문제 정의
이 문제는 덧셈(Add), 뺄셈(Sub), 곱셈(Mul), 나눗셈(Div)을 처리하는 4개의 클래스를 만들고 각 클래스에서 두 개의 정수와 연산자(+, -, *, /)를 입력받아 해당 연산을 처리하는 프로그램이다. 각 클래스는 int형 변수 a와 b를 가지고 setValue() 함수로 두 정수를 입력 받아 calculate() 함수로 연산 결과를 반환한다.

문제 해결 방법
클래스별로 역할을 분리하여 모듈화된 구조를 만든다.
예외 처리를 통해 나눗셈에서 0으로 나누는 경우 등 오류를 안전하게 처리한다.
무한 루프를 사용하여 사용자가 계속해서 입력하고 연산을 반복하도록 설계한다.
코드 중복을 줄이고 유지보수가 쉬운 구조로 최적화한다.

아이디어 평가
클래스별로 역할을 분리하여 모듈화된 구조를 만든 결과, 클래스 간의 독립성을 높여 확장 가능하고, 유지보수가 용이한 코드를 작성할 수 있었다.

예외 처리를 통해 예기치 못한 상황(특히 나눗셈에서 0으로 나누는 경우)을 처리한다. 나눗셈 연산에서 두 번째 입력이 0일 경우에는 예외를 발생시켜 프로그램이 비정상 종료되지 않도록 안전하게 설계한다. 예를 들어 나눗셈 연산에서 if (y == 0) 조건을 확인하고, 예외를 던지는 구조를 통해 에러 메시지를 출력하거나 try-catch 블록을 사용하여 에러 상황을 처리한다.

무한 루프를 사용하여 프로그램이 계속해서 사용자 입력을 받고 연산을 수행할 수 있게 한다. 연산자 이외의 문자 등 사용자가 잘못된 입력을 했을 때는 적절한 에러 메시지를 출력하고, 다시 입력받도록 유도한다.

동일한 연산 처리 방식이 반복되지 않도록 코드를 최적화합니다. 다형성을 사용하면 연산자에 따른 클래스 선택 이후의 setValue()와 calculate() 호출 코드가 중복되지 않고 간결해진다. 포인터로 객체를 가리키도록 하면, 조건문 이후 동일한 함수 호출 코드가 한 번만 작성되므로 코드 중복이 제거됩니다.

문제를 해결한 키 아이디어 또는 알고리즘