

객체지향 프로그래밍

객체지향 이론 : 실제 세계는 사물(객체)로 이루어져 있으며, 발생하는 모든 사건들은 사물간의 상호작용이다.

💡 객체지향 언어의 특징

- 1 * 코드의 재사용성이 높다.
 - 2 - 새로운 코드를 작성할 때 기존의 코드를 이용하여 쉽게 작성할 수 있다.
- 3 * 코드의 관리가 용이하다.
 - 4 - 코드간의 관계를 이용해서 적은 노력으로 쉽게 코드를 변경할 수 있다.
- 5 * 신뢰성이 높은 프로그래밍이 가능하다.
 - 6 - 제어자와 메서드를 이용해서 데이터를 보호하고 올바른 값을 유지하도록 한다.
 - 7 - 코드의 중복을 제거하여 코드의 불일치로 인한 오동작을 방지할 수 있다.

객체지향 개념을 학습할 때 **재사용성**과 **유지보수**, 그리고 **코드의 중복제거** 관점에서 보면 보다 쉽게 이해할 수 있다.

클래스와 객체

클래스의 정의 : 객체를 정의해 놓은 것 (객체의 설계도)

클래스의 용도 : 객체를 생성하는데 사용

클래스를 정의하고, 클래스를 통해서 **객체**를 만든다. 그렇다면 객체는 무엇인가?

객체의 정의 : 실제로 존재하는 것. 사물 혹은 어떠한 개념

객체의 용도 : 객체가 가지고 있는 기능과 속성에 따라 다르다.

객체의 사전적인 정의는 **실제로 존재하는 것이다**. 실제 생활에서 볼 수 있는 책, 컴퓨터, 휴대폰 등 사물 뿐만이 아니라 우리의 관념상에서 존재하는 성적, 수학 공식, 예러 등 무형 적인 것들도 포함한다.

그렇다면, 클래스와 객체의 관계가 이해 가는가?

예를 들자면, 클래스는 자동차의 설계 도면이고 자동차를 만드는데 사용된다. 이 자동차 설계를 이용해서 만들어진 자동차가 객체가 되는 것이다. 우리가 필요한건 설계도가 아니고 자동차이다. 따라서 **우선 클래스를 작성하여 설계를 만들고, 클래스를 통해서 객체를 만들어서 사용하는 개념이다**.

자동차는 만들기 매우 복잡하고 어렵지 않을까? 이렇게 복잡한 자동차를 설계도 없이 만든다는 것은 상상할 수 없을 것이다. 또한, 자동차를 여러대 만든다고 가정할 때 설계도가 없다면 매번 자동차를 만들때마다 결과물이 달라질 것은 물론이요, 만들때마다 고민하고 고민해야 할 것이다. 따라서, **한 번 클래스를 잘 만들어 놓으면, 객체가 필요할 때마다 고민 안 하고 객체를 생성할 수 있을 것이다.**

객체와 인스턴스

인스턴스화(instantiate) : 클래스로부터 객체를 만드는 과정

인스턴스(instance) : 클래스로부터 만들어진 객체

객체나 인스턴스는 거의 같은 의미이지만, 객체가 더 개괄적인 느낌이다.

객체의 구성요소 - 속성 / 기능

객체는 속성과 기능, 두 종류의 구성요소로 이루어져 있다.

일반적으로 객체는 다수의 속성과 다수의 기능을 가진다.

멤버 (member) : 객체가 가지고있는 속성과 기능

속성 (property) : **멤버변수(member variable)**, 특성(attribute), 필드(field), 상태(state)


- 프로그래밍 시 변수로 표현한다.

기능 (function) : **메서드(method)**, 행위(behavior), 함수(function)

- 프로그래밍 시 함수로 표현한다.

객체의 속성과 기능은 위에처럼 다양한 용어로 불리지만 보통은 멤버변수 / 메서드라고 주로 부른다.

예를 들면,

 TV를 객체지향 느낌으로 재작성 해보면 요런느낌이 된다.

클래스	설계도면
객체	TV
멤버변수(속성)	크기, 길이, 높이, 색깔, 영상, 소리
메서드(기능)	전원키기/ 전원 끄기/ 소리 키우기/ 소리 줄이기/ 채널 변경 ...

느낌이 조금 오는가? 그러면 이것을 java 로 한 번 작성 해보자.

원지 모르겠지만, 일단 보자.

```

1  package javaExampleCode;
2
3  class Tv
4  {
5      String color;
6      boolean power;
7      int channel;
8
9      void power() {
10         power = !power;
11     }
12     void channelUp() {
13         ++channel;
14     }
15     void channelDown() {
16         --channel;
17     }
18 }
19
20 public class TvTest {
21     public static void main(String[] args) {
22         Tv t;
23         t = new Tv();
24         t.channel = 7;
25         t.channelDown();
26         System.out.println("현재 보고계시는 채널은 "+t.channel+"번 입니다.");
27     }
28 }

```

1. 클래스 생성

객체지향 프로그래밍에서는 속성과 기능을 각각 변수와 함수로 표현한다. (코드의 5~17번째 줄)

- 멤버변수(속성) : 변수로 표현

```

1  int channel;

```

멤버변수는 각 특징에 맞는 자료형을 선택해야 한다. Tv채널의 경우 6번, 7번 이런 식으로 표현되기 때문에 `int` 형을 선택했다.

만약, '사람'을 객체로 표현하고 싶었다고 치면 그 사람의 나이는 `int` 로, 이름은 `String` 으로 선택하는 식으로 해야 한다.

- 메소드(기능) : 함수로 표현

```

1 void power() {
2     power = !power;
3 }

```

함수 작성법은 나중에 더 자세하게 설명하겠지만, 대충 이런 모양새를 가진다. `void`는 함수의 리턴(결과값)이 없다는 것을 표현한다.

2. 인스턴스의 생성과 사용

클래스를 생성했으니, Tv클래스의 인스턴스(객체)를 만들어 사용해보자.

클래스를 생성하는 방법은 아래와 같다.(코드 22~23)

```

1 클래스명 변수명; //클래스의 객체를 참조하기 위한 참조변수를 선언한다.
2 변수명 new 클래스명(); //클래스의 객체를 생성하고, 객체의 주소를 참조변수에 저장한다.

```

요거를 예제에서 보면,

```

1 Tv t; //Tv클래스의 참조변수 t를 선언한다.
2 t = new Tv(); //Tv인스턴스를 생성한 후, 생성된 인스턴스의 주소를 t에게 준다.

```

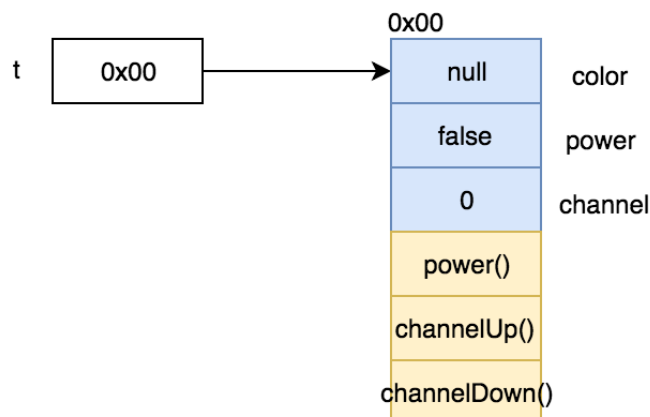
다시, 잘 설명해보자.

- `Tv t;`

t

Tv클래스 타입의 참조변수 t를 선언한다. 메모리에 그림처럼 t를 위한 공간이 마련된다.

- `t = new Tv();`

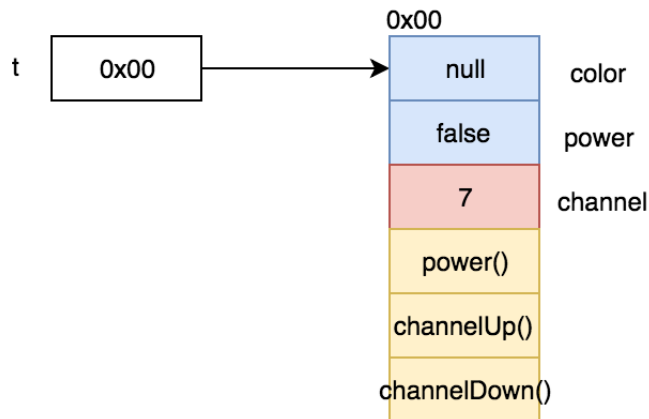


연산자 `new`에 의해 Tv클래스의 인스턴스가 메모리의 빈 공간에 생성된다. 0x00

객체의 멤버변수와 함수가 생성되고 멤버변수는 각 자료형에 맞는 값으로 초기화된다. null/false/0

대입연산자 = 에 의해 생성된 참조변수 t 에 인스턴스에 대한 주소값 0x00이 저장된다. 이제, t에 담겨있는 주소값을 보고 객체를 찾아가는 것이다.

- `t.channel = 7;`



t에 저장된 주소에 있는 인스턴스의 멤버변수 channel에 7을 대입한다. 그러면 0으로 초기화 되어있던 채널값이 7로 변한다.

**인스턴스의 멤버변수를 사용하려면 참조변수.멤버변수 t.channel 이렇게 쓰면 된다.

- `t.channelDown();`

참조변수 t가 참조하고 있는 Tv인스턴스의 channelDown(); 함수를 호출한다. 그러면 t.channel값이 얼마게요?

여기서 포인트는, 인스턴스는 오직 참조변수를 통해서만 다룰 수 있으며, 참조변수의 타입 Tv 은 인스턴스의 타입 Tv과 같다.

-

