쾌적한
Clone Checker

OOPT Stage 2050 Construct, 2060 Test

Software Modeling & Analysis

유준범 교수님

Team. T1

201111388 조연호

201211374 이창오

201211379 장종훈

201314196 양동혁

목차

Contents

- ✓ Revise Plan
- ✓ Implement Class & Methods Definitions (Activity 2051)
- ✓ Implement Windows (Activity 2052)
- ✓ Unit Testing
- ✓ Demonstration

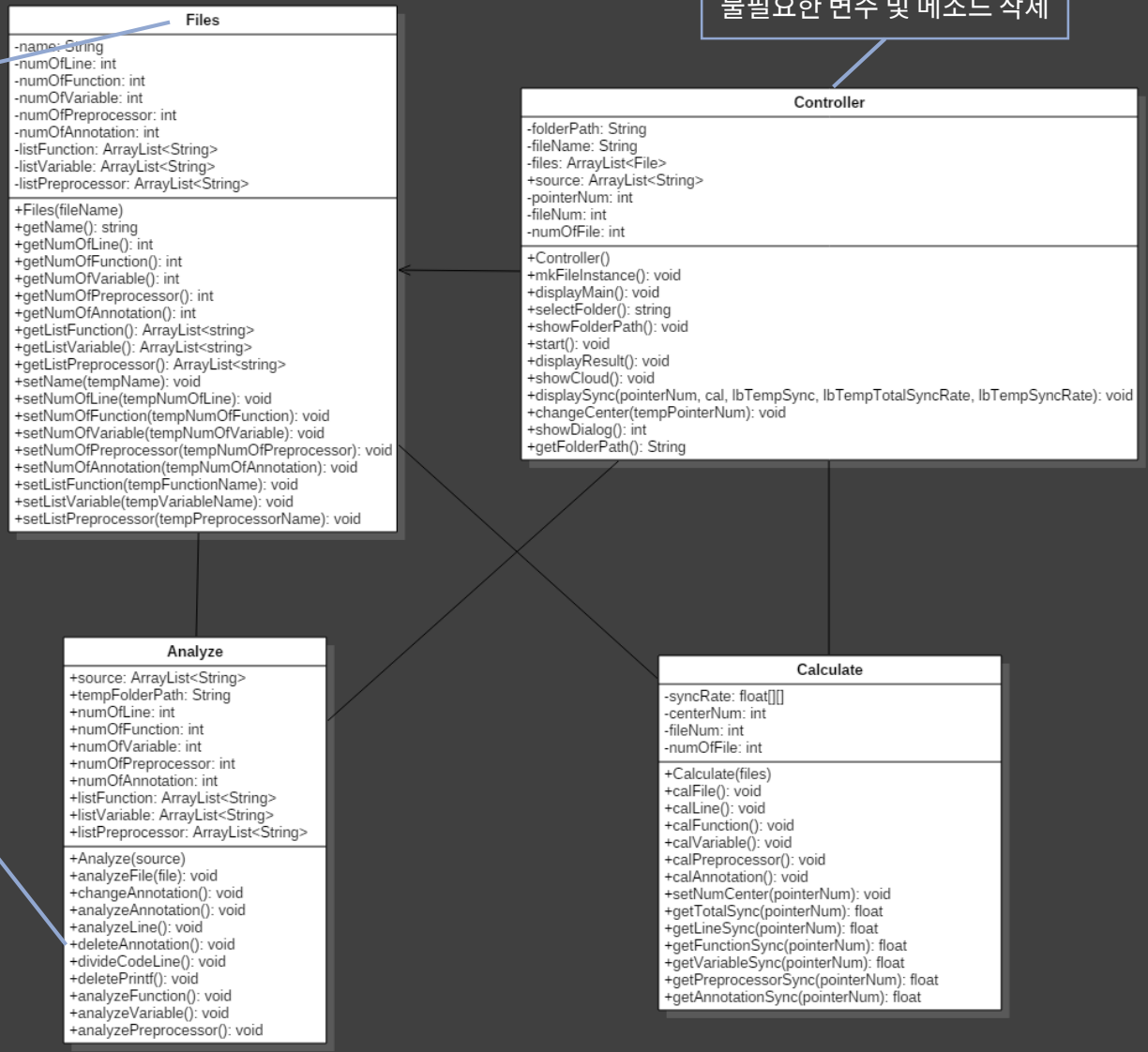
Revise Plan

JAVA 기본 'File' 클래스와 이름 충돌
클래스 이름 변경(File → Files)

불필요한 변수 및 메소드 삭제

각 클래스별 생성자 추가

정밀한 소스코드 분석을 위한
changeAnnotation(),
deleteAnnotation(),
divideCodeLine(),
deletePrintf() 메소드 추가



Implement Class & Methods Definitions _ Class

Type	Class	Class	Class	Class
Name	Controller	Analyze	Calculate	Files
Purpose	전반적인 프로그램의 수행을 제어하는 클래스	생성된 파일 객체들 안에 저장된 코드를 분석하는 클래스	분석된 값을 비교하여 일치율을 계산하는 클래스	각각의 파일들의 소스 코드 및 분석된 값을 저장하는 클래스
Overview (Class)	폴더 선택, 파일 분석, 일치율 계산, 화면 출력 등을 제어한다.	ArrayList source에 저장된 코드들을 수정을 거쳐 분석한 뒤 값을 저장한다.	분석된 값을 비교하여 일치율을 계산한다.	소스코드 및 분석된 값을 저장하고 반환한다.
Cross Reference	Select Folder Start Analyze File Change Center Calculate File Display Sync	N/A	Calculate File Calculate Line Sync-Rate Calculate Function Sync-Rate Calculate Variable Sync-Rate Calculate Preprocessor Sync-Rate Calculate Annotation Sync-Rate	Change Center

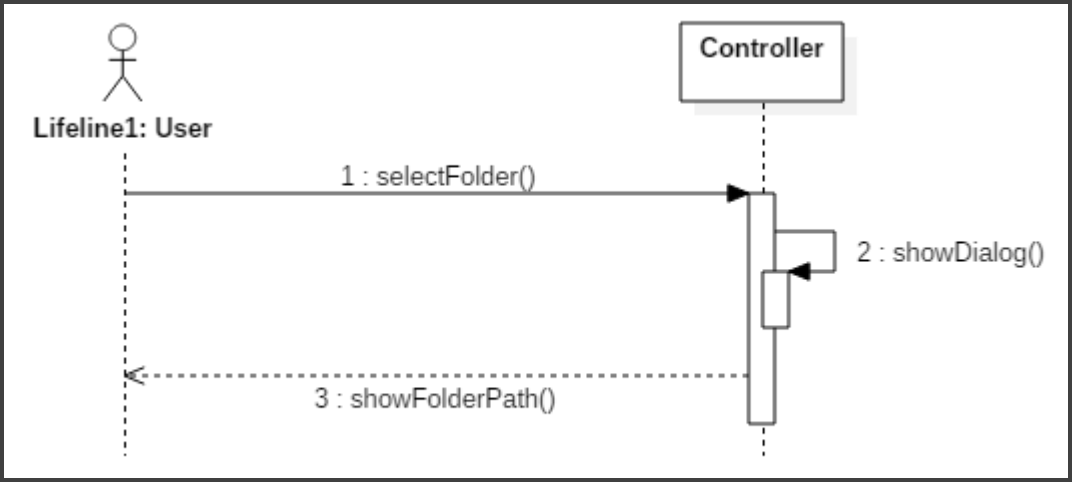
Implement Class & Methods Definitions _ Analyze

Type	Method	Method	Method	Method	Method
Name	analyzeLine()	analyzeFunction()	analyzeVariable()	analyzePreprocessor()	analyzeAnnotation()
Purpose	코드의 라인 수를 계산하는 메소드	코드의 함수 수, 이름을 분석하는 메소드	코드의 변수 수, 이름을 분석하는 메소드	코드의 전처리기 수, 이름을 분석하는 메소드	주석 처리된 줄의 개수를 계산하는 메소드
Overview (Class)	코드의 라인 수를 계산한다.	코드의 함수 수, 이름을 분석한다.	코드의 변수 수, 이름을 분석한다.	코드의 전처리기 수, 이름을 분석한다.	주석 처리된 줄의 개수를 계산한다.
Cross Reference	Analyze File	Analyze File	Analyze File	Analyze File	Analyze File
Input (Method)	N/A	N/A	N/A	N/A	N/A
Output (Method)	void	void	void	void	void
Abstract Operation (Method)	코드의 라인 수를 계산하여 numOfLine에 저장한다.	코드의 함수 수, 이름을 분석하여 numOfFunction과 listFunction에 저장한다.	코드의 변수 수, 이름을 분석하여 numOfVariable, listVariable에 저장한다.	코드의 전처리기 수, 이름을 분석하여 numOfPreprocessor, listPreprocessor에 저장한다.	주석 처리된 줄의 개수를 계산하여 numOfAnnotation에 저장한다.
Exceptional Courses of Events	N/A	N/A	N/A	N/A	N/A

Implement Class & Methods Definitions _ Analyze

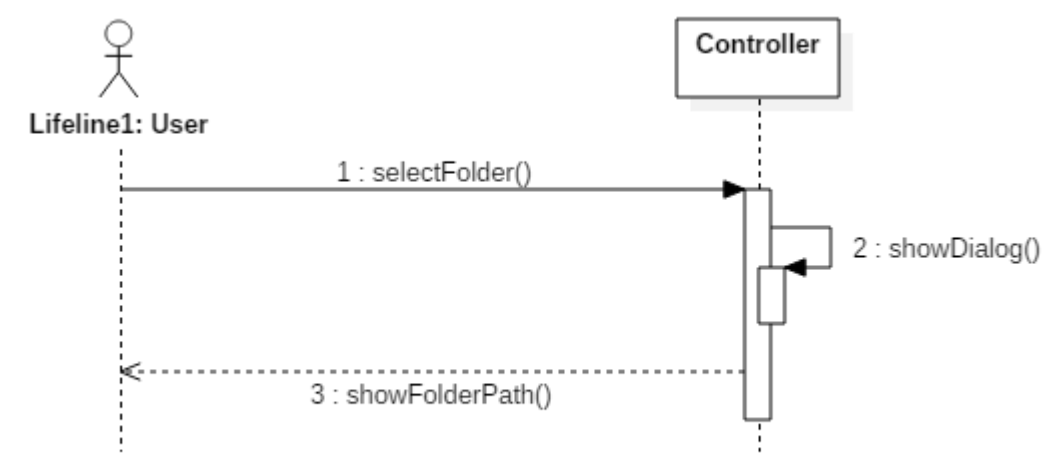
Type	Method	Method	Method	Method
Name	changeAnnotation()	deleteAnnotation()	divideCodeLine()	deletePrintf()
Purpose	주석 분석 전에 '/**/'으로 주석 처리된 줄을 '//'으로 변경하는 메소드	분석 기능에 불필요한 주석 처리된 부분을 삭제하는 메소드	한 줄 안에 코드가 여러 개인 경우 여러 줄로 분할하는 메소드	분석 기능에 불필요한 printf문의 괄호 안의 내용들을 삭제하는 메소드
Overview (Class)	'/**/'으로 주석 처리된 줄을 '//'으로 변경한다.	'//'으로 주석 처리된 부분을 삭제한다.	한 줄 안에 코드가 여러 개인 경우 여러 줄로 분할한다.	printf문의 괄호 안의 내용들을 삭제한다.
Cross Reference	Analyze File	Analyze File	Analyze File	Analyze File
Input (Method)	N/A	N/A	N/A	N/A
Output (Method)	void	void	void	void
Abstract Operation (Method)	ArrayList source에서 '/**/'으로 주석 처리된 줄을 '//'으로 변경한다.	ArrayList source에서 '//'으로 주석 처리된 부분을 삭제한다.	ArrayList source를 ';'를 기준으로 여러 개로 분할하여 추가한다.	ArrayList source에서 printf문의 괄호 안의 내용들을 삭제한다.
Exceptional Courses of Events	N/A	N/A	N/A	N/A

Implement Windows _ Select Folder



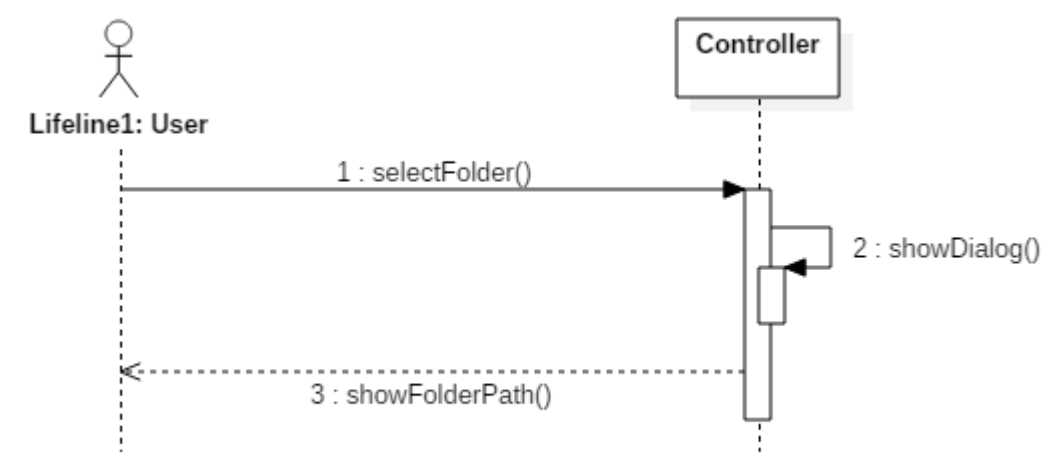
Name	selectFolder
Responsibilities	폴더 선택 버튼을 클릭한다.
Type	GUI
Cross Reference	Select Folder
Notes	폴더를 지정하는 창을 보여주는 showDialog() 메소드를 호출한다.
Pre-Conditions	N/A
Post-Conditions	showDialog()가 실행된다.

Implement Windows _ Select Folder



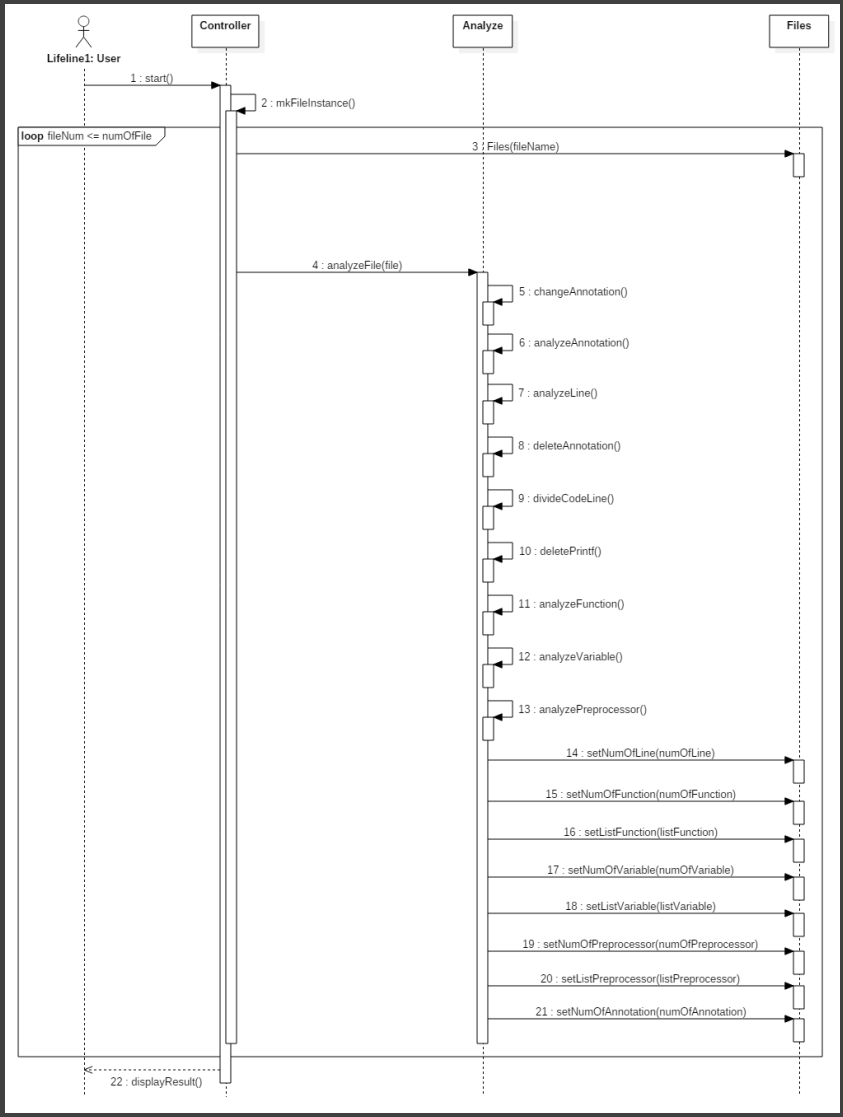
Name	showDialog()
Responsibilities	폴더 경로를 지정한다.
Type	GUI
Cross Reference	Select Folder
Notes	폴더를 지정하는 창이 출력된다.
Pre-Conditions	폴더 선택 버튼을 누른다.
Post-Conditions	메인 화면에 지정된 폴더 경로를 출력한다.

Implement Windows _ Select Folder



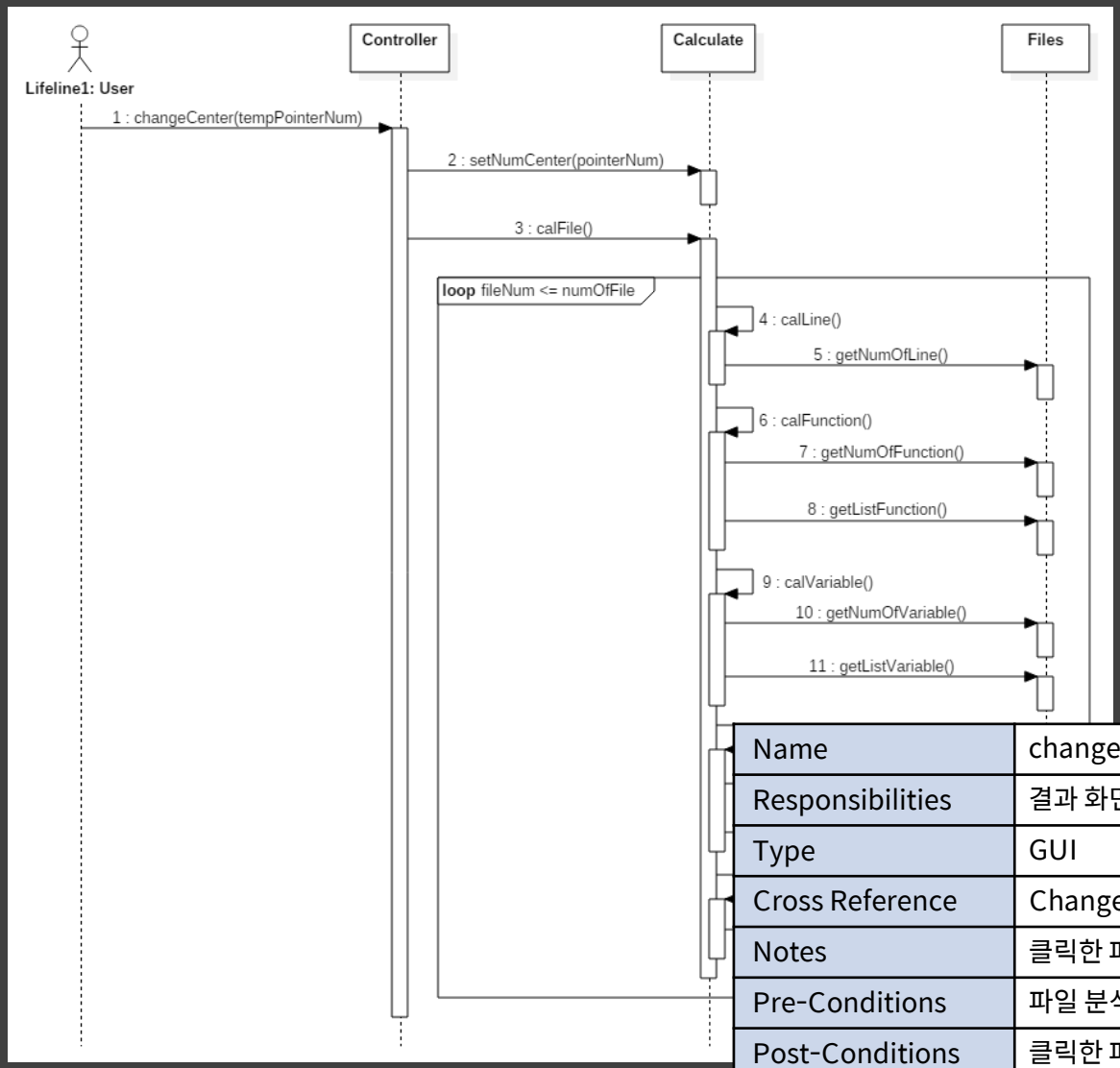
Name	showFolderPath()
Responsibilities	사용자가 폴더를 지정한 후 메인 화면으로 돌아온다.
Type	GUI
Cross Reference	Select Folder
Notes	메인 화면에 지정된 폴더 경로가 출력된다.
Pre-Conditions	폴더 선택을 완료한다.
Post-Conditions	메인 화면에 지정된 폴더 경로가 출력된다.

Implement Windows _ Start



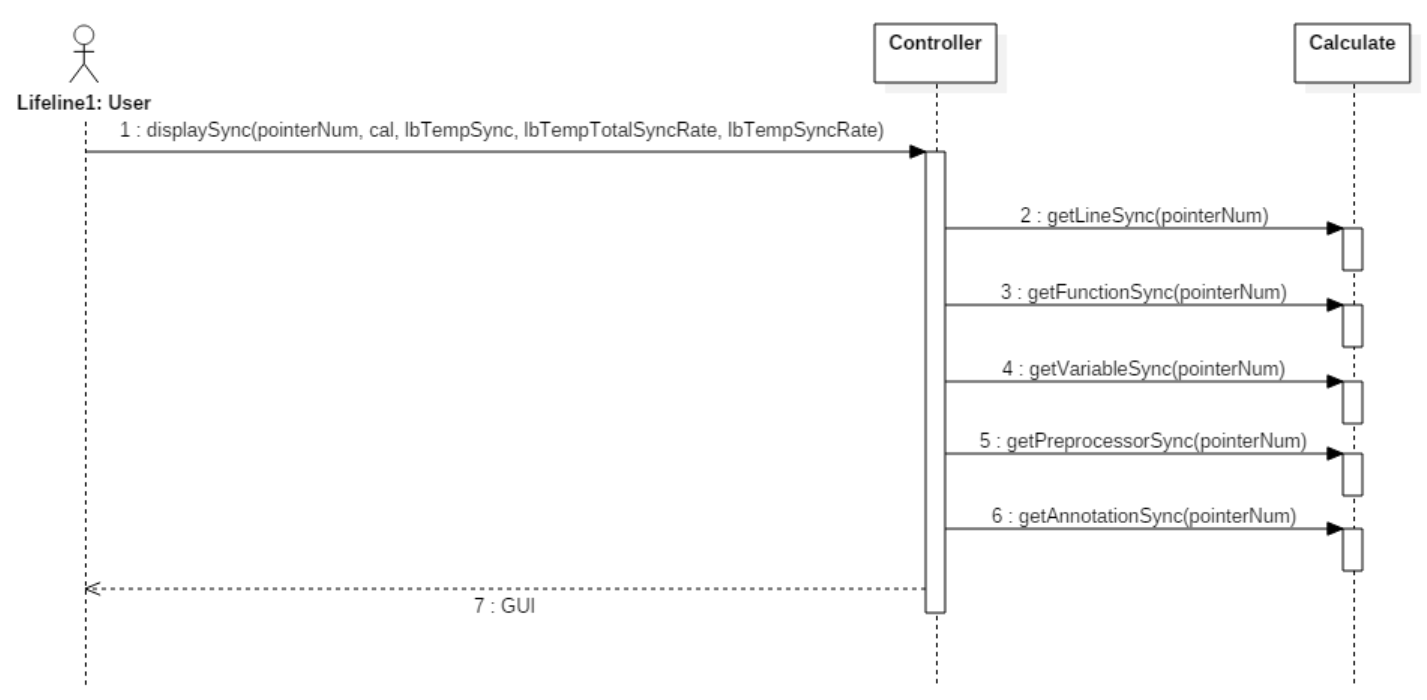
Name	displayResult()
Responsibilities	시작하기 버튼을 누른다.
Type	GUI
Cross Reference	Start
Notes	분석 결과 창을 출력한다.
Pre-Conditions	폴더 경로 지정과 파일 분석을 완료한다.
Post-Conditions	분석 결과 창을 출력한다.

Implement Windows _ Change Center



Name	changeCenter()
Responsibilities	결과 화면의 클라우드 또는 오른쪽 리스트에서 파일을 클릭한다.
Type	GUI
Cross Reference	Change Center
Notes	클릭한 파일을 기준으로 비교 결과 화면을 새로 출력한다.
Pre-Conditions	파일 분석이 끝나고 결과 화면이 나타난다.
Post-Conditions	클릭한 파일을 기준으로 비교 결과 화면을 새로 출력한다.

Implement Windows _ Display Sync



Name	displaySync()
Responsibilities	클라우드 내 소스 코드 파일 이름 위에 마우스를 오버한다.
Type	GUI
Cross Reference	Display Sync
Notes	기준 파일과 마우스 오버된 파일의 상세 일치율 정보를 출력한다.
Pre-Conditions	기준 파일이 선택되어 비교 결과가 나타난다.
Post-Conditions	기준 파일과 마우스 오버된 파일의 상세 일치율 정보를 출력한다.

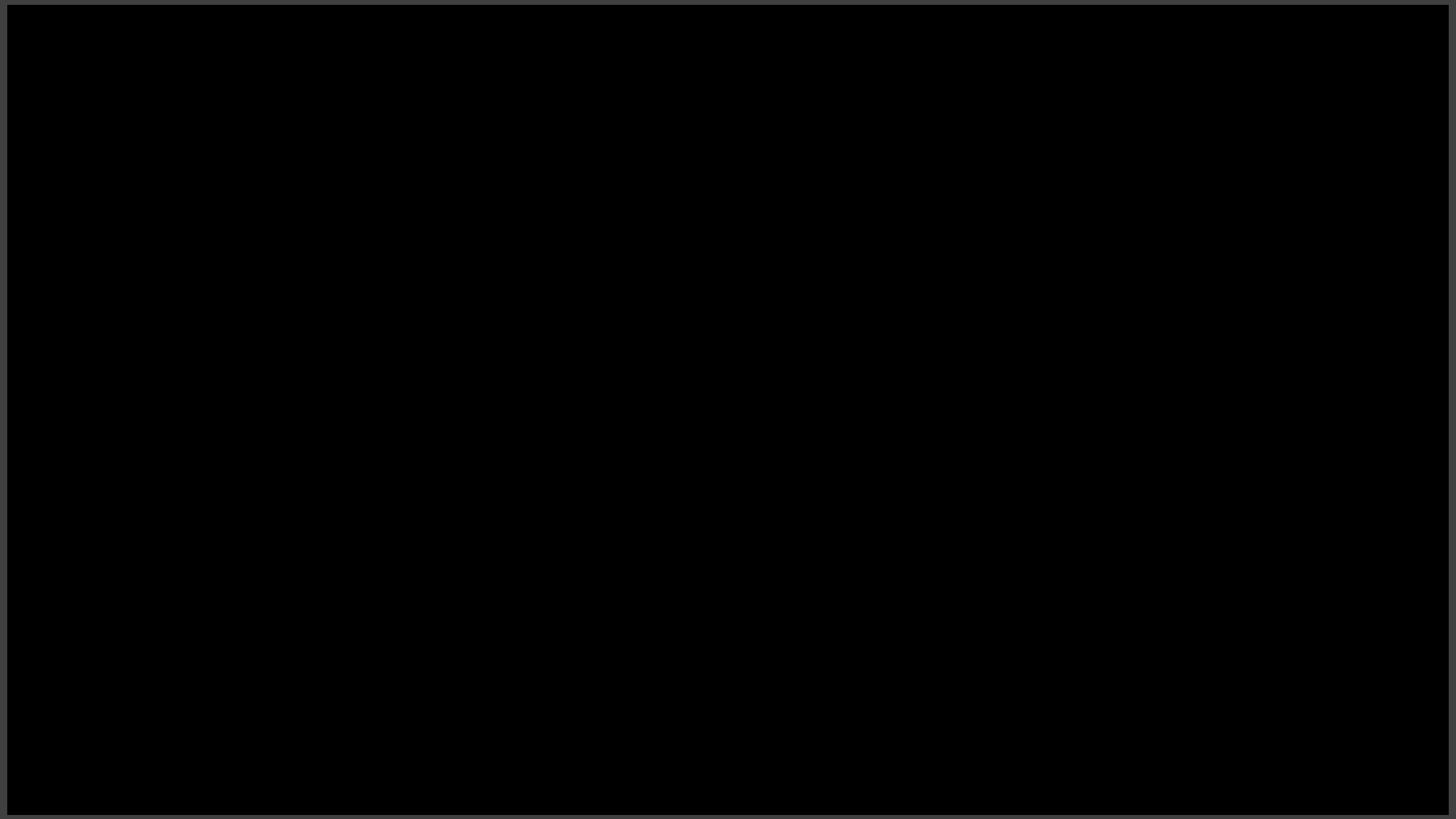
Unit Testing _ Analyze

1	Name = "analyzeLine"	Input	Output	Result
1.1	assertEquals(19, result); 19줄짜리 코드를 입력 시 올바른 값을 출력	19줄짜리 코드	19	PASS
2	**Name = "analyzeFunction"**	**Input**	**Output**	**Result**
1.2.1	assertEquals(2, result); 함수 2개가 포함된 소스코드 입력 시 올바른 값을 출력	함수가 2개인 소스코드	2	PASS
1.2.2	assertEquals("int main(void)", result.get(0)); *analyzeFuntion*이 함수가 포함된 소스코드 입력 시 배열 순서에 따라 함수를 순서적으로 저장하고 있는지를 출력	함수 이름	함수 이름	PASS
1.2.3	assertEquals("void f(void)", result.get(1)); 1.2.2 와 같음. 단지 순서가 2번째로 오는 함수가 출력	함수 이름	함수 이름	PASS
3	**Name = "analyzeVariable"**	**Input**	**Output**	**Result**
1.3.1	assertEquals(2, result); 변수 개수가 2개인 소스코드를 입력할 시 그 개수를 출력	static int x, int y	2	PASS
1.3.2	assertEquals("static int x", result.get(0)); 소스코드 안에 있는 변수들을 배열에 저장하여, 순서대로 출력	static int x	Static int x	PASS
4	**Name = "analyzePreprocessor"**	**Input**	**Output**	**Result**
1.4.1	assertEquals(1, result); 전처리기가 1개인 소스 코드 입력 시 그 개수를 출력	#include <stdio.h>	1	PASS
1.4.2	assertEquals("#include<stdio.h>", result); 소스코드에 포함된 전처리기를 순서대로 배열에 저장하여, 배열 값에 따라 출력	#include <stdio.h>	#include <stdio.h>	PASS
5	**Name = "analyzeAnnotation"**	**Input**	**Output**	**Result**
1.5.1	assertEquals(2, result); 주석 개수가 2개인 소스코드 입력 시 그 개수를 출력	주석 2개 소스코드	2	PASS

Unit Testing _ Calculate

1	Name = "CalLine"	Input	Output	Result
2.1	assertEquals(40, result); 코드의 라인 개수가 각각 100개, 90개인 두 개의 소스코드 입력 시 그 일치율을 출력	두개의 소스코드	40	PASS
2	**Name = "CalFunction"**	**Input**	**Output**	**Result**
2.2	assertEquals(60, result); *CalFuntion*에서 확인하고자 하는 함수의 개수의 차이에 따른 일치율과 함수 이름에 따른 일치율의 *SyncRate*를 출력한다.	두 개의 소스코드	60	PASS
3	**Name = "CalVariable"**	**Input**	**Output**	**Result**
2.3	assertEquals(60, result); *CalVariable*에서 확인하고자 하는 변수의 개수의 차이에 따른 일치율과 변수 이름에 따른 일치율의 *SyncRate*를 출력한다.	두 개의 소스코드	60	PASS
4	**Name = "CalPreprocessor"**	**Input**	**Output**	**Result**
2.4	assertEquals(60, result); *CalPreprocessor*에서 확인하고자 하는 전처리기의 개수의 차이에 따른 일치율과 전처리기 이름에 따른 일치율의 *SyncRate*를 출력한다.	두 개의 소스코드	60	PASS
5	**Name = "CalAnnotation"**	**Input**	**Output**	**Result**
2.5	assertEquals(60, result); 코드의 주석 개수가 각각 100개, 90개인 두 개의 소스코드 입력 시 그 일치율을 출력	두 개의 소스코드	40	PASS

Demonstration



https://youtu.be/0Z-TATPa8U0

쾌적한
Clone Checker

고맙습니다.

Software Modeling & Analysis

유준범 교수님