문제 정의

문제는 사용자가 여러 도형(선, 원, 사각형)을 삽입하고, 삭제하며, 전체 도형을 출력할 수 있는 간단한 **그래픽 에디터**를 구현하는 것이다.

1. 도형들은 연결 리스트 구조를 이룬다.

2. 삽입, 삭제, 모두보기, 종료 기능이 있고, 사용자는 도형의 종류와 삭제할 인덱스를 선택할 수 있다.

문제 해결 방법

1. ***데이터 구조 선택***

   도형의 삽입 및 삭제를 효율적으로 처리하기 위해 연결 리스트 구조를 사용한다.
   ① 도형 간 관계를 연결 리스트 구조인 `Shape* next`로 연결한다.
   ② `pStart`와 `pLast` 포인터로 리스트의 시작과 끝을 만든다.

2. ***추상 클래스와 다형성 사용***

   ① 도형의 공통 기능(`draw()`)은 추상 클래스 `Shape`로 정의한다.
   ② 3개의 도형(Line, Circle, Rectangle)은 `Shape`를 상속받는다.

3. ***UI***

   ① static 클래스 `UI`로 사용자 인터페이스를 관리한다.
   ② 삽입, 삭제, 출력 기능이 있고, 잘못된 입력은 처리하지 않는다.

4. ***기능 구현***

   삽입

   ① 새로운 도형을 생성하고, 리스트의 끝에 연결한다.
   ② 리스트가 비어 있으면 리스트의 시작과 끝인 `pStart`와 `pLast`를 새 도형으로 설정한다.

   삭제

① 삭제 대상의 이전 도형과 삭제 대상을 추적하여 리스트 연결을 유지.
② 삭제 후 메모리 해제.

출력

① 리스트를 순회 및 각 도형의 정보를 출력.

아이디어 평가

데이터 구조 선택

- 구조를 단일 연결 리스트 구조로 선택함으로써 중간에 아무 도형을 삭제할 때 원소들을 옮기는 비효율적인 과정이 생략됨으로써 효율성이 증가한다.

추상 클래스와 다형성의 사용

- Shape를 상속받음으로써 가독성이 좋아지고, 개별 동작을 구현할 수 있다.

UI

- 정적 클래스는 상태를 유지하지 않는다. 그러므로 초기화 과정 없이 사용이 가능하며, 이는 메모리 효율성으로 연결되고 또한 코드가 간결해져 가독성도 좋아진다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

키 아이디어

단일 연결 리스트를 이용하여 도형을 동적으로 추가 및 삭제하였다.

다형성을 활용해 도형별 `draw()`를 오버라이딩하여 출력한다.

1. **삭제 알고리즘**

for (int i = 0; i < num; i++) { p = del; del = del->getNext(); }

여기에서 삭제 대상의 이전 도형(p) 삭제 대상(del)을 추적하고, 연결을 갱신한다. 삭제 대상이 첫 번째 도형이면 pStart = pStart->getNext();를, 중간 도형이면 p-

>setNext(del->getNext());를 사용한다.

2. 삽입 알고리즘

```
pLast = pLast->add(new ShapeType());
```

리스트가 비어있으면 pStart와 pLast를 새 도형으로 설정한다

3. 출력 알고리즘

```
for (int i = 0; i < count; i++) {
cout << i << ": "; p->paint();
p = p->getNext();
}
```

반복문으로 리스트 내용을 출력한다.