



경기과학기술대학교 · 인공지능학과

자바 프로그래밍

상속의 개념과 protected

9주차

부모가 가진 것은 물려받고,
필요한 것만 새로 더한다.

오늘의 강의

1 교시 상속의 개념, 2 교시 상속과 객체, 3 교시 protected 접근 지정

1교시



상속의 개념

공통 코드를 부모에
extends 와 슈퍼·서브

2교시



상속과 객체

자식 객체 안의 멤버
private 은 그대로

3교시



protected 접근 지정

자식에게 여는 protected
표 5-1 과 연습문제

1교시

상속의 개념

같은 코드를 부모 클래스 하나에 모으고 자식이 물려받게 한다.

상속

extends

슈퍼 · 서브 클래스

지난주 protected 칸에 남겨 둔 「자식 클래스」

	protected
같은 패키지의 클래스	O
다른 패키지의 클래스	X
접근 가능 영역	동일 패키지와 <u>자식 클래스</u>

지난주에는 「자식 클래스」를 뒤로 미뤘다.

오늘은 클래스가 클래스를 물려받는 상속을 배운다

클래스가 다른 클래스의 필드와 메소드를
물려받는 것을 상속이라고 한다.

1교시 상속의 개념

2교시 상속과 객체

3교시 protected

사람의 상속은 유산, 객체 지향의 상속은 유전에 가깝다

객체 지향의 상속

유산 상속

부모가 가진 재산을 자식에게 넘긴다

넘기면 부모에게서 없어진다

유전적 상속

부모의 특징을 자식이 태어날 때 갖고 나 온다

부모도 그대로 갖고 있다

객체 지향의 상속은 부모 유전자를 물려받는 것과 비슷하다

객체 지향 언어에서의 상속은 부모 유전자를 자식이 물려받는 유전적 상속과 유사하다.

부모 클래스: 유전자를 준다



자식 클래스: 태어날 때 갖고 나온다

왜 클래스에게 물려받게 만들까

필드와 메소드를 클래스마다 새로 쓰면 안 될까.

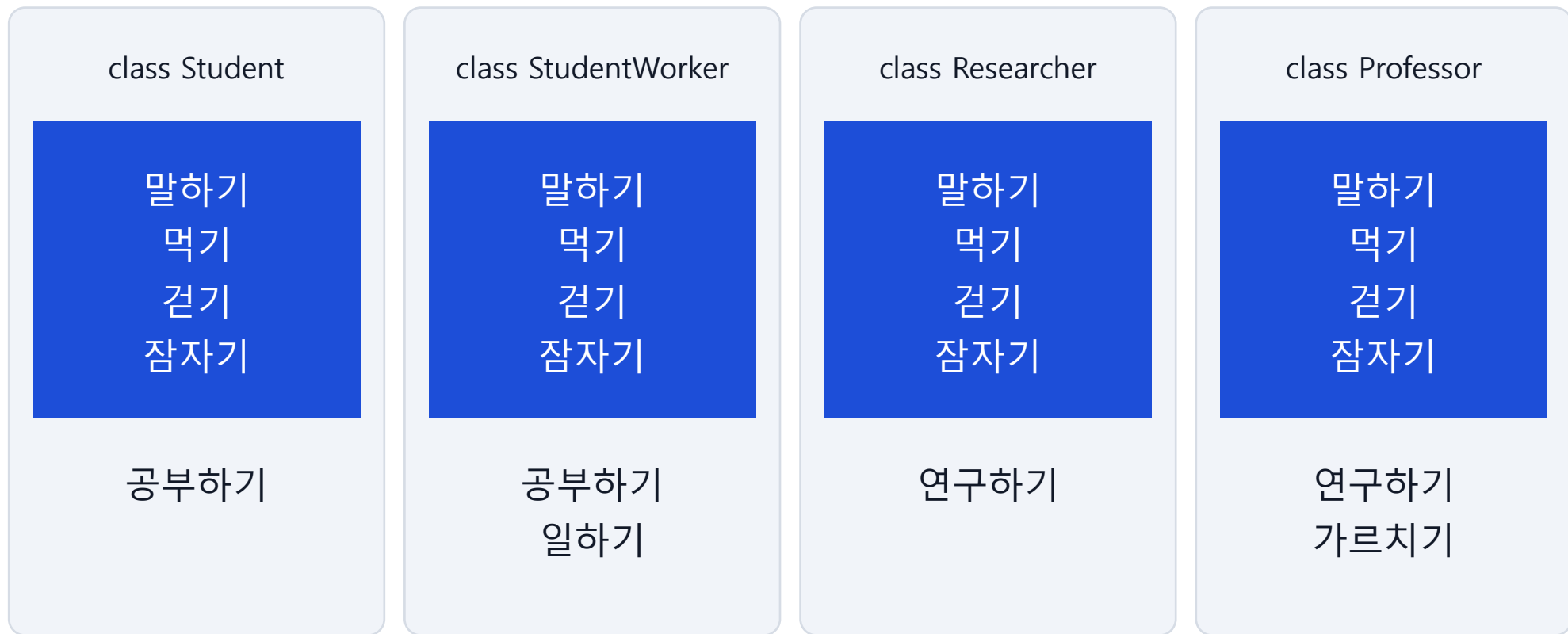


① 클래스마다 새로 쓴다



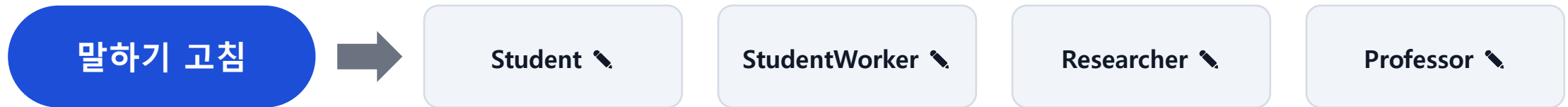
② 한 곳에 쓰고 물려받는다

네 클래스에 같은 기능이 네 번씩 들어 있다



네 번
반복

같은 코드가 여러 클래스에 있으면 고치기가 번거롭다



이처럼 동일한 코드가 여러 클래스에 중복되면 클래스의 유지 보수가 여간 번거롭지 않다.

네 곳 가운데 한 곳을 빠뜨리기 쉽다

클래스	말하기 고침
Student	✓
StudentWorker	✓
Researcher	× 빠뜨림
Professor	✓

여러 곳을 같이 고쳐야 하는 일은 꼭 한 곳이 남는다.

먹기와 잠자기가 두 번씩 있는 Learner 와 Teacher

Ex01_DupClasses.java

```
class Learner {  
    String name;  
  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
  
    void study() { System.out.println(name + " 공부한다"); }  
}  
class Teacher {  
    String name;  
  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
  
    void teach() { System.out.println(name + " 가르친다"); }  
}
```

학생과 교수 객체를 하나씩 만들어 부르기

Ex01_DupClasses.java

```
public class Ex01_DupClasses {  
    public static void main(String[] args) {  
        Learner s = new Learner();  
        s.name = "김자바";  
        s.eat();  
        s.study();  
        Teacher t = new Teacher();  
        t.name = "이교수";  
        t.sleep();  
        t.teach();  
    }  
}
```



김자바는 먹고 공부하고, 이교수는 자고 가르친다

김자바 밥을 먹는다

김자바 공부한다

이교수 잠을 잔다

이교수 가르친다

두 클래스가 같은 메소드를 따로 갖고 있다

eat() 과 sleep() 은 두 클래스에 똑같이 적혀 있다

Learner

```
void eat()  
void sleep()
```

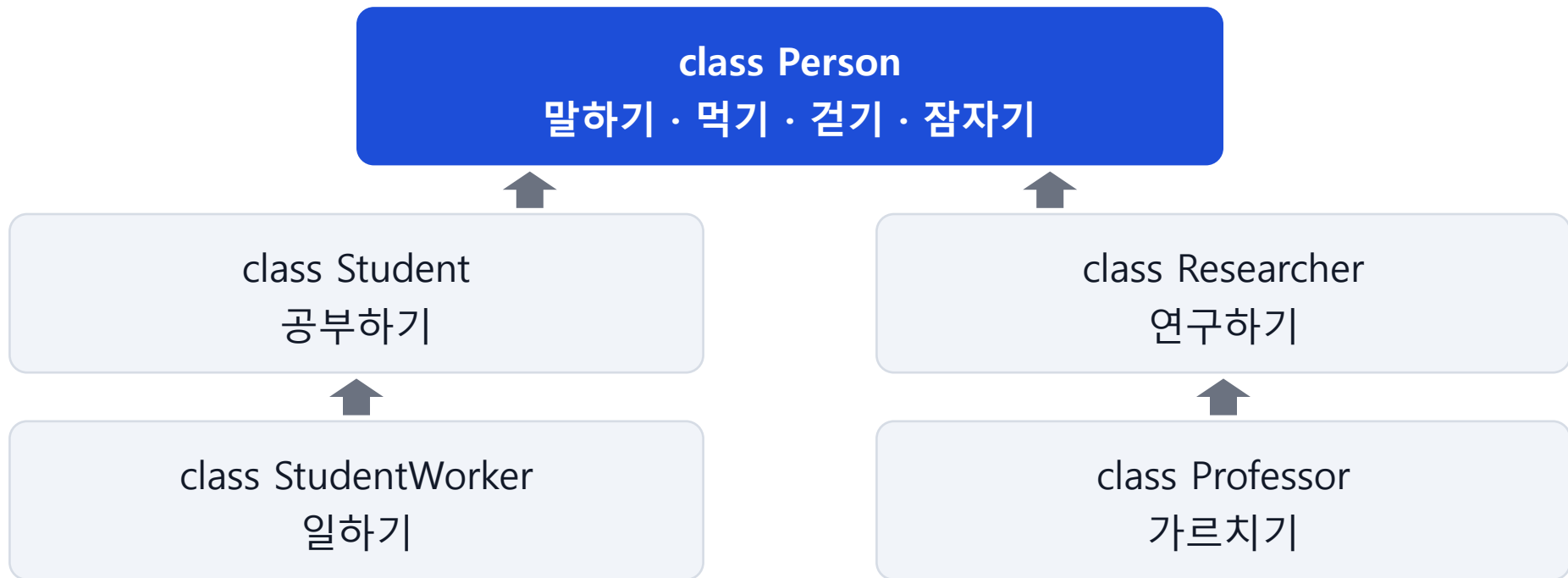
글자까지 같다

Teacher

```
void eat()  
void sleep()
```

잠자기 문구를 바꾸려면 두 곳을 고친다.

공통 기능을 Person 으로 추리고 나머지가 물려받는다



상속은 클래스 사이의 코드 중복을 없앤다

상속은 클래스 사이에 코드 중복을 제거하여 클래스를 간결하게 구현할 수 있게 한다.

말하기 고침 → Person 한 곳

걷기 개선 → Person 한 곳

상속의 장점은 세 가지다

①

클래스 사이의 멤버 중복 선언 불필요 - 클래스의 간결화

②

클래스들의 계층적 분류로 클래스 관리 용이

③

클래스 재사용과 확장을 통한 소프트웨어의 생산성 향상

extends

자바에서 상속을 선언하는 키워드

자바에서는 extends 키워드를 사용하여 상속을 선언한다.

```
class 자식클래스 extends 부모클래스 { ... }
```

슈퍼 클래스와 서브 클래스

상속하는 부모와 상속받는 자식

자바에서는 상속하는 부모 클래스를 슈퍼 클래스(super class), 상속받는 자식 클래스를 서브 클래스(sub class)라고 부른다.



ColorPoint 는 Point 를 물려받아 색을 더한다

```
class Point {  
    int x, y;  
}  
class ColorPoint extends Point { // Point를 상속받는 ColorPoint 클래스 선언  
    String color;  
}
```

슈퍼 클래스

서브 클래스

ColorPoint 에는 color 만 적혀 있지만 x 와 y 도 갖는다.

물려받은 필드와 메소드는 다시 선언하지 않는다

ColorPoint 클래스는 Point 클래스를 물려받으므로 Point 클래스에 선언된 필드나 메소드를 다시 선언할 필요가 없다.

물려받은 것: x, y

새로 더한 것: color

eat() 을 Person 으로 올리고 extends 를 붙인 코드

```
class Person {  
    String name;  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
    // 여기서 부터 코드를 작성하세요.  
  
    // 여기 까지  
}  
class Student extends Person {  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
    void study() { System.out.println(name + " 공부한다"); }  
}  
class Professor extends Person {  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
    void teach() { System.out.println(name + " 가르친다"); }  
}
```

Ex02_PersonInherit.java

한 곳으로 올림

아직 두 번

아직 두 번

main 에서 Student 객체로 eat() 부르기

Ex02_PersonInherit.java

```
public class Ex02_PersonInherit {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.name = "김자바";  
        s.eat();  
        s.study();  
        Professor p = new Professor();  
        p.name = "이교수";  
        p.sleep();  
        p.teach();  
    }  
}
```

Student 에는 없는 메소드



Student 에 eat() 이 없어도 김자바는 밥을 먹는다

김자바 밥을 먹는다

김자바 공부한다

이교수 잠을 잔다

이교수 가르친다

eat() 은 Person 에게서 물려받았다

직접 실습 ①: sleep() 도 Person 으로 올리기



① Person 빈칸에 sleep() 을 한 줄로 쓴다



② Student 와 Professor 의 sleep() 줄을 지운다

결과 네 줄은 바뀌지 않아야 한다.

Person 빈칸에 sleep() 을 쓰고 두 자식의 sleep() 을 지워 보기

```
class Person {  
    String name;  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
    // 여기서 부터 코드를 작성하세요.  
  
    // 여기 까지  
}  
class Student extends Person {  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
    void study() { System.out.println(name + " 공부한다"); }  
}  
class Professor extends Person {  
    void sleep() { System.out.println(name + " 잠을 잔다"); }  
    void teach() { System.out.println(name + " 가르친다"); }  
}
```

main 은 그대로 두고 실행하기

Ex02_PersonInherit.java

```
public class Ex02_PersonInherit {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.name = "김자바";  
        s.eat();  
        s.study();  
        Professor p = new Professor();  
        p.name = "이교수";  
        p.sleep();  
        p.teach();  
    }  
}
```

main 은 손대지 않는다



결과 네 줄은 그대로다

김자바 밥을 먹는다

김자바 공부한다

이교수 잠을 잔다

이교수 가르친다

코드는 줄었고 결과는 같다

고칠 곳이 두 곳에서 한 곳으로 줄었다

상속 없이(Ex01)

eat() · sleep() 이 두 번씩

고칠 곳 두 곳



상속으로(Ex02)

eat() · sleep() 은 Person 에 한 번

고칠 곳 한 곳

클래스가 늘수록 차이가 커진다.

문제: 휴대폰과 음악폰을 상속으로 간결하게

상속을 이용하여 다음 클래스들을 [그림 5-3]과 같이 간결하게 구성하라.

class MobilePhone: 전화 걸기 · 전화 받기 · 무선 기지국 연결 · 배터리 충전하기

class MusicPhone: 전화 걸기 · 전화 받기 · 무선 기지국 연결 · 배터리 충전하기 · 음악 다운로드받기 · 음악 재생하기



MobilePhone 을 물려받은 MusicPhone

Ex03_PhoneInherit.java

```
class MobilePhone {  
    void call() { System.out.println("전화를 건다"); }  
    void receive() { System.out.println("전화를 받는다"); }  
    void charge() { System.out.println("배터리를 충전한다"); }  
}  
class MusicPhone extends MobilePhone {  
    void download() { System.out.println("음악을 내려받는다"); }  
    void play() { System.out.println("음악을 재생한다"); }  
}
```

MusicPhone 객체로 물려받은 call() 과 charge() 를 부른다.

물려받은 call() 과 charge() 부르기

Ex03_PhoneInherit.java

할 일: 빈칸에 두 줄을 써서 call() 과 charge() 부르기

```
public class Ex03_PhoneInherit {  
    public static void main(String[] args) {  
        MusicPhone mp = new MusicPhone();  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        mp.download();  
        mp.play();  
    }  
}
```



전화를 걸고 충전하고 음악을 내려받아 재생한다

전화를 건다

배터리를 충전한다

음악을 내려받는다

음악을 재생한다

MusicPhone 객체로 MobilePhone 의 메소드를 불렀다



빈칸 그대로 실행하면

음악을 내려받는다

음악을 재생한다

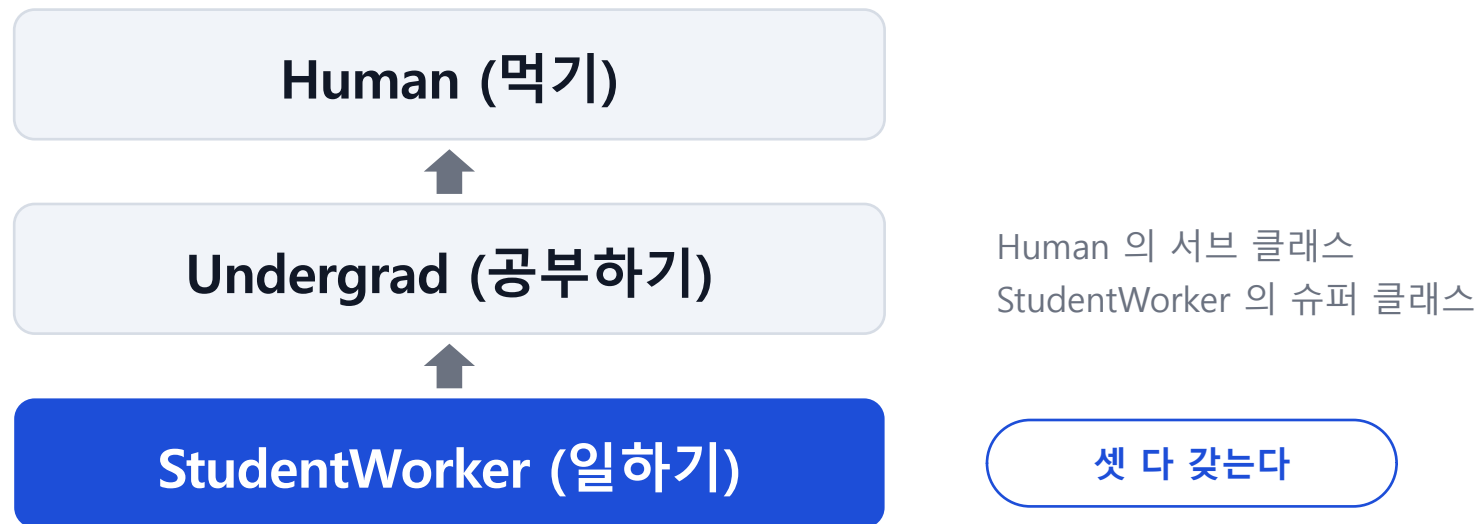
MusicPhone 자기 메소드 둘만 불렀다

MusicPhone 에 적지 않은 call() 도 부를 수 있다

메소드	적힌 클래스	mp 로 부를 수 있나
call() · receive() · charge()	MobilePhone	된다
download() · play()	MusicPhone	된다

서브 클래스 객체는 슈퍼 클래스의 메소드도 갖고 있다.

상속은 여러 층으로 이어진다



서브 클래스도 다시 슈퍼 클래스가 될 수 있다.

Human, Undergrad, StudentWorker 세 층 클래스

Ex04_ThreeLevels.java

```
class Human {  
    String name;  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
}  
class Undergrad extends Human {  
    void study() { System.out.println(name + " 공부한다"); }  
}  
class StudentWorker extends Undergrad {  
    void work() { System.out.println(name + " 일한다"); }  
}
```

extends Human

extends Undergrad

main 에서 sw 로 work() 하나만 부르기

Ex04_ThreeLevels.java

```
public class Ex04_ThreeLevels {  
    public static void main(String[] args) {  
        StudentWorker sw = new StudentWorker();  
        sw.name = "박알바";  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        sw.work();  
    }  
}
```



빈칸 그대로 실행하면 박알바는 일만 한다

박알바 일한다

StudentWorker 자기 메소드 하나만 불렀다

직접 실습 ③:

맨 아래 객체로 위 두 층의 메소드 부르기

할 일

sw 로 Human 의 eat() 과 Undergrad 의 study() 를 부른다.

eat() ← Human

study() ← Undergrad

work() ← StudentWorker

세 층 클래스는 그대로 둔다

Ex04_ThreeLevels.java

```
class Human {  
    String name;  
    void eat() { System.out.println(name + " 밥을 먹는다"); }  
}  
class Undergrad extends Human {  
    void study() { System.out.println(name + " 공부한다"); }  
}  
class StudentWorker extends Undergrad {  
    void work() { System.out.println(name + " 일한다"); }  
}
```

클래스 세 개는 손대지 않는다. 메소드가 있는 층만 확인한다.

sw 로 eat() 과 study() 를 불러 보기

Ex04_ThreeLevels.java

빈칸에 두 줄을 쓴다

```
public class Ex04_ThreeLevels {  
    public static void main(String[] args) {  
        StudentWorker sw = new StudentWorker();  
        sw.name = "박알바";  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        sw.work();  
    }  
}
```



박알바는 먹고 공부하고 일한다

박알바 밥을 먹는다

박알바 공부한다

박알바 일한다

맨 아래 객체가 세 층의 메소드를 모두 불렀다

손자 클래스도 맨 위 클래스의 멤버를 갖는다

클래스

갖고 있는 메소드

Human

eat()

Undergrad

eat()

study()

StudentWorker

eat()

study()

work()

아래로 갈수록 갖는 것이 늘어난다.

상속은 클래스 사이의 관계, 멤버를 가진 것은 객체

클래스



설계도

extends 로 설계도끼리 이어진다



객체



new 로 만든 실체

물려받은 멤버까지 모두 들고 생긴다

상속은 설계도에 적고, 쓰는 것은 객체다.

부모 객체로 자식의 메소드를 부르면 어떻게 될까

Animal 객체 a 로 Bird 의 fly() 를 부르면?

① 하늘을 난다

② 오류

Animal 객체 a 로 fly() 부르기

Ex05_ParentCannot.java

```
class Animal {  
    void eat() { System.out.println("먹이를 먹는다"); }  
}  
class Bird extends Animal {  
    void fly() { System.out.println("하늘을 난다"); }  
}  
  
public class Ex05_ParentCannot {  
    public static void main(String[] args) {  
        Animal a = new Animal();  
        a.eat();  
        a.fly();  
    }  
}
```

Animal 에는 없다

컴파일 오류 원문

Ex05_ParentCannot.java:12: error: cannot find symbol

```
a.fly();
```

^

symbol: method fly()

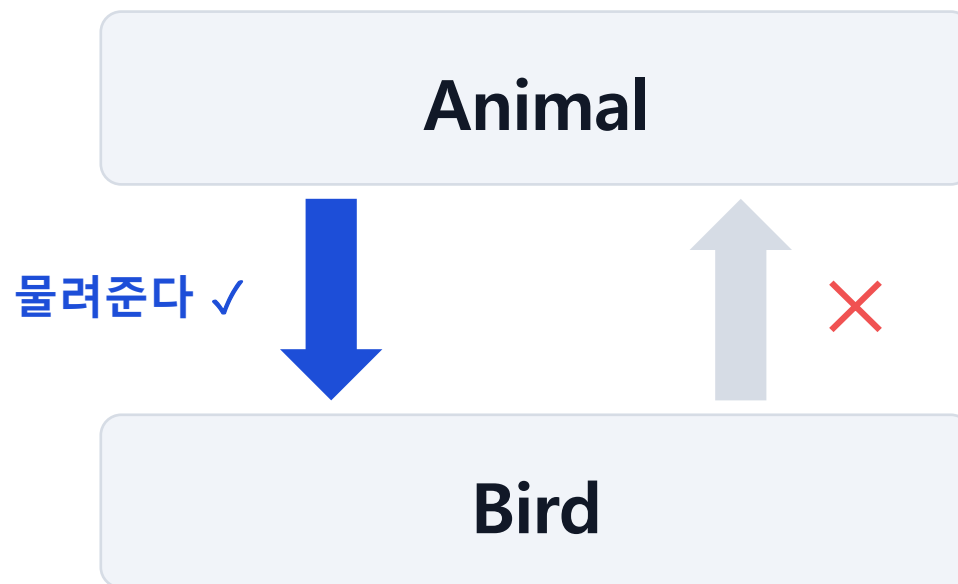
location: variable a of type Animal

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

Animal 에는 fly() 가 없다

물려받는 방향은 아래로만이다



자식은 부모의 것을 갖지만 부모는 자식의 것을 갖지 않는다.

1교시에서 본 것

1

상속은 클래스가 다른 클래스의 필드와 메소드를 물려받는 것이다

2

공통 코드를 슈퍼 클래스에 두면 고칠 곳이 한 곳이 된다

3

class 자식 extends 부모 로 선언한다

4

물려받는 방향은 아래로만이다

1교시 실습 살펴보기

실습 ①

sleep() 을 Person 으로 올려도
결과 네 줄은 그대로였다.

**상속은 결과가 아니라
코드 모양을 바꾼다.**

실습 ③

StudentWorker 에 적지 않은
eat() 과 study() 도 sw 로 불렀다.

2교시

상속과 객체

자식 객체 안에 무엇이 들어 있고 그중 무엇을 쓸 수 있는지 본다.

예제 5-1

서브 클래스 객체

다중 상속 · Object

문제: 점 클래스와 색 있는 점 클래스를 만들고 쓰기

예제 5-1

(x, y)의 한 점을 표현하는 Point 클래스와 이를 상속받아 점에 색을 추가한 ColorPoint 클래스를 만들고 활용해보자.

좌표를 감춘 Point 클래스

Ex06_ColorPointEx.java

```
class Point {  
    private int x, y; // 한 점을 구성하는 x, y 좌표  
    void set(int x, int y) {  
        this.x = x; this.y = y;  
    }  
    void showPoint() { // 점의 좌표 출력  
        System.out.println("(" + x + "," + y + ")");  
    }  
}
```

Point 를 물려받아 색을 더한 ColorPoint

Ex06_ColorPointEx.java

```
class ColorPoint extends Point { // Point를 상속받은 ColorPoint 선언
    private String color; // 점의 색
    void setColor(String color) {
        this.color = color;
    }
    void showColorPoint() { // 컬러 점의 좌표 출력
        System.out.print(color);
        showPoint(); // Point 클래스의 showPoint() 호출
    }
}
```

Point 의 메소드

Point 객체와 ColorPoint 객체를 하나씩 만들어 쓰기

Ex06_ColorPointEx.java

```
public class Ex06_ColorPointEx {  
    public static void main(String[] args) {  
        Point p = new Point(); // Point 객체 생성  
        p.set(1, 2); // Point 클래스의 set() 호출  
        p.showPoint();  
  
        ColorPoint cp = new ColorPoint(); // ColorPoint 객체 생성  
        cp.set(3, 4); // Point 클래스의 set() 호출  
        cp.setColor("red"); // ColorPoint 클래스의 setColor() 호출  
        cp.showColorPoint(); // 컬러와 좌표 출력  
    }  
}
```

Point 의 set()

(1,2) 와 red(3,4)

(1,2)

red(3,4)

cp 는 물려받은 set() 으로 좌표를 넣었다

cp.set(3, 4) 는 Point 에게서 물려받은 set() 이다

```
cp.set(3, 4); // Point 클래스의 set() 호출
```



```
Point
```

```
void set(int x, int y)
```

서브 클래스 객체로 슈퍼 클래스의 메소드를 부른다.

p 와 cp 는 서로 다른 객체다

이 둘은 별개의 객체이다.

객체 p는 Point 클래스의 멤버만 가지고, cp는 Point와 ColorPoint의 멤버를 모두 가진다.

p

int x 1 · int y 2
set() · showPoint()

Point
클래스 멤버

cp

int x 3 · int y 4
set() · showPoint()

Point
클래스 멤버

color "red"
setColor()
showColorPoint()

ColorPoint
클래스 멤버

cp 안에는 Point 멤버 칸과 ColorPoint 멤버 칸이 함께 있다

cp 객체

객체는 하나

Point 클래스 멤버

x, y, set(), showPoint()

ColorPoint 클래스 멤버

color, setColor(), showColorPoint()

상속은 두 클래스의 멤버를 모두 갖고 탄생하게 하는 것이다

상속은 바로 cp 객체처럼 슈퍼 클래스의 멤버와 서브 클래스의 멤버를 모두 갖고 탄생하게 하는 것이다.

바깥에서는 두 클래스의 public 멤버를 모두 부른다

외부에서 ColorPoint 클래스의 public 멤버와 슈퍼 클래스 Point의 public 멤버를 모두 접근할 수 있다.
그러나 x, y, color는 private 속성이므로 외부에서 접근할 수 없다.

부를 수 있다: `set()` · `showPoint()` · `setColor()` · `showColorPoint()`

못 쓴다: `x` · `y` · `color`

private x 를 가진 Point7 과 그 자식 ColorPoint7

Ex07_OutsidePrivate.java

```
class Point7 {  
    private int x, y;  
    void set(int x, int y) { this.x = x; this.y = y; }  
    void showPoint() { System.out.println("(" + x + "," + y + ")"); }  
}  
class ColorPoint7 extends Point7 {  
    private String color;  
    void setColor(String color) { this.color = color; }  
}
```

main 에서 cp.x 에 바로 값 넣기

Ex07_OutsidePrivate.java

```
11 public class Ex07_OutsidePrivate {  
12     public static void main(String[] args) {  
13         ColorPoint7 cp = new ColorPoint7();  
14         cp.set(3, 4);  
15         cp.x = 10;  
16         cp.showPoint();  
17     }  
18 }
```

private

컴파일 오류 원문

Ex07_OutsidePrivate.java:15: error: x has private access in Point7

```
cp.x = 10;
```

^

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

물려받아도 private 은 private 이다

지난주와 같은 규칙이다.

private 은 클래스 안에서만 쓴다

private 멤버는 그 클래스 안에서만 쓴다(7주차).



`cp.x = 10;`

바깥에서 `x` 에 바로 값을 넣는다



`cp.set(10, 4);`

`Point` 의 `set()` 을 거친다

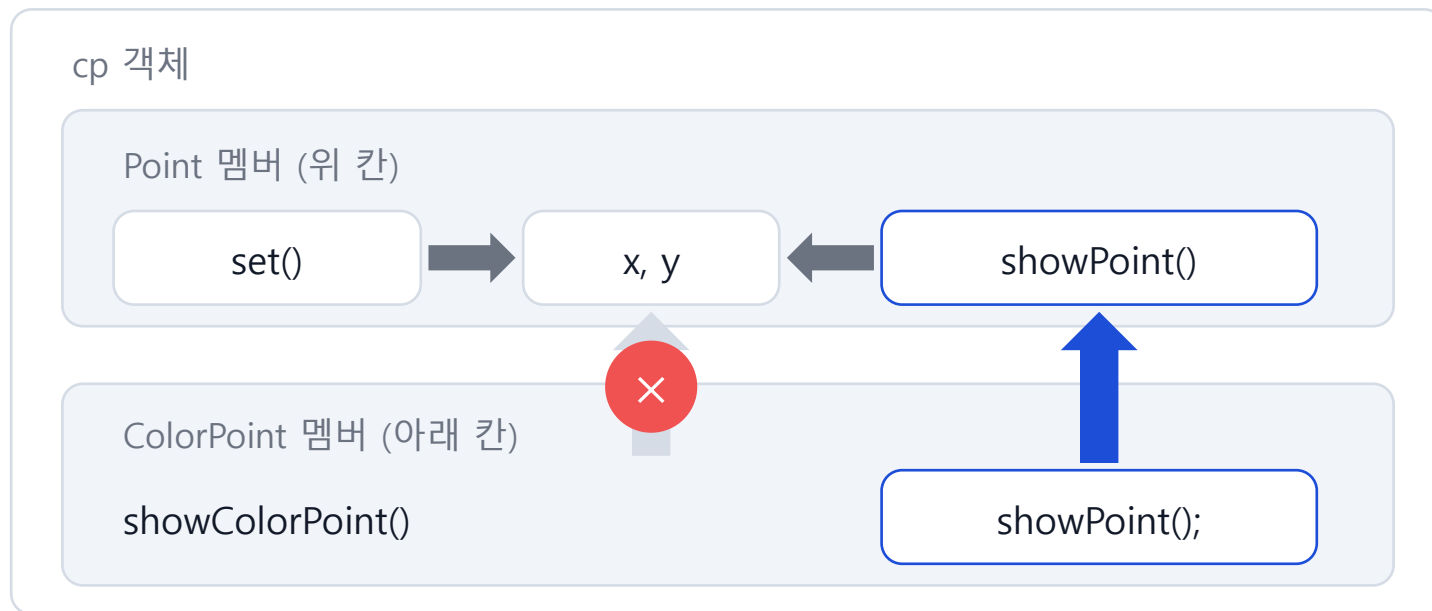
서브 클래스는 슈퍼 클래스의 private 멤버만 빼고 모두 쓴다

서브 클래스는 슈퍼 클래스의 private 멤버 외 모든 멤버를 접근할 수 있다.

쓸 수 있다: private 이 아닌 멤버

못 쓴다: private 멤버

showColorPoint() 가 슈퍼 클래스의 showPoint() 를 부른다



showColorPoint() 는
위 칸 showPoint() 를
부른다

아래 칸에서 x, y 로
가는 길은 막혀 있다

x, y 는 set() 과 showPoint() 만 만질 수 있다

Point

private x, y

set()

showPoint()

ColorPoint

x, y 접근 불가

x, y는 Point의 private 멤버이므로, ColorPoint 클래스의 멤버들이 접근할 수 없다.

x, y는 오직 set()과 showPoint()만 접근이 허용된다.

자식 메소드가 부모의 x 를 바로 찍으면 어떻게 될까

showColorPoint() 안에서 showPoint() 대신 x, y 를 바로 찍으면?

```
System.out.println(color + "(" + x + "," + y + ")");
```

ColorPoint8 이 Point8 의 x, y 를 바로 찍는 코드

```
class Point8 {  
    private int x, y;  
    void set(int x, int y) { this.x = x; this.y = y; }  
}  
class ColorPoint8 extends Point8 {  
    private String color;  
    void setColor(String color) { this.color = color; }  
    void showColorPoint() {  
        System.out.println(color + "(" + x + "," + y + ")");  
    }  
}
```

부모의 private

색 있는 점 하나를 만들어 출력하기

Ex08_SubPrivate.java

```
public class Ex08_SubPrivate {  
    public static void main(String[] args) {  
        ColorPoint8 cp = new ColorPoint8();  
        cp.set(3, 4);  
        cp.setColor("red");  
        cp.showColorPoint();  
    }  
}
```

컴파일 오류 원문

Ex08_SubPrivate.java:9: error: x has private access in Point8

```
System.out.println(color + "(" + x + "," + y + ")");  
                                ^
```

Ex08_SubPrivate.java:9: error: y has private access in Point8

```
System.out.println(color + "(" + x + "," + y + ")");  
                                ^
```

2 errors

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

자식 클래스 안에서도 부모의 private 은 못 쓴다

부모의 showPoint() 를 부르면 된다



```
System.out.println(color + "(" + x + "," + y + ")");
```



```
System.out.print(color);  
showPoint();
```

private 필드는 그 클래스의 메소드를 거친다.

Point9 와 그 자식 ColorPoint9

Ex09_TwoObjects.java

결과를 먼저 예측한 뒤 실행해 비교한다.

```
class Point9 {
    private int x, y;
    void set(int x, int y) { this.x = x; this.y = y; }
    void showPoint() { System.out.println("(" + x + "," + y + ")"); }
}
class ColorPoint9 extends Point9 {
    private String color;
    void setColor(String color) { this.color = color; }
}
```

p 와 cp 에 set() 을 하나씩 부르기

Ex09_TwoObjects.java

```
public class Ex09_TwoObjects {  
    public static void main(String[] args) {  
        Point9 p = new Point9();  
        ColorPoint9 cp = new ColorPoint9();  
        p.set(1, 2);  
        cp.set(3, 4);  
        p.showPoint();  
        cp.showPoint();  
    }  
}
```

두 줄은 무엇을 찍을까

실행할 줄

p.showPoint();

cp.showPoint();

찍힐 값

()

()

오른쪽 칸은 실행 뒤에 채운다.

(1,2) 와 (3,4)

(1,2)

(3,4)

객체마다 x , y 가 따로 있다

같은 `set()` 을 불러도 같은 객체마다 따로 들어간다

p 객체

x 1 · y 2

`set()` 은 같은 코드,
칸은 따로

cp 객체

x 3 · y 4

color null

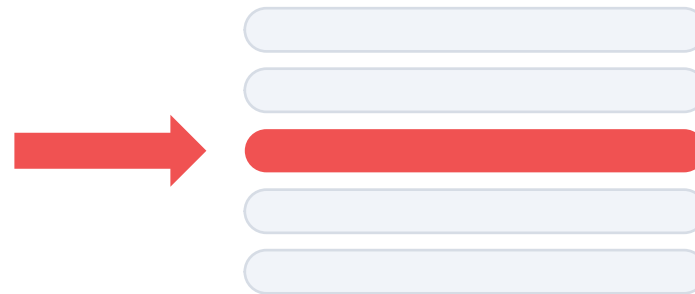


스택인데 가운데에 끼워 넣을 수 있다

자바 1.0(1996)부터 있던 `java.util.Stack` 은 `class Stack extends Vector` 로 선언되어 `Vector` 의 메소드를 모두 물려받았다.

그래서 맨 위에서만 넣고 빼야 할 스택인데도 `Vector` 에게서 물려받은 `insertElementAt()` 으로 가운데에 끼워 넣을 수 있다.

스택: 접시 쌓기처럼
맨 위에서만 넣고 뺀다



`insertElementAt()`

자바 공식 문서는 이렇게 적는다: "A more complete and consistent set of LIFO stack operations is provided by the `Deque` interface and its implementations, which should be used in preference to this class."

서브 클래스는 물려받을 멤버를 고르지 못한다



extends 하면 슈퍼 클래스의 멤버를
모두 갖고 태어난다



필요 없는 메소드만 빼고 받을 수는
없다

그래서 extends 앞에서

「정말 부모와 같은 종류인가」를 먼저 묻는다.

직접 실습 ④: Dog 가 Pet 의 이름을 찍게 하기

할 일

Dog 의 showDog() 빈칸에서 Pet 의 메소드를 불러 이름을 찍는다.



name 은 Pet 의 private



showName() 은 Pet 의 메소드

showDog() 빈칸에서 부모의 메소드 부르기

```
class Pet {  
    private String name;  
    void setName(String name) { this.name = name; }  
    void showName() { System.out.print("이름: " + name); }  
}  
  
class Dog extends Pet {  
    private String breed;  
    void setBreed(String breed) { this.breed = breed; }  
    void showDog() {  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        System.out.println(", 품종: " + breed);  
    }  
}
```

초코라는 푸들 하나를 만들어 출력하기

Ex10_DogPet.java

```
public class Ex10_DogPet {  
    public static void main(String[] args) {  
        Dog d = new Dog();  
        d.setName("초코");  
        d.setBreed("푸들");  
        d.showDog();  
    }  
}
```

main 은 고치지 않고 그대로 실행한다.



이름: 초코, 품종: 푸들

이름: 초코, 품종: 푸들

이름은 Pet 의 `showName()` 이 찍었다



빈칸 그대로 실행하면

, 품종: 푸들

이름이 빠지고 품종만 남는다

name 대신 showName() 을 거친 까닭

System.out.print("이름: " + name);



has private access

showName();

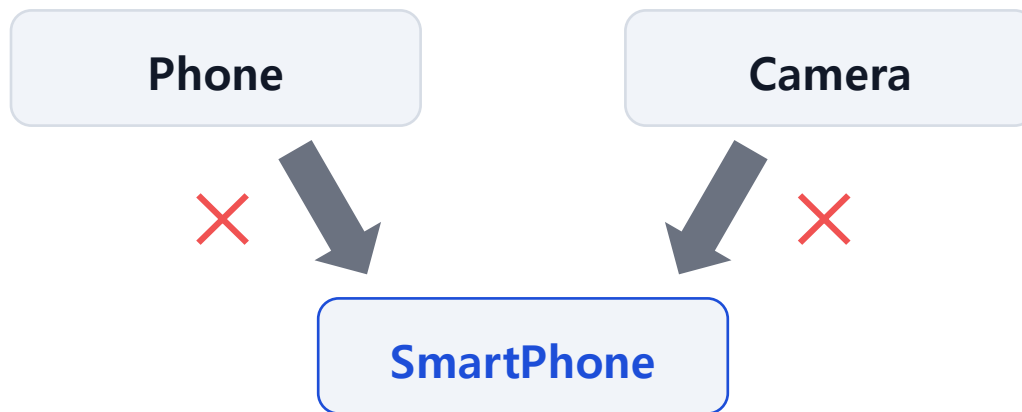


된다

서브 클래스는 슈퍼 클래스의 private 을 메소드로 거쳐 쓴다.

자바는 클래스를 여러 개 물려받는 다중 상속을 지원하지 않는다

자바는 C++와 달리 클래스를 여러 개 상속받는 다중 상속(multiple inheritance)을 지원하지 않는다.



extends 뒤에 부모 클래스를 둘 쓰면

교수 시연

Ex11_TwoParents.java

```
class Phone {  
    void call() { System.out.println("전화를 건다"); }  
}  
class Camera {  
    void shoot() { System.out.println("사진을 찍는다"); }  
}  
class SmartPhone extends Phone, Camera {  
}
```

```
public class Ex11_TwoParents {  
    public static void main(String[] args) {  
        SmartPhone sp = new SmartPhone();  
        sp.call();  
    }  
}
```

둘째 부모

컴파일 오류 원문

```
Ex11_TwoParents.java:7: error: '{' expected
class SmartPhone extends Phone, Camera {
                        ^
```

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

쉼표 자리에서 막힌다

부모 클래스는 하나, 여러 것을 물려받는 길은 인터페이스

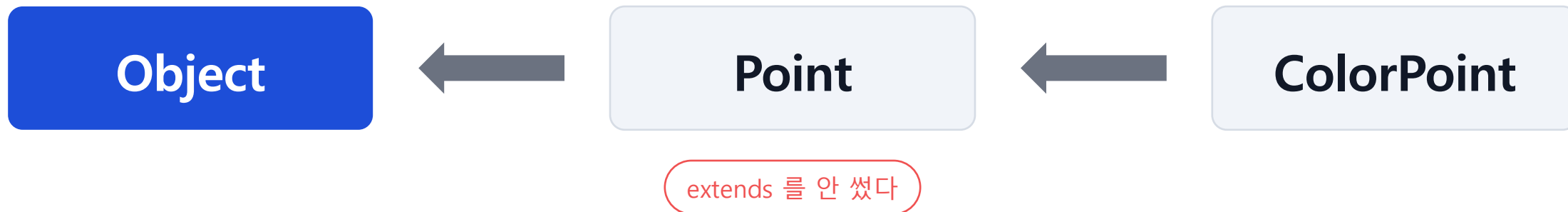
extends 뒤에 오는 클래스
는 하나다

여러 개를 물려받는 길은 인
터페이스(12주차)

```
class SmartPhone extends Phone {
```

extends 를 쓰지 않아도 모든 클래스는 Object 를 물려받는다

자바의 모든 클래스는 자바에서 제공하는 Object 클래스를 자동으로 상속받도록 컴파일된다.



Object 의 경로명은 java.lang.Object, 슈퍼 클래스가 없는 유일한 클래스다.

클래스는 설계도끼리 물려받고, 객체는 그 멤버를 모두 갖고 생긴다

클래스

- extends 로 부모 하나를 정한다
- 맨 위에는 Object

객체

- 물려받은 멤버까지 모두 갖고 생긴다
- private 은 갖고 있어도 바로 못 쓴다

확인 질문

cp 안에 x 가 있나?

ColorPoint 안에서 x 를
쓸 수 있나?

ColorPoint 의 부모는 몇
개인가?

2교시에서 본 것

1

서브 클래스 객체는 슈퍼 클래스와 서브 클래스의 멤버를 모두 갖고 태어난다

2

바깥에서도 서브 클래스 안에서도 슈퍼 클래스의 `private` 은 못 쓴다

3

자바는 클래스 다중 상속을 지원하지 않는다

4

모든 클래스는 `Object` 를 자동으로 물려받는다

2교시 실습 살펴보기

1

실습 ④ 강아지 이름

이름은 Pet 의 private 이라
showName() 을 거쳐 찍었다

2

private 접근 오류

cp.x = 10 도 자식 안의 x 도
같은 has private access 오류다

3교시

protected 접근 지정

자식 클래스에게 열어 주는 protected 를 표 5-1 로 정리한다.

표 5-1

protected

연습문제

지난주 표 4-1

접근 지정자에 따른 멤버 접근 레벨

멤버에 접근하는 클래스	private	디폴트 접근 지정	protected	public
같은 패키지의 클래스	X	O	O	O
다른 패키지의 클래스	X	X	X	O
접근 가능 영역	클래스 내	동일 패키지 내	동일 패키지와 자식 클래스	모든 클래스

오늘

표 4-1의 자식 클래스가 오늘 배운 서브 클래스다.

네 접근 지정자 가운데 이번에는 디폴트와 protected 를 본다

자바의 접근 지정자는 private, public, protected, 디폴트의 4가지로
서, 모든 멤버는 이 중 하나로 반드시 지정되어야 한다.

private ✓ 7주차

public ✓ 7주차

디폴트 오늘

protected 오늘

슈퍼 클래스의 디폴트 멤버는 같은 패키지에서만 쓴다

- 슈퍼 클래스의 디폴트 멤버는 동일한 패키지에 있는 클래스들에게만 접근이 허용된다.
- 만일 서브 클래스가 슈퍼 클래스와 다른 패키지에 있다면, 슈퍼 클래스의 디폴트 멤버를 접근할 수 없다.

같은 패키지의 서브 클래스: O

다른 패키지의 서브 클래스: X

표 5-1 슈퍼 클래스 멤버에 대한 접근 지정

슈퍼 클래스 멤버에 접근하는 클래스 종류	private	디폴트	protected	public
같은 패키지의 클래스	X	O	O	O
다른 패키지의 클래스	X	X	X	O
같은 패키지의 서브 클래스	X	O	O	O
다른 패키지의 서브 클래스	X	X	O	O

O 는 접근 가능, X 는 접근 불가능

표 4-1 에 서브 클래스 두 줄이 늘었다

멤버에 접근하는 클래스	private	디폴트	protected	public
같은 패키지의 클래스	X	O	O	O
다른 패키지의 클래스	X	X	X	O
같은 패키지의 서브 클래스	X	O	O	O
다른 패키지의 서브 클래스	X	X	O	O

표 4-1 과 같다

새로 늘었다

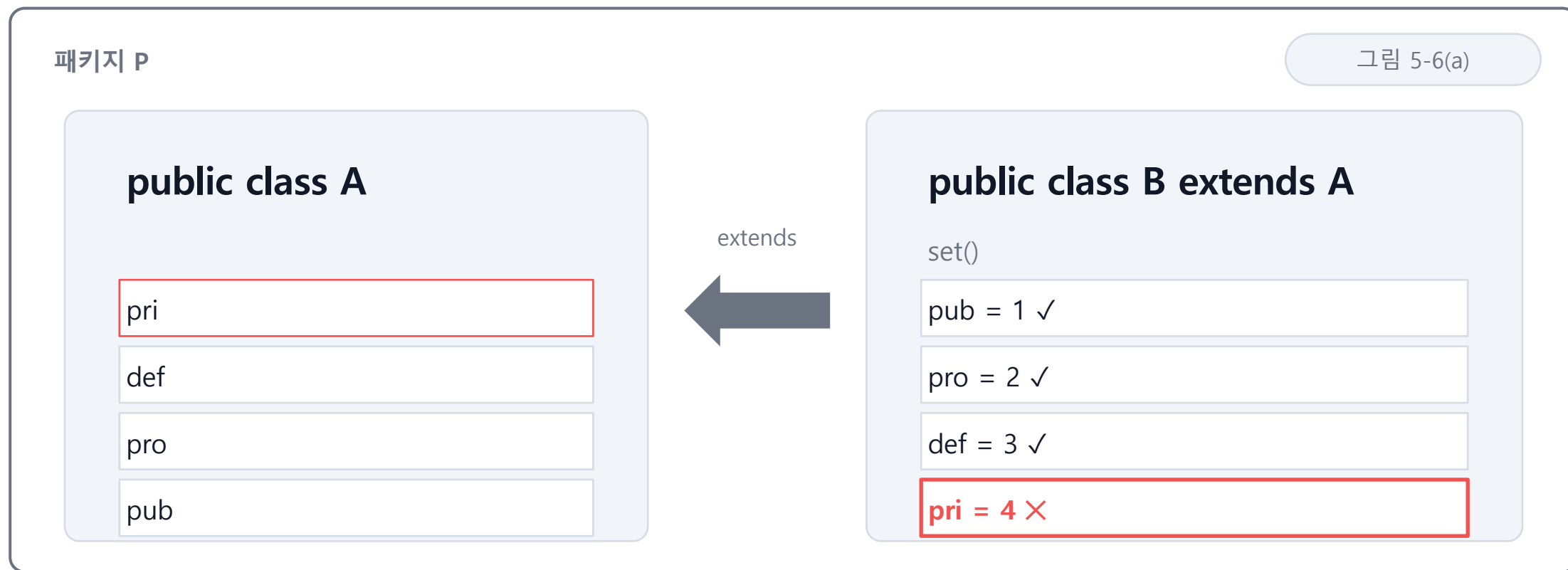
protected 멤버는 두 경우에 쓸 수 있다

슈퍼 클래스의 protected 멤버는 다음 두 가지 경우에 접근이 허용된다.

① 같은 패키지에 속한 모든 클래스

② 상속되는 서브 클래스(같은 패키지든 다른 패키지든 상관 없음)

같은 패키지의 서브 클래스는 private 만 못 쓴다



다른 패키지의 서브 클래스는 protected 와 public 만 쓴다



패키지를 나누는 법은 13주차. 오늘은 그림으로만 본다.

실습실 파일 하나는 같은 패키지라 오늘은 표 5-1 의 1.3행을 실행으로 본다

슈퍼 클래스 멤버에 접근하는 클래스 종류	private	디폴트	protected	public
같은 패키지의 클래스	X	O	O	O
다른 패키지의 클래스	X	X	X	O
같은 패키지의 서브 클래스	X	O	O	O
다른 패키지의 서브 클래스	X	X	O	O

오늘 실행

그림으로만

오늘 실행

그림으로만

한 파일 안의 클래스는 모두 같은 패키지다.

접근 지정자가 다른 필드 넷과 그 넷에 값을 넣는 자식

```
class A {  
    private int pri;  
    int def;  
    protected int pro;  
    public int pub;  
}  
class B extends A {  
    void set() {  
        pub = 1;  
        pro = 2;  
        def = 3;  
        pri = 4;  
    }  
}
```

Ex13_AccessFour.java

B 의 set() 이
A 의 필드 넷 모두에
값을 넣으려 한다.

private

B 객체를 만들어 set() 부르기

Ex13_AccessFour.java

```
public class Ex13_AccessFour {  
    public static void main(String[] args) {  
        B b = new B();  
        b.set();  
    }  
}
```

main 은 B 객체를 만들고 set() 을 부르기만 한다.

컴파일 오류 원문

Ex13_AccessFour.java:12: error: pri has private access in A

```
pri = 4;  
^
```

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

pri 하나만 막힌다

pri 줄을 빼고 세 필드만 채우는 B14

Ex14_AccessThree.java

```
class A14 {  
    private int pri;  
    int def;  
    protected int pro;  
    public int pub;  
}  
class B14 extends A14 {  
    void set() {  
        pub = 1;  
        pro = 2;  
        def = 3;  
    }  
}
```

세 필드 값을 바깥에서 출력하기

Ex14_AccessThree.java

```
public class Ex14_AccessThree {  
    public static void main(String[] args) {  
        B14 b = new B14();  
        b.set();  
        System.out.println(b.pub + " " + b.pro + " " + b.def);  
    }  
}
```

main 도 같은 패키지에 있어 pub · pro · def 를 모두 읽는다.



1 2 3

1 2 3

같은 패키지의 서브 클래스는 세 필드를 모두 썼다

여러 자식이 함께 쓸 필드는 **protected** 로 둔다

private



- 자식도 못 쓴다
- 자식마다 getter 를 거쳐야 한다

protected



- 자식은 바로 쓴다
- 바깥 다른 패키지에는 닫혀 있다

자식에게는 열고 남에게는 닫고 싶을 때 **protected**.

protected 크기를 물려받아 넓이를 구하는 Rect 와 Triangle

Ex15_ShapeRect.java

```
class Shape {  
    protected int width, height;  
    void setSize(int w, int h) { width = w; height = h; }  
}  
class Rect extends Shape {  
    int area() { return width * height; }  
}  
class Triangle extends Shape {  
    int area() { return width * height / 2; }  
}
```

protected 필드

4와 5 크기로 사각형과 삼각형 넓이 구하기

Ex15_ShapeRect.java

```
public class Ex15_ShapeRect {  
    public static void main(String[] args) {  
        Rect r = new Rect();  
        r.setSize(4, 5);  
        Triangle t = new Triangle();  
        t.setSize(4, 5);  
        System.out.println("사각형 넓이: " + r.area());  
        System.out.println("삼각형 넓이: " + t.area());  
    }  
}
```

사각형 넓이 20, 삼각형 넓이 10

사각형 넓이: 20

삼각형 넓이: 10

두 자식이 부모의 protected 필드를 바로 썼다

직접 실습 ⑤: Healer 가 Unit 의 체력을 올리게 하 기

heal() 빈칸에서 protected hp 에 30 을 더한다.

hp 는 Unit 의 protected

처음 체력 50

heal() 빈칸에서 hp 를 30 올려 보기

Ex16_HealerUnit.java

할 일: protected hp 에 30 을 더한다.

```
class Unit {  
    protected int hp = 50;  
    void showHp() { System.out.println("체력: " + hp); }  
}  
class Healer extends Unit {  
    void heal() {  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

회복 전과 후의 체력 출력하기

Ex16_HealerUnit.java

```
public class Ex16_HealerUnit {  
    public static void main(String[] args) {  
        Healer h = new Healer();  
        h.showHp();  
        h.heal();  
        h.showHp();  
    }  
}
```

main 은 손대지 않고 실행한다.

체력 50 에서 80

체력: 50

체력: 80

자식이 부모의 protected 필드를 바꿨다



빈칸 그대로 실행하면

체력: 50

체력: 50

heal() 이 아무것도 하지 않는다

hp 가 private 이었다면 heal() 에서 바로 쓸 수 없다

Unit

private int hp = 50;



Healer.heal()

hp = hp + 30;

hp has private access in Unit

protected 라서 자식이 바로 썼다.

문제: 컴파일 오류가 나는 곳은 어디인가

4. 접근 지정에 관한 문제이다. 컴파일 오류가 발생하는 곳은 어디인가?

```
class A {  
    private int x, y;  
    protected void setX(int x) { this.x = x; } // ①  
}  
class B extends A {  
    private int w;  
    protected void setXY(int a, int b, int c) {  
        setX(a+b); // ②  
        y = b; // ③  
        w = c; // ④  
    }  
}
```

연습문제 4 를 그대로 옮긴 A17 과 B17

Ex17_Exercise4.java

```
class A17 {  
    private int x, y;  
    protected void setX(int x) { this.x = x; }  
}  
class B17 extends A17 {  
    private int w;  
    protected void setXY(int a, int b, int c) {  
        setX(a + b);  
        y = b;  
        w = c;  
    }  
}
```



B17 객체로 setXY() 부르기

Ex17_Exercise4.java

```
public class Ex17_Exercise4 {  
    public static void main(String[] args) {  
        B17 b = new B17();  
        b.setXY(1, 2, 3);  
    }  
}
```

컴파일 오류 원문

Ex17_Exercise4.java:9: error: y has private access in A17

```
y = b;  
^
```

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

y 는 A17 의 private 이다

풀이: ③ `y = b;` 는 부모의 `private` 을 바로 쓴다

번호	줄	결과	까닭
①	<code>this.x = x;</code>	된다	A 안에서 자기 <code>private</code>
②	<code>setX(a+b);</code>	된다	부모의 <code>protected</code> 메소드
③	<code>y = b;</code>	오류	부모의 <code>private</code>
④	<code>w = c;</code>	된다	B 안에서 자기 <code>private</code>

`y` 는 A 의 `private` 이라 자식 B 가 바로 쓸 수 없다.

직접 실습 ⑥: Bus 가 Vehicle 의 protected 필드를 채우게 하기

할 일

setBus() 빈칸에서 **protected speed** 에 값을 넣는다.

number · speed 는 Vehicle 의 protected

number 줄은 채워 두었다

setBus() 빈칸에서 speed 채워 보기

Ex18_BusVehicle.java

할 일: this.number 줄을 따라 speed 를 채운다.

```
class Vehicle {  
    protected String number;  
    protected int speed;  
    void showInfo() { System.out.println(number + " 속도 " + speed); }  
}  
class Bus extends Vehicle {  
    void setBus(String number, int speed) {  
        this.number = number;  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

버스 두 대를 만들어 정보 출력하기

Ex18_BusVehicle.java

```
public class Ex18_BusVehicle {  
    public static void main(String[] args) {  
        Bus b1 = new Bus();  
        b1.setBus("경기 77", 60);  
        b1.showInfo();  
        Bus b2 = new Bus();  
        b2.setBus("서울 12", 40);  
        b2.showInfo();  
    }  
}
```

main 은 고치지 않는다.



경기 77 속도 60, 서울 12 속도 40

경기 77 속도 60

서울 12 속도 40

자식이 채운 값을 부모의 showInfo() 가 찍었다



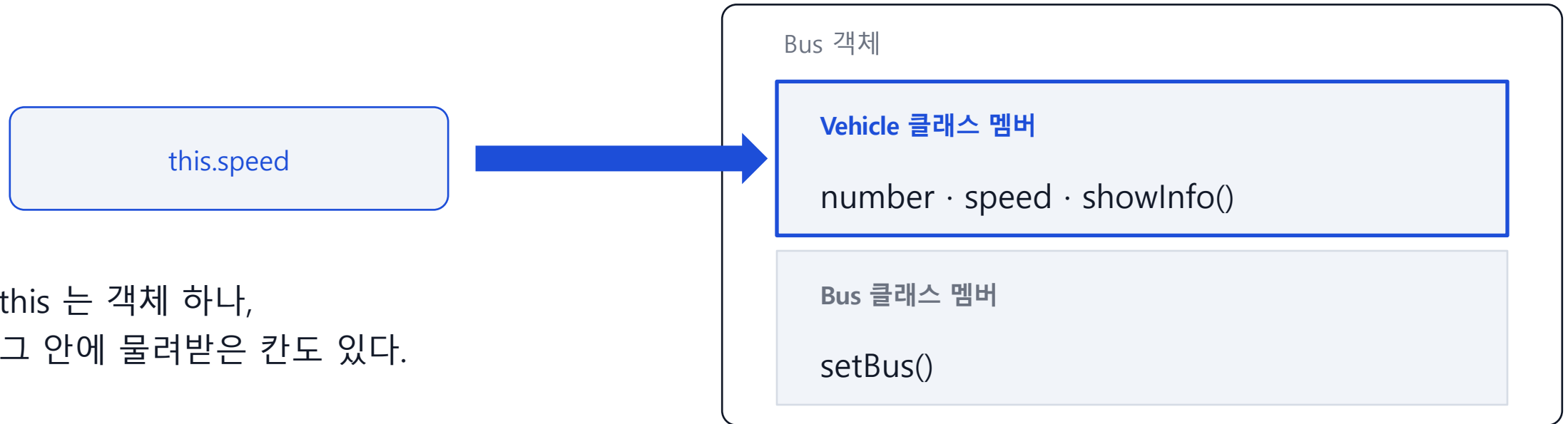
빈칸 그대로 실행하면

경기 77 속도 0

서울 12 속도 0

speed 를 넣지 않으면 0 이다

this.speed 는 Bus 가 Vehicle 에게서 물려받은 필드다



문제: objA, objB, objC 의 멤버 나열

1. 다음 코드에 대해 물음에 답하라.

```
class A {  
    private int a, b;  
    public void set(int a, int b) { this.a = a; this.b = b; }  
}  
class B extends A {  
    protected int c, d;  
}  
class C extends B {  
    public int e, f;  
}
```

- (1) A objA = new A();에 의해 생성되는 객체 objA의 멤버들을 모두 나열하라.
- (2) B objB = new B();에 의해 생성되는 객체 objB의 멤버들을 모두 나열하라.
- (3) C objC = new C();에 의해 생성되는 객체 objC의 멤버들을 모두 나열하라.

풀이: 아래 층 객체일수록 멤버가 많다

객체	멤버
objA	a, b set()
objB	a, b set(), c, d
objC	a, b set(), c, d, e, f

a, b

private 이지만 객체 안에 있다

세 층 클래스 A19, B19, C19

Ex19_Exercise1.java

```
class A19 {  
    private int a, b;  
    public void set(int a, int b) { this.a = a; this.b = b; }  
}  
class B19 extends A19 {  
    protected int c, d;  
}  
class C19 extends B19 {  
    public int e, f;  
}
```

결과를 먼저 예측한 뒤 실행해 비교한다.

objC 에 값을 넣고 c, d, e, f 의 합 출력하기

Ex19_Exercise1.java

```
public class Ex19_Exercise1 {  
    public static void main(String[] args) {  
        C19 objC = new C19();  
        objC.set(1, 2);  
        objC.c = 3; objC.d = 4;  
        objC.e = 5; objC.f = 6;  
        System.out.println(objC.c + objC.d + objC.e + objC.f);  
    }  
}
```

출력되는 수는 얼마일까

```
System.out.println(objC.c + objC.d + objC.e + objC.f);
```

()

문자열이 아니라 int 넷의 덧셈이다.



18

18

$3 + 4 + 5 + 6$

private a, b 도 objC 안에 있다.
바로 쓰지 못할 뿐이다

objC 객체

A19 멤버

a 1 · b 2 · set()

B19 멤버

c 3 · d 4

C19 멤버

e 5 · f 6

a · b 있지만 바로 못 쓴다

문제: 상속과 접근 지정자에 대한 설명 가운데 틀린 것은

2. 상속에 관한 접근 지정자에 대한 설명이다. 틀린 것은?

- ① 슈퍼 클래스의 private 멤버는 서브 클래스에서 바로 접근할 수 없다.
- ② 슈퍼 클래스의 protected 멤버는 같은 패키지에 있는 서브 클래스에서만 접근할 수 있다.
- ③ 슈퍼 클래스의 public 멤버는 다른 모든 클래스에서 접근할 수 있다.
- ④ 슈퍼 클래스의 디폴트 멤버는 같은 패키지에 있는 다른 모든 클래스에서 접근 가능하다.

풀이: ② protected 는 다른 패키지의 서브 클래스에서도 쓴다

틀린 것은 ②. protected 멤버는 같은 패키지든 다른 패키지든 서브 클래스에서 쓸 수 있다.

	private	디폴트	protected	public
다른 패키지의 서브 클래스	X	X	O	O

표 5-1 넷째 줄 · 다른 패키지의 서브 클래스

자식 객체가 생길 때 부모의 생성자는 어떻게 될까

cp 를 new 로 만들 때 Point 의 생성자도 실행될까?

상속과 생성자

super()

다음 주에 답한다.

3교시에서 본 것

1

디폴트 멤버는 같은 패키지에서만 쓴다

2

protected 멤버는 같은 패키지와 서브 클래스가 쓴다

3

같은 패키지의 서브 클래스는 부모의 private 만 못 쓴다

4

여러 자식이 함께 쓸 필드는 protected 로 둔다

3교시 실습 살펴보기

1

hp 는 protected 라 Healer 안에서 바로 30 을 더했다(실습 ⑤)

2

speed 를 넣지 않으면 0 이 찍힌다.
물려받은 필드도 채우지 않으면 기본값이다(실습 ⑥)

안 될 때 여기부터

틀림

- 부모 객체로 자식 메소드를 부름
- 자식이 부모 private 을 씀
- extends 뒤에 부모를 둘 씀

맞음

- ✓ 자식 객체를 만들어 부른다
- ✓ 부모 메소드 · protected
- ✓ 부모는 하나만 쓴다

오류 줄의 클래스가 부모인지 자식인지 먼저 본다

오늘 공부한 내용

상속은 클래스가 다른 클래스의 필드와 메소드를 물려받는 것이다

class 자식 extends 부모 로 선언하고, 부모는 하나뿐이다

서브 클래스 객체는 두 클래스의 멤버를 모두 갖고 태어난다

슈퍼 클래스의 private 은 자식도 바로 쓰지 못한다

protected 멤버는 같은 패키지과 서브 클래스가 쓴다

다음 주엔 상속 심화

자식 객체가 생길 때 부모의 생성자가 어떻게 불리는지 봅니다



다음 주 예고

- 상속과 생성자
- `super()` · 생성자 호출 순서