**package** basic;

**class** Person {

//필드

**private** String name; // private : 외부에서 접근을 못하게 막아주는 것

**private** **int** age; // 자신의 클래스 안에서만 접근 허용

**public** **void** setName(String n) { // 구현

name = n;

}

**public** **void** setAge(**int** a) {

age = a;

}

**public** **void** setData(String n, **int** a) { //함수의 이름이 똑같이 생긴 걸 Overload라고 한다.

name = n;

age = a;

}

// 돈이 들어오는 메소드를 잡고 커피를 만들어서 내보내는 메소드를 하나 만든다.

// 돈이 들어오는 메소드를 setter(데이터를 받는), 데이터를 내보내는 메소드를 getter라고 한다.

**public** **void** setData() {} //함수의 이름이 똑같이 생긴 걸 Overload라고 한다. 메소드명은 같으나 인수형이 틀리거나 인수의 개수가 틀리거나

// 같은 클래스 내 같은 이름의 메소드를 여러개 만드는 것.

**public** String getName() { //메소드 이름 : get + 필드이름 : getName

**return** name;

}

**public** **int** getAge() {

**return** age;

}

}

//--------------객체지향프로그램

**public** **class** PersonMain {

**public** **static** **void** main(String[] args) {

Person p; // 객체

p = **new** Person(); // 생성 메모리가 위치해 있는 주소

System.***out***.println("객체 p =" + p); // 주소를 가져온다. 객체 p =basic.Person@7de26db8

// 16진수 레퍼런스를 가져온다.

System.***out***.println("객체 p =" + p.toString());

p.setName("홍길동"); // <- p.name = "홍길동";

// p에 살고 있는 name한테 홍길동이라는 이름을 준다.

p.setAge(25); // <- p.age = 25;

System.***out***.println("이름=" + p.getName() + "\t 나이=" + p.getAge());

// new라는 연산자를 이용해서 생성하면 메모리 heap에 할당된다

// heap 영역에 할당된 것은 이름이 없다. 항상 주소를 줘야한다.

Person p2 = **new** Person();

System.***out***.println("객체 p2 =" + p2);

p2.setName("코난");// 호출

p2.setAge(13);

System.***out***.println("이름=" + p2.getName() + "\t 나이=" + p2.getAge());

Person p3 = **new** Person();

p3.setData("라이언", 35);

System.***out***.println("이름=" + p3.getName() + "\t 나이=" + p3.getAge());

Person p4 = **new** Person();

p4.setData(); //원하는 데이터를 넣어야 갈 수 있지 처리되게 하려면 메소드를 만들어줘야

System.***out***.println("이름="+p4.getName()+"\t 나이="+p4.getAge());

System.***out***.println();

}

}

//함수()안에 들어가는 건 parameter 매개변수 혹은 argument 인수라고 한다

[결과]

객체 p =basic.Person@7de26db8

객체 p =basic.Person@7de26db8

이름=홍길동 나이=25

객체 p2 =basic.Person@1175e2db

이름=코난 나이=13

이름=라이언 나이=35