

OSP Stage 2050

< Implementation >

그놈! Clone Checker

Project Team

T4

Date

2016-06-02

Team Information

201411258 강태준

201411265 김서우

201411321 홍유리

Contents

Activity2051. Implement Class & Method Definitions

Activity2052. Implement Windows

Activity2053. Implement Reports

Activity2055. Write Unit Test Code

[참고 자료] 검사 항목 별 기준

Activity2051. Implement Class & Method Definitions

- Class Definitions

Type	Class
Name	GUI
Purpose	Actor 가 직접적으로 프로그램과 연결되는 클래스로 Actor 에게 UI 를 보여줌
Overview (Class)	
Cross Reference	Functional Requirements : R1.1, R 2.1, R 3.1, R 3.2, R 4.1 UseCase : Display Main, Input Path, Start Analyze, Show X_File, Show Detail
Exceptional Courses of Events	N/A

Type	Class
Name	MainController
Purpose	GUI 클래스와 SetupSystem, AnalysisSystem 의 중간에 위치하는 클래스로 시스템 전반적인 진행을 도와준다.
Overview (Class)	
Cross Reference	Functional Requirements : R 1.2, R 2.1, R 3.1, R 3.2, R 4.1 UseCase : Input Path, Start Analyze, Show X_File, Show Detail
Exceptional Courses of Events	N/A

Type	Class
Name	AnalysisSystem
Purpose	정리된 소스코드의 데이터를 바탕으로 변수 유사도, 조건문 유사도, 반복문 유사도, 함수 유사도 점수를 측정하여 최종 유사도를 구하고, 검사의 세부 내용이 담긴 .txt 파일을 생성하여 내용을 작성하고, X_File 을 찾는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 2.1, R 2.1.1, R 2.1.2, R 2.1.3, R 2.1.4, R 2.1.5, R 2.1.6 UseCase : Start Analyze, Analyze Variable, Analyze Loop, Analyze Conditional, Analyze Function, Make Detail, Find X_File
Exceptional Courses of Events	N/A

Type	Class
Name	SetupSystem
Purpose	소스 코드의 정보를 비교, 분석하기 전에 코드를 분석할 수 있도록 주석 제거, 소문자 통일, 소스코드 정보 저장 등의 초기작업을 하는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 1.2, R 1.2.1 UseCase : Input Path, Setting Files
Exceptional Courses of Events	N/A

Type	Class
Name	SourceCode
Purpose	소스 코드의 정보들을 모아두는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 1.2.1, R 2.1, R 2.1.1, R 2.1.2, R 2.1.3, R 2.1.4, R 2.1.5, R 2.1.6 UseCase : Setting Files, Start Analyze, Analyze Variable, Analyze Loop, Analyze Conditional, Analyze Function, Make Detail, Find X_File
Exceptional Courses of Events	N/A

Type	Class
Name	Conditional
Purpose	소스 코드에 있는 조건문에 관한 정보를 저장하는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 2.1, R 2.1.3 UseCase : Start Analyze, Analyze Conditional
Exceptional Courses of Events	N/A

Type	Class
Name	Function
Purpose	소스 코드에 있는 함수에 관한 정보를 저장하는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 2.1, R 2.1.4 UseCase : Start Analyze, Analyze Function
Exceptional Courses of Events	N/A

Type	Class
Name	Loop
Purpose	소스 코드에 있는 반복문에 관한 정보를 저장하는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 2.1, R 2.1.2 UseCase : Start Analyze, Analyze Loop
Exceptional Courses of Events	N/A

Type	Class
Name	Variable
Purpose	소스 코드에 있는 변수에 관한 정보를 저장하는 클래스
Overview (Class)	
Cross Reference	Functional Requirements : R 2.1, R 2.1.1 UseCase : Start Analyze, Analyze Variable
Exceptional Courses of Events	N/A

-Method Definitions

1.GUI

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 Actor 에게 보여준다.
Exceptional Courses of Events	N/A

2.MainController

Type	Method
Name	input_path
Purpose	Actor 가 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.folder_list()으로 넘겨준다. 2. SetupSystem.folder_list()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
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	

Type	Method
Name	show_x_file
Purpose	X_File 파일 이름 반환
Cross Reference	Function : R 3.1 Use Case : Show X_File
Input (Method)	N/A
Output (Method)	temp : String
Abstract operation (Method)	1. getX_file_name()을 호출하여 X_File 의 이름을 받아온다. 3. temp 에 X_File 의 파일명을 저장하여 return 한다.
Exceptional Courses of Events	

Type	Method
Name	show_detail
Purpose	검사의 세부 내용이 담긴 파일 이름 반환
Cross Reference	Function : R 3.2 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 4.1 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	

3. AnalysisSystem

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)	<ol style="list-style-type: none"> 1. SourceCode[] sc 를 clone()을 이용하여 SourceCode[] sc_arr 에 복사한다. 2. 최종 유사도를 저장할 double[][] final_result를 초기화 시킨다. 3. 이 메소드가 호출된 순간의 "연도월일시분초"를 이름으로 하는 검사의 세부 내용을 저장할 .txt 파일의 이름을 만든다..
Exceptional Courses of Events	N/A

Type	Method
Name	analyzeCode
Purpose	SourceCode 객체 2 개를 묶어서 분석을 진행한다.
Cross Reference	Function : R 2.1, R 2.1.1, R 2.1.2, R 2.1.3, R 2.1.4, R 2.1.5, R 2.1.6 Use Case : Start Analyze, Analyze Variable, Analyze Loop, Analyze Conditional, Analyze Function, Make Detail, Find X_File
Input (Method)	N/A
Output (Method)	result : String
Abstract operation (Method)	<ol style="list-style-type: none"> 1. 각각의 유사도 검사 결과 값을 담은 double 형 이차원 배열 score 를 생성한다. double[][] score = new double[4][13] 3. 이중 for 문을 통하여 소스코드를 2 개씩 묶어서 유사도 검사를 진행한다. 4. analyzeVar()를 호출하여 변수 유사도 검사 결과를 score[0]에 저장한다.

	<p>5. analyzeFunc()를 호출하여 함수 유사도 검사 결과를 score[1]에 저장한다.</p> <p>6. analyzeLoop()를 호출하여 반복문 유사도 검사 결과를 score[2]에 저장한다.</p> <p>7.analyzeCond()를 호출하여 조건문 유사도 검사 결과를 score[3]에 저장한다.</p> <p>8. make_Detail()을 호출하여 4. 5. 6. 7. 의 세부 내용을 .txt 파일에 저장하고, 최종 유사도를 계산한다.</p> <p>9. find_x_file()을 호출하여 X_File 의 인덱스를 찾아 result_index 에 담아 value 에 return 한다.</p> <p>7. sc_arr 에서 value 인덱스 값의 file_name 을 String 변수 result 에 저장하여 return 한다.</p>
Exceptional Courses of Events	

Type	Method
Name	analyzeVar
Purpose	코드에 있는 변수의 유사도를 분석한다.
Cross Reference	Function : R 2.1.1 Use Case : Analyze Variable
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	<p>1. 변수 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다.</p> <p>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[2]와 score[3]에 저장한다.(참고 자료 1. - 1))</p> <p>3. s1 과 s2 사이에 동일한 자료형 내의 동일한 변수명의 개수를 세어 score[5]에 저장하고 기준에 의하여 점수를 부여하고 이를 2 로 나눈 후 score[4]에 저장한다. (참고 자료 1. - 2))</p> <p>4. 변수 유사도 검사의 최종 점수를 score[6]에 저장한다. (참고 자료 1 - 3))</p> <p>5. score 를 return 한다.</p>
Exceptional Courses of Events	

Type	Method
Name	analyzeLoop
Purpose	코드에 있는 반복문의 유사도를 분석한다.
Cross Reference	Function : R 2.1.2 Use Case : Analyze Loop
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	<ol style="list-style-type: none"> 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 2 - 1) 3. s1 내의 while 문의 개수와 s2 내의 while 문의 개수의 차를 기준으로 점수를 부여하여 이를 score[1]에 저장한다. 또한 각각의 while 문의 개수를 score[5]과 score[6]에 저장한다. (참고 자료 2 - 2) 4. 반복문 유사도 검사의 최종 점수를 score[2]에 저장한다. (참고 자료 2 - 3) 5. score 를 return 한다.
Exceptional Courses of Events	

Type	Method
Name	analyzeCond
Purpose	코드에 있는 조건문의 유사도를 분석한다.
Cross Reference	Function : R 2.1.3 Use Case : Analyze Conditional
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	<ol style="list-style-type: none"> 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.

	<p>또한 각각의 if 문의 개수를 score[5]과 score[6]에 저장한다. (참고 자료 3 - 1))</p> <p>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 3 - 2))</p> <p>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 3 - 3))</p> <p>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 3 - 4))</p> <p>6. 조건문 유사도 검사의 최종 점수를 score[4]에 저장한다. (참고 자료 3 - 5))</p> <p>7. score 를 return 한다.</p>
Exceptional Courses of Events	

Type	Method
Name	analyzeFunc
Purpose	코드에 있는 함수의 유사도를 분석한다.
Cross Reference	Function : R 2.1.4 Use Case : Analyze Function
Input (Method)	s1 : SourceCode , s2 : SourceCode
Output (Method)	score : double[]
Abstract operation (Method)	<p>1. 함수 유사도 검사의 데이터를 저장할 double 형 배열 score 를 생성한다.</p> <p>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 4 - 1))</p> <p>3. s1 내의 모든 함수와 s2 내의 모든 함수의 이름을 비교하여 이름이 동일한 함수의 개수를 세어 이를 score[3]에 저장하고, 기준에 의해 점수를 부여하여 score[4]에 저장한다. (참고 자료 4 - 2))</p>

	<p>4. s1 내의 모든 함수 body 를 하나의 문자열로 이어 붙이고, s2 내의 모든 함수 body 를 하나의 문자열로 이어 붙인 뒤에, 그 안에서 등장하는 변수는 모두 "attr"로, 함수는 "oper"로 바꾸어 두 문자열 사이의 유사도를 분석하여 score[7]에 저장하고, 그에 따라 점수를 부여하여 score[6]에 저장한다. (참고 자료 4 - 3)</p> <p>4. 가중치를 적용하여 함수 유사도 검사의 최종 점수를 score[5]에 저장한다. (참고 자료 4 - 4)</p> <p>5. score 를 return 한다.</p>
Exceptional Courses of Events	

4. SetupSystem

Type	Method
Name	folder_list
Purpose	MainController 가 넘겨준 path 를 가지고, 입력된 경로에 있는 파일의 리스트를 가져온다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
Input (Method)	path : String
Output (Method)	flag : int
Abstract operation (Method)	<p>1. path 가 유효한지 확인한다.</p> <p>2. path 유효한 경우 (flag = 0) path 의 파일 리스트를 fileList 에 받아온다.</p> <p>3. fileList 의 길이를 토대로 String 형 배열 file_name 과 short_file 을 생성한다.</p> <p>4. for 문을 통하여 short_file 에는 파일의 이름만을, file_name 에는 절대경로를 포함한 파일의 이름을 저장한다.</p> <p>5. file_open()을 호출하여 return 값을 flag 에 받는다.</p> <p>6. flag 를 return 한다.</p>
Exceptional Courses of Events	폴더 경로가 유효하지 않은 경우 (flag = 1) flag 를 return 한다.

Type	Method
-------------	--------

Name	file_open
Purpose	파일 리스트에 들어있는 파일을 하나씩 열어준다.
Cross Reference	Function : R 1.2.1 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 sc 를 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_flag 를 return 한다.

Type	Method
Name	setting_files
Purpose	분석하기에 앞서 분석할 파일을 주석 제거 및 대소문자 통합해준다.
Cross Reference	Function : R1.2.1 Use Case : Setting Files
Input (Method)	fp : String , index : int
Output (Method)	N/A
Abstract operation (Method)	1. fp 를 경로로 가진 파일을 열어 파일을 하나의 문자열에 저장하고 문자열의 주석을 삭제한다. 2. 문자열의 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)	<ol style="list-style-type: none"> 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 초기화시켜준다.(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

Type	Method
Name	make_Detail
Purpose	분석결과를 메모장에 저장한다.
Cross Reference	Function : R 2.1.6 R 3.2 Use Case : Make Detail, Show Detail
Input (Method)	N/A
Output (Method)	w_flag : boolean
Abstract operation (Method)	<ol style="list-style-type: none"> 1. boolean 형 변수 w_flag 생성 2. 생성자에서 미리 정해 놓은 detail_file 을 엽니다. 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 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_flag 를 return 한다.

Type	Method
Name	find_x_file
Purpose	분석결과를 토대로 가장 핵심이 되는 것 같은 파일을 찾는다.
Cross Reference	Function : R 2.1 Use Case : Start Analyze
Input (Method)	N/A
Output (Method)	result_index : int
Abstract operation (Method)	1. 최종 유사도를 저장한 double[][] final_result의 각 행의 평균 값을 행을 인덱스로 하는 double[] avr에 저장한다. 2. double[] avr의 원소 중 최댓값을 찾는다. 3. 최댓값에 해당하는 double[] avr의 인덱스를 return_index에 저장하여 return 한다.
Exceptional Courses of Events	

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	

6. SourceCode

Type	Method
Name	SourceCode
Purpose	안에 들어있는 변수들을 초기화 한다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
Input (Method)	file_name : String
Output (Method)	N/A
Abstract operation (Method)	1. 클래스 변수들을 객체화 시켜준다.
Exceptional Courses of Events	

7. Variable

Type	Method
Name	Variable
Purpose	안에 들어있는 변수들을 초기화 한다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
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	

8. Loop

Type	Method
Name	Loop
Purpose	안에 들어있는 변수들을 초기화 한다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
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	

9. Conditional

Type	Method
Name	Conditional
Purpose	안에 들어있는 변수들을 초기화 한다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
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	

10. Function

Type	Method
Name	Function
Purpose	안에 들어있는 변수들을 초기화 한다
Cross Reference	Function : R 1.2.1 Use Case : Setting Files
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

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	분석진행중/분석완료 팝업창
Post-Conditions	UI 초기화면

Name	Input
Responsibilities	Textbox 에 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	경로탐색 성공/실패에 대한 팝업창
Post-Conditions	UI 초기화면

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	XFile 을 보여주는 팝업창
Post-Conditions	UI 초기화면

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	메모장
Post-Conditions	UI 초기화면

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	종료를 확인하는 팝업창
Post-Conditions	UI 초기화면

Activity2055. Write Unit Test Code

[Class AnalysisSystemTest]

```
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_이
명재_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"));

    }

}

```

[Class SetupSystemTest]

```

import static org.junit.Assert.*;

import org.junit.Test;

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_이명
재_R02_V01\\MClient\\asd\\good");

    assertEquals("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명
재_R02_V01\\MClient\\asd\\good\\AbsoCompare - 복사본 - 복사본 - 복사
본.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_이명  
재_R02_V01\\MClient\\asd\\good");
```

```
    assertEquals(2,ss.file_open());
```

```
    //2-2 디렉토리내에 ".c"말고 다른파일이 존재할때, flag가 1을 반환하는지 확  
인
```

```
    ss.folder_list("C:\\Users\\서우\\Desktop\\반바지\\NP4039_201111377_이명  
재_R02_V01\\MClient\\asd");
```

```
    assertEquals(1,ss.file_open());
```

```
}
```

```
@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_이명
```

```

재_R02_V01\\MClient\\asd\\good");

    assertNotNull(ss.sc[0].func.get(0).var_list.get(0));

    //3-2 Start() 실행시 i번째 소스코드의 함수 개수와 sc[i].num_of_func 같은
지 확인

    assertEquals(3,ss.sc[0].num_of_func); //첫번째 소스코드,즉,"AbsoCompare
- 복사본 - 복사본 - 복사본.c"안에는 3개의 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_이명
재_R02_V01\\MClient\\asd\\good");

    assertEquals(ss.sc,ss.getSc());

}

}

```

[참고 자료] 검사 항목 별 기준

1. 변수 검사 기준

1) 전체 변수 개수 비교 (50%)

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

2) 동일한 이름을 갖는 변수 개수 세기 (50%)

(동일한 이름의 변수 개수의 총합/소스 코드의 전체 변수 개수) * 100(%) 를 계산하여 50% 이상일 경우 각각의 소스코드에 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) 최종적인 변수 유사도 계산

(전체 변수 개수 비교 점수) * 0.5 + (동일한 이름을 갖는 변수 개수) * 0.5

2. 반복문 검사 기준

1) for 문의 개수 비교 (50%)

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

2) while 문의 개수 비교 (50%)

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

3) 최종적인 반복문 유사도 계산

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

3. 조건문 검사 기준

1) if 문의 개수 비교 (30%)

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

2) else 문의 개수 비교 (20%)

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

3) else if 문의 개수 비교 (25%)

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

4) case 문의 개수 비교 (25%)

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

5) 최종적인 조건문 유사도 계산

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

4. 함수 검사 기준

1) 전체 함수 개수 비교 (20%)

차이	유사도 점수
0	100
1	70
2	50
그 이상	30

2) 함수 이름 비교 (30%)

동일한 이름을 갖는 함수의 개수를 센다.

(총 개수) * 10 점

3) 함수의 내용 비교(50%)

소스 코드의 전체 함수 body를 하나로 모으고 모든 변수의 이름을 "attr"로, 모든 함수의 이름을 "oper"로 바꾼 상태에서 두 함수 body 모음을 비교한다. 유사도에 따라서 점수를 부여한다.

결과	점수
90 ~ 100	100
80 ~ 90	90
70 ~ 80	80
60 ~ 70	70
50 ~ 60	60

4) 최종적인 함수 유사도 계산

(전체 함수 개수 비교 점수) * 0.2 + (함수 이름 비교 점수) * 0.3 + (함수 내용 비교 점수) * 0.5

5. 최종 검사 결과

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