객체지향 프로그래밍 6차 과제

문제정의

그래픽 편집기를 콘솔 기반으로 구현하는 문제입니다.

1. **삽입**: "Line", "Circle", "Rectangle" 중 하나의 도형을 삽입.
2. **삭제**: 삽입된 도형 중 하나를 선택하여 삭제.
3. **모두 보기**: 현재 삽입된 모든 도형을 출력.
4. **종료**: 프로그램을 종료.

주어진 조건에 따라 도형의 삽입, 삭제, 조회 기능을 구현하고, 프로그램의 메모리 관리를 통해 동적으로 생성된 객체를 정확히 관리해야 합니다.

소스 수행 결과 화면 캡쳐

```
그래픽 에디터입니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 1
도형이 추가되었습니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 2
도형이 추가되었습니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 3
도형이 추가되었습니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Circle
2: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >>
```

문제 해결 방법

- **객체지향 설계**:
  - `Shape` **추상 클래스**: 공통 인터페이스를 제공하며, 모든 도형의 기반 클래스.
  - `Line`, `Circle`, `Rect` **클래스**: `Shape` 클래스를 상속받아 각각의 도형을 구현.
- **도형 관리**:
  - 도형 객체는 동적으로 생성되며, `std::vector<Shape*>`를 사용해 저장하고 관리.
  - 삽입, 삭제, 모두 보기 기능은 `GraphicEditor` 클래스에서 처리.
- **유저 인터페이스**:
  - 입력을 통해 메뉴 선택(삽입/삭제/조회/종료)을 수행.
  - 삭제 시 유효하지 않은 인덱스 처리.
- **메모리 관리**:
  - 동적으로 생성된 도형 객체는 삭제 시 메모리를 해제.
  - 프로그램 종료 시 모든 객체를 정리.

아이디어 평가

1. **객체지향 설계**:
   - `Shape` 추상 클래스를 이용하여 도형을 추상화함으로써 새로운 도형이 추가될 경우 기존 코드를 수정하지 않아도 되는 유연한 구조를 제공.
   - 동적 할당을 통해 프로그램 실행 중 필요한 도형 객체를 생성 및 관리.
2. **유저 인터페이스**:
   - 입력 오류를 처리하여 사용자의 잘못된 선택에 대해 적절한 메시지를 제공.
3. **메모리 관리**:
   - 삽입된 객체를 삭제하거나 프로그램 종료 시 메모리를 해제하여 메모리 누수를 방지.

알고리즘 설명

- **도형 삽입**:
  - `GraphicEditor::insertShape(int shapeType)`:

        - 사용자로부터 입력받은 도형 타입(1: Line, 2: Circle, 3: Rect)에 따라 새로운 `Shape` 객체를 생성.
        - 생성된 객체를 벡터(`std::vector<Shape*>`)에 추가.
- **도형 삭제**:
    - `GraphicEditor::deleteShape(int index)`:
        - 사용자로부터 삭제할 도형의 인덱스를 입력받아 해당 객체를 삭제.
        - 유효하지 않은 인덱스 입력 시 적절한 오류 메시지 출력.
        - 삭제된 객체의 메모리를 해제하고 벡터에서 제거.
- **종료**
    - `main()` 함수에서 메뉴 반복 루프 종료.
    - 프로그램 종료 시 벡터에 저장된 모든 객체의 메모리를 해제하여 메모리 누수 방지.