```
#ifndef SHAPE_H
#define SHAPE_H
#include<iostream>
using namespace std;

class Shape {
    Shape* next;
protected:
    virtual void draw() = 0;
public:
    Shape() { next = NULL; }
    virtual ~Shape() {}
    void paint();
    Shape* add(Shape* s);
    Shape* getNext() { return next; }
    void setNext(Shape* s) { this->next = s->next; }
};

class Circle :public Shape {
protected:
    virtual void draw();
};

class Rect :public Shape {
protected:
    virtual void draw();
};

class Line :public Shape {
protected:
    virtual void draw();
};

class GraphicEditor {
    Shape* pStart;
    Shape* pLast;
    int count;
public:
    GraphicEditor();
    void Make(int num);
    void Delete(int num);
    void call();
};


#endif
```

도형을 관리할 Shape 추상 클래스를 만들어준다 수업시간에 만들었던 형태의 Shape 클래스에 삭제를 진행할때 next를 수정해줄 setNext 함수를 만들어주었다. 그리고 Shape를 상속받는 각각의 Circle, Rect, Line 클래스를 만들어 주었다.

그래픽 편집기 GraphicEditor 클래스를 만들고 Shape형식의 pStart, pLast라는 변수를 선언해주었다. pStart, pLast는 생성된 도형의 객체의 순서를 파악하기 위한 용도이다. int count는 현재 객체가 몇개 생성되어있는지를 파악하기위한 변수이다.
GraphicEditor()의 생성자를 선언해주고 Make(int num), Delete(int num), call() 함수를 선언해준다. Make는 도형을 생성할 때 쓸 함수이고 Delete는 도형을 지울 때 call은 UI를 불러올 때 쓰인다.

20210423 신윤섭

```h
#ifndef UI_H
#define UI_H
#include<iostream>
class UI {
public:
    static int mainMenu();
    static int shapeMenu();
    static int deleteMenu();
};
#endif
```

```cpp
#include<iostream>
#include"UI.h"
using namespace std;
int UI::mainMenu() {
    int n;
    cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
    cin >> n;
    return n;
}

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

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

UI를 담당할 class UI를 생성해 주었다.
mainMenu()함수는 처음 뜨는 UI로 실행시 도형을 삽입, 삭제, 모두보기 또는 종료 할지를 받아서 n에 저장한다음 리턴한다. shapeMenu또한 어떤 도형을 삽입할지 1, 2, 3,을 받아서 리턴한다.
deleteMenu는 삭제하고자하는 도형이 몇번째 도형인지를 입력받아 리턴한다.

```cpp
#include<iostream>
#include "Shape.h"
#include"UI.h"
using namespace std;

void Shape::paint() {
    draw();
}

Shape* Shape::add(Shape* p) {
    this->next = p;
    return p;
}

void Circle::draw() {
    cout << "Circle" << endl;
}

void Rect::draw() {
    cout << "Rectangle" << endl;
}

void Line::draw() {
    cout << "Line" << endl;
}
```

클래스의 구현부이다. Shape의 paint함수 실행시 draw()함수을 실행 하는데 Shape 클래스의 draw()함수는 순수가상 함수이기 때문에 실행시 파생 클래스의 draw()함수가 실행된다. draw()함수는 각 객체가 어떤 도형인지를 출력하는 함수이다.

add함수는 Shape객체의 포인터를 인수로 받아서 현재객체의 next 포인터를 다음객체를 가리키게 하는 함수이다.

```
GraphicEditor::GraphicEditor() {
    pStart = NULL;
    pLast = NULL;
    count = 0;
    call();
}

void GraphicEditor::Make(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;
    }
}
```

처음 GraphicEditor클래스의 객체를 생성하면 pStart, pLast는 NULL을 할당해 비워두고, 현재 생성된 도형은 없으니까 count = 0, 그리고 call()함수로 UI를 불러온다.
Make(int num)함수는 도형을 어떤것을 삽입 할껀지를 num으로 받아서 1이면 Line 객체를 동적 생성한다. 이때 count가 0이면 처음으로 생성한 것이므로 동적 생성한 객체를 pStart와 pLast로 가리킨다.
만약 처음이 아니라면 add함수를 사용해 pLast가 새로이 동적생성한 객체를 가리키게 한다. 그다음 count를 1더해서 현재 도형의 수가 몇개인지 갱신한다음 break해서 스위치문을 탈출한다. 나머지 Circle, Rect객체도 똑같이 생성한다.

```
void GraphicEditor::Delete(int num) {
    Shape* p = pStart;
    Shape* del = pStart;

    if (num < count) {
        for (int i = 0; i < num; i++) {
            p = del;
            del = del->getNext();
        }
        if (num == 0)
            pStart = p->getNext();
        else
            p->setNext(del);
        count--;
        if (count == 1) pLast = pStart;
        delete del;
    }
    else
        cout << "인덱스를 잘못 입력하셨습니다." << endl;
}
```

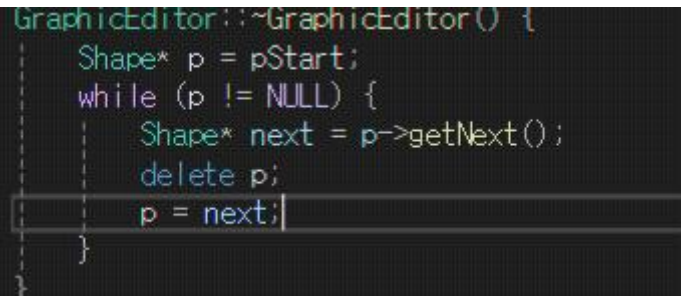
Delete(int num)함수는 해당 도형을 찾아서 삭제하는 함수이다.
이 함수에서 del은 삭제할 도형을 뜻하고, p는 del의 전번째 도형을 뜻한다.
삭제할 도형을 첫번째 도형부터 찾아야 하기 때문에 del은 첫번째 도형인 pStart가 된다.
처음 for문으로 num으로 받은 수만큼 del을 다음으로 이동시킨다.
p가 del과 같은곳을 가리킨다음 del은 해당 객체의 next포인터가 가리키는
객체를 가리키게 한다. 이렇게 하면 p다음 del의 순서로 도형을 조사할 수있다. 만약 num이 0이면 첫번째 도형을 삭제한다는것을 뜻한다. 그렇게 되면 for문이 동작을 하지않아 p와 del은 초깃값이 된다.
그러므로 첫번째 도형을 가리키는 pStart를 p->getNext()즉 2번째 도형을 가리키게 한다. 이외의 경우에는 p->setNext(del)을해서 del전의 도형인 p가 del의 다음 도형을 가리키게 한다. 그다음 만약 count=1이라면 도형은
총 1개 있다는 뜻이고 마지막 도형을 뜻하는 pLast는 pStart와 같은 도형을 가리키게 한다.

```
void GraphicEditor::call() {
    bool exit = true;
    cout << "그래픽 에디터입니다." << endl;
    while (exit) {
        switch (UI::mainMenu()) {
        case 1:
            Make(UI::shapeMenu());
            break;
        case 2:
            Delete(UI::deleteMenu());
            break;
        case 3: {
            Shape* p = pStart;
            for (int i = 0; i < count; i++) {
                cout << i << ": "; p->paint();
                p = p->getNext();
            }
            break;
        }
        case 4:
            exit = false;
            break;
        }
    }
}
```

call함수는 처음에 불리는 함수로 처음에 bool형식의 exit를 true로 선언해 준다. 이는 후에 그래픽 에디터를 종료할때를 판단하는 변수이다.
그다음 "그래픽 에디팅비니다."를 출력한다음 while문에 exit를 매개변수로받아. exit가 true일때 동안 무한 반복하게 한다.
switch문의 매개변수로 UI::mainMenu()함수의 리턴값을 받는데 이는 위에서 봤던 UI클래스의 함수이다. 1을 받으면 삽입, 2는 삭제, 3은 현재 도형들을 보여주고, 4는 exit를 false로 바꾸어서 그래픽 에디터를 종료 시킨다.

case3을 보면 Shape형식의 포인터 p를 생성한다음 pStart(처음 도형)을 가리킨다.
그다음 for문으로 0부터 count(현재 생성된 도형의 갯수)만큼 출력을 진행하는데 출력은 i번째의 p->paint() (이때 paint는 각 도형객체의 draw()함수를 실행한다)를 출력한다. 그다음 p를 p의 다음 도형을 가리키게해서 첫도형부터 마지막 도형까지 출력한다.

추가로 그래픽에디터가 종료 될때 동적생성된 Shape객체도 메모리를 반환시킨다.



메인 함수는 GraphicEditor객체를 동적 생성한다음
종료되면 할당된 메모리를 반환한다.

```
#include<iostream>
#include"Shape.h"
#include"UI.h"
using namespace std;

int main() {
    GraphicEditor* gEditor = new GraphicEditor;
    delete gEditor;
}
```

```
그래픽 에디터입니다.
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 1
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 2
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 3
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 2
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 1
선:1, 원:2, 사각형:3 >> 3
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Circle
2: Rectangle
3: Circle
4: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 2
삭제하고자 하는 도형의 인덱스 >> 2
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 3
0: Line
1: Circle
2: Circle
3: Rectangle
삽입:1, 삭제:2, 모두보기:3, 종료:4 >> 4
```