객체지향프로그래밍 - 과제 #5

- 202184021 소프트웨어전공 박지민

1. 문제 정의

1. 그래픽 에디터를 구현해야 한다.
2. 그래픽 에디터는 선(Line), 원(Circle), 사각형(Rectangle)을 삽입, 삭제, 조회할 수 있어야 한다.
3. 사용자는 다음의 기능을 선택할 수 있다:
   - 삽입 : 새로운 도형을 추가한다.
   - 삭제 : 특정 인덱스의 도형을 삭제한다.
   - 모두보기 : 현재 그래픽 에디터에 있는 모든 도형을 보여준다.
   - 종료 : 프로그램을 종료한다.
4. 도형들은 순서대로 삽입된다.
5. 선, 원, 사각형은 모두 도형이다.

2. 문제 해결 방법

2-1. 클래스 설계

- `Shape` 클래스
  - 다음 Shape 객체를 가리킬 수 있는 멤버 변수 `next`를 가진다.
  - 상속 받는 클래스들이 오버라이딩할 수 있는 `getType()` 함수를 가진다.
- `Line`, `Circle`, `Rectangle` 클래스
  - `Shape` 클래스를 상속받는다
  - `getType()` 함수를 오버라이딩하여 각 도형의 이름을 반환한다.
- `GraphicEditor` 클래스
  - `pStar` : 첫 번째 도형을 가리키는 포인터
  - `pLast` : 마지막 도형을 가리키는 포인터
  - `addShape(Shape* pShape)` : 도형을 추가한다.
  - `removeShape(int index)` : 특정 인덱스의 도형을 삭제한다.
  - `showAllShapes()` : 모든 도형을 출력한다.

이때 GraphicEditor는 도형을 **Linked-List** 형태로 관리한다.

2-2. 동작 과정

1. main 함수에서 `GraphicEditor` 객체를 생성하여 프로그램을 시작한다.
2. 사용자에게 다음 메뉴를 반복적으로 제공한다:
   - 삽입 : 추가할 도형의 종류를 선택하여 해당 도형 객체를 생성하고 addShape 함수를 통해 리스트에 추가한다.
   - 삭제 : 삭제하고자 하는 도형의 인덱스를 입력받아 removeShape 함수를 호출하여 해당 도형을 삭제한다.
   - 모두보기 : showAllShapes 함수를 호출하여 현재 리스트에 저장된 모든 도형의 인덱스와 타입을 출력한다.
   - 종료 : 프로그램을 종료한다.
3. 사용자의 선택에 따라 적절한 함수를 호출하여 기능을 수행
4. 예외 처리를 통해 잘못된 입력에 대한 대응을 한다.
5. 프로그램 종료 시 GraphicEditor 객체의 소멸자를 통해 모든 동적 할당된 도형 객체들을 delete를 사용하여 해제한다.

3. 아이디어 평가

- 객체지향 프로그래밍의 효율적 활용
  - 상속과 다형성을 통해 코드의 재사용성과 유연성을 높였다.
  - Shape 클래스를 추상화하여 공통된 인터페이스를 제공한다.
- **Linked-List** 자료구조를 활용한 도형 관리
  - 도형들의 추가와 삭제가 빈번히 일어나기 때문에,
  - 배열보다 연결 리스트를 사용하여 효율성을 높였다.
- 메모리 관리의 중요성 인식
  - 동적 할당된 도형 객체들을 소멸자에서 적절히 해제하여 메모리 누수를 방지했다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

- 상속 및 다형성 활용
  - GraphicEditor 클래스에서는 **Shape** 클래스의 포인터를 사용하여 다형성을 활용했다.
  - getType() 함수를 **virtual** 함수로 선언하여, 상속 받는 클래스에서 각 클래스에 맞는 이름을 반환하도록 했다.
- **Linked-List** 자료구조 활용
  - 도형들이 추가될 때마다 리스트의 끝에 새로운 노드를 추가하여 O(1)의 시간 복잡도로 삽입을 수행하였다.
  - 리스트의 시작(pStart)과 끝(pLast)을 가리키는 포인터를 사용하여 삽입과 삭제를 효율적으로 수행했다.
- 동적 메모리 관리
  - 도형 객체들은 new를 통해 동적 할당되었기 때문에, GraphicEditor의 소멸자에서 delete를 사용하여 메모리를 해제했다.