문제 정의
추상 클래스와 상속 종합을 응용하여 그래픽 에디터를 구현하는 문제이다.
도형(선, 원, 사각형) 삽입, 특정 도형 삭제, 현재 저장된 모든 도형을 출력
문제 해결의 핵심은 이러한 기능을 구조적으로 설계하고, 새로운 도형이 추가될 경우에도 확장 가능하도록 구현하는 것이다.

문제 해결 방법
추상 클래스 Shape : 모든 도형이 공통으로 가져야 할 인터페이스를 정의 (draw, paint 등)
도형 클래스 구현 (Line, Circle, Rectangle) : 각각의 도형은 Shape를 상속받아 고유의 draw 메서드를 구현
연결 리스트 사용 : 삽입, 삭제, 순회를 효율적으로 처리하기 위해 도형들을 단일 연결 리스트로 관리
사용자 인터페이스 제공 (UI 클래스) : 메뉴 기반 입력 시스템으로 사용자와 상호작용
모듈화 : 유지보수성과 가독성을 높이기 위해 파일을 각 클래스별로 분리.

아이디어 평가
도형 삽입 기능 : 여러 도형을 추가하며 리스트 구조가 정상적으로 작동하는지 확인했다. 결과적으로 새로운 도형이 리스트에 올바르게 추가되는 것을 확인했다. 또 없는 도형을 추가하려고 할 시 잘못된 도형이라고 안내메세지를 출력하도록 했다.
삭제 기능 : 특정 인덱스의 도형을 삭제했을 때 연결 리스트가 무결성을 유지하는지 확인했다. 첫 번째, 중간, 마지막 도형 삭제 테스트를 완료했고 도형이 없는 인덱스를 고를시 도형이 없다는 안내와 도형이 몇 번부터 몇 번까지 있는지를 확인해 출력하도록 했다.
출력 기능 : 현재 저장된 모든 도형을 출력하며 순서와 내용이 정확한지 확인했고 도형이 아무것도 저장되어있지 않을시 저장된 도형이 없다는 안내메시지를 출력하도록 했다.
그 외에도 목록의 지정되어 있지 않은 인덱스를 고를시 잘못된 인덱스라는 안내 메시지를 출력하도록 했다.

문제를 해결한 키 아이디어 또는 알고리즘 설명
Shape 추상 클래스를 통해 모든 도형이 공통된 인터페이스를 제공.
draw 메서드를 각 도형 클래스(Line, Circle, Rectangle)에서 구현하여 도형별 출력 방식을 따로 정의 하였고 이로 인해 새로운 도형이 추가될 때 기존 코드를 최소한으로 수정 가능.

Shape 클래스의 next 포인터를 사용하여 도형들을 단일 연결 리스트로 관리했다.
리스트 끝에 새로운 도형을 연결하였고 삭제할 도형은 이전 노드가 다음 노드를 가리키도록 해 도형의 동적 추가/삭제를 효율적으로 진행하였으며, 메모리를 낭비하지 않음.

모듈화
각 클래스(Shape, Line, Circle, Rectangle, GraphicEditor, UI)를 별도의 파일로 나누어 관리해 유지보수성이 높아지고, 특정 클래스의 변경이 다른 코드에 영향을 미치지 않도록 설계하였다.