객체지향 프로그래밍 과제 보고서

-교제 P532 16번 문제-

과목명	객체지향프로그래밍
담당 교수	최인엽
학번	202104309 202104389
이름	이진용 황형진

문제 정의

현대적인 그래픽 소프트웨어는 다양한 도형(예: 선, 원, 사각형)을 삽입, 삭제, 관리할 수 있는 기능을 제공한다. 이 프로젝트의 목표는 vector<Shape*>를 사용하여 간단한 그래픽 편집기를 콘솔 환경에서 구현하는 것이다. 사용자는 도형을 삽입하고 삭제하며, 모든 도형을 확인할 수 있다. 이를 통해 객체지향 프로그래밍의 원리와 다형성을 활용한 문제 해결 방법을 실습하고자 한다.

문제 해결 방법

1. Shape 추상 클래스를 기반으로 상속받은 Line, Circle, Rectangle 클래스를 생성하여 도형별 동작을 캡슐화한다.
2. 각 도형 클래스는 고유의 draw() 메서드를 구현하여 화면에 출력한다.
3. 도형 객체는 vector<Shape*>에 저장된다. 이를 통해 삽입, 삭제 및 반복자를 활용한 순회가 용이하다.
4. 사용자 입력을 받아 도형 삽입, 삭제, 보기와 같은 기능을 수행하는 메뉴를 제공한다.
5. 동적으로 생성된 도형 객체는 삭제 시 메모리에서 해제된다.

아이디어 평가

1. 삽입 기능 : 사용자가 입력한 도형 번호에 따라 적절한 도형 객체를 생성하여 vector에 추가한다. 모든 테스트 케이스에서 원하는 도형이 정상적으로 삽입되었다.
2. 삭제 기능 : 입력받은 인덱스를 기반으로 벡터에서 도형을 삭제한다. 삭제 후 메모리 누수가 발생하지 않도록 동적 할당된 객체를 해제했다.
3. 모두 보기 기능 : 벡터에 저장된 모든 도형 객체를 순회하며 paint() 메서드를 호출하여 도형을 화면에 출력한다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

1. 추상 클래스와 다형성 : Shape 추상 클래스는 순수 가상 함수인 draw()를 통해 각 도형의 구체적인 동작을 하위 클래스에서 정의하도록 강제한다. 이를 통해 코드는 확장 가능하고 유지보수가 용이하게 설계됐다.
2. 벡터를 활용한 도형 관리 : 도형 객체는 벡터에 저장되며, push_back()으로 삽입하고 erase()로 삭제한다. 반복자를 사용하여 삭제 대상 도형의 위치를 찾는다.
3. UI와 GraphicEditor의 분리 : UI 클래스는 사용자 인터페이스 관련 로직만 처리하며, 실제 도형 관리 로직은 GraphicEditor 클래스에서 처리한다. 이는 단일 책임 원칙을 준수하며 코드의 가독성과 확장성을 높였다.
4. 다형성을 통한 도형 출력 : paint() 메서드는 도형 객체의 실제 타입에 따라 알맞은 darw() 메서드를 호출한다.

실행 결과

```
그래픽 에디터입니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 1
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 2
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 3
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Circle
2: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 2
삭제하고자 하는 도형의 인덱스 >> 1
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 4

D:\Study\cpp\HW#06\x64\Debug\HW#06.exe(프로세스 31956)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```