과제5 소스코드 설명 - 서신우

문제 정의

- Shape 객체를 생성하는데 Line, Circle, Rect 클래스의 객체들이 사용할 draw() 메소드를 가상 함수로 선언해야 한다.
- 각 인스턴스의 next 필드에 다른 인스턴스를 가리키게 해서 도형들이 서로 연결되어 있는 구조로 만들어야 한다.
- 사용자에 입력에 따라 새로운 도형을 추가, 삭제, 리스트로 보여주기 등의 기능을 구현해야 한다.

문제 해결

- Shape 클래스 안에 다음 연결될 도형을 담아둘 next 필드를 만들고 그 next 필드에 새로운 도형을 연결시킬 수 있는 add() 함수, 다음 도형을 리턴해주는 getNext() 함수를 구현했다.
- GraphicEditor 클래스는 안내문을 출력하고 사용자에게 입력값을 받는 함수들을 구현했다.
- 사용자에게 '도형 추가' 입력을 받으면 어떤 도형을 추가할 것인지 확인하고 해당 도형을 현재 도형의 next 필드에다 추가시켜주는 기능을 구현했다.
- 사용자에게 '모두보기' 입력을 받으면 처음 도형으로부터 줄줄이 연결된 모든 도형들을 보여주는 기능을 구현했다.
- '삭제' 기능 끝끝내 구현하지 못했다..

Code

Shape.h

이 클래스를 상속받는 클래스에서도 draw() 함수를 구현하도록 virtual 로 선언했다.

```
class Shape {
protected:
        virtual void draw();
~~
```

```
~~
};
```

GraphicEditor.h

이 클래스에서는 사용자로부터 입력값을 받거나 안내문을 출력하는 함수들을 선언했다.
간단한 작업이기에 전부 static 을 사용해 클래스 인스턴스를 만들지 않고도 호출할 수 있도록 했다.

```
class GraphicEditor {
public:
    static string action;
    static string shape;
    static string deleteNum;

    static void showAction(){cout << "삽입:1, 삭제:2, 모두보기:3,
    static void showShape(){cout << "선:1, 원:2, 사각형:3 >> ";
    static void setAction(){
        cin >> action;
    }
    static void setShape(){
        cin >> shape;
    }
    static string getAction(){return action;}
    static string getShape(){return shape;}
    static void showDelete(){cout << "삭제하고자 하는 도형의 인덱스
    static void setDeleteNum(){
        cin >> deleteNum;
    }
};
```

main.cpp

우선 코드 윗 부분에 `Shape*` 타입의 변수 3개를 선언했다.
pStart 는 도형들의 연결에서 첫 번째를 맡을 부분이다.

```
int main() {
        Shape* pStart=NULL;
        Shape* pLast;
        Shape* next;

        ~~~
        ~~~
}
```

프로그램은 계속 반복되어야 하므로 while() 함수 안에서 돌아가게 해주었다.
첫 번째로 추가, 삭제, 전부보기, 종료 와 같은 선택이 계속 반복되도록 했다.
switch 문을 사용해 선택에 따라 다른 동작이 실행되도록 했다.

```
while(isRunning)
{
        GraphicEditor::showAction();
        GraphicEditor::setAction();

        switch (stoi(GraphicEditor::getAction()))
        {
                case 1:
                    ~~
                    break;
                case 2:
                    ~~
                    break;
                ~~~
                ~~~
        }
}
```

사용자가 추가 동작을 선택했다면 어떤 도형을 추가할 것인지 묻고 해당 도형을 현재 도형의 next 필드에다가 저장한다. 만약 pStart 가 NULL, 즉 첫 번째 도형이라면 pStart 에 바로 동적 생성을 하도록 했다.

```
{
    GraphicEditor::showShape();
    GraphicEditor::setShape();

    if (GraphicEditor::getShape() == "1")
    {
        if (pStart == NULL)
        {
            pStart = new Line();
            pLast = pStart;
        }
        else
        {
            Line *line = new Line();
            pLast->next = line;
            pLast = line;
        }

    }
    else if (GraphicEditor::getShape() == "2")
    {
        if (pStart == NULL)
        {
            pStart = new Circle();
            pLast = pStart;
        }
        else
        {
            Circle *circle = new Circle();
            pLast->next = circle;
            pLast = circle;
        }
    }
    else if (GraphicEditor::getShape() == "3")
    {
        if (pStart == NULL)
        {
            pStart = new Rect();
```

```
            pLast = pStart;
        }
        else
        {
            Rect *rect = new Rect();
            pLast->next = rect;
            pLast = rect;
        }
    }
    else
    {
        cout << "잘못 입력하셨습니다." << endl;
        break;
    }
}
```

사용자가 '모두보기' 동작을 실행했다면 첫 도형(pStart) 부터 getNext() 함수로 뒤에 이어져있는 도형으로 이동하면서 각 도형을 그리도록 했다.

```
Shape* p = pStart;

int index = 0;

while ( p != NULL)
{
    cout << index << " : ";
    p->paint();
    p = p->getNext();
    index++;
}
```

안타깝게도 '식제' 동작은 구현하지 못했다..
많이 실행해 보았지만 번번이 실패했다..
더욱 열심히 정진하겠다.