기초 중의 기초

- 객체 지향형 프로그래밍이란?

객체지향형 프로그래밍 방법론

객체 지향 프로그래밍은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것이다. 각각의 객체는 메시지를 주고받고, 데이터를 처리할 수 있다.

집을 만들어 봅시다.

- 만약 컴퓨터로 집을 만든다고 한다면 어떻게 해야할까요?

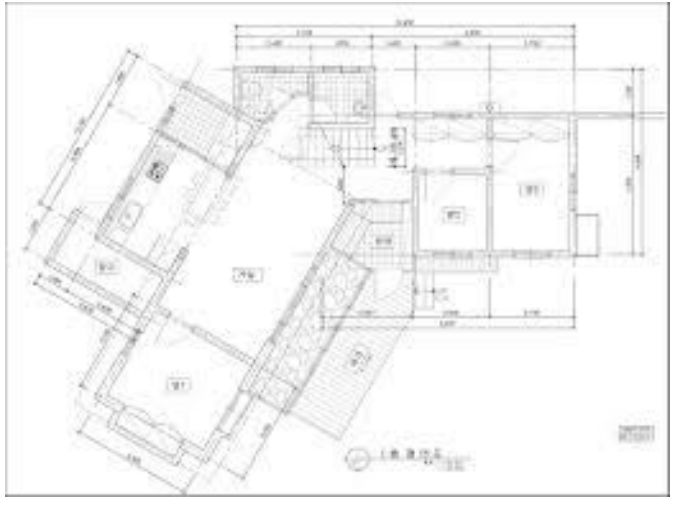
기본 구성 요소

- **클래스(Class)** - 같은 종류(또는 문제 해결을 위한)의 집단에 속하는 속성(attribute)과 행위(behavior)를 정의한 것으로 객체지향 프로그램의 기본적인 사용자 정의 데이터형(user define data type)이라고 할 수 있다. 클래스는 프로그래머가 아니지만 해결해야 할 문제가 속하는 영역에 종사하는 사람이라면 사용할 수 있고, 다른 클래스 또는 외부 요소와 독립적으로 디자인하여야 한다.

- **객체(Object)** - 클래스의 인스턴스(실제로 메모리상에 할당된 것)이다. 객체는 자신 고유의 속성(attribute)을 가지며 클래스에서 정의한 행위(behavior)를 수행할 수 있다. 객체의 행위는 클래스에 정의된 행위에 대한 정의를 공유함으로써 메모리를 경제적으로 사용한다.

- **메서드(Method), 메시지(Message)** - 클래스로부터 생성된 객체를 사용하는 방법으로서 객체에 명령을 내리는 메시지라 할 수 있다. 메서드는 한 객체의 서브루틴(subroutine) 형태로 객체의 속성을 조작하는 데 사용된다. 또 객체 간의 통신은 메시지를 통해 이루어진다.

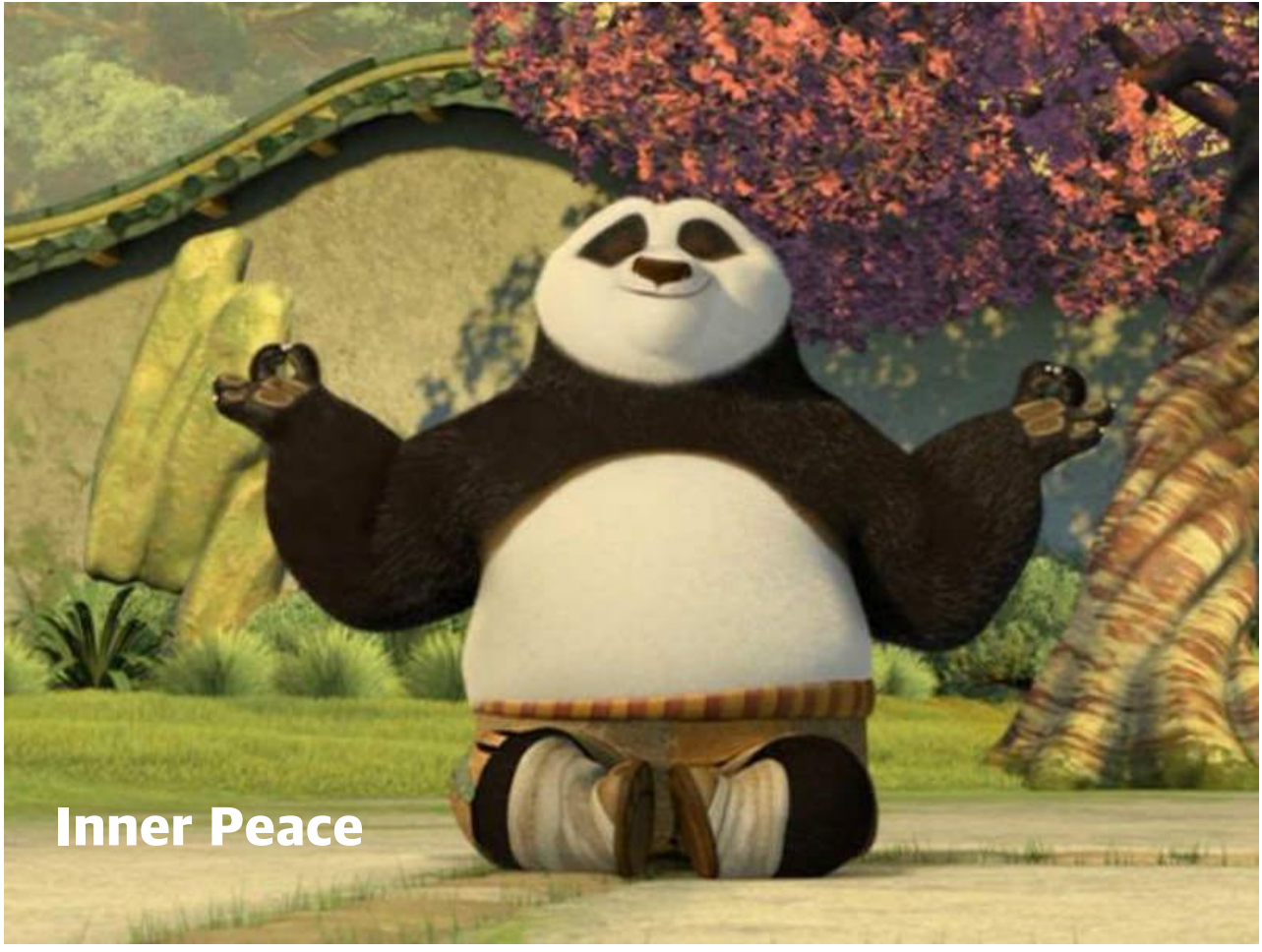
클래스 vs 객체



클래스 vs 객체

클래스 vs 객체

- 여러분은 어떤 차이를 느끼시나요?



객체지향형 프로그래밍 방법론

객체 지향 프로그래밍은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것이다. 각각의 객체는 메시지를 주고받고, 데이터를 처리할 수 있다.

객체지향형 프로그래밍 특징

- 추상화
- 캡슐화
- 은닉화
- 상속성
- 다형성

Swift Class Architecture

```
class ClassName : superClass
{
  var vName1 = “1”
  var vName2 = 4

  func fName1() – > Any
  {


  }

  func fName2(_ ani:Bool)
  {


  }

}
```

<CalssName.swift>