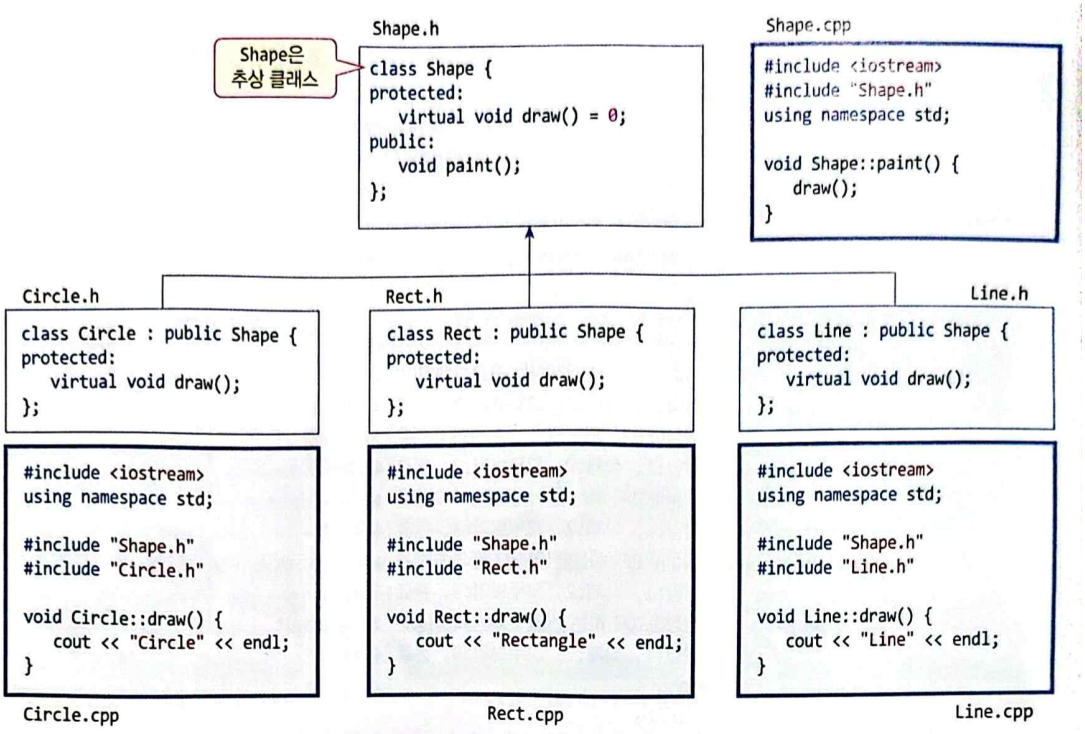
소스 파일 구현 설명

202304148 조하린

vector<Shape*>v; 이용하여 간단한 그래픽 편집기를 콘솔 바탕으로 만들어보자.
그래픽 편집기의 기능은 “삽입”, “삭제”, “모두보기”, “종료”의 4가지이다.
생성된 도형 객체를 v에 삽입하고 관리하라. (9번 실습 문제 10번 힌트 참고)



문제 정의

간단한 그래픽 에디터가 있다. 삽입, 삭제, 모두보기, 종료 4가지의 기능을 제공한다. 선택할 수 있는 도형에는 선(Line), 원(Circle), 사각형(Rect)이 있고 이 세 도형은 부모 클래스인 Shape에서 멤버들을 상속받는다. 키보드로부터 사용자가 사용할 기능에 해당하는 숫자를 입력받는다. 삽입(1) 선택 시 도형을 선택할 수 있고 그 도형은 리스트에 추가된다. 삭제(2) 선택 시 도형의 인덱스를 선택할 수 있고 그 도형은 삭제된다. 모두보기(3) 선택 시 인덱스 순서대로 ‘인덱스:도형 명’이 출력된다. 종료(4)를 선택하기 전까지 계속 기능 선택을 반복하고 종료(4) 선택 시 프로그램이 종료된다.

- 도형 리스트 - vector<Shape*> 활용하여 동적 배열
- 도형 추가/삭제 시 리스트를 업데이트 - 저장된 순서 유지

파일에 대한 설명

GraphicEditorVector 프로젝트

헤더파일
(1) graphiceditorvector.h - GraphicEditorVector 클래스 선언, 함수 선언
(2) ui.h - UI 클래스 선언, 함수 선언(static)
(3) shape.h - 도형들의 부모 Shape 클래스 선언, 함수 선언
(4) line.h - Line 클래스 선언, 가상 함수 선언
(5) circle.h - Circle 클래스 선언, 가상 함수 선언
(6) rect.h - Rect 클래스 선언, 가상 함수 선언

소스파일
(7) GraphicEditorVector.cpp - GraphicEditorVector 클래스의 멤버 함수 구현
(8) UI.cpp - UI 클래스의 멤버 함수 구현
(9) Shape.cpp - Shape 클래스의 멤버 함수 구현
(10) Line.cpp - Line 클래스의 멤버 함수 구현
(11) Circle.cpp - Circle 클래스의 멤버 함수 구현
(12) Rect.cpp - Rect 클래스의 멤버 함수 구현

(13) Main.cpp
    프로그램 시작 및 실행 - 객체 생성
    객체 생성: GraphicEditorVector의 객체 g - 동적 할당
    함수 호출 및 결과 출력

문제 해결 방법 (아이디어)

1. 클래스 - 변수, 함수

(1) GraphicEditorVector 클래스

- 멤버 변수
  - vector<Shape*> v: 도형 객체 포인터 저장하는 동적 배열
- 멤버 함수
  - GraphicEditorVector 생성자: 멤버 변수 초기화
  - 소멸자: 동적 생성된 도형의 객체 메모리 해제
  - void insert(int shapeNum): 선택된 도형 리스트에 추가
  - void remove(int shapeIndex): 선택된 도형 리스트에서 삭제
  - void show(): 리스트의 도형 순서대로 출력
  - void call(): 선택된 기능에 따른 입출력 함수 호출

(2) UI 클래스

- 멤버 함수
  - static int selectFn(): 입력받은 기능의 정수 fnNum변수에 저장
  - static int selectShape(): 입력받은 도형의 정수 shapeNum변수에 저장
  - static int deleteShape(): 삭제할 도형의 인덱스 shapeIndex변수에 저장

(3) Shape 클래스 - 도형들의 부모

- 멤버 함수
  - 생성자: 멤버 변수 초기화
  - 소멸자: 클래스의 소멸자 호출
  - virtual void draw(): 각 도형의 고유 동작을 정의하는 순수 가상 함수
  - void paint(): draw() 호출하는 함수

(4) Line 클래스

- 멤버 함수 virtual void draw() : 선 draw 동작 정의하는 순수 가상 함수

(5) Circle 클래스

- 멤버 함수 virtual void draw() : 원 draw 동작 정의하는 순수 가상 함수

(6) Rect 클래스

- 멤버 함수 virtual void draw() : 사각형 draw 동작 정의하는 순수 가상 함수

2. 벡터(Vector)

- 도형 간의 연결 vector<Shape*>
- 동적 크기 관리 - 벡터의 크기 자동 조정
- push_back, erase 메서드 통해 코드 간결화
- 벡터의 인덱스 통해 도형 순서대로 출력

3. 추상 클래스
   - Shape 클래스는 도형들의 공통 기능을 정의한 추상 클래스이다.
   - 공통 기능 virtual void draw() 순수 가상 함수
   - 도형 간의 연결 vector<Shape*> 사용하여 리스트로 관리

4. 상속
   - 기본 클래스 Shape로부터 상속받은 Line, Circle, Rect 클래스
   - 상속을 통해 Shape 클래스의 멤버들을 파생 클래스에서 재사용하게 하는데, 파생 클래스가 기본 클래스를 public으로 상속받아 기본 클래스의 public, protected 멤버들을 모두 그대로 물려받아 사용한다.

5. 키보드 입출력
   - UI 클래스 - selectFn() 기능 선택, selectShape() 도형 선택,
     deleteShape() 삭제할 도형의 인덱스 선택
   - 입력 스트림 객체 cin을 통해 선택할 기능, 도형, 도형 인덱스에 해당하는 각각의 정수를 입력받아 fnNum, shapeNum, ShapeIndex 변수에 저장한다.

6. 반복문과 조건문
   1) while 반복문을 통해 전체 프로그램을 계속 실행한다.
      사용자가 종료를 선택할 때까지 계속 실행한다.
   2) switch문을 통해 case를 나눈다.
      사용자가 선택한 기능에 따른 작업을 실행한다.
   3) if-else 조건문을 통한 검사
      도형 empty나 잘못된 인덱스 검사

7. break 문 - 반복문 진행 중에 현재 반복문을 종료

8. 객체 동적 할당
   - new 연산자를 사용하여 GraphicEditorVector의 객체를 동적으로 생성

9. 소멸자
   - new 연산자를 사용하여 GraphicEditorVector의 객체를 동적으로 생성하였으므로 소멸자를 이용해 메모리에서 해제되도록 한다.

문제 해결 키 (아이디어)

벡터<Vector>

- 도형 간의 연결 vector<Shape*>
- 동적 크기 관리 - 벡터의 크기 자동 조정
- push_back, erase 메서드 통해 코드 간결화
- 벡터의 인덱스 통해 도형 순서대로 출력

프로그램 순서

1. 프로그램 시작
2. 에디터 기능 4가지 (삽입:1, 삭제:2, 모두보기:3, 종료:4) 제시
3. 4가지 기능에 해당하는 숫자 선택
   1) 삽입
      도형 3가지 (선:1, 원:2, 사각형:3) 제시
      선택 -> 도형 추가
   2) 삭제
      삭제할 도형의 인덱스 선택 -> 도형 삭제
   3) 모두보기
      '인덱스:도형 명' 순서대로 출력
   4) 종료
4. while 반복 - 종료(4) 선택하지 않으면 2단계로 돌아감
5. 프로그램 종료 - 종료(4) 선택 시

아이디어 평가

1. 벡터(Vector)
   - 벡터의 크기 자동으로 조정할 수 있어 메모리 관리에 편리하다
   - push_back, erase 메서드 통해 코드를 간단하게 구현 할 수 있다.

2. 추상 클래스
   - 파생 클래스를 정의할 때 최소한의 변경으로 프로그램을 확장할 수 있다.
   - 유연성과 확장성에 좋다.

3. 상속
   - 기본 클래스 Shape - 공통 기능 virtual void draw()
   - 파생 클래스 Line
   - 파생 클래스 Circle
   - 파생 클래스 Rect
   - 상속을 통해 기본 클래스에는 없는 파생 클래스가 가지는 정보만을 추가하므로 코드 중복을 줄일 수 있다.

4. 접근 지정자
   - 클래스 간의 상속을 통해 기본 클래스의 기능을 파생 클래스에서 재사용한다.
   - 기본 클래스의 멤버들이 상속을 통해 파생 클래스의 멤버로 확장될 때, 기본 클래스 멤버의 접근 지정은 상속 조건에 따라 달라진다.
   - public 상속
     기본 클래스를 public으로 상속받으면, 기본 클래스의 protected, public 멤버들을 모두 그대로 물려받아 파생 클래스에 상속 확장된다.

5. 객체 동적 생성
   - new 연산자를 사용하여 객체를 동적 생성 함으로써 메모리를 할당할 수 있다.

6. 소멸자와 delete
   - 동적으로 할당된 객체는 프로그램 종료 시 반드시 메모리를 해제가 필요하기 때문에 소멸자를 명시적으로 정의해야 한다.

7. 파일 분리
   파일 분리를 통해 각 파일의 역할에 맞게 나뉘어 코드의 가독성을 향상할 수 있었다. 또한 코드를 재사용할 수 있고, 각 클래스를 독립적으로 관리할 수 있게 되었으며 디버깅에 용이해졌다.