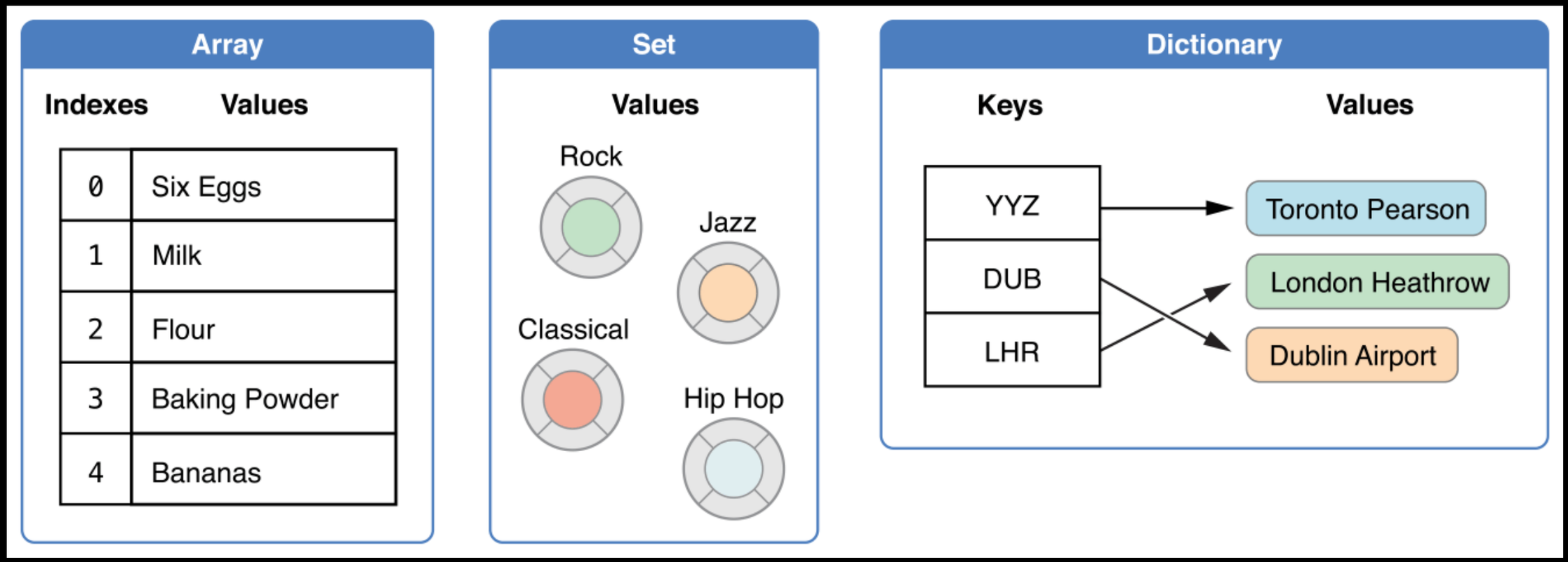
SWIFT
Collection Types

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Collection Types

Swift에서는 콜렉션(리스트) 타입으로 아래의 세가지 형태의 타입을 지원합니다.



Array

순서(인덱스)가 있는 리스트 형태의 컬렉션 타입

```swift
// 빈 Int Array 생성
var integers: Array<Int> = Array<Int>()

// 다른 생성 방법
// var integers: Array<Int> = [Int]()
// var integers: Array<Int> = []
// var integers: [Int] = Array<Int>()
// var integers: [Int] = [Int]()
// var integers: [Int] = []
// var integers = [Int]()

// 추가
integers.append(1)
integers.append(100)

print(integers) // [1, 100]

// 비었는지 확인
if integers.isEmpty {
    print("It's empty.")
} else {
    print("It's not empty.")
}
```

Array

```swift
// 수정
integers[0] = 99
integers[1] = 10

// 삭제
integers.remove(at: 0)
integers.removeLast()
integers.removeAll()
```

```swift
// 멤버 갯수 확인
print(integers.count)

// 배열 순회
for item in integers {
    print(item)
}
```

Dictionaries

키와 값의 쌍으로 이루어진 컬렉션 타입입니다.

```swift
// Key가 String 타입이고 Value가 Any인 빈 Dictionary 생성
var anyDictionary: Dictionary<String, Any> = [String: Any]()

// 다른 생성 방법
// var anyDictionary: Dictionary <String, Any> = Dictionary<String, Any>()
// var anyDictionary: Dictionary <String, Any> = [:]
// var anyDictionary: [String: Any] = Dictionary<String, Any>()
// var anyDictionary: [String: Any] = [String: Any]()
// var anyDictionary: [String: Any] = [:]
// var anyDictionary = [String: Any]()

// 추가
anyDictionary["someKey"] = "value"
anyDictionary["anotherKey"] = 100

print(anyDictionary) // ["someKey": "value", "anotherKey": 100]
```

Dictionaries

```swift
// 비었는지 확인
if anyDictionary.isEmpty {
    print("It's empty.")
} else {
    print("It's not empty.")
}

// 수정
anyDictionary["someKey"] = "dictionary"
print(anyDictionary) // ["someKey": "dictionary", "anotherKey": 100]

// 삭제
anyDictionary.removeValue(forKey: "anotherKey")
anyDictionary["someKey"] = nil
print(anyDictionary) // [:]

// 멤버 갯수 확인
print(anyDictionary.count) // 0
```

Set

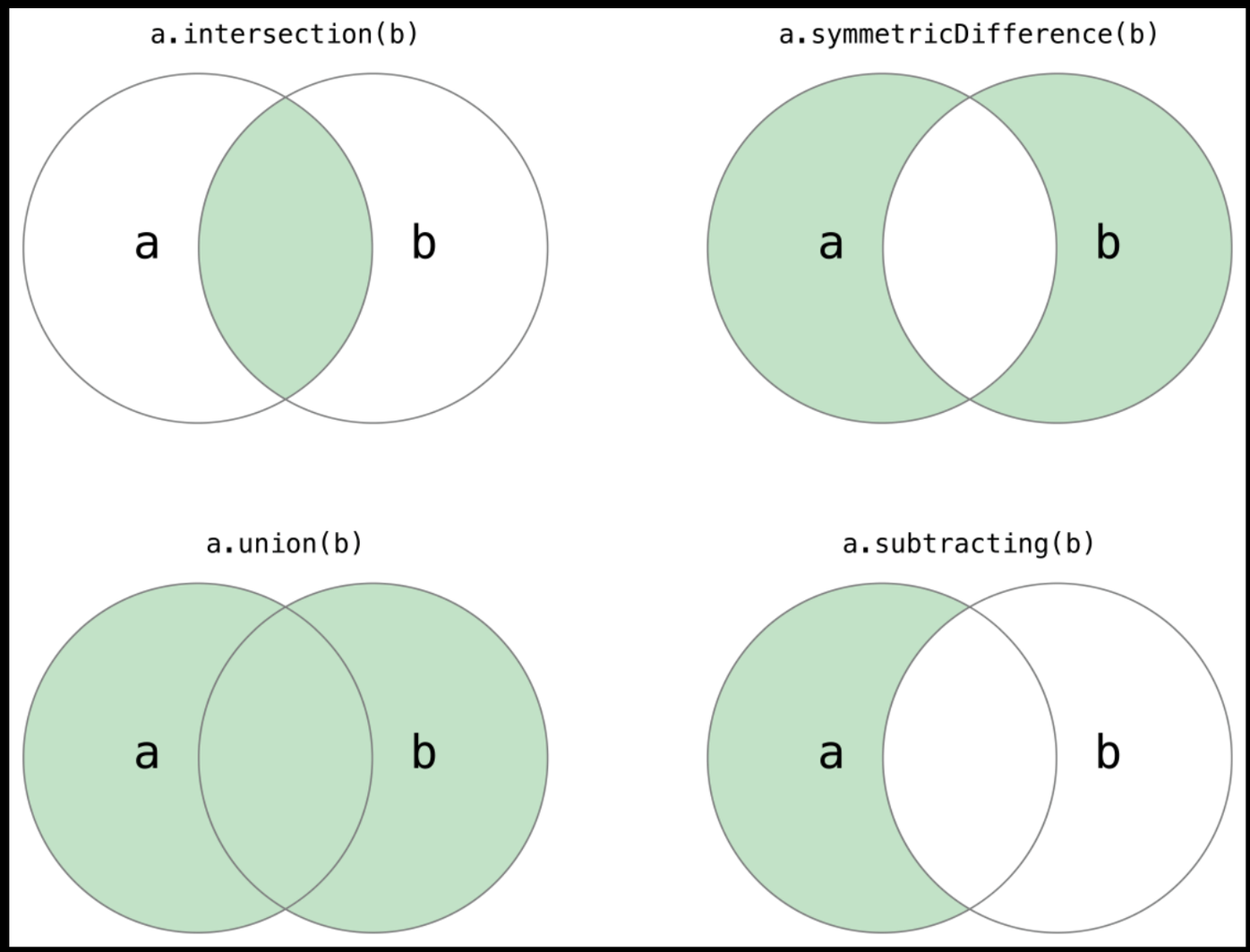
순서가 없고 멤버가 유일한 것을 보장하는 컬렉션
값에 대해서 중복 허용 안됨

```swift
// 빈 Int Set 생성
var integerSet: Set<Int> = Set<Int>()

// 추가
integerSet.insert(1)
integerSet.insert(100)
integerSet.insert(99)
integerSet.insert(99) // 99가 이미 있기 때문에 추가 안됨
integerSet.insert(99) // 99가 이미 있기 때문에 추가 안됨

print(integerSet) // [100, 99, 1]
print(integerSet.contains(1)) // true
print(integerSet.contains(2)) // false

// 비었는지 확인
if anyDictionary.isEmpty {
    print("It's empty.")
} else {
    print("It's not empty.")
}
```

Set

```swift
// 삭제
integerSet.remove(100)
integerSet.removeFirst()

// 멤버 갯수 확인
print(integerSet.count) // 1

// Set 멤버 순회
for item in integerSet {
    print("\(item)")
}
```

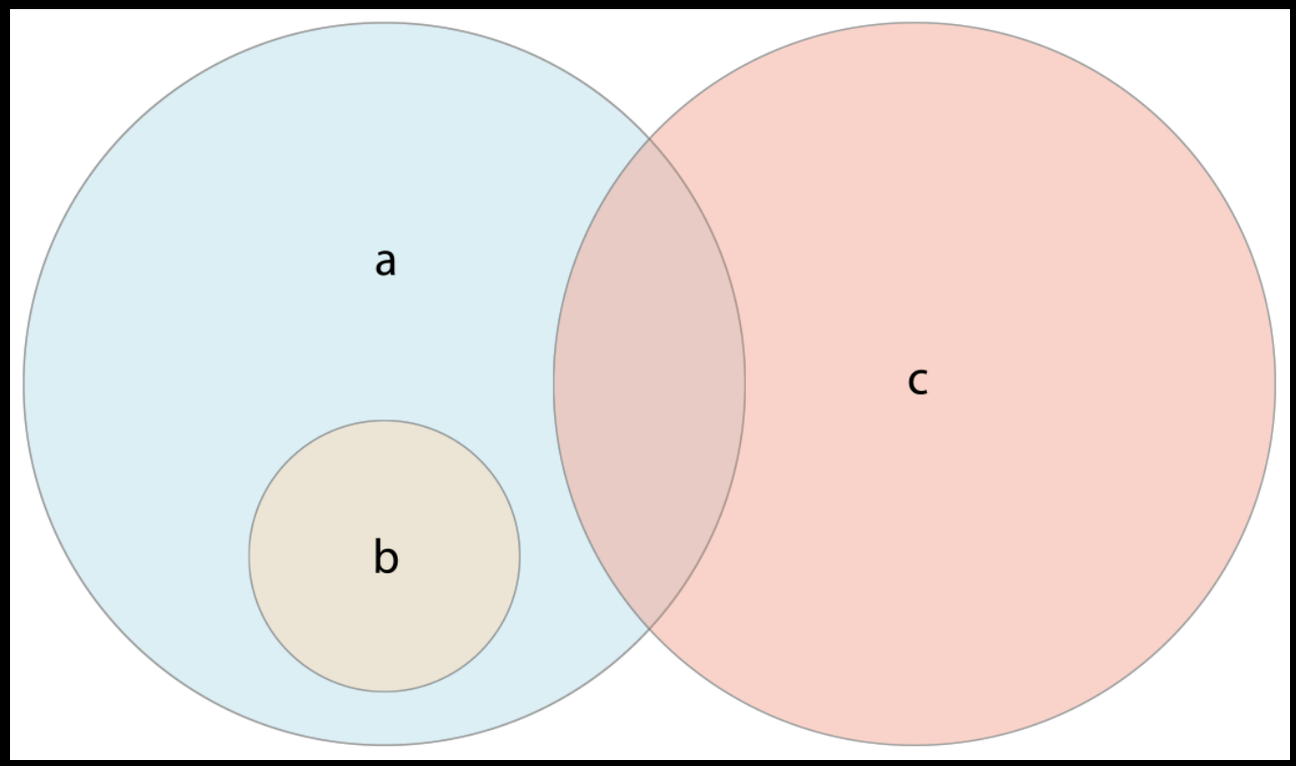
Set


a.intersection(b)
a
b
a.symmetricDifference(b)
a
b
a.union(b)
a
b
a.subtracting(b)
a
b

Set

```swift
// Set는 집합 연산에 꽤 유용합니다
let setA: Set<Int> = [1, 2, 3, 4, 5]
let setB: Set<Int> = [3, 4, 5, 6, 7]

// 합집합
let union: Set<Int> = setA.union(setB)
print(union) // [4, 6, 5, 1, 2, 7, 3]

// 합집합 오름차순 정렬
let sortedUnion: [Int] = union.sorted()
print(sortedUnion) // [1, 2, 3, 4, 5, 6, 7]

// 교집합
let intersection: Set<Int> = setA.intersection(setB)
print(intersection) // [3, 5, 4]

// 차집합
let subtracting: Set<Int> = setA.subtracting(setB)
print(subtracting) // [2, 1]

// 대칭차
let symDiffSet: Set<Int> = setA.symmetricDifference(setB)
print(symDiffSet) // [6, 2, 7, 1]
```

Set



a는 b의 Superset
b는 a의 Subset
b와 c는 서로 Disjoint 관계

Set

```swift
let houseAnimals: Set = ["🐶", "🐱"]
let farmAnimals: Set = ["🐮", "🐔", "🐑", "🐶", "🐱"]
let cityAnimals: Set = ["🐳", "🐭"]

// isSubset(of:) : A.isSubset(of:B) : A가 B의 부분집합이면 true 아니면 false 반환
// isSuperset(of:) : B.isSuperset(of:A) : B가 A의 상위 집합이면 true 아니면 false 반환
// isStrictSubset(of:) : subset이면서 subset != superset이면 true 아니면 false 반환
// isStrictSuperset(of:) : superset이면서 subset != superset이면 true 아니면 false 반환
// isDisjoint(with:) : 두 집합 사이에 어떤 공통 값이 없을 때 true, 하나라도 있으면 false 반환

print(houseAnimals.isSubset(of: farmAnimals)) // 참(true)
print(farmAnimals.isSuperset(of: houseAnimals)) // 참(true)
print(houseAnimals.isStrictSubset(of: farmAnimals)) // 참(true)
print(farmAnimals.isStrictSuperset(of: houseAnimals)) // 참(true)
print(farmAnimals.isDisjoint(with: cityAnimals)) // 참(true)
```

References

[1] 콜렉션 타입 (Collection Types) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/04-collection-types

[2] 컬렉션 타입 : https://yagom.github.io/swift_basic/contents/03_collection_types/

[3] [Swift]Collection Types 정리 : http://minsone.github.io/mac/ios/swift-collection-types-summary

[4] [Swift 공부] 컬렉션 타입이란 ? : https://noahlogs.tistory.com/14

[5] Swift - 함수, 콜렉션 타입 : https://blog.yagom.net/528/

References

[6] Swift - 컬렉션 타입(Collection Types) : http://seorenn.blogspot.com/2014/06/swift-collection-types.html

[7] [Swift 3] 컬렉션 타입 (Collection Types_Array, Set, Dictionary) : https://beankhan.tistory.com/155

[8] 컬렉션 타입(Collection Types) : https://kka7.tistory.com/141

[9] 3 swift 컬렉션 : https://www.slideshare.net/donggyupark2/3-swift-56764853

[10] [Swift] 016 Working with Collection (컬렉션 사용하기) for beginners : https://creativeworks.tistory.com/entry/Swift-016-Working-with-Collection-컬렉션-사용하기-for-beginners

Thank you!