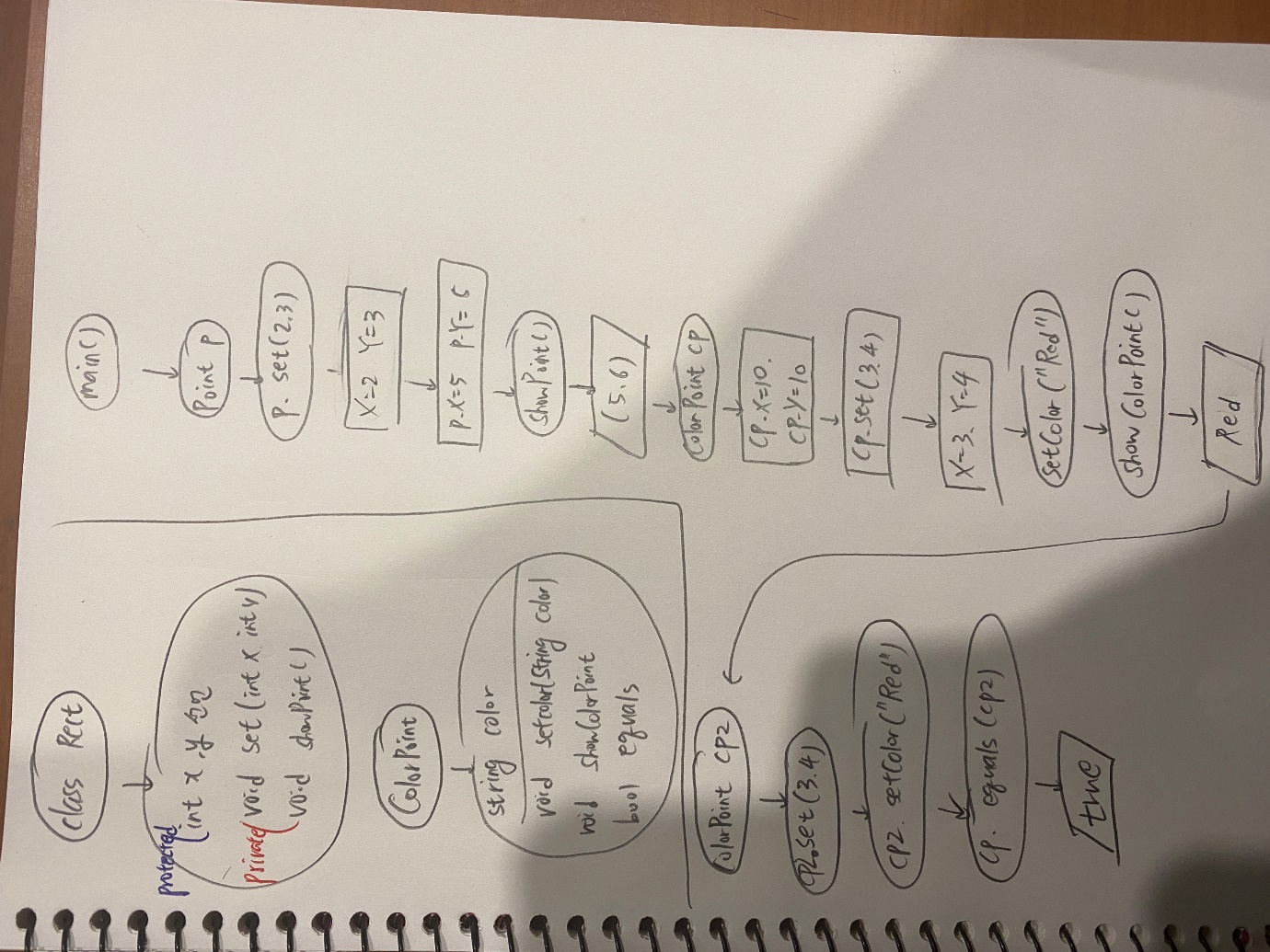
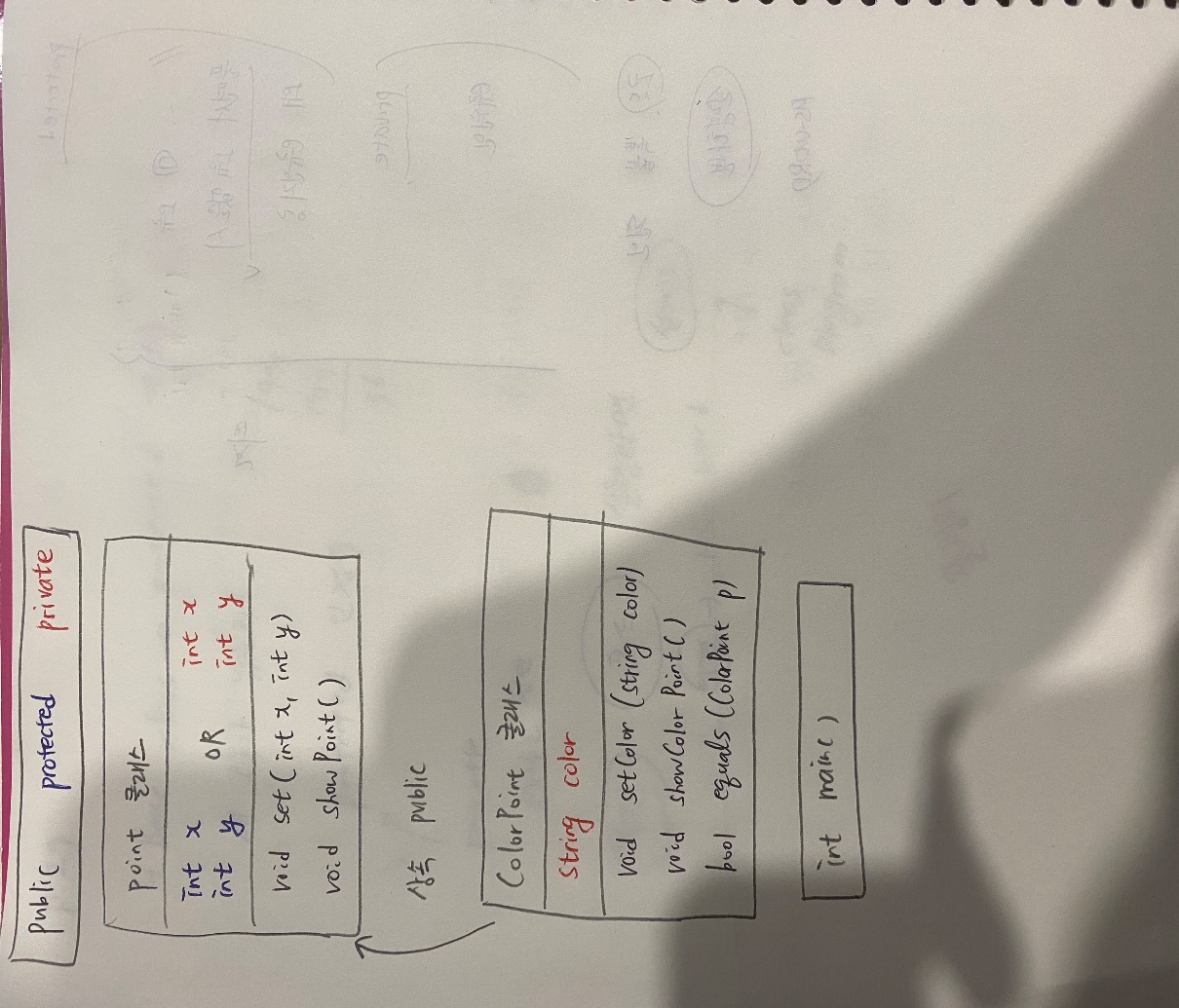
<클래스 구조도>





<소스 코드-protected 변수를 사용하였을 때>

#include<iostream>

#include<string>

using namespace std;

class Point {

protected:

int x, y;//변수 x,y는 Point클래스와 이 클래스를 상속받는 클래스에서만 사용 가능

public:

void set(int x, int y);//int x와 y를 매개변수로 이용하는 set메소드

void showPoint();

//set메소드와 showPoint메소드는 어디에서나 사용 가능

};

void Point::set(int x, int y) {

this->x = x;

this->y = y;

}

void Point::showPoint() {

cout << "(" << x << "," << y << ")" << endl;

}

class ColorPoint :public Point {//ColorPoint 클래스가 Point클래스를 상속받음. 즉 int x,y는 protected로, set과 showPoint메소드는 public의 형태로 상속받음.

string color;//private형태의 color변수

public ://public의 형태이기 때문에 어디에서나 아래의 이 세개의 메소드를 사용할 수 있음

void setColor(string color);

void showColorPoint();

bool equals(ColorPoint p);

};

void ColorPoint::setColor(string color) {

this->color = color;

}

void ColorPoint::showColorPoint() {

cout << color << ":";

showPoint();//Point 클래스의 showPoint는 public의 형태이고 Point클래스를 public의 형태로 상속받았으니 사용할 수 있음.

}

bool ColorPoint::equals(ColorPoint p) {

if (x == p.x && y == p.y && color == p.color)

//원래 protected형태의 변수나 함수는 다른 클래스에서는 사용하지 못하지만 상속관계에 있을 경우에는 그 클래스에서 사용가능하다.

return true;

else return false;

}

int main() {

Point p;

p.set(2, 3);

//set메소드는 public이므로 사용가능

***p.x = 5;***

***p.y = 5;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

p.showPoint();

//반면 showPoint메소드는 public이므로 사용가능

ColorPoint cp;

***cp.x = 10;***

***cp.y = 10;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

cp.set(3, 4);

cp.setColor("Red");

cp.showColorPoint();

//set, setColor, showColorPoint는 public 형태이므로 사용가능

ColorPoint cp2;

cp2.set(3, 4);

cp2.setColor("Red");

cout << ((cp.equals(cp2)) ? "true" : "false");

//set, setColor, equals 메소드는 public형태이므로 사용가능

}

<소스 코드-private 변수를 사용하였을 때>

#include<iostream>

#include<string>

using namespace std;

class Point {

private:

int x, y;//변수 x,y는 Point클래스와 이 클래스를 상속받는 클래스에서만 사용 가능

public:

void set(int x, int y);//int x와 y를 매개변수로 이용하는 set메소드

void showPoint();

//set메소드와 showPoint메소드는 어디에서나 사용 가능

};

void Point::set(int x, int y) {

this->x = x;

this->y = y;

}

void Point::showPoint() {

cout << "(" << x << "," << y << ")" << endl;

}

class ColorPoint :public Point {//ColorPoint 클래스가 Point클래스를 상속받음. 즉 int x,y는 protected로, set과 showPoint메소드는 public의 형태로 상속받음.

string color;//private형태의 color변수

public ://public의 형태이기 때문에 어디에서나 아래의 이 세개의 메소드를 사용할 수 있음

void setColor(string color);

void showColorPoint();

bool equals(ColorPoint p);

};

void ColorPoint::setColor(string color) {

this->color = color;

}

void ColorPoint::showColorPoint() {

cout << color << ":";

showPoint();//Point 클래스의 showPoint는 public의 형태이고 Point클래스를 public의 형태로 상속받았으니 사용할 수 있음.

}

bool ColorPoint::equals(ColorPoint p) {

***if (x == p.x && y == p.y && color == p.color)***

**//int x,y가 private 형태이기 때문에 Point클래스에서만 사용할 수 있으므로 오류 발생**

**//ColorPoint클래스가 Point클래스를 public의 방법으로 상속하고 있기 때문에 private이다!**

return true;

else return false;

}

int main() {

Point p;

p.set(2, 3);

//set메소드는 public이므로 사용가능

***p.x = 5;***

***p.y = 5;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 private의 형태이기 때문에 선언된 클래스에서만 사용가능

p.showPoint();

//반면 showPoint메소드는 public이므로 사용가능

ColorPoint cp;

***cp.x = 10;***

***cp.y = 10;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 private의 형태이기 때문에 선언된 클래스에서만 사용가능

cp.set(3, 4);

cp.setColor("Red");

cp.showColorPoint();

//set, setColor, showColorPoint는 public 형태이므로 사용가능

ColorPoint cp2;

cp2.set(3, 4);

cp2.setColor("Red");

cout << ((cp.equals(cp2)) ? "true" : "false");

//set, setColor, equals 메소드는 public형태이므로 사용가능

}

<protected형식으로 상속할 경우>

#include<iostream>

#include<string>

using namespace std;

class Point {

protected:

int x, y;//변수 x,y는 Point클래스와 이 클래스를 상속받는 클래스에서만 사용 가능

public:

void set(int x, int y);//int x와 y를 매개변수로 이용하는 set메소드

void showPoint();

//set메소드와 showPoint메소드는 어디에서나 사용 가능

};

void Point::set(int x, int y) {

this->x = x;

this->y = y;

}

void Point::showPoint() {

cout << "(" << x << "," << y << ")" << endl;

}

class ColorPoint :protected Point {//ColorPoint 클래스가 Point클래스를 상속받음. 즉 int x,y, set과 showPoint메소드는 protected의 형태로 상속받음.

string color;//private형태의 color변수

public ://public의 형태이기 때문에 어디에서나 아래의 이 세개의 메소드를 사용할 수 있음

void setColor(string color);

void showColorPoint();

bool equals(ColorPoint p);

};

void ColorPoint::setColor(string color) {

this->color = color;

}

void ColorPoint::showColorPoint() {

cout << color << ":";

showPoint();//Point 클래스의 showPoint는 public의 형태이고 Point클래스를 public의 형태로 상속받았으니 사용할 수 있음.

}

bool ColorPoint::equals(ColorPoint p) {

if (x == p.x && y == p.y && color == p.color)

//protected 형태이므로 상속관계에 있어서 사용가능

return true;

else return false;

}

int main() {

Point p;

p.set(2, 3);

//set메소드는 public이므로 사용가능

p.x = 5;

p.y = 5;

//x와 y는 protected이므로 상속관계에 있는 클래스에서만 사용가능하기 때문에 오류

p.showPoint();

//반면 showPoint메소드는 public이므로 사용가능

ColorPoint cp;

cp.x = 10;

cp.y = 10;

//x와 y는 protected이므로 상속관계에 있는 클래스에서만 사용가능하기 때문에 오류

cp.set(3, 4);

//set메소드는 public의 형태이지만 상속할때 protected로 상속하였기 때문에 public도 protected로 상속됨

cp.setColor("Red");

cp.showColorPoint();

//set, setColor, showColorPoint는 public 형태이므로 사용가능

ColorPoint cp2;

cp2.set(3, 4);

//set메소드는 public의 형태이지만 상속할때 protected로 상속하였기 때문에 public도 protected로 상속됨

cp2.setColor("Red");

cout << ((cp.equals(cp2)) ? "true" : "false");

//set, setColor, equals 메소드는 public형태이므로 사용가능

}

<private형태로 상속할 경우>

#include<iostream>

#include<string>

using namespace std;

class Point {

protected:

int x, y;//변수 x,y는 Point클래스와 이 클래스를 상속받는 클래스에서만 사용 가능

public:

void set(int x, int y);//int x와 y를 매개변수로 이용하는 set메소드

void showPoint();

//set메소드와 showPoint메소드는 어디에서나 사용 가능

};

void Point::set(int x, int y) {

this->x = x;

this->y = y;

}

void Point::showPoint() {

cout << "(" << x << "," << y << ")" << endl;

}

class ColorPoint :private Point {//ColorPoint 클래스가 Point클래스를 상속받음. 즉 int x,y, set과 showPoint메소드는 private의 형태로 상속받음.

string color;//private형태의 color변수

public ://public의 형태이기 때문에 어디에서나 아래의 이 세개의 메소드를 사용할 수 있음

void setColor(string color);

void showColorPoint();

bool equals(ColorPoint p);

};

void ColorPoint::setColor(string color) {

this->color = color;

}

void ColorPoint::showColorPoint() {

cout << color << ":";

showPoint();//Point 클래스의 showPoint는 public의 형태이므로 사용할 수 있음.

}

bool ColorPoint::equals(ColorPoint p) {

if (x == p.x && y == p.y && color == p.color)

return true;

else return false;

}

int main() {

Point p;

p.set(2, 3);

//set메소드는 public이므로 사용가능

p.x = 5;

p.y = 5;

//x와 y는 protected이므로 상속관계에 있는 클래스에서만 사용가능하기 때문에 오류

p.showPoint();

//반면 showPoint메소드는 public이므로 사용가능

ColorPoint cp;

cp.x = 10;

cp.y = 10;

//x와 y는 protected이므로 상속관계에 있는 클래스에서만 사용가능하기 때문에 오류

cp.set(3, 4);

//set메소드는 public의 형태이지만 상속할때 private로 상속하였기 때문에 public도 private로 상속됨. 따라서 선언된 클래스에서만 사용가능

cp.setColor("Red");

cp.showColorPoint();

//set, setColor, showColorPoint는 public 형태이므로 사용가능

ColorPoint cp2;

cp2.set(3, 4);

//set메소드는 public의 형태이지만 상속할때 private로 상속하였기 때문에 public도 private로 상속됨. 따라서 선언된 클래스에서만 사용가능

cp2.setColor("Red");

cout << ((cp.equals(cp2)) ? "true" : "false");

//set, setColor, equals 메소드는 public형태이므로 사용가능

}

<성공 소스코드>

#include<iostream>

#include<string>

using namespace std;

class Point {

***public:***

***int x, y;***//변수 x,y는 Point클래스와 이 클래스를 상속받는 클래스에서만 사용 가능

public:

void set(int x, int y);//int x와 y를 매개변수로 이용하는 set메소드

void showPoint();

//set메소드와 showPoint메소드는 어디에서나 사용 가능

};

void Point::set(int x, int y) {

this->x = x;

this->y = y;

}

void Point::showPoint() {

cout << "(" << x << "," << y << ")" << endl;

}

class ColorPoint :public Point {//ColorPoint 클래스가 Point클래스를 상속받음. 즉 int x,y는 protected로, set과 showPoint메소드는 public의 형태로 상속받음.

string color;//private형태의 color변수

public ://public의 형태이기 때문에 어디에서나 아래의 이 세개의 메소드를 사용할 수 있음

void setColor(string color);

void showColorPoint();

bool equals(ColorPoint p);

};

void ColorPoint::setColor(string color) {

this->color = color;

}

void ColorPoint::showColorPoint() {

cout << color << ":";

showPoint();//Point 클래스의 showPoint는 public의 형태이고 Point클래스를 public의 형태로 상속받았으니 사용할 수 있음.

}

bool ColorPoint::equals(ColorPoint p) {

if (x == p.x && y == p.y && color == p.color)

return true;

else return false;

}

int main() {

Point p;

p.set(2, 3);

//set메소드는 public이므로 사용가능

p.x = 5;

p.y = 5;

p.showPoint();

//반면 showPoint메소드는 public이므로 사용가능

ColorPoint cp;

cp.x = 10;

cp.y = 10;

cp.set(3, 4);

cp.setColor("Red");

cp.showColorPoint();

//set, setColor, showColorPoint는 public 형태이므로 사용가능

ColorPoint cp2;

cp2.set(3, 4);

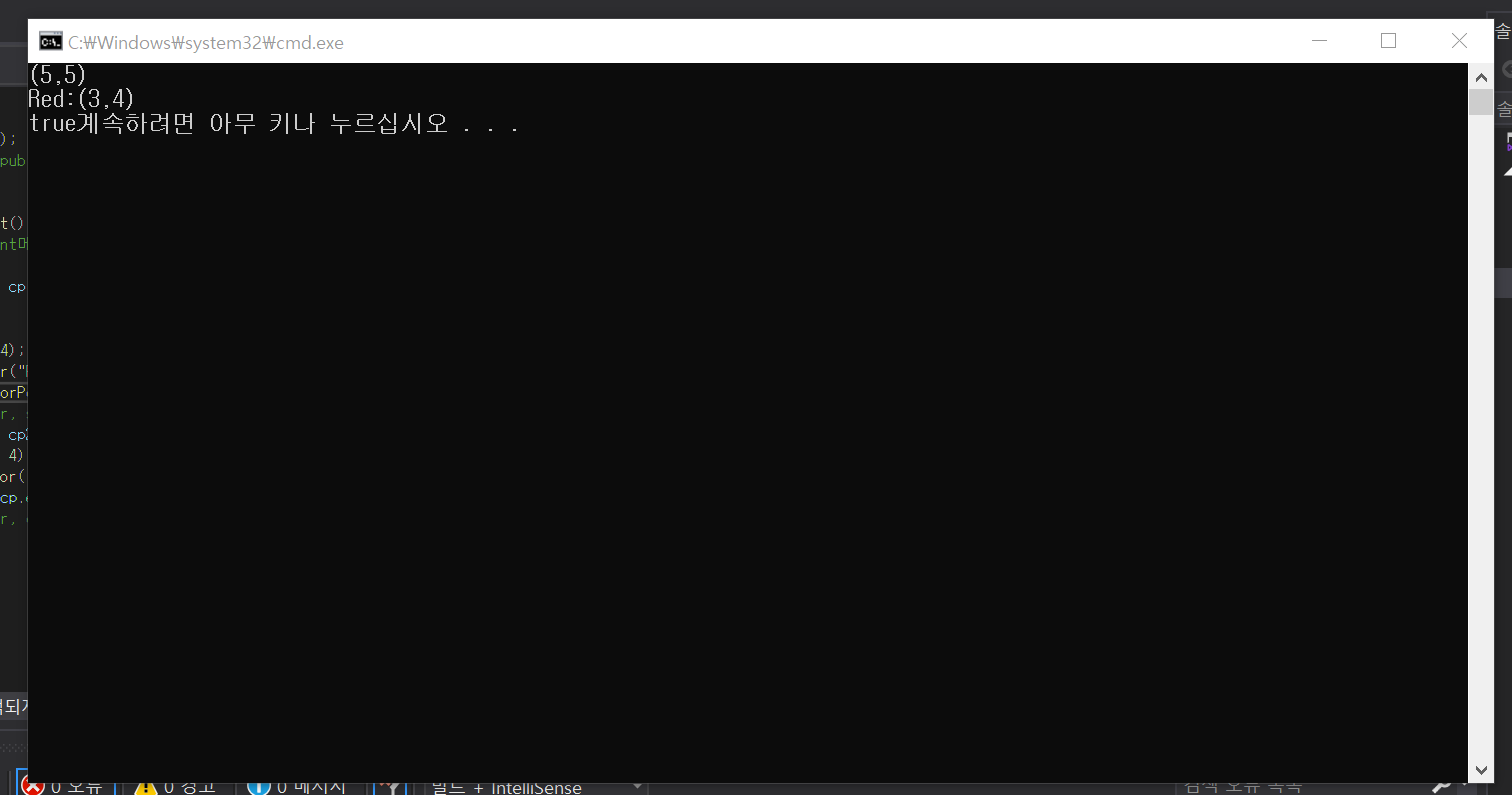
cp2.setColor("Red");

cout << ((cp.equals(cp2)) ? "true" : "false");

//set, setColor, equals 메소드는 public형태이므로 사용가능

}

<실행결과>



<의견>

Public을 public으로 상속한 경우: 문제 없음. 어디에서나 사용가능

Protected를 public으로 상속한 경우: 상속 관계에 있는 클래스에서만 사용이 가능

***p.x = 5;***

***p.y = 5;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

***cp.x = 10;***

***cp.y = 10;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

Private를 public으로 상속한 경우: 그 변수가 private로 선언된 클래스에서만 사용가능

***if (x == p.x && y == p.y && color == p.color)***

**//int x,y가 private 형태이기 때문에 Point클래스에서만 사용할 수 있으므로 오류 발생**

**//ColorPoint클래스가 Point클래스를 public의 방법으로 상속하고 있기 때문에 private이다!**

***p.x = 5;***

***p.y = 5;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

***cp.x = 10;***

***cp.y = 10;***

//ColorPoint클래스가 Point클래스를 상속할 때에 int x,y는 protected의 형태이기 때문에 상속관계에 있는 클래스에서만 사용가능하다.

//현재 main메소드는 Point클래스와 상속관계가 아니기 때문에 사용이 불가능하다.

자바에서는 상속을 하면 extends를 사용하여 상속한다고 한다. C++은 자바와 다르게 어떤 형식으로 상속을 하는지도 정할 수 있어 신기하였다.