

경기과학기술대학교 · 인공지능학과

자바 프로그래밍

클래스라는 틀, 객체라는 실체

틀을 한 번 만들면,
객체는 몇 개든 찍어 낸다.

5주차



오늘의 강의

1 교시 객체 지향과 클래스, 2 교시 객체 생성과 메소드, 3 교시 객체 배열

1교시



클래스 만들기

특성은 필드, 행동은 메소드
클래스는 틀

2교시



객체 쓰기

new 로 만들고 점으로 쓴다
메소드는 값을 돌려준다

3교시



객체 배열

원소는 레퍼런스
원소마다 new

1교시

객체 지향과 클래스

세상의 물건을 특성과 행동으로 나눠 틀을 만든다.

특성과 행동

캡슐화

클래스 선언

지난주에서 이어받는 것



지난주

- 배열 선언은 레퍼런스 변수를 만드는 것이고, 배열 생성은 new 로 공간을 만드는 것이다.
- 선언만 한 배열 변수는 null 이고, 생성해야 원소에 값을 넣을 수 있다.

→ 오늘은 배열이 아닌 객체를 new 로 만든다.

세상 모든 것이 객체

TV · 의자 · 책 · 집 · 카메라 · 컴퓨터

TV, 컴퓨터, 책, 건물, 의자, 사람 등 실세계는 객체들의 집합이다.



교재 그림 4-1 재구성

객체는 저마다 다른 특성과 행동을 가진다



특성 (state)

명사로 적으면 특성이다



행동 (behavior)

동사로 적으면 행동이다

다른 객체에게 행동을 요구하거나 정보를 주고받는다.

TV 한 대의 특성과 행동

특성

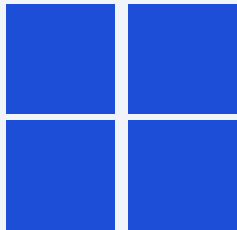
- 전원 켜짐/꺼짐
- 채널 번호
- 음량

행동

- 켜고 끄기
- 채널 바꾸기
- 소리 키우기

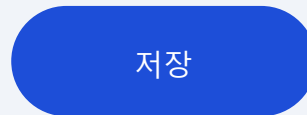
프로그램 속 객체

게임 블록



블록 하나가 객체

메뉴·버튼



버튼 하나가 객체

정리



블록도 버튼도
객체다



글자로 그린 블록

1984년 모스크바 계산 센터에서 알렉세이 파지트노프가 일렉트로니카 60이라는 기기로 테트리스를 만들었다.

그래픽이 없는 컴퓨터라 블록의 한 칸을 글자 두 개로 그렸다.

색도 소리도 없었지만 블록은 떨어지고, 돌고, 옆으로 움직였다.

게임 조각의 특성과 행동

특성

- 모양
- 색
- 자리(가로, 세로)

행동

- 떨어지기
- 돌기
- 옆으로 가기

특성은 필드가 되고 행동은 메소드가 된다.

내 주변 객체

1. 책상 위 물건 하나를 고른다
2. 특성 셋과 행동 둘을 생각해 본다

예: 휴대폰



특성: 기종·배터리 양·화면 밝기

행동: 전화 걸기·사진 찍기



캡슐화



교재 정의

캡슐화란 객체를 캡슐로 싸서 그 내부를 보호하고 볼 수 없게 하는 것으로 객체의 가장 본질적인 특징이다.

캡슐에 든 약은 어떤 색인지, 어떤 성분인지 보이지 않는다.



껌데기가 속을 감싼다



껌슐약



TV



자판기



카메라



사람

TV에 케이스가 없다면 외부의 껌촉으로부터 보호할 수 없다.



감춘 것과 공개한 것

캡슐화가 기본이고, 밖과 통하는 몇 부분만 공개한다.

감춘 것

- 회로
- 부품

공개한 것

- ✓ On/Off 버튼
- ✓ 밝기 조절 버튼
- ✓ 채널 버튼 · 음량 버튼
- ✓ 리모컨과 통신하는 부분

클래스로 캡슐화

```
class Animal {  
    String name;  
    int age;  
  
    void eat() {...}  
    void speak() {...}  
    void love() {...}  
}
```

교재 그림의 표기 그대로

Animal 객체 두 개 생성

name "lion", age 4

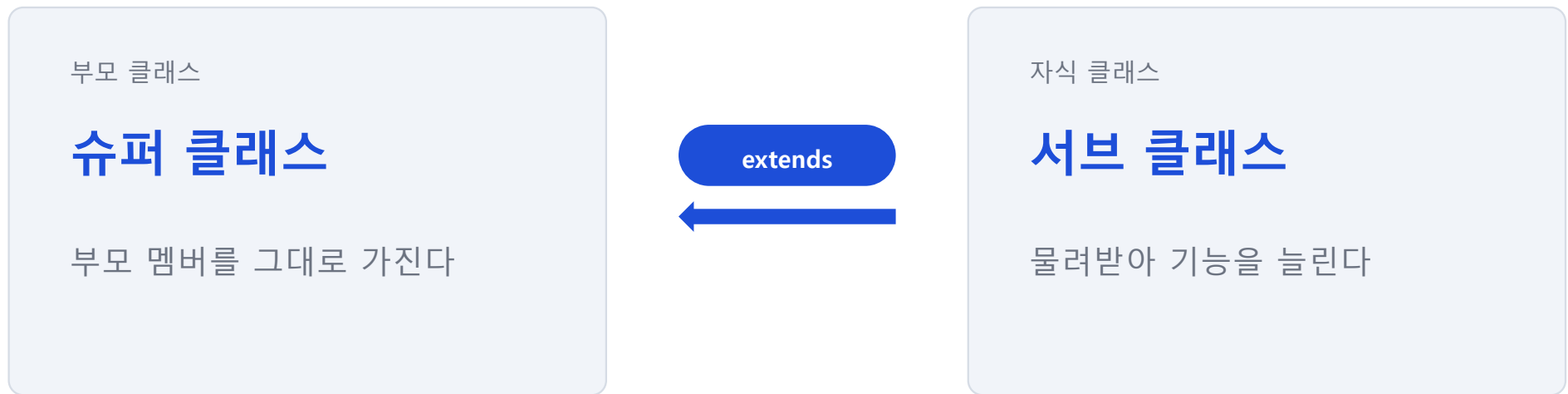
name "bear", age 8

상속

상위 개체의 속성을 하위 개체가 그대로 물려받는다



슈퍼 클래스와 서브 클래스



물려받으므로 같은 코드를 다시 쓰지 않는다.

Animal 을 물려받은 Human

```
class Animal {  
    ...  
}
```

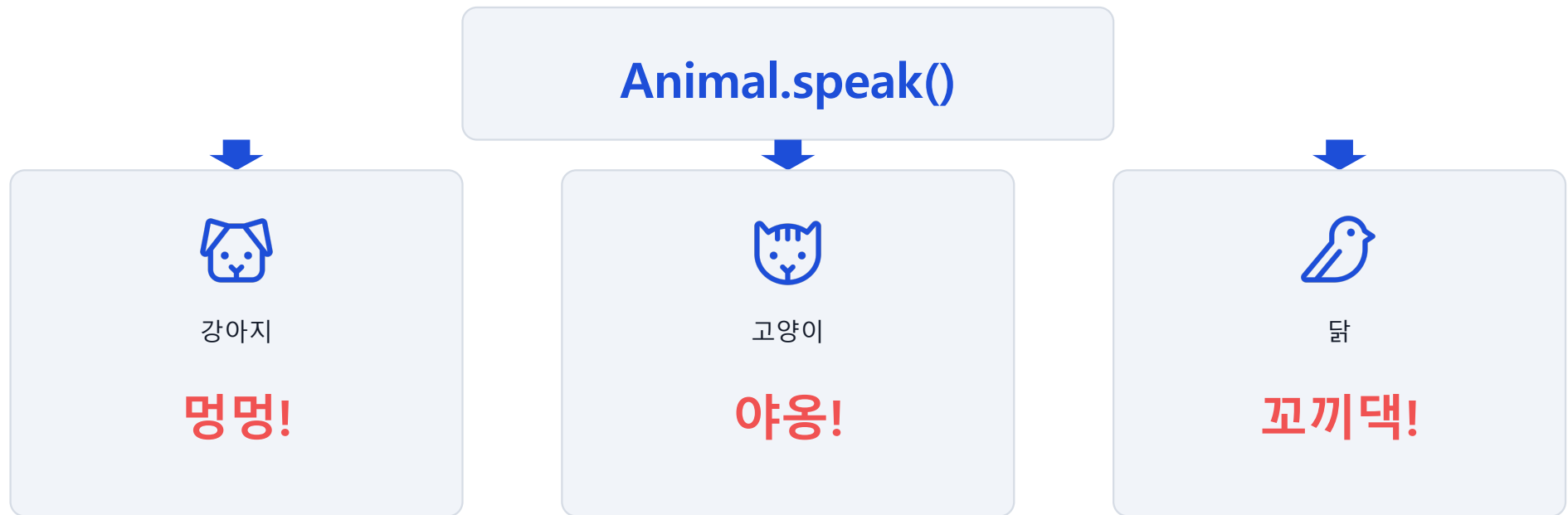
```
class Human extends Animal {  
    String hobby;  
    String job;  
    void work() {...}  
    void cry() {...}  
    void laugh() {...}  
}
```

Animal 객체 · 멤버 5개

Human 객체 · 멤버 10개

hobby · job · work · cry · laugh

다형성 - 같은 이름의 메소드가 클래스마다 다르게 구현됨



이름은 하나, `speak()` 는 객체마다 다르게 움직인다.

세 가지 특성



캡슐화

감싸서 보호하고 몇 부분만 공개

TV 케이스와 버튼



상속

물려받아 확장

생물 → 동물 → 사람



다형성

같은 이름, 다른 동작

멍멍 · 야옹 · 꼬끼댕

문제: 예문으로 특성 설명하기

1

'메뚜기는 생물이다.'는 문구로 상속을 설명하라.

2

'TV는 튼튼한 플라스틱의 외형 때문에 내부를 볼 수 없다.'는 문구로 캡슐화를 설명하라.

3

'볼펜, 만년필, 연필은 쓰는 도구이지만, 각기 쓰는 방법이 다르다.'는 문구로 다형성을 설명하라.

풀이: 세 예문

메뚜기는 생물의 속성을 물려받는다



상속

플라스틱 외형이 속을 감싸 보호한다



캡슐화

같은 "쓰기"를 도구마다 다르게 한다



다형성

객체 지향 언어의 목적



소프트웨어의 생산성 향상

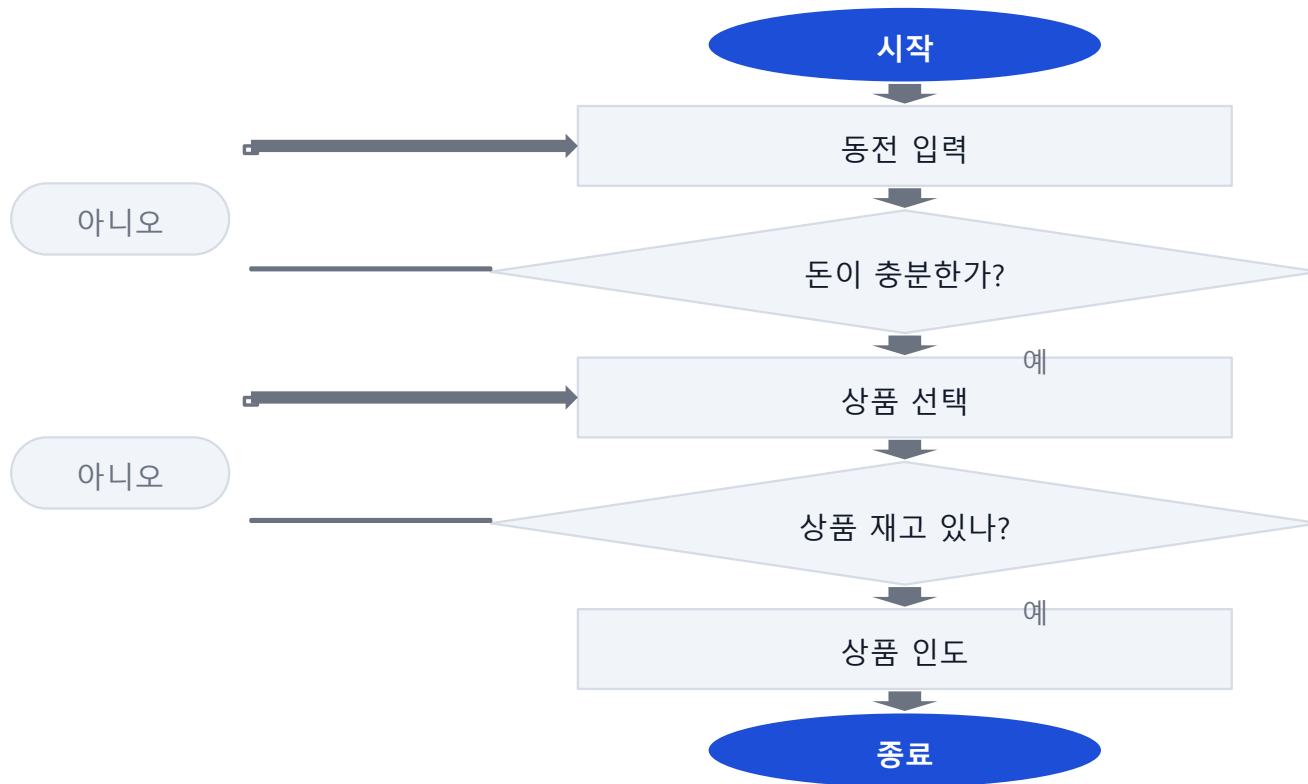
이미 만든 것을 상속받거나 재사용하기 쉽다



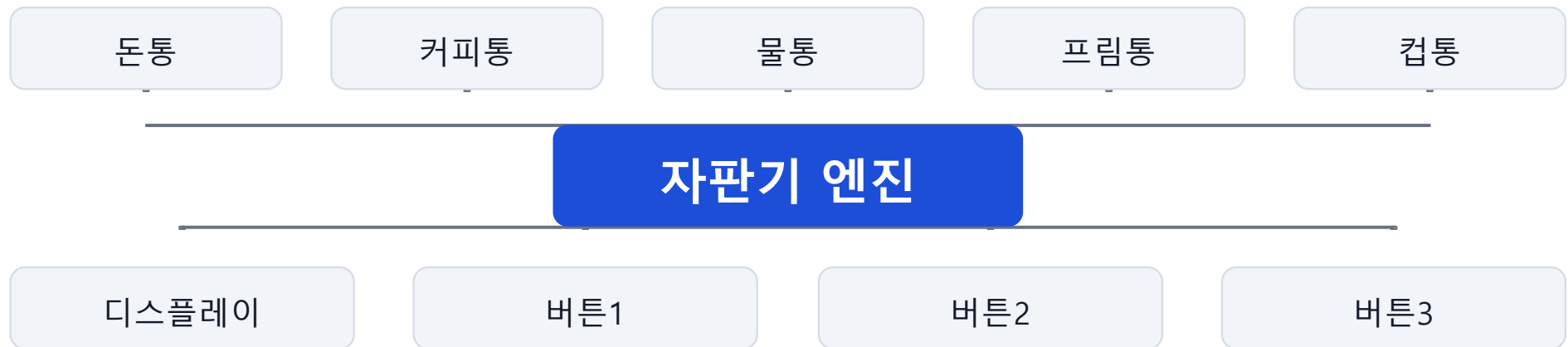
실세계에 대한 쉬운 모델링

실세계의 일을 더 쉽게 프로그래밍한다

절차 지향으로 만든 자판기



객체 지향으로 만든 자판기



객체를 추출하고 객체들의 관계를 결정하고
상호 작용에 필요한 메소드와 필드를 설계한다.

두 방식 나란히

절차 지향

객체 지향

중심

일의 순서

물체와 관계

먼저 그리는 것

흐름도

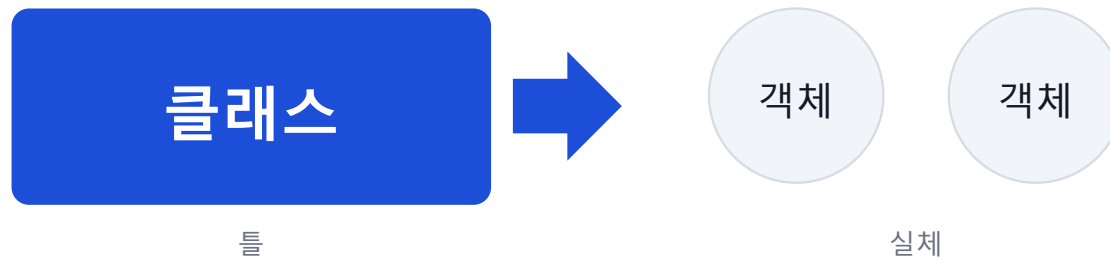
객체 관계도

예 언어

C

자바

클래스와 객체



클래스는 객체를 만들어 내기 위한 틀이고, 그 틀대로 만들어진 실체가 객체다.

객체를 클래스의 인스턴스라고도 부른다.

붕어빵 틀과 붕어빵



붕어빵 틀은 클래스이며, 이 틀의 형태로 구워진 붕어빵은 객체입니다.

붕어빵은 틀의 모양대로 만들어지지만 서로 조금씩 다릅니다.

하나의 클래스로 객체를 수없이 많이 만들 수 있다.

같은 속성, 다른 값

클래스: 사람 (이름·직업·나이·성별·혈액형)

행동: 밥 먹기·잠자기·말하기·걷기

속성	최승희	이미녀	김미남
이름	최승희	이미녀	김미남
직업	의사	골프 선수	교수
나이	45	28	47
성별	여	여	남
혈액형	A	O	AB

사람 클래스를 코드로

표의 속성 다섯이 필드 다섯이 된다.

```
class Person {  
    String name;  
    String job;  
    int age;  
    String gender;  
    String blood;  
}
```

문제: 틀과 흐름도

문제 1

자바에서 객체를 정의하는 틀은 무엇인가?

문제 3

다음 중 자바의 객체 지향 개념에 포함되지 않는 것은?

① 상속 ② 흐름도 ③ 캡슐화 ④ 다형성

풀이: 틀과 흐름도



문항 1

클래스

객체를 만드는 틀은 클래스다.



문항 3

② 흐름도

흐름도는 절차 지향 설계에서 그린다.

class 로 새 타입을 만든다



기본 제공 타입

언어가 이미 만들어 둔 타입이다

int · double · boolean · char



class 로 만든 타입

구조체처럼 개발자가 정의한다

- 필드
- 메소드

클래스 선언의 구성

```
public class Circle {  
    int radius;  
    String name;  
  
    public double getArea() {  
        return 3.14*radius*radius;  
    }  
}
```

public ← 클래스에 대한 접근 권한

class ← 클래스 선언

Circle ← 클래스 이름

int radius; String name; ← 필드(변수)

public double getArea() ← 메소드

필드와 메소드



필드

객체의 상태 값을 저장할 멤버 변수다.



메소드

실행 가능한 함수로 객체의 행동을 구현한다.



getArea()

반지름으로 면적을 계산해 알려준다.

멤버는 중괄호 안에만

클래스 외부에 필드나 메소드를 결코 둘 수 없다.

```
class Circle {  
    int radius;  
}  
String name;
```

①

public 은 공개한다는 뜻

public

```
public class Circle { ... }
```

다른 클래스에서 이 클래스를
쓸 수 있게 공개한다는 뜻이다.

한 파일에 클래스 둘

```
class Circle {  
    필드와 메소드
```

```
public class Ex01_CircleEx {  
    public static void main(String[] args) {  
        Circle 객체를 만들어 쓴다
```

파일 이름은 Ex01_CircleEx.java

객체 만들기 두 단계

1

변수 선언

레퍼런스 변수를 선언해도 객체는 생기지 않는다.

```
Circle pizza;
```

2

객체 생성

new 로 객체를 만들고, 점(.)으로 필드와 메소드에 쓴다.

```
pizza = new Circle();
```

예제 4-1 Circle 클래스

Ex01_CircleEx.java

```
class Circle {  
    int radius; // 원의 반지름을 지정하는 멤버 변수  
    String name; // 원의 이름을 지정하는 멤버 변수  
  
    public double getArea() { // 멤버 메소드  
        return 3.14*radius*radius;  
    }  
}
```

예제 4-1 main 에서 쓰기

Ex01_CircleEx.java

```
public class Ex01_CircleEx {  
    public static void main(String[] args) {  
        Circle pizza; // 객체에 대한 레퍼런스 변수 pizza 선언  
        pizza = new Circle(); // Circle 객체 생성  
        pizza.radius = 10; // 피자 반지름을 10으로 설정  
        pizza.name = "자바피자"; // 피자의 이름 설정  
        double area = pizza.getArea(); // 피자의 면적 알아내기  
        System.out.println(pizza.name + "의 면적은 " + area);  
        Circle donut = new Circle(); // Circle 객체 생성  
        donut.radius = 2; // 도넛의 반지름을 2로 설정  
        donut.name = "자바도넛"; // 도넛의 이름 설정  
        area = donut.getArea(); // 도넛의 면적 알아내기  
        System.out.println(donut.name + "의 면적은 " + area);  
    }  
}
```

예제 4-1 실행 결과

자바피자의 면적은 314.0

자바도넛의 면적은 12.56

같은 틀로 만든 두 객체의 면적이 다르다

틀 하나로 객체 둘

pizza 객체

```
pizza.radius = 10;
```

donut 객체

```
donut.radius = 2;
```

Circle 은 한 번 썼고 객체는 둘이다.

예제 4-2 Rectangle 객체 만들기

Ex02_RectangleEx.java · 할 일: rect 객체를 만들고 너비 4, 높이 5를 넣은 뒤 면적을 출력한다

```
class Rectangle {
    int width;
    int height;
    int getArea() {
        return width*height;
    }
}

public class Ex02_RectangleEx {
    public static void main(String[] args) {

        Rectangle rect = new Rectangle(); // 객체 생성
        rect.width = 4;
        // 여기서 부터 코드를 작성하세요.

        // 여기 까지
    }
}
```

예제 4-2 실행 결과

사각형의 면적은 20

4 곱하기 5, 면적은 20

탱크 객체로 메소드 불러 보기

Ex03_TankHalf.java

```
class Tank {  
    int liters;  
    int getHalf() {  
        return liters / 2;  
    }  
}  
  
public class Ex03_TankHalf {  
    public static void main(String[] args) {  
        Tank t = new Tank();  
        t.liters = 60;  
        System.out.println("절반은 " + t.getHalf() + "리터");  
    }  
}
```

전등 두 개의 요금 예측해 보기

Ex04_LampCost.java

두 줄 출력을 공책에 먼저 적은 뒤 실행해 확인한다

```
class Lamp {  
    String place;  
    int watt;  
    int getCost() {  
        return watt * 2;  
    }  
}
```

전등 두 개 만들어 요금 출력하기

Ex04_LampCost.java

```
public class Ex04_LampCost {  
    public static void main(String[] args) {  
        Lamp a = new Lamp();  
        a.place = "거실";  
        a.watt = 60;  
        Lamp b = new Lamp();  
        b.place = "방";  
        b.watt = 30;  
        System.out.println(a.place + " " + a.getCost());  
        System.out.println(b.place + " " + b.getCost());  
    }  
}
```

전등 요금 결과

거실 120
방 60

메소드는 하나, 결과는 객체마다

고양이 객체에 값 넣어 보기

Ex05_CatInfo.java

```
class Cat {  
    String name;  
    int age;  
}  
  
public class Ex05_CatInfo {  
    public static void main(String[] args) {  
        Cat c = new Cat();  
        c.name = "나비";  
        c.age = 2;  
        System.out.println(c.name + " " + c.age);  
    }  
}
```

개 두 마리에 값 채워 보기

Ex06_DogPair.java · d1 에는 초코·3살, d2 에는 보리·5살을 넣는다

```
class Dog {
    String name;
    int age;
}

public class Ex06_DogPair {
    public static void main(String[] args) {
        Dog d1 = new Dog();
        Dog d2 = new Dog();
        // 여기서 부터 코드를 작성하세요.

        // 여기 까지
        System.out.println(d1.name + " " + d1.age + " / " + d2.name + " " + d2.age);
    }
}
```

개 두 마리 결과

초코 3 / 보리 5

두 객체의 필드가 다르다

필드 셋인 기차 클래스

교수 시연

Ex07_TrainInfo.java

```
class Train {  
    int number;  
    String line;  
    double speed;  
}  
  
public class Ex07_TrainInfo {  
    public static void main(String[] args) {  
        Train t = new Train();  
        t.number = 101;  
        t.line = "경부선";  
        t.speed = 300.5;  
        System.out.println(t.number + " " + t.line + " " + t.speed);  
    }  
}
```

버스 클래스에 필드 더해 보기

Ex08_BusInfo.java · Bus 클래스에 색(color) 필드를 하나 더 만들고 값을 넣어 출력 줄 끝에 붙인다

```
class Bus {  
    int number;  
    int seats;  
}  
  
public class Ex08_BusInfo {  
    public static void main(String[] args) {  
        Bus bus = new Bus();  
        bus.number = 720;  
        bus.seats = 40;  
        System.out.println(bus.number + "번 버스, 좌석 " + bus.seats);  
    }  
}
```

버스 정보 결과

720번 버스, 좌석 40, 파랑

틀에 칸이 하나 늘었다

1교시에서 본 것



1. 객체는 특성과 행동을 가진다
2. 클래스는 틀, 객체는 그 틀로 만든 실체
3. 특성은 필드, 행동은 메소드
4. 선언만으로는 객체가 없고 new 로 만든다

1교시 실습 살펴보기

- Rectangle 은 public 이 없는데도 main 에서 쓸 수 있다(같은 파일 안)
- rect.getArea() 를 println 안에 바로 넣었다. 메소드를 부른 자리가 값이 된다(2교시)

보강 실습

- 메소드는 하나여도 계산은 부른 객체의 필드로 한다(전등 120 · 60)
- 필드를 하나 늘리면 그 틀로 만든 모든 객체에 칸이 생긴다. 대입과 출력도 함께 고친다(버스 color)

2교시

객체 생성과 메소드

new 로 만든 객체에 점으로 값을 넣고 메소드를 부른다.

new 와 레퍼런스

기본값

리턴 타입

레퍼런스 변수 선언

선언문

Circle pizza;

- 이 줄로는 객체가 생기지 않는다
- pizza는 가리킬 자리일 뿐이다
- 다르게는 객체 이름이라 한다

그림으로 보면

Circle

pizza

가리키는 대상이 아직 없다

new 로 객체 만들기

```
pizza = new Circle();
```

new 연산자는 Circle 타입의 크기만한 메모리를 할당받아 메모리에 대한 레퍼런스(주소)를 리턴한다.



객체 메모리 할당 및 객체 생성

Circle 객체

radius

name

```
getArea() { ... }
```

선언과 생성을 한 줄로

두 줄로 쓰면

```
Circle pizza;  
pizza = new Circle();
```

지난 장의 두 단계

=

한 줄로 쓰면

```
Circle pizza = new Circle();
```

실무에서 가장 흔한 형태

객체 없이 필드에 넣으면

내 작업공간/Ex09_NoObject.java

```
class Cup {  
    int size;  
}
```

```
public class Ex09_NoObject {  
    public static void main(String[] args) {  
        Cup cup;  
        cup.size = 350;  
        System.out.println(cup.size);  
    }  
}
```

컴파일 오류 화면

Ex09_NoObject.java:8: error: variable cup might not have been initialized

```
    cup.size = 350;
```

^

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

new 로 만들지 않은 레퍼런스는 쓸 수 없다.

점(.)으로 멤버에 쓴다

형식

객체레퍼런스.멤버

`pizza.radius = 10; // pizza 객체의 radius 값을 10으로 설정`

필드

`pizza.name = "자바피자"; // pizza 객체의 name에 "자바피자" 대입`

필드

`double area = pizza.getArea(); // pizza 객체의 getArea() 메소드 호출`

메소드

필드에 값이 들어가는 모습

```
pizza.radius = 10;
```



radius = 10

radius 값 변경

```
pizza.name = "자바피자";
```



name = "자바피자"

name 값 변경

없는 필드에 넣으면

내 작업공간/Ex10_NoField.java

```
class Phone {  
    String model;  
    int battery;  
}  
  
public class Ex10_NoField {  
    public static void main(String[] args) {  
        Phone phone = new Phone();  
        phone.model = "자바폰";  
        phone.color = "검정";  
        System.out.println(phone.model);  
    }  
}
```

컴파일 오류 화면

Ex10_NoField.java:10: error: cannot find symbol

```
    phone.color = "검정";
```

^

symbol: variable color

location: variable phone of type Phone

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

클래스에 없는 필드는 점 뒤에 올 수 없다

메소드를 부르면

① 호출

```
double area = pizza.getArea();
```

② 메소드 안

```
return 3.14*radius*radius;
```

getArea() 메소드 실행

③ 결과 대입

```
area 314.0
```

메소드가 돌려준 값이 변수에 들어온다.

객체마다 필드가 따로

● pizza

- radius 10
- name "자바피자"

● donut

- radius 2
- name "자바도넛"

두 객체 모두 같은 `getArea()`



값을 안 넣은 필드는

```
Pen pen = new Pen();  
System.out.println(pen.color);  
System.out.println(pen.price);
```

기본값 확인해 보기

Ex11_DefaultField.java

```
class Pen {  
    String color;  
    int price;  
}  
  
public class Ex11_DefaultField {  
    public static void main(String[] args) {  
        Pen pen = new Pen();  
        System.out.println(pen.color);  
        System.out.println(pen.price);  
    }  
}
```

기본값 확인 결과

null
0

글자 필드는 null, 정수 필드는 0

필드의 기본값

int

0

double

0.0

boolean

false

String

null

지난주 배열 원소의 기본값과 같다.

레퍼런스를 옮기면

```
Point p1 = new Point();  
p1.x = 1;  
Point p2 = p1;  
p2.x = 100;
```

객체 공유 확인해 보기

```
class Point {  
    int x;  
    int y;  
}  
  
public class Ex12_PointShare {  
    public static void main(String[] args) {  
        Point p1 = new Point();  
        p1.x = 1;  
        Point p2 = p1;  
        p2.x = 100;  
        System.out.println("p1.x 는 " + p1.x);  
        System.out.println("p2.x 는 " + p2.x);  
    }  
}
```

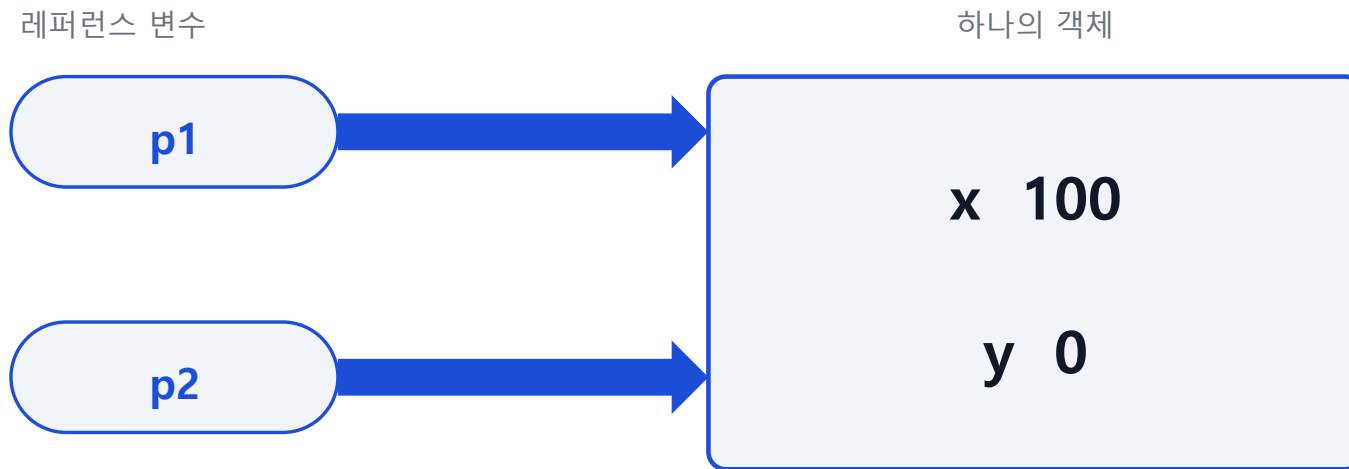
객체 공유 결과

p1.x 는 100

p2.x 는 100

p1 과 p2 는 같은 객체를 가리킨다

이름 둘, 객체 하나



레퍼런스를 대입하면 객체는 그대로 하나다.

새 객체의 필드는 기본값

Ex13_BallDefault.java

```
class Ball {  
    int size;  
}  
  
public class Ex13_BallDefault {  
    public static void main(String[] args) {  
        Ball a = new Ball();  
        a.size = 5;  
        Ball b = new Ball();  
        System.out.println(a.size + " " + b.size);  
    }  
}
```

로봇 세 줄 출력 예측해 보기

Ex14_RobotPredict.java

세 줄 출력을 먼저 적은 뒤 실행해 확인한다. r3 줄을 맞춰 본다

```
class Robot {  
    String name;  
    int power;  
}
```

로봇 셋 만들어 출력하기

Ex14_RobotPredict.java

```
public class Ex14_RobotPredict {  
    public static void main(String[] args) {  
        Robot r1 = new Robot();  
        r1.name = "알파";  
        Robot r2 = r1;  
        r2.power = 50;  
        Robot r3 = new Robot();  
        System.out.println(r1.name + " " + r1.power);  
        System.out.println(r2.name + " " + r2.power);  
        System.out.println(r3.name + " " + r3.power);  
    }  
}
```

로봇 출력 결과

알파 50
알파 50
null 0

r2 는 r1 과 같은 객체, r3 는 새 객체

학생 객체 둘 만들어 보기

Ex15_MyStudent.java · 나와 친구, Student 객체 둘을 만들어 각자 값을 넣고 한 줄씩 출력한다

```
class Student {  
    String name;  
    int year;  
    String dept;  
}  
  
public class Ex15_MyStudent {  
    public static void main(String[] args) {  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

학생 객체 결과

김자바 1학년 인공지능학과
이객체 1학년 인공지능학과

같은 틀, 다른 값의 두 객체

메소드는 객체의 행동



메소드의 정의

- 클래스 안의 함수다.
- 객체의 필드로 일한다.



getArea() 예시

그 객체의 radius로 면적을 계산한다.
radius가 다르면 결과도 다르다.

리턴 타입과 return

1. 메소드 이름 앞의 타입은 돌려줄 값의 타입이다
2. return 뒤의 값을 부른 쪽으로 돌려준다
3. 돌려줄 값이 없으면 void

타입이 오는 자리

타입

메소드이름()

return 값;

→ 값이 부른 쪽으로 돌아간다

void

돌려줄 값 없음

double 과 int 를 돌려주는 메소드

예제 4-1

double

```
public double getArea() {  
    return 3.14*radius*radius;  
}
```

→ **314.0**

예제 4-2

int

```
int getArea() {  
    return width*height;  
}
```

→ **20**

리턴 타입이 결과 모양을 정한다.

메소드를 부른 자리가 값이 된다

```
double area = pizza.getArea();  
System.out.println(rect.getArea());
```

1. 첫 줄: `getArea()` 값을 `area` 변수에 담는다.
2. 둘째 줄: `println()` 안에서 값을 바로 쓴다.

첫 줄은 저장하고 둘째 줄은 바로 쓴다.



큰 판 하나, 작은 판 두 개



반지름 10cm 짜리 한 판을 살까, 반지름 7cm 짜리 두 판을 살까.

$$3.14 \times 10 \times 10 = 314.0, 3.14 \times 7 \times 7 \times 2 = 307.72.$$

넓이로 두 판 비교해 보기

Ex16_PizzaCompare.java

```
class Pizza {  
    int radius;  
    double getArea() {  
        return 3.14*radius*radius;  
    }  
}
```

두 판의 넓이 구하기

Ex16_PizzaCompare.java

```
public class Ex16_PizzaCompare {  
    public static void main(String[] args) {  
        Pizza big = new Pizza();  
        big.radius = 10;  
        Pizza small = new Pizza();  
        small.radius = 7;  
  
        System.out.println("큰 판 하나 " + big.getArea());  
        System.out.println("작은 판 두 개 " + small.getArea() * 2);  
    }  
}
```

넓이 비교 결과

큰 판 하나 314.0

작은 판 두 개 307.72

큰 판 하나가 작은 판 두 개보다 넓다.

메소드 안의 지역 변수

```
class Coupon {  
    int price;  
    int getDiscount() {  
        int d = price * 15 / 100;  
        return d;  
    }  
}  
  
public class Ex17_CouponDiscount {  
    public static void main(String[] args) {  
        Coupon c = new Coupon();  
        c.price = 20000;  
        System.out.println("할인 " + c.getDiscount() + "원");  
    }  
}
```

연비 메소드 채워 보기

Ex18_TripFuel.java

getKmPerLiter() 안에서 km 을 liters 로 나눈 값을 result 에 넣는다. 정수끼리 나누지 않는다.

```
class Trip {  
    int km;  
    int liters;  
    double getKmPerLiter() {  
        double result = 0;  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        return result;  
    }  
}
```

연비 출력하기

Ex18_TripFuel.java

```
public class Ex18_TripFuel {  
    public static void main(String[] args) {  
        Trip t = new Trip();  
        t.km = 150;  
        t.liters = 12;  
        System.out.println("1리터에 " + t.getKmPerLiter() + "km");  
    }  
}
```

연비 결과

1리터에 12.5km

150km 에 12리터면 1리터에 12.5km

점수 객체로 합계 내 보기

Ex19_ScoreSum.java · 할 일: Score 객체를 만들어 세 점수를 넣고, getSum() 으로 합계와 평균을 출력한다

```
class Score {  
    int kor, eng, math;  
    int getSum() {  
        return kor + eng + math;  
    }  
}  
  
public class Ex19_ScoreSum {  
    public static void main(String[] args) {  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

합계와 평균 결과

합계 270
평균 90.0

합계 270, 평균 90.0

나눗셈을 double 로 돌려주기

교수 시연

Ex20_ShareEach.java

```
class Share {  
    int pizzas, people;  
    double getEach() {  
        return (double) pizzas / people;  
    }  
}  
  
public class Ex20_ShareEach {  
    public static void main(String[] args) {  
        Share s = new Share();  
        s.pizzas = 10;  
        s.people = 4;  
        System.out.println("한 사람에 " + s.getEach() + "판");  
    }  
}
```

평균 오류 고쳐 보기

Ex21_GradeFix.java

85점과 90점의 평균은 87.5 인데 87.0 이 나온다. 한 곳을 고쳐 87.5 가 나오게 한다

```
class Grade {  
    int mid, fin;  
    double getAverage() {  
        return (mid + fin) / 2;  
    }  
}  
  
public class Ex21_GradeFix {  
    public static void main(String[] args) {  
        Grade g = new Grade();  
        g.mid = 85;  
        g.fin = 90;  
        System.out.println("평균 " + g.getAverage());  
    }  
}
```

고친 평균 결과

평균 87.5

2 를 2.0 으로 바꾸면 87.5

배열과 객체

배열

담는 것: 같은 타입 값 여럿

꺼내는 법: 번호 `a[0]`

만드는 법: `new int[5]`

객체

담는 것: 다른 타입 필드 여럿

꺼내는 법: 이름 `pizza.radius`

만드는 법: `new Circle()`

둘 다 `new` 로 만들고 레퍼런스로 가리킨다.

2교시에서 본 것

1

선언만 한 레퍼런스로는 필드를
쓸 수 없다

2

값을 안 넣은 필드는 0, false, null

3

레퍼런스를 대입하면 객체는 하나

4

메소드는 return 으로 값을 돌려준다

3교시

객체 배열

객체를 번호가 붙은 칸에 담는다.

레퍼런스 배열

원소마다 new

null 원소

정사각형이 다섯 개면

이름 다섯으로 구분

```
Square s0, s1, s2, s3, s4;
```

번호 하나로 구분

```
Square [] sq = new Square[5];
```

이름 다섯 대신 번호 하나

객체 배열

- 객체 배열은 객체에 대한 레퍼런스를 원소로 갖는 배열이다.
- 배열을 만든 뒤에도 원소마다 new 로 객체를 만들어야 한다.
- 자바의 객체 배열은 C/C++ 의 객체 배열과 만드는 방법이 다르다.



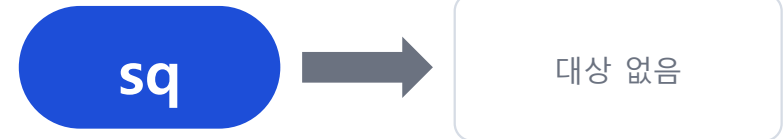
레퍼런스가 객체를 가리킨다. 객체 자체가 아니다.

1단계 배열 레퍼런스 선언

구문

```
Square [] sq;
```

지금 상태



sq 라는 레퍼런스 변수 이름만 만들어진다. 배열은 아직 생기지 않는다.

선언에 크기를 적으면

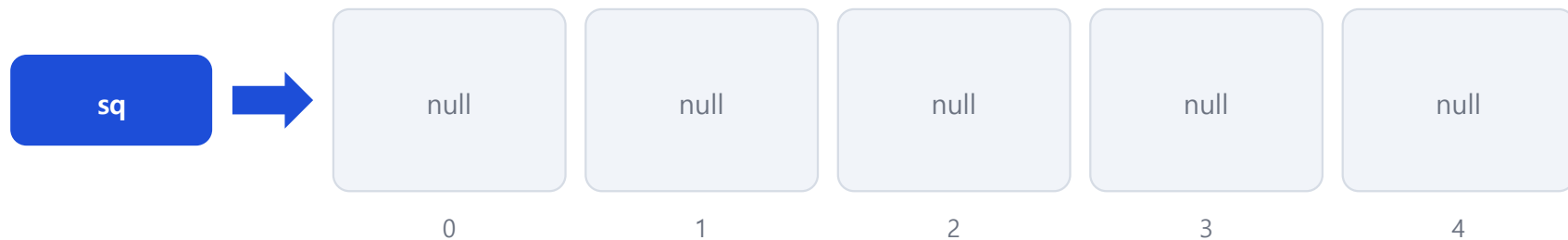


Square[5] sq;

2단계 레퍼런스 배열 생성

`new` 는 객체가 아니라 레퍼런스 칸을 만든다.

```
sq = new Square[5];
```



칸 다섯 개가 생기지만 안은 비어 있다.

지금 객체는 몇 개일까

```
Square [] sq;  
sq = new Square[5];
```

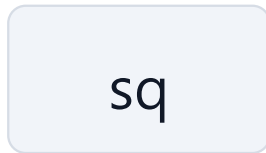
3단계 원소 객체 만들기

```
for(int i=0; i<sq.length; i++) {  
    sq[i] = new Square(); // i 번째 원소 객체 생성  
    sq[i].side = i+1; // 한 변의 길이를 i+1 로 설정  
}
```

배열의 크기만큼 객체를 만들어 하나씩 대입해야 비로소 객체 배열이 된다.

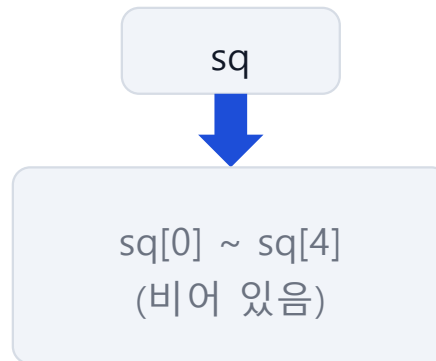
객체 배열이 만들어지는 과정

① Square[] sq;



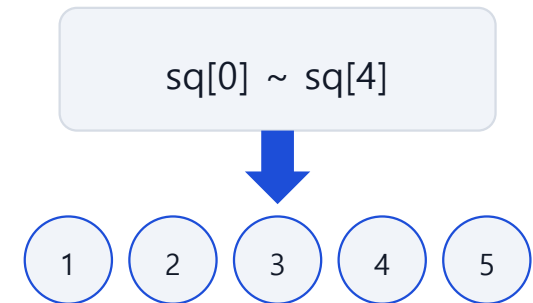
상자 하나

② sq = new Square[5];



빈 칸 다섯

③ 반복 두 줄



side=1~5 Square 객체 다섯

원소 객체 쓰기

sq[i]는 레퍼런스다

- i번째 객체를 가리킨다
- sq[i] 뒤에 점을 찍어 멤버에 간다
- 메소드도 점으로 부른다

배열 원소 접근

```
for(int i=0; i<sq.length; i++)  
    System.out.print(sq[i].getArea() + " ");
```

예제 4-6 Square 클래스

교수 시연

Ex22_SquareArray.java

```
class Square {  
    int side; // 한 변의 길이  
    int getArea() {  
        return side*side;  
    }  
}
```

예제 4-6 Square 배열 만들기

교수 시연

Ex22_SquareArray.java

```
public class Ex22_SquareArray {
    public static void main(String[] args) {
        Square [] sq; // Square 배열에 대한 레퍼런스 sq 선언
        sq = new Square[5]; // 5개의 레퍼런스 배열 생성

        for(int i=0; i<sq.length; i++) { // 배열의 개수 만큼
            sq[i] = new Square(); // i 번째 원소 객체 생성
            sq[i].side = i+1; // 한 변의 길이를 i+1 로 설정
        }

        for(int i=0; i<sq.length; i++) // 배열의 모든 원소 객체의 면적 출력
            System.out.print(sq[i].getArea() + " ");

    }
}
```

예제 4-6 실행 결과

1 4 9 16 25

한 번 1부터 5까지, 넓이 1 4 9 16 25

반복 안의 두 줄

1

```
sq[i] = new Square();
```

객체를 만든다

2

```
sq[i].side = i+1;
```

필드에 값을 넣는다

{ }

문장이 둘이면 중괄호가 필요하다.

원소마다 다른 객체

Ex23_KeyArray.java

```
class Key {  
    int no;  
}
```

```
public class Ex23_KeyArray {  
    public static void main(String[] args) {  
        Key [] ring = new Key[2];  
        ring[0] = new Key();  
        ring[0].no = 1;  
        ring[1] = new Key();  
        ring[1].no = 2;  
        System.out.println(ring[0].no + " " + ring[1].no);  
    }  
}
```

좌석 번호 예측해 보기

직접 실습

Ex24_SeatPredict.java · 두 값을 먼저 적은 뒤 실행해 확인한다

```
class Seat {  
    int number;  
}  
  
public class Ex24_SeatPredict {  
    public static void main(String[] args) {  
        Seat [] row = new Seat[2];  
        row[0] = new Seat();  
        row[0].number = 1;  
        row[1] = row[0];  
        row[1].number = 2;  
        System.out.println(row[0].number + " " + row[1].number);  
    }  
}
```

좌석 번호 결과

2 2

두 칸이 같은 객체를 가리킨다

과일 바구니 채워 보기

Ex25_FruitArray.java

```
class Fruit {
    String name;
    int price;
}

public class Ex25_FruitArray {
    public static void main(String[] args) {
        Fruit [] basket = new Fruit[3];
        // 여기서 부터 코드를 작성하세요.

        // 여기 까지
        for (int i = 0; i < basket.length; i++)
            System.out.println(basket[i].name + " " + basket[i].price + "원");
    }
}
```

과일 바구니 결과

사과 1000원
바나나 3000원
귤 500원

세 칸에 세 객체

원소 객체를 안 만들면

배열만 만들고 원소에 점을 찍어 본다.

```
Member [] team = new Member[3];  
team[0].name = "김자바";
```

컴파일 오류

또는

실행 오류

원소 없이 필드에 넣어 보기

Ex26_MemberNull.java

```
class Member {  
    String name;  
}  
  
public class Ex26_MemberNull {  
    public static void main(String[] args) {  
        Member [] team = new Member[3]; // 레퍼런스 배열만 생성  
        System.out.println("배열 크기는 " + team.length);  
        team[0].name = "김자바"; // 객체를 만들지 않고 필드에 접근  
        System.out.println(team[0].name);  
    }  
}
```

실행 오류 화면

배열 크기는 3

```
Exception in thread "main" java.lang.NullPointerException: Cannot assign field "name" because "<local1>[0]" is null
    at Ex26_MemberNull.main(Ex26_MemberNull.java:9)
```

실행이 오류로 끝났어요 (종료 코드 1)

원소는 null, 점을 찍으면 실행 중에 멈춘다



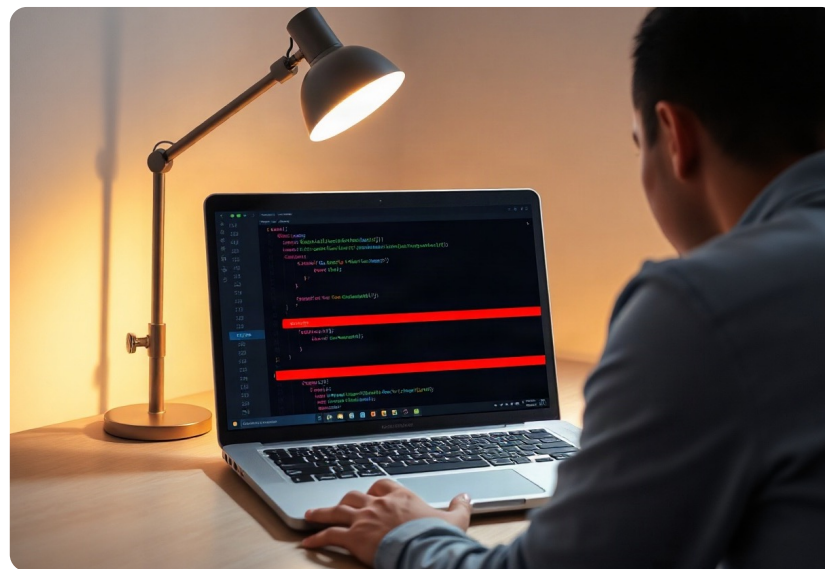
24년 동안 말하지 않던 오류



1996 → 2020 · JEP 358 → 자바 15

자바가 처음 나온 1996년부터 `NullPointerException` 은 무엇이 `null` 인지 말해 주지 않았다.

자바 15 부터는 따로 켜지 않아도 무엇을 하려다 멈췄는지를 말해 준다.



예외 줄에서 null 자리 찾기

java.lang.NullPointerException

← 예외 이름

Cannot assign field "name"

← 하려던 일

because "..." is null

← null 이었던 것

따옴표 안이 null 이었던 자리다.

실습실에서는 변수 이름 대신 <local1>[0] 처럼 번호로 나온다.

배열이 null, 원소가 null

4주차

```
int intArray[] = null;
```

배열 자체가 없다

오늘

```
Member [] team = new Member[3];
```

배열은 있고 원소가 null

둘 다 **NullPointerException**이다

예외 줄로 null 칸 찾기

Ex27_TagNull.java

```
class Tag {
    String text;
}

public class Ex27_TagNull {
    public static void main(String[] args) {
        Tag [] tags = new Tag[2];
        tags[0] = new Tag();
        tags[0].text = "A";
        tags[1].text = "B";
        System.out.println(tags[0].text + tags[1].text);
    }
}
```

노래 목록 오류 고쳐 보기

직접 실습

Ex28_SongFix.java

실행하면 NullPointerException 으로 멈춘다. 반복 안에 한 줄을 더해 노래 세 곡이 출력되게 한다.

```
class Song {
    String title;
}

public class Ex28_SongFix {
    public static void main(String[] args) {
        Song [] list = new Song[3];
        for (int i = 0; i < list.length; i++) {
            list[i].title = "노래" + (i + 1);
        }
        for (Song s : list)
            System.out.println(s.title);
    }
}
```

고친 노래 목록 결과

노래1
노래2
노래3

칸마다 new 를 넣었다

책 두 권 입력받기

제목과 저자 필드를 가진 Book 클래스로 2개짜리 배열을 만들고, 입력받은 제목과 저자로 채운다.

입력 형식

제목>>사랑의 기술
저자>>에리히 프롬

입력은 한 줄씩 Enter · 교재 p.158

예제 4-7 Book 클래스와 준비

Ex29_BookArray.java

1 / 2

```
import java.util.Scanner;
class Book {
    String title, author;
}

public class Ex29_BookArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Book [] book = new Book[2]; // Book 배열 선언
```

예제 4-7 입력받아 채우기

Ex29_BookArray.java

```
for(int i=0; i<book.length; i++) {
    System.out.print("제목>>");
    String title = scanner.nextLine();
    System.out.print("저자>>");
    String author = scanner.nextLine();
    book[i] = new Book(); // 배열 원소 객체 생성
    book[i].title = title; // 제목 필드에 값 넣기
    book[i].author = author; // 저자 필드에 값 넣기
}

for(int i=0; i<book.length; i++)
    System.out.print("(" + book[i].title + ", " + book[i].author + ")");
}
```

예제 4-7 실행 결과

제목>>사랑의 기술

저자>>에리히 프롬

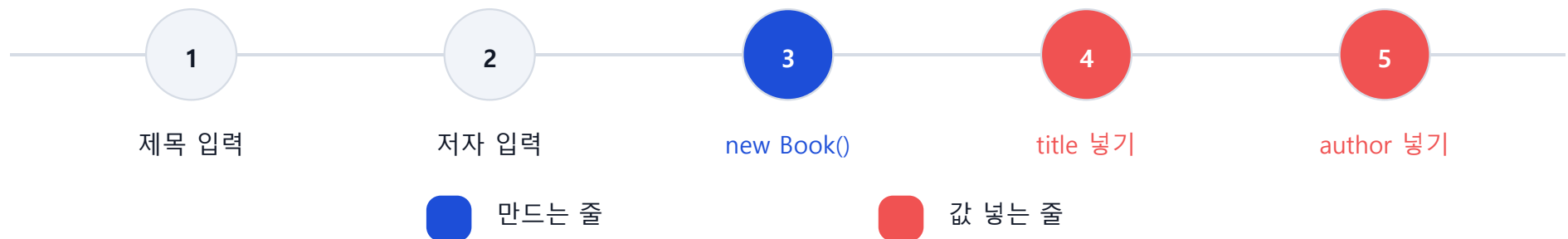
제목>>시간의 역사

저자>>스티븐 호킹

(사랑의 기술, 에리히 프롬)(시간의 역사, 스티븐 호킹)

두 권을 입력받아 괄호로 출력

입력을 받은 뒤에 객체를 만든다



객체를 만드는 줄과 값을 넣는 줄이 따로다.

객체 배열과 for-each

for 문

```
for(int i=0; i<book.length; i++)  
System.out.print(book[i].title);
```

for-each

```
for (Book b : book)  
System.out.print(b.title);
```

b 는 원소 객체를 차례로 가리키는 레퍼런스다.

직접 실습

영화 클래스와 두 편

Ex30_MovieForEach.java

```
class Movie {  
    String title;  
    int year;  
}
```

for-each 로 영화 목록 출력해 보기

Ex30_MovieForEach.java · for-each 로 두 영화의 제목과 연도를 한 줄씩 출력한다

```
public class Ex30_MovieForEach {  
    public static void main(String[] args) {  
        Movie [] list = new Movie[2];  
        list[0] = new Movie();  
        list[0].title = "인사이드 아웃";  
        list[0].year = 2015;  
        list[1] = new Movie();  
        list[1].title = "쿵푸 팬더";  
        list[1].year = 2008;  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

영화 제목 결과

인사이드 아웃 2015

쿵푸 팬더 2008

두 객체의 제목과 연도가 차례로

for-each 로 병 크기 읽기

```
class Bottle {  
    int ml;  
}  
  
public class Ex31_BottleList {  
    public static void main(String[] args) {  
        Bottle [] shelf = new Bottle[2];  
        shelf[0] = new Bottle();  
        shelf[0].ml = 300;  
        shelf[1] = new Bottle();  
        shelf[1].ml = 1000;  
        for (Bottle b : shelf)  
            System.out.println(b.ml + "ml");  
    }  
}
```

직접 실습

케이크 가격 합 구해 보기

Ex32_CakeTotal.java · for-each 로 상자 속 케이크 두 개의 가격을 total 에 더한다

```
class Cake {  
    int price;  
}
```

케이크 합계 채우기

Ex32_CakeTotal.java

```
public class Ex32_CakeTotal {  
    public static void main(String[] args) {  
        Cake [] box = new Cake[2];  
        box[0] = new Cake();  
        box[0].price = 4000;  
        box[1] = new Cake();  
        box[1].price = 3500;  
        int total = 0;  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
        System.out.println("합계 " + total + "원");  
    }  
}
```

케이크 합계 결과

합계 7500원

$4000 + 3500 = 7500$

문제: 옳은 객체 배열 코드

①

```
Square[5] sq;
```

②

```
Square [] sq = new Square[5];  
sq[0].side = 1;
```

③

```
Square [] sq = new Square[5];  
for (int i = 0; i < sq.length; i++)  
    sq[i] = new Square();
```

풀이: 셋의 차이

①

선언에 크기

컴파일 오류

②

원소 객체를 안 만들

실행 중 `NullPointerException`

③

원소마다 `new`

정답

원소마다 `new` 를 한 ③만 객체 배열이 된다.

종합 실습으로 할 일

오늘 배운 객체와 배열과 레퍼런스가 한 파일에 모인다.

- 1 Player 클래스(이름·점수)는 준다
- 2 세 명을 Player 배열에 담는다(김자바 85, 이객체 92, 박배열 78)
- 3 점수가 가장 높은 Player 를 찾는다
- 4 그 사람의 이름과 점수를 출력한다

우리 팀 최고점 찾아 보기

Ex33_TeamBest.java · 할 일은 앞 장 네 줄

```
class Player {  
    String name;  
    int score;  
}  
  
public class Ex33_TeamBest {  
    public static void main(String[] args) {  
        // 여기서 부터 코드를 작성하세요.  
  
        // 여기 까지  
    }  
}
```

종합 실습 결과

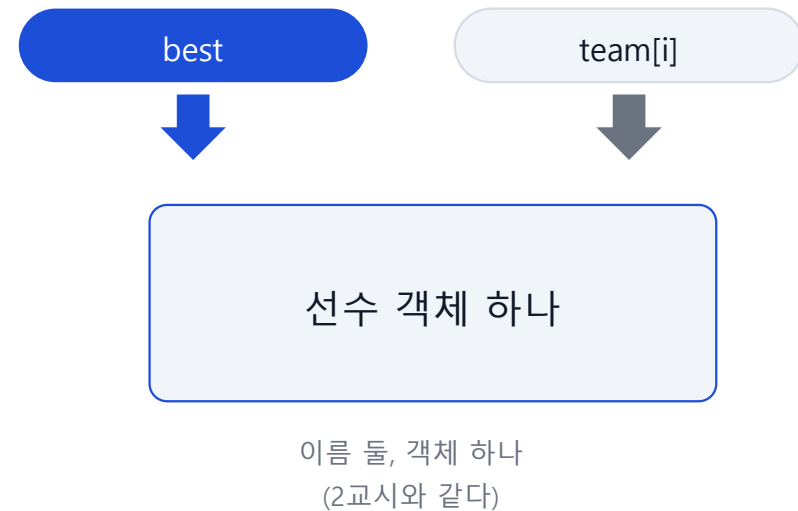
최고점은 이객체 92점

최고점은 이객체 92점

best 는 복사본이 아니다

```
Player best = team[0];  
best = team[i];
```

best 는 배열 안의 객체를
가리키기만 한다.



3교시에서 본 것



레퍼런스 배열

객체 배열의 원소는 레퍼런스다.



생성 직후 0개

배열을 만들어도 원소는 0개다.



원소마다 new

원소마다 new, 필드는 점으로.



null 원소

null 원소는 실행 중 멈춘다.

3교시 실습 살펴보기

- ④ 실습 ④: 반복 없이 세 칸을 쓰면 new 가 세 번 필요하다
- ⑤ 실습 ⑤: for-each 변수 m 의 타입은 Movie
- ⑥ 실습 ⑥: best 를 team[0] 으로 시작했다. 점수가 모두 음수여도 맞다

보강 실습

칸끼리 대입하면 같은 객체, 칸마다 new 를 해야 따로 생긴다. 예외 줄 따옴표 안이 null 인 칸이다(좌석 · 노래)

합을 담는 total 은 반복 밖에서 0 으로 시작한다(케이크)

내 책장 객체 배열

미션 조건

- 클래스: 제목·저자·쪽수 필드를 가진 Book
- 배열: 책 세 권을 담고 가장 두꺼운 책 출력
- 제출: 없음 (자율 학습)

혼자 해 볼 때 볼 세 가지

원소마다 new 로 객체를 만들었나

필드는 점(.)으로 넣었나

가장 두꺼운 책을 Book 레퍼런스로 들고 있나

안 될 때 여기부터

틀림

- 선언만 한 레퍼런스에 점
- 클래스에 없는 필드 이름
- 배열만 만들고 원소에 점

맞음

- ✓ new 로 객체를 먼저 만든다
- ✓ 클래스의 필드 이름을 확인한다
- ✓ 원소마다 new 를 한다

예외 줄은 because 뒤 따옴표 안부터 읽는다

오늘 공부한 내용

객체는 특성과 행동을 가진다. 특성은 필드, 행동은 메소드

클래스는 틀, 객체는 그 틀로 new 해서 만든 실체

레퍼런스 변수를 선언해도 객체는 생기지 않는다

점(.)으로 필드에 값을 넣고 메소드를 부른다

객체 배열은 레퍼런스 배열, 원소마다 new

다음 주엔 생성자부터

6주차도 강의실에서 대면으로 수업합니다



다음 주 예고

- 생성자와 this
- 메소드 오버로딩



반드시 지참

- 노트북과 충전기
- 교재 명품 자바 에센셜