ios 입문 강의

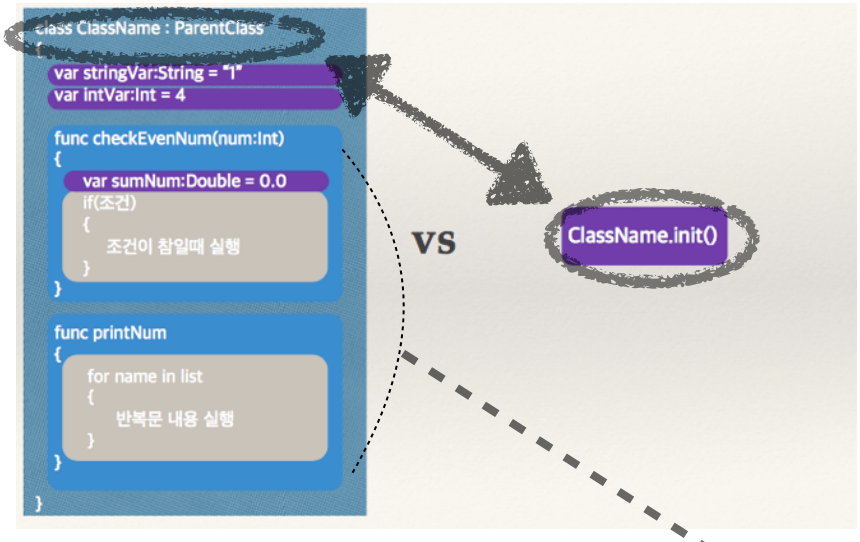
8강 <이전 학습 복습>

0. 복습

(1) 객체 지향 프로그래밍

* 클래스
  * 특정 기능, 값 등을 하나의 구조로 만들어놓은 객체의 틀
    * 변수와 함수들의 집합
* 객체
  * 클래스를 메모리에 적재시켜 실질적으로 사용하는 상태
  * 도면의 기능을 하는 파일 자체가 클래스 / 그것을 메모리 위에 적재 시킨 것이 객체

* 클래스 vs 객체



메모리 => Ram

		stringVar:String = "1"	intVar:Int = 4	
		checkEvenNum		
		printNum		

메모리 한 칸당 들어가는 데이터는 1과 0

클래스는 컴퓨터 입장에서는 하나의 텍스트일 뿐이다.
이 클래스가 객체가 되는 순간,
클래스 안에 있는 각각의 변수, 함수 등이 메모리에
적재될 것이다.

이렇게 객체화된 클래스는

메모리에 적재시켜 실질적으로 사용하는 상태는 이렇다.

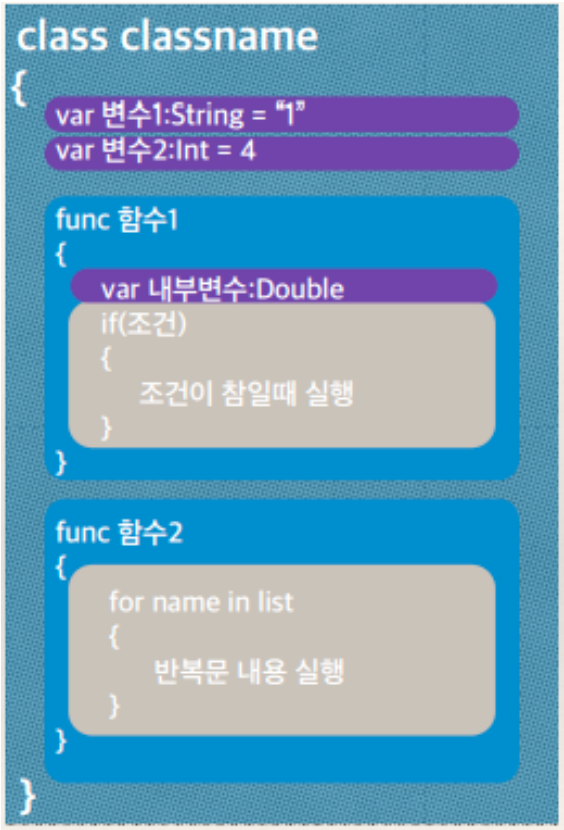
var person:Person = Person()

person은 Person의 주소값을 가지고 있는 구조

(2) 변수와 함수

위와 같은 클래스 안에 들어있는 것들이 변수와 함수이다.

* Swift Calss Architecture (그림참고) 👉



* Class 형태로 크게 이루어짐
  * 그 안에는 변수와 함수로 이루어짐
  * 그 함수 안에는 조건문, 반복문, 변수 등 여러가지 요소들이 존재
* Class들의 모음으로 프로그램이 구성

* 프로그래밍에서 가장 중요한 것

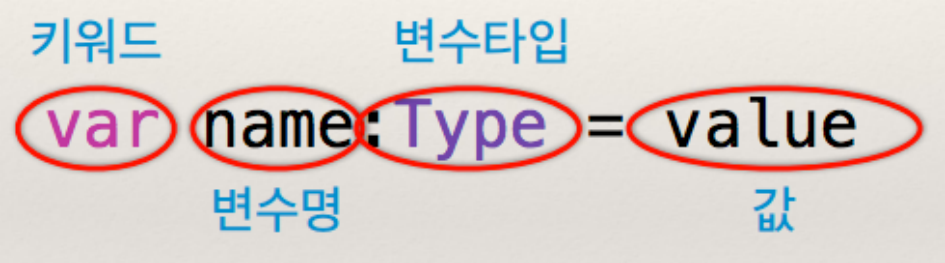
*** 변수(Variable)**

  * 데이터가공
  * 저장소 개념
  * 메모리에 저장됨

*** 함수(Method)**

  * 행동지시
  * 클래스가 할 행동(행위)의 정의
  * CPU가 연산 후, 변수에 집어넣는다 (메모리에 집어 넣는다)
    * 일정한 행동을 끝내면 변수에 집어넣고
    * 행동이 끝일 때에는 CPU에서 액션이 끝난다.

* Swift에서 변수를 만드는 문법



* 변수 타입

기본형

타입이름	타입	설명	Swift 문법 예제
정수	Int	1, 2, 3, 10, 100	var intName:Int
실수	Double	1.1, 2.35, 3.2	var doubleName:Double
문자열	String	"this is string"	var stringName:String
불리언	Bool	true or false	var boolName:Bool

참조형

타입이름	타입	설명	Swift 문법 예제
클래스명	ClassName	클래스 객체를 다른곳에서 사용할 경우	var customView:UIView
			var timer:NSTimer

※ Array는 클래스로 참조형타입이다.

- 형태 : [Type]

* 선언

- var vName:[Type] = [value1, value2…]
- var vName:Array <Type> = [value1, value2…]

* 변수를 만드는 키워드

```
var num:Int = 3
num = 4 ——O
```

*변수: 변할 수 있는 값 (var)

*var name:Type

*name을 나중에 바꿀 수 있음

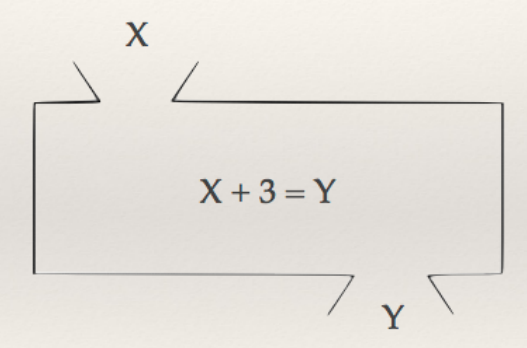
```
let num:Int = 3
num = 4 ——X
```

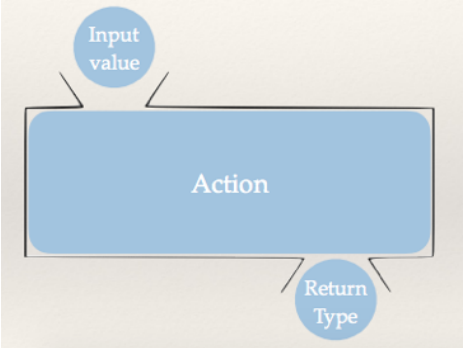
*상수: 변할 수 없는 고정값(let)

*let name:Type

*name변경 불가. 삭제하거나 새로 만들어야 한다.

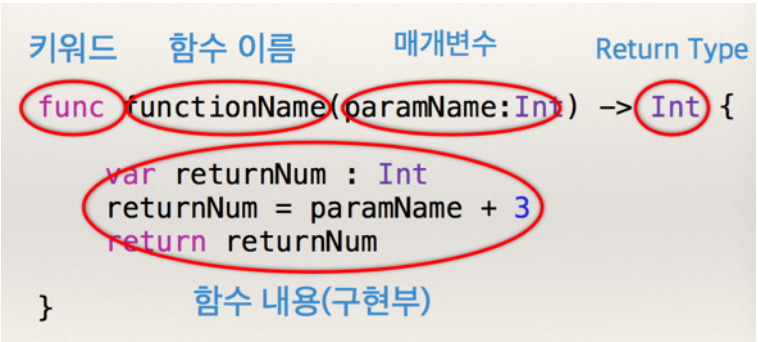
* 함수에 대하여





* 함수란, 입력을 통해 어떠한 일을 수행한 다음 결과물을 내어놓는 것

* Swift 문법 - 함수



* 중괄호 {} 를 통해 구현부 영역을 정해준다.

- 함수의 특정한 성질을 나타내는 변수를 말한다.
- 매개변수는 () 안에 변수로 작성된다.

• 매개변수는 해당 함수 안에서 만 사용된다

(3) 함수를 꾸미는 여러 구문

(3-1) 조건문 (if- else문)

* 일정한 조건이 충족될 때 선택적으로 실행

* if - else문 - 조건 비교

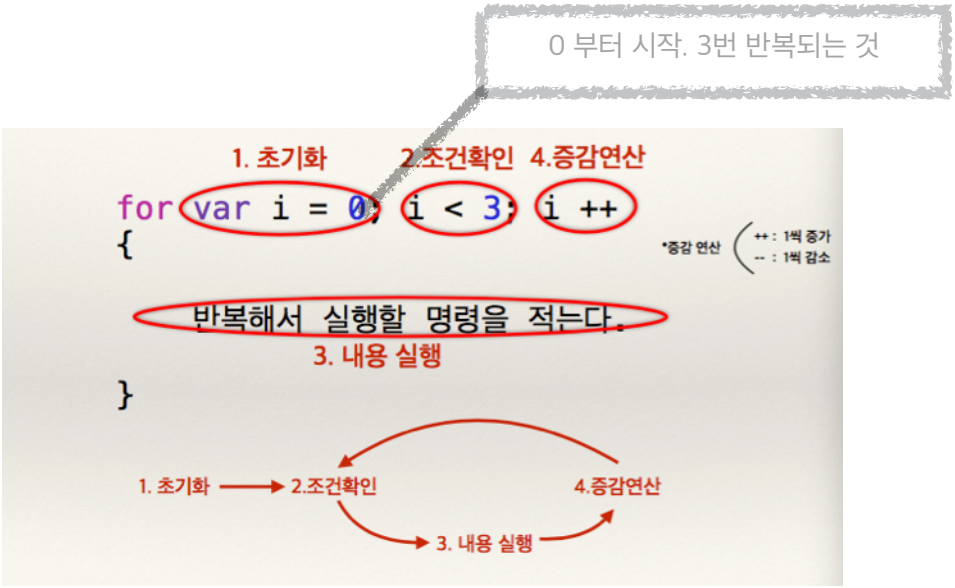
```
if(조건) {
   //조건이 만족되면 실행
}else{
   //조건이 만족되지 않을때 실행
}
```

*조건값은 꼭 참거짓으로 나타나는 불리언으로 나타나야된다.

(3-2) 반복문

- 반복적으로 실행되는 코드를 만드는 구문

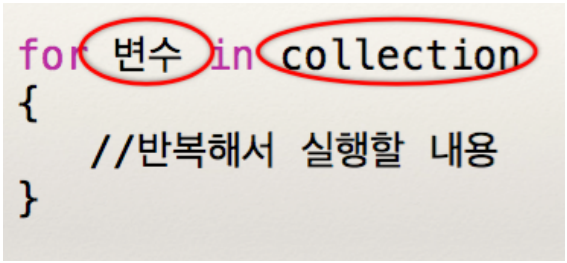
***문법 - for문**



But, 기본 for문은 스위프트 3.0에서 사라지기 때문에

우리는 아래의 for in문을 사용합니다.

(3-3) for-in문



1. collection에 있는 값을 하나씩 꺼내서 변수에 넣는다.
2. collection에 있는 값만큼 반복한다.

<실습>

성적 계산기 만들기

*Person Class - 이름, 과목들(array)

*Subject Class - 과목명, 과목 점수 속성

*PrintClass - 정보 출력 클래스 -> 총점, 평균

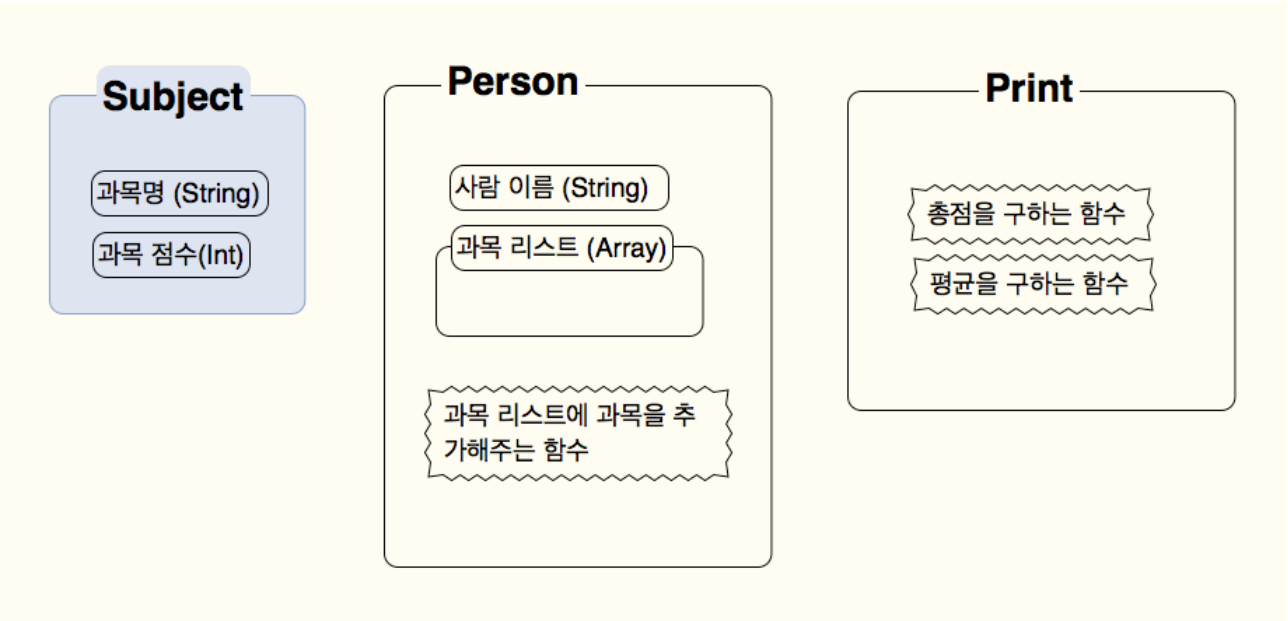
* ViewController - 메인 컨트롤

(다 풀어보시고 예제 답안 확인해주세요!)

ios 입문 5기 화이팅! 복습 꼭꼭 해주세용 :)

<예제 풀이>

저희가 만들 클래스와 구성은 각각 다음과 같습니다.



먼저, 과목 안에는 과목 명과 과목 점수를 세팅해 줄 것입니다.

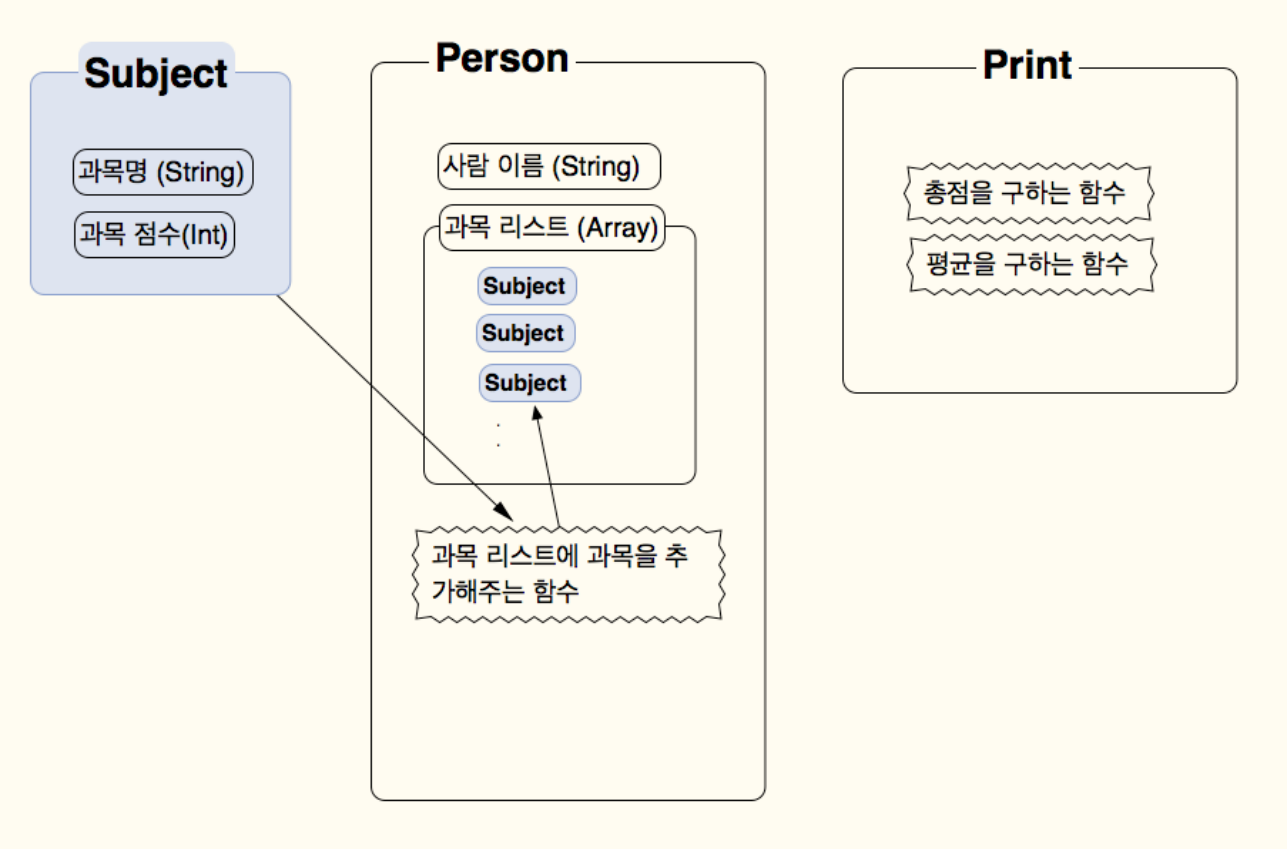
<Subject 클래스 코드>

```
import UIKit

class Subject: NSObject {

    var name:String
    var score:Int

    init(name:String, score:Int) {
        self.name = name
        self.score = score
    }

}
```

과목 명과 과목에 대한 점수를 받아 객체로 시켜줄 수 있는 init함수도 만들어 주었습니다.

다음으로는 이름과 과목 리스트를 가지는 사람 클래스를 만들 것입니다.

person이 가지고 있는 과목 리스트의 요소는 과목 명과 과목 점수가 세팅된 subject 객체들 일 것입니다.



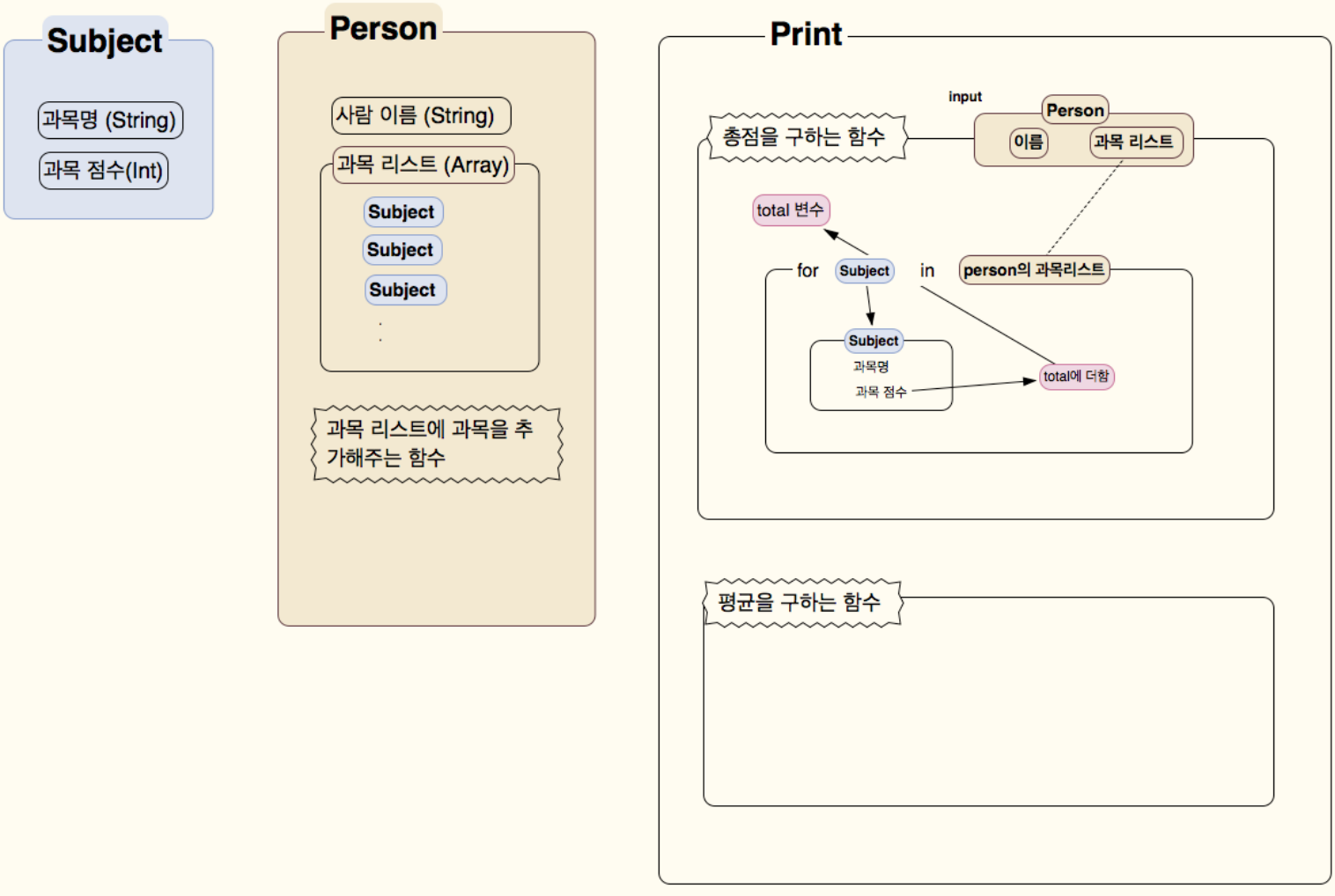
<person 클래스 코드>

```
class Person: NSObject {

    //사람이름
    var name:String?
    //과목들
    var subjectList:[Subject] = []

   init(name:String) {
        self.name = name
    }

    func addSubjectWithName(sName:String, sScore:Int) {
        //과목 객체 생성
        let subject:Subject = Subject(name: sName, score: sScore)

        //subjectList에 append
        subjectList.append(subject)
    }

}
```

이를 통해 우리는 viewDidLoad에서 Person객체를 만들고, 해당 객체의 메소드 addSubjectWithName을 통해 Subject객체를 생성하여 subjetList에 추가 (append)할 수 있습니다.

다음으로는 Print클래스입니다.
print 클래스에는 subject리스트를 가지고 있는 person을 넣었을 때, 총점과 평균이 구해지는 메소드를 구현할 것입니다.

먼저 총점을 구하는 함수의 구조입니다.



이름과 과목 리스트가 세팅된 Person타입의 객체를 input으로 넣었을 때, 우리는 해당 객체를 통해 과목 리스트에 접근할 수 있습니다. 이 리스트를 for in문으로 돌리면, 각각의 subject 객체들을 추출할 수 있습니다.

각각의 subject객체들은 과목명과 과목 점수를 가지고 있었습니다. 우리는 이 과목 점수를 함수 내에 만들어둔 total 변수에 계속 합할 것입니다. 이를 통해 total점수를 구할 수 있습니다.

<Print 클래스의 코드> - 총합 구하는 함수 부분

```
class Print: NSObject {
    func printTotalScore(person:Person) -> Int {

        var total:Int = 0

        for subject in person.subjectList
        {
            total += subject.score
        }

        print("\(person.name)의 총점은 \(total)이다.")

        return (total)
    }
    func printAverage(person:Person) {
        let total:Int = printTotalScore(person)
        let count:Int = person.subjectList.count
        print ("\(person.name)의 평균은 \(Double(total) / Double(count))이다.")
    }

}
```

총합 구현 부분

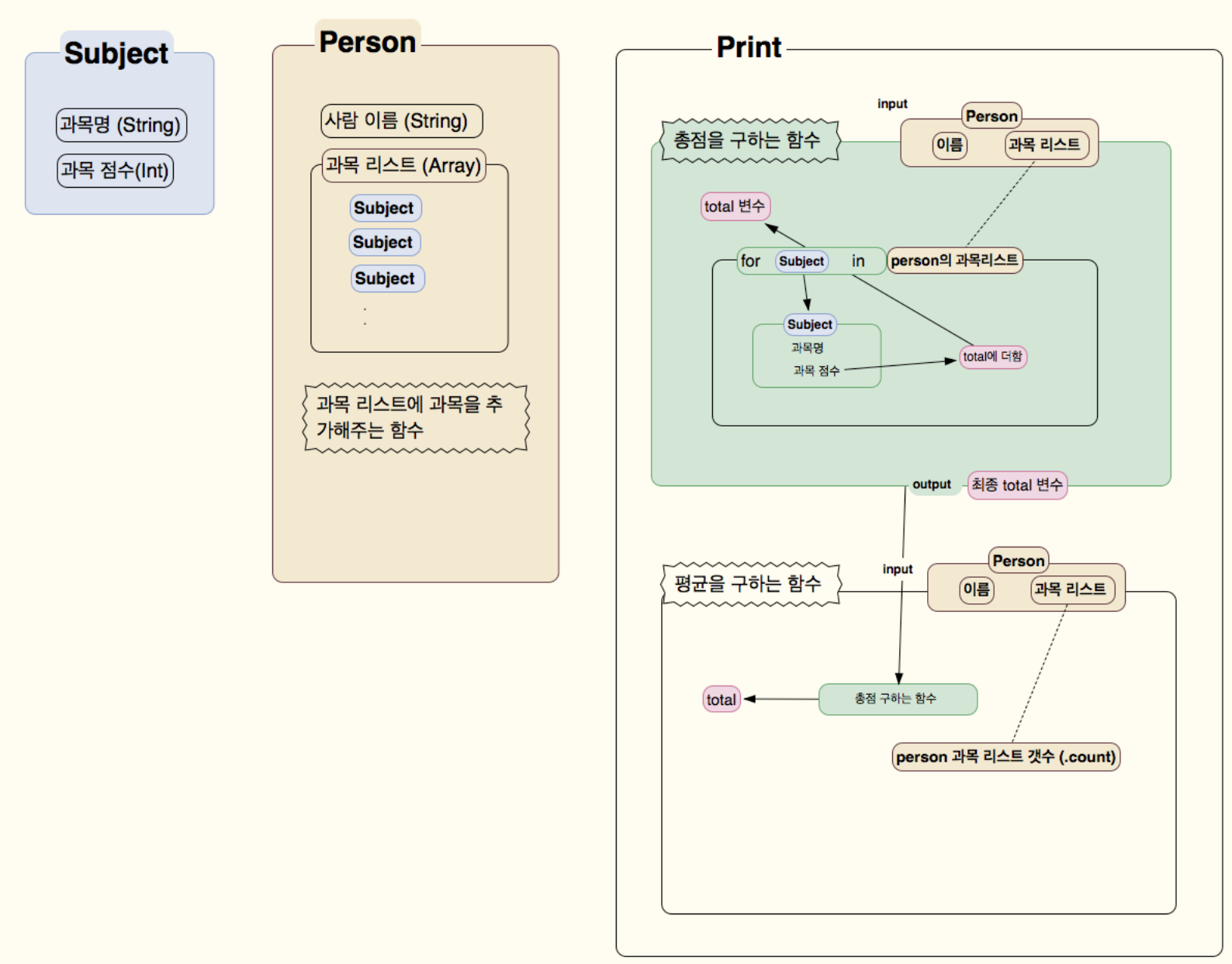
다음으로는 평균을 구하는 함수입니다.
평균을 구하는 함수 내에는 반드시 total에 대한 값이 있어야 할 것입니다. (평균이란, 총 점수 / 총 과목의 갯수 이기 때문에)

보통은 total 변수를 전역 변수로 둔 다음에, 총점을 구하는 함수에서 더하고, 평균을 구하는 함수에서 이를 활용하여 계산할 수 있습니다.

그러나 위와 같은 방식은 total 구하는 함수를 먼저 구현해준 다음에야 평균을 구하는 함수를 사용할 수 있는 구조입니다.

ios 입문 5기 화이팅! 복습 꼭꼭 해주세용 :)

때문에 순서에 상관없이 평균을 구하는 함수를 먼저 불렀을 때에도 알아서 제대로된 값을 구해주려면 다음과 같은 구조를 취합니다.



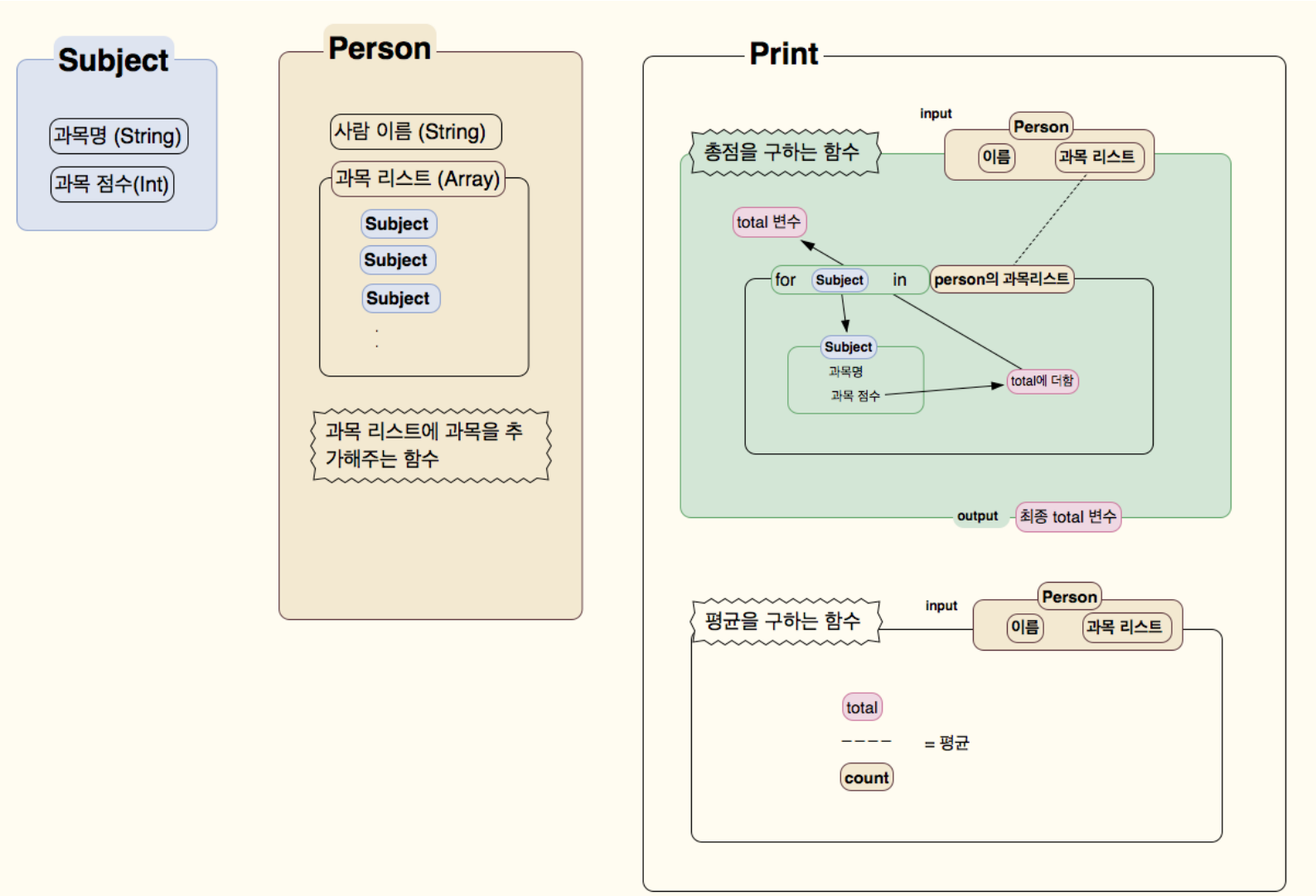
평균을 구하는 함수에는 총점을 구하는 함수와 마찬가지로 Person타입의 객체가 input됩니다.

들어온 Person타입의 객체에는 이름과 과목 리스트가 세팅되어 있을 것입니다.

우리는 total 변수를 하나 만들고, 이전에 구현해놓은 총점을 구하는 함수를 평균을 구하는 함수를 불러 output된 total을 받을 수 있도록 합니다.(평균을 구하는 함수 내에서 총점을 구하는 함수를 불러 결과값을 total 변수로 취한 것입니다)

ios 입문 5기 화이팅! 복습 꼭꼭 해주세용 :)

위와 같은 과정을 거쳐 얻은 total을 person이 가진 과목 리스트의 갯수로 나눔으로서 한 person객체가 가지고 있는 과목의 평균 점수를 구할 수 있습니다.



총 과목의 갯수(count)는 Person객체가 가지고 있던 과목 리스트를 .count하여 가져올 수 있습니다.

ios 입문 5기 화이팅! 복습 꼭꼭 해주세용 :)

<Print 클래스 >

```
class Print: NSObject {

    func printTotalScore(person:Person) -> Int {

        var total:Int = 0

        for subject in person.subjectList
        {
            total += subject.score
        }

        print("\(person.name)의 총점은 \(total)이다.")

        return (total)
    }

    func printAverage(person:Person) {
        let total:Int = printTotalScore(person)
        let count:Int = person.subjectList.count
        print ("\(person.name)의 평균은 \(Double(total) / Double(count))이다.")
    }
}
```

평균 구하는 함수 부분

최종적으로

<ViewDidLoad에서 사용할 모습>

```
override func viewDidLoad() {
    super.viewDidLoad()

    var person1:Person = Person(name: "김사람")

    person1.addSubjectWithName("수학", sScore: 90)
    person1.addSubjectWithName("영어", sScore: 80)
    person1.addSubjectWithName("국어", sScore: 82)

    var printMachine:Print = Print()
    printMachine.printTotalScore(person1)
    printMachine.printAverage(person1)


}
```

Person 객체를 만든 후. 초기값으로 이름을 설정해주고

addSubjectWithName 메소드를 통해 Subject객체를

Person 함수 내에 있는 subjectList에 추가시킬 수 있습니다.

또한 Print 클래스를 통해 또한 세팅된 사람 객체를 input하여

총점과 평균을 추출 해 올 수 있습니다.