<객체지향프로그래밍_12주차과제_소스구현설명>

7조(202104140김경록,2021042760이선우>

```
#include <iostream>
#include <vector>
using namespace std;

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

class Circle : public Shape {
protected:
    virtual void draw(){ cout << "Circle" << endl; }
};

class Rect : public Shape {
protected:
    virtual void draw() { cout << "Rectangle" << endl; }
};

class Line : public Shape {
protected:
    virtual void draw() { cout << "Line" << endl; }
};

class UI {
public:
    static int seleteMenu() {
        int n;
        cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
        cin >> n;
        return n;
    }
   static int seleteShape() {
        int n;
        cout << "선:1, 원:2, 사각형:3 >> ";
        cin >> n;
        return n;
    }
    static int seleteDelIndex() {
        int n;
        cout << "삭제하고자 하는 도형의 인덱스 >> ";
        cin >> n;
        return n;
    }
    static void showAll(vector<Shape*> &v, vector<Shape*>::iterator &it) {
        int i=0;
        for(it = v.begin();it!=v.end(); it++, i++){
            cout << i << ": ";
```

```
            v.at(i)->paint();
        }
    }
};

class GraphicEditor {
    vector<Shape*> v;
    vector<Shape*>::iterator it;
public:
    GraphicEditor() {
        cout << "그래픽 에디터입니다.₩n";
        start();
    }
    void start() {
        while(true){
            int n;
            n = UI::seleteMenu();
            switch(n){
                case 1:
                    n = UI::seleteShape();
                    switch(n){
                        case 1:
                            v.push_back(new Line());
                            break;
                        case 2:
                            v.push_back(new Circle());
                            break;
                        case 3:
                            v.push_back(new Rect());
                            break;
                        default:
                            cout << "잘못 선택하셨습니다.₩n";
                            break;
                    }
                    break;
                case 2:{
                    n = UI::seleteDelIndex();
                    if(n >= v.size() || n < 0){
                        cout << "없는 인덱스 입니다.₩n";
                        break;
                    }
                    it = v.begin();
                    Shape* tmp = *(it+n);
                    v.erase(it+n);
                    delete tmp;
                    break;
                }
                case 3:
                    UI::showAll(v, it);
                    break;
                case 4:
                    return;
                default:
                    cout << "잘못 입력하셨습니다.₩n";
```

```
                    break;
                }
            }
        }

};

int main () {
    new GraphicEditor();
}
```

----문제 정의----

도형을 삽입, 삭제, 조회할 수 있는 간단한 그래픽 에디터를 구현합니다. 프로그램은 다음과 같은 작업을 지원합니다:
선(Line), 원(Circle), 사각형(Rectangle)을 삽입
특정 도형을 삭제
저장된 모든 도형을 조회

----문제 해결 방법, 아이디어들----

클래스 구조 설계측면입니다.
Shape 클래스: 모든 도형의 공통 부모 클래스입니다. 순수 가상 함수 draw()를 선언하여 다형성을 제공합니다.
Circle, Rect, Line 클래스: 각각 Shape 클래스를 상속받아 draw()를 구현합니다. 도형의 이름을 출력하는 역할을 합니다.
UI 클래스: 사용자 인터페이스 관련 기능을 담당합니다. 메뉴 선택, 도형 선택, 삭제할 인덱스 선택, 모든 도형 조회 등의 기능을 제공합니다.
GraphicEditor 클래스: 전체 프로그램의 로직을 관리하며, 사용자 입력에 따라 적절한 작업(삽입, 삭제, 조회)을 수행합니다.
데이터 관리측면입니다.
vector<Shape*>: 삽입된 도형들을 동적으로 저장하고 관리합니다.
iterator를 사용하여 벡터의 요소를 순회하고 삭제합니다.
메모리 관리측입니다.
동적으로 생성된 도형 객체는 삭제 시 delete를 사용하여 명시적으로 해제합니다. 이를 통해 메모리 누수를 방지합니다.

----아이디어 평가----

다형성 활용: 도형 클래스를 Shape로 통합하고, paint() 메서드를 통해 동적으로 호출하여 객체지향 프로그래밍의 장점을 살렸습니다.
벡터 기반 데이터 관리: STL의 vector를 사용하여 데이터를 동적으로 관리하며, 삽입, 삭제 작업이 간단하고 효율적입니다.
메모리 관리: 삭제 시 동적으로 생성된 객체를 명확히 해제하여 메모리 누수를 방지했습니다.

----문제를 해결한 키 아이디어 또는 알고리즘 설명----

객체지향 프로그래밍(OOP)와 다형성
도형마다 상속받은 draw()를 오버라이딩하여 각 도형의 출력 방식을 정의했습니다.
이를 통해 도형의 종류와 관계없이 paint() 메서드 호출만으로 도형을 출력할 수 있도록 했습니다.
벡터를 이용한 도형 관리

도형 삽입 시 vector에 포인터를 저장하며, 삽입 순서를 유지합니다.
삭제 시 사용자로부터 인덱스를 입력받아 해당 도형을 벡터에서 제거하고, 동적으로 할당된 메모리를 해제합니다.