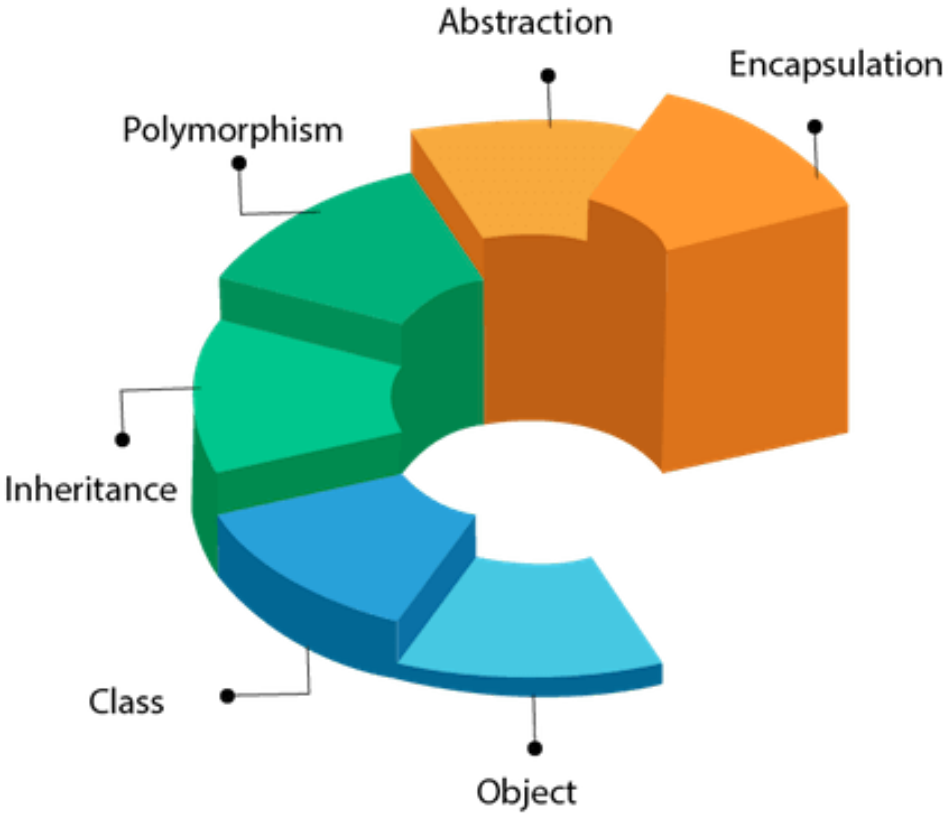
Chapter.01 파이썬 프로그래밍-08. OOP

💡 OOP : Object-Oriented Programming의 약자로 객체(Object)를 기반으로 프로그램을 만드는 방법론입니다.



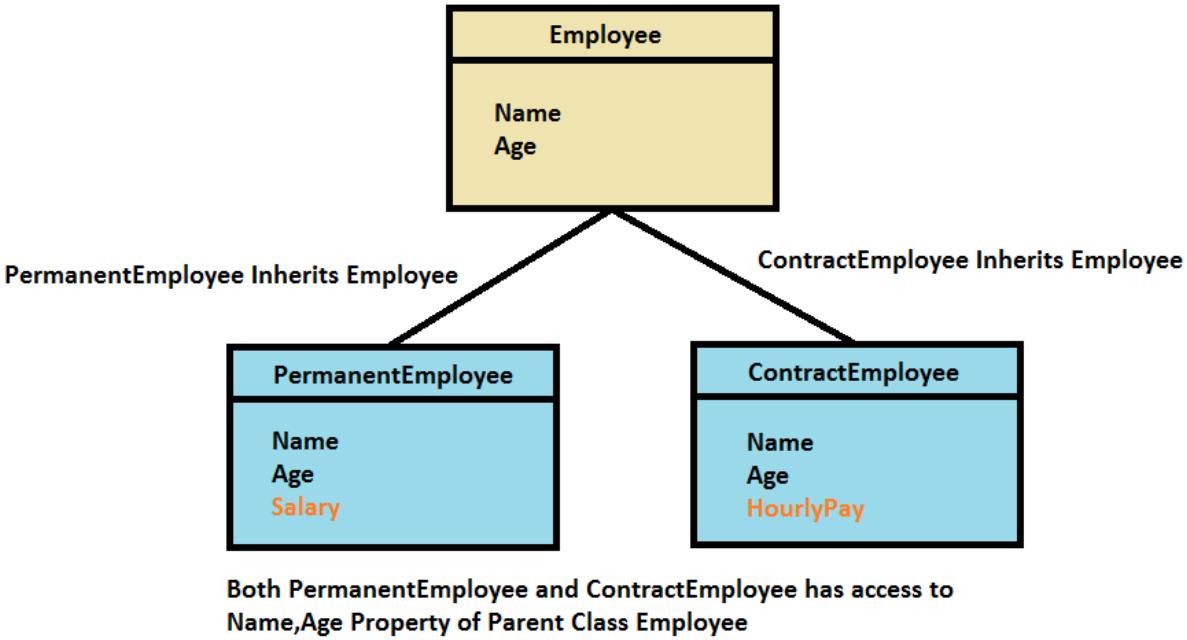
Source : https://velog.io/@gil0127/Object-Oriented-Programming-in-Javascript

- OOP는 실제 세상을 표현하고 있는, 여러가지 개념들을 프로그램으로 옮겨서 구현할 수 있기 위한 컨셉의 프로그래밍 패러다임(paradigm)입니다.
  → **개념의 추상화(abstraction)**

- 객체(object)라는 개념은 실제 사물 하나하나를 의미할 수 있고, 이러한 사물들이 공유하는 속성을 정의한 것을 클래스(Class)라고 합니다.
- 기존 방식은 데이터(변수, variable)와 데이터를 처리하는 기능(함수, function)이 독립적이었지만, OOP는 이를 하나의 개념(Class)로 묶어서 생각하기 때문에 그 객체가 처리되는 기능을 자연스럽게 정의할 수 있습니다.
- Class에는 Class를 기술하는 정보를 나타내는 변수인 `Class variable`과 Class의 특징을 설명하는 기능인 `Class method`를 포함합니다.
- 추상적인 개념 Class를 실제로 사용하려면 하나 하나의 개별 사물로 만들어야 하는데, 이를 개별 사물인 객체(object)라고 합니다.
  → **개념의 구체화(instantiation)**
- 위와 같은 개념을 살려 프로그램을 만드는 방식은 OOP의 장점은 다음과 같습니다.

1. 클래스 구조를 잘 설계하면 라이브러리 형태로 재사용이 쉬워진다.
   → **생산성 향상**
2. 일상 생활에 존재하는 개념을 그대로 프로그램에 구현 가능하다.
   → **자연적인 모델링**
3. 클래스의 상속의 개념 때문에, 프로그래밍 자체의 재사용성이 극대화된다.
   → **재사용성 증가**
4. OOP를 이용하여 개발을 하게되면, 다른 기능을 수정하더라도 클래스가 서로 다르게 구현되어 있어 다른 기능에 끼치는 영향이 매우 적어질 수 있다.
   → **유지보수 용이성 증가**

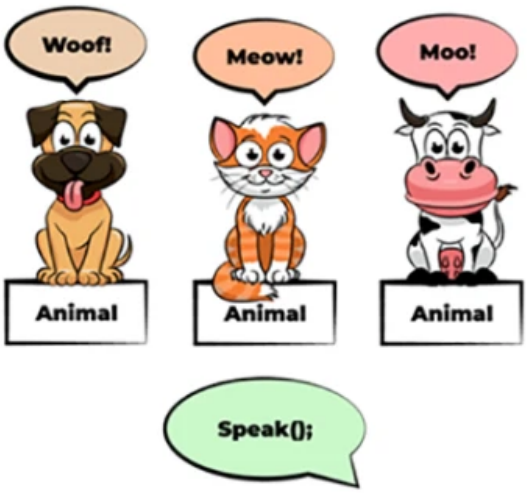
1. Inheritance(상속)



Source : https://masterdotnet.com/csharptutorial/csharpinheritance/

- 클래스는 개념의 추상화이기 때문에, 해당 개념을 **계승**하는 하위 개념을 만들 수 있습니다.
- 상위/하위 개념이 상대적으로 존재하며, 상속하는 클래스는 `superclass`, 상속받는 클래스는 `subclass`라고 얘기합니다.
- subclass는 superclass의 모든 개념을 이어받기 때문에, **class variable**, **class method** 도 그대로 이어받습니다.

2. polymorphism(다형성)



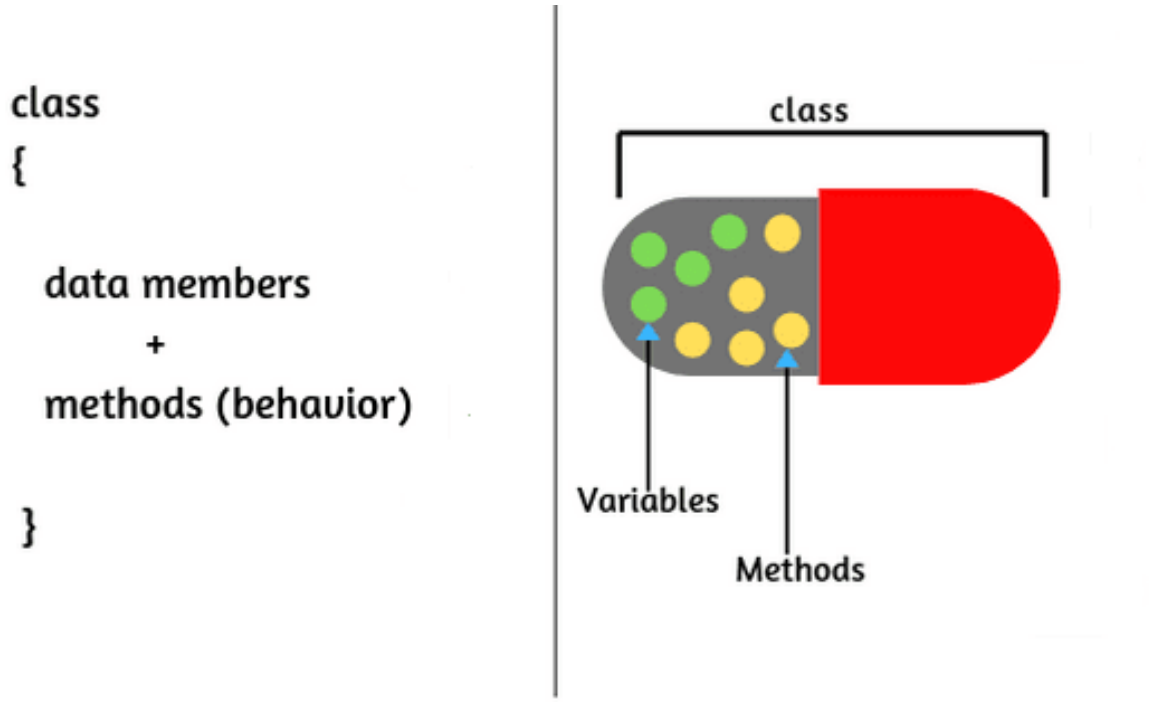
Source : https://codegym.cc/groups/posts/polymorphism-in-java

- 여러 하위 클래스가 같은 class method를 상속받게 되면, 그 기능을 다르게 구현할 수 있습니다.
- 예를 들어 Animal 이라는 Class에 Speak()이라는 method가 있다면, 이 기능은 다른 동물을 표현하는 subclass들마다 다르게 구현될 수 있습니다.
  → Dog : “Woof” , Cat : “Meow” , Cow : “Moo”
- 이렇게 하나의 기능을 나타내는 개념이 실제 구현해서 다양한 형태로 표현 가능한 것을 `Polymorphism`이라고 합니다.
- 이러한 다형성을 구현할 수 있는 기능을 `Method Overriding` 이라고 합니다.

3. Abstraction(추상화)

- Abstraction(추상화)는 Class 내부에 구현된 Class variable이나 Class method를 직접 보지 않아도 개념상으로 사용할 수 있게 하는 개념입니다.
- 기능에 대한 명세나 변수의 의미만 확실하게 알면, 내부 구현은 살펴보지 않아도 됩니다.

4. Encapsulation(은닉화)



Source : https://medium.com/javarevisited/why-should-encapsulation-to-be-used-e82a81f5c47c

- Encapsulation은 Class variable과 Class method까지 단일 개념으로 구성되어 있어, 사용자가 개념 구현의 혼선을 막고 심플하게 사용할 수 있게 만드는 특징을 말합니다.
- Encapsulation이 잘되면 사용자는 **Class의 내부 구현 코드를 보지 않아도** 내부 데이터와 기능을 사용하는데 아무런 문제가 없습니다.
- 우리가 사용해왔던 모든 함수들, Data type들의 내부 구현 코드를 보지 않아도 개념적으로 이해하고 사용할 수 있는 이유도 Encapsulation이 잘되기 때문입니다.

e.g. List.append()를 예로 들 수 있습니다.

Key Points

1. OOP는 대규모 코드의 유지/보수를 편리하게 하기 위해서 생긴 프로그래밍 방법론입니다.
2. OOP는 구현하고자 하는 대상을 Class로 정의하여 코드를 구현할 수 있게 해줍니다.
3. 다양한 특징들이 Class를 통해서 구현된 코드가 재사용되기 편하게 해줍니다.
4. **이러한 이유로 잘 만들어진 open-source library들을 손쉽게 이용할 수 있습니다. (**)**