**< 객체지향 – Review>**

- 객체(“데이터+메서드”를 가짐)들의 집합 = 프로그램

- 객체는 속성+기능의 집합, 이 때 속성과 기능이 멤버 : 속성은 변수, 기능은 메서드로 정의

- 객체가 인스턴스보다 포괄적이고 넓은 의미로 분석, 설계 단계에서도 사용

<> 코딩 상에서 new 키워드로 만들어내는 것은 주로 인스턴스라고 함

1. 캡슐화

- 코드의 은닉, 숨기기 : 외부 접근 차단, 접근 제한자(private…) 사용 > 외부에서는 메서드로 접근할 수 있게 함

2. 상속

- 코드의 재사용

3. 다형성

3.1 오버로딩(overloading) – 쌓는 것

3.2 오버라이딩(overriding) – 덮는 것

**\* 상속**

- 코드 재사용성 측면에서 사용하는 것이 유리

- OO is a kind of XX : 말이 될 경우에 상속 관계

- 학생은 말하고 먹고 걷고 자고 공부하는 것이 가능

사람 학생

말하기() 공부하기()

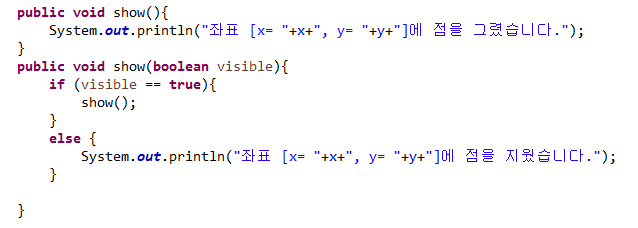
먹기()

걷기()

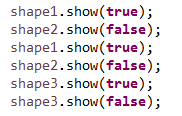
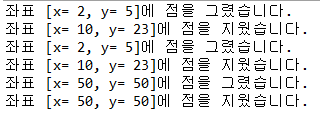
잠자기()

**\* 메서드 오버로딩**

실습예제 6

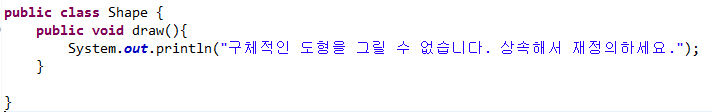


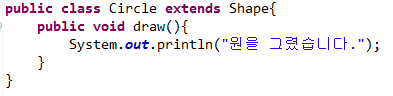
((result))

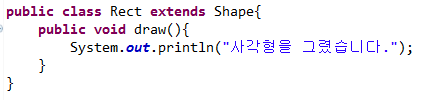
 

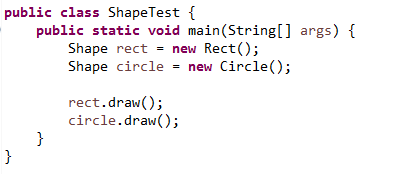
**\* 메소드 오버라이딩**

- 부모클래스의 메서드를 동일한 이름으로 재작성 > 부모클래스 메서드를 무시함





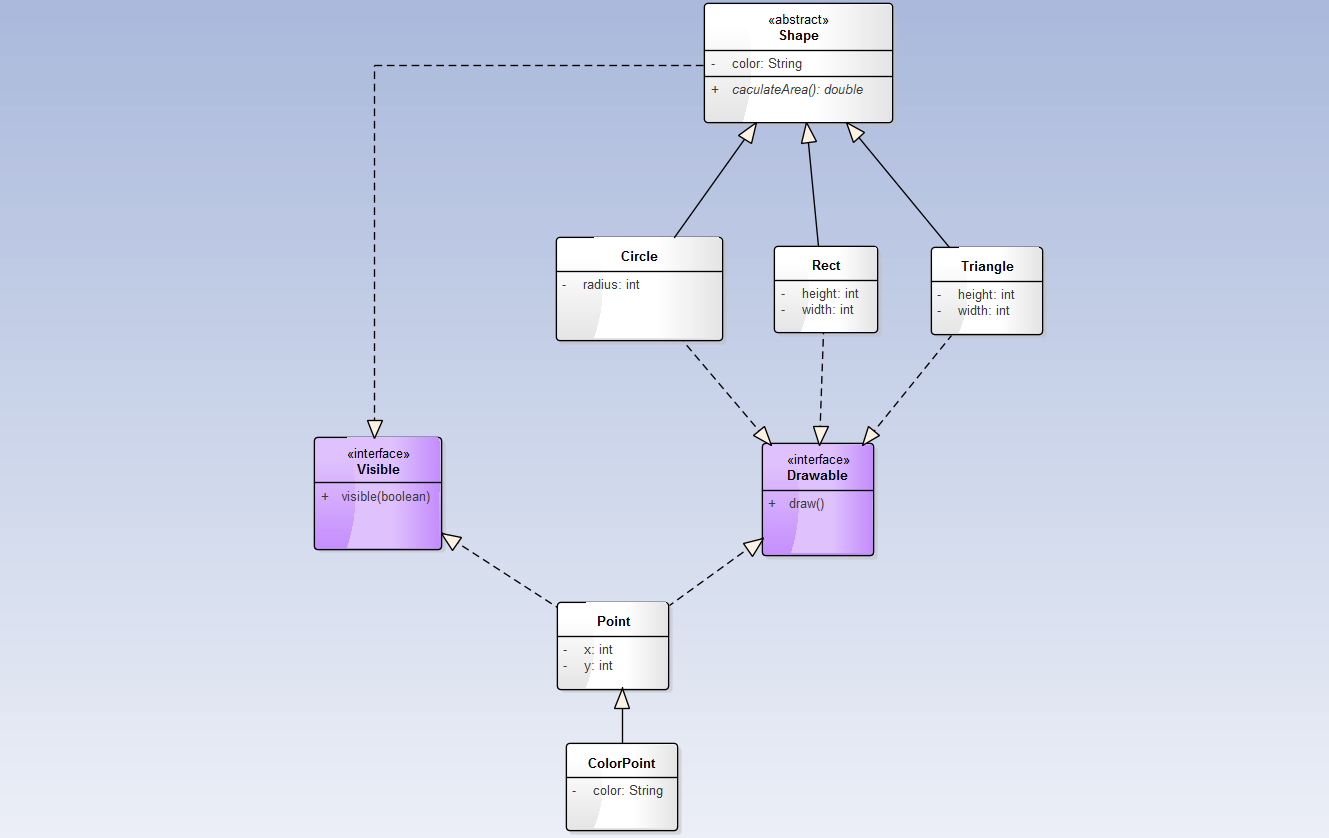




((result))



**\* 예제**

****

**Shape**

**package** Shape;

**public** **abstract** **class** Shape **implements** Visible{

**private** String color;

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**abstract** **public** **double** calculateArea();

}

**Drawable**

**package** Shape;

**public** **interface** Drawable {

**void** draw();

}

**Visible**

**package** Shape;

**public** **interface** Visible {

**void** Visible(**boolean** visible);

}

**Cirlce**

**package** Shape;

**public** **class** Circle **extends** Shape **implements** Drawable{

**private** **int** radius;

**final** **double** pi = 3.14;

**public** **int** getRadius() {

**return** radius;

}

**public** **void** setRadius(**int** radius) {

**this**.radius = radius;

}

**public** **void** draw(){

System.***out***.println("원을 그렸습니다.");

}

@Override

**public** **double** calculateArea() {

**double** area = radius\*radius\*pi;

**return** area;

}

@Override

**public** **void** Visible(**boolean** visible){

**if** (visible == **true**){

draw();

}

**else** {

System.***out***.println("원을 지웠습니다.");

}

}

}

**Rectangle**

**package** Shape;

**public** **class** Rect **extends** Shape **implements** Drawable{

**private** **int** height;

**private** **int** width;

**public** **int** getHeight() {

**return** height;

}

**public** **void** setHeight(**int** height) {

**this**.height = height;

}

**public** **int** getWidth() {

**return** width;

}

**public** **void** setWidth(**int** width) {

**this**.width = width;

}

**public** **void** draw(){

System.***out***.println("사각형을 그렸습니다.");

}

@Override

**public** **double** calculateArea() {

**int** area = width\*height;

**return** area;

}

@Override

**public** **void** Visible(**boolean** visible){

**if** (visible == **true**){

draw();

}

**else** {

System.***out***.println("사각형을 지웠습니다.");

}

}

}

**Triangle**

**package** Shape;

**public** **class** Triangle **extends** Shape **implements** Drawable {

**int** height;

**int** width;

**public** **int** getHeight() {

**return** height;

}

**public** **void** setHeight(**int** height) {

**this**.height = height;

}

**public** **int** getWidth() {

**return** width;

}

**public** **void** setWidth(**int** width) {

**this**.width = width;

}

**public** **void** draw() {

System.***out***.println("삼각형을 그렸습니다.");

}

**public** **double** calculateArea() {

**double** area = height \* width \* 0.5;

**return** area;

}

@Override

**public** **void** Visible(**boolean** visible){

**if** (visible == **true**){

draw();

}

**else** {

System.***out***.println("삼각형을 지웠습니다.");

}

}

}

**Point**

**package** Shape;

**public** **class** Point **implements** Drawable{

**private** **int** x;

**private** **int** y;

**public** Point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

**public** **int** getX() {

**return** x;

}

**public** **void** setX(**int** x) {

**this**.x = x;

}

**public** **int** getY() {

**return** y;

}

**public** **void** setY(**int** y) {

**this**.y = y;

}

**public** **void** draw() {

System.***out***.println("좌표 [x= "+x+", y= "+y+"]에 점을 그렸습니다.");

}

**public** **void** Visible(**boolean** visible){

**if** (visible == **true**){

draw();

}

**else** {

System.***out***.println("좌표 [x= "+x+", y= "+y+"]에 점을 지웠습니다.");

}

}

**public** **void** setColor(String Color){

}

}

**ColorPoint**

**package** Shape;

**public** **class** ColorPoint **extends** Point{

**private** String color;

**public** ColorPoint(**int** x, **int** y) {

**super**(x, y);

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **void** draw() {

System.***out***.println("좌표 [x= "+getX()+", y= "+getY()+"]에 "+color+" 점을 그렸습니다.");

}

**public** **void** Visible(**boolean** visible){

**if** (visible == **true**){

draw();

}

**else** {

System.***out***.println("좌표 [x= "+getX()+", y= "+getY()+"]에 "+color+" 점을 지웠습니다.");

}

}

}

**ShapeTest**

**package** Shape;

**public** **class** ShapeTest {

**public** **static** **void** main(String[] args) {

Rect rect = **new** Rect();

Circle circle = **new** Circle();

Triangle triangle = **new** Triangle();

rect.draw();

circle.draw();

triangle.draw();

Drawable drawable = **new** Point(10, 20);

drawable.draw();

Point Colorpoint = **new** ColorPoint(20, 30);

Colorpoint.setColor("red");

Colorpoint.draw();

// calculate area

rect.setWidth(10);

rect.setHeight(10);

circle.setRadius(10);

triangle.setWidth(10);

triangle.setHeight(10);

System.***out***.println(rect.calculateArea());

System.***out***.println(circle.calculateArea());

System.***out***.println(triangle.calculateArea());

}

}