Object Oriented Programming

Jonghwa Park
suakii@gmail.com

0. OOP

1. 소프트웨어의 위기
   a. 소프트웨어의 개발 생산성이 높아지지 못하였기 때문에 발생함.
2. 객체지향
   a. 클래스와 객체를 기반으로 한 새로운 프로그래밍 접근 방법
   b. 객체는 상태와 행동으로 구성됨

0. OOP

1. 객체지향의 개념
    a. 객체지향이란 현실 세계의 객체 모델을 바탕으로 프로그램을 구조화하고 개발하는 프로그래밍 기법.

    b. 프로그래밍 언어는 크게 구조적 특징에 따라 객체지향 프로그래밍 언어(Object-Oriented Programming Language)와 절차지향 프로그래밍 언어(Procedure-Oriented Programming Language)로 나눈다.

0. OOP

1. 절차지향 프로그래밍 언어
   a. 절차지향 프로그래밍 언어는 프로시저Procedure(프로그램 처리 절차)의 호출 개념에 바탕.
   b. C언어의 함수와 같은 형태.
   c. 프로시저는 전체 애플리케이션을 구성하는 부분을 말하는 것으로, 필요한 기능을 구현한 각각의 함수를 호출한 결과로 다음 작업을 진행하는 구조.
   d. C언어, 포트란Fortran, 베이직Basic 등.

0. OOP

1. 객체지향 프로그래밍 언어
    a. 프로그래밍을 현실 세계의 시각으로 접근
    b. 객체는 하나의 독립적인 개체로, 함수와 유사한 메서드(Method)와 속성으로 구성
    c. 상속, 추상화, 캡슐화 등 객체지향의 특징을 활용한 구조적이고 재활용 가능한 모듈화로 생산성과 유지보수 효율성을 높인 프로그래밍 언어
    d. SmallTalk, C++, C#, Java, Python

0. OOP

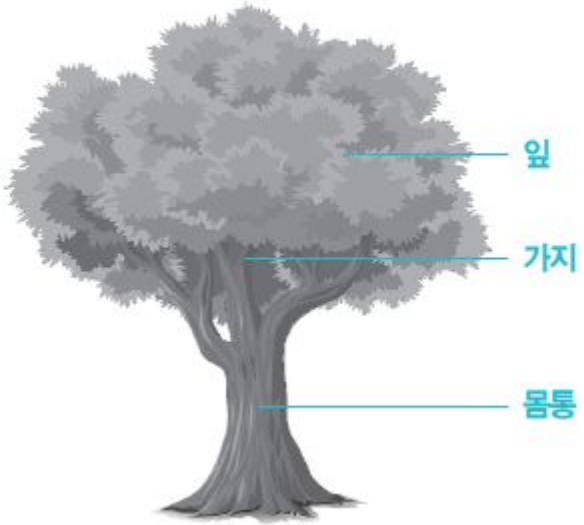
표 4-1 절차지향 프로그래밍 언어와 객체지향 프로그래밍 언어의 주요 요소 비교

분류	절차지향 프로그래밍 언어	객체지향 프로그래밍 언어
호출 단위	함수	메서드
처리 단위	모듈(함수)	객체
데이터를 저장하는 곳	변수	속성
확장	라이브러리	라이브러리, 상속, 추상 클래스, 인터페이스
대표 언어	C언어, 포트란, 베이직 등	스몰토크, C++, C#.Net, 자바, 파이썬 등

0. Object

객체를 뜻하는 ‘Object’는 기본적으로는 ‘사물’이라고 해석된다. 사물은 우리 눈에 보이는 모든 것을 말하며, 그 사물이 가진 속성(Attribute)과 행위(Behavior)로 설명할 수 있는 대상이다.

그림 4-1 객체지향 관점으로 정의한 나무



나무

[속성]

- 잎의 재질 : 섬유질
- 잎의 성질 : 부드럽다.
- 잎의 색상 : 초록색
- 몸통의 성질 : 딱딱하다.
- 몸통의 크기 : 크다. 또는 구체적인 길이

[행위]

- 광합성을 한다.
- 햇빛에 따라 방향을 바꾼다.
- 자라난다.
- 꽃을 피운다.

0. Object

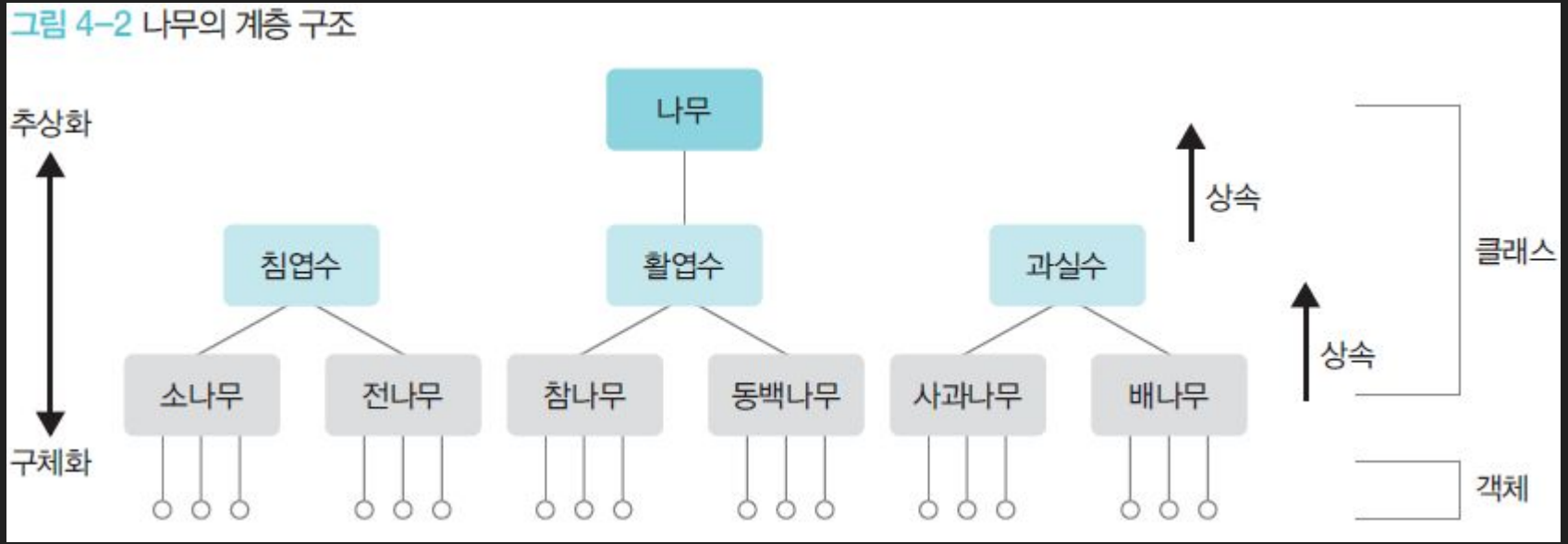
작성 예

앞의 설명에 따라 각 요소를 정리하면 다음과 같다.

번호	객체	속성	행위
1	자동차	제조사, 모델 이름, 색상, 엔진 형식, 마력, 속도 등	전진, 후진, 가속, 감속, 정차, 헤드라이트 On/Off, 와이퍼 On/Off 등
2	토끼	털 색, 키, 귀 크기, 울음소리 등	네 발 걷기, 두 발 뛰기, 잠자기 등
3	전화기	재질, 색상, 유 · 무선 구분, 스피커폰 유무, 다이얼 종류	전화 걸기, 전화 끊기, 다이얼 동작, 볼륨 크게/작게 등

0. OOP

객체를 정의하는 작업은 단순하지 않기 때문에 객체들을 일반화(추상화)하거나 구체화하는 작업을 병행해야 함.



0. OOP

1. Class
    a. 속성이 같은 객체들을 대표할 수 있는 대상을 클래스Class라고 한다.
    클래스는 객체를 정의하는 틀이며, 필드(속성)와 메서드(행위)로 구성.
    클래스는 추상화로 슈퍼 클래스(상위 클래스, 부모 클래스)와 서브 클래스(하위 클래스, 자식 클래스)로 구분.
2. Instance
    a. 인스턴스는 클래스에서 생성한 객체로, 고유한 상태가 있다.

0. OOP

1. 상속
    a. 상속은 클래스를 추상화하거나 구체화하는 과정에서 발생.

    b. 상속을 이용하면 슈퍼 클래스의 기본 구성 요소(필드, 메서드)를 물려받으며(상속), 자신만의 필드나 메서드를 추가하여 구체화하는 것이 가능하다. 물론 물려받은 메서드의 내용을 수정하는 것도 가능하다.

0.OOP

1. 객체지향 프로그래밍은 여러 단위의 객체 간의 협업을 통해 프로그램이 수행하고자 하는 목적을 이루는 방법.

2. 각각의 클래스는 서로 다른 역할을 담당하고 있고, 여러 클래스로 만들어진 객체(object)가 모여서 프로그램을 완성.

3. 각각의 객체는 서로 메시지를 주고 받음.

0. OOP

1. 클래스는 객체를 생성하기 위한 청사진.
   a. 객체 상태를 저장하는데 사용될 속성을 정의

      운전대 클래스: 좌/우로 움직임. 버튼이 하나 있고 누르면 소리가 남.

      → 차종에 관계없이 비슷한 기능을 제공, 그러나 차종에 따라 추가되는 속성이 있을 수 있음 (예: 오디오 콘트롤 버튼)

      엔진 클래스: 회전을 하여 동력을 바퀴로 전달. 멈춤 속성과 시동 속성이 있음.
2. 객체를 이해할 수 있는 메시지와 메시지에 응답하는 과정을 정의 (메소드의 역할)
   a. 엔진 클래스: 엑셀 오브젝트로 부터 메시지가 전달되면 엔진 시동 메소드를 실행하여 엔진을 움직임. 브레이크 오브젝트로부터 메시지가 전달되면 엔진 멈춤 메소드를 실행.

      이 과정에서 클래스가 내부적으로 어떻게 움직이는지 클래스 외부에서 알 필요가 없음 → 예: String에서 sort 메소드.

0. nonOOP OOP

```
// Simple non OOP Car
color c;
float xpos;
float ypos;
float xspeed;

void setup() {
  size(200,200);
  c = color(255);
  xpos = width/2;
  ypos = height/2;
  xspeed = 1;
}

void draw() {
  background(0);
  display();
  drive();
}

void display () {
  rectMode(CENTER);
  fill(c);
  rect(xpos,ypos,20,10);
}

void drive() {
  xpos = xpos + xspeed;
  if (xpos > width) {
    xpos = 0;
  }
}
```

→

```
class Car {
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() {
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
}
```

→ The class name

→ Data

→ Constructor

→ Functionality

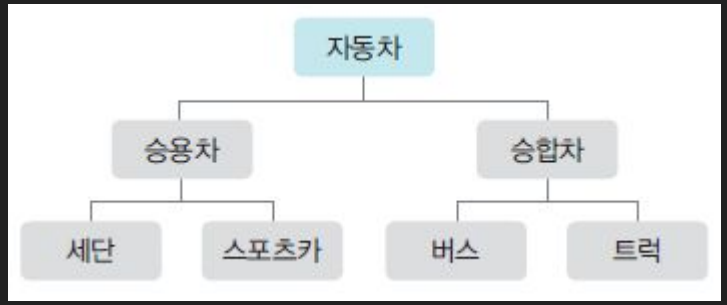
0. OOP

```
// Step 1. Declare an object.
Car myCar;

void setup()  {
  // Step 2. Initialize object.
  myCar = new Car();
}

void draw()  {
  background(255);
  // Step 3. Call methods on the object.
  myCar.drive();
  myCar.display();
}
```

0. OOP 상속

1. 상속은 클래스를 추상화하거나 구체화하는 과정에서 발생.

2. 상속을 이용하면 슈퍼 클래스의 기본 구성 요소(필드, 메서드)를 물려받으며(상속), 자신만의 필드나 메서드를 추가하여 구체화하는 것이 가능하다. 물론 물려받은 메서드의 내용을 수정하는 것도 가능하다.



0. OOP

Overriding: 클래스간 상속 관계에서 메서드를 재 정의

다형성: 똑같은 행위를 해도 객체마다 방식이 다른 것을 다형성, 하나의 참조 변수로 자손 타입의 객체를 참조할 수 있는 것

Overloading:한 클래스 내에서 동일한 이름의 메서드를 추가 정의