문제 정의

주어진 문제는 Shape라는 추상 클래스를 기반으로 Circle, Line, Rect 등의 도형 클래스를 상속받아 그래픽 편집기를 콘솔 바탕으로 구현하는 것입니다. 구현된 편집기는 다음과 같은 기능을 제공합니다:

1. 도형 삽입: 선(Line), 원(Circle), 사각형(Rect) 중 하나를 선택하여 추가.
2. 도형 삭제: 도형 리스트에서 특정 인덱스에 있는 도형을 삭제.
3. 도형 모두 보기: 현재 삽입된 모든 도형을 출력.
4. 프로그램 종료: 사용자가 종료를 선택하면 프로그램 종료.

문제 해결 방법

1. Shape 클래스를 추상 클래스로 선언하고, draw 메서드를 순수 가상 함수로 정의하여 자식 클래스들이 이를 구현하도록 설계. Circle, Line, Rect 클래스를 Shape 클래스를 상속받아 각자의 draw 메서드를 구현합니다.
2. 벡터를 활용한 도형 관리합니다. std::vector를 사용하여 생성된 도형 객체를 저장 및 관리합니다. 삽입(push_back)과 삭제(erase) 메서드를 통해 도형 리스트를 동적으로 조작.
3. 사용자 입력 및 출력 기능은 UI 클래스에서 처리합니다. 또한 그래픽 편집기의 도형 삽입, 삭제, 출력 등의 로직은 GraphicEditor 클래스에서 처리.

아이디어 평가

벡터를 활용하여 도형 객체를 동적으로 관리할 수 있어 삽입 및 삭제가 효율적입니다. 또한, erase를 사용해 특정 인덱스의 도형을 삭제 가능합니다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

Shape를 추상 클래스로 선언하고, 모든 도형 클래스는 이를 상속받아 draw 메서드를 구현. paint 메서드는 다형성을 활용하여 객체 타입에 따라 적절한 draw 메서드를 호출합니다.

벡터를 활용한 동적 도형 관리합니다. std::vector를 사용해 삽입(push_back) 및 삭제(erase) 연산으

로 동적으로 도형을 관리합니다.
각 코드에 주석을 통해 설명을 마무리하도록 하겠습니다.

```
#include <iostream>

#include <string>

#include "Shape.h"

#include "Circle.h"

#include "Rect.h"

#include "Line.h"

#include <vector>

using namespace std;

// UI 클래스: 사용자 입력과 출력 관련 기능을 담당

class UI {

    static int n; // 입력받은 숫자를 저장하는 변수

public:

    static void start(); // 프로그램 시작 메시지 출력

    static int menu(); // 메뉴 출력 및 사용자 입력 받기 (삽입, 삭제, 모두 보기, 종료)

    static int insert(); // 도형 삽입 선택 메뉴 출력 및 입력

    static int del(); // 삭제할 도형의 인덱스 입력 받기

};

int UI::n = 0; // 정적 멤버 변수 초기화

// 프로그램 시작 메시지 출력

void UI::start() {

    cout << "그래픽 에디터입니다." << endl;

}
```

```
// 메뉴 출력 및 선택값 입력받기
int UI::menu() {
	cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
	cin >> n; // 사용자가 입력한 메뉴 번호 저장
	return n; // 메뉴 번호 반환
}
// 도형 삽입 메뉴 출력 및 선택값 입력받기
int UI::insert() {
	cout << "선:1, 원:2, 사각형:3 >> ";
	cin >> n; // 삽입할 도형 번호 저장
	return n; // 도형 번호 반환
}
// 삭제할 도형 인덱스 입력받기
int UI::del() {
	cout << "삭제하고자 하는 도형의 인덱스 >> ";
	cin >> n; // 삭제할 도형의 인덱스 저장
	return n; // 인덱스 반환
}
// GraphicEditor 클래스: 도형 관리 및 주요 동작 수행
class GraphicEditor {
	vector<Shape*> v; // 도형 객체를 저장하는 벡터
	vector<Shape*>::iterator it; // 벡터 순회를 위한 반복자
public:
```

```
    GraphicEditor() {} // 기본 생성자
    void insert(); // 도형 삽입
    void deleteShape(); // 도형 삭제
    void showAll(); // 모든 도형 출력
    void start(); // 그래픽 에디터 실행
};
// 도형 삽입 함수
void GraphicEditor::insert() {
    int n = UI::insert(); // 사용자가 삽입할 도형 종류 입력
    if (n == 1) // 선(Line) 삽입
        v.push_back(new Line());
    else if (n == 2) // 원(Circle) 삽입
        v.push_back(new Circle());
    else if (n == 3) // 사각형(Rect) 삽입
        v.push_back(new Rect());
    else
        cout << "입력 에러" << endl; // 잘못된 입력 처리
}
// 도형 삭제 함수
void GraphicEditor::deleteShape() {
    int n = UI::del(); // 삭제할 도형의 인덱스 입력
    if (n >= 0 && n < v.size()) { // 입력된 인덱스가 유효한지 확인
        Shape* del = v[n]; // 삭제할 도형 객체 포인터 저장
```

```
            v.erase(v.begin() + n); // 벡터에서 해당 도형 삭제
            delete del; // 동적 메모리 해제
        } else {
            cout << "잘못된 인덱스입니다." << endl; // 잘못된 인덱스 처리
        }
}
// 모든 도형 출력 함수
void GraphicEditor::showAll() {
        for (int i = 0; i < v.size(); ++i) { // 벡터에 저장된 모든 도형 순회
            cout << i << ": "; // 도형 인덱스 출력
            v[i]->paint(); // 다형성을 활용하여 도형 출력
        }
}
// 그래픽 에디터 시작 함수
void GraphicEditor::start() {
        UI::start(); // 시작 메시지 출력
        for (;;) { // 무한 루프를 통해 사용자 입력 반복 처리
            int m = UI::menu(); // 메뉴 선택
            if (m == 1) // 삽입 선택
                insert();
            else if (m == 2) // 삭제 선택
                deleteShape();
            else if (m == 3) // 모두 보기 선택
```

```
            showAll();
        else if (m == 4) // 종료 선택
            break; // 루프 종료
        else
            cout << "입력 에러 " << endl; // 잘못된 메뉴 입력 처리
    }
}
int main() {
    GraphicEditor* g = new GraphicEditor; // GraphicEditor 객체 동적 생성
    g->start(); // 그래픽 에디터 실행
    delete g; // 동적 메모리 해제
}
```