

소스 구현 설명

202304178 구서연

문제 정의 :

상속관계에 대해서 정확하게 이해하고 vector를 이용하여 객체를 동적으로 관리할 수 있다.

문제 해결 방법 :

- 상속관계 정리

문제에서 제시한 상속관계를 정의하고, 클래스 선언부와 구현부를 나누었다.

주어진 조건에서 Shape는 순수 가상 함수인 draw함수를 가진 추상 클래스로, Circle, Line, Rect 클래스가 이를 상속받는 형태가 된다.

```
#ifndef SHAPE_H
#define SHAPE_H

class Shape {
    Shape* next;
protected:
    virtual void draw()=0;
public:
    void paint();
};

#endif
```

이들 각각을 선언부와 구현부로 Shape.h, Shape.cpp와 같이 분리하고, 중복 선언으로 오류가 나는 것을 피해주고자 아래와 같은 방법을 사용했다.

```
#ifndef SHAPE_H
#define SHAPE_H

#endif
```

또한, 각 자식 클래스(Circle, Line, Rect)에서는 Shape를 상속받아 Shape의 가상 함수 draw()와 동일한 이름의 함수를 선언하고 오버라이딩으로 각각 다른 역할을 수행한다.

```
#ifndef CIRCLE_H
#define CIRCLE_H
#include "Shape.h"
class Circle : public Shape {
public:
    virtual void draw();
};

#endif
```

```
#include <iostream>
using namespace std;
#include "Circle.h"
void Circle::draw() {
    cout << "Circle" << endl;
}
```

- 동적 메모리 관리

주어진 조건에 맞춰 `vector<Shape*> shapes`를 사용하여 도형을 동적으로 관리했다. `vector`의 경우 동적 크기를 지원하기 때문에 삽입과 삭제가 빈번하게 발생하는 이번 문제의 경우에 사용하기 적합하다. `shapes` 벡터는 `Shape*` 타입의 포인터를 저장하고 이를 이용하여 각 도형에 알맞은 `paint` 메서드를 실행한다.

```
void GraphicEditor::input_new(int n) {
    Shape* newShape = nullptr;
    switch (n) {
        case 1:
            newShape = new Line();
            break;
        case 2:
            newShape = new Circle();
            break;
        case 3:
            newShape = new Rect();
            break;
        default:
            cout << "잘못된 입력입니다.\n";
            return;
    }
    shapes.push_back(newShape);
}
```

또한, `input_new(int n)`함수에서 명령에 알맞은 도형을 동적으로 생성하고 이를 벡터에 집어넣어준다.

```
bool GraphicEditor::del(int n) {
    if (n < 0 || n >= shapes.size()) {
        cout << "잘못된 인덱스입니다.\n";
        return false;
    }
    delete shapes[n];
    shapes.erase(shapes.begin() + n);
    return true;
}
```

도형을 삭제하고 싶은 경우는 `del(int n)`에서 벡터에서 삭제할 도형의 메모리를 해제를 해준 후, 벡터에서 해당 인덱스 제거하는 방식으로 진행을 하였다.

```
GraphicEditor::~GraphicEditor() {
    for (Shape* shape : shapes) {
        delete shape;
    }
    shapes.clear();
}
```

그리고 만약 프로그램이 종료될 때 도형이 남아있는 경우, 그 도형들도 프로그램 종료 전 메모리를 해제해주기 위해서 소멸자를 이용하여 `~GraphicEditor()`에서 남은 도형들을 삭제해줄도록 한다.

아이디어 평가 :

- 상속과 virtual의 활용

문제 조건에 맞춰 shape을 각 도형 클래스가 상속하도록 하였다. virtual을 이용하여 파생 클래스에서 함수 오버라이드한 Line, Circle, Rect 클래스의 draw() 메서드를 paint()호출 시 알맞은 도형의 draw() 메서드가 호출하도록 동적 바인딩을 해주었다.

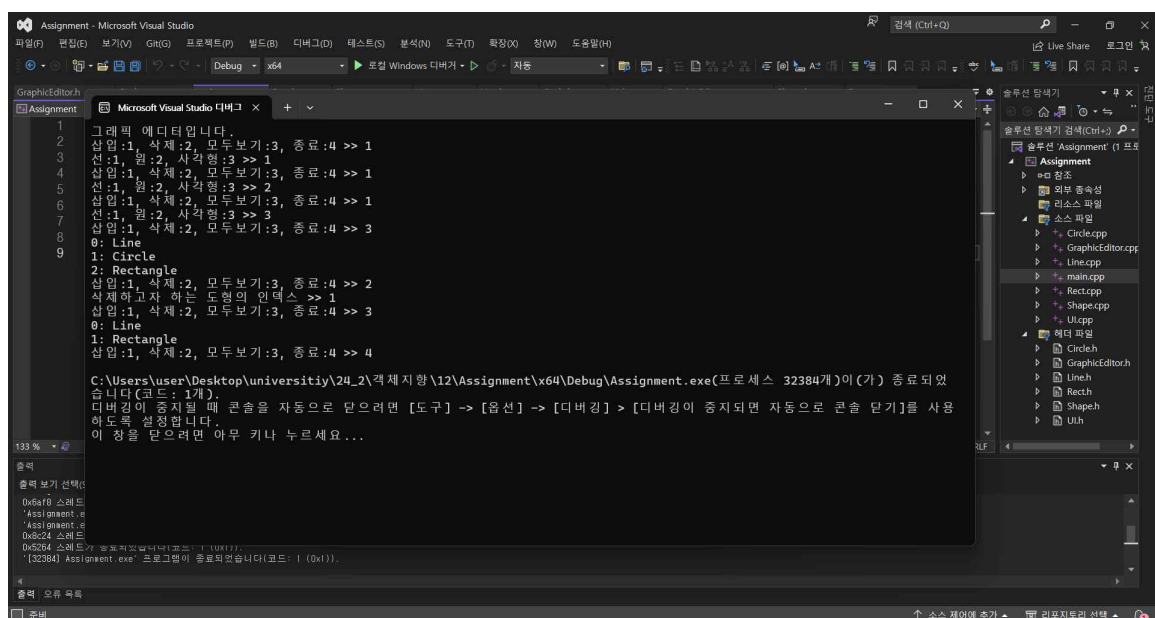
또한, 코드의 가독성을 높이고 코드의 유지보수성을 높이기 위해 각 클래스의 선언부와 구현부를 분리하였다. 그 과정에서 #ifndef, #define, #endif를 사용하여 중복 선언으로 인한 에러를 방지하였다.

- 동적 메모리 관리

vector를 사용하여 도형의 삽입과 삭제를 동적으로 진행하였다. 부모 클래스인 Shape의 포인터를 이용하여 각 도형에 알맞은 paint 메서드를 실행하기 때문에, 새로운 도형이 추가하거나 하는 코드 유지보수성을 높일 수 있다.

도형을 삭제하는 경우에는 del(int n)에서 delete를 사용하고, 프로그램 종료 시 메모리가 누수되지 않도록 소멸자를 통해 프로그램 종료 전 모든 객체를 해제해주고 종료하였다.

이러한 방법들로 문제를 풀면 다음과 같이 올바르게 실행이 되는 것을 볼 수 있다.



```
Assignment - Microsoft Visual Studio
파일(F) 편집(E) 보기(V) Git(G) 프로젝트(P) 빌드(B) 디버그(D) 테스트(S) 분석(A) 도구(T) 확장(X) 창(W) 도움말(H)
Debug x64
실행 Windows 디버거
저장
Live Share 로그인
GraphicEditor.h
Assignment
1 그래픽 에디터입니다.
2 삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
3 선:1, 원:2, 사각형:3 >> 1
4 삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
5 선:1, 원:2, 사각형:3 >> 2
6 삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
7 선:1, 원:2, 사각형:3 >> 3
8 삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
9 0: Line
1: Circle
2: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 2
삭제하고자 하는 도형의 인덱스 >> 1
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 4
C:\Users\user\Desktop\university\24_2\객체지향\12\Assignment\x64\Debug\Assignment.exe(프로세스 32384개)이(가) 종료되었
습니다(코드: 17).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
133 %
출력
출력 보기 선택:
Dx5a16 스택트
Assignment
Assignment
Dx5c24 스택트
Dx5264 스택트
[32384] Assignment.exe 프로그램이 종료되었습니다(코드: 1 (0x1)).
출력 오류 목록
준비
소스 제어에 추가 리포트지표리 선택
```