유스케이스 명세 및 다이어그램

<Use Case 예시>

시스템 제목	SimpleCalc 계산기 프로그램
유스케이스 이름	계산 수행
액터	사용자
시작 조건	프로그램이 실행되어 입력 대기 상태여야 함
기본 흐름	1. 사용자가 사칙연산, 퍼센트(%), 나머지(%%) 등이 포함된 수식을 입력한다. 2. 시스템은 입력된 식을 구문 분석(Parser)하여 연산 종류를 판별한다. 3. 시스템은 계산을 수행하고 결과를 출력한다. 4. 결과는 소수점 2자리(필요 시 4자리 반올림)로 표시된다. 5. 계산 결과가 히스토리에 저장된다. 5-1. 히스토리에 저장된 계산 결과가 5개를 초과할 경우, 가장 오래된 계산기록을 삭제하고 최신 기록을 업데이트한다. 6. 다시 입력란으로 돌아온다.
대안 흐름	**3A.** 입력 구문 오류 혹은 연산 불가능 값 입력 시: 1. 시스템이 `"Error: 잘못된 입력입니다."` 메시지를 출력한다. 2. 프로그램은 종료되지 않고 다시 입력을 대기한다.
종료 조건	계산 결과가 출력, 해당 결과 히스토리에 저장 후, 다시 입력 대기 상태로 복귀

시스템 제목	SimpleCalc 계산기 프로그램
유스케이스 이름	history 조회
액터	사용자
시작 조건	프로그램이 실행되어 입력 대기 상태여야 함
기본 흐름	1. 사용자가 history 키워드를 입력한다. 2. 시스템은 history에 저장된 이전 입력식과 계산 결과를 출력한다. 3. 다시 입력란으로 돌아온다.
대안 흐름	2A history가 존재하지 않을 경우: 1. 시스템이 `"There is no previous calculation"` 메시지를 출력한다. 2. 프로그램은 종료되지 않고 다시 입력을 대기한다.
종료 조건	히스토리 출력 완료 후 입력 대기 상태로 복귀

시스템 제목	SimpleCalc 계산기 프로그램

<Class Diagram 사전작업>

1) 범위 먼저 못박기

- **액터**: 사용자(외부).
- **시스템 경계**: SimpleCalc 콘솔 앱.
- **유스케이스 매핑**: UC-1(계산 수행), UC-2(히스토리 조회).

2) 핵심 클래스 후보(최소 구성)

- **Calculator**: 퍼사드(진입점). `calculate(expr: str): str`
- **Parser**: 토큰화/구문분석. `%=percent`, `%%=mod`. `parse(expr: str): AST`
- **Evaluator**: AST 평가. `eval(ast: AST): Decimal`
- **Formatter**: 소수 2~4자리 반올림 규칙 적용. `format(x: Decimal): str`
- **HistoryManager**: 최근 5개 순환 저장. `add(item: HistoryItem)`, `list(limit=5): List<HistoryItem>`
- **CLI**(선택): 콘솔 I/O 담당. `read()`, `print()`
- **HistoryItem**(값 객체): `expression: str`, `result: str`, `timestamp: datetime`

> 너무 작게 가려면 CLI를 생략하고 Calculator가 I/O까지 맡아도 되지만, 입출력과 도메인 로직을 분리하면 테스트가 쉬워집니다.

3) 관계(연관/의존) 잡는 법

- `CLI → Calculator` (uses)
- `Calculator — Parser` (aggregation or dependency)
- `Calculator — Evaluator` (dependency)
- `Calculator — Formatter` (dependency)
- `Calculator — HistoryManager` (association 1..1)
- `HistoryManager — HistoryItem` (composition 1..* 최대 5)

> 의존(점선 화살표) 는 “메서드 인자로 쓰거나 잠깐 생성해 쓰는” 약한 결합.
>
> **연관(실선)** 은 “필드로 보유”하는 지속 관계.
>
> **합성(채운 다이아몬드)** 은 생명주기 함께함(HistoryItem은 Manager가 소유).

4) 속성/오퍼레이션 최소 표기

- **Calculator**
  - +calculate(expr: str): str
  - parser: Parser
  - evaluator: Evaluator
  - formatter: Formatter
  - history: HistoryManager
- **Parser**
  - +parse(expr: str): AST
- **Evaluator**
  - +eval(ast: AST): Decimal
- **Formatter**
  - +format(x: Decimal): str // 기본 2자리, 필요 시 4자리 반올림
- **HistoryManager**
  - items: List (size≤5, 최신이 앞)
  - +add(item: HistoryItem): void
  - +list(limit: int=5): List
- **HistoryItem**
  - +expression: str
  - +result: str
  - +timestamp: datetime

> 공개/비공개는 +/- 로, 인터페이스가 필요하면 <<interface>> 표기.

5) 제약/노트(꼭 다이어그램에 적기)

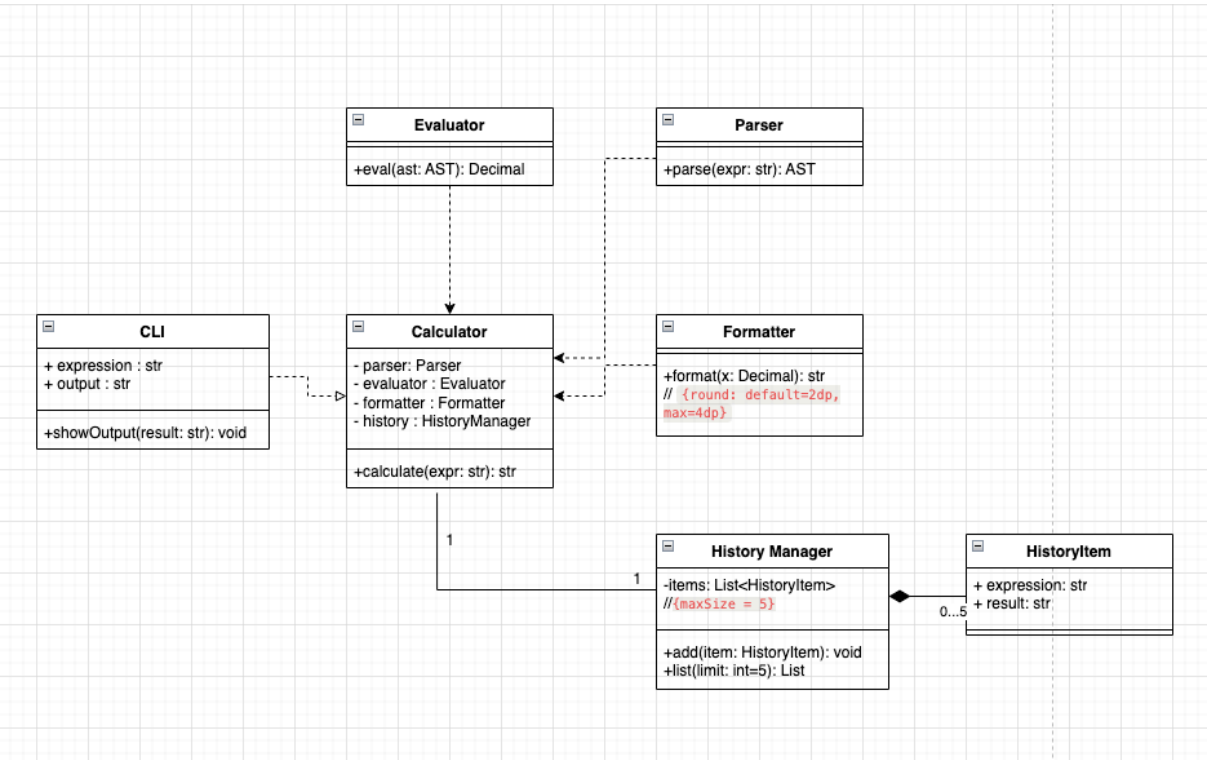
- `{maxSize = 5}` on HistoryManager.items
- `{percent %; modulus %%}` note on Parser
- `{round: default=2dp, max=4dp}` note on Formatter
- `{no eval(); allowed tokens: digits,+,-,*,/,(,),%,%%}` note on Parser/Evaluator

6) 그릴 때 순서 팁

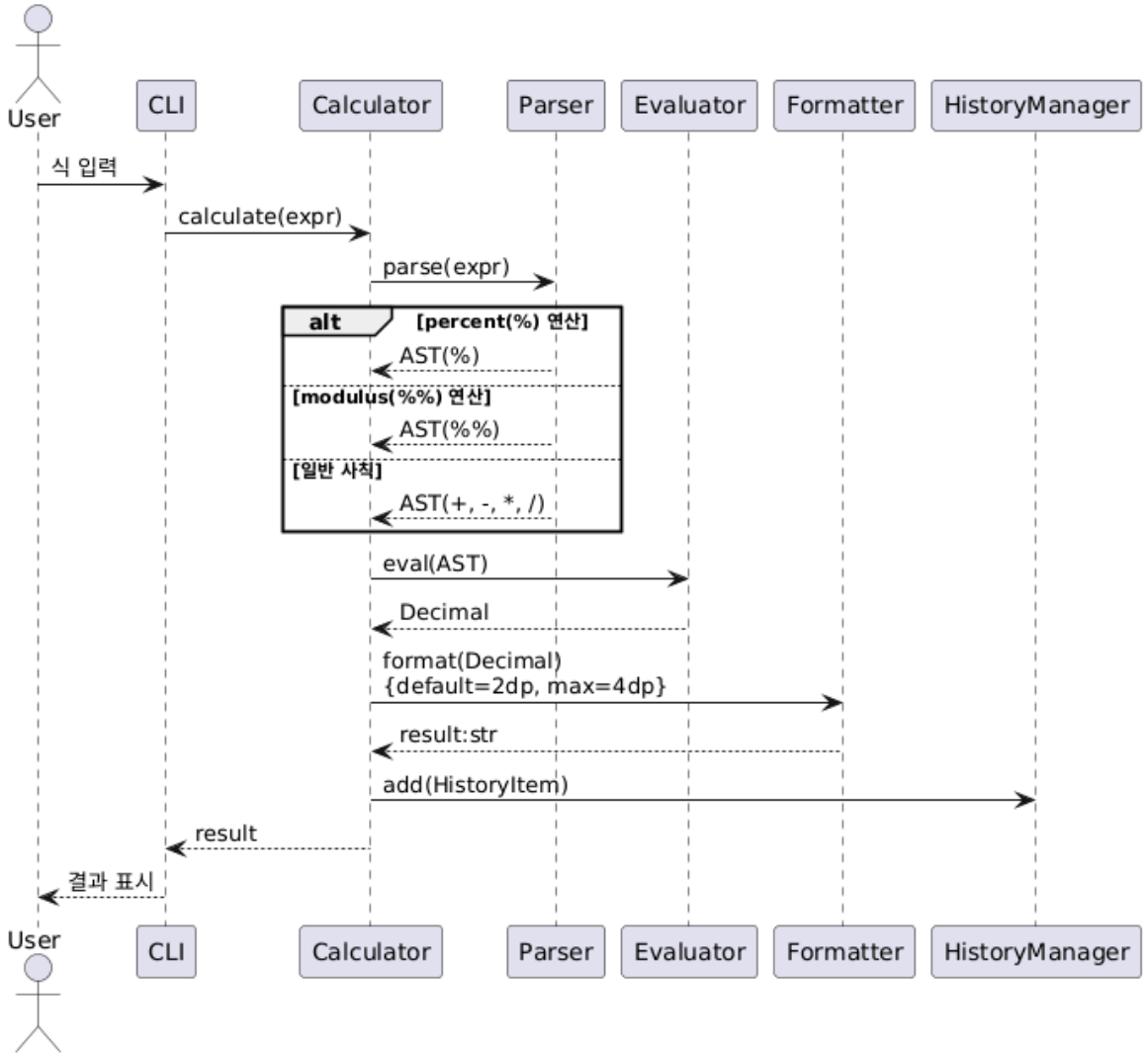
1. 상단 중앙에 **Calculator** (퍼사드).
2. 오른쪽에 **Parser / Evaluator / Formatter** 를 가볍게 배치(점선 의존).
3. 왼쪽에 **HistoryManager**(실선 연관) → 아래에 **HistoryItem**(합성 1..* / {maxSize=5}).
4. 맨 바깥에 **사용자(액터)** 를 두고 `CLI`를 사이에 두어 `사용자 —(association)→ CLI → Calculator`.

7) 검증 체크리스트

- 유스케이스의 **메시지(입력→계산→출력→저장)** 흐름이 클래스들 사이 연결로 표현됐는가?
- NFR(정밀도, 보안, 성능) 관련 제약이 **노트/제약**으로 박혀 있는가?
- 테스트 용이성을 위해 **로직과 I/O 분리**가 보이는가?


Evaluator
+eval(ast: AST): Decimal
Parser
+parse(expr: str): AST
CLI
+ expression : str
+ output : str
+showOutput(result: str): void
Calculator
- parser: Parser
- evaluator : Evaluator
- formatter : Formatter
- history : HistoryManager
+calculate(expr: str): str
Formatter
+format(x: Decimal): str
// {round: default=2dp, max=4dp}
1
1
History Manager
-items: List<HistoryItem>
//{maxSize = 5}
+add(item: HistoryItem): void
+list(limit: int=5): List
0...5
HistoryItem
+ expression: str
+ result: str

<시퀀스 다이어그램>



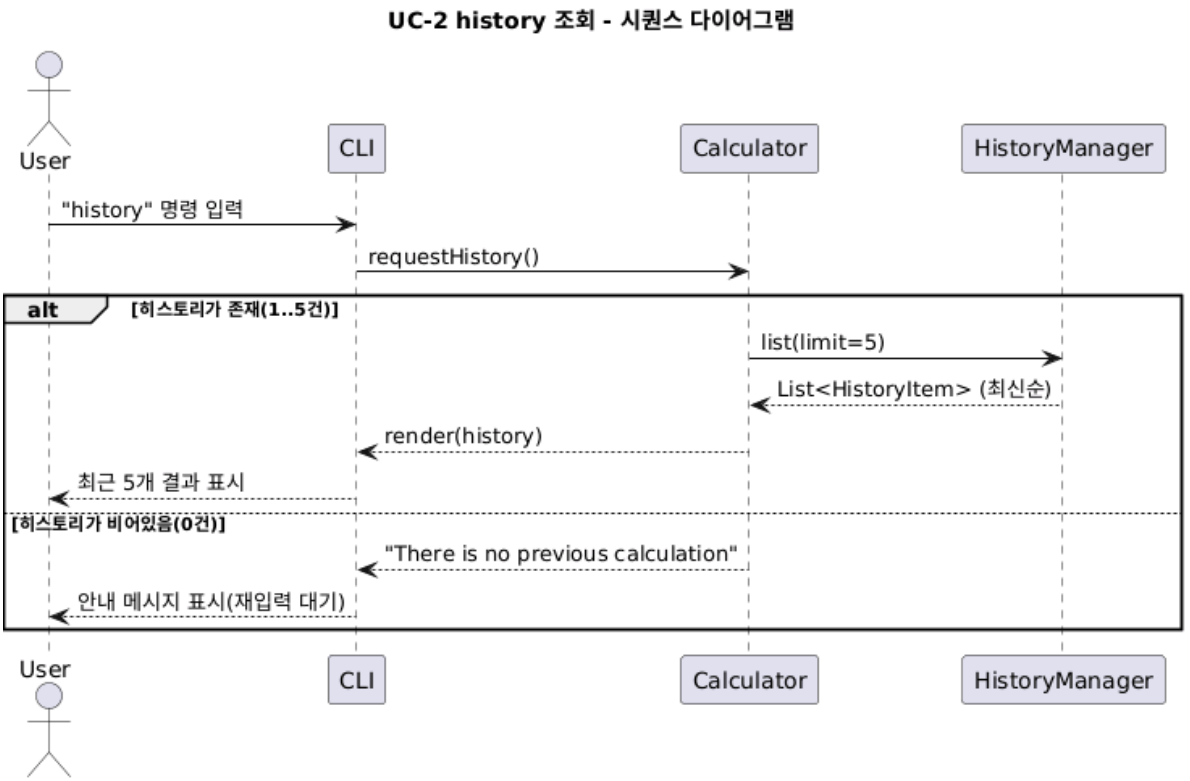
▼ UC1 : code

```
@startuml
actor User
participant CLI
participant Calculator
participant Parser
participant Evaluator
participant Formatter
participant HistoryManager

User → CLI: 식 입력
CLI → Calculator: calculate(expr)
Calculator → Parser: parse(expr)
```

alt percent(%) 연산
Parser → Calculator: AST(%)
else modulus(%%) 연산
Parser → Calculator: AST(%%)
else 일반 사칙
Parser → Calculator: AST(+, -, *, /)
end

Calculator → Evaluator: eval(AST)
Evaluator → Calculator: Decimal
Calculator → Formatter: format(Decimal)\n{default=2dp, max=4dp}
Formatter → Calculator: result:str
Calculator → HistoryManager: add(HistoryItem)
Calculator → CLI: result
CLI → User: 결과 표시
@enduml



▼ UC2 : code

```
@startuml
title UC-2 history 조회 - 시퀀스 다이어그램

actor User
participant CLI
participant Calculator
participant HistoryManager

User → CLI: "history" 명령 입력
CLI → Calculator: requestHistory()

alt 히스토리가 존재(1..5건)
Calculator → HistoryManager: list(limit=5)
HistoryManager --> Calculator: List<HistoryItem> (최신순)
Calculator --> CLI: render(history)
CLI --> User: 최근 5개 결과 표시

else 히스토리가 비어있음(0건)
Calculator --> CLI: "There is no previous calculation"
CLI --> User: 안내 메시지 표시(재입력 대기)
end

@enduml
```