

1. 객체지향 개요

1. 객체지향

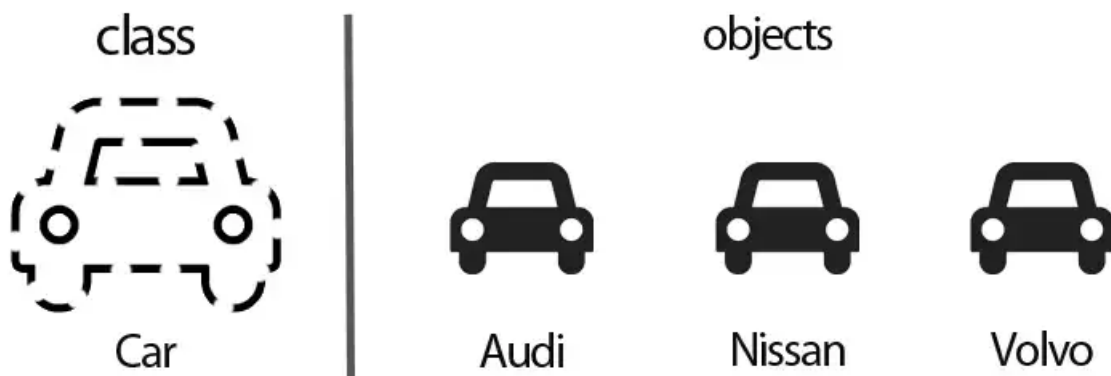
1). 다음과 같은 목적으로 사용하길 바란다

- 재사용과 확장 (응용 프로그램의 상당 부분을 변경하지 않고도 추가하거나 업그레이드).
- 유지보수

2. Class & Instance

1). 객체란?

데이터 (속성, 상태, 변수, 자료구조) 와 함수 (매서드, 동작) 로 구성된것.



ⓐ Class : Type

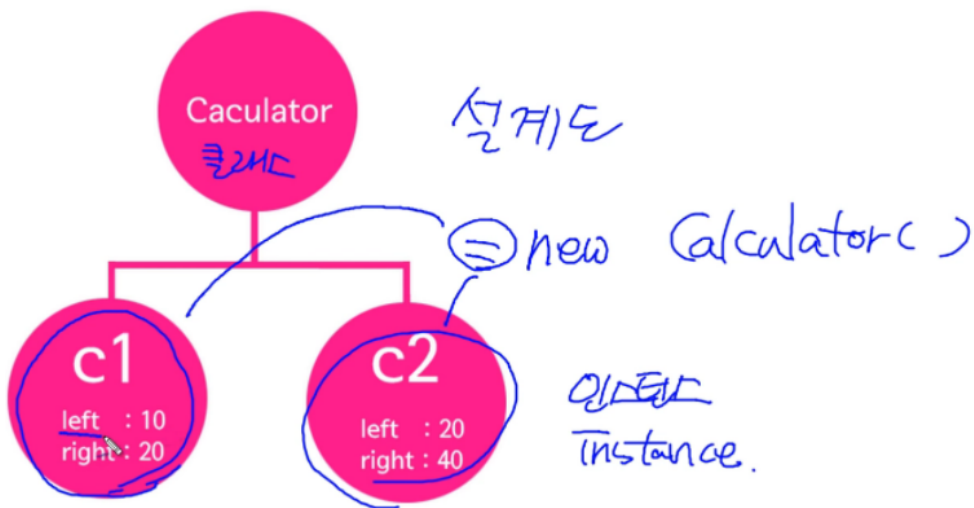
공장이다.

객체의 **집합**이다.

- $SomethingClass \ni SomethingInstance$: Class 집합의 원소로는 Instance가 있다.
- $Car = \{Audi, Nissan, Volvo...\}$

⑥ Instance(Object) : Product

Class가 Type이면 이제 **Type형태로 이루어진 Product**이다.



📖 2. 주요 객체지향 개념

📖 1. 캡슐화

- 결국 코드에서 쓸 때는 **데이터와 함수를 중괄호로 감싼 행위**
- 리모컨 버튼이 어떻게 작동하는지 몰라도 리모컨을 사용할 수 있지?
마찬가지로 구현을 몰라도 `.method()` 하면 사용 가능하다.

📖 2. 상속

- "상속 모체인 클래스"의 데이터와 함수를 "상속 하는 클래스"가 물려받는다. (재사용)
- **자식클래스의 멤버의 개수 >= 부모의 멤버개수**

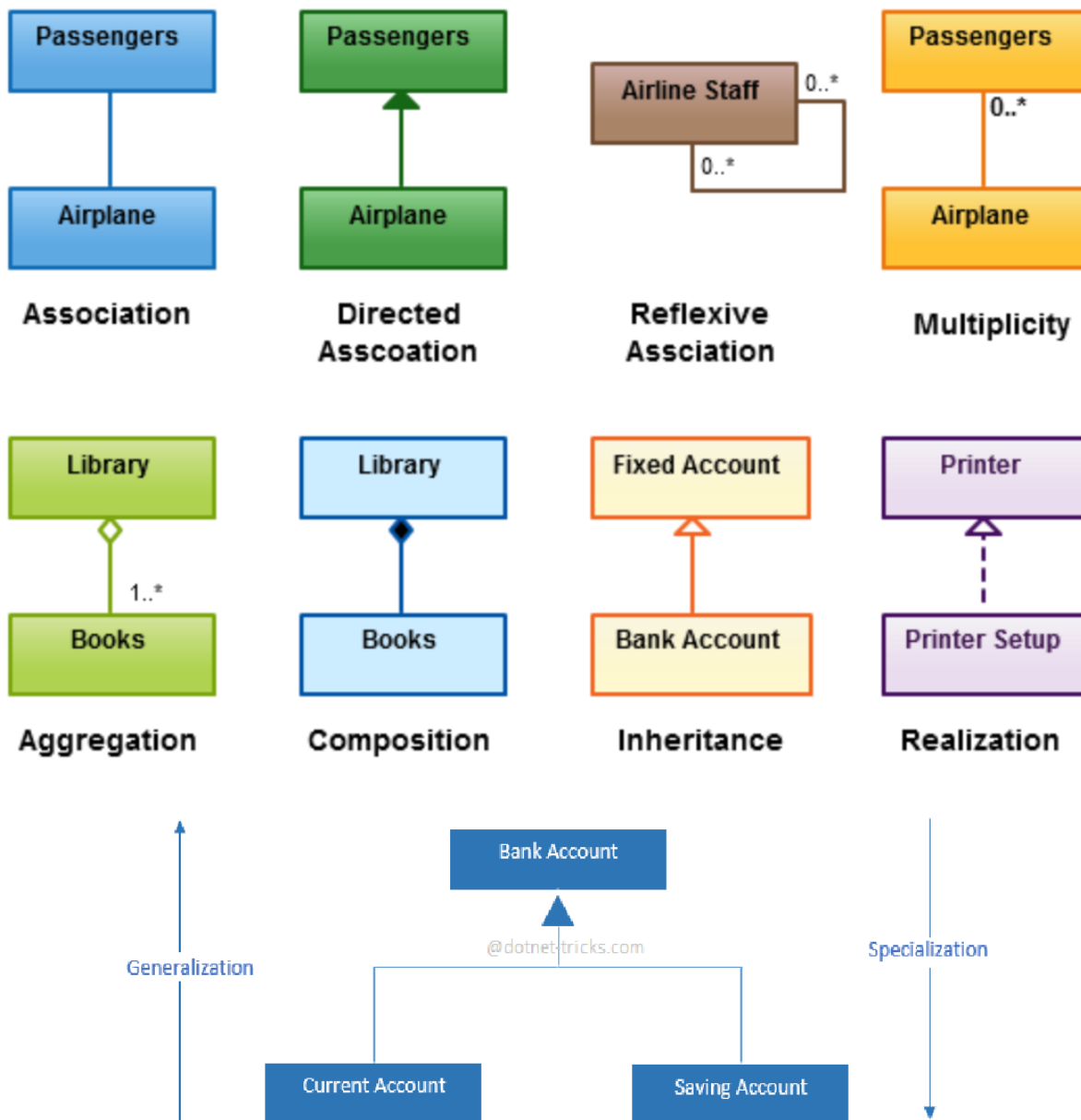
📖 3. 다형성

1. "부모클래스 타입"의 참조 변수로 "자식클래스 타입"의 인스턴스를 참조할 수 있다.
 - C#코드

```
class Parent { ... }  
class Child extends Parent { ... }  
ParentClass P = new ChildClass();
```

2. 그럼 P는 ChildClass만이 가지고있는 멤버에 접근할수 있나?? : ✕
 - ParentClass 타입 변수는 오직 ParentClass 멤버만 접근할 수 있다.
 - 그말인 즉슨 비록 자식이 가지고 있다 하더라도 부모에 구현이 안되어 있으면
 - 쓸수가 없다는 말이다.

4. 연관성



1. **Association** 연관화 : is member of
2. **Classification** 분류화 : is instance of
3. **Aggregation** 집단화 : is part of
4. **Generalization** 일반화 : 공통적인 성질들로 추상화한 상위 객체를 구성하는것
5. **Specialization** 특수화 : 상위 객체를 구체화하여 하위객체를 구성하는것

