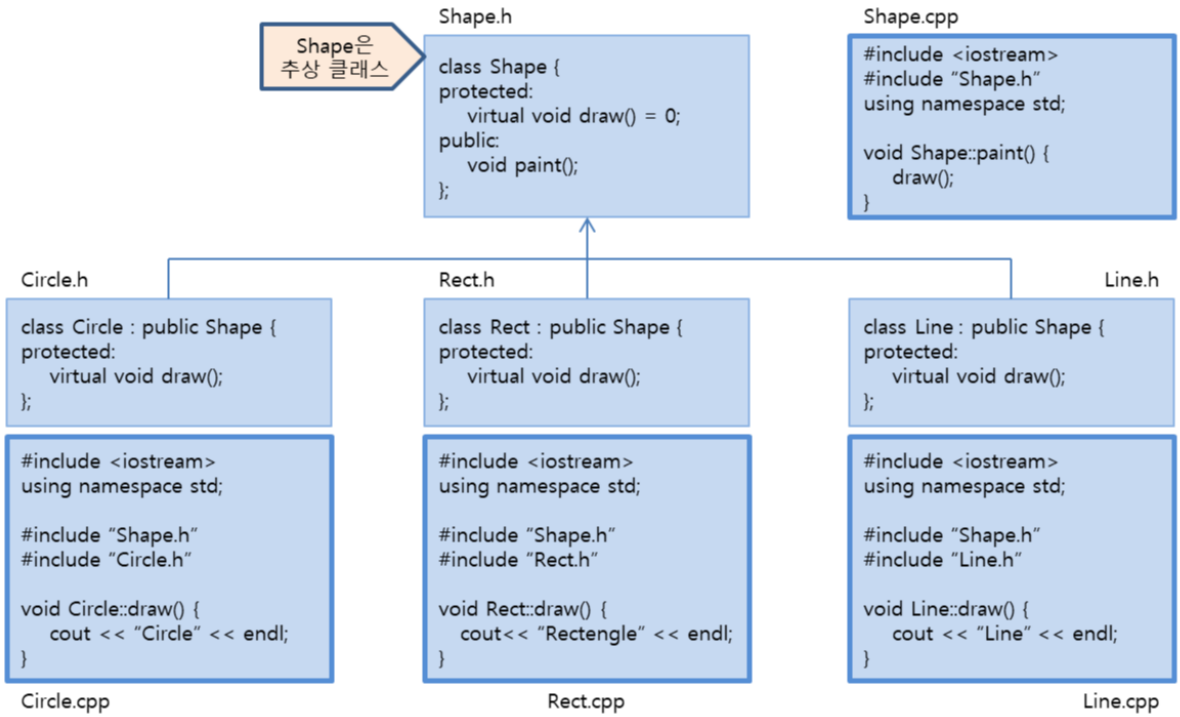
객체지향 프로그래밍 과제6

202284011 김연재

202004006 고창석

[문제 정의]



vector<Shape*> v;를 이용하여 간단한 그래픽 편집기를 콘솔 바탕으로 만들어보자.

[문제 해결 방법]

shape 클래스: 도형들의 공통된 인터페이스를 제공하는 추상 클래스이며 paint(), draw() 메서드를 가지고 있다

circle, rect, line 클래스: shape 클래스를 상속받으며 오버라이딩한 draw() 메서드로 각각 원,사각형, 선의 이름을 출력한다.

```
class UI {
    static int n;
public:
    static void start();
    static int menu();
    static int insert();
    static int del();
};

int UI::n = 0;
void UI::start() {
    cout << "그래픽 에디터입니다." << endl;
}

int UI::menu() {
    cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
    cin >> n;
    return n;
}

int UI::insert() {
    cout << "선:1, 원:2, 사각형:3 >> ";
    cin >> n;
    return n;
}

int UI::del() {
    cout << "삭제하고자 하는 도형의 인덱스 >> ";
    cin >> n;
    return n;
}
```

UI 클래스: 정적 멤버 함수와 변수를 사용하여 사용자 인터페이스를 담당.

start(): 프로그램 시작 메시지 출력.

menu(): 메뉴를 출력하고 사용자의 선택을 입력받는다.

insert(): 도형 삽입 시 사용자로부터 도형 종류를 입력받는다.

del(): 도형 삭제 시 삭제할 도형의 인덱스를 입력받는다.

```
class GraphicEditor {
    vector<Shape*> v;
    vector<Shape*>::iterator it;
public:
    GraphicEditor() {}
    void insert() {
        int n = UI::insert();
        if (n == 1)
            v.push_back(new Line());
        else if (n == 2)
            v.push_back(new Circle());
        else if (n == 3)
            v.push_back(new Rectangle());
        else cout << "입력 에러" << endl;
    }
    void deleteShape() {
        int n = UI::del();
        Shape* del;
        it = v.begin();
        for (int i = 0; i < n; ++i)
            ++it;
        del = *it;
        it = v.erase(it);
        delete del;
    }
    void showAll() {
        for (int i = 0; i < v.size(); ++i) {
            cout << i << ": ";
            v[i]->paint();
        }
    }
    void start() {
        UI::start();
        for (;;) {
            int m = UI::menu();
            if (m == 1)
                insert();
            else if (m == 2)
                deleteShape();
            else if (m == 3)
                showAll();
            else if (m == 4) break;
            else cout << "입력 에러 " << endl;
        }
    }
};
```

GraphicEditor 클래스: 그래픽 에디터의 핵심 로직을 담당.

v: Shape* 포인터를 저장하는 벡터. 삽입된 도형을 저장

it: 벡터의 이터레이터, 삭제 시 사용

insert(): 사용자의 입력에 따라 Line, Circle, Rect 객체를 동적으로 생성하고 벡터에 추가

deleteShape(): 벡터에서 사용자가 지정한 인덱스의 도형을 삭제. 메모리 누수를 방지하기 위해 삭제된 객체를 동적으로 해제

showAll(): 벡터에 저장된 모든 도형의 paint() 메서드를 호출하여 도형 정보를 출력

start(): UI 클래스를 사용하여 프로그램의 흐름(삽입, 삭제, 보기, 종료)을 제어.

```
int main() {
    GraphicEditor* g=new GraphicEditor;
    g->start();
    delete g;
}
```

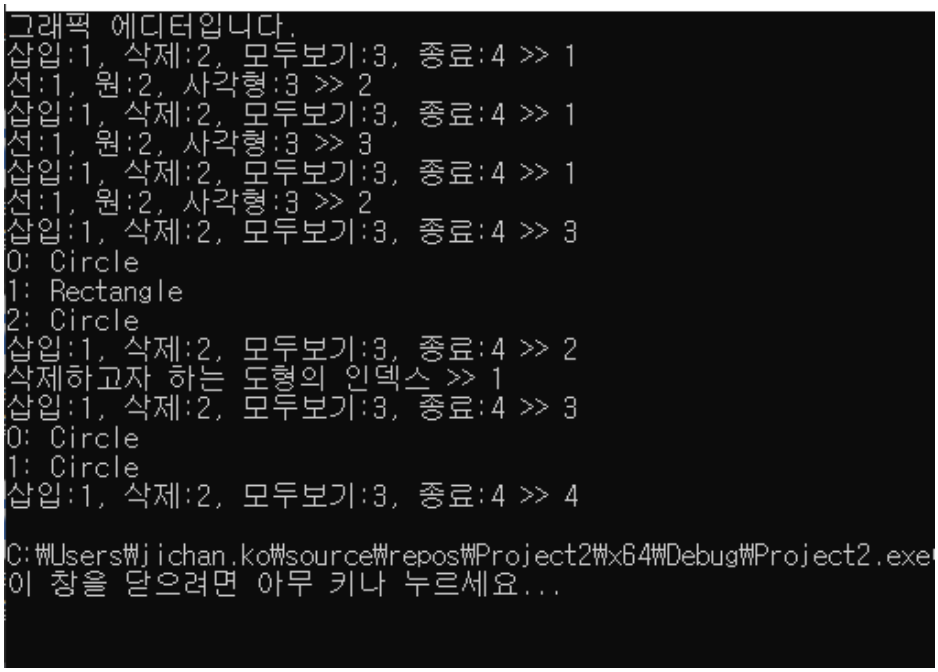
main 함수:

GraphicEditor 객체를 동적으로 생성

GraphicEditor::start()를 호출하여 프로그램을 시작

프로그램 종료 시 GraphicEditor 객체를 삭제하여 메모리를 해제

[소스 수행 결과 화면 캡처]



[평가]

김연재: 배열을 사용할때보다 벡터를 사용하는 것이 훨씬 효율적으로 데이터를 관리할 수 있었고 벡터의 크기를 따로 신경쓰지 않아도 된다는 점이 편리하였다.

고창석: 이터레이터를 사용하면 컨테이너 내부 요소의 특정 위치에 접근하거나 삽입, 삭제할때 코드를 간결하게 작성할 수 있었다.