**객체** (클래스의 인스턴스)

┌→ 데이터(=변수)

- 상태 = 객체의 속성 🡪 **필드**(인스턴스 변수) ex) 채널, 볼륨, 전원

- 동작 = 객체의 기능 🡪 **메소드** ex) 채널 변경, 볼륨 변경, 전원 켜기, 전원 끄기

└→ 알고리즘 (특정한 작업 수행)

* **캡슐화**

- 객체는 데이터와 알고리즘으로 이루어진 하나의 캡슐

- 서로 관련되어 있는 데이터와 알고리즘을 묶어 캡슐화

* 코드 재수정 없이 재사용하기 위함
* **정보 은닉**

- 객체의 실제 구현 내용을 외부로부터 감추는 것 (객체 내부 보호)

- 사용자는 외부로 공개된 몇 개의 메소드만 사용해야 함

* **상속**

- 기존의 코드 재사용

- 기존의 코드 = 부모 클래스

새로운 코드 = 자식 클래스 🡪 부모 클래스의 모든 속성과 동작을 물려받음.

기능 추가, 변경 가능

* 코드 중복을 없애기 위함
* **다형성**

- 상황에 따라 객체가 취하는 동작이 달라지는 것

- 서로 다른 타입에 속하는 객체들이 같은 이름의 멤버 함수에 응답하여 서로 다른 동작을

나타내는 것이 가능

* **추상화**

- 불필요한 정보는 숨기고, 중요한 정보만 표현하는 것

- 복잡성을 관리함

**클래스**

- 객체의 형태를 정하는 틀

- 클래스로부터 만들어진 객체를 클래스의 인스턴스라고 함

Ex) **class 클래스 이름 {**

**자료형 필드1 ;**

**자료형 필드2 ;**

**반환형 메소드1( ) { }**

**반환형 메소드2( ) { }**

**}**

**객체 생성**

Ex) **Television tv = new Television( );**

객체로 만들 클래스 이름, 참조 변수 선언, =, new 연산자, 객체로 만들 클래스 이름 ( );

**참조 변수**

- 객체의 참조 값(객체의 주소) 저장

- 객체가 직접 저장되는 것이 아닌 주소를 가리키는 화살표가 저장되어 있는 것

**객체 멤버 접근**

Ex) **tv . channel = 7;**

참조 변수 이름, “ . “, 접근할 필드 이름 ;

**메소드**

- 클래스 안에 정의된 함수

- 특정한 작업을 수행하는 문장들의 모임

- 메소드 이름의 첫 글자는 소문자+동사로 시작, 두 번째 단어부터 대문자+명사/형용사 가능

Ex) **int add (int x, int y) { }**

반환형(데이터 타입), 메소드 이름(변수 이름), 매개변수

- 필요한 데이터를 전달받을 수 있음

- 결과를 호출자에게 반환할 수 있음 (반환하지 않는 반환형: void)

**인수 (argument)**

- 메소드에 보내는 값

**매개변수 (parameter)**

- 사용하는 변수

**메소드 오버로딩 (method overloading)** (중복 정의 or 다중 정의)

- 이름이 같은 메소드를 여러 개 정의하는 것

- 각각의 데이터 타입과 받는 매개변수의 개수 또는 자료형이 달라야 함.

- 다형성을 구현하는 방법 중 하나

**String 클래스**

- 문자열(String)은 문자들의 나열(sequence)

- String은 기본 자료형이 아닌 문자열을 저장하고 처리하는 클래스가 존재하는 객체

Ex) **String s = new String(“Hello World!”) ;** // 선언과 동시에 초기화

**s = “Hello World!” ;** // new 연산자 생략 가능