로 리스트에 들어가있는지 확인.	

Activity2061. Unit Testing

Class = SetupSystem

4	Name = testFolder_list()
4-1	assertEquals(2,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MClient\\asd")); ".c" 파일만 들어있는 폴더의 경로를 입력했을때, 2 가 반환되는지 확인
4-2	assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명재_R02_V01\\MServer")); ".c" 파일만 들어있지 않은 폴더의 경로를 입력했을때, 1 이 반환되는지 확인
4-3	assertEquals(1,ss.folder_list("is_not_directory")); 경로형태가 아닌 이상한 문자를 입력했을때, 1 이 반환되는지 확인
4-4	assertEquals(1,ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\IS_NOT_DIRECTORY")); 없는 경로를 입력했을때, 1 이 반환되는지 확인

4-5	assertEquals("C:\\Users\\서우\\Desktop\\반바지\\NP4039_20111377_이명재_R02_V01\\MClient\\asd\\FS_Function.c",ss.file_name[0]); file_name 에 경로 디렉토리에 있는 파일이 제대로 들어가는지 확인	Pass
5	Name = testFile_open()	
5-1	assertEquals(2,ss.file_open()); 디렉토리내에 ".c"파일들만 존재할 때, flag 가 2 를 반환하는지 확인	
5-2	assertEquals(1,ss.file_open()); 디렉토리내에 ".c"말고 다른파일이 존재할때, flag 가 1 을 반환하는지 확인	
6	Name = testStart()	
6-1	assertNotNull(ss.sc[0].var_list[0].get(0)); Start() 실행시 Variable 객체가 제대로 리스트에 들어가있는지 확인	

6-2	assertEquals(7,ss.sc[0].num_of_func); //첫번째 소스코드,즉,"FS_Function.c"안에는 7 개의 함수가있음. Start() 실행시 i 번째 소스코드의 함수 개수와 sc[i].num_of_func 같은지 확인
7	Name = testGetSc()
7-1	assertSame(ss.sc,ss.getSc()); GetSc() 실행시 Sc[]가 제대로 반환되는지 확인

Activity2063. System Testing

Test Number	Test 항목	Description	Use Case	System Function
1-1	UI	프로그램의 시작	1. Display Main	R 1.1
2-1	경로 탐색	존재하지 않는 경로를 입력한 경우	2. Input Path	R 1.2
2-2	경로 탐색	폴더의 경로를 입력하고, 그 폴더 안에 .c 파일외의 파일이 들어있는 경우	2. Input Path	R 1.2
3-1	분석 작동	경로 입력을 성공하고 Start 를 누른 경우	3. Start Analyze	R 2.1
4-1	버튼 비활성	분석이 진행되는 도중 버튼을 누르는 경우	4. Show X_File	R 3.2
4-2	그놈 파일 찾기	똑같은 소스 5 개로 Start 누른 후 누르는 경우	4. Show X_File	R 3.2
5-1	버튼 비활성	분석이 진행되는 도중 버튼을 누르는 경우	5. Show Detail	R 3.3
5-2	동일 내용 저장 여부	똑같은 소스 5 개로 Start 누른 후 누르는 경우	5. Show Detail	R 3.3
6-1	버튼 비활성	분석이 진행되는 도중 버튼을 누르는 경우	6. Exit	R 3.4
6-2	종료 작동	분석이 끝나고 버튼을 누르는 경우	6. Exit	R 3.4

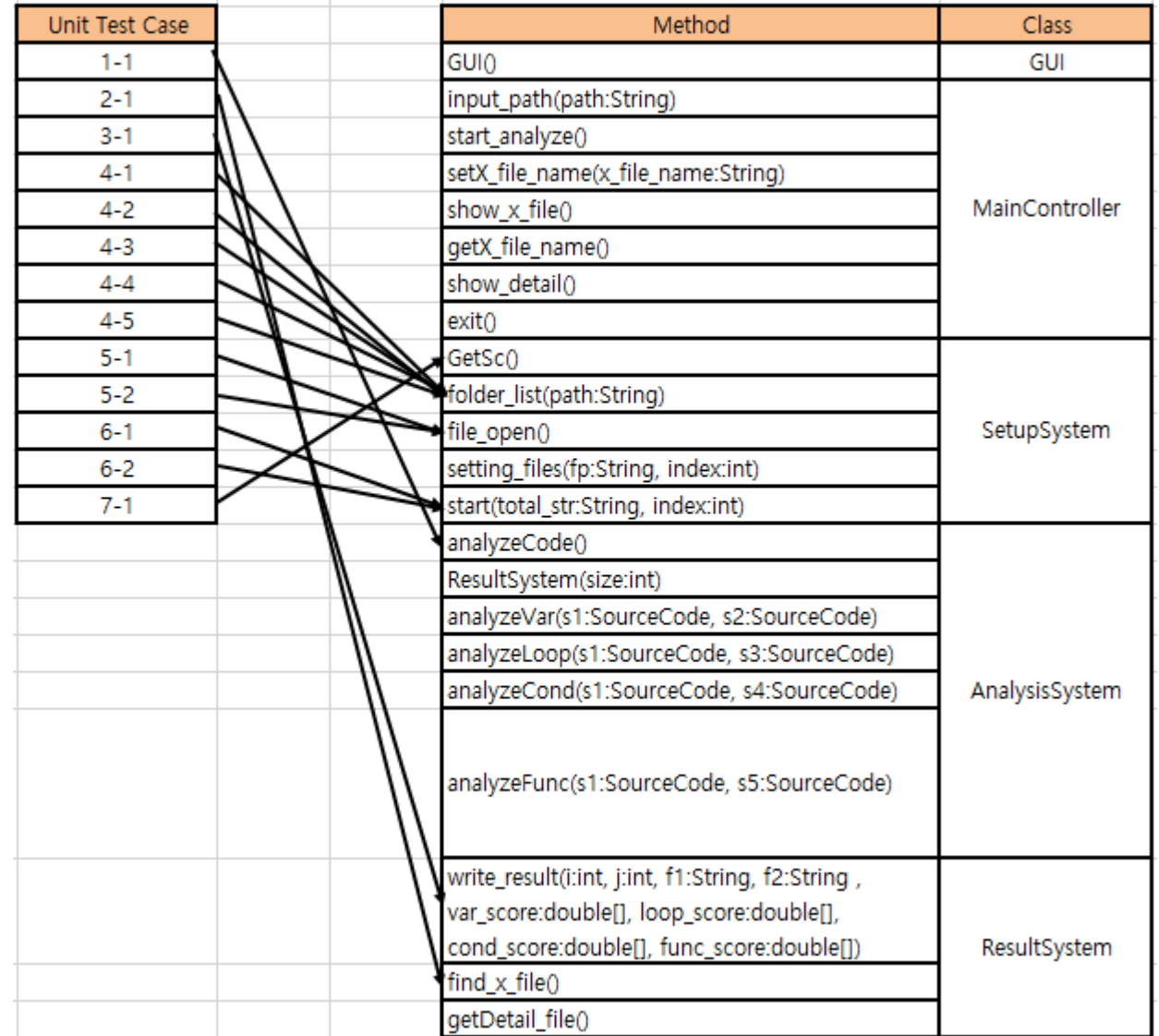
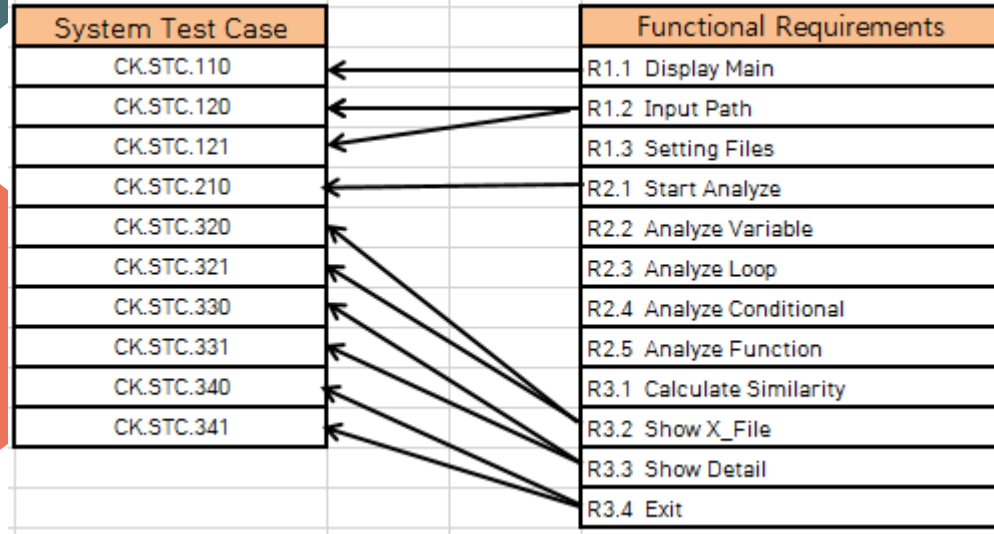
Activity2064. Performance Testing

번호	파일 개수	걸린 시간	
		(nsec)	(sec)
1	10	197439552	0.197439552
2	20	633340903	0.633340903
3	30	1371466352	1.371466352
4	40	2338068314	2.338068314
5	50	3617846584	3.617846584
6	60	5548379386	5.548379386
7	70	7716827443	7.716827443

<성능 비교 방법>

- .c 파일의 개수를 늘려가면 서 개수가 늘어날 때 마다 시간이 얼마나 걸리는지 비교한다.

Activity2067. Testing Traceability Analysis





THANK YOU

<http://leehyekang.com>

