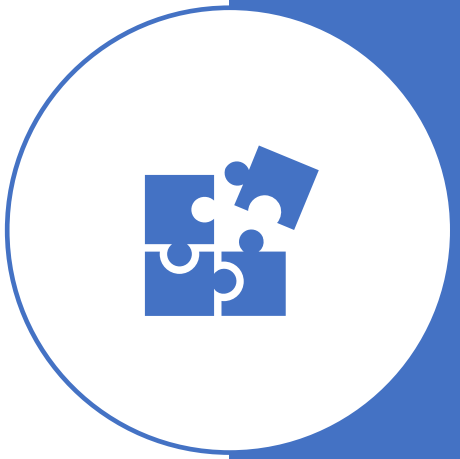
Java Fundamental

객체 지향 프로그래밍

객체 지향 프로그래밍
(Object-oriented programming)



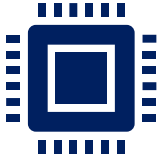
- 코드를 현실 세계의 사물에 빗대어 표현하는 방식

현실 세계와 프로그래밍의 세계

사물
(현실)

상태(State)
- 차종: SUV
- 색상: 빨간색
- 속력: 72.4 km/h

행동(Behavior)
- 시동을 걸다
- 속력을 높이다
- 제동을 걸다



클래스
(프로그래밍)

필드(Field)
- String type = "SUV";
- String color = "red";
- double velocity = 72.4;

메서드(Method)
- set();
- accel();
- break();

객체 지향 프로그래밍의 장점

재사용하기 쉬운 코드

편리한 유지보수

직관적인 코드 구성

객체 지향 프로그래밍의 특징

추상화

핵심 요소 추출하기

캡슐화

데이터와 처리 방법의 묶음

상속

요소의 재사용과 기능의 확장

다형성

같은 이름, 다른 기능

클래스, 객체, 인스턴스

클래스

- 객체를 만들기 위한 설계도

객체

- 클래스로 만들어진 대상
- 클래스의 인스턴스

인스턴스

- 메모리에 적재된 실체

클래스 작성 방법

```
class [클래스명] {
        [자료형] [변수명];

        [반환형] [메서드명]([자료형] [매개변수명]) {
                ...
        }
}
```

객체 생성 방법

❖선언

[클래스명] [변수명];

❖초기화

[변수명] = new [클래스명]();

❖선언 및 초기화

[클래스명] [변수명] = new [클래스명]();

객체 사용 방법

[클래스명] [객체명] = new [클래스명]();

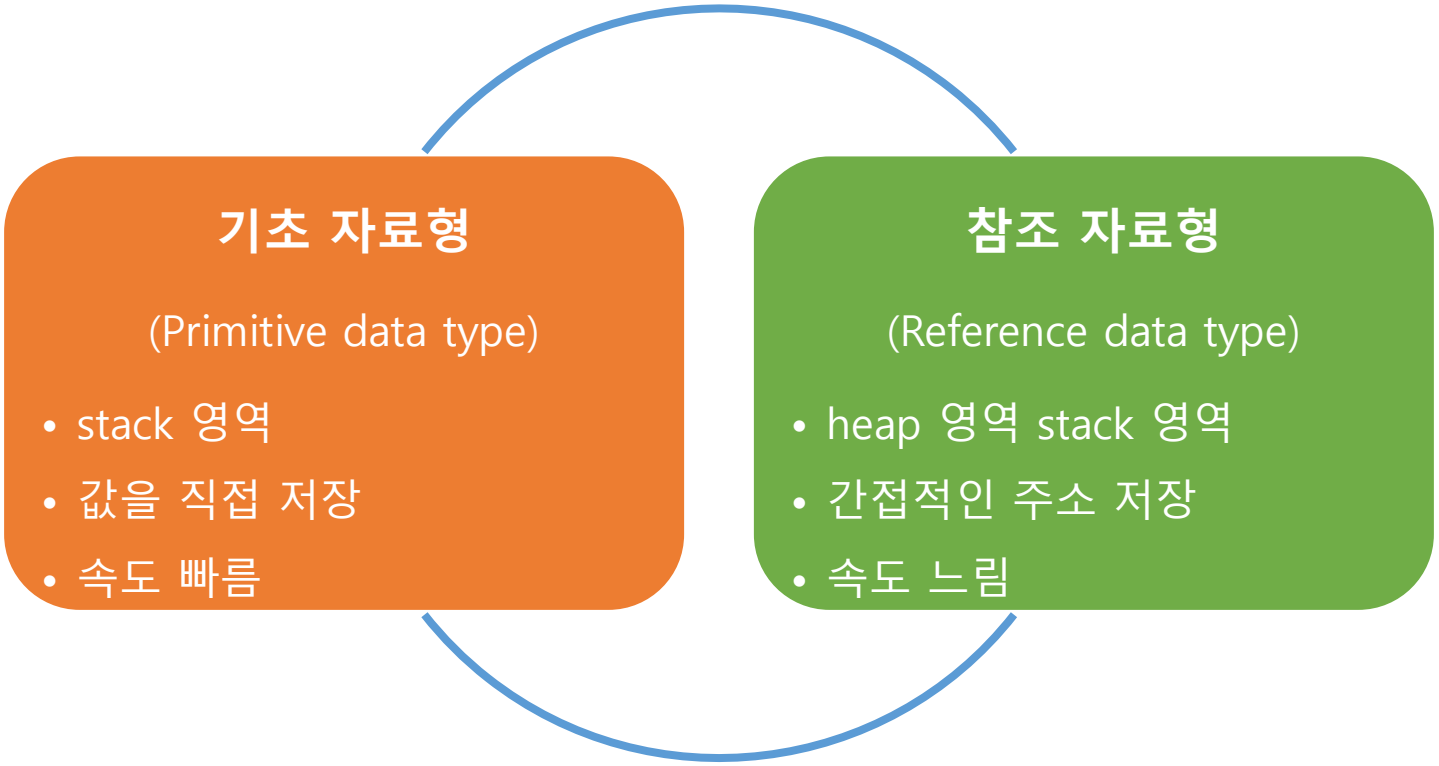
❖필드 호출

[객체명].[필드명];

❖메서드 호출

[객체명].[메서드명]([인수]);

기초 자료형과 참조 자료형



자료형의 기본값

자료형	기본값
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'₩u0000'
boolean	false
참조 자료형	null

객체의 생애주기

- 어떤 변수도 heap 영역의 데이터를 참조하지 않으면 해당 데이터는 제거 대상이 됨
- 제거 대상이 된 데이터는 JVM의 Garbage Collector(GC)에 의해 제거됨
- GC는 유휴 시간(idle time)에 동작함