Homework#05

문제 정의

1:UI, GraphicEditor의 클래스를 정의해라(단, GraphicEditor의 클래스를 정의할 땐, 각 도형들의 클래스가 정의되어 있는데 그 중, 공통으로 묶이는 변수는 Shape 클래스로 처리하고 다른 변수들은 각각 코딩하도록 한다.)

2: GraphicEditor에는 삽입, 삭제, 모두보기, 종료 4가지의 명령을 가지고 있으며 각각 1,2,3,4의 문자열 값에 따라 명령을 수행한다. 고로 각 문자열값에 실행되는 결과값을 코딩해야한다.

3: 텍스트에 해당하는 명령의 값을 수행하도록 코딩하고 특정 문자값이 일치할 때까지 프로그램을 무한 반복(While)시킨다.

문제 해결 방법

1: 메인화면을 담당하면서 동시에 내가 입력을 하고 결과를 도출 할 UI 클래스부터 정의한다.(메인 텍스트와 삽입, 삭제에 물음이 있으므로 3개를 정의한다.)

```
class UI {
public:
    static int main_menu() {
        int n;
        cout << endl << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
        cin >> n;
        return n;
    }
    static int show_shape() {
        int n;
        cout << endl << "선:1, 원:2, 사각형:3 >> ";
        cin >> n;
        return n;
    }
    static int del_shape() {
        int n;
        cout << endl << "삭제하고자 하는 도형의 인덱스 >> ";
        cin >> n;
        return n;
    }
};
```

2: Shape 클래스는 이전 9장 강의에서 사용한 예시를 사용하고 저기서 getNext()를 결과값 도출을 위한 인자로 setNext() 함수를 삭제를 할 때 어느 인덱스 값인지를 찾기 위한 변수로서 정의한다.

```
class Shape { // Node
    Shape* next;
protected:
    virtual void draw() { cout << "--Shape--" << endl; }
public:
    Shape() { next = NULL; }
    virtual ~Shape() {}
    Shape* add(Shape* p) {
        this->next = p;
        return p;
    }
    Shape* getNext() { return next; }
    void paint() { draw(); }
    void setNext(Shape* p) { this->next = p->next; }
};
```

←Shape()를 default로 설정하고 순수 추상 변수를 통해 draw()를 헷갈리지 않도록 설정

3.GraphicEditor에는 도형 삽입, 삭제, 모두보기를 제공하므로 GraphicEditor()에 노드의 크기와 시작지점, 끝지점의 변수를 설정하고 각 숫자값에 해당하는 결과를 객체로서 도출 시키도록 한다.

```
class GraphicEditor {
    int node_size;
    Shape* pStart;
    Shape* pLast;
public:
    GraphicEditor() {
        pStart = NULL;
        node_size = 0;
    }
```

←우선 default값(?)인 비어있는 스택을 선언

```
int run() {
    cout << "그래픽 에디터입니다." << endl;
    while (true) {
        int n = UI::main_menu();
        switch (n) {
        case 1: {
            n = UI::show_shape();
            input_shape(n);
            break;
        }
        case 2: {
            if (pStart == NULL) {
                cout << "리스트가 없습니다." << endl;
                break;
            }
            n = UI::del_shape();
            del(n);
            break;
        }
        case 3: {
            show();
            break;
        }
        case 4: {
            exit(0);
        }
        default:
            cout << "메뉴를 잘못 선택했습니다." << endl;
        }
    }
}
```

← 1,2,3,4에 해당하는 도출해야 되는 값을 맨 위에서 쓰인 UI에서 main_menu로부터 인수를 부여받은 n을 사용해 이 후, Switch ~case 문으로 구분했음.

4: 도출해야하는 값을 하나씩 객체로 설정했는데 그럼 그 객체를 정의를 해야함.

4-1:input_Shape()(삽입) : 위에 명령에 선:1, 원:2, 사각형:3의 명령에서 인수를 부여받고 이 또한, switch ~case문으로 삽입 한 후 노드의 사이즈(top)을 1 올린다.

```
void input_shape(int n) {
    switch (n) {
    case 1: {
        if (node_size == 0) {
            pStart = new Line();
            pLast = pStart;
        }
        else
            pLast = pLast->add(new Line());
        node_size++;
        break;
    }
```

← Line부분만 보여드렸지만 저 형태랑 똑같이 클래스만 바꾸어 정의했습니다.

4-2:del(int n)(삭제) : n에서 입력 받은 값을 target_node(포인터 지역 변수이며 해당 노드를 찾는 변수) priv_node(target_node의 -1 변수 -> 교체하기 위함)을 사용해 노드의 주소를 찾고 같으면 삭제, 이 후, 노드를 하나씩 앞당기며 빈 노드가 없게 함.

```
bool del(int n) {
    int k = 0;
    Shape* target_node = pStart;
    Shape* priv_node = 0;
    if (n == 0) {
        pStart = pStart->getNext();
        delete target_node;
    }
    else {
        while ((target_node != NULL) && (k < n)) {
            priv_node = target_node;
            target_node = target_node->getNext();
            k++;
        }
        if (target_node == NULL) {
            cout << "없는 노드입니다.₩n";
            return false;
        }
        else {
            priv_node->setNext(target_node);
            delete target_node;
        }
    }
    node_size--;
}
```

4-3: show()(모두보기) : p의 첫 값부터 내가 추가한 모든 노드까지 반복문으로 1씩 올리며 검토하며 검토할 때 마다 해당 노드에 뭘 삽입했는지 보여준다.

```
void show() {
    Shape* p = pStart;
    int i = 0;
    if (p == NULL)
        cout << "List Empty₩n";
    else
        while (p != NULL) {
            cout << i << ": ";
            p->paint();
            p = p->getNext();
            i++;
        }
}
```

문제 해결을 위한 아이디어 제공 및 평가

1: 우선 이 프로그램이 특정 값을 이용하면 끝나는 것을 생각해 Switch ~case로 각각에 해당하는 도출값을 설정하는 것이 좋겠다는 생각이 들었고, 영원히 끝나지 않는다는 점을 고려해 While로 프로그램을 시작하는 것이 좋겠다고 생각했다.

2: 해당 도형을 삽입하고 제거한다는 내용으로 자료구조 시간때 배운 push, pop이 생각 났으며 이를 이번 과제에 적용해보면 어떨까 생각하게 됐다. 결론적으론 성공적이었던것 같다.

3: switch ~case로 각 결과값을 도출하는 것은 좋은데, GraphicEditor에 어떻게 이를 표현할 것인가에 대한 생각에서 하나씩 무명클래스 마냥 정의하는 것은 좋지 않다고 생각, 차라리 객체를 만들어 이에 쓰는것이 적합하다고 판단했다.