#include <iostream>

#include <vector>

using namespace std;

class Shape { //기본클래스

protected: //멤버 선언한 클래스 및 파생클래스에서 접근 가능

int x, y;

Shape(int x, int y) { //생성자

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

}

virtual ~Shape() {} //소멸자

public:

virtual void draw() = 0; //순수 가상 함수 draw()작성

virtual double getArea() = 0; //순수 가상 함수 getArea()작성

};

class Rectangle :public Shape { //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) :Shape(x, y) { //Shap생성자 명시적 호출//위치 정보 전달

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

}

//사각형의 위치 정보 출력

virtual void draw() {

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

}

//사각형의 넓이 출력

virtual double getArea() {

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

}

//소멸자

virtual ~Rectangle() {}

};

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

int radius;

public:

//도형의 위치.반지름 등을 매개변수로 전달받아 저장

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

this->radius = radius;

}

//원의 위치 정보 출력

virtual void draw() {

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

}

//원의 넓이 출력

virtual double getArea() {

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

}

//소멸자

virtual ~Circle() {}

};

int main() {

vector<Shape\*> vShape; //Shape 포인터 변수만 들어가는 vector

vShape.push\_back(new Rectangle(10, 10, 30, 40));

vShape.push\_back(new Circle(30, 30, 5));

vShape.push\_back(new Rectangle(20, 30, 10, 10));

vector< Shape\*>::iterator it; //iterator 변수

double area\_Sum=0; //원소들의 면적의 총합을 저장하는 변수

//순차 출력 및 면적 총합계 계산 코드 추가

for (auto it = vShape.begin(); it != vShape.end(); it++){

//it은 원소에 대한 포인터이므로 실질적으로 it에는 실제 도형 객체의 주소값의 주소값이 들어감

Shape\* sPoint = \*it; //\*it 사용을 통해 sPoint는 실제 도형의 주소값을 가르킨다.

sPoint->draw(); //포인터 변수 sPoint이므로->를 이용해 멤버에 접근

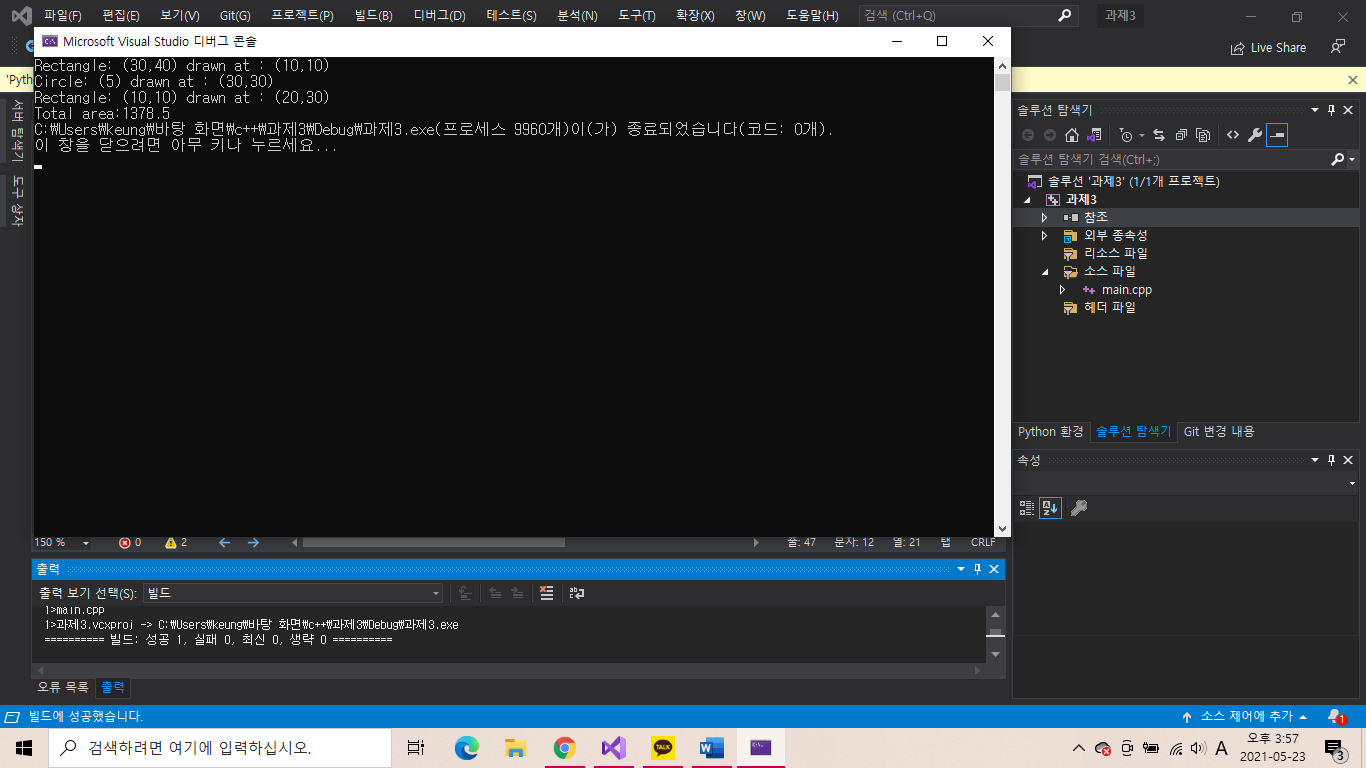
area\_Sum += sPoint->getArea();

}

cout << "Total area:"<<area\_Sum;

return 0;

}



벡터에 저장될 원소의 타입은? : vector<Shape\*> vShape 이므로 Shape 객체의 주소값을 가르키는 포인터 변수가 저장된다. 이때 파생클래스인 Circle이나 Rectangle 객체를 생성하면 업캐스팅 된다.

벡터의 원소 접근을 위해 사용한 방법은? : it에는 실제 도형 객체의 주소값의 주소값을 가르키기 때문에 Shape 타입의 포인터 변수 sPoint를 만들어준 후, 여기에 \*it(실제 도형 객체의 주소값)을 저장해준, 그 포인터 변수를 이용하여 원소의 멤버 함수 등에 접근하였다.

전역변수(총면적)로 static 변수 활용에 대한 생각은? : 총면적은 벡터 하나당 하나만 생기는 것이 므로 전역변수로 선언하는 것도 좋겠다고 생각했다. 다만 전역변수를 선언할 때, 클래스 내에서 선언해주고 클래스 외부에서 초기화 시켜줘야 하는데 보통 벡터는 이미 만들어져 있는 클래스를 #include <vector>하기 때문에 따로 벡터 클래스 코드를 불러오는 것이 좀 번거롭지 않나 싶다.