<객체지향프로그래밍_11주차과제_소스구현설명>

7조(202104140김경록,202104276이선우>

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

class Shape {
	Shape* connect;
protected:
	virtual void draw() = 0;
public:
	Shape() { connect = NULL; }
	virtual ~Shape() { }
	void paint() { draw(); }
	void setconnect(Shape* p) { this->connect = p->connect; }
	Shape* getconnect() { return connect; }
	Shape* add(Shape* p) { this->connect = p; return p; }
};

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

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

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

class UI {
public:
	static int memu() {
		cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
		int answer;
		cin >> answer;
		return answer;
	}
	static int menushape() {
		cout << "선:1, 원:2, 사각형:3 >> ";
		int answer;
		cin >> answer;
```

```
		return answer;
	}
	static int deleteindex() {
		cout << "삭제하고자 하는 도형의 인덱스 >> ";
		int answer;
		cin >> answer;
		return answer;
	}
};

class GraphicEditor {
	Shape* pStart;
	Shape* pLast;
	int count;

public:
	GraphicEditor() { pStart = NULL; count = 0; }
	void Add(int num) {
		switch (num) {
		case 1:
			if (count == 0) {
				pStart = new Line();
				pLast = pStart;
			}
			else
				pLast = pLast->add(new Line());
```

```
                count++;
                break;

        case 2:
                if (count == 0) {
                        pStart = new Circle();
                        pLast = pStart;
                }
                else
                        pLast = pLast->add(new Circle());
                count++;
                break;

        case 3:
                if (count == 0) {
                        pStart = new Rect();
                        pLast = pStart;
                }
                else
                        pLast = pLast->add(new Rect());
                count++;
                break;

        }
}
void Case() {
```

```
	cout << "그래픽 에디터입니다." << endl;
	while (true) {
		switch (UI::memu()) {
		case 1:
			Add(UI::menushape());
			break;
		case 2:
			Del(UI::deleteindex());
			break;
		case 3: {
			Shape* p = pStart;
			for (int i = 0; i < count; i++) {
				cout << i << ": "; p->paint();
				p = p->getconnect();
			}
			break;
		}
		case 4:
			return;

		}
	}
}
void Del(int num) {
	Shape* p = pStart;
	Shape* del = pStart;
```

```
                if (num < count) {
                        for (int i = 0; i < num; i++) {
                                p = del;
                                del = del->getconnect();
                        }
                        if (num == 0)
                                pStart = p->getconnect();
                        else
                                p->setconnect(del);
                        count--;
                        if (count == 1)
                                pLast = pStart;
                        delete del;

                }
        }
};

int main() {
        GraphicEditor* editor = new GraphicEditor;
        editor->Case();
        delete editor;
}
```

----문제 정의----

이 문제는 그래픽에디터라는 편집기라는 것을 만들어 1번을 선택하면 선,원,사각형 중에 선택하여 인덱스에 삽입하는 것, 2번은 인덱스에 있는 도형을 삭제, 3번은 인덱스 보기, 4번은 프로그램을 종료하는 시스템을 만드는 프로그램이다.

----문제 해결 방법, 아이디어들----

먼저 문제에서 적어놓은 GraphicEditor, UI 클래스를 만들고 GraphicEditor 클래스 안에는 pStart, pLast를 통해 연결리스트로 도형들을 관리할 수 있게 해주었다. UI 클래스 안에는 말그대로 화면 출력 및 키 입력함수가 나오게 하도록 static 멤버들로만 구성하였다. 또한 virtual을 통한 draw() 가상 함수를 선언해서 각각의 클래스에서 자신만의 것을 구현하게 만들었다. 그리고 Shape이라는 클래스를 상속받은 public의 Circle, Line, Rect 클래스를 통해 동일한 구조를 따르게 만들었다.

----아이디어 평가----

문제 해결을 위해서 제시한 위의 아이디어들, 특히 가상함수와 상속의 아이디어를 통해 클래스를 각각 동일한 구조 그리고 다른 구현을 하게 만든 것이 코드의 길이와 중복을 줄이는 것에 도움이 되었다.

----문제를 해결한 키 아이디어 또는 알고리즘 설명----

결국, 키 아이디어는 가상함수, 상속 그리고 연결리스트이다. 연결리스트를 통해 도형의 삽입, 삭제를 가능하게 하였고 가상함수를 통해 draw()를 오버라이딩해서 자신의 도형을 그리도록 유도하였고 마지막으로 상속을 통해 부모클래스를 상속받아 간결한 클래스로 각각 동작을 구현시키게 만든 것이다.