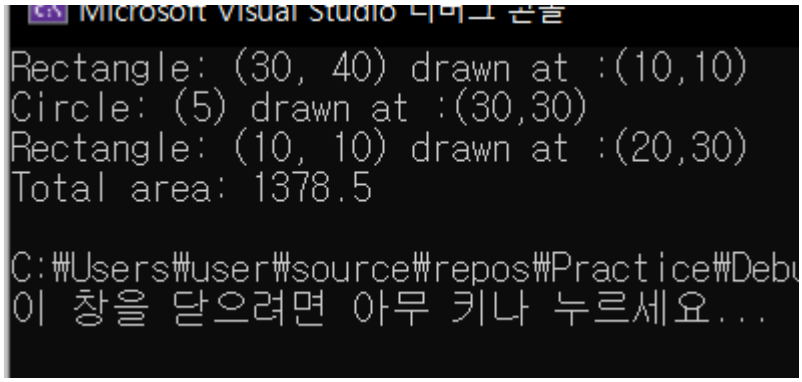
```
#include <iostream>
#include <vector>
using namespace std;

class Shape {
protected:
	int x, y;
public:
	Shape() { }//생성자 선언
	virtual void draw() = 0;//순수 가상 함수 선언
	virtual double getArea() = 0;
	virtual ~Shape();//가상 소멸자
};
Shape::~Shape() {//가상 소멸자 구현
}
class Rectangle : public Shape {
public:
	int width, height;
	Rectangle(int x, int y, int width, int height) { this->x = x, this->y = y, this->width = width, this->height = height;//생성자 선언
	}
	virtual void draw() {//위치 정보 출력
		cout << "Rectangle: "<< "("<<width<<", "<<height <<") "<<"drawn at :(" << x << "," << y << ")" << endl;
	}
	virtual double getArea() {//면적 계산
		int area = (width * height);
		return area;
	}
	virtual ~Rectangle();//가상 소멸자
};
Rectangle::~Rectangle() {//가상 소멸자 구현
}

class Circle : public Shape {
public:
	int radius;
	Circle(int x, int y, int radius) { this->x = x, this->y = y, this->radius=radius;//생성자 선언
	}
	virtual void draw() {//위치 정보 출력
		cout << "Circle: "<<"("<<radius <<") " <<"drawn at :(" << x << "," << y << ")" << endl;
	}
	virtual double getArea() {//면적 계산
		double area = (3.14 * radius*radius);
		return area;
	}
	virtual ~Circle();//가상 소멸자
};
Circle::~Circle() {//가상 소멸자 구현
}
```

```
int main() {
	vector<Shape*> vShape;//클래스 포인터 벡터 생성
	double sum=0;
	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));

	for (auto it = vShape.begin(); it != vShape.end(); it++) {//iterator를 이용하여 모든 원소 탐색
		auto tmp = *it; //it의 클래스 포인터로 자동 선언
		tmp->draw();
		sum += tmp->getArea();
	}
	cout << "Total area: " << sum << endl;
```




벡터는 가변 길이 배열을 구현한 제네릭 클래스이며 벡터에 저장된 원소는 인덱스로 접근 가능하다. 또한 벡터의 길이에 대한 고민을 할 필요가 없다

Auto를 이용해서 변수 타입을 자동으로 설정하여 복잡한 변수 선언을 간소하게 할 수 있고 오류를 줄일 수 있다.

벡터에 저장될 원소의 타입: 포인터 클래스

벡터의 원소 접근을 위해 사용한 방법: For문과 iterator를 이용하여 모든 원소에 접근하고 클래스의 함수를 호출하여 원하는 값을 출력하였다.

전역변수(총면적)로 static 변수 활용에 대한 생각: Static 지역변수는 전역변수보다 상대적으로 안정적이다. 그 이유는 전역변수는 어디서든지 접근가능하기 때문에 전역변수로 여러 소스들을 활용하게 되면 소스가 복잡해진다. 그렇기 때문에 전역변수로 static 변수 활용은 비효율적이라고 생각한다.