코드

#include <iostream>

using namespace std;

class Shap { //기본클래스

protected: //멤버 선언한 클래스 및 파생클래스에서 접근 가능

int x, y;

Shap(int x, int y) { //생성자

this->x = x; this->y = y; //도형의 위치 정보 저장

cout << "Shap 생성자" << endl;

}

~Shap(){ //소멸자

cout << "Shap 소멸자" << endl;

}

};

class Rectangle :public Shap { //파생클래스//Shap 상속

int width, height;

public:

//도형의 위치.가로/세로 길이 등을 매개변수로 전달받아 저장

Rectangle(int x,int y,int width,int height):Shap(x,y){ //Shap생성자 명시적 호출//위치 정보 전달

this->width = width; this->height = height;

cout << "Rectangle 생성자:" << endl;

}

//사각형의 위치 정보 출력

void draw() {

cout << "Rectangle drawn at : (" << x << "," << y << ")" << endl;

}

//사각형의 넓이 출력

double getArea() {

return this->width \* this->height;

}

//소멸자

~Rectangle() {

cout << "Rectangle 소멸자" << endl;

}

};

class Circle:public Shap { //파생클래스//Shap 상속

int radius;

public:

//도형의 위치.반지름 등을 매개변수로 전달받아 저장

Circle(int x,int y,int radius):Shap(x, y) {//Shap생성자 명시적 호출//위치 정보 전달

this->radius = radius;

cout << "Circle 생성자:" << endl;

}

//원의 위치 정보 출력

void draw() {

cout << "Circle drawn at : (" << x << "," << y << ")" << endl;

}

//원의 넓이 출력

double getArea() {

return this->radius \* this->radius \* 3.14;

}

//소멸자

~Circle() {

cout << "Circle 소멸자" << endl;

}

};

int main() {

Rectangle r(10, 20, 100, 100); // (10, 20)위치에 가로 100, 세로 100인 사각형 객체 생성

r.draw();

cout << "사각형 면적 = " << r.getArea() << endl << endl;

Circle c(120, 150, 40); // (120,150)위치에 반지름 40인 원을 생성

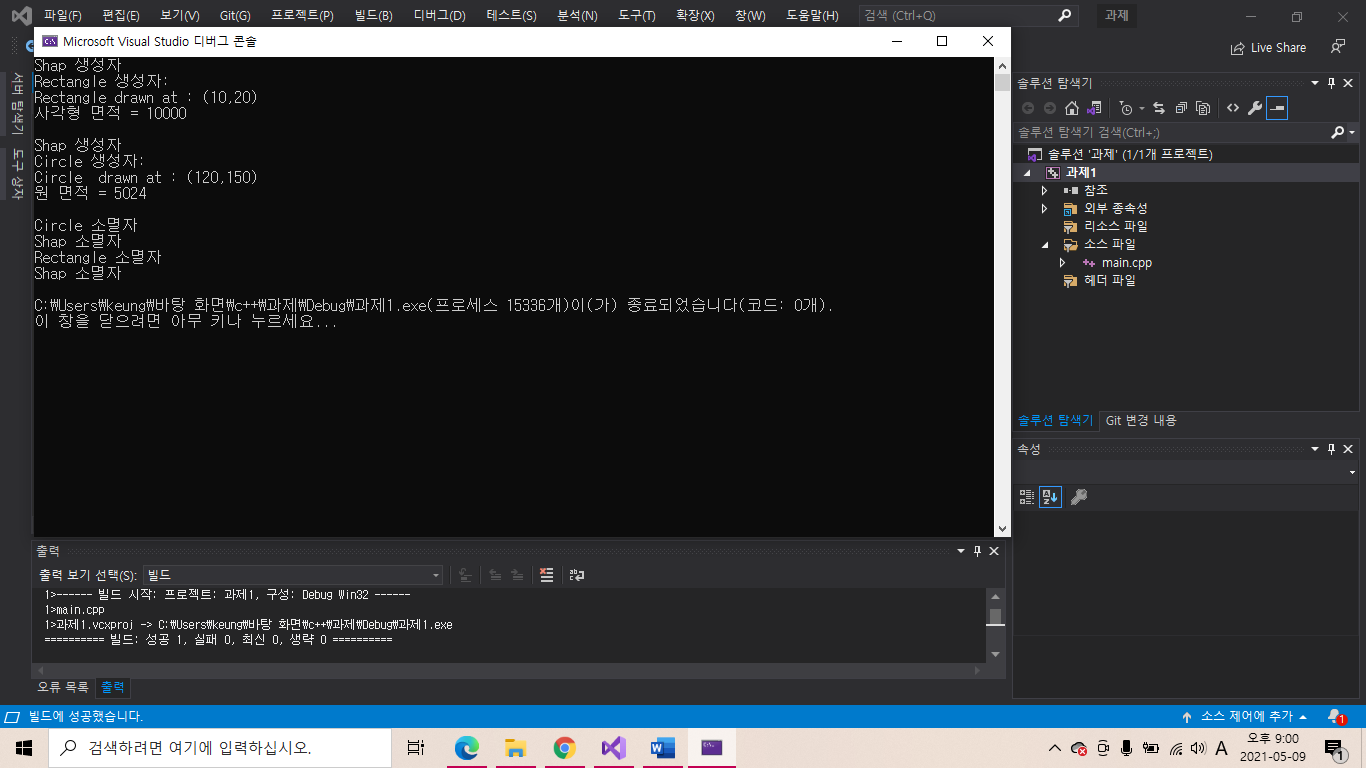
c.draw();

cout << "원 면적 = " << c.getArea() << endl << endl;

return 0;

}

출력화면



후기

* 생성자 소멸자 실행순서: 생성자는 기본클래스의 생성자부터 실행된다. 이 코드에서는 사각형 객체 생성->원 객체 생성 했으므로 생성자는 Shap->사각형-> Shap -> 원 순서로 실행된다. 반면에 소멸자는 생성자 실행순서와는 반대로 실행된다. 따라서 원->Shap->사각형-> Shap 순서로 실행된다
* 기본클래스 생성자 선택 방법: 명시적인 기본클래스 생성자를 선택하였다. 기본클래스에서 도형의 위치 정보인 X,Y 값을 저장해주고 싶었기 때문에 명시적으로 X,Y값을 전달받아 값을 저장하는 Shap클래스 생성자를 작성해주었다.
* X, Y 좌표에 대한 접근지정자: 파생 클래스에서 매개변수 x,y를 받아 값을 저장해주는 과정이 필요하기 때문에 x,y를 다른 파생 클래스에서 접근할 수 있는 protected로 해주었다.

\*X, Y 좌표를 포함해서 기본 클래스를 public으로 하지 않은 이유는 Shap 클래스는 구체화되어 있지 않은 도형이기 때문에 main()에서 따로 객체를 생성하고 호출할 일이 없을 것이라 생각해 파생클래스에서만 접근할 수 있도록 접근제한을 두었다.

* 전체 학습 총평: 접근지정자나 기본클래스의 생성자 선택 방법에 있어서 고민을 조금 했던 것 같다. 크게 어려운 내용은 아니였지만 프로그램 전체 실행에 있어서 어떤 방식을 사용해야 할지 생각하는 과정이 있었다. 생성자와 소멸자 실행 순서는 앞선 강의에서 배워서 알고는 있었지만 이번 과제를 하면서 다시 한 번 신경쓰면서 개념을 익힐 수 있었던 것 같다.