1. 객체지향언어

● 객체지향의 기본 개념: 실제 세계는 사물(객체)로 이루어져 있으며, 발생하는 모든 사건들은 사물간의 상호작용이다. 실제 사물의 속성과 기능을 분석한 다음, 데이터(변수)와 함수로 정의함으로써 실제 세계를 컴퓨터 속에 옮겨 놓은 것과 같은 가상 세계를 구현하고, 이 가상세계에서 모의실험을 함으로써 많은 시간과 비용을 절약한다. 프로그램의 규모가 커지고 사용자의 요구가 빠르게 변화해가는 상황을 절차적 언어로는 극복하기 어렵다는 한계가 있어 객체지향언어를 이용한 개발방법론이 대안으로 떠올랐다. 객체지향언어는 코드 간에 서로 관계를 맺어줌으로써 보다 유기적으로 프로그램을 구성할 수 있다.

● 객체지향언어의 주요특징
1. 코드의 재사용성이 높다: 새로운 코드를 작성할 때 기존의 코드를 이용하여 쉽게 작성할 수 있다.
2. 코드의 관리가 용이하다: 코드간의 관계를 이용해서 적은 노력으로 쉽게 코드를 변경할 수 있다.
3. 신뢰성이 높은 프로그래밍을 가능하게 한다: 제어자와 메서드를 이용해서 데이터를 보호하고 올바든 값을 유지하도록 하며, 코드의 중복을 제거하여 코드의 불일치로 인한 오작동을 방지할 수 있다.
- 코드의 재사용성이 높고 유지보수가 용이하여 프로그램의 개발과 유지보수에 드는 시간과 비용을 획기전으로 개선할 수 있다.

2. 클래스와 객체

● 클래스

클래스(Class)	정의	객체를 정의해놓은 것 (객체의 설계도 또는 틀)
	용도	객체를 생성하는데 사용

● 객체

객체 (Object)	정의	실제로 존재하는 것. 사물 또는 개념.
	용도	객체가 가지고 있는 기능과 속성에 따라 다름.
	유형의 객체 Example	책상, TV 의자, 자동차 같은 사물
	무형의 객체 Example	수학공식, 프로그램 에러와 같은 논리나 개념

- 프로그래밍에서는 먼저 클래스를 작성한 다음, 클래스로부터 객체를 생성하여 사용한다. 객체를 사용한다는 것은 객체가 가지고 있는 속성과 기능을 사용한다는 것이다.
- 예를 들면 제품 설계도가 클래스라면 제품 설계도는 제품(객체)를 정의한 것이고, 제품(객체)를 만드는데 사용된다. 하나의 설계도를 잘 만들어놓으면 설계도대로만 만들면 되기 때문에 제품을 만드는 일이 쉬워진다. 이와 마찬가지로 클래스를 한번 잘 만들어놓으면 매번 객체를 생성할 때마다 어떻게 객체를 만들어야 할지 고민하지 않아도 된다.

- JDK(Java Development Kit)에서는 프로그래밍을 위해 많은 수의 유용한 클래스(Java API)를 기본적으로 제공하고 있으며, 이 클래스를 이용하면 원하는 기능의 프로그램을 보다 쉽게 작성할 수 있다.

▶ 객체와 인스턴스

클래스의 인스턴스화(instantiate)	클래스로부터 객체를 만드는 과정	
인스턴스(instance)	어떤 클래스로부터 만들어진 객체. ~클래스의 인스턴스다. 라고 표현한다.	객체에 비교해, 어떤 클래스로부터 만들어진 것인지를 강조하는 보다 구체적인 의미를 갖고 있다.
객체(Object)	실제로 존재하는 것, 사물 또는 개념. ~은 객체다. 라고 표현한다.	인스턴스와 비교해, 모든 인스턴스를 대표하는 포괄적인 의미를 갖고 있다.

▶ 객체의 구성요소 (속성과 기능, 클래스 내부에 정의)

· 객체의 구성요소: 속성과 기능. 일반적으로 객체는 다수의 속성과 다수의 기능을 가진다. 즉, 객체는 속성과 기능의 집합이다.

객체의 구성요소 (객체의 멤버)	다른 용어
속성(property)	멤버변수(member variable), 특성 (attribute), 필드 (field), 상태 (state)
기능(function)	메서드(method), 함수(function), 행위 (behaviour)

· 객체의 멤버(구성원, member): 객체가 가지고 있는 속성과 기능
· 클래스란 객체를 정의한 것이므로 클래스에는 객체의 모든 속성과 기능이 정의되어 있다. 클래스로부터 인스턴스를 생성하면, 클래스에 정의된 속성과 기능을 가진 객체가 만들어진다. 일반적으로 멤버변수를 메서드보다 먼저 선언하고, 멤버 변수는 멤버변수끼리 메서드는 메서드끼리 모아 놓는다.

```
// 클래스 생성 예시. Tv클래스엔 Tv(객체)의 모든 속성과 기능이 정의돼 있다.

class Tv {
    String color;          // Tv클래스의 멤버변수 (속성)
    boolean power;
    int channel;

    void power() { power != power; }          // Tv클래스의 메서드 (기능)
    void channelUp () { channel++; }
    void channelDown () { channel--; }
}
```

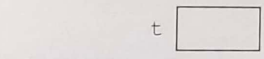
▶ 인스턴스의 생성과 사용: 인스턴스는 오직 참조변수를 통해서만 다룰 수 있으며, 때문에 인스턴스를 사용하려면, 같은 클래스 타입의 참조변수가 필요하다. 참조변수의 타입은 인스턴스의 타입과 일치해야 한다.

```
클래스명 변수명;          // 클래스의 인스턴스를 참조하기 위한 참조변수 선언
변수명 = new 클래스명();          // 클래스의 인스턴스를 생성 후, 그 주소를 참조변수에 저장

Tv t;          // Tv클래스 타입의 참조변수 t선언
t = new Tv();        // Tv 인스턴스를 생성 후, 생성된 Tv인스턴스의 주소를 참조변수 t에 저장.
// 참조변수 t가 가리키고(참조하고) 있는 인스턴스를 간단히 인스턴스 t라고 한다.
```

```
// 인스턴스 생성하여 인스턴스의 속성과 메서드를 사용하는 예제

class Tv {
    String color;          // Tv클래스의 멤버변수 (속성)
    boolean power;
    int channel;

    void power() { power != power; }          // Tv클래스의 메서드 (기능)
    void channelUp () { channel++; }
    void channelDown () { channel--; }
}

class TvTest {
    public static void main(String args[]) {
        Tv t;    // Tv인스턴스를 참조하기 위한 변수 t를 선언
        t = new Tv(); // Tv 클래스의 인스턴스를 생성해서 그 메모리주소를 참조변수 t에 저장.
        t.channel = 7;    // 인스턴스 t의 멤버변수 channel의 값을 7로 한다.
        t.channelDown();    // 인스턴스 t의 메서드 channelDown()을 호출한다.
        System.out.println("현재 채널은 " + t.channel + "입니다.");  // 현재 채널은 6입니다
    }
}

// 메모리 공간의 그림으로 위의 과정을 보자면.
```

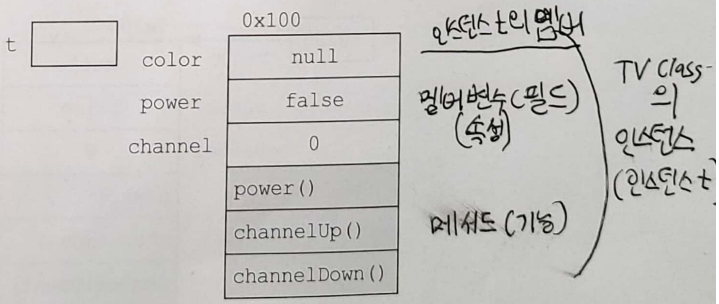
1. Tv t;

Tv클래스 타입의 참조변수 t를 선언한다. 메모리에 참조변수 t를 위한 공간이 마련된다. 아직 인스턴스가 생성되지 않았으므로 참조변수로 아무것도 할 수 없다.



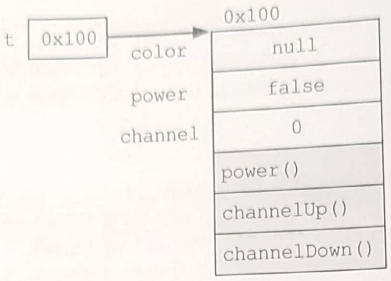
2. t = new Tv();

연산자 new에 의해 Tv클래스의 인스턴스가 메모리의 빈 공간에 생성된다. 주소가 0x100인 곳에 생성되었다고 가정하자. 이 때, 멤버변수는 각 자료형에 해당하는 기본값으로 초기화 된다.
color는 참조형이므로 null로, power는 boolean이므로 false로, 그리고 channel은 int이므로 0으로 초기화 된다.



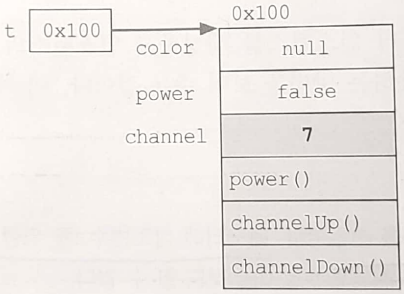
그 다음에는 대입연산자(=)에 의해서 생성된 객체의 주소값이 참조변수 t에 저장된다. 이제는 참조변수 t를 통해 Tv인스턴스에 접근할 수 있다. 인스턴스를 다루기 위해서는 참조변수가 반드시 필요하다.

| 참고 | 아래 그림에서의 화살표는 참조변수 t가 Tv인스턴스를 참조하고 있다는 것을 알기 쉽게 하기 위해 추가한 상징적인 것이다. 이 때, 참조변수 t가 Tv인스턴스를 '가리키고 있다' 또는 '참조하고 있다'라고 한다.



3. `t.channel = 7 ;`

참조변수 t에 저장된 주소에 있는 인스턴스의 멤버변수 channel에 7을 저장한다. 여기서 알 수 있는 것처럼, 인스턴스의 멤버변수(속성)를 사용하려면 '참조변수.멤버변수'와 같이 하면 된다.

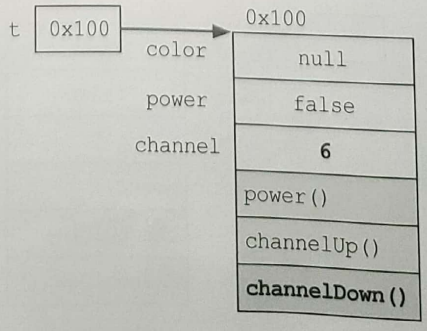


4. `t.channelDown();`

참조변수 t가 참조하고 있는 Tv인스턴스의 channelDown메서드를 호출한다. channel Down메서드는 멤버변수 channel에 저장되어 있는 값을 1 감소시킨다.

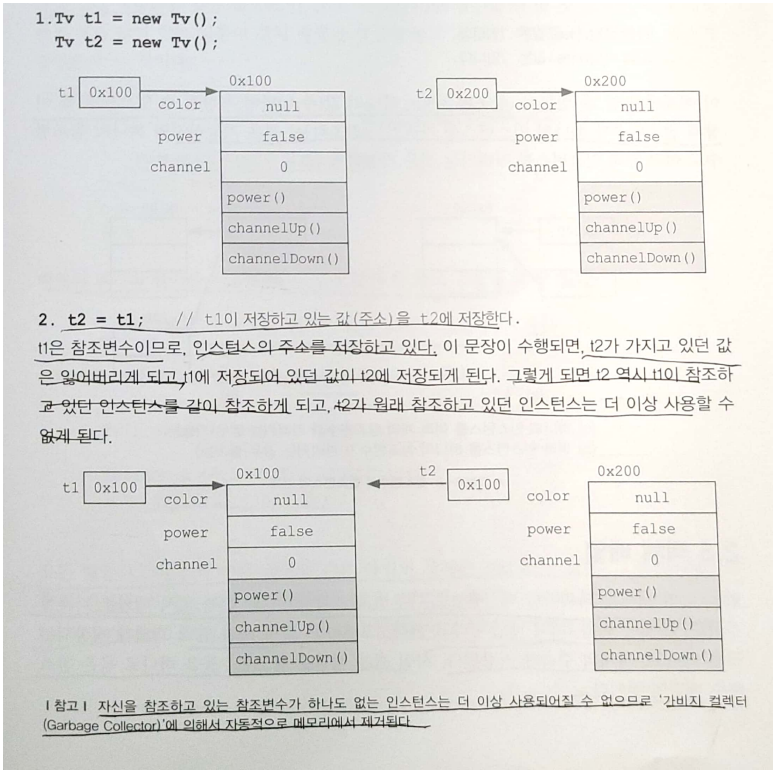
```
void channelDown() {  --channel;  }
```

channelDown( )에 의해서 channel의 값은 7에서 6이 된다.



· 같은 클래스로부터 생성되었을지라도 인스턴스의 속성(멤버변수)은 서로 다른 값을 유지할 수 있으며, 메서드의 내용은 모든 인스턴스에 대해 동일하다.

· 참조변수에는 하나의 값(주소)만이 저장될 수 있다. 둘 이상의 참조변수가 하나의 인스턴스를 가르키는(참조하는) 것은 가능하다.



▶ 객체 배열: 참조 변수 배열이다. 많은 수를 객체를 다룰 때 배열로 다루는 것이 가능하다. 객체 배열 안에 객체의 주소가 저장된다. 즉, 객체 배열은 참조 변수들을 하나로 묶은 참조 변수 배열이다.

```
Tv[] tvArr = new Tv[3];     // 크기가 3인 객체 배열 tVArr (참조변수 배열) 생성.
// 각 요소는 참조변수의 기본값인 null로 자동 초기화 된다. tvArr[0-3] = null
// 이 객체 배열은 3개의 객체, 정확히는 객체의 주소를 저장할 수 있다.

// 객체 배열을 생성한다는 것은 객체를 다루기 위한 참조 변수들이 만들어지는 것이라서,
// 객체를 생성해서 객체 배열의 각 요소에 저장하는 것을 잊으면 안 된다.
// 객체를 생성해서 배열의 각 요소에 저장
tvArr[0] = new Tv();
tvArr[1] = new Tv();
tvArr[2] = new Tv();
// 배열의 초기화 블럭을 사용하면, 객체 배열과 객체 생성과 저장을 한 줄로 할 수 있다.
Tv[] tvArr = { new Tv(), new Tv(), new Tv() };
// 객체의 수가 많을 때는, for문을 사용하면 된다.
Tv[] tvArr = new Tv[100];

for (int i = 0; i < tvArr.length; i++) {
    tvArr[i] = new Tv();
}

// 모든 배열이 그렇듯 객체 배열도 같은 타입의 객체만 저장할 수 있지만,
// 다형성(polymorphism)을 통해 하나의 배열로 여러 종류의 객체를 다룰 수 있게 된다.
```

▶ 클래스의 프로그래밍관점적 정의와 의미

1) 클래스: 구조체와 함수의 결합

- 데이터 처리를 위한 데이터 저장형태의 발전 과정

변수	하나의 데이터를 저장할 수 있는 공간	하나의 데이터를 저장하기 위해
배열	같은 종듀의 여러 데이터를 하나의 집합으로 저장할 수 있는 공간	같은 종듀의 데이터를 보다 효율적으로 다두기 위해
구조체	서로 관련된 여러 데이터를 종듀에 관계없이 하나의 집합으로 저장할 수 있는 공간	자료형의 종듀와 상관없이 서로 관계가 깊은 변수들을 하나로 묶어서 다두기 위해
클래스	데이터와 함수의 결합 (구조체 + 함수)	서로 관계가 깊은 변수와 함수들을 함께 다두기 위해.

클래스는 서로 관련된 변수들을 정의하고 이들에 대한 작업을 수행하는 함수들을 함께 정의한 것이다. 예를 들면 자바에서는 String이라는 클래스로 문자열을 다두는데, 문자열을 단순히 문자의 배열로 정의하지 않고 클래스로 정의한 이유는 문자열과 문자열을 다두는데 필요한 함수들을 함께 묶기 위해서이다. String 클래스 내부에 value라는 문자형 배열이 선언돼 있고, 문자열의 길이를 알아내는 함수 등은 항상 문자열을 대상으로 필요로 하기 때문에 문자열과 깊은 관게에 있으므로 함께 정의되었다.

2) 클래스: 사용자 지정 타입(user-defined type)
- 기본 자료형 (primitive type)외에 프로그래머가 서로 관련된 변수들을 묶어서 하나의 타입으로 새로 추가하는 것을 사용자정의 타입(user-defined type)이라고 한다. 클래스가 곧 사용자 정의 타입이다. 기본형의 개수는 8개로 정해져 있는 반면 참조형의 개수가 정해져 있지 않은 이유는 이처럼 프로그래머가 새로운 타입을 추가할 수 있기 때문이다.
- 클래스를 정의하면 멤버 변수들을 하나의 단위로 묶어서 다두기 때문에 다든 변수의 데이터와 섞이는 일도 발생하지 않고, 제어자와 메서드를 이용해 추가적인 제약 조건들을 코드에 쉽게 반영할 수 있다 (ex. 예를 들면 제어자를 이용해서 변수의 값을 직접 변경하지 못하도록 하고, 대신 메서드를 통해서 값을 변경하도록 작성한 다음 값을 변경할 때 지정된 값의 유효성을 조건문으로 점검한 다음 유효한 값일 경우에만 변경하도록 할 수 있다).

Resource: 자바의 정석