문제정의

위 문제는 그래픽 편집기를 구현합니다 기능은 도형을 생성하고 관리하는 프로그램입니다 문제에서 어떤 기능을 요구하냐면 삽입, 모두보기, 삭제, 종료 이렇게 4기능을 통해 도형 추가, 모든 도형 확인, 선택한 도형 삭제, 프로그램 종료를 할 수 있어야 합니다.
그리고 문제를 보면 Shape 클래스를 기반으로 Circle, Line, Rect 등의 도형 클래스를 상속받아 각각의 도형을 생성하는 구조로 설계해야 하며 GraphicEditor라는 클래스에서 이 도형 객체들을 관리하는 방식으로 프로그램을 작성해야 함을 알 수 있습니다.

문제 해결 방법

Shape 클래스와 상속 구조 활용:

Shape라는 추상 클래스를 만들고, 이를 상속받은 Line, Circle, Rect 클래스에서 도형을 구현해모든 도형이 같은 인터페이스를 같도록 했습니다.

링크드 리스트 기반 도형 관리:

GraphicEditor 클래스는 Shape 포인터를 사용해 연결 리스트 구조로 pStart는 첫 번째 도형을 가리키고, pLast는 마지막 도형을 가리킴으로써 삽입과 삭제 작업을 쉽게 수행할 수 있도록 했습니다.

UI 클래스를 통한 사용자 입력 관리:

UI 클래스에서 메뉴 입력, 도형 선택, 삭제할 인덱스 선택을 처리해, 사용자와의 상호작용을 구현했습니다. 이는 코드의 모듈화를 높여 GraphicEditor의 run 메서드에서 사용자가 입력한 명령을 처리할 수 있게 합니다.

다형성을 활용한 draw 함수:

Shape 클래스의 draw() 메서드는 순수 가상 함수로 정의되어 있고, 각 도형(Line, Circle, Rect)에서 이를 구현하여 자신의 타입을 출력하도록 했습니다. 이는 특정 도형의 종류를 확인하지 않고도 모든 도형을 일관되게 출력할 수 있습니다.

아이디어 평가

GraphicEditor 클래스에서 Shape* 포인터를 사용해 효율적인 삽입, 삭제가 가능하고 UI 클래스를 따라 구축해 가독성과 유지보수성이 향상 Shape라는 추상 클래스와 이를 상속하는 Line, Circle, Rect 클래스 구조는 각 도형이 공통의 인터페이스를 갖도록해 도형관리가 쉬워졌습니다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

-Shape 클래스를 추상 클래스 로 정의하고, 이 클래스가 각 도형 의 공통적인 인터페이스 역할을 하도록 설계해 Shape 클래스의 메서드만을 사용해 다양한 도형을 처리할 수 있게 해주며, 프로그램이 새로운 도형 클래스를 쉽게 추가할 수 있도록 유지보수나 확장이 용이합니다

-도형을 관리하는 방법으로 GraphicEditor 클래스에서 연결 리스트를 사용해 삽입과 삭제 작업을 pStart와 pLast 포인터를이용해 동적으로 작동하게 만들었습니다.

-UI 클래스는 정적 메서드를 통해 입력 기능을 제공하고 프로그램의 작동부분과 분리되어 사용자와의 상호작용 부분을 따로 관리할 수 있습니다

-Shape 클래스의 draw() 메서드를 순수 가상 함수로 선언하고, Line, Circle, Rect 클래스에서 각자 고유의 draw()를 구현해 show() 메서드에서 각 도형의 draw()를 호출해 도형 타입에 따라 다르게 동작하도록 하였으며, 이를 통해 각 도형이 자신의 타입을 알아서 출력합니다.