객체지향프로그래밍
11주차 과제 소스구현설명

ICT융합학부 소프트웨어학과
202284002 고범주

1.문제 정의:

사용자가 선, 원, 사각형과 같은 도형을 삽입, 삭제, 출력할 수 있는 간단한 그래픽 에디터를 구현합니다. 도형들은 연결 리스트 구조로 관리되며, 삽입 순서대로 리스트에 저장됩니다. 사용자는 메뉴를 통해 도형을 추가하거나 삭제하고, 모든 도형을 화면에 출력할 수 있습니다.

2.문제 해결 방법:

(1) 추상 클래스 활용
Shape 클래스를 추상 클래스로 정의하고, 도형의 공통 동작인 draw() 메서드를 virtual로 선언하여 자식 클래스에서 구체적인 동작을 구현하도록 강제합니다.

(2) 도형별 클래스 구현
Line, Circle, Rectangle 클래스는 각각 Shape 클래스를 상속받아 고유한 draw() 메서드를 구현합니다. 이를 통해 도형별 출력 동작을 정의합니다.

(3) 연결 리스트 구조 구현
Shape 클래스에 next 포인터를 포함하여 도형들이 연결 리스트 형태로 연결됩니다.
GraphicEditor 클래스는 연결 리스트를 관리하며, 도형 삽입과 삭제, 출력 기능을 제공합니다.

(4) UI 클래스 설계
UI 클래스를 통해 사용자 입력을 처리하고, 삽입, 삭제, 보기와 같은 작업의 메뉴 선택을 단순화하여 사용자 편의를 높입니다.

3.아이디어 평가

(1) 추상 클래스와 다형성 활용
추상 클래스와 다형성을 통해 코드의 확장성과 재사용성이 높아졌습니다. 새로운 도형을 추가할 때 최소한의 변경만으로 시스템에 통합할 수 있습니다.

(2) 연결 리스트 구조 채택
동적으로 도형을 삽입하거나 삭제할 수 있는 연결 리스트 구조는 가변적인 데이터를 다루는 데 적합하며, 리스트를 순회하여 출력하는 과정도 간단히 구현할 수 있었습니다.

(3) 사용자 중심의 UI 설계
메뉴 선택과 입력 프로세스를 UI 클래스에서 처리하도록 분리하여 코드의 가독성과 유지보수성을 향상시켰습니다.

4.문제를 해결한 키 아이디어 또는 알고리즘 설명

(1) Shape 추상 클래스 설계
Shape 클래스는 연결 리스트의 노드 역할을 하며, 각 도형의 고유한 draw() 메서드를 호출할 수 있도록 가상 함수를 제공합니다.
도형의 삽입 및 삭제는 next 포인터를 통해 구현되며, 연결 리스트를 간단하게 관리할 수 있습니다.

(2) GraphicEditor 클래스
GraphicEditor 클래스는 도형 관리의 중심으로, 삽입 시에는 리스트의 끝에 새로운 도형을 추가하고, 삭제 시에는 리스트의 특정 노드를 제거합니다.
show() 메서드는 연결 리스트를 순회하며 모든 도형을 출력합니다.

(3) 동적 메모리 관리와 예외 처리
삽입 시에는 new를 사용하여 동적으로 도형을 생성하고, 삭제 시에는 delete를 사용하여 메모리를 해제합니다.
도형이 없는 상태에서 삭제를 시도할 경우, 경고 메시지를 출력하여 사용자의 실수를 방지합니다.