# Scala

출처

O'Reilly Media, Inc, USA. (2014). Programming Scala.

```
val book = "Programming Scala"
println(book)
```

- val 키워드: 불변(immutable) 변수 선언
- 타입 표기(type annotation): 타입 정보를 보여주거나, 선언에 명시적으로 타입 정보를 추가하는 경우, 대상 이름 뒤에 콜론(:)을 붙인 다음 추가
- 주석 - 한 줄 주석 처리의 경우 //, 여러줄 주석 처리의 경우 /* */
- 스크립트 파일의 확장자는 .scala

```
class Upper{
  def upper(strings: String*): Seq[string] = {
    strings.map((s:String) => s.toUpperCase())
  }
}

val up = new Upper  // upper의 인스턴스 생성
println(up.upper("Hello","World!")) //"Hello", "World!"의 2개의 문자열을 입력으로 받고 결과로 얻은 Seq를 마지막에 출력
```

Upper 클래스 내의 upper 메서드: 하나 이상의 입력 문자열을 대문자로 바꿔서 Seq(시퀀스)로 반환

class 키워드: 클래스 생성

- 전체 클래스 본문은 중괄호 사이에 위치 - 클래스의 주 생성자(primary constructor)
  - 생성자가 매개변수를 받아야 하면 클래스 이름 Upper 뒤에 매개변수 목록 추가
  - 매개변수(parameter) - 함수를 사용할 때 사용한 형식인자(formal parameter) 혹은 함수를 호출하면서 전달한 실제값을 의미하는 인자(argument)로 혼용하여 사용

def 키워드: 메서드 정의

- def → 메서드_이름 → 매개변수_목록 → (선택적으로) 반환 타입
- def upper(strings: String*): Seq[string] 에서 strings는 매개변수, 괄호로 둘러싼 (strings: String*)는 매개변수 목록

  <예시>

  add(a:Int, b:Int) 는 a,b 2개의 인자를 가지고 있고 인자 목록은 1개

addPrint(a:Int, b:Int)(fmt:String) 는 2개의 인자 목록을 가지고 있고 첫번째 인자목록은 2개의 인자를, 두번째 인자목록은 1개의 인자가 들어있음

  - 매개변수 목록은 실제로는 String으로 이루어진 가변 길이 매개변수 목록(variable-length argument list) → String 뒤에 *를 붙임(원하는 만큼 문자열(빈문자열 포함)을 콤마로 분리해서 인자로 넘길 수 있음)
  - 메서드 안에서 strings 매개변수의 타입은 실제로는 자바 배열을 감싼 WrappedArray
- 반환 타입은 콜론 뒤에 타입을 붙여서 표기(많은 경우 컴파일러가 추론할 수 있음)
  - Seq은 일정한 순서로 반복(iteration) 가능한 추상적 컬렉션 → 자바의 제네릭 타입처럼 매개변수화한 타입(parameterized type)으로 어떤 것의 시퀀스라는 의미
  - 매개변수화한 타입 정할 때 각괄호([]) 사용
- 마지막으로 등호(=)가 메서드 시그니처(signature)와 본문 사이에 옴
  - 왜 중괄호를 바로 사용하지 않는가?
    - 메서드가 매개변수를 받지 않는 경우 매개변수 목록 생략 가능
    - 값이나 함수가 더 깊이 연관된 개념이라는 함수형 프로그래밍(functional programming)의 원칙 강조 → 함수를 다른 함수에 인자로 넘길 수 있고, 함수가 함수를 반환할 수 있고, 변수에 대입할 수 있음
    - 메서드 본문이 단일식으로 이루어진 경우 중괄호 생략 허용
- 본문
  - strings 컬렉션에 대해 map 메서드 호출
    - map은 함수 리터럴(function literal, 익명 함수)을 인자로 받아서 그 함수 리터럴에 각각의 String을 넘겨서, 그 함수가 반환한 결과를 모은 새 컬렉션을 만듦
      - 리터럴이란? 리터럴은 코드에서 상수 값을 설명하는 데 사용되는 일련의 기호

      (s: String) ⇒ s.toUpperCase()
      - String 타입의 변수 s 1개
      - 함수 리터럴의 본문은 '화살표' ⇒ 다음에 나옴
      - s에 대해 toUpperCase()를 호출 → 호출한 결과가 자동으로 함수 리터럴의 결과로 반환
      - 스칼라에서는 함수나 메서드의 가장 마지막 식(expression)이 반환값이 됨
        - return의 경우 메서드에서만 사용할 수 있음
        - 메서드(method): 클래스나 객체 안에서 정의된 함수
        - 함수(function): 메서드가 아닌 경우를 가리키기 위해 사용(조금 더 큰 범위)

- Upper 인스턴스 만들기 → new Upper
  - 주 생성자가 인자를 받지 않으므로 인자 목록을 지정할 필요가 없음

```
#upper1.sc를 간략화한 똑같은 코드
```

```
Object Upper{
  def upper(strings: String*) = strings.map(_.toUpperCase())
}

println(Upper.upeer("Hello", "World!"))
```

object 선언: 싱글턴(singleton) 객체

- 스칼라 실행 환경은 Upper 인스턴스를 오직 하나만 생성(new Upper 사용할 수 없음)
- Upper에 별도의 상태가 들어 있지 않기 때문에, 실제로 인스턴스가 둘 이상 필요 없음

스칼라는 반환 타입 추론 가능(매개변수 타입은 추론할 수 없음)

메서드 본문에 식이 하나만 있으므로 중괄호도 없앰