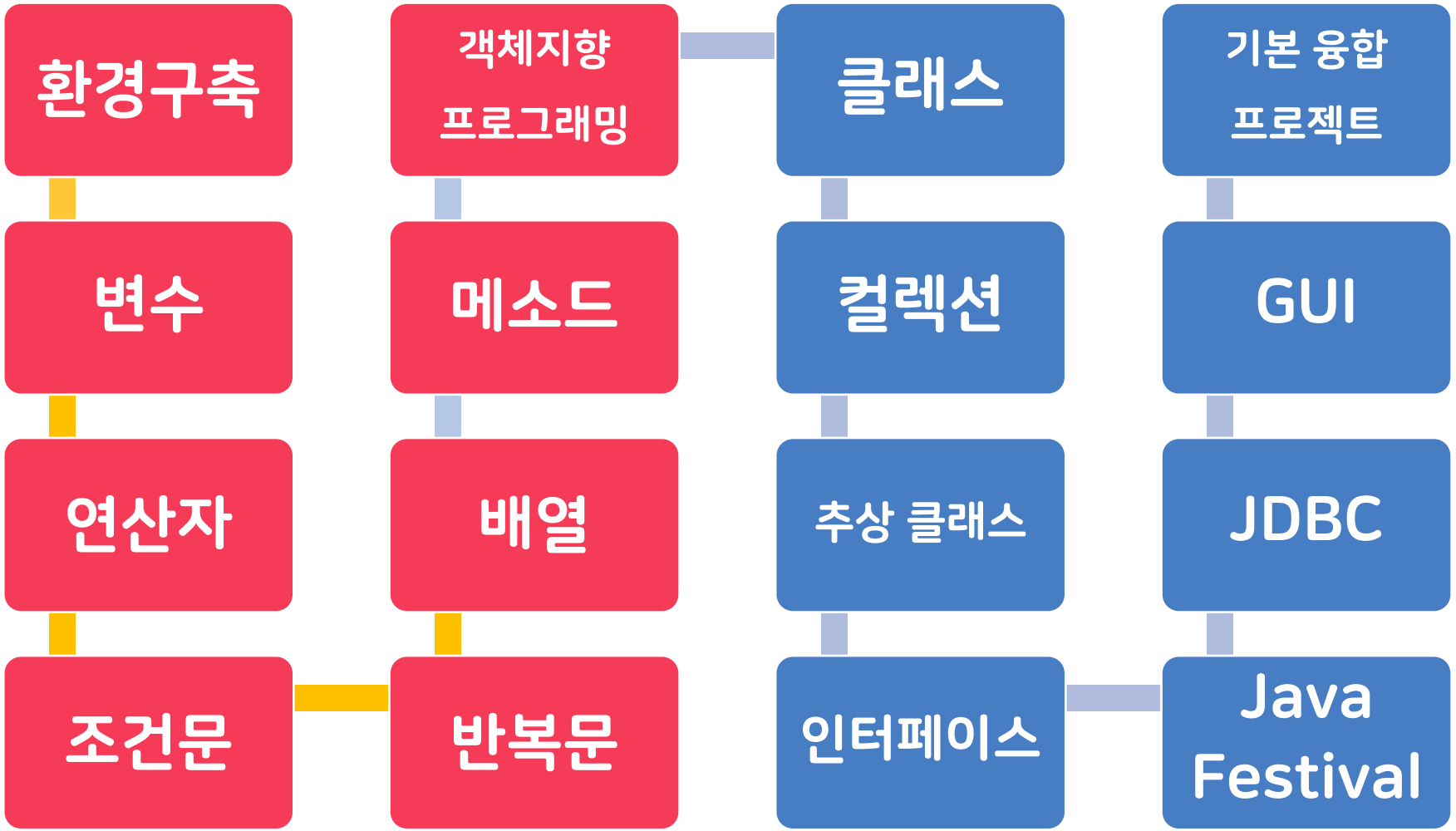
스마트인재개발원
Smart Human Resources Development

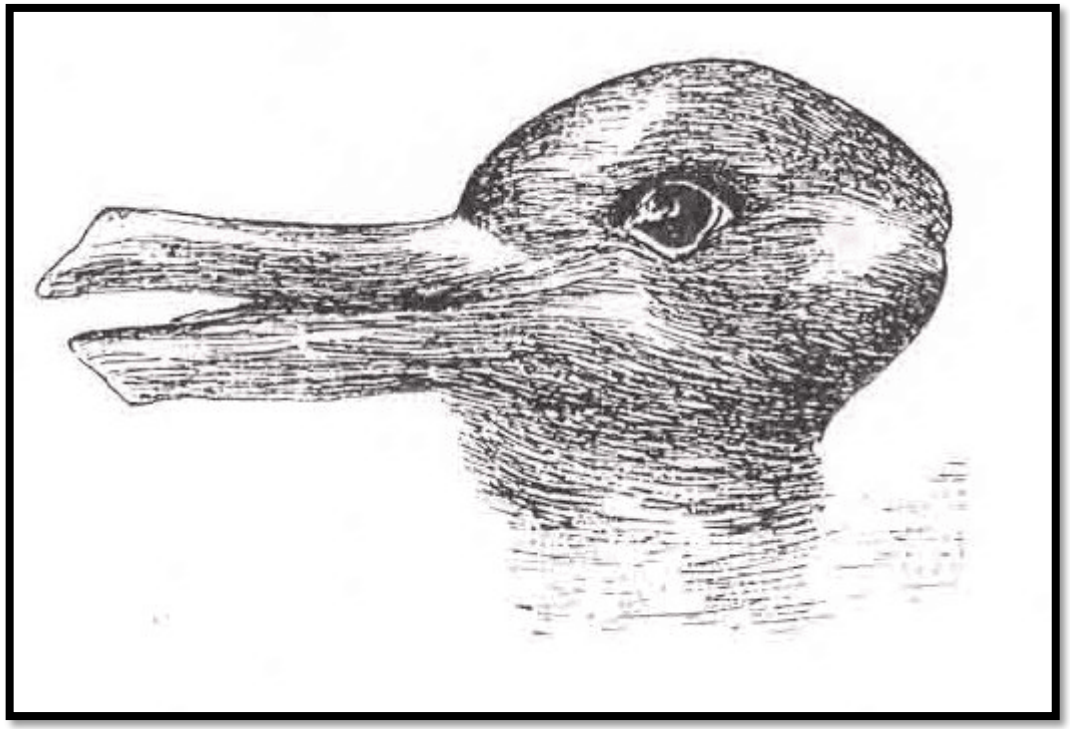
박 수 현 연구원

수업 진행방향



스마트인재개발원
Smart Human Resources Development

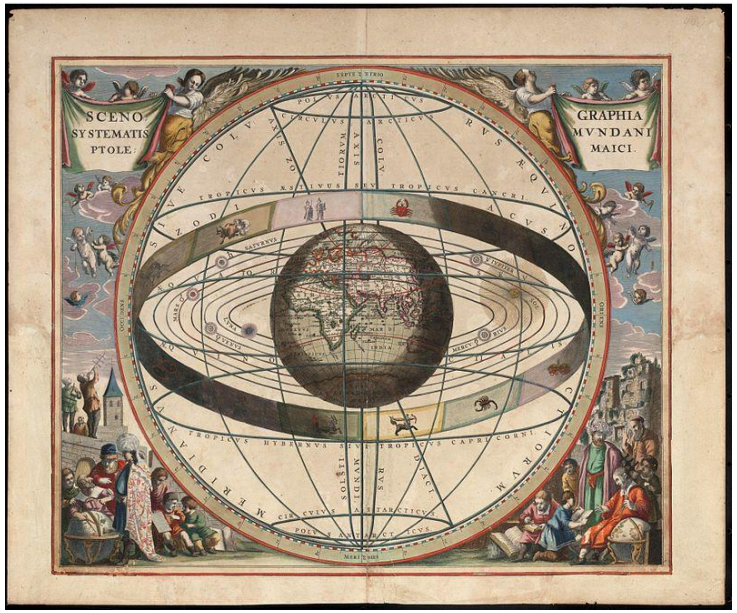
Paradigm



Paradigm

사람들의 견해나 사고를 근본적으로 규정하는 테두리

Paradigm



Programming Paradigm

프로그래밍을 할 때 가지는 체계, 관점, 틀

Java Procedural Paradigm

스마트인재개발원 Smart Human Resources Development

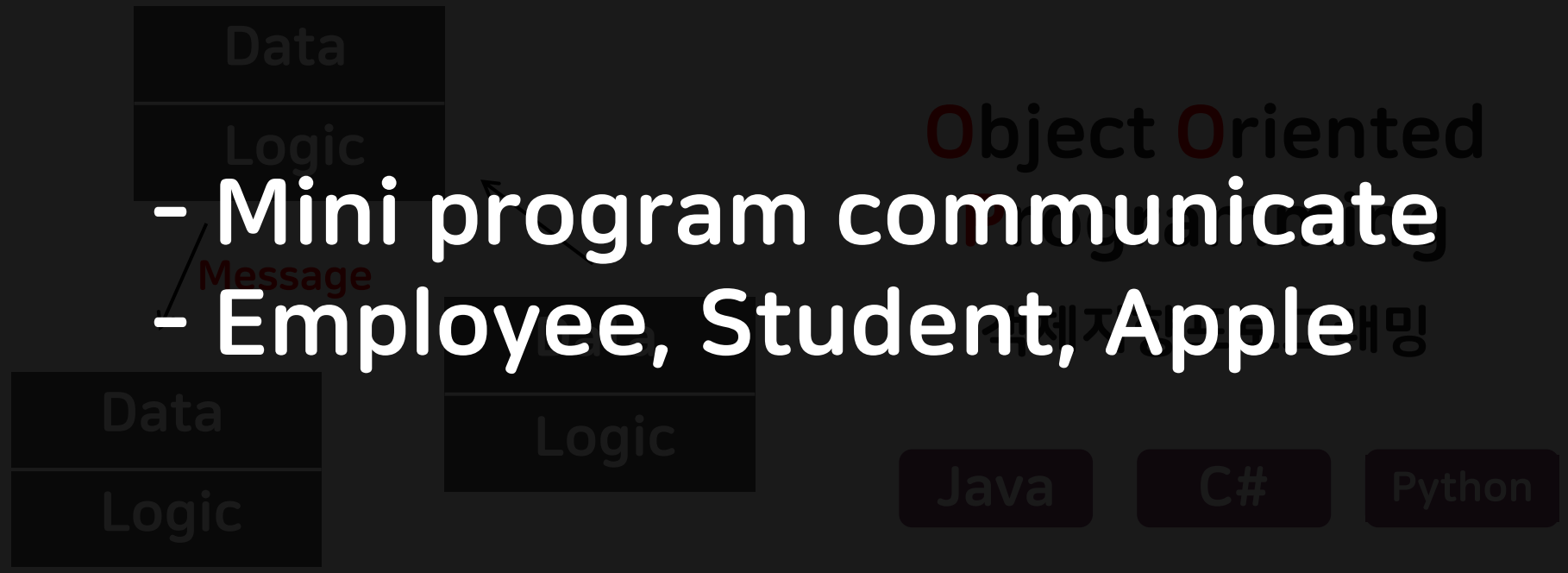
```
System.out.println("==Music Play List");
System.out.println();
while(true) {
System.out.print("[1] 노래추가 [2] 노래조회 [3] 노래삭제 [4]
종료 >> ");
int num = sc.nextInt();

//노래추가
if(num==1) {
System.out.println("==================================");
System.out.print("[1] 원하는 위치에 추가 [2] 마지막 위치에 추
가 >> ");
num = sc.nextInt();

//노래조회
else if(num ==2) {
if(musicList.size()==0) {
System.out.println("재생목록이 없습니다.");
}else {
for(int i=0; i<musicList.size(); i++) {
System.out.println(i+1+". "+musicList.get(i));
}
}
}
//노래삭제
else if(num ==3) {
System.out.println("========================");
System.out.print("[1] 선택 삭제 [2] 전체 삭제 : ");
num = sc.nextInt();
if(num ==1) {
System.out.print("삭제 할 번호를 입력하세요 : ");
index = sc.nextInt();
musicList.remove(index-1);
System.out.println("삭제되었습니다. ");
}else if(num ==2) {
```

- Step by Step (Process)
- Data and Logic Mixed

Procedural Programming

절차적 프로그래밍

C fortran COBOL



- Mini program communicate
- Employee, Student, Apple

객체 지향 프로그래밍

(Object Oriented Programming : OOP)

여러 개의 독립된 단위, 즉, **객체들의 모임** 으로 파악하고자 하는 것

각각의 객체는 **메시지** 를 주고 받고 데이터를 처리

자동차 바퀴
로고
자동차 문
엔진
사이드미러
부품 7
유리
부품 8

컴퓨터가 수행하는 작업을 객체들간의 상호 작용으로 표현

객체들의 집합으로 프로그램 작성

Class Object

Class(클래스)

객체를 정의해 놓은 것
객체를 정의하는 틀 또는 설계도

Object(객체)

클래스의 인스턴스
설계도를 통해 만들어진 것

Class(클래스)

Field

- 해당 클래스 내에 정의된 변수를 의미
- 정의된 변수는 객체의 특성을 나타낸다

Method

- 객체가 행해야 하는 작업을 기술한 것
- 객체의 기능

자동차의 Class

객체의 특성

Field

- 모델명 : G바겐
- 모델연도 : 2022
- 모델색깔 : 검정색

객체의 기능

Method

- 엑셀( );
- 브레이크( );

instance

메모리에 할당된 객체

- 내 차 : 설계도에 의해 생산된 차량
- 친구 차 : 설계도에 의해 생산된 또 다른 차량

추상화(Abstract)

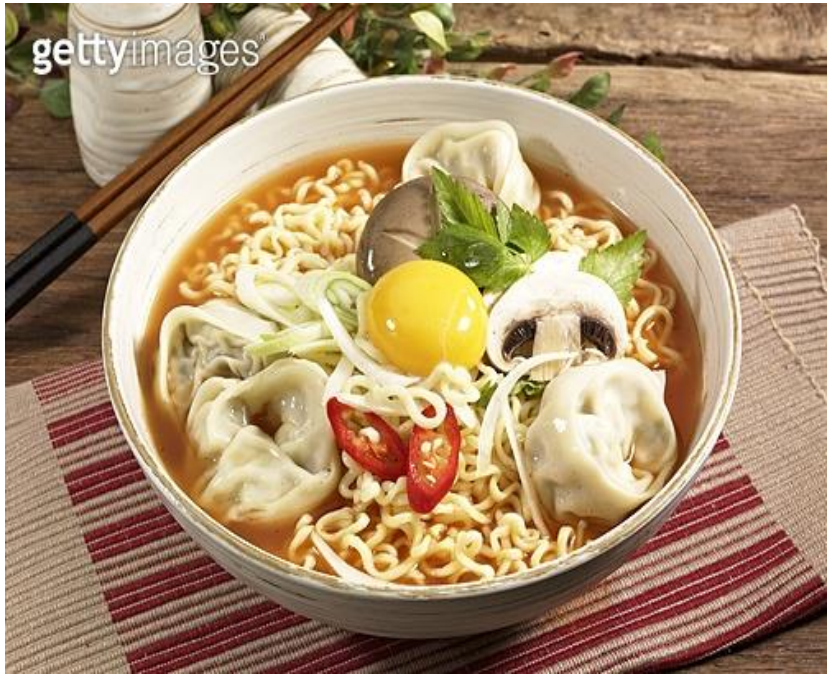
캡슐화(Encapsulation)

캡 상 추 다

상속(Inheritance)

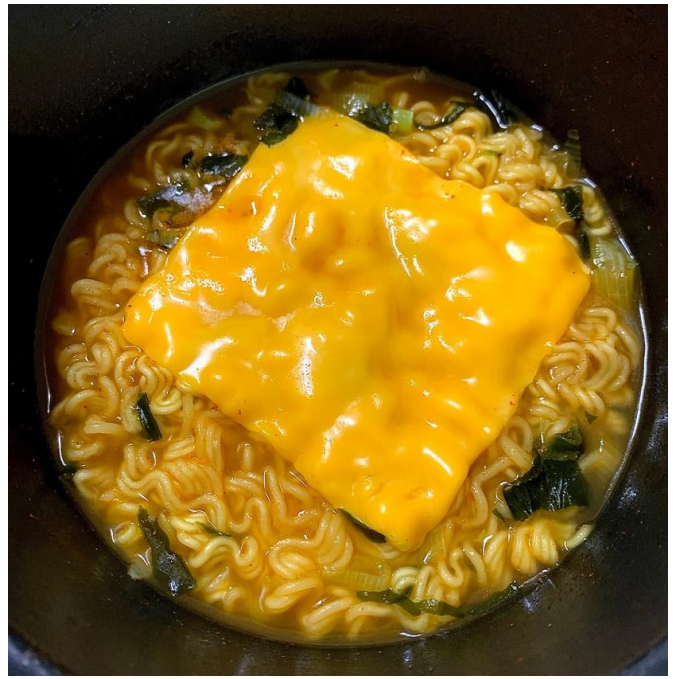
다형성(Polymorphism)

객체지향프로그래밍의 특징 - 추상화



만두라면

떡라면

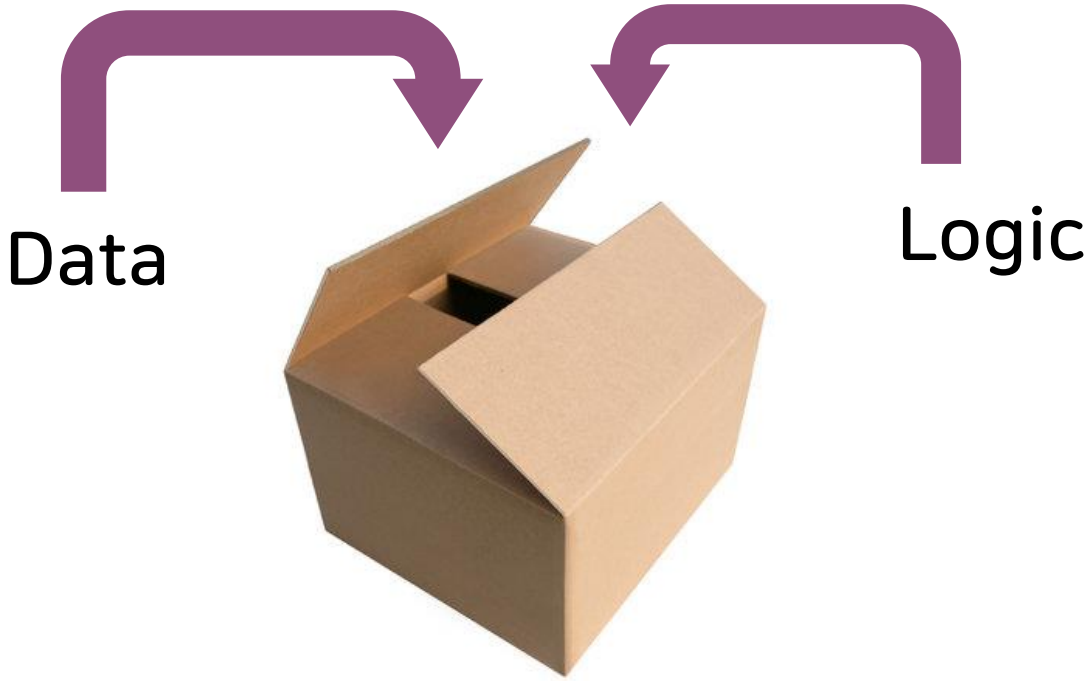


치즈라면

추상화(Abstraction)

- 객체에서 공통된 속성(필드)과 행위를 추출 하는 기법
- 코드 상에서 구현(로직)부분을 제외한 오직 선언 부분만을 설계
- 상세한 정보는 무시하고 필요한 정보들만 간추려서 구성

객체지향프로그래밍의 특징 - 캡슐화



캡슐화(Encapsulation)

- 관련된 필드(속성)와 메소드(기능)를 하나로 묶고, 실제 구현 내용을 외부로부터 감추는 기법(정보은닉)

- 만일의 상황(타인이 외부에서 조작)을 대비해서 특정 속성이나 메소드를 사용자가 조작할 수 없도록 숨겨 놓은 것.

- 외부에서는 공개된 메소드(기능)의 인터페이스를 통해 접근할 수 있다.

객체지향프로그래밍의 특징 - 캡슐화

개발자

내부 보지마!
내부 열면 AS 안해준다?

사용자

내부에 흥미 없어!
Tv 프로가 보이면 OK~

음량조절이나 채널변경
같은 처리만 공개

인터페이스

공개된 처리만 사용

객체지향프로그래밍의 특징 - 캡슐화

Scanner

next()
nextLine()
nextInt()
nextFloat()

Random

nextInt()

Arrays

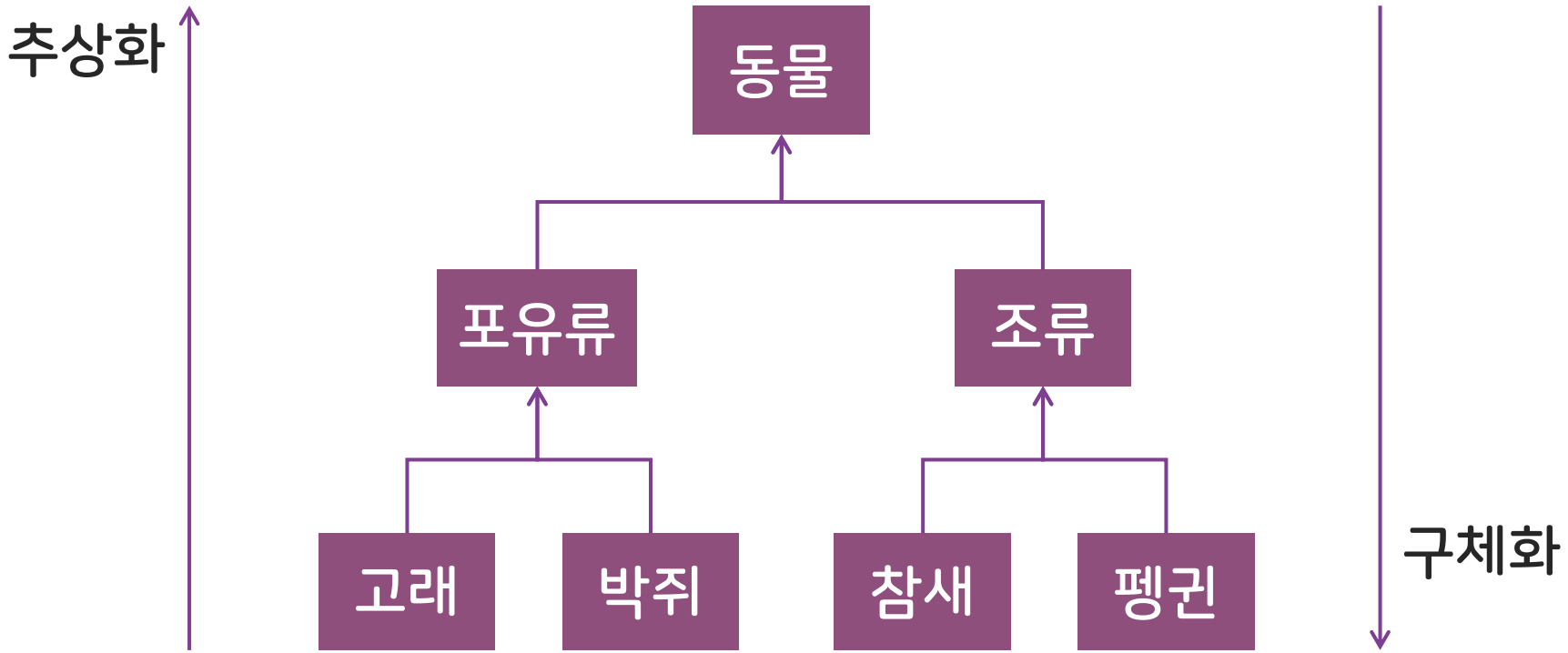
sort()
binarySearch()

Math

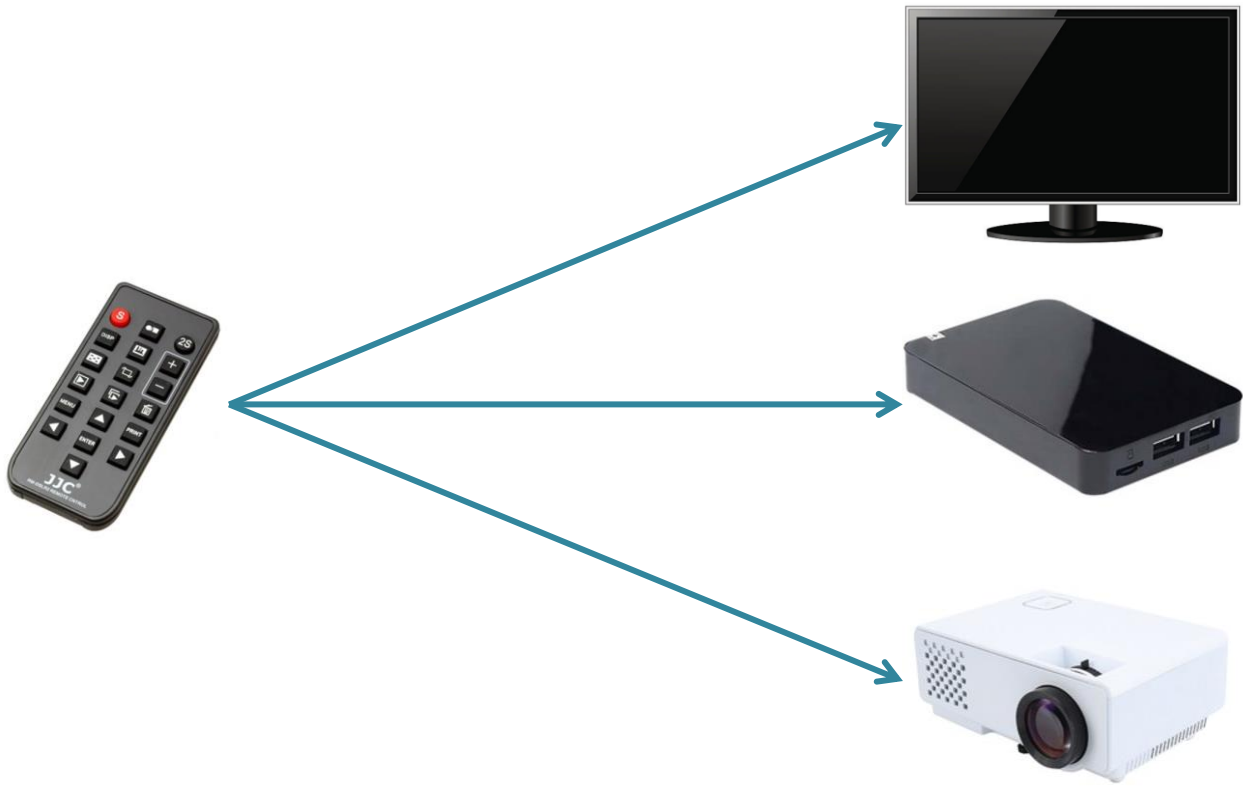
pow()
round()
cos()
min()
max()

상속(Inheritance)

- 이미 작성된 클래스(상위클래스)의 특성을 그대로 이어받아 새로운 클래스 (하위클래스)를 생성하는 기법
- 기존 코드를 그대로 재사용하거나 재정의 -> 재사용 + 확장



객체지향프로그래밍의 특징 - 다형성



다형성(Polymorphism)

- 사전적 의미 '다양한 형태로 나타날 수 있는 능력'
- 같은 기능(메소드)를 호출하더라도 객체에 따라 다르게 동작하는 것
- 상위클래스의 동작을 하위클래스에서 다시 정의하여 사용 하는 것 또한 다형성으로 볼 수 있다.

Overriding(오버라이딩)

객체지향프로그래밍의 장점

- 신뢰성 있는 소프트웨어를 쉽게 작성할 수 있다.
- 코드를 재사용하기 쉽다.
- 유지보수가 용이하다.
- 직관적인 코드 분석이 가능하다.
- 소프트웨어 생산성이 향상된다.

클래스(Class)의 구조

public class 클래스명{

자료형 필드명1;
자료형 필드명2;
… …

→ 객체의 속성 정의

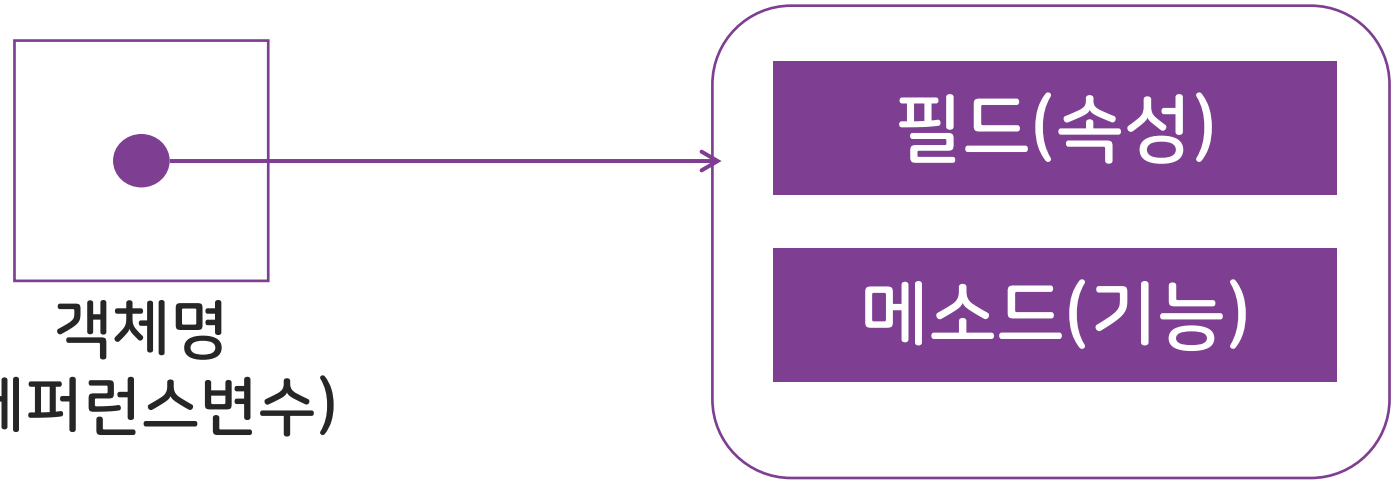
반환형 메소드1(){…}
반환형 메소드2(){…}
…

→ 객체의 기능(행위) 정의

}

객체(Object)의 생성

```
public class 클래스명{
    public static void main(String[] args){
        클래스 객체명 = new 클래스();
    }
```



나를 만들어보자!

실습 – 저금통(PiggyBank)

저금통 클래스
속성(필드)
금액(money)
기능(메소드)
돈을 넣는다(deposit) 돈을 인출한다(withdraw) 잔액을 보여준다(showMoney)

실습 – 저금통(PiggyBank)

1. 내 재산(money)를 필드에 정의하세요.

2. 입금할 수 있는 기능을 정의하세요

3. 출금할 수 있는 기능을 정의 하세요

4. 잔액을 출력하는 기능을 정의하세요

클래스와 객체 실습 – 저금통(PiggyBank)

1. 저금통을 실행할 수 있는 Main메소드가 있는 클래스를 만든 후 저금통에 1500원을 입금하세요.

2. 현재 잔액을 출력하세요.

3. 저금통에서 500원을 인출 후 잔액을 출력하세요.

실습 – 학생정보 관리 시스템

학생의 정보를 담을 수 있는 Student클래스를 작성하세요.
Student클래스는 다음과 같은 필드를 갖습니다.

자료형태	변수 이름	설명
String	name	이름
String	birth	생일
int	age	나이
int	scoreJava	Java 점수
int	scoreWeb	Web 점수
int	scoreAndroid	Android 점수

실습 – 학생정보 관리 시스템

Main클래스에서 각각 student1, student2객체를 생성하고
다음과 같이 초기화하세요.

student1		student2	
변수 이름	학생 정보	변수 이름	학생 정보
name	본인이름	name	짝궁이름
birth	990126	birth	900416
age	25	age	33
scoreJava	50	scoreJava	90
scoreWeb	89	scoreWeb	25
scoreAndroid	77	scoreAndroid	30

실습 – 학생정보 관리 시스템

초기화한 학생의 정보를 화면에 출력하는 show()메소드를
Student 클래스 안에 작성하고 호출하여 학생의 정보를 출력하세요.

```
박수현님 안녕하세요.
[990126, 25살]
박수현님의 Java점수는 100입니다.
박수현님의 Web점수는 90입니다.
박수현님의 Android점수는 80입니다.
평균 점수는 90입니다.
-----------------------
이주희님 안녕하세요.
[900416, 33살]
이주희님의 Java점수는 100입니다.
이주희님의 Web점수는 100입니다.
이주희님의 Android점수는 100입니다.
평균 점수는 100입니다.
```

this

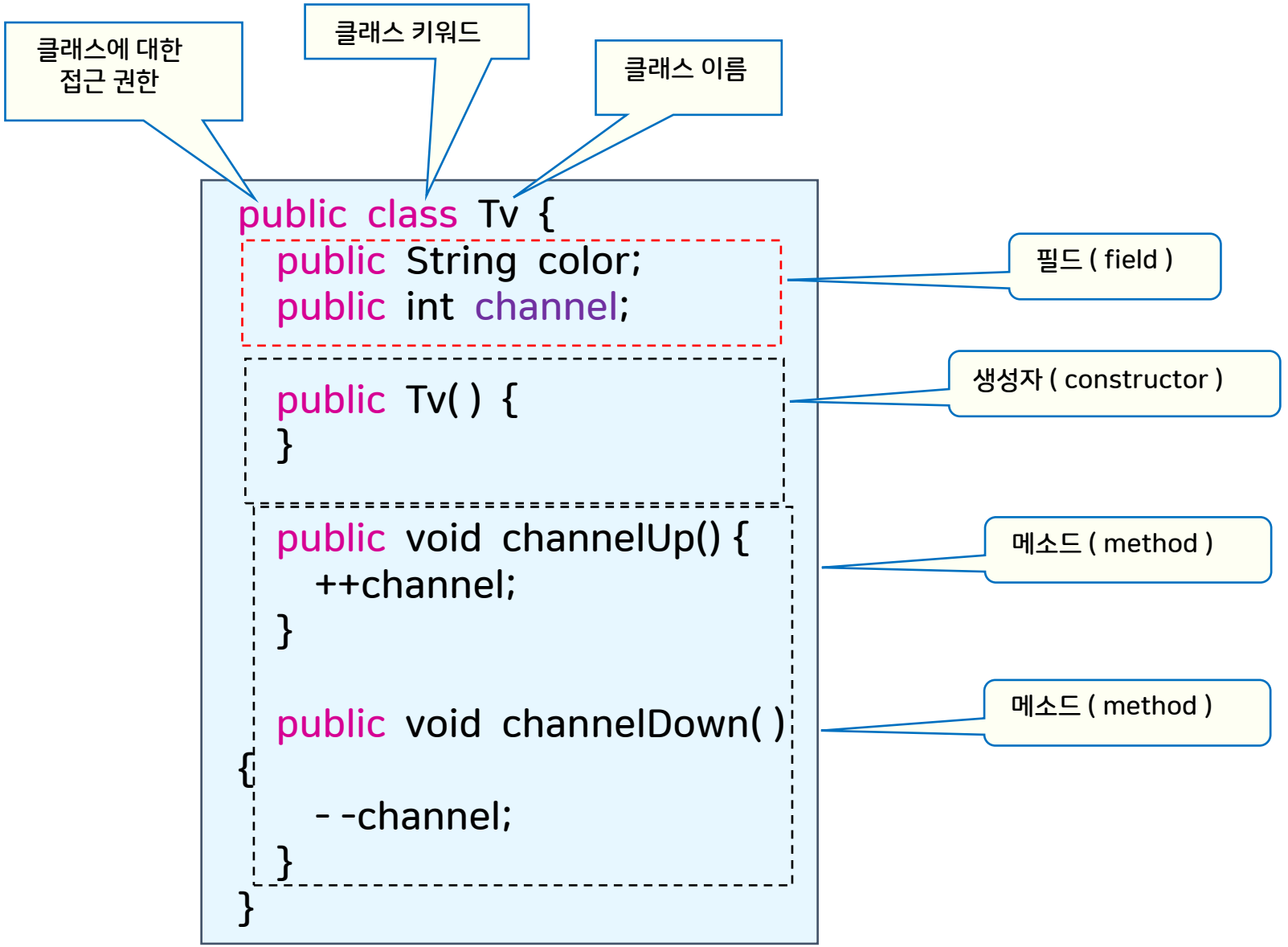
자기 자신의 객체

= 현재 실행되는 메소드가 속한 객체

생성자의 특징

- 생성자는 메소드이다
- 생성자 이름은 클래스 이름과 동일
- 생성자는 리턴 타입을 지정할 수 없다.
- 생성자는 new를 통해 객체를 생성할 때만 호출됨
- 생성자는 하나 이상 선언되어야 함
  - ✓ 개발자가 생성자를 정의하지 않으면 자동으로 기본 생성자가 정의됨
    - 컴파일러에 의해 자동 생성
    - 기본 생성자를 디폴트 생성자(default constructor)라고도 함
    - 만약 default 생성자만 존재하고 default에 아무런 기능이 없을 경우 생략이 가능함

클래스 구조


클래스에 대한
접근 권한
클래스 키워드
클래스 이름
public class Tv {
public String color;
public int channel;
public Tv( ) {
}
public void channelUp() {
++channel;
}
public void channelDown( )
{
- -channel;
}
}
필드 (field)
생성자 (constructor)
메소드 (method)
메소드 (method)

실습 - 포켓몬!

다음시간에는?

상속, 추상클래스