

Java 객체지향 프로그래밍

객체지향 이해하기

객체지향 기본개념

: 도서관대여점 프로그램의 예



- ✓ 책
- ✓ 책장
- ✓ 고객
- ✓ 고객장부
- ✓ 금고
- ✓ 대여장부
- ✓ ...

현실세계의 객체들

- ▶ 기능이 많아지고 복잡해진다
- ▶ 현실 세계를 시뮬레이션 할 필요의 대두
- ▶ 새로운 개발 방법이 필요해짐



✓ **객체지향 프로그래밍**
(OOP: Object Oriented Programming)

객체지향 기본 개념

: 도서대여점 객체지향 설계

✓ 책

✓ 책장

✓ 고객

✓ 고객장부

✓ 금고

✓ 대여장부

✓ ...

• 책을 의인화 해봅시다.

• 책이 살아 있다면 책에게 다음과 같은 질문을 할 수 있을 것입니다.

– 책 제목은 어떻게 되니?

– 저자는 어떻게 되니?

– 출판사는 어떻게 되니?

– 가격은 어떻게 되니?

– ...

객체지향 기본 개념

: 도서대여점 객체지향 설계

- ✓ 책
- ✓ 책장
- ✓ 고객
- ✓ 고객장부
- ✓ 금고
- ✓ 대여장부
- ✓ ...

• 책장도 의인화 해봅시다.

• 책장이 살아 있다면 책장에게 다음과 같은 질문을 할 수 있을 것입니다.

- 가지고 있는 책은 모두 몇권이니?
- "해리포터" 라는 소설을 가지고 있니?
- C. S. 루이스가 쓴 판타지 소설은 뭐지?

• 책을 가지고 있는 것은 책장입니다.

• 의인화 해보면 책장이 책을 관리하고 있습니다.

객체지향 기본 개념

: 도서관대여점 객체지향 설계

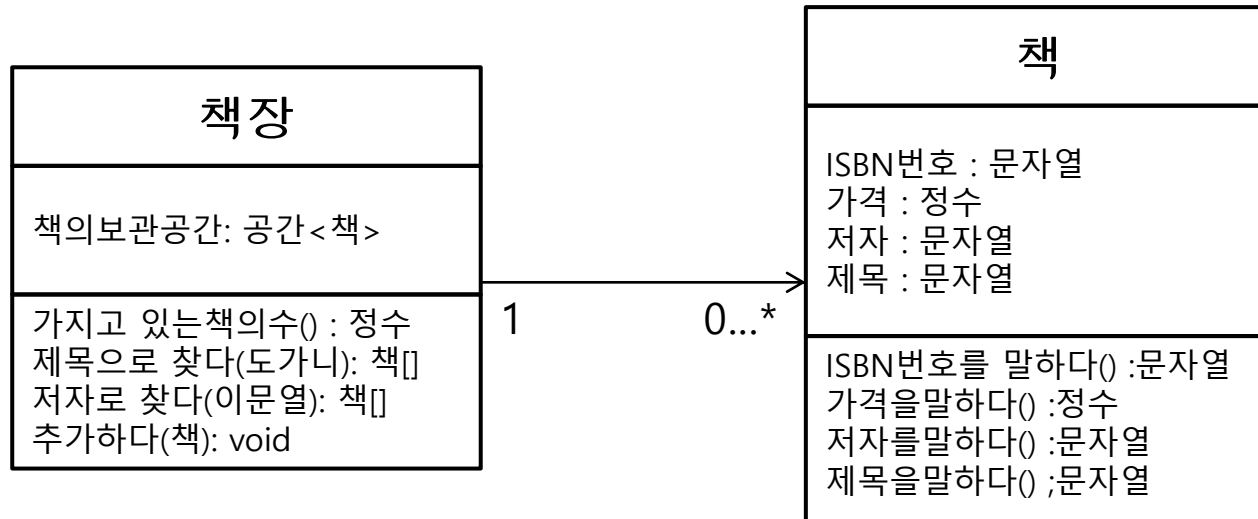
- ✓ 책
- ✓ 책장
- ✓ 고객
- ✓ 고객장부
- ✓ 금고
- ✓ 대여장부
- ✓ ...

• 책과 책장을 도식화 해보면



객체지향 기본 개념

: 도서관대여점 객체지향 설계



▶ 관계를 표현해 보면

- ▶ 책장은 책을 저장하는 공간을 가지고 있다(속성)
- ▶ 책은 ISBN 번호, 가격, 저자, 제목을 가지고 있다 (속성)
- ▶ 책장은 자신이 가지고 있는 속성을 이용하는 기능을 가지고 있다
- ▶ 책장은 책을 보유할 수 있다 (관계)

객체지향 기본 개념

: 개념의 확인

- ✓ 책
- ✓ 책장
- ✓ 고객
- ✓ 고객장부
- ✓ 금고
- ✓ 대여장부
- ✓ ...

• 고객과 고객장부의 관계를 그려보기

객체지향 기본 개념

: 도서대여점 객체지향 설계

- ▶ 도서대여점 고객 vs 신발가게 고객
 - ▶ 도서 대여점의 고객에게 신발 사이즈를 물어본다면?
 - ▶ 도서 대여점의 고객에게 몸무게를 물어본다면?
 - ▶ 신발 가게 고객에게 신발 사이즈를 물어본다면?

- 도서대여점 고객은 이름과 연락처가 중요
- 신발가게는 신발사이즈가 중요

**중요한 것은 남기고, 불필요한 것을 없애는 것을 객체지향에서
“추상화” 라고 말한다.**

객체지향 기본 개념

: 도서대여점 객체지향 설계

- ▶ 모든 것이 중요한 것은 아니다

- ▶ 도서대여점의 책장은 색이 중요하지 않지만, 가구점에서는 색이 중요하다
- ▶ 어느 관점에서 보느냐에 따라 중요한 속성도 있고, 그렇지 않은 것도 있다.

- 객체지향 프로그래밍은 관점에 따라 객체가 가지는 속성과 기능을 다르게 표현하게 된다.

- ▶ 관점을 어떻게 잡고 어떻게 설계하느냐에 따라 재사용이 어려울 수도 있다

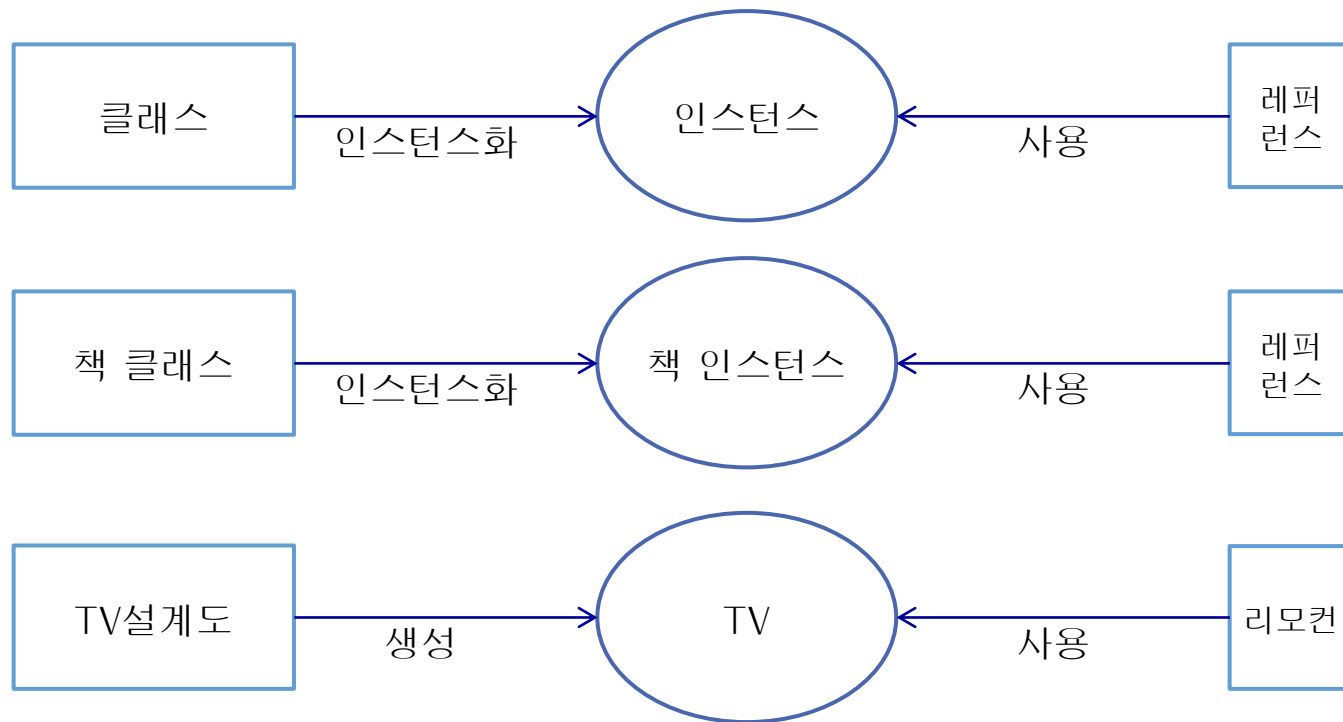
객체 지향 기본 개념

: 객체 지향이 추구하는 것

- ▶ 현실세계를 그대로 옮긴다
 - ▶ 객체들이 가지고 있는 속성과 기능 중 필요한 것만 남기고 불필요한 것은 제거한다
 - ▶ 추상화한다
- ▶ 재사용한다
 - ▶ 관점을 잘 파악하고 그 요구사항에 맞게 설계하면 재사용이 가능해진다
 - ▶ 설계도를 그려두면 양산이 가능해진다

객체 지향 기본 개념

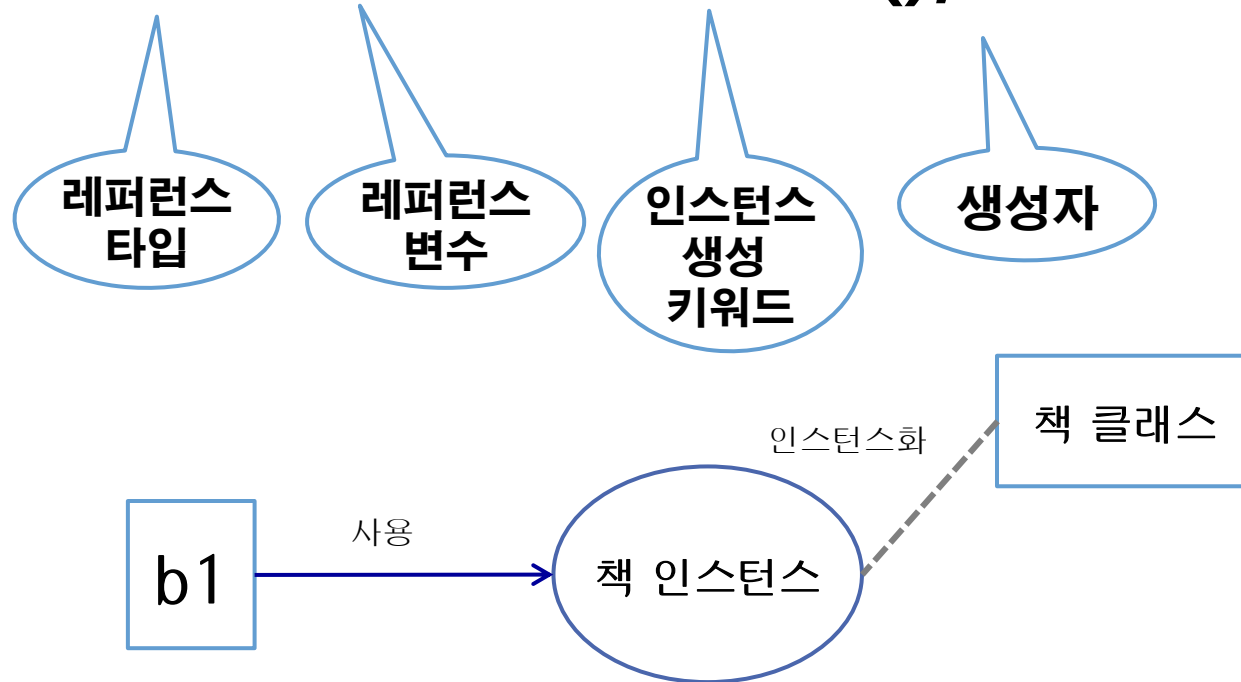
: 클래스, 인스턴스, 레퍼런스



객체 지향 기본 개념

: Java 문법으로 이해하기

책 b1 = new 책();



책
ISBN번호 : 문자열 가격 : 정수 저자 : 문자열 제목 : 문자열
ISBN번호를 말하다() : 문자열 가격을말하다() : 정수 저자를말하다() : 문자열 제목을말하다() : 문자열

객체 지향 기본 개념

: Java 문법으로 이해하기

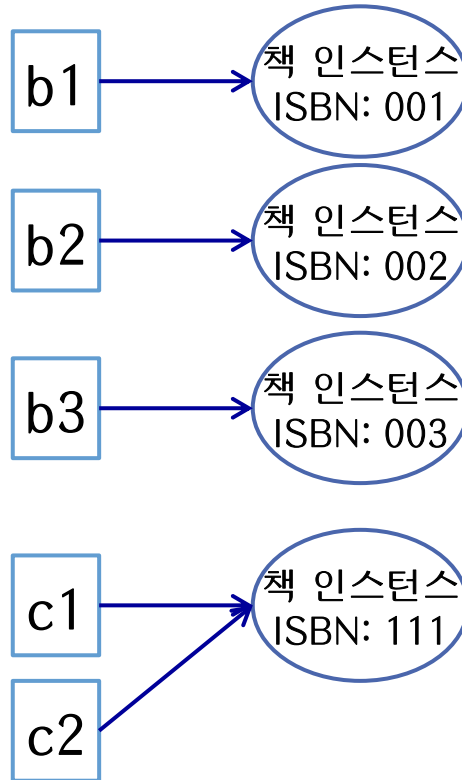
```
public static void main(){
```

```
책 b1 = new 책(001);  
책 b2 = new 책(002);  
책 b3 = new 책(003);
```

```
책 c1 = new 책(111);  
책 c2 = c1;
```

```
c1= null;
```

```
}
```



책
ISBN번호 : 문자열 가격 : 정수 저자 : 문자열 제목 : 문자열
ISBN번호를 말하다() : 문자열 가격을말하다() : 정수 저자를말하다() : 문자열 제목을말하다() : 문자열

객체 지향 기본 개념

: 자동차 중 버스, 스포츠카, 포크레인을 정의해 봅시다

- ▶ 버스, 스포츠카, 포크레인의 공통점을 추출하여 자동차를 구상해 봅시다.

버스	스포츠카	포크레인
달리다():void 뒷문열다():void 안내방송하다():void	달리다():void 지붕을열다():void	달리다(): void 기계삽사용하다():void

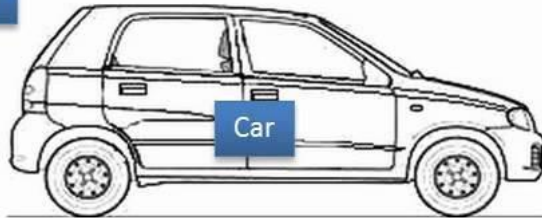
객체 지향 기본 개념

: 자동차의 예

What is a Class?

A class is the blueprint from which individual objects are created.

Class



```
1 public class Car
2 {
3     private String brand = null;
4     private String model = null;
5     private String color = null;
6
7     public String getBrand()
8     {
9         return brand;
10    }
11
12    public void setBrand(String brand)
13    {
14        this.brand = brand;
15    }
16
17    public String getModel()
18    {
19        return model;
20    }
21
22    public void setModel(String model)
23    {
24        this.model = model;
25    }
26
27    public String getColor()
28    {
29        return color;
30    }
31
32    public void setColor(String color)
33    {
34        this.color = color;
35    }
36
37 }
```

Objects

```
Car maruthiAltoK10 = new Car();
maruthiAltoK10.setBrand("Maruthi Alto");
maruthiAltoK10.setModel("K10");
maruthiAltoK10.setColor("Orange");
```

brand = Maruthi Alto
model = K10
color = Orange



```
Car swift = new Car();
swift.setBrand("Swift");
swift.setModel("ZDI");
swift.setColor("Red");
```

brand = Swift
model = ZDI
color = Red

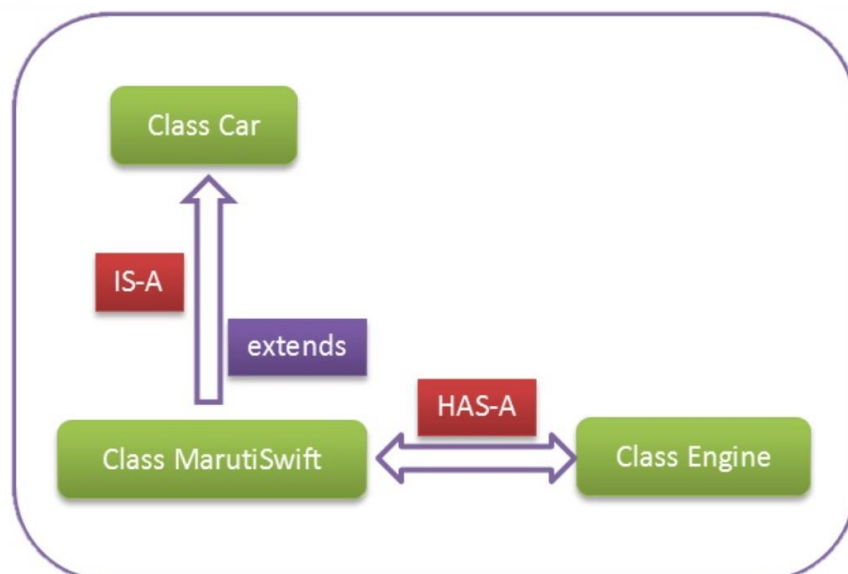
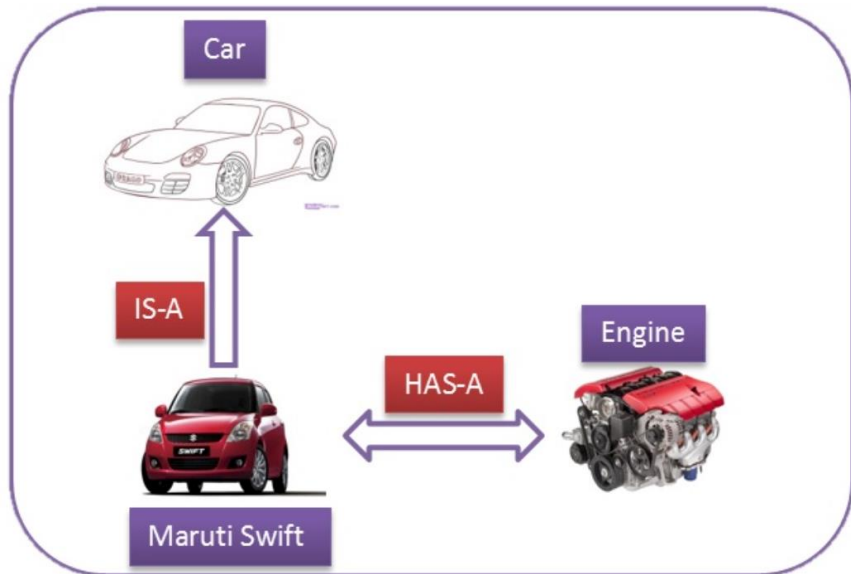
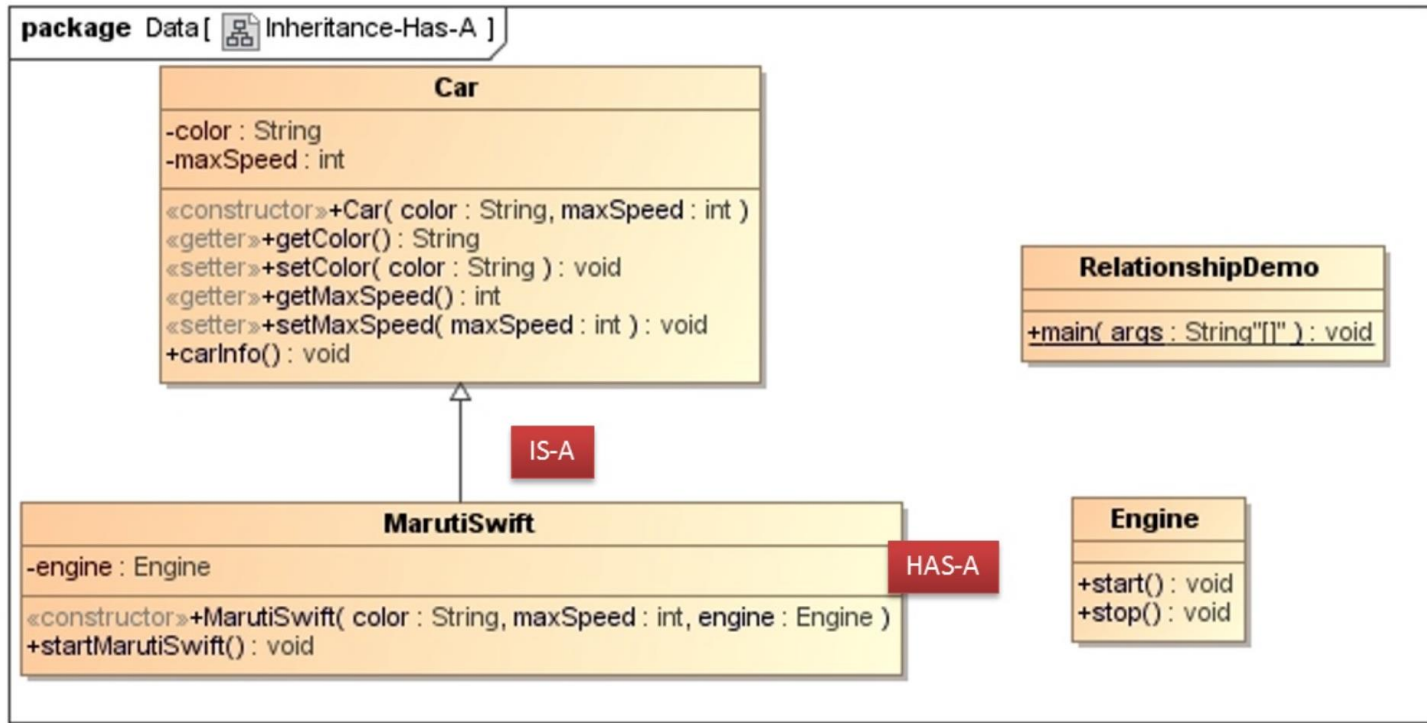


```
Car maruthiAlto800 = new Car();
maruthiAlto800.setBrand("Maruthi Alto");
maruthiAlto800.setModel("800");
maruthiAlto800.setColor("Blue");
```

brand = Maruthi Alto
model = 800
color = Blue



객체지향 기본 개념 : 자동차의 예

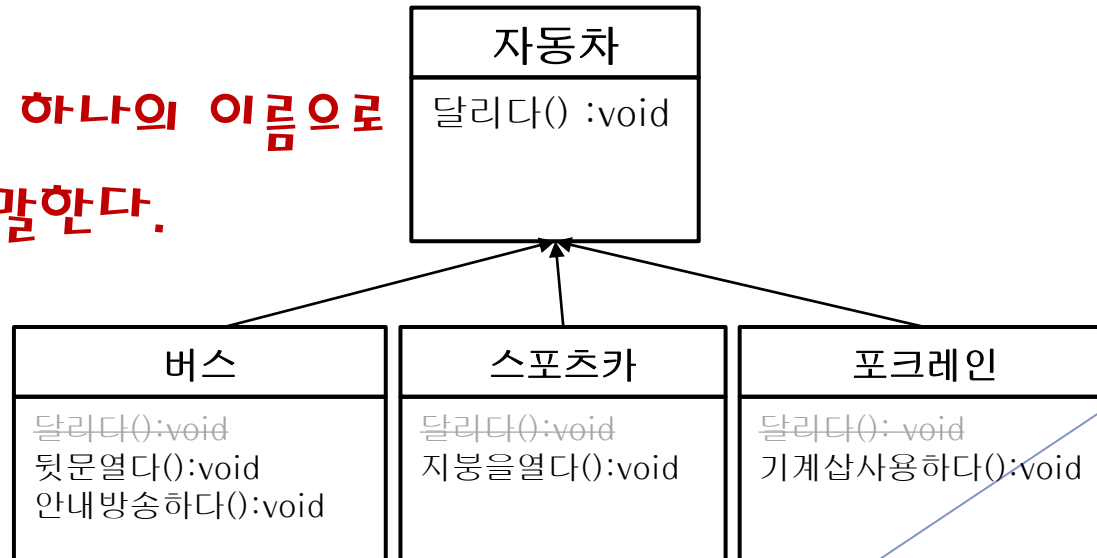


객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

- ▶ 버스, 스포츠카, 포크레인은 자식 클래스
- ▶ 자동차는 부모 클래스
- ▶ 버스, 스포츠카, 포크레인을 일반화 한 것은 자동차
- ▶ 자동차를 상속(확장)한 것은 버스, 스포츠카, 포크레인

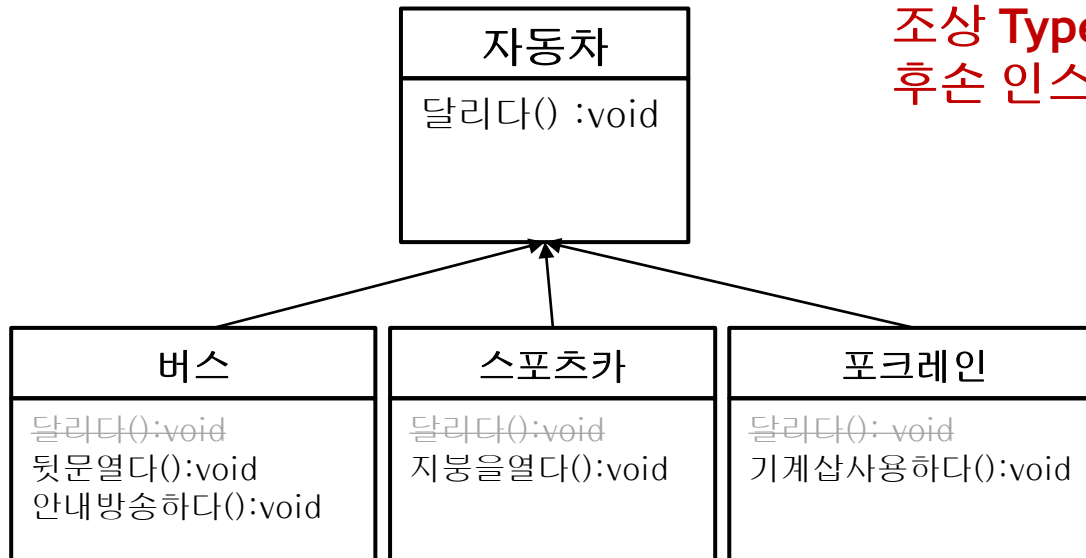
공통점이 있는 여러가지 물체들을 하나의 이름으로 부르는 것을 “일반화” 라고 말한다.



객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

- ▶ 버스는 자동차다
- ▶ 스포츠카는 자동차다
- ▶ 포크레인은 자동차다



자동차 `c1 = new` 버스();
자동차 `c2 = new` 스포츠카();
자동차 `c3 = new` 포크레인();

버스 `bus = new` 버스();
스포츠카 `sportCar = new` 스포츠카();
자동차 `poclain = new` 포크레인();

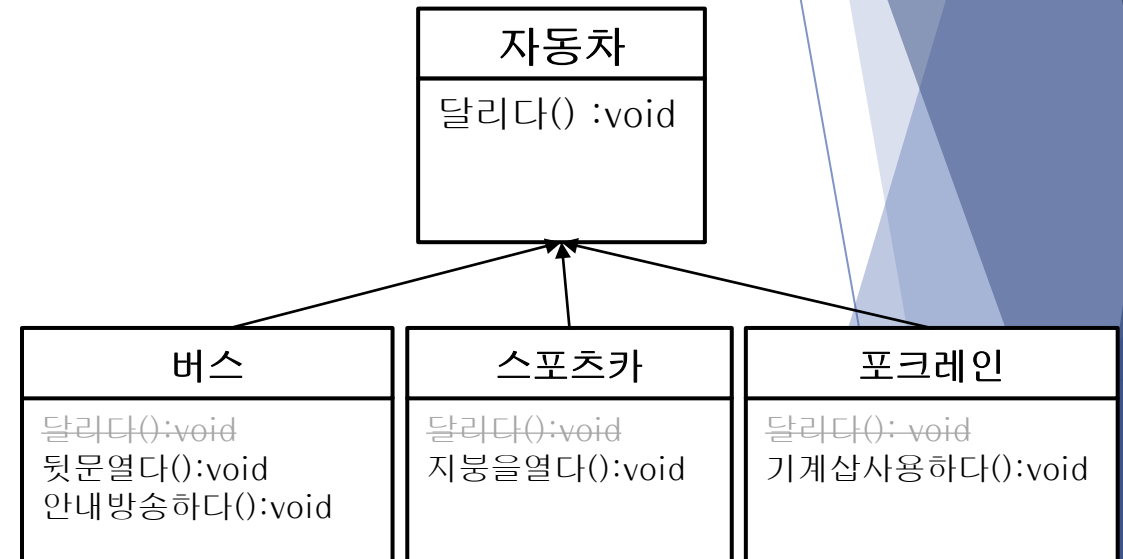
조상 **Type**의 레퍼런스 변수는
후손 인스턴스를 레퍼런스 할 수 있다.

객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

```
버스 bus1 = new 버스();  
bus1.달리다();  
bus1.뒷문열다();  
bus1.안내방송하다();
```

- ▶ 버스 주차를 요청하니 안내방송을 실행
- ▶ 스포츠카 주차를 요청하니 지붕을 열고 닫음
- ▶ 포크레인 주차를 요청하니 땅을 파고 있음

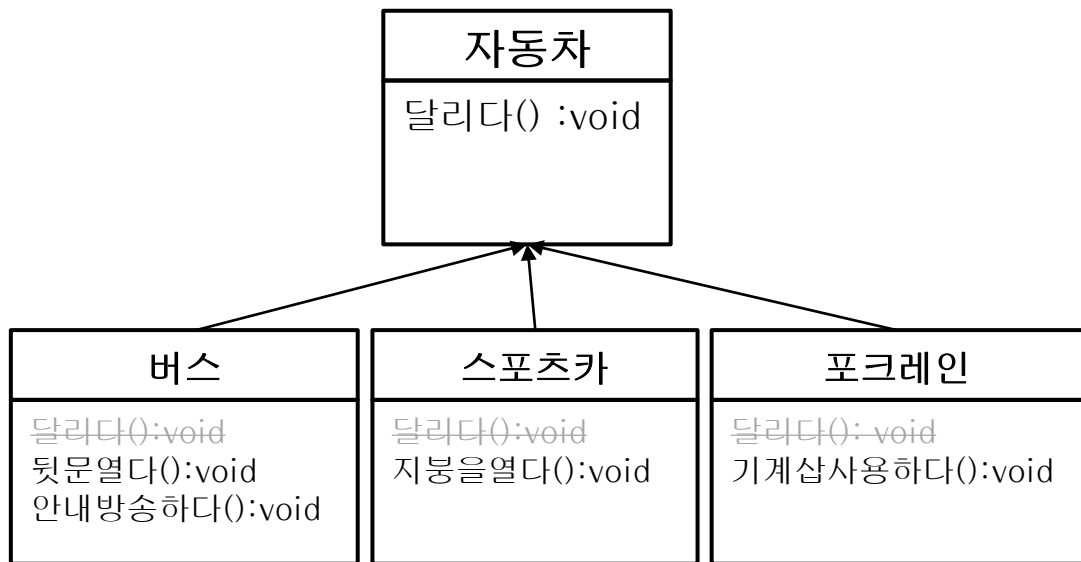


**자동차의 주인은 자동차의 기본 기능만
이용하여 주차하길 바란다.**

객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

- ▶ 레퍼런스 타입 클래스에 설계되어 있는 기능만 사용할 수 있다.



버스와 그 조상 설계도에
있는 기능만 사용 가능

버스 bus1 = new 버스();
bus1.달리다();
bus1.뒷문열다();
bus1.안내방송하다();

자동차와 그 조상 설계도에
있는 기능만 사용 가능

자동차 bus2 = new 버스();
bus2.달리다();
bus2.뒷문열다(); (X)
bus2.안내방송하다(); (X)

설계도에 구현된 부분 이외의 기능은 사용할 수 없다.

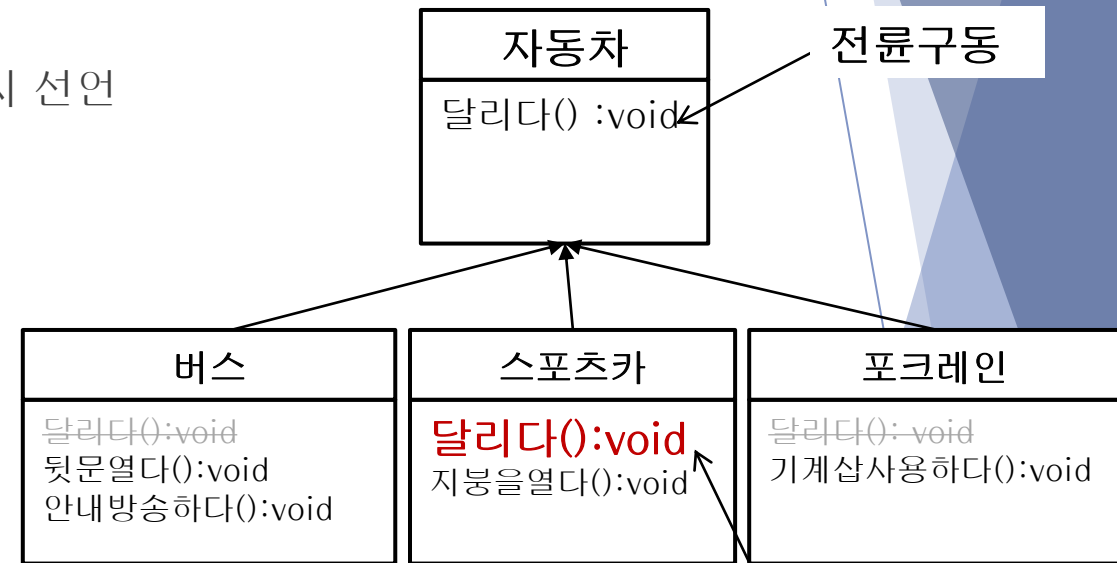
객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

- ▶ 스포츠카는 자동차가 가지고 있는 본래의 "달리다" 기능을 자신만의 구동방식으로 바꾸고자 합니다.
 - ▶ 부모의 기능과 완전히 같은 모양으로 내용을 다시 선언
-> 메서드 오버라이드

```
자동차 bus = new 버스();  
bus.달리다();  
bus.뒷문열다(); (X)  
bus.안내방송하다(); (X)
```

```
자동차 spoCar = new 스포츠카();  
spoCar.달리다(); ← 후륜구동  
spoCar.뒷문열다(); (X)  
spoCar.안내방송하다(); (X)
```



메소드 오버라이드(Override: 덮어쓰다)
← 오버로드(Overload) 와 자주 혼동되니 주의

객체 지향 기본 개념

: 주차장 프로그램의 설계 예시

▶ 주차 공간의 확보

```
자동차 bus1= new 버스();  
자동차 spo1= new 스포츠카();  
자동차 pocl1= new 포크레인();
```

```
자동차[] carArray = new 자동차[15]  
carArray[0] = bus1;  
carArray[1] = spo1;  
carArray[2] = fork1;
```

```
버스[] busArray = new 버스[5]  
스포츠카[] spoArray = new 스포츠카[5]  
포크레인[] poclArray = new 포크레인[5]
```

