module01_project02 공학용 계산기

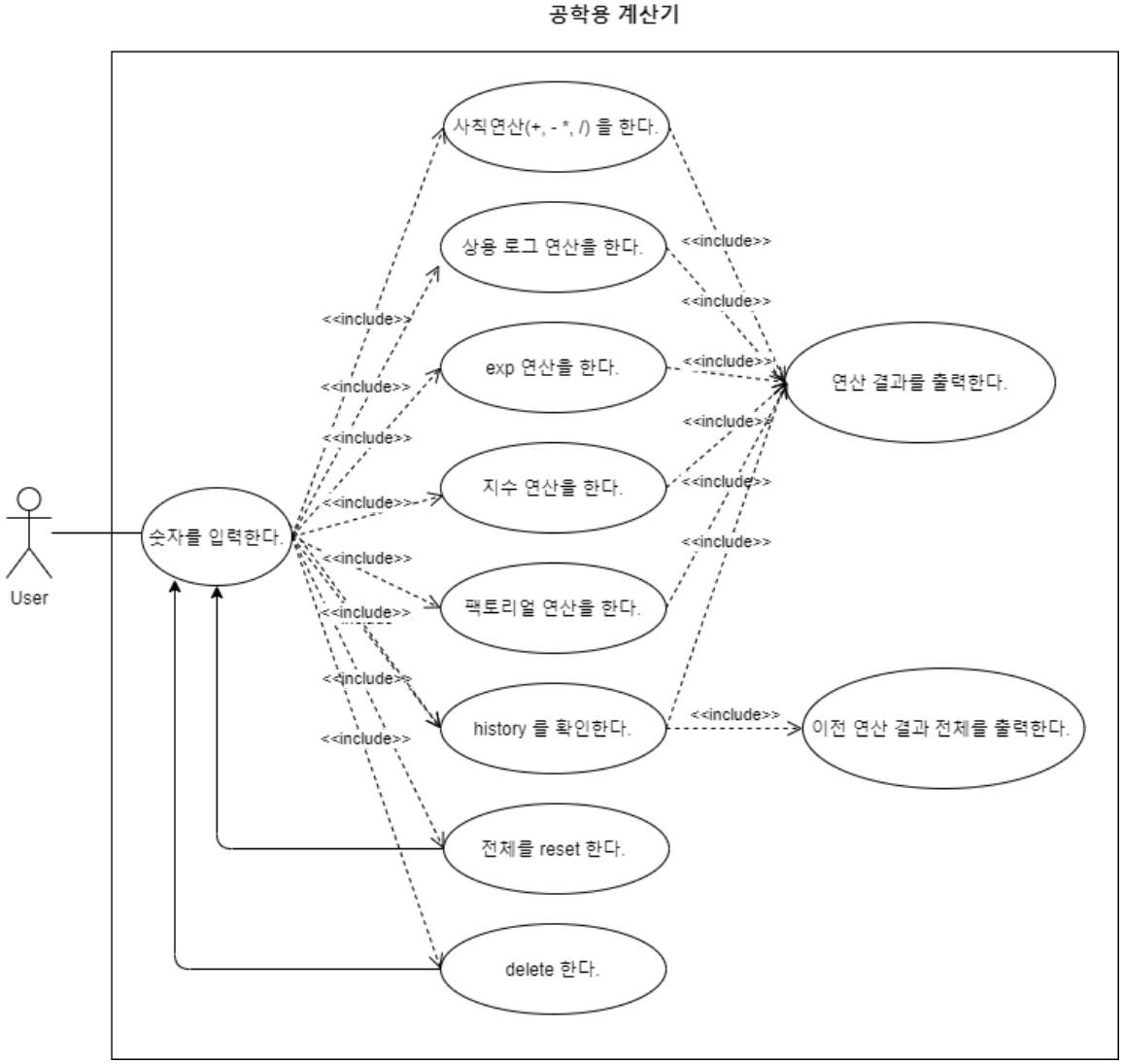
요구 사항 분석

- 자바, 안드로이드 스튜디오를 활용한 공학용 계산기 구현

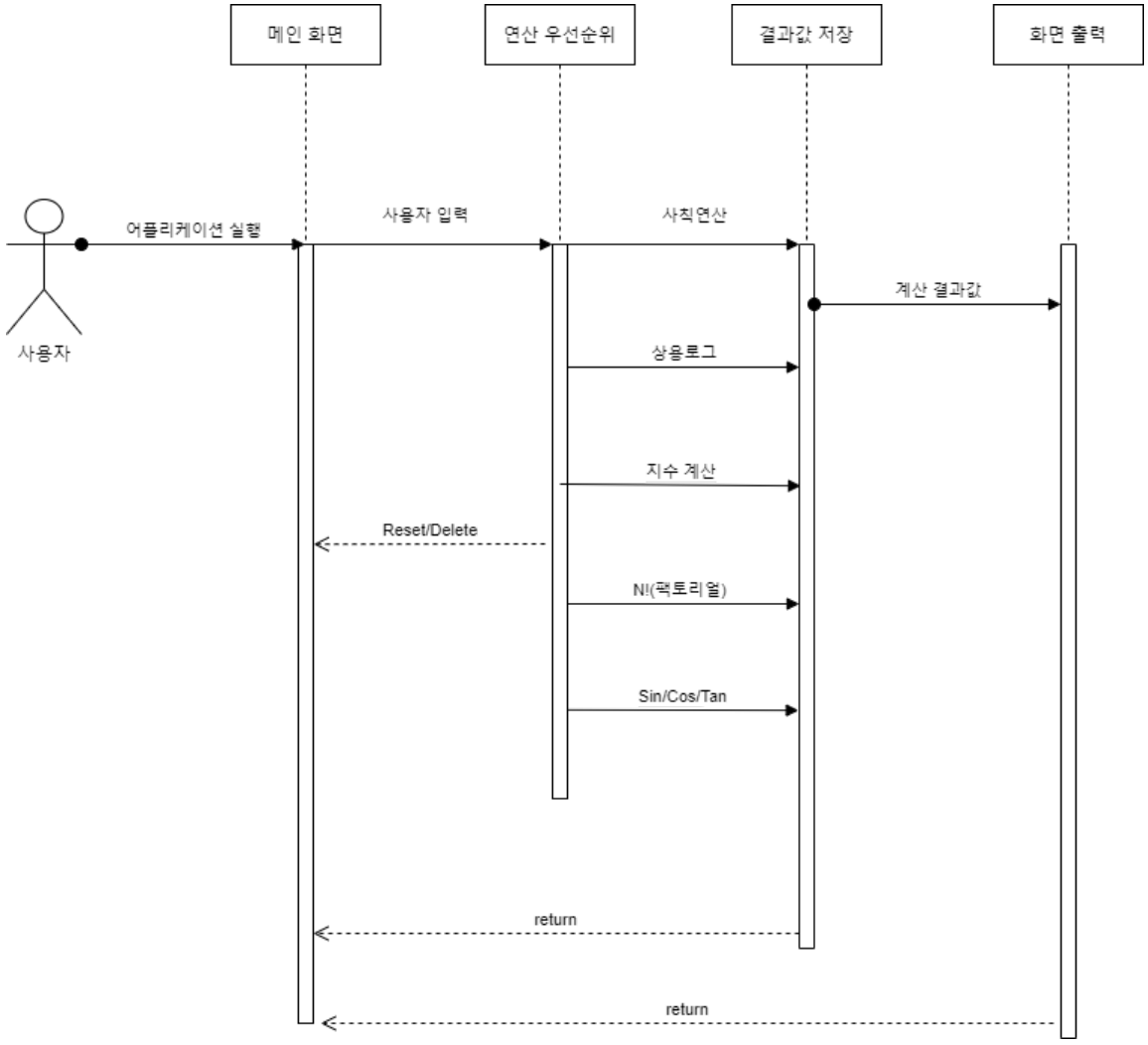
구조 설계

UML

- Use Case Diagram



- Sequence Diagram



- Class Diagram

클래스 및 함수 설명

☐ MainActivity

a) 설명

- 공학용 계산기를 나타는 메인 화면으로 크게 3가지로 구분됨
  i) 계산기 입력 화면 : TextView

ii) 연산자 , 피연산자 입력 버튼 : setNumListener , setOperListener
iii) 연산된 결과 화면 : 계산이 완료된 데이터를 화면에 보여준다.

b) 필드
i) 버튼

- btn0 ~ 9 : 피연산자에 해당하는 버튼필드로 계산을 필요로 하는 필드
- btnPlus , btnMinus , btnDiv , btnMulti, btnLog , btnFac , btnInvolution , btnFac : 피연산자에 해당하는 버튼 필드 (+ , - , * , / , % , ! ,^ , mod ,log ..)
- btnReset , btnEqual , btnDel , btnDot , btnLeftParenthesis , btnRightParenthesis : 연산자,피연산자 이외의 버튼으로 (초기화 , '=', 삭제 . '.' , '(' , ')')

ii) 화면

- outExDisplay : 입력한 버튼을 표현해 주는 textview
- historyDisplay : 계산된 결과를 보여주는 textview

iii) 데이터 입력

- input : 버튼 입력
- outexPress : 출력
- operExpress : 연산자 출력
- hisData : history data
- double result : 결과값 (소수점 한자리까지 출력)

c) 함수

- setNumListener() : 버튼을 눌렀을 경우 입력한 버튼을 텍스트뷰로 전달한다.
- setOperListener() : 연산자 버튼을 눌렀을 경우 입력한 버튼을 텍스트 뷰로 전달한다.
- init() : numListener ,operListener 등을 호출하는 함수

☐ FunctionParent

a) 설명

- CalculatorFunction, InputFunction, PriorityFunction 클래스의 부모 클래스

b) 필드

- operationPriorityMap : 연산자 우선순위를 저장한 Map

c) 함수

- isOperation : 연산자 여부를 확인한다.
- makeOperationPriority : 자식 클래스에서 오버라이딩을 한다.
- init : 자식 클래스에서 오버라이딩을 한다.

☐ InputFunction

a) 설명

- FunctionParent 의 자식 클래스
- 계산기에 값을 입력하고 '=' 연산자를 누르게 되면 가장 먼저 호출되는 클래스
- 계산기에 입력된 값이 String 으로 들어오게 되고 여기서는 이를 List 로 저장하게 된다.

(b) 필드

- inputList : 계산기에 입력하면 String 타입으로 입력 내용이 저장되고 이를 List 에 저장하기 위한 List
- PriorityFunction의 객체 선언

(c) 함수

- @Override / makeOperationPriority : 연산자의 우선순위에 맞게 key, value 를 넣어준다.
- @Override / isOperation : 부모의 isOperation을 호출한다.
- @Override / init : 부모의 init 함수를 오버라이딩하였고, inputList가 비어있는지를 확인하여 clear해주고 객체 선언을 해준다.
- makeInputStringToList : String 타입으로 들어온 input을 List로 바꾸는 함수이다. 이때 숫자가 한 자리 수가 아닌 경우를 위해서 lastIndex 를 설정했고, 만약 연산자가 아니고 List의 lastIndex에 해당하는 값이 숫자이면 lastIndex에 값을 합쳐주는 식으로 List를 만든다. 이후 PriorityFunction의 makePostfixOperation를 return 하는데 inputList를 매개변수로 넘겨준다.

☐ PriorityFunction

a) 설명

- FunctionParent 의 자식 클래스
- InputFunction 에서 List 로 저장한 값을

(b) 필드

- outputList : inputList를 후위 연산식으로 변환하여 이를 저장할 List
- postfixOperationStack : inputList를 후위 연산식으로 변환하기 위해 필요한 Stack
- CalculatorFunction의 객체 선언

(c) 함수

- @Override / isOperation : 부모의 isOperation을 호출한다.
- @Override / init : 부모의 init 함수를 오버라이딩하였고, outputList가 비어있는지를 확인하여 clear해주고 객체 선언을 해준다.
- makePostfixOperation : 후위연산식을 만들어준다. inputList에서 값을 하나씩 가져오면서 "("인 경우에는 postfixOperationStack에 push해주고 ")"가 나오게되면 "("가 나올때까지 stack에서 값을 가져와서 outputList에 넣어준다. 또한 postfixOperationStack의 peek 연산자 우선순위가 더 높거나 같으면 outputList에 넣어준다. 그 외에 숫자가 나오면 ouputList에 바로 값을 넣어서 후위연산식을 만들어주고 최종적으로 완성된 outputList를 CalculatorFunction의 getResult를 return 하는데 이때 매개변수로 넘겨준다.

☐ CalculatorFunction

a) 설명

- FunctionParent 의 자식 클래스

(b) 필드

- finalStack : 최종적으로 계산 시 필요한 Stack
- Operation의 객체 선언

(c) 함수

- @Override / isOperation : 부모의 isOperation을 호출한다.

- @Override / init : 부모의 init 함수를 오버라이딩하였고, finalStack의 사이즈가 1보다 크거나 같으면 stack을 비우기 위해 clear 해준다.
- getResult : 이제 최종적으로 outputList를 통해 숫자면 finalStack에 push해주고 연산자가 나오면 pop해주어 계산을 한다. 이때 두 숫자가 필요한 경우 finalStack 의 사이즈가 1보다 큰지 확인해야하고, 숫자 하나를 통해 연산을 해야하면 finalStack의 사이즈가 0보다 큰지 확인해주어야 한다. 이를 통해 최종적으로 계산된 값이 finalStack에 push되어져 있고 이를 return 해주면 최종 계산된 값을 확인할 수 있다.

☐ History

- MainActivity 에서 operExpress 와 result를 더해주고 이를 historyDis에 append 해준다.
- History에서 makeHistory 에서는 "\n"을 더해줌으로써 줄바꿈을 해준다.

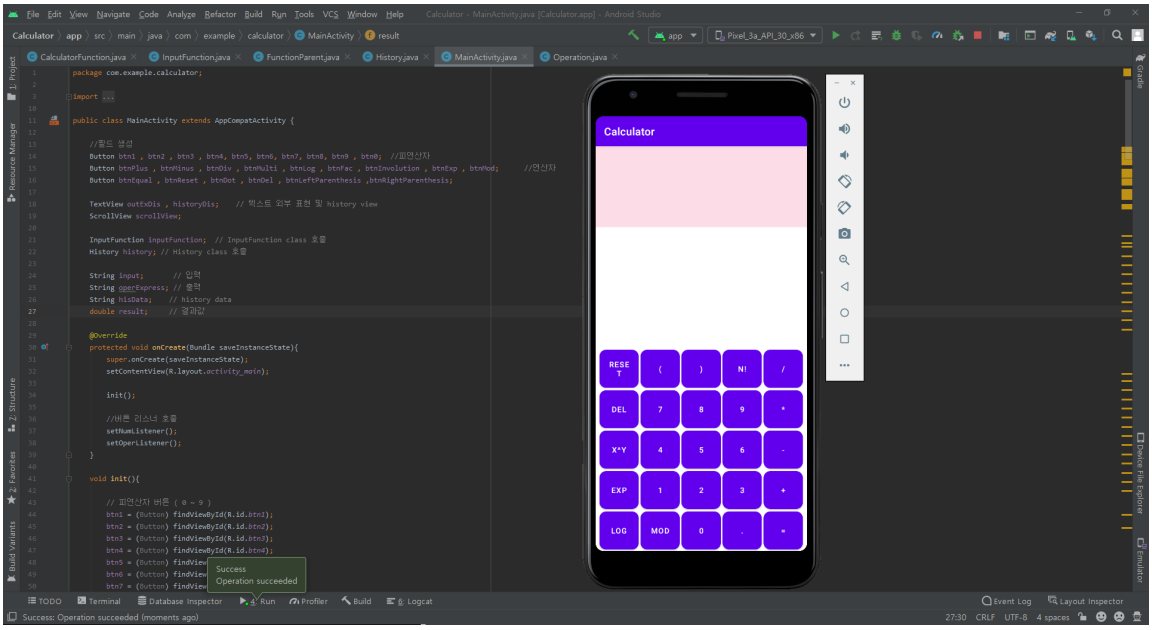
☐ Operation

- CalculatorFunction에서 계산할 때 Operation에 정의되어 있는 계산식을 호출하게 된다.

알고리즘 설계 및 구현

코딩 및 테스트

- 실행


Calculator
RESET
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=
Success
Operation succeeded

- 덧셈


Calculator
2+3=5.0
5.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 뺄셈


Calculator
2+3=5.0
5-6=-1.0
-1.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 곱셈


Calculator
2+3=5.0
5-6=-1.0
8*9=72.0
72.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 나눗셈


Calculator
6mod5=1.0
8*4=32.0
8/4=2.0
2.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 괄호 계산


Calculator
2+3=5.0
5-6=-1.0
8*9=72.0
(7+8)*2=30.0
30.0
RESET
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 나머지


Calculator
6mod5=1.0
1.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 로그


Calculator
5-6=-1.0
8*9=72.0
(7+8)*2=30.0
log2=0.3010299956639812
0.3010299956639812
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 지수


Calculator
(7+8)*2=30.0
log2=0.3010299956639812
4^2=16.0
exp2=7.38905609893065
7.38905609893065
RESET
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 팩토리얼


Calculator
6mod5=1.0
8*4=32.0
8/4=2.0
4!=24.0
24.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=

- 거듭제곱


Calculator
8*9=72.0
(7+8)*2=30.0
log2=0.3010299956639812
4^2=16.0
16.0
RESE T
(
)
N!
/
DEL
7
8
9
*
X^Y
4
5
6
-
EXP
1
2
3
+
LOG
MOD
0
.
=