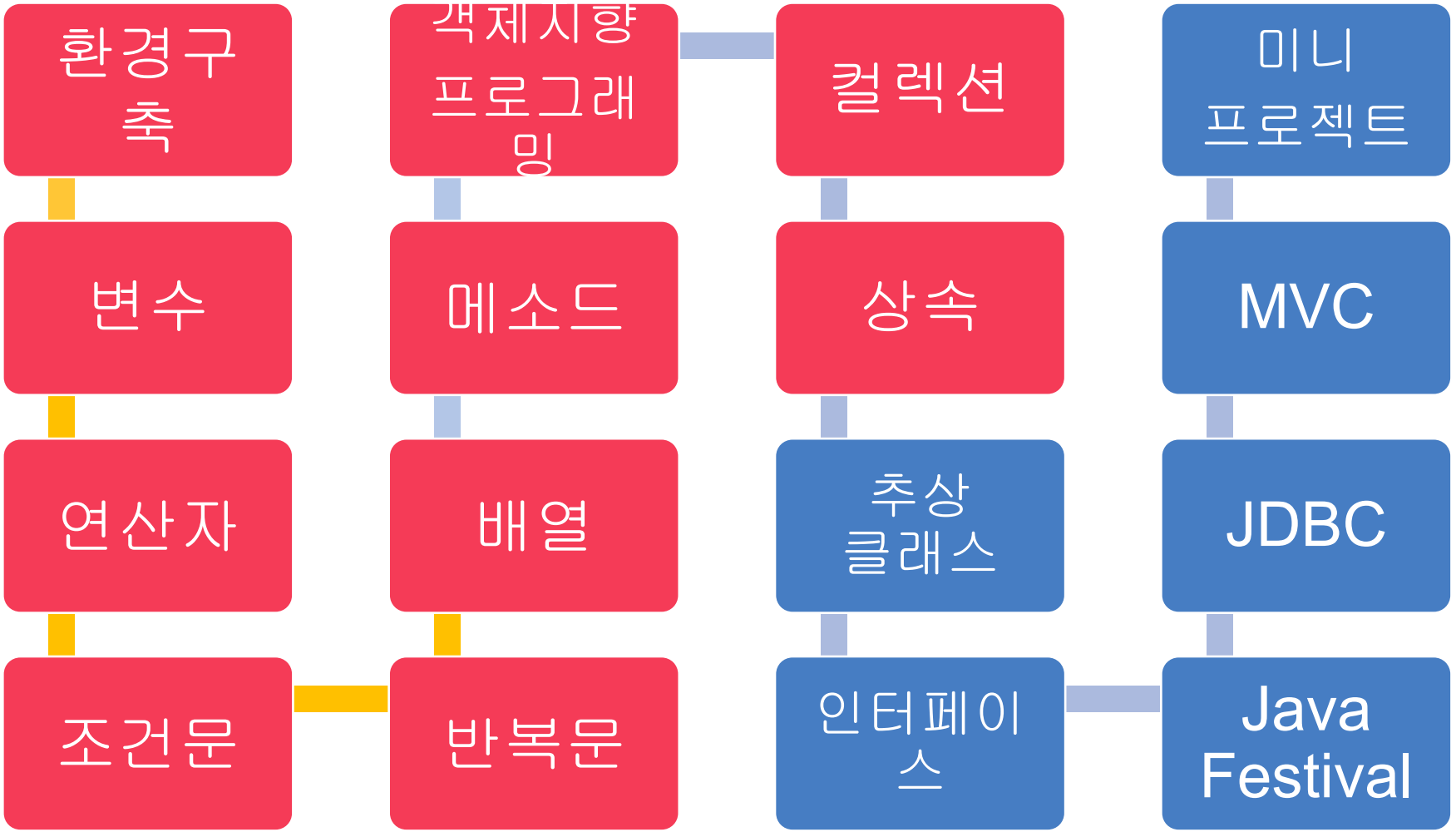
스마트인재개발원
Smart Human Resources Development

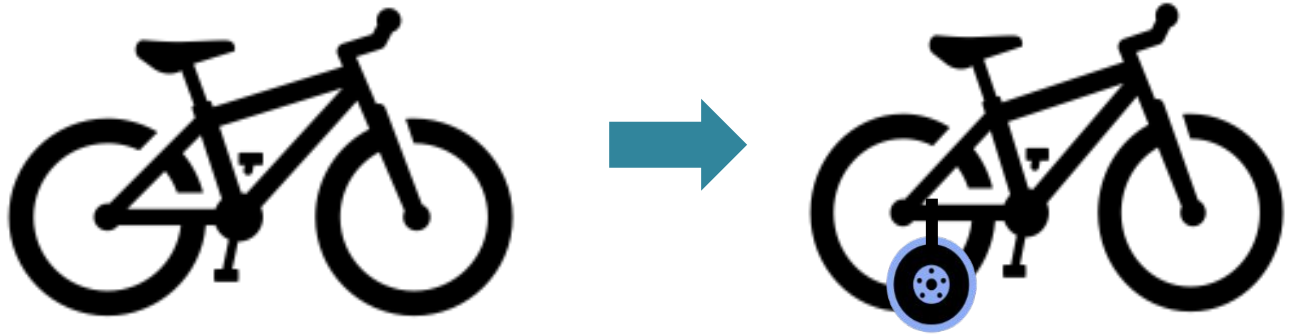
박 수 현
연구원

수업 진행방향


환경구축
변수
연산자
조건문
객체지향 프로그래밍
메소드
배열
반복문
컬렉션
상속
추상 클래스
인터페이스
미니 프로젝트
MVC
JDBC
Java Festival

상속 : 뒤를 잇다, 이어받다, 물려받다

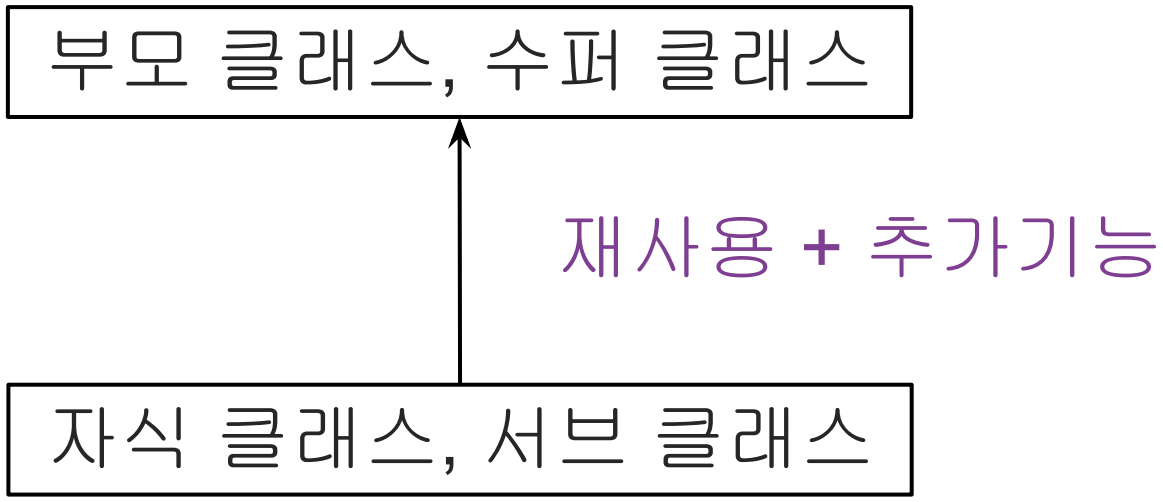
ex) 네 발 자전거를 만들고 싶다.



ex) 휠 마우스를 만들고 싶다.

Java의 상속

: 기존 클래스의 변수 (데이터)와 메소드(로직, 코드)를 물려받아 새로운 클래스를 구성하는 것.



상속의 필요성

class
마우스

좌클릭하기
메소드
우클릭하기
메소드
드래그하기
메소드

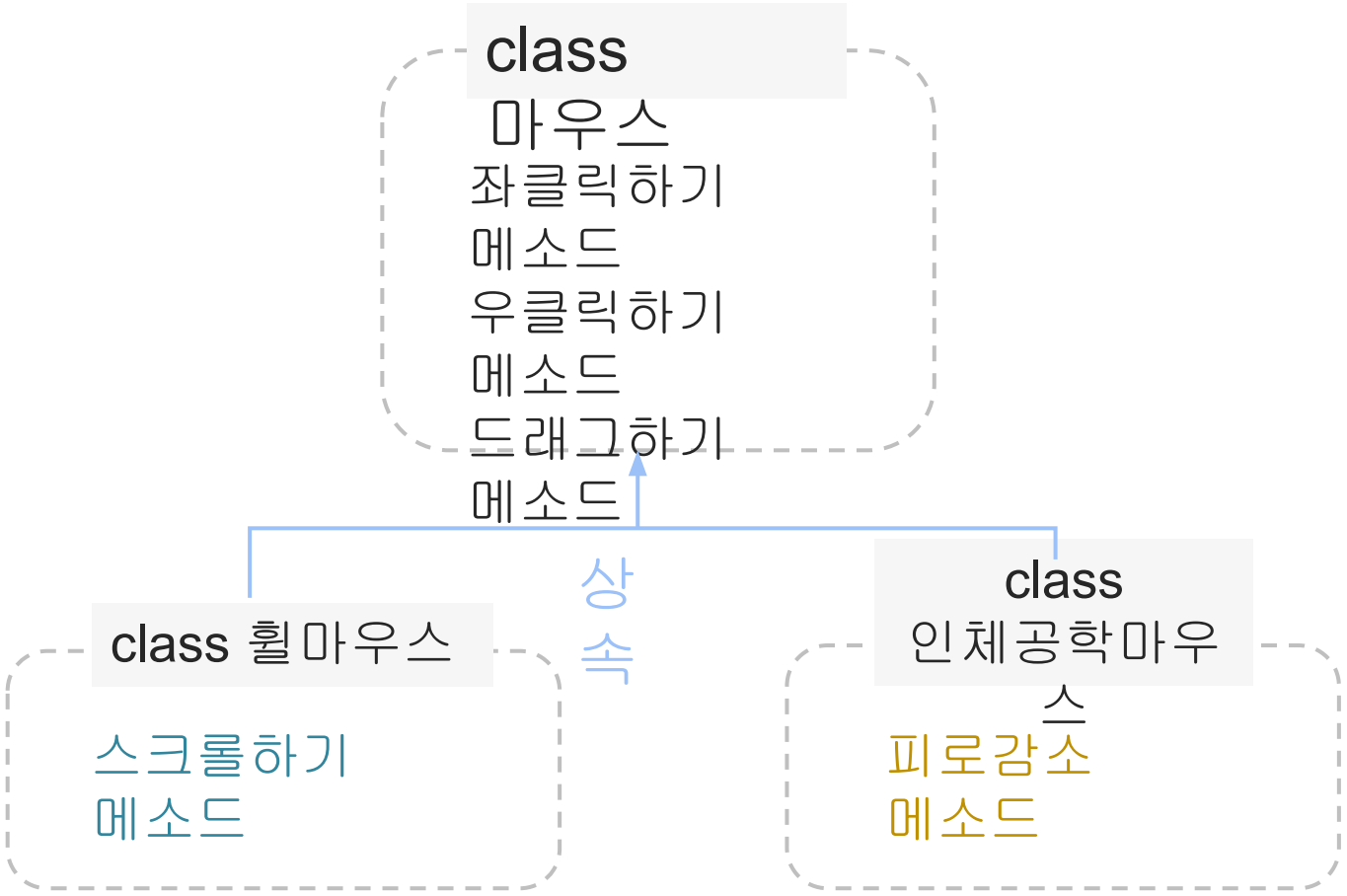
class 휠마우스

~~좌클릭하기~~
~~메소드~~
~~우클릭하기~~
스크롤하기
메소드
드래그하기
메소드

class
인체공학마우스

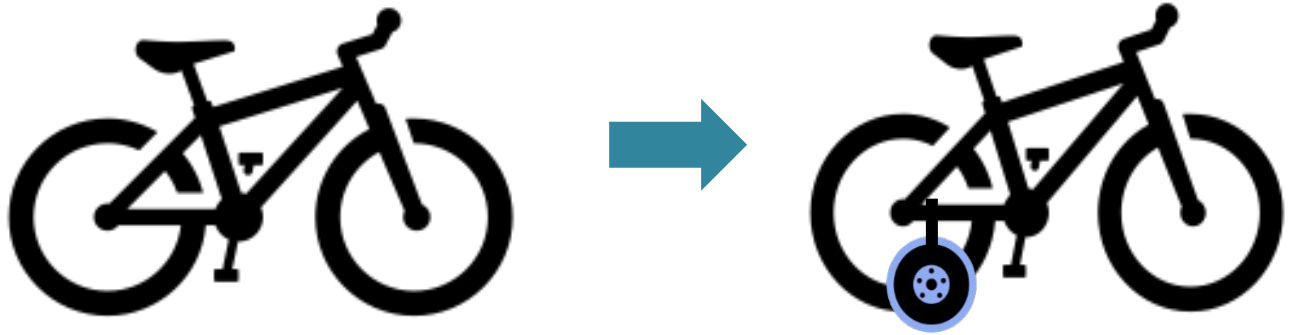
~~좌클릭하기~~
~~메소드~~
~~우클릭하기~~
페이지업소
메소드
드래그하기
메소드

상속의 필요성


class
마우스
좌클릭하기
메소드
우클릭하기
메소드
드래그하기
메소드
상
속
class 휠마우스
스크롤하기
메소드
class
인체공학마우
스
피로감소
메소드

상속 : 뒤를 잇다, 이어받다, 물려받다

ex) 네 발 자전거를 만들고 싶다.



ex) 휠 마우스를 만들고 싶다.

상속의 필요성

상속을 사용해서 얻는 장점

-기존 클래스의 변수와 코드를 재사용

→ 코드의 중복 감소

클래스 간결화

→ 신뢰성 있는 프로그램

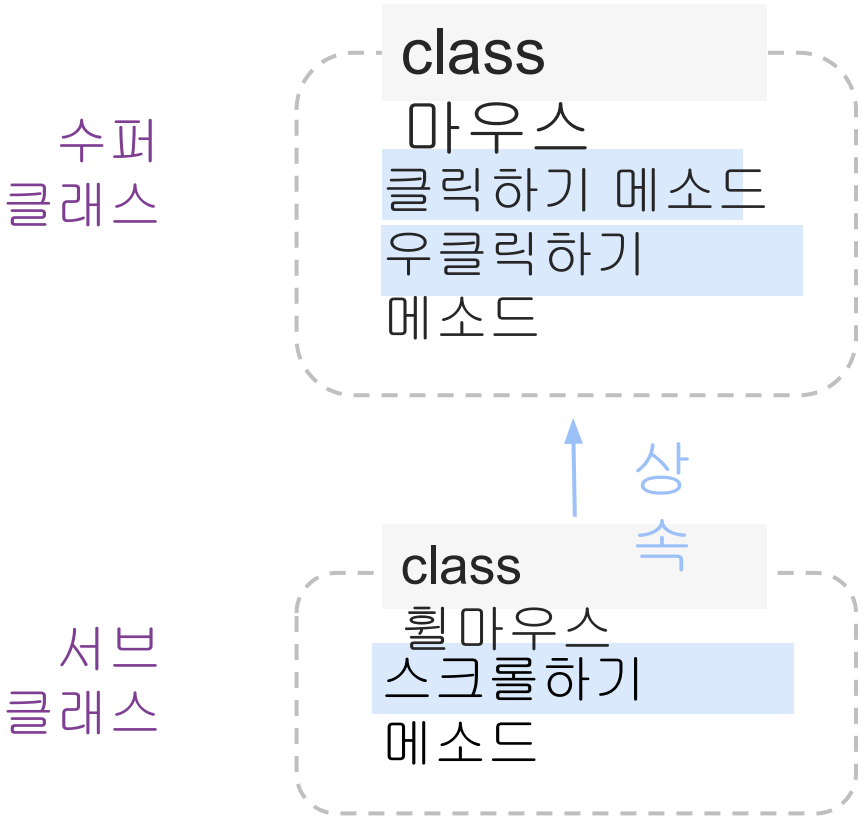
-먼저 작성된 검증된 프로그램을 재사용

손쉽게 개발

→ 유지보수 용이

-클래스간 계층적 분류 및 관리

상속 문법의 이해



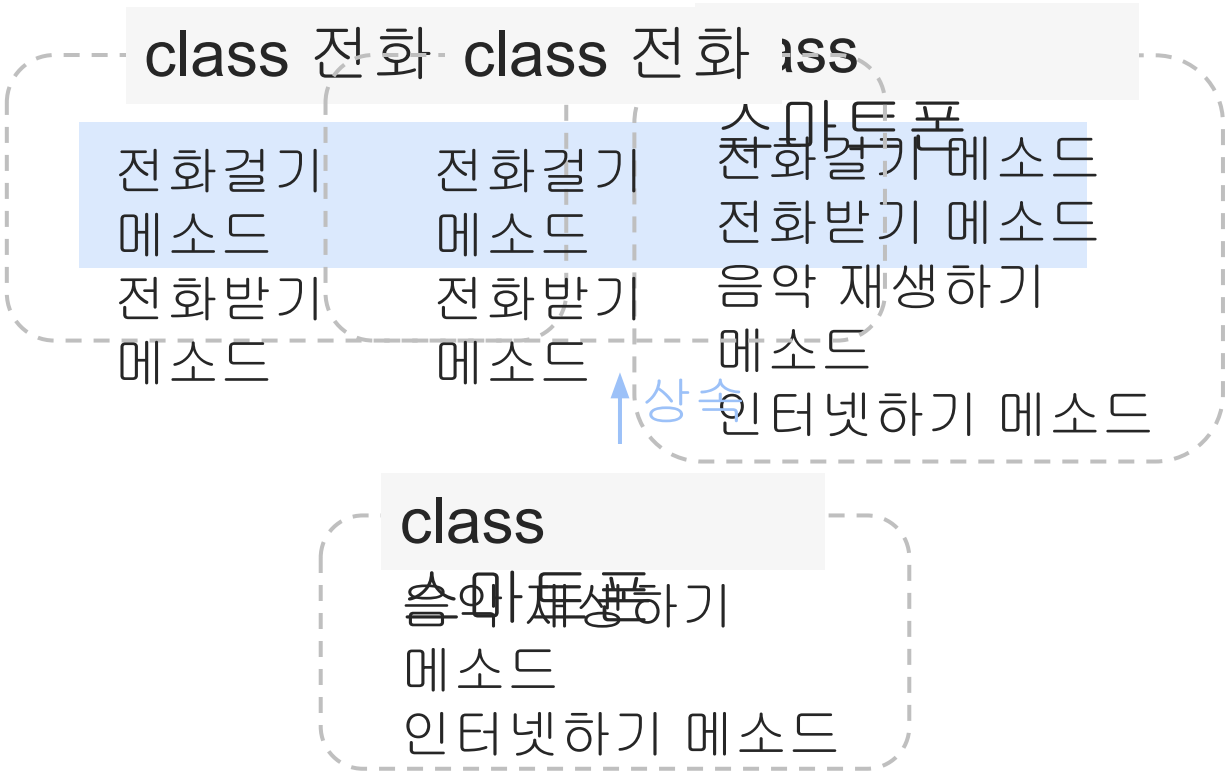

```
public class Mouse {

    public void click(){
      System.out.println("클릭하기");
    }

    public void rightClick(){
      System.out.println("우클릭하기");
    }

}
```

마우스를 상속받는 휠마우스 클래스 선언

: class 서브클래스 extends 수퍼클래스

```
public class WheelMouse extends Mouse
{

    public void scroll(){
      System.out.println("스크롤하기");
    }

}
```

상속 예제 ① :
상속을 이용하여 다음 클래스들을 간결하게 재구성하라



자바 상속의 특징 다중상속을 지원하지 않는다.

1.

```
public class Telephone {

    public void call(){
      System.out.println("전화걸기");
    }
    public void answerCall(){
      System.out.println("전화받기");
    }

}
```

```
public class Camera {

    public void photo(){
      System.out.println("사진찍기");
    }

}
```

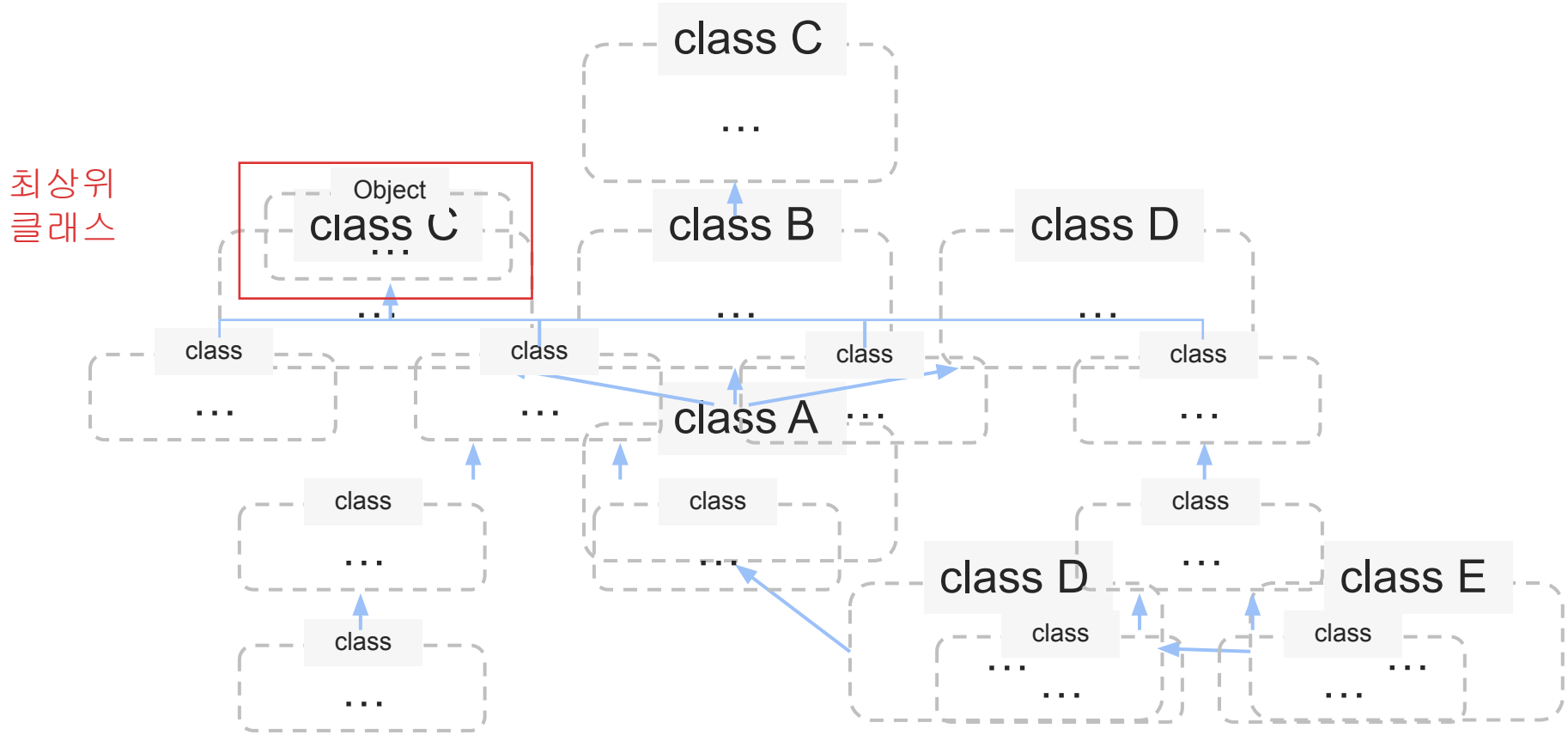

```
public class Smartphone exten
{

    public void playMusic(){
      System.out.println("음악 재생하기");
    }
    public void internet(){
      System.out.println("인터넷하기");
    }

}
```

하나의 수퍼 클래스만 가질 수 있음.

상속의 특징

자바 상속의 특징 2. 상속의 횟수에 제한을 두지 않는다.

자바 상속의 특징 3. 모든 클래스는 java.lang.Object를 상속받는다.



상속의 특징

- ✔ 다중 상속을 지원하지 않는다
  - 한 서브 클래스는 하나의 슈퍼 클래스만 가질 수 있음
- ✔ 상속의 횟수에 제한을 두지 않는다
  - 서브 클래스가 상속받는 슈퍼클래스의 또다른 슈퍼클래스가 존재할 수 있음
- ✔ 모든 클래스는 java.lang.Object를 상속받는다
  - Object가 모든 클래스의 최상위 클래스

상속 예제 - Phone


StarTAC
MOTOROLA


SKY

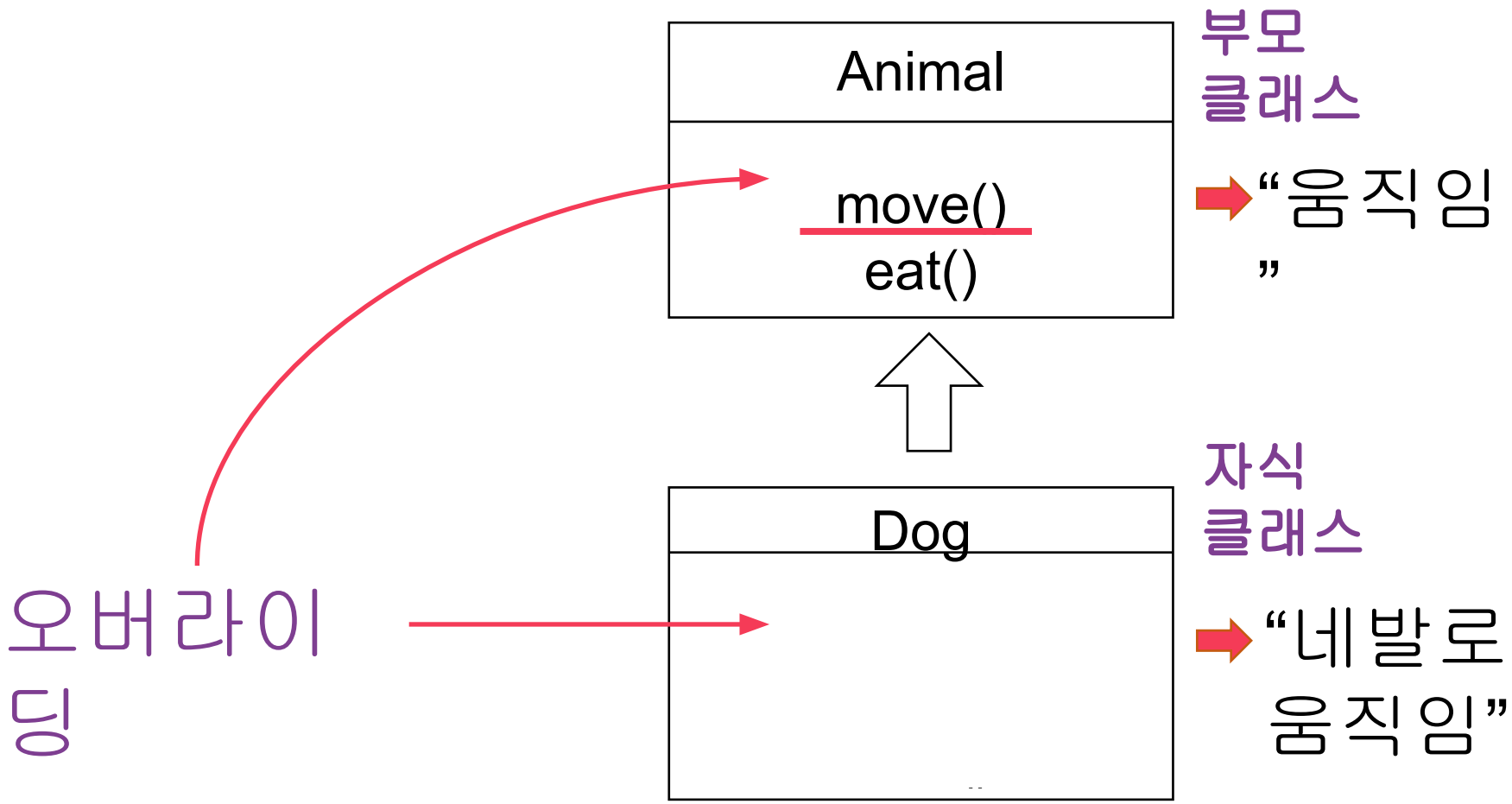

9:41 AM
100%
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Notes
Reminders
Stocks
Game Center
Newsstand
iTunes Store
App Store
iBooks
Health
Passbook
Settings
Phone
Mail
Safari
Music
iPhone

모든 핸드폰을 특징을 가지고 있는 Phone 클래스를 설계한 후 Phone 클래스를 상속받은 FolderPhone 클래스와 SmartPhone 클래스를 설계하고 각 객체를 생성하시오

Phone	FeaturePhone	SmartPhone
call 전화	call 전화	call 전화
text 문자	text 문자	text 문자
	camera 카메라	camera 카메라
		Internet 인터넷

오버라이딩(Overriding)

자식 클래스에서 부모 클래스와
동일한 메소드를 재정의 하는것


Animal
move()
eat()
부모 클래스
"움직임"
Dog
자식 클래스
"네발로 움직임"
오버라이딩

오버로딩(Overloading)

매개변수의 개수나
순서, 타입이 다른
같은 이름의 메소드를
여러 개 정의 하는 것

중복정의

오버라이딩(Overriding)

부모클래스의
메소드의 동작 방법을
변경하는 것
(매개변수의 개수,
순서,타입과
반환타입은 같음)

재정의

모든 동물들의 특징/기능을 가지고 있는 Animal 클래스를 설계한 후 해당 클래스를 상속받는 Dog 클래스와 Cat 클래스를 작성한 후 cry 메소드 기능을 오버라이딩 하시오.

Animal	Dog	Cat
cry (“울다”) play(“놀다”)	cry(“멍멍”) play(“터그놀이”)	cry(“야옹”) play(“레이저놀이”)

Casting(캐스팅)

기존 데이터 타입을 다른 데이터 타입으로
변환하는 것

* Reference Type 데이터 캐스팅

Upcasting	Downcasting

- ✔ 하위 클래스가 상위 클래스 타입으로 자동 타입 변환하는 것
  - OOP 특징 중 다형성에 해당
- ✔ 업캐스팅 된 경우 원래 객체 내 모든 변수, 메소드에 접근할 수 없고
  - 상위 클래스의 변수, 메소드에만 접근 가능
- ✔ 하위 클래스가 상위클래스의 메소드를 오버라이딩 한 경우 하위 클래스의 메소드 호출 가능

✔ 업캐스팅된 것을 명시적 타입 변환으로 원래 상태로 되돌리는 것

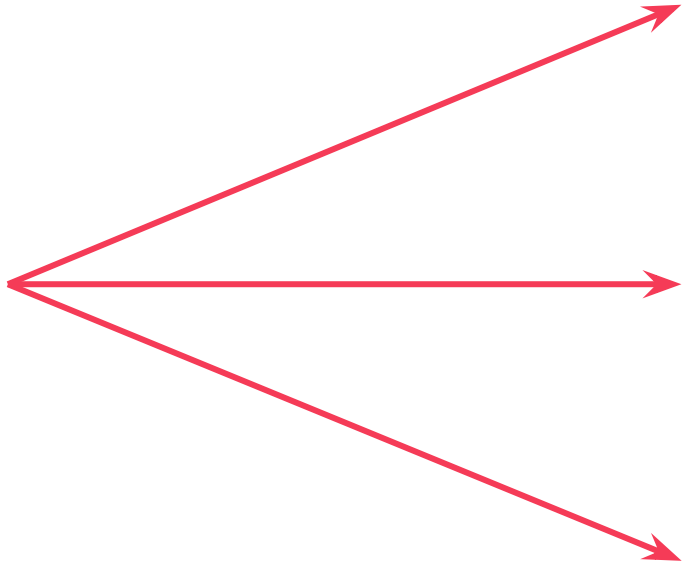
- 업캐스팅이 되면 기존 자식클래스의 멤버(필드, 메소드)를 사용하지 못하기 때문에 자식 클래스의 멤버(필드, 메소드)를 사용하기 위해서 다운캐스팅 진행
- 업캐스팅 되지 않은 객체를 다운캐스팅 하게 되면 오류 발생

* instanceof 연산자

업캐스팅한 경우 레퍼런스 변수가 가리키는 실체 객체가
어떤 클래스의 타입인지 구분하기 위한 연산자

다형성

블루투스
마우스

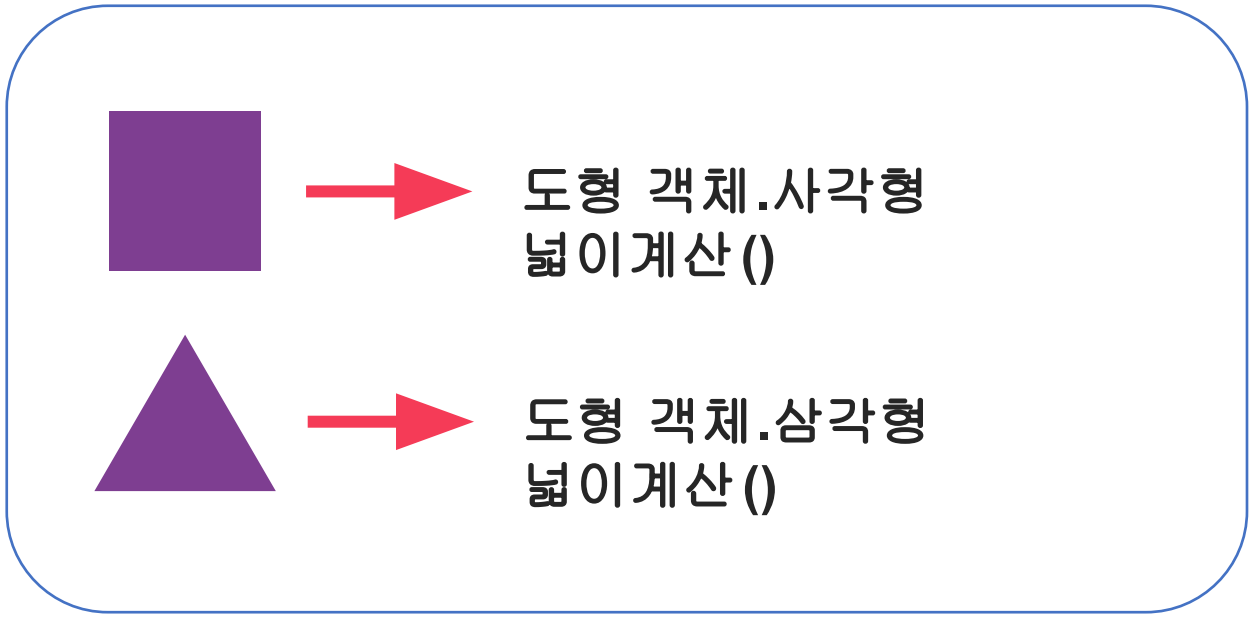


다형성(Polymorphism)

- 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)

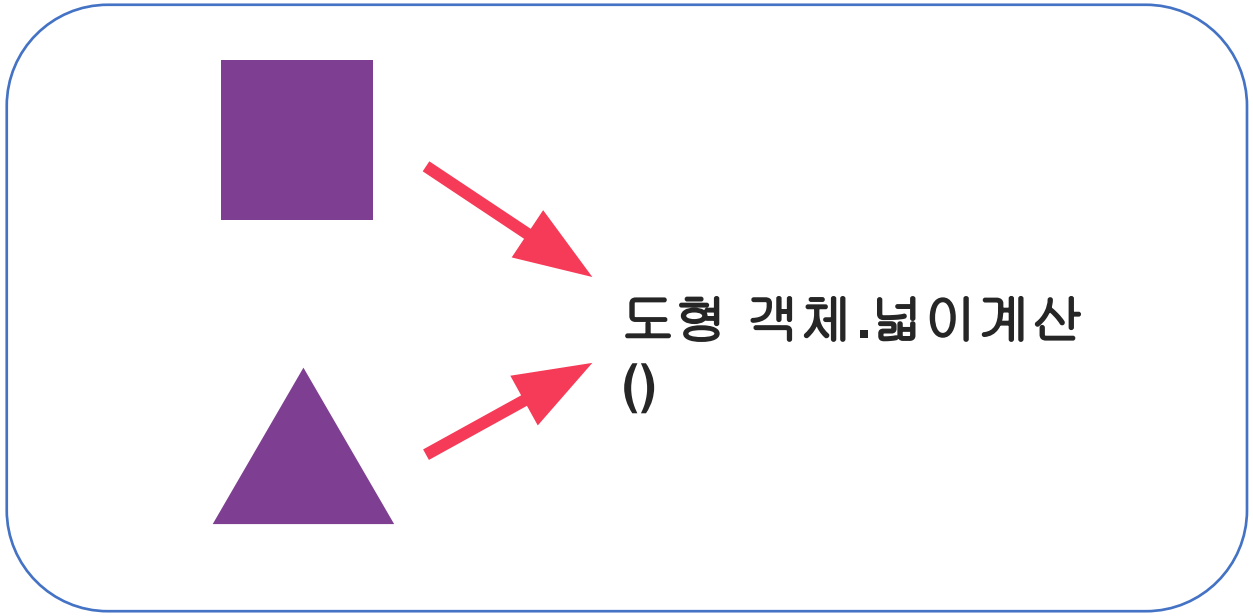
다형성을 적용하지 않은 경우



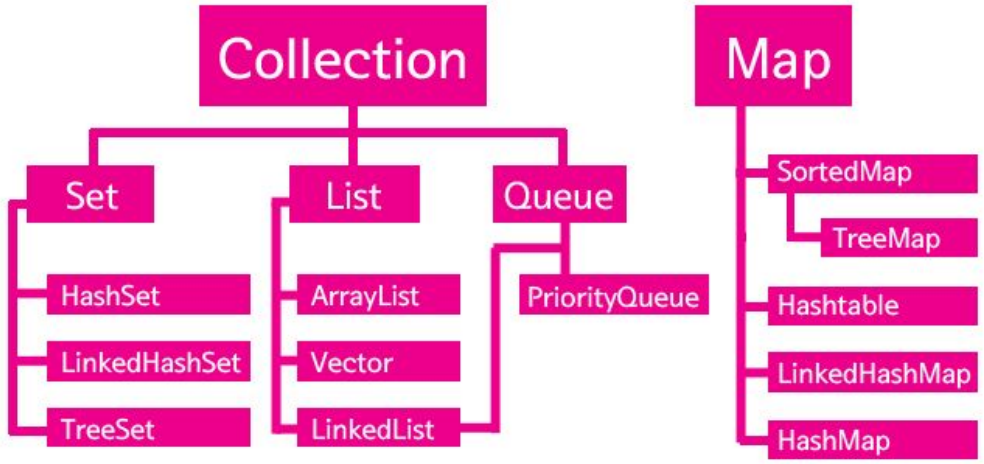
다른 도형을 추가하고 싶다면
메소드가 계속 늘어나야한다.
수정해야될 파일이 늘어나고
생산성이 떨어진다.

새로운 요구사항에 유연하지 못한 프로그램이된다.

다형성을 적용한 경우



변화에 유연한 프로그램을
작성 할 수 있다.
(기능 확장에 열려있다.)



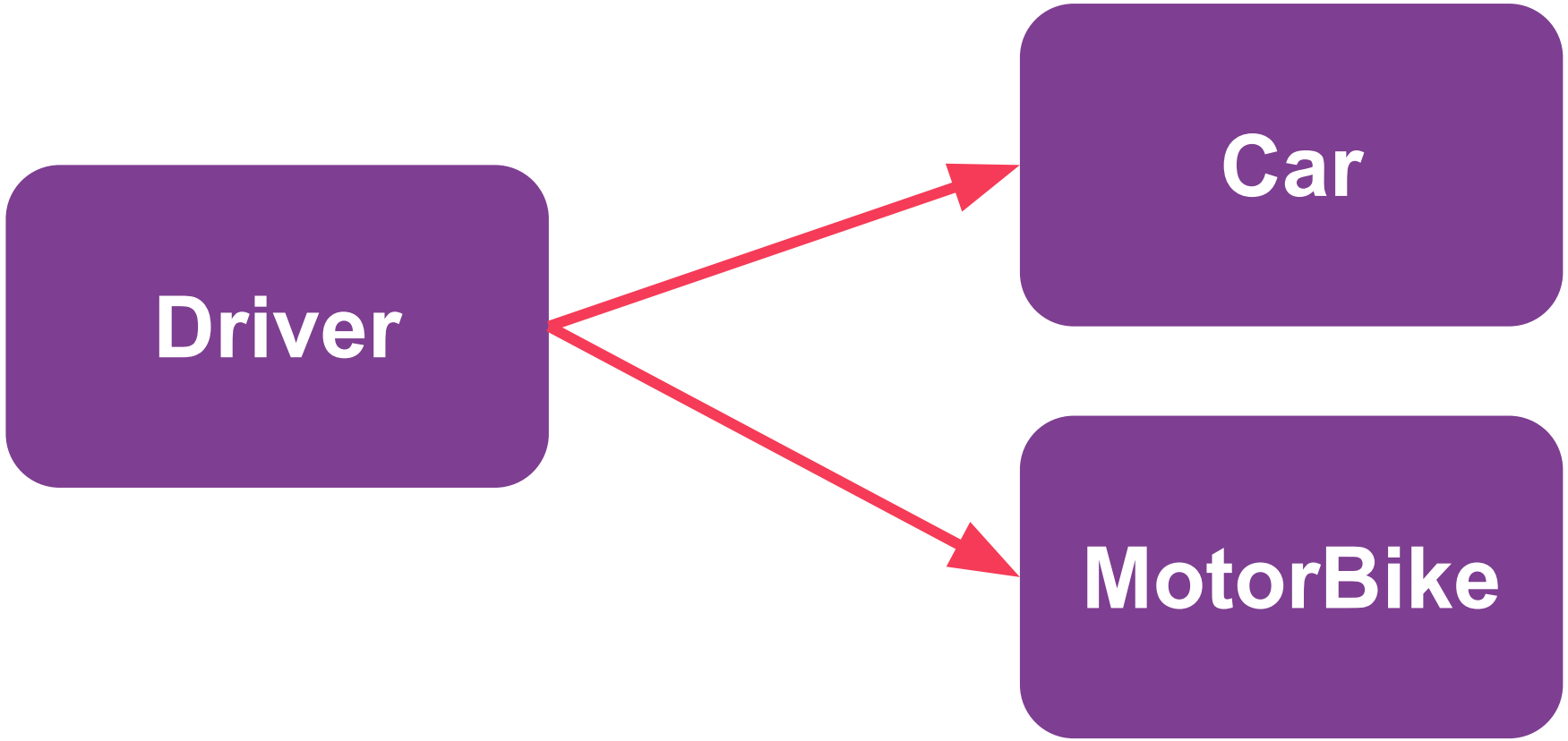
List
계열

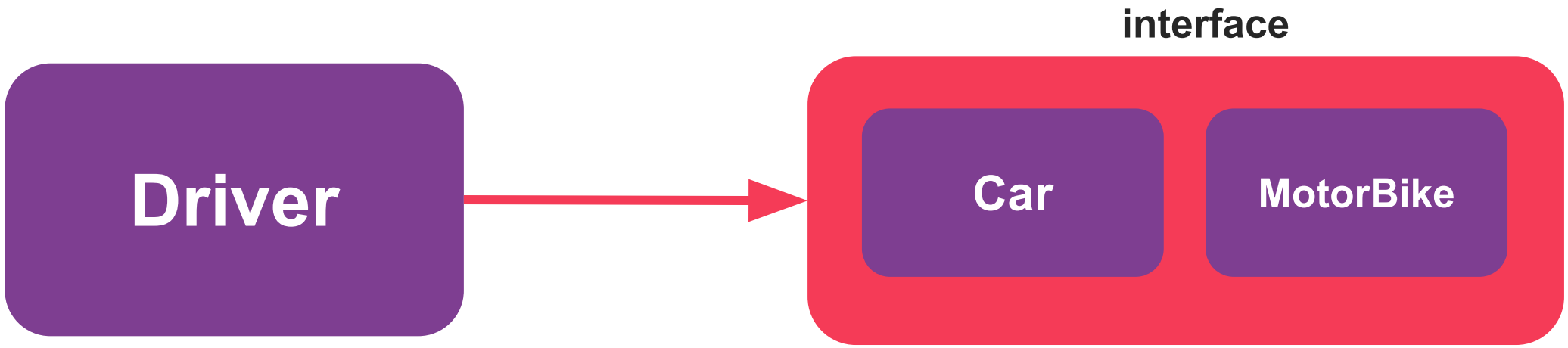
ArrayList

Vector

LinkedLis
t

add 메소드
get 메소드
remove 메소드
.
.
.


Driver
Car
MotorBike


interface
Driver
Car
MotorBike

추상화(Abstract)

캡슐화(Encapsulation)

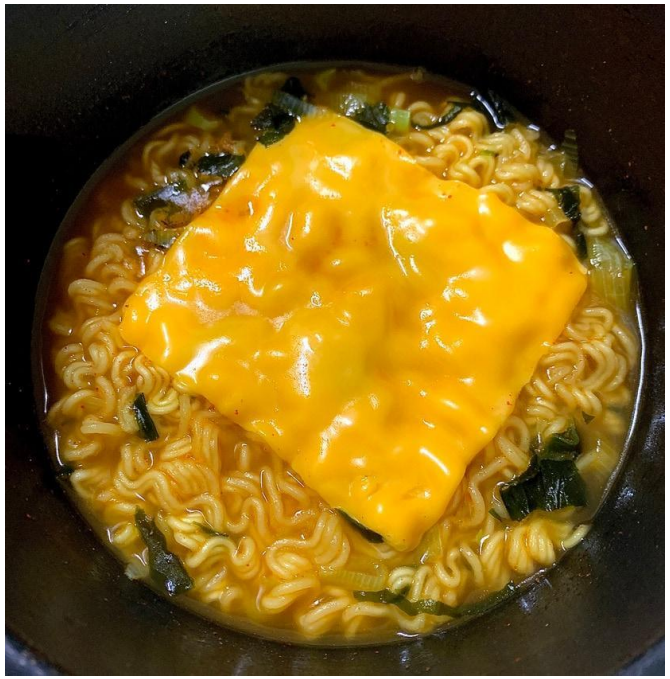
캡 상 추 다

상속(Inheritance)

다형성(Polymorphism)

만두라면

떡라면

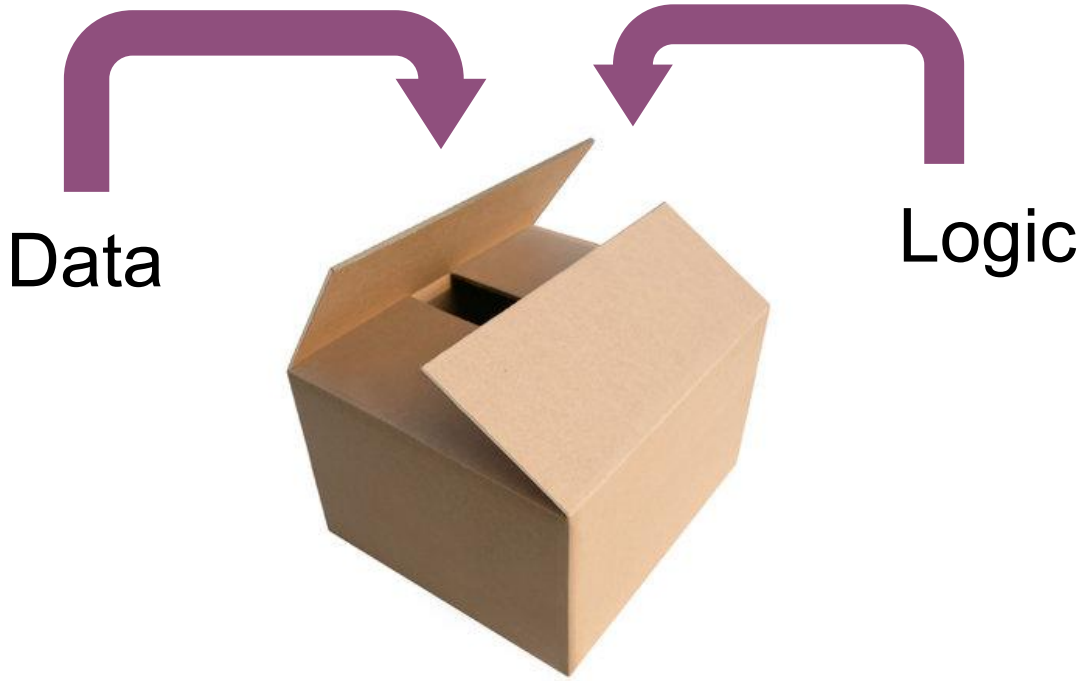


치즈라면

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)

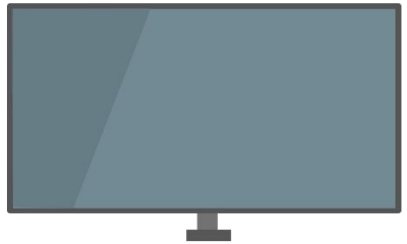
- 만일의 상황(타인이 외부에서 조작)을 대비해서 특정 속성이나 메소드를
  사용자가 조작할 수 없도록 숨겨 놓은 것.

- 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

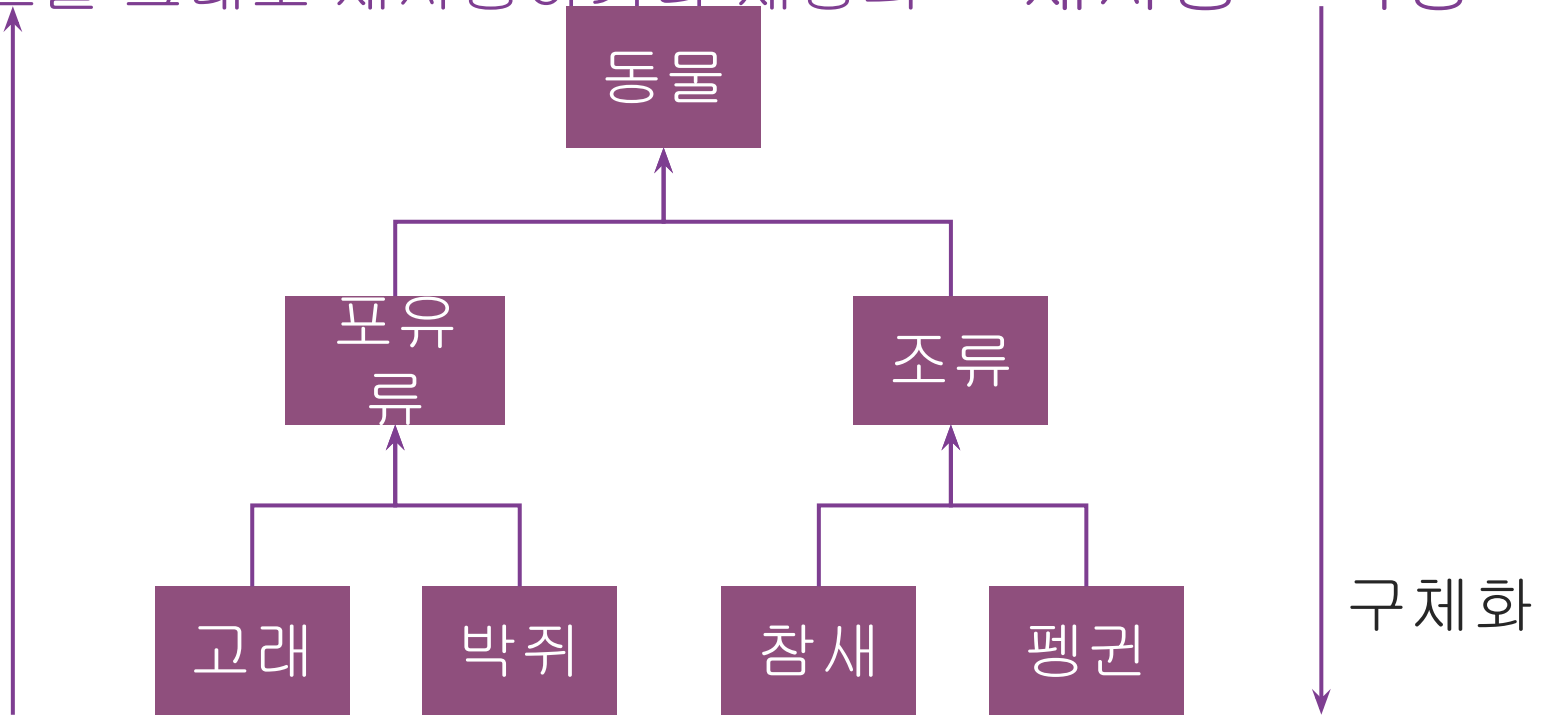
sort()
binarySearch()

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 장점

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

다음시간에는?

추상클래스