PyQt6 기반 계산기 시스템

괄호 처리, 연산자 우선순위, 상태 기반 입력 흐름을 지원하는 GUI 계산기

발표자: 장진혁

1. GUI 화면 구성

1-1. 디스플레이

구성 요소	설명	Qt 객체명
상부 디스플레이	사용자가 입력한 전체 수식 표시	lineEdit_2
하부 디스플레이	현재 입력 중인 숫자 또는 계산 결과 표시	lineEdit
텍스트 메시지 영역	입력 오류 발생 시 메시지를 2초간 표시	textBrowser

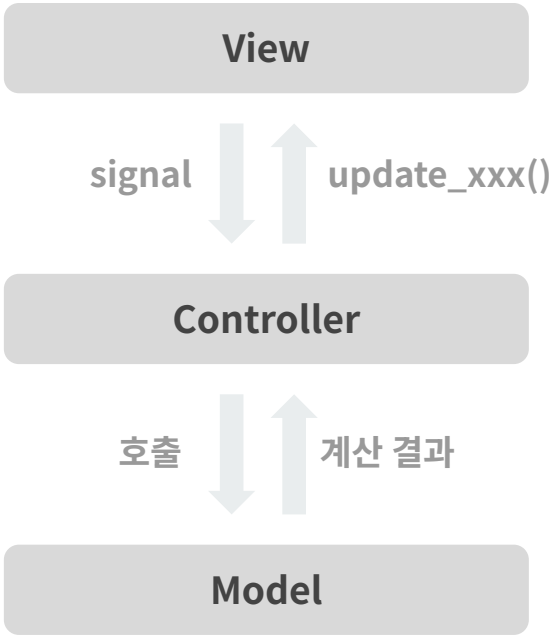
Note
입력 오류, 잘못된 괄호, 중복 소수점 등 **사용자 실수에 대한 안내 메시지**를 **텍스트 메시지 영역**에 표시합니다. QTimer를 통해 2초 후 자동으로 사라지도록 처리하여 UX를 개선했습니다.

1-2. 버튼

버튼 분류	연결된 동작	호출 함수 (Slot)
숫자 버튼	숫자 입력	handle_digit('7')
소수점 버튼	소수점 입력	handle_digit('.')
연산자 버튼	연산자 입력	handle_operator()
괄호 버튼	괄호 입력	handle_lparen(), handle_rparen()
부호 전환 버튼	부호 전환	handle_sign()
등호 버튼	수식 계산	handle_equal()
C 버튼	부분 삭제	handle_c()
AC 버튼	전체 초기화	handle_ac()

2. 설계 특징

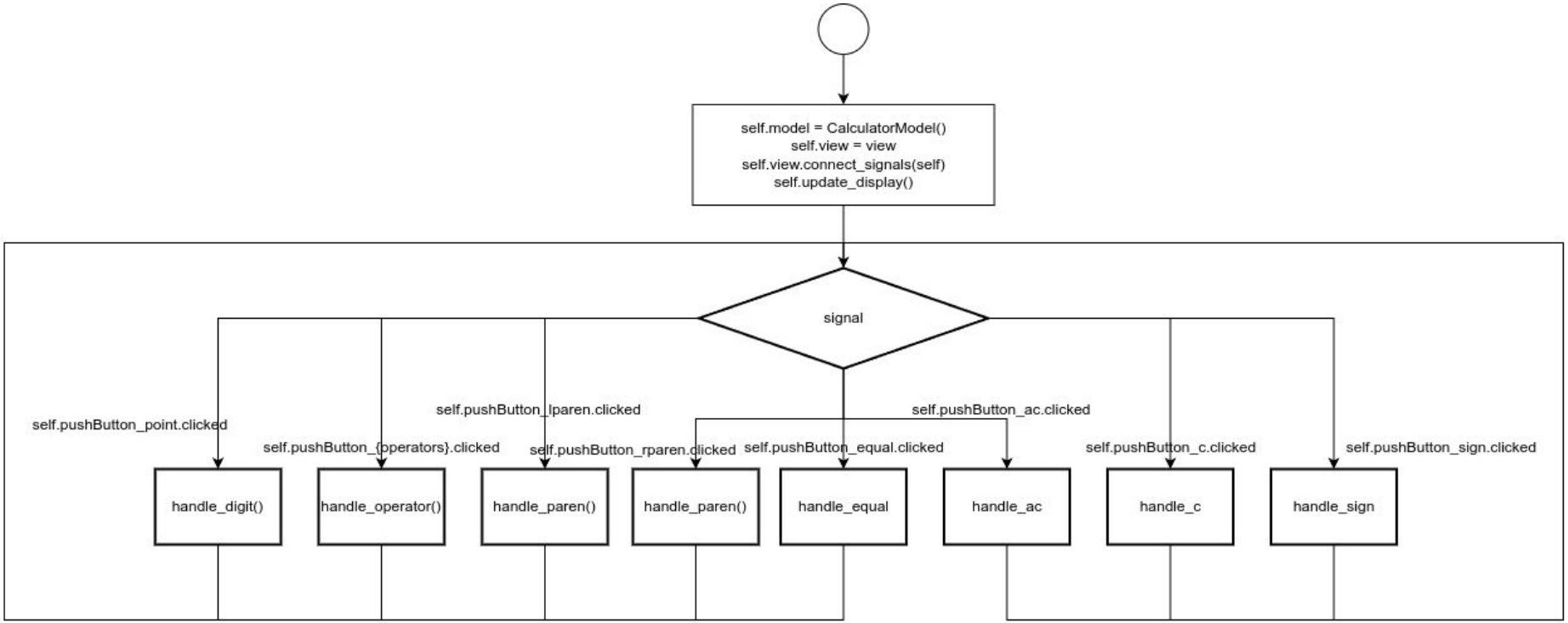
2-1. MVC 구조



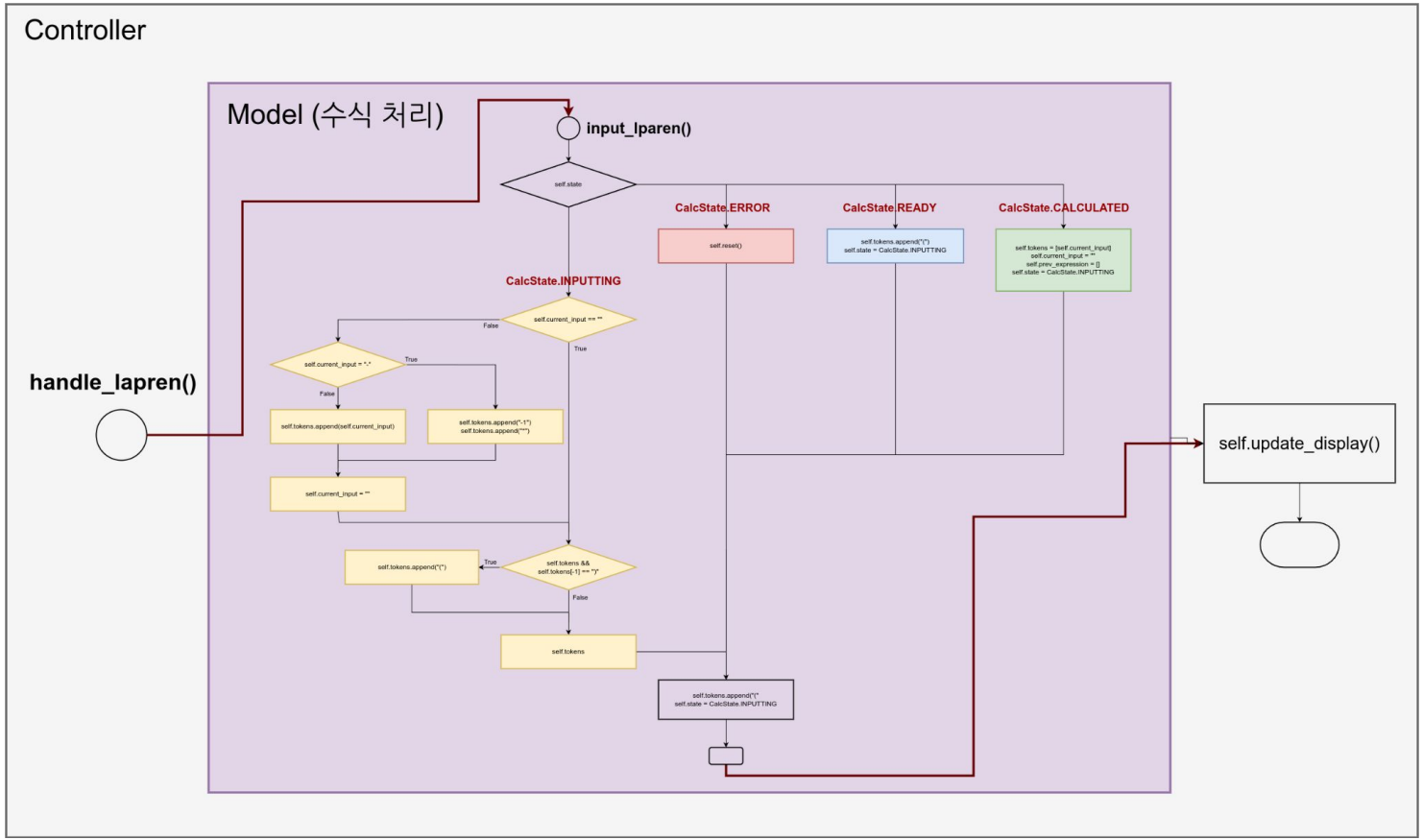
핵심 내용	• 전체 프로그램을 Model, View, Controller로 역할 분리 • Model과 View 모듈은 서로 직접 접근하지 않고 Controller를 통해 간접 연결

각 요소별 설명	
Controller	**중심 허브**. View와 Model 모두를 알고 있고, 호출과 반영을 직접 처리함.
Model	**계산 로직, 상태 처리 전담**. 직접 UI 업데이트는 하지 않음.
View	**사용자 인터페이스와 signal 전달자 역할**. 자체 로직 없음.

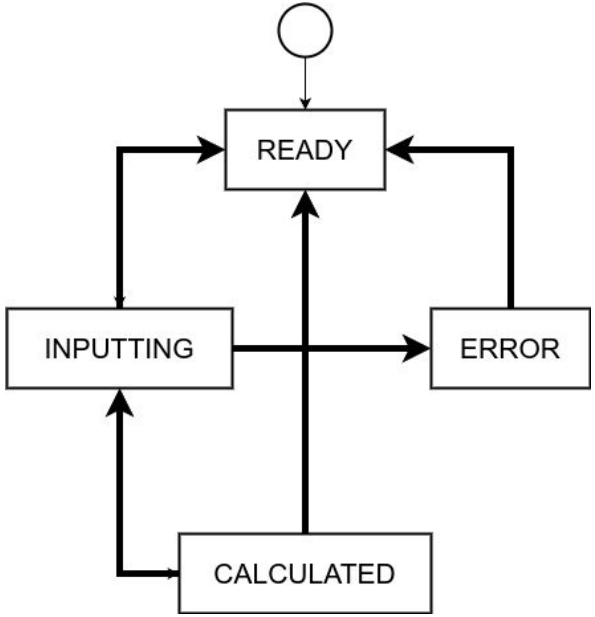
2-1-1. Controller 내부 동작



2-1-2. Model 호출 및 상태 처리 흐름



2-2. 상태 기반 입력 처리



핵심 내용	• **CalcState 상태(Enum)** ◦ READY, INPUTTING, CALCULATED, ERROR • **각 상태에 따라 입력 흐름을 제어함** ◦ CALCULATED 상태에서 숫자 입력 시 자동 초기화 ◦ ERROR 상태에서 입력 시 자동 리셋 • **입력 무결성 유지 및 사용자 의도 반영 설계**
예시 흐름	• **계산 완료(CALCULATED)** → 숫자 입력 → 수식 초기화 후 입력 시작 • **에러 발생(ERROR)** → 숫자 입력 → 전체 리셋 후 입력 시작

2-3. 입력-연산 파이프라인

숫자/연산자 입력

↓

tokens 리스트에 저장

↓

후위 표기법으로 변환

↓

스택 기반 계산

↓

결과 출력

핵심 내용	• **to_postfix()**에서 **후위 표기법**으로 변환 • **evaluate_postfix()**에서 **스택 계산** 수행 • **연산자 우선순위 적용** ◦ *, / 에 +, - 보다 높은 우선순위 적용 • **괄호 중첩, 소수점 연산 등 복합 수식 지원**
효과	• **복잡한 수식도 정확하게 계산** • **표준 계산기 수준의 수식 평가 기능 구현**

2-4. QTimer 기반 UX 개선

핵심 내용	• **입력 오류 발생 시 메시지 출력** ◦ textBrowser.setText() • **2초 후 자동 메시지 삭제** ◦ QTimer.singleshot()
메시지 출력 대상	• 괄호 개수 불일치 • 중복 소수점 입력 • 연산자 중복 입력 등
UX 개선 효과	• 오류 메시지 수동 제거 불필요 • 자연스럽고 끊김 없는 입력 흐름 유지 • 사용자 실수는 알려주되, 흐름은 방해하지 않음
메시지 예시	• 괄호 뒤에는 '-'만 입력 가능합니다. • 빈 괄호가 자동으로 채워집니다. • 수식에 오류가 있습니다.

3. 기능 구현

3-1. 주요 기능

주요 기능

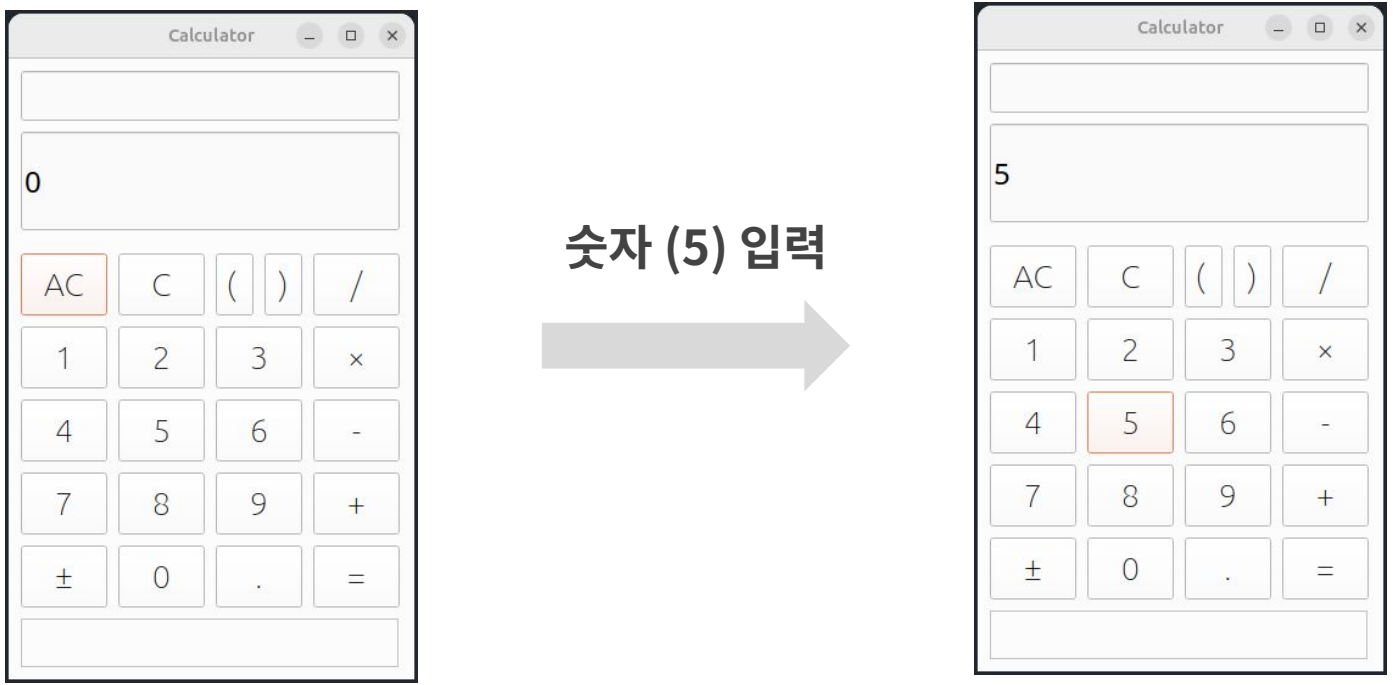
#	분류	System Requirements	
1	**입력 처리 기능**	SR_01	숫자를 입력하는 기능
		SR_02	소수점을 입력하고 중복을 방지하는 기능
		SR_03	± 부호를 토글하는 기능
		SR_04	연산자를 입력하고 중복을 처리하는 기능
		SR_05	괄호를 입력하고 곱셈을 자동 처리하는 기능
		SR_06	괄호의 개수를 검증하고 빈 괄호를 방지하는 기능
2	**입력 및 상태 초기화 기능**	SR_07	계산 후 새로운 입력을 초기화하는 기능
		SR_08	부분 삭제(C)를 수행하는 기능
		SR_09	전체 초기화(AC)를 수행하는 기능

주요 기능

#	분류	System Requirements	
3	**계산 처리 기능**	SR_10	중위 수식을 후위 표기법으로 변환하는 기능
		SR_11	후위 표기 수식을 스택으로 계산하는 기능
4	**예외 처리 기능**	SR_12	괄호와 연산자를 자동 보완하는 기능
		SR_13	입력 UX 보완 메시지를 출력하는 기능
		SR_14	에러 발생 시 오류 메시지를 출력하는 기능
5	**결과 출력 기능**	SR_15	수식과 결과를 라인에 나누어 출력하는 기능
		SR_16	정수 결과 시 .0을 제거하여 출력하는 기능

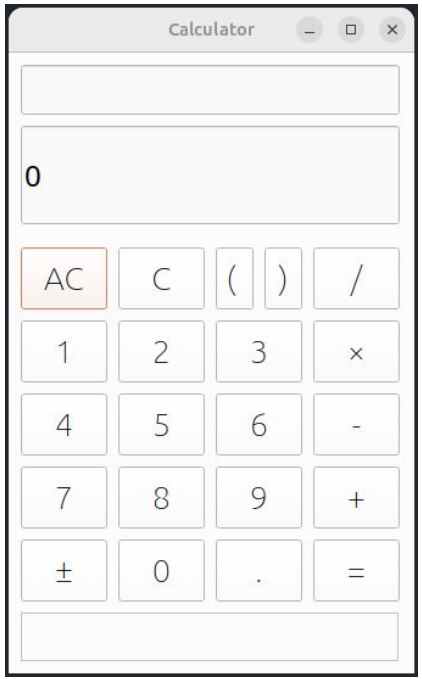
3-2. 입력 처리 기능

SR_01. 숫자(0-9)를 입력하는 기능

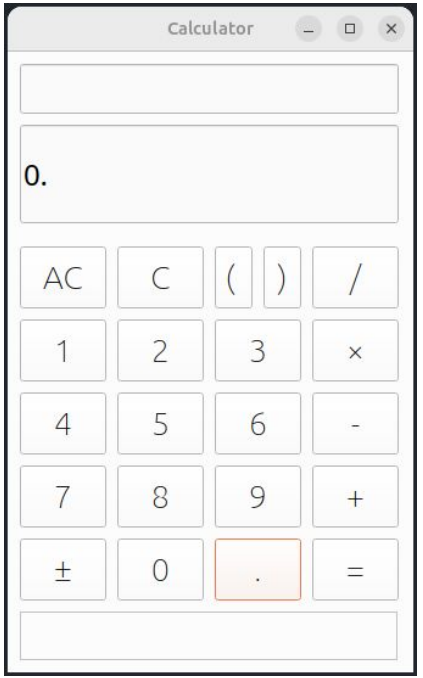


기능 요약	0~9 숫자 버튼 클릭 시 숫자를 current_input에 추가

SR_02. 소수점을 입력하고 중복을 방지하는 기능

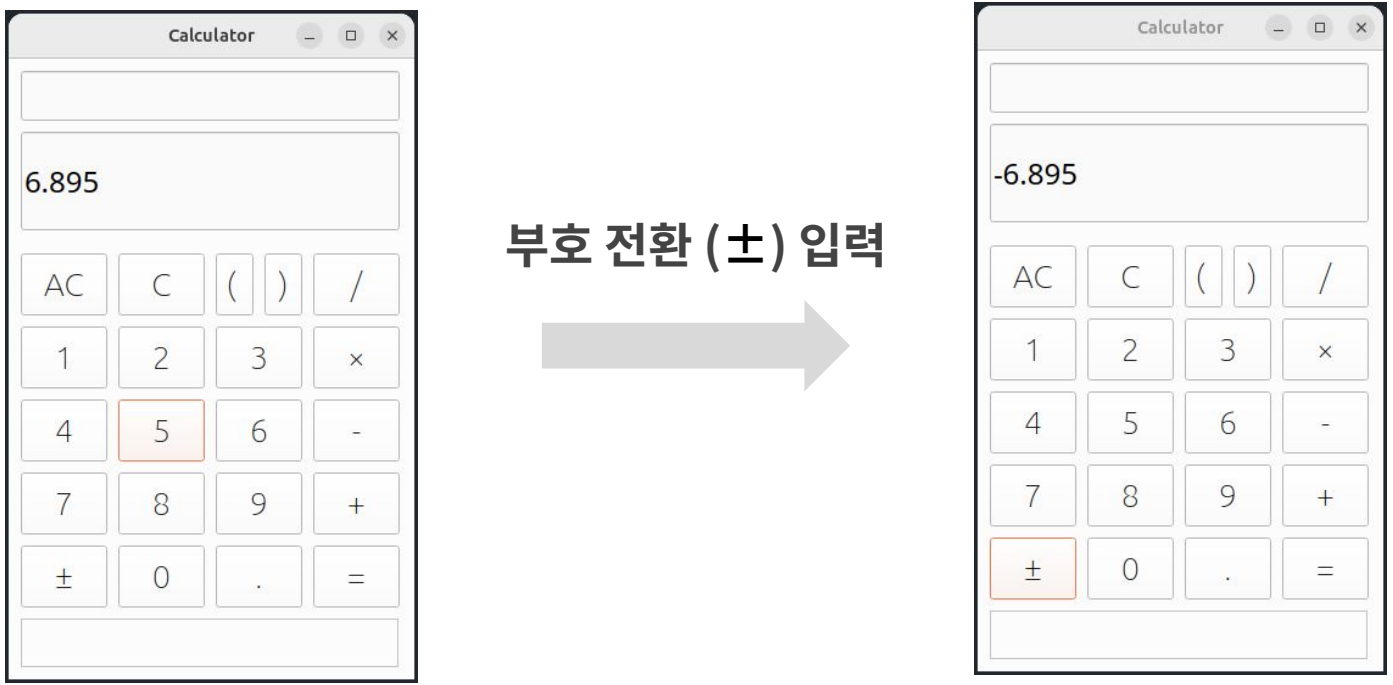


소수점 (.) 입력



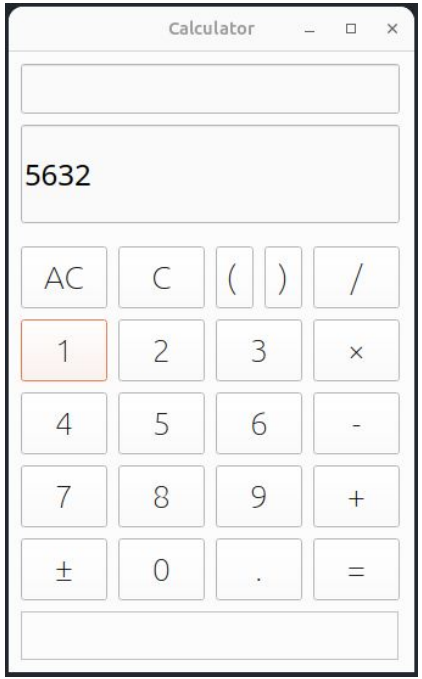
기능 요약	‘.’ 버튼 입력 시 ‘0.’ 또는 ‘-0.’으로 자동 보완 소수점 중복 입력 차단

SR_03. ± 부호를 토글하는 기능



기능 요약	부호 전환 버튼 클릭 시 현재 입력 수에 '-' 부호를 토글

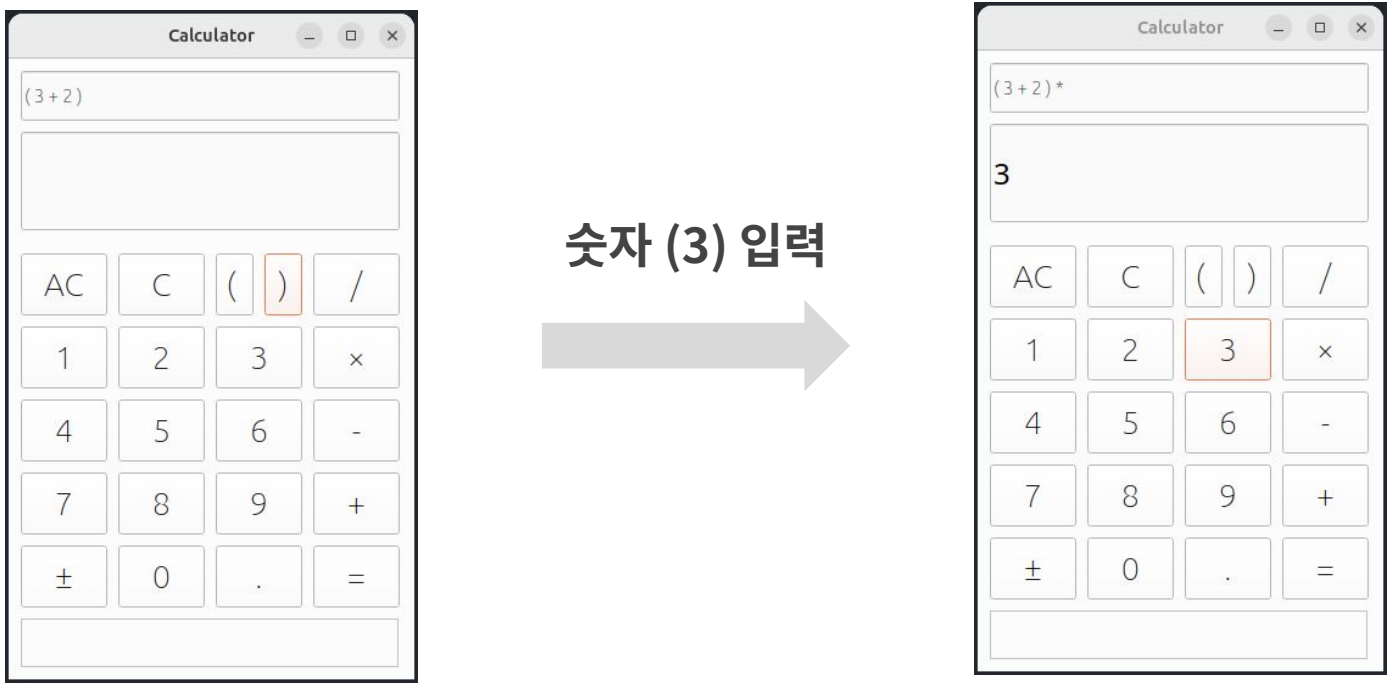
SR_04. 연산자를 입력하고 중복을 처리하는 기능



연산자 (+) 입력

기능 요약	+, -, *, / 연산자 버튼 클릭 시 현재 수식 뒤에 연산자 추가 이전에 연산자가 있으면 새 연산자로 덮어씀

SR_05. 괄호를 입력하고 곱셈을 자동 처리하는 기능



기능 요약	숫자 뒤 또는 닫는 괄호 뒤에 여는 괄호 입력시 * 자동 추가

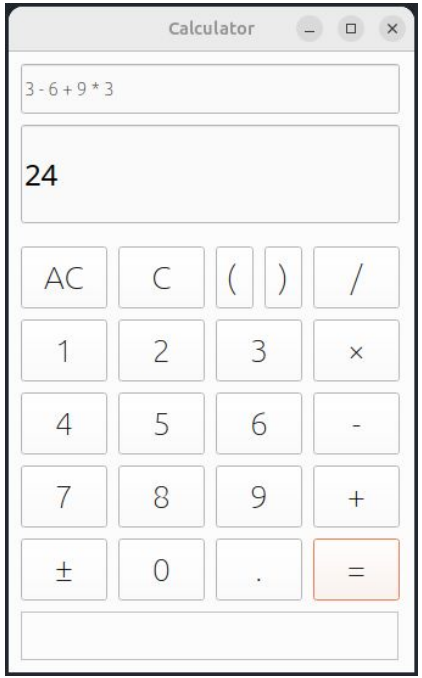
SR_06. 빈 괄호를 방지하는 기능

닫는 괄호 ( )) 입력

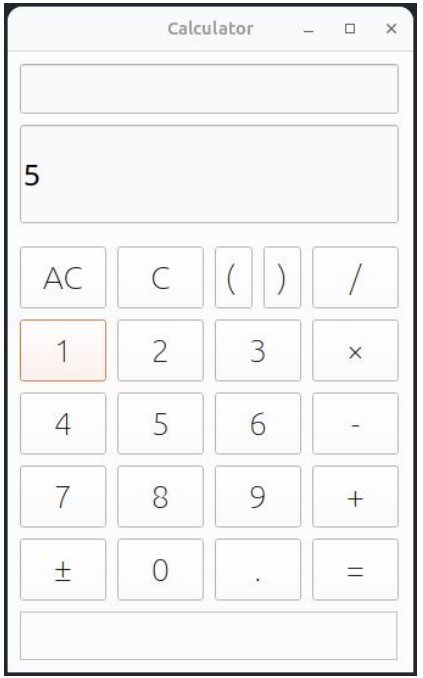
기능 요약	빈 괄호 입력 방지

3-3. 입력 및 상태 초기화 기능

SR_07. 계산 후 새로운 입력을 초기화하는 기능

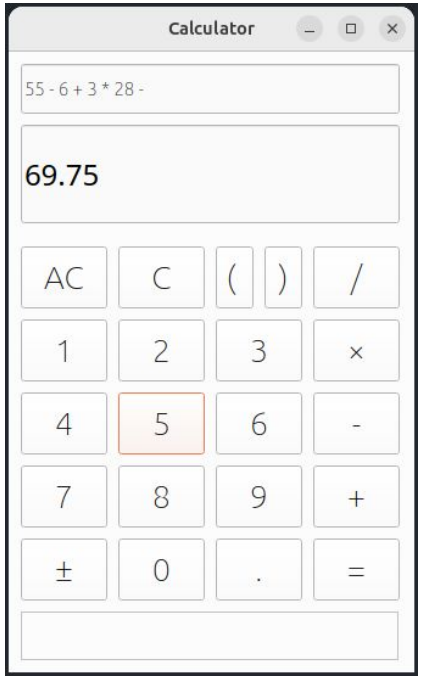


숫자 (5) 입력



기능 요약	계산 완료 상태(CALCULATED)에서 숫자 입력 시 기존 수식을 초기화하고 새 입력 시작

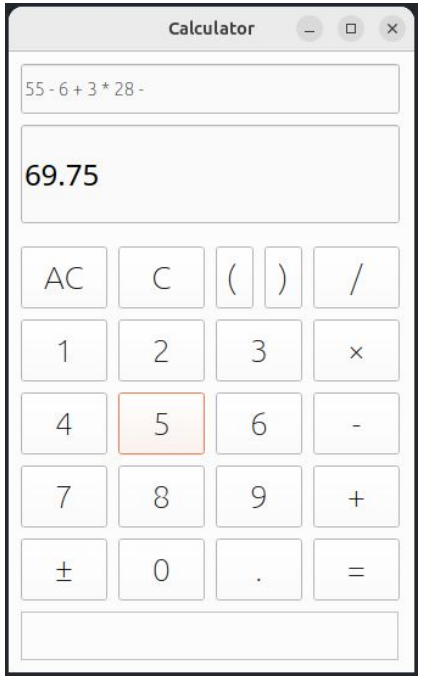
SR_08. 부분 삭제(C)를 수행하는 기능



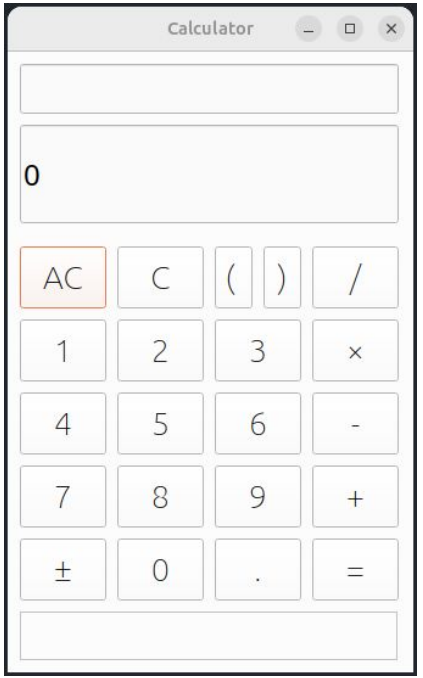
부분 삭제 (C) 입력

기능 요약	C 버튼은 최근 입력만 제거, AC 버튼은 수식 전체 초기화

SR_09. 전체 초기화(AC)를 수행하는 기능



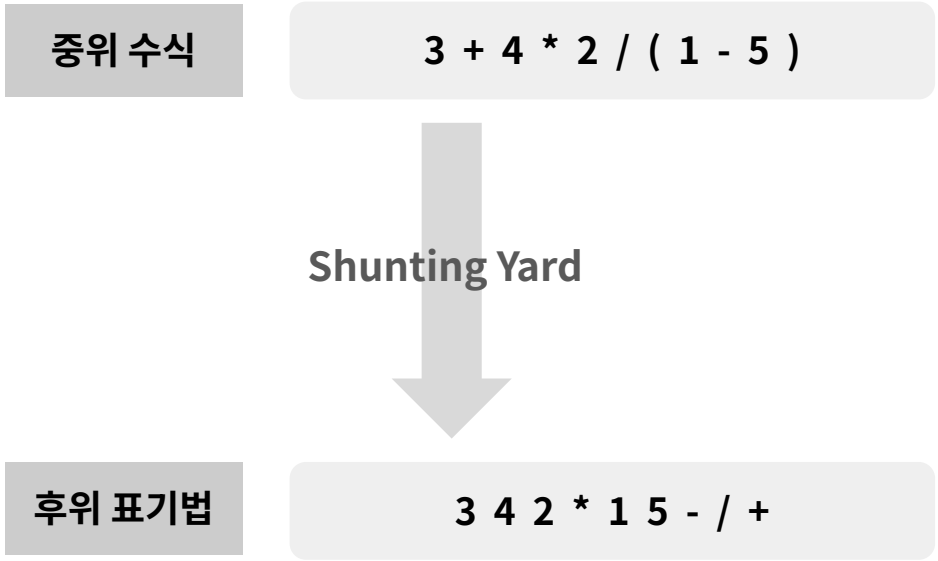
전체 초기화 (AC) 입력



기능 요약	C 버튼은 최근 입력만 제거, AC 버튼은 수식 전체 초기화

3-4. 계산 처리 기능

SR_10. 중위 수식을 후위 표기법으로 변환하는 기능



기능 요약	Shunting Yard 알고리즘으로 중위 수식을 후위 표기법으로 변환

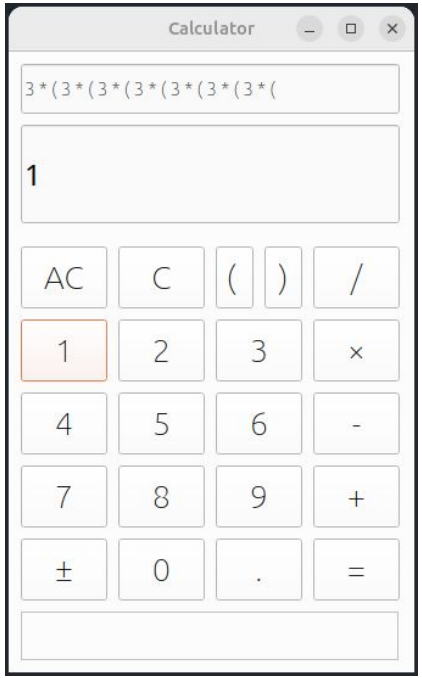
SR_11. 후위 표기 수식을 스택으로 계산하는 기능

토큰	3	4	2	*	1	5	-	/	+
스택 상태						5			
			2		1	1	-4		
		4	4	8	8	8	8	-2	
	3	3	3	3	3	3	3	3	1

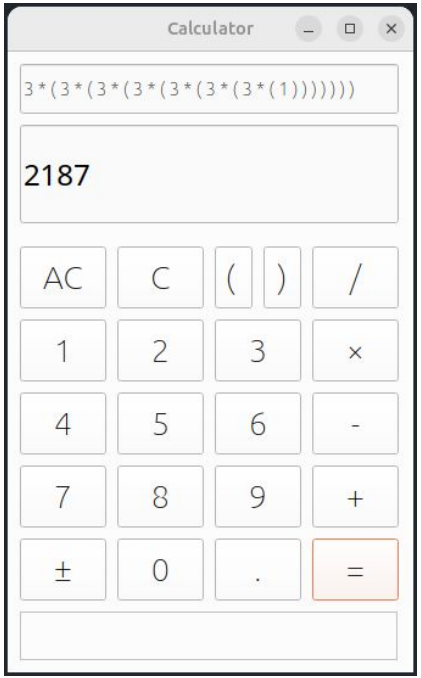
기능 요약	변환된 후위 표기 수식을 스택을 이용해 계산

3-5. 예외 처리 기능

SR_12. 괄호와 연산자를 자동 보완하는 기능

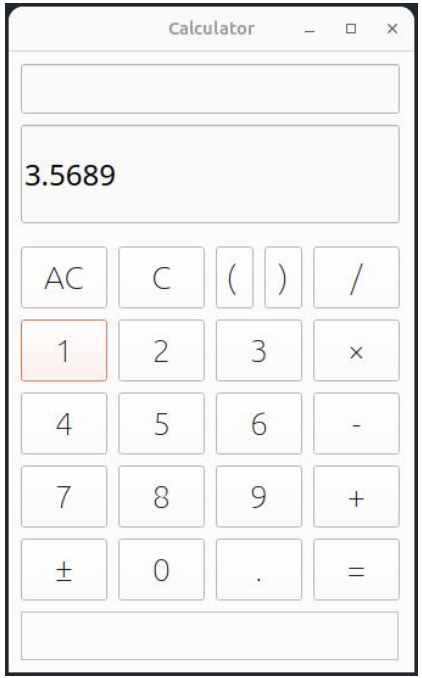


등호 (=) 입력

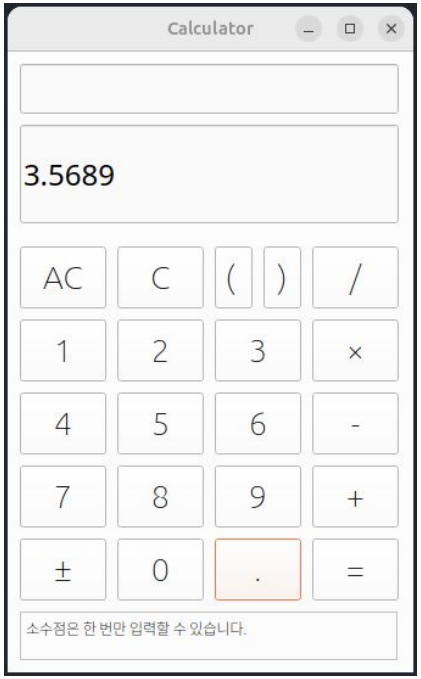


기능 요약	계산 시도 시, 자동으로 괄호 쌍을 맞추고, 수식 말단의 불완전한 연산자를 제거하여 계산 가능하도록 보장함

SR_13. 입력 UX 보완 메시지를 출력하는 기능

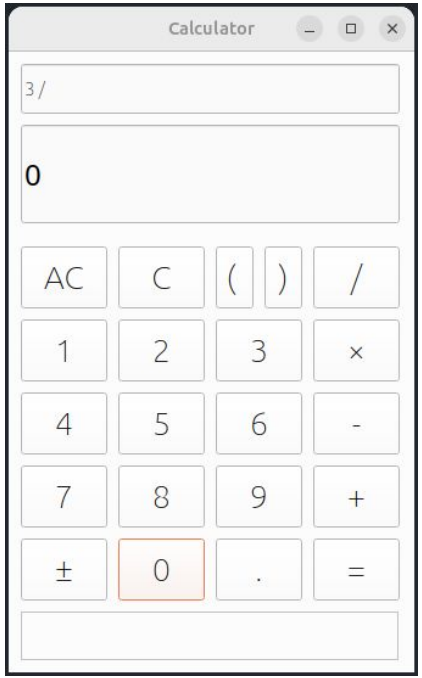


소수점 (.) 입력

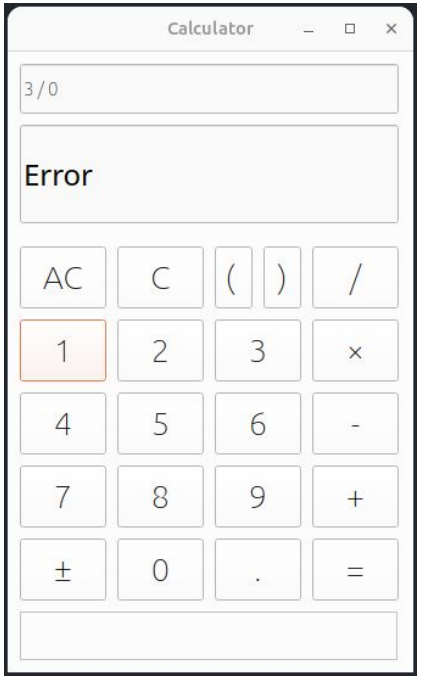


기능 요약	사용자 입력 실수(소수점 중복, 연산자 누락 등)에 대해 즉시 메시지 출력

SR_14. 에러 발생 시 오류 메시지를 출력하는 기능



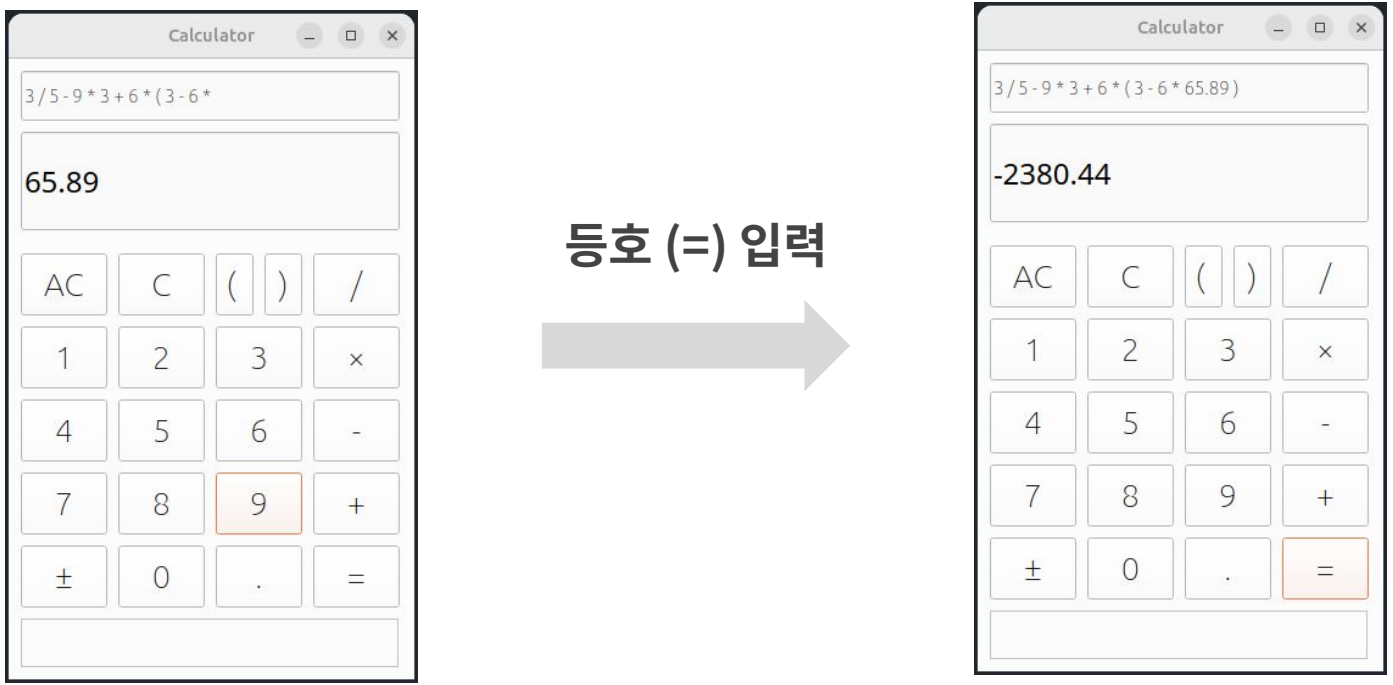
등호 (=) 입력



기능 요약	evaluate 중 오류 발생 시 사용자에게 메시지를 자동 출력

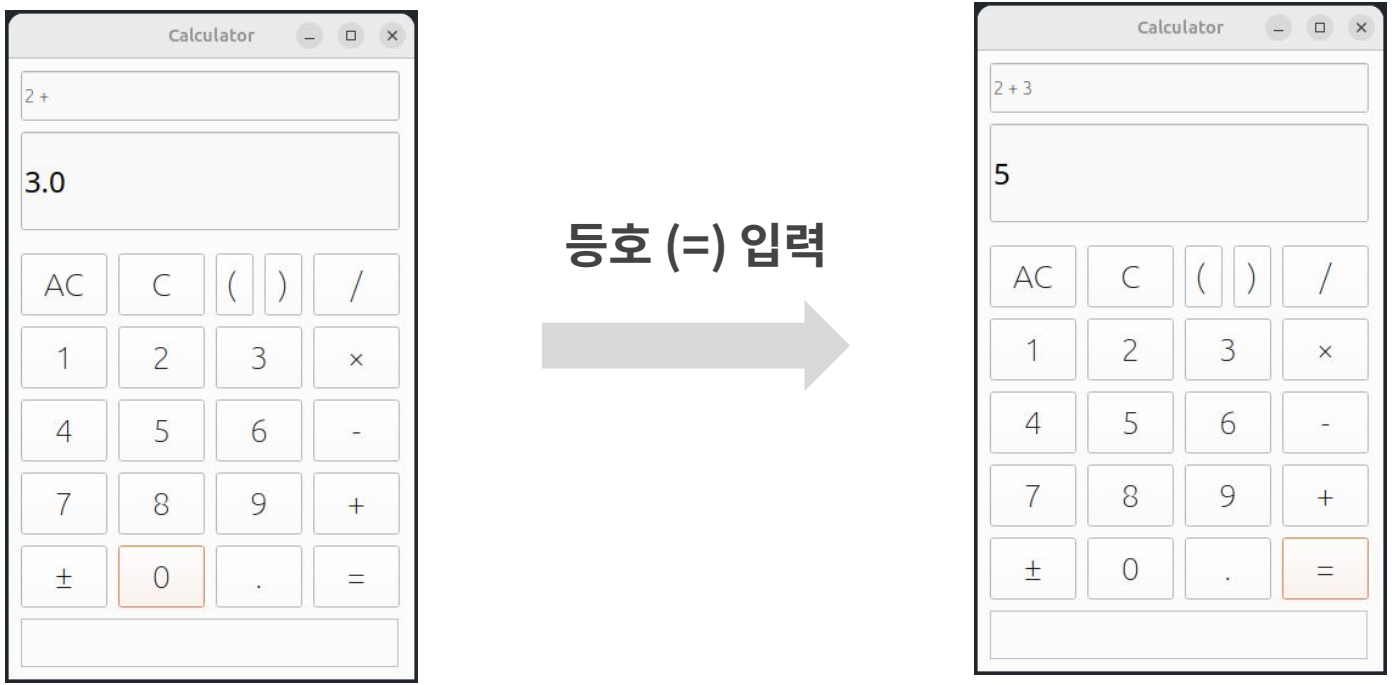
3-6. 결과 출력 기능

SR_15. 수식과 결과를 라인에 나누어 출력하는 기능



기능 요약	수식과 결과를 각각 다른 영역(lineEdit, lineEdit_2)에 분리하여 출력

SR_16. 정수 결과 시 .0을 제거하여 출력하는 기능

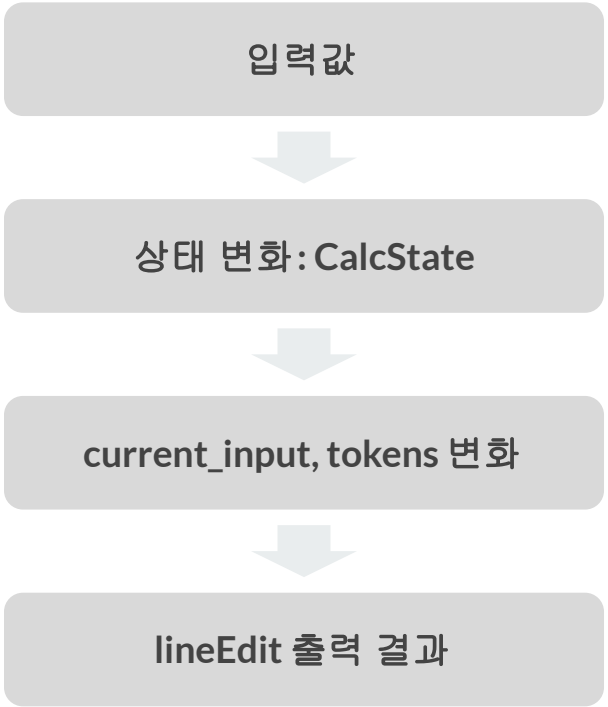


기능 요약	계산 결과가 .0으로 끝나는 정수일 경우, .0을 제거하여 깔끔하게 출력

4. 테스트 및 검증

4-1. 테스트 설계

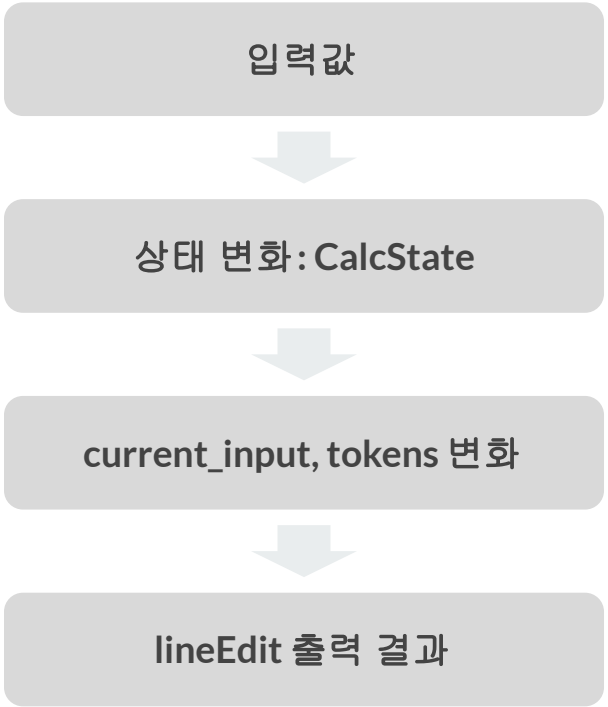
4-1. 테스트 설계



- 총 **6가지 기능 분류** 기준
- 총 **45개**의 **테스트 시나리오**

기능 분류	초기 상태 입력 흐름	9
	오류 직후 입력 복구 흐름	9
	계산 후 입력 처리 흐름	9
	괄호/우선순위 처리	8
	에러 처리 및 에러 방지	6
	초기화 흐름	4

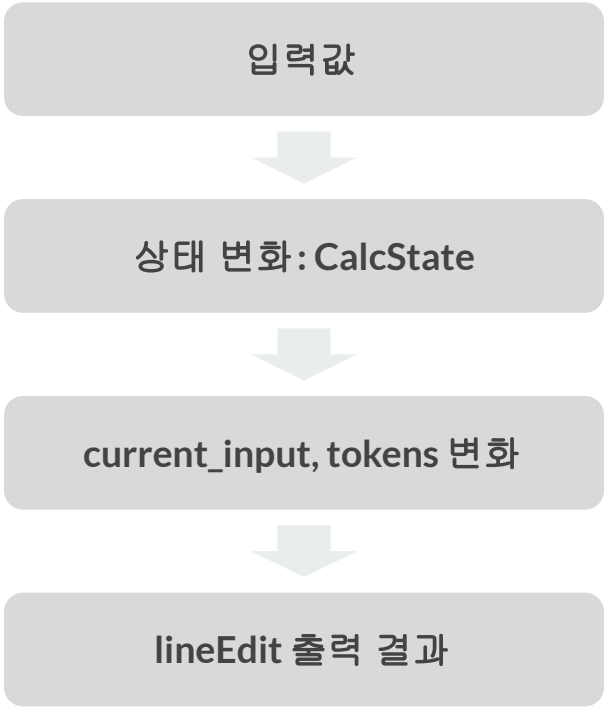
4-1. 테스트 설계



각 테스트는 **하나의 입력**을 기준으로, **내부 상태 변화**와 **화면 출력**까지의 전체 흐름을 검증

이 흐름에 따라 **예상 동작과 실제 결과를 비교**하여
기능의 정확성 및 안정성을 판단함

4-1. 테스트 설계



테스트 결과 요약 (4-2)

45개 기능 시나리오의 정상 작동 여부를 확인

테스트 케이스 (정상흐름) (4-3)

대표 TC의 내부 상태 흐름 추적 분석

테스트 케이스 (예외상황) (4-4)

잘못된 입력에 대한 시스템 보완 및 UX 보호 기능 검증

4-2. 테스트 결과 요약

4-2-1. 초기 상태 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_01	초기상태 → 입력 없음	없음	없음	기본값(0)	✅
TC_02	초기상태 → AC 또는 C 입력	AC 또는 C	없음	기본값(0)	✅
TC_03	초기상태 → = 입력	=	없음	기본값(0)	✅
TC_04	초기상태 → (입력	(	(	없음	✅
TC_05	초기상태 →) 입력	)	없음	기본값(0)	✅
TC_06	초기상태 → ± 입력	±	없음	기본값(0)	✅
TC_07	초기상태 → 소수점 입력	.	없음	0.	✅
TC_08	초기상태 → 0 2회 입력	0 → 0	없음	기본값(0)	✅
TC_09	초기상태 → 0이 아닌 숫자 입력	1부터 9까지의 임의의 수	없음	입력과 동일	✅

4-2-2. 오류 직후 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_10	오류 상태 → 숫자 입력	1/0= → 5	없음	5	✅
TC_11	오류 상태 → 연산자 입력	1/0= → +	없음	기본값(0)	✅
TC_12	오류 상태 → (입력	1/0= → (	(	없음	✅
TC_13	오류 상태 →) 입력	1/0= →)	없음	기본값(0)	✅
TC_14	오류 상태 → 소수점 입력	1/0= → .	없음	0.	✅
TC_15	오류 상태 → ± 입력	1/0= → ±	없음	기본값(0)	✅
TC_16	오류 상태 → C 입력	1/0= → C	없음	기본값(0)	✅
TC_17	오류 상태 → AC 입력	1/0= → AC	없음	기본값(0)	✅
TC_18	오류 상태 → = 입력	1/0==	없음	기본값(0)	✅

4-2-3. 계산 직후 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_19	계산 직후 → 숫자 입력	1+2= → 5	없음	5	✅
TC_20	계산 직후 → 연산자 입력	1+2= → +	3 +	없음	✅
TC_21	계산 직후 → (입력	1+2= → (	3 * (	없음	✅
TC_22	계산 직후 →) 입력	1+2= →)	1+2	3	✅
TC_23	계산 직후 → 소수점 입력	1+2= → .	없음	0.	✅
TC_24	계산 직후 → ± 입력	1+2= → ±	없음	-3	✅
TC_25	계산 직후 → C 입력	1+2= → C	없음	기본값(0)	✅
TC_26	계산 직후 → AC 입력	1+2= → AC	없음	기본값(0)	✅
TC_27	계산 직후 → = 입력	1+2= → =	3	3	✅

4-2-4. 우선순위 및 괄호

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_28	곱셈 우선	1+2*3=	1+2*3	7	✅
TC_29	괄호로 우선순위 변경	(1+2)*3=	(1+2)*3	9	✅
TC_30	중첩 괄호 계산	(1+(2+3))*2=	(1+(2+3))*2	12	✅
TC_32	괄호 안 곱셈 먼저	1+(2*3)=	1+(2*3)=	7	✅
TC_33	괄호만 사용	((3))=	((3))	3	✅
TC_34	암시적 곱셈 처리	2(2+3)4=	2*(2+3)*4	20	✅

4-2-5. 에러 감지 및 에러 방지

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_35	0으로 나누기	1/0=	1/0	Error	✅
TC_36	괄호 생략 곱셈 테스트	2(3+4)=	2*(3+4)	14	✅
TC_37	수식이 연산자로 끝남	5+ → =	5	5	✅
TC_38	자동 괄호 닫기	(1+(1+=	(1+(1))	2	✅
TC_39	괄호 이중 닫힘	(2+3) →)	(2+3)	없음	✅
TC_40	괄호 안 연산자로 끝남	(4+ →)	(4)	없음	✅
TC_41	연속 연산자 입력	5+ → *3=	5 * 3	15	✅
TC_42	소수점 중복 입력	2.3 → .	없음	2.3	✅

4-2-6. 초기화

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_43	하단 입력 상태에서 AC 입력	2+(3 → AC	없음	기본값(0)	✅
TC_44	하단 입력 상태에서 C 입력	2+(3 → C	2+(	기본값(0)	✅
TC_45	하단 입력 없는 상태에서 C 입력	2+(→ C	2+	기본값(0)	✅

4-3. 테스트 케이스 (정상 흐름)

4-3. 테스트 케이스 (정상 흐름)

TC_ID	시나리오	검증 포인트
TC_10	오류 상태에서 숫자 입력 시 정상 입력 흐름으로 전환	오류 복구 처리 및 상태 초기화 확인
TC_20	계산 직후 연산자 입력 시 수식이 이어지는지 확인	계산 결과의 연산 연장 처리 확인
TC_22	계산 직후 여는 괄호 입력 시 수식 확장 흐름 확인	계산 직후 여는 괄호 입력시 자동 곱셈 여부 확인
TC_30	중첩 괄호 수식 계산	Shunting Yard 기반 우선순위 처리 검증
TC_34	숫자 뒤 괄호 입력 시 자동 곱셈 삽입 여부 확인	암시적 곱셈 처리 로직 검증

단순 기능 검증만으로는 내부 로직의 정확성을 완전히 판단하기 어려움

따라서 **대표 TC 5개**를 선정하여 **내부 상태(state, tokens, current_input)**를 단계별로 추적

이를 통해 **정상 흐름**, **괄호 처리**, **에러 복구**, **UX 보호**의 정확성 검증

4-3-1. 오류 상태에서 숫자 입력 (TC_10)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	(	INPUTTING	1	[]	-	1
2	/	INPUTTING	-	[‘1’, ‘/’]	1/	-
3	0	INPUTTING	0	[‘1’, ‘/’]	1/	0
4	=	ERROR	Error	[‘1’, ‘/’, ‘0’]	1/0	Error
5	5	INPUTTING	5	[]	-	5

4-3-2. 계산 직후 연산자 입력 (TC_20)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	1	INPUTTING	1	[]	-	1
2	+	INPUTTING	-	[‘1’, ‘+’]	1+	-
3	2	INPUTTING	2	[‘1’, ‘/’]	1/	2
4	=	CALCULATED	3	[‘1’, ‘+’, ‘2’]	1+2	3
5	+	INPUTTING	-	[‘3’, ‘+’]	3+	-

4-3-3. 계산 직후 괄호 입력 (TC_22)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	1	INPUTTING	1	[]	-	1
2	+	INPUTTING	-	[‘1’, ‘+’]	1+	-
3	2	INPUTTING	2	[‘1’, ‘/’]	1/	2
4	=	CALCULATED	3	[‘1’, ‘+’, ‘2’]	1+2	3
5	(	INPUTTING	-	['3', '*', '(']	3*(	-

4-3-4. 중첩 괄호 수식 계산 (TC_30)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	(	INPUTTING	-	['(']	(	-
2	1	INPUTTING	1	['(']	(	1
3	+	INPUTTING	-	['(', '1', '+']	(1+	-
4	(	INPUTTING	-	['(', '1', '+', '(']	(1+(	-
5	2	INPUTTING	2	['(', '1', '+', '(']	(1+(	2
6	+	INPUTTING	-	['(', '1', '+', '(', '2', '+']	(1+(2+	-
7	3	INPUTTING	3	['(', '1', '+', '(', '2', '+']	(1+(2+	3
8	)	INPUTTING	-	['(', '1', '+', '(', '2', '+', '3', ')']	(1+(2+3)	-
9	)	INPUTTING	-	['(', '1', '+', '(', '2', '+', '3', ')', ')']	(1+(2+3))	-
10	2	INPUTTING	2	['(', '1', '+', '(', '2', '+', '3', ')', ')', '*']	(1+(2+3))*	2
11	=	CALCULATED	12	['(', '1', '+', '(', '2', '+', '3', ')', ')', '*', '2']	(1+(2+3))*2	12

4-3-5. 숫자 뒤 괄호 입력 입력 (TC_34)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	2	INPUTTING	2	-	-	2
2	(	INPUTTING	-	['2', '*', '(']	2*(	-
3	3	INPUTTING	3	['2', '*', '(']	2*(	3
4	+	INPUTTING	-	['2', '*', '(', '3', '+']	2*(3+	-
5	4	INPUTTING	4	['2', '*', '(', '3', '+']	2*(3+	4
6	)	INPUTTING	-	['2', '*', '(', '3', '+', '4', ')']	2*(3+4)	-
7	=	CALCULATED	14	['2', '*', '(', '3', '+', '4', ')']	2*(3+4)	14

4-4. 테스트 케이스 (예외 상황)

4-4. 테스트 케이스 (예외 상황)

TC_ID	시나리오	검증 포인트
TC_37	연산자로 끝나는 수식 후 등호 입력	수식 자동 보정 및 Error 처리 여부 확인
TC_37	닫는 괄호 부족	닫는 괄호 부족 시 자동 괄호 추가 확인
TC_39	괄호를 이중으로 닫음	여는 괄호 없이 닫는 괄호 무시 처리
TC_40	괄호 안에 연산자만 존재	불완전 수식 정리 및 자동 보완 확인
TC_42	소수점을 두 번 입력	중복 방지 및 메시지 출력 확인

예외 조건에 대한 보호 로직과 UX 대응에 초점을 둔 검증

예외 상황 검증을 통해 사용자 오류 입력을 방어하고 시스템의 안정성을 높임

4-4-1. 수식 끝에 연산자가 오는 경우 (TC_37)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	5	INPUTTING	5	-	-	5
2	+	INPUTTING	-	['5', '+']	5+	-
3	=	CALCULATED	5	['5']	5	5

4-4-2. 닫는 괄호 부족 (TC_38)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	1	INPUTTING	1	[]	-	1
2	+	INPUTTING	-	['1', '+']	1+	-
3	(	INPUTTING	-	['1', '+', '(']	1+(	-
4	1	INPUTTING	1	['1', '+', '(']	1+(	1
5	+	INPUTTING	-	['1', '+', '(', '1', '+']	1+(1+	-
6	=	INPUTTING	2	['1', '+', '(', '1', ')']	1+(1)	2

4-4-3. 괄호를 이중으로 닫음 (TC_39)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	(	INPUTTING	-	['(']	(	-
2	2	INPUTTING	2	['(']	(	2
3	+	INPUTTING	-	['(', '2', '+']	(2+	-
4	3	INPUTTING	3	['(', '2', '+']	(2+	3
5	)	INPUTTING	-	['(', '2', '+', '3', ')']	(2+3)	-
6	)	INPUTTING	-	['(', '2', '+', '3', ')']	(2+3)	-

4-4-4. 괄호 닫기 전 연산자로 끝남 (TC_40)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)
1	(	INPUTTING	-	['(']	(	-
2	4	INPUTTING	4	['(']	(	4
3	+	INPUTTING	-	['(', '4', '+']	(4+	-
4	)	INPUTTING	-	['(', '4', ')']	(4)	-

4-4-5. 소수점 중복 입력 처리 (TC_42)

	입력	상태	current_input	tokens	lineEdit_2(상단)	lindEdit(하단)	메시지
1	2	INPUTTING	2	-	-	2	-
2	.	INPUTTING	2.	-	-	2.	-
3	3	INPUTTING	2.3	-	-	2.3	-
4	.	INPUTTING	-	-	-	-	소수점은 한 번만 입력할 수 있습니다.

4-5. 테스트 결과 요약

4-5. 테스트 결과 요약

항목	테스트 수	통과	실패	통과율
기본 기능 테스트	9	9	0	100%
오류 후 입력 처리	9	9	0	100%
계산 후 입력 처리	9	9	0	100%
우선순위 및 괄호	8	8	0	100%
오류 방지/정정	6	6	0	100%
초기화	4	4	0	100%
총계	45	45	0	100%