

이펙티브자바 Ch6.

열거 타입과 애너테이션

Item 37.

Enum의 효율전략: Ordinal 인덱싱 대신 EnumMap을 사용하라

2025.05.19

김진수

Item 37.

Enum의 효율전략:

Ordinal 인덱싱 대신 EnumMap을 사용하라

01 Ordinal()기반 인덱싱의 구조와 한계

02 EnumMap이 가져다주는 구조적 해결

01 Ordinal()기반 인덱싱의 구조와 한계

Ordinal의 존재

Ordinal() + 배열 인덱싱 예제

작동은 하지만.. 무슨 문제가 있나?

Ordinal()의 존재

```
public abstract class Enum ... {  
  
    private final String name;  
    private final int ordinal;  
  
    protected Enum(String name, int ordinal) {  
        this.name = name;  
        this.ordinal = ordinal;  
    }  
  
    public final int ordinal() {  
        return ordinal;  
    }  
  
}
```

Ordinal()의 존재

```
public abstract class Enum ... {  
  
    private final String name;  
    private final int ordinal;  
  
    protected Enum(String name, int ordinal) {  
        this.name = name;  
        this.ordinal = ordinal;  
    }  
  
    public final int ordinal() {  
        return ordinal;  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        4 usages  
        enum Season {SPRING, SUMMER, AUTUMN, WINTER}  
  
        /**  
         * 컴파일 타임 - Season 생성자 내부  
         * SPRING = new Season("SPRING", 0);  
         * SUMMER = new SUMMER("SUMMER" 1);  
         * AUTUMN = new SUMMER("AUTUMN" 2);  
         * WINTER = new SUMMER("WINTER" 3);  
         */  
        int ordinal0 = Season.SPRING.ordinal();  
        int ordinal1 = Season.SUMMER.ordinal();  
        int ordinal2 = Season.AUTUMN.ordinal();  
        int ordinal3 = Season.WINTER.ordinal();  
  
        System.out.println(ordinal0); // 0  
        System.out.println(ordinal1); // 1  
        System.out.println(ordinal2); // 2  
        System.out.println(ordinal3); // 3  
    }  
}
```

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기: 봄에는 무슨과일, 여름에는 무슨과일~

Enum Season

0

1

2

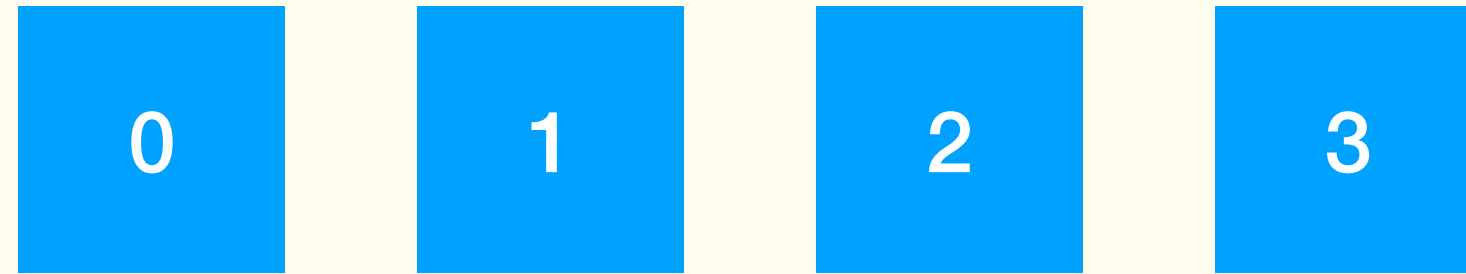
3

```
public class Main {  
    public static void main(String[] args) {  
        4 usages  
        enum Season {SPRING, SUMMER, AUTUMN, WINTER}  
  
        /**  
         * 컴파일 타임 - Season 생성자 내부  
         * SPRING = new Season("SPRING", 0);  
         * SUMMER = new SUMMER("SUMMER" 1);  
         * AUTUMN = new SUMMER("AUTUMN" 2);  
         * WINTER = new SUMMER("WINTER" 3);  
         */  
        int ordinal0 = Season.SPRING.ordinal();  
        int ordinal1 = Season.SUMMER.ordinal();  
        int ordinal2 = Season.AUTUMN.ordinal();  
        int ordinal3 = Season.WINTER.ordinal();  
  
        System.out.println(ordinal0); // 0  
        System.out.println(ordinal1); // 1  
        System.out.println(ordinal2); // 2  
        System.out.println(ordinal3); // 3  
    }  
}
```

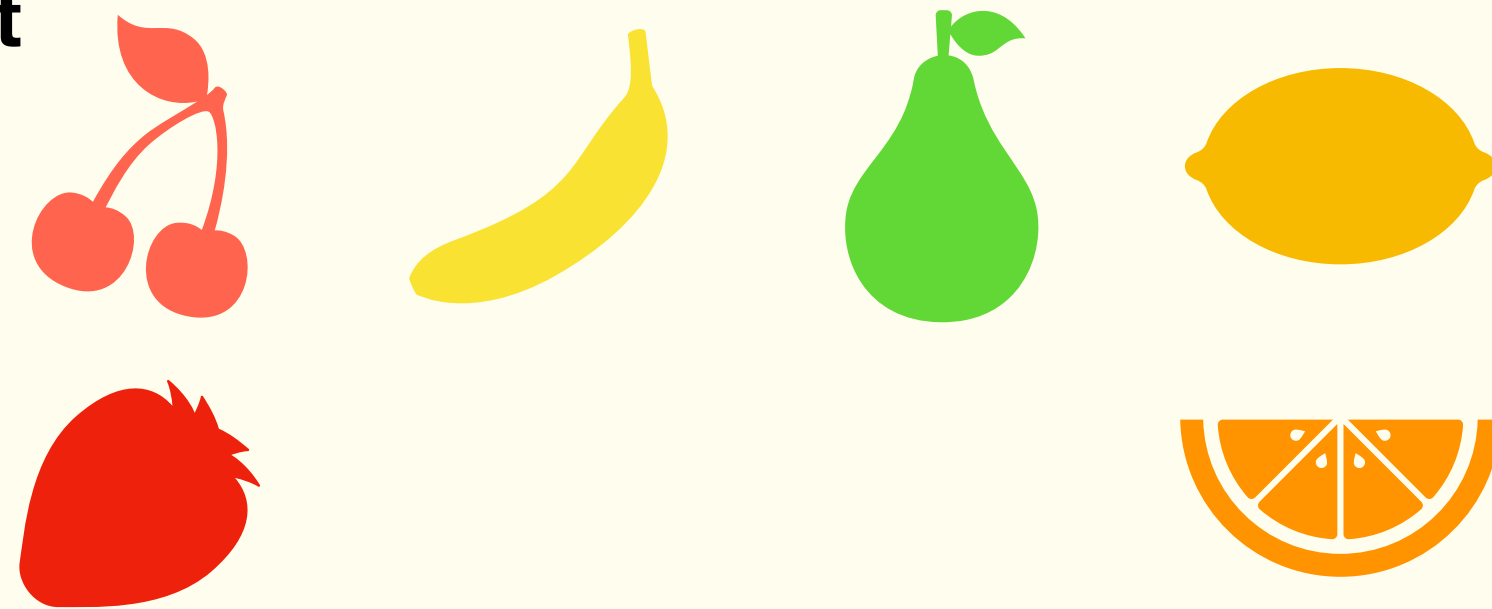

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Enum Season



Class Fruit

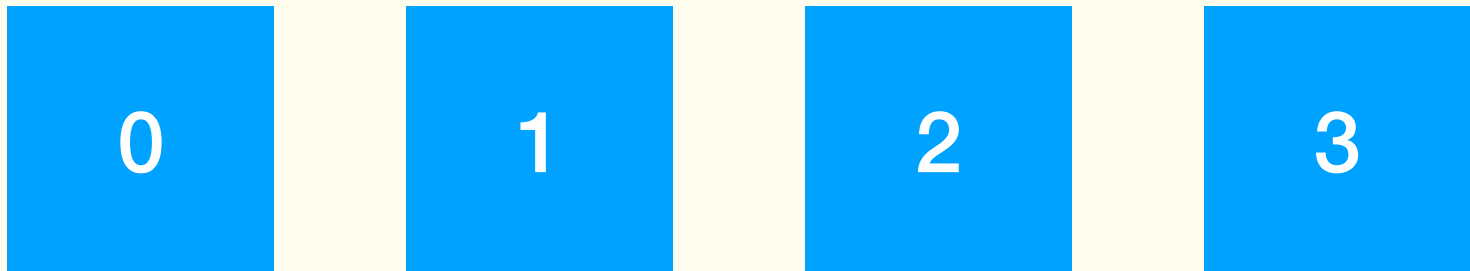


```
public class Main {  
    public static void main(String[] args) {  
        4 usages  
        enum Season {SPRING, SUMMER, AUTUMN, WINTER}  
  
        /**  
         * 컴파일 타임 - Season 생성자 내부  
         * SPRING = new Season("SPRING", 0);  
         * SUMMER = new SUMMER("SUMMER" 1);  
         * AUTUMN = new SUMMER("AUTUMN" 2);  
         * WINTER = new SUMMER("WINTER" 3);  
         */  
  
        int ordinal0 = Season.SPRING.ordinal();  
        int ordinal1 = Season.SUMMER.ordinal();  
        int ordinal2 = Season.AUTUMN.ordinal();  
        int ordinal3 = Season.WINTER.ordinal();  
  
        System.out.println(ordinal0); // 0  
        System.out.println(ordinal1); // 1  
        System.out.println(ordinal2); // 2  
        System.out.println(ordinal3); // 3  
    }  
}
```

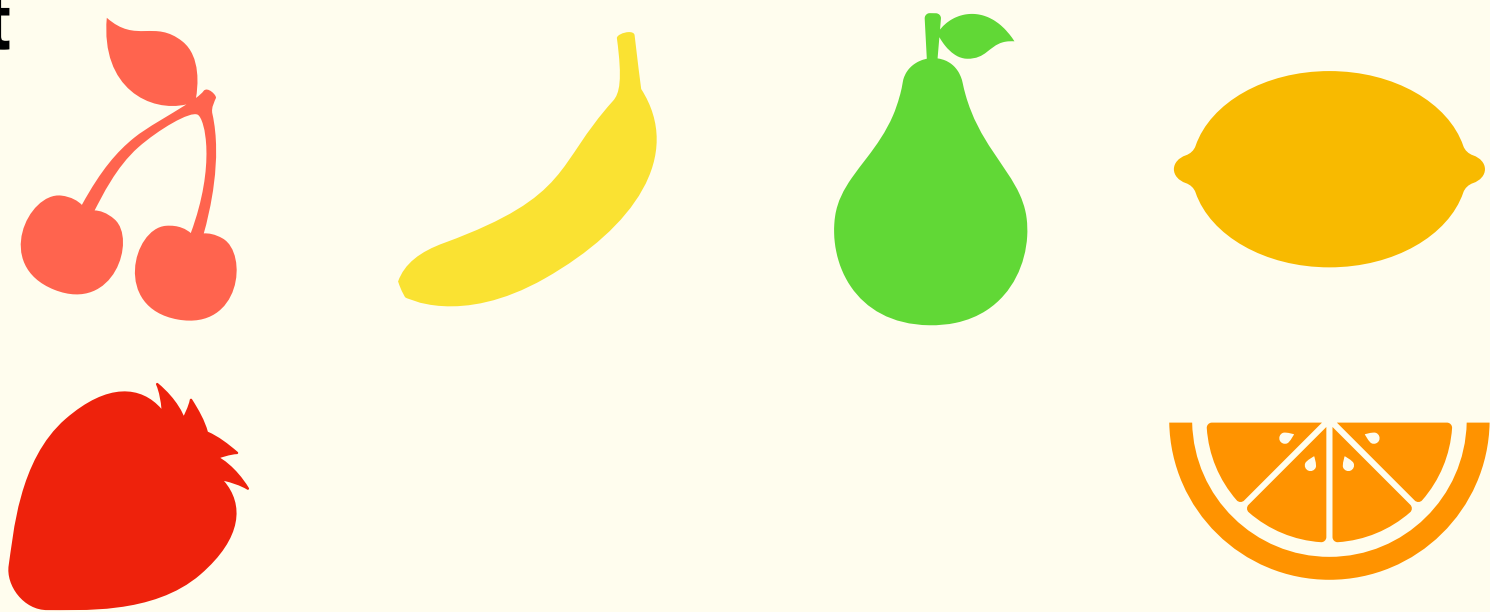
Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

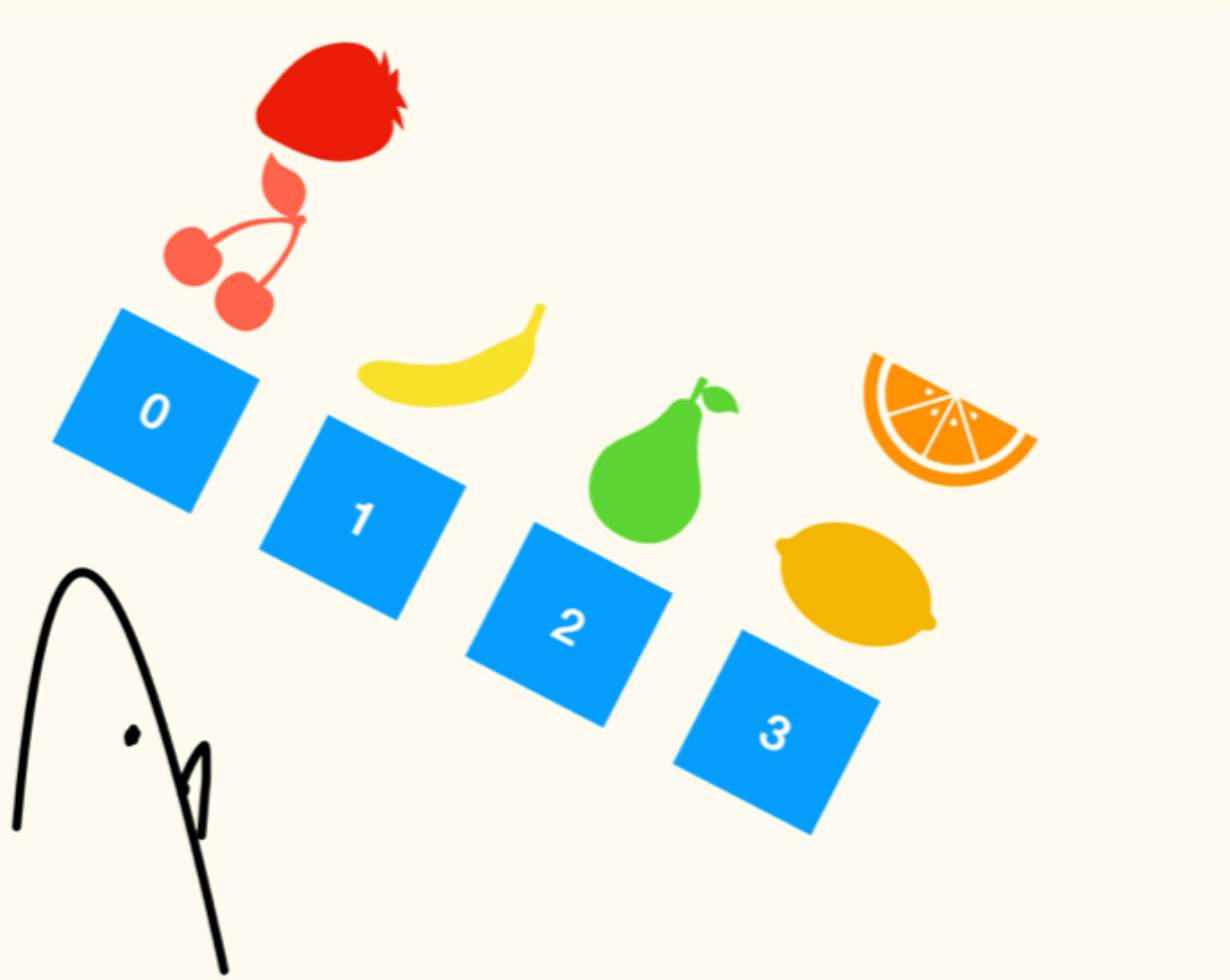
Enum Season



Class Fruit



UseCase



```
public class Main {
    public static void main(String[] args) {
        4 usages
        enum Season {SPRING, SUMMER, AUTUMN, WINTER}

        /**
         * 컴파일 타임 - Season 생성자 내부
         * SPRING = new Season("SPRING", 0);
         * SUMMER = new SUMMER("SUMMER" 1);
         * AUTUMN = new SUMMER("AUTUMN" 2);
         * WINTER = new SUMMER("WINTER" 3);
         */

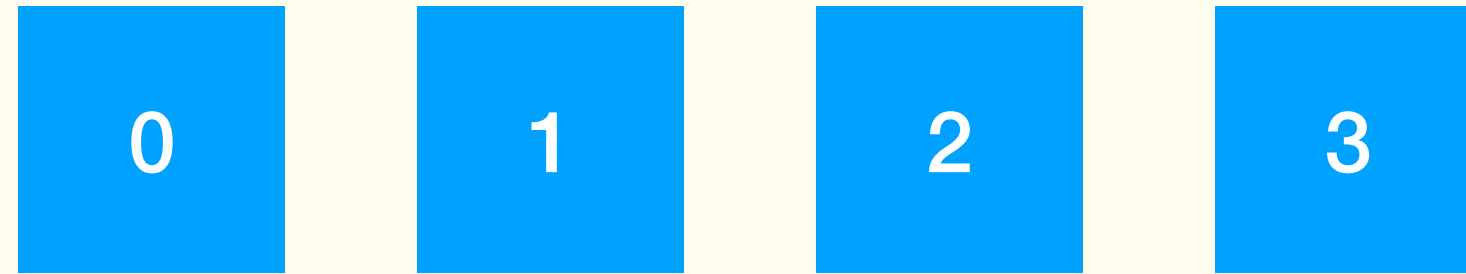
        int ordinal0 = Season.SPRING.ordinal();
        int ordinal1 = Season.SUMMER.ordinal();
        int ordinal2 = Season.AUTUMN.ordinal();
        int ordinal3 = Season.WINTER.ordinal();

        System.out.println(ordinal0); // 0
        System.out.println(ordinal1); // 1
        System.out.println(ordinal2); // 2
        System.out.println(ordinal3); // 3
    }
}
```

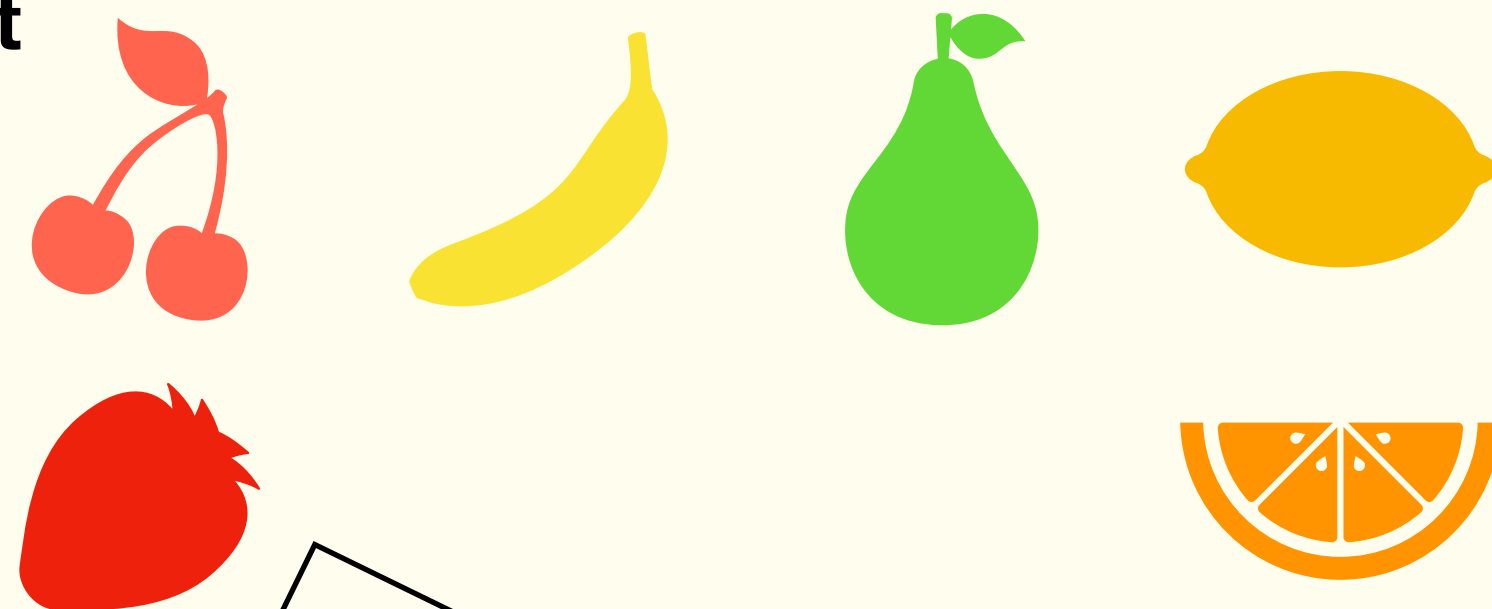

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Enum Season



Class Fruit



UseCase



```
public class Main {  
    public static void main(String[] args) {  
        4 usages  
        enum Season {SPRING, SUMMER, AUTUMN, WINTER}  
  
        /**  
         * 컴파일 타임 - Season 생성자 내부  
         * SPRING = new Season("SPRING", 0);  
         * SUMMER = new SUMMER("SUMMER" 1);  
         * AUTUMN = new SUMMER("AUTUMN" 2);  
         * WINTER = new SUMMER("WINTER" 3);  
         */  
  
        int ordinal0 = Season.SPRING.ordinal();  
        int ordinal1 = Season.SUMMER.ordinal();  
        int ordinal2 = Season.AUTUMN.ordinal();  
        int ordinal3 = Season.WINTER.ordinal();  
  
        System.out.println(ordinal0); // 0  
        System.out.println(ordinal1); // 1  
        System.out.println(ordinal2); // 2  
        System.out.println(ordinal3); // 3  
    }  
}
```

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기



Season 배열

Index 0 - Set

Index 1 - Set

Index 2 - Set

Index 3 - Set

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기



Season 배열

- Index 0 - Set<Fruit>
- Index 1 - Set<Fruit>
- Index 2 - Set<Fruit>
- Index 3 - Set<Fruit>

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```



Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```



Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```

```
(Set<Fruit>[]) new Set[Season.values().length];
```



Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```

```
(Set<Fruit>[])new Set[Season.values().length];
```

비검사 경고 발생! ClassCastException 주의



Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```

```
Set<Fruit>[] buckets = (Set<Fruit>[])new Set[Season.values().length];
```

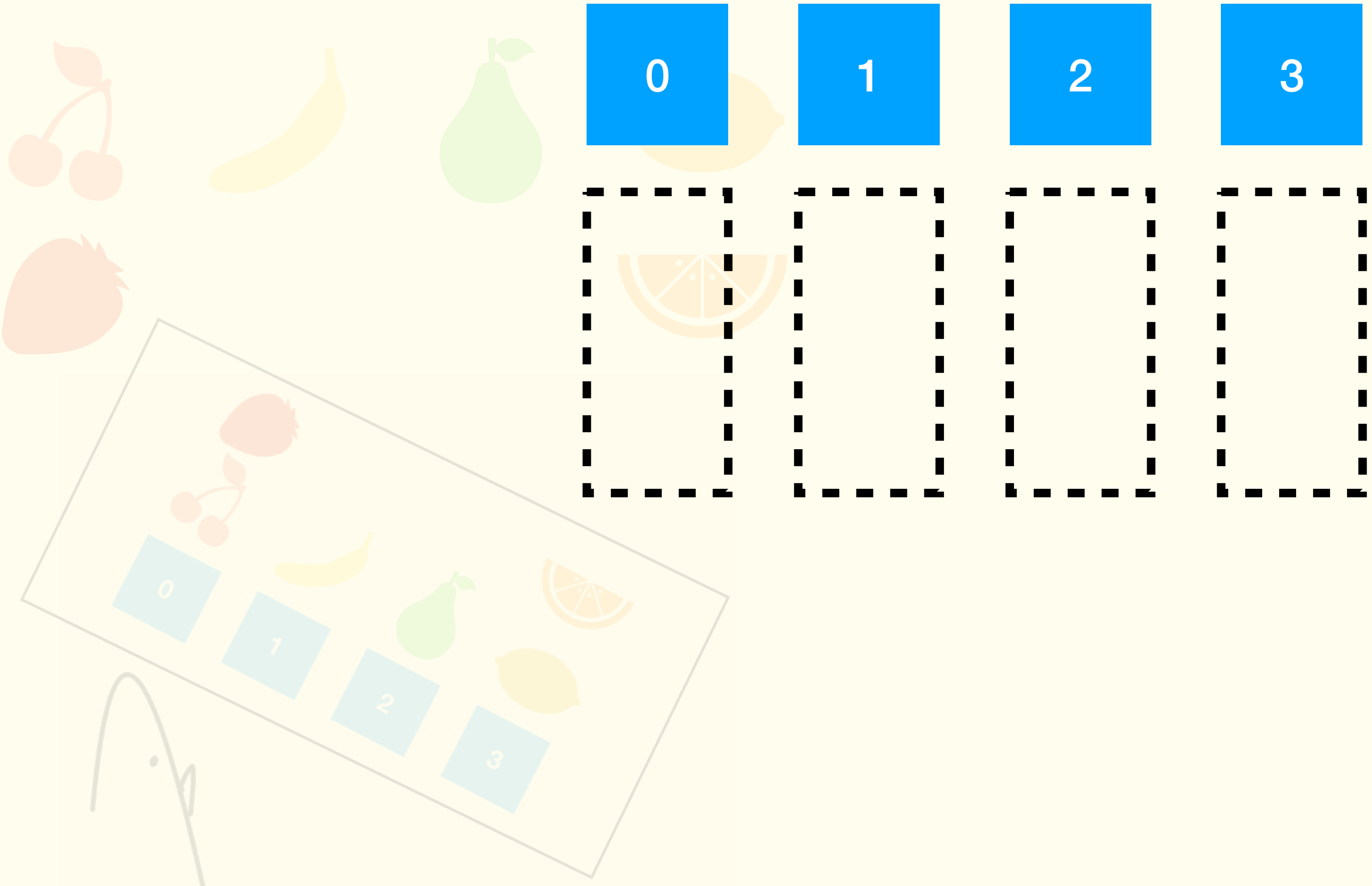


Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];  
new Set<Fruit>[Season.values().length];
```

```
Set<Fruit>[] buckets = (Set<Fruit>[])new Set[Season.values().length];
```



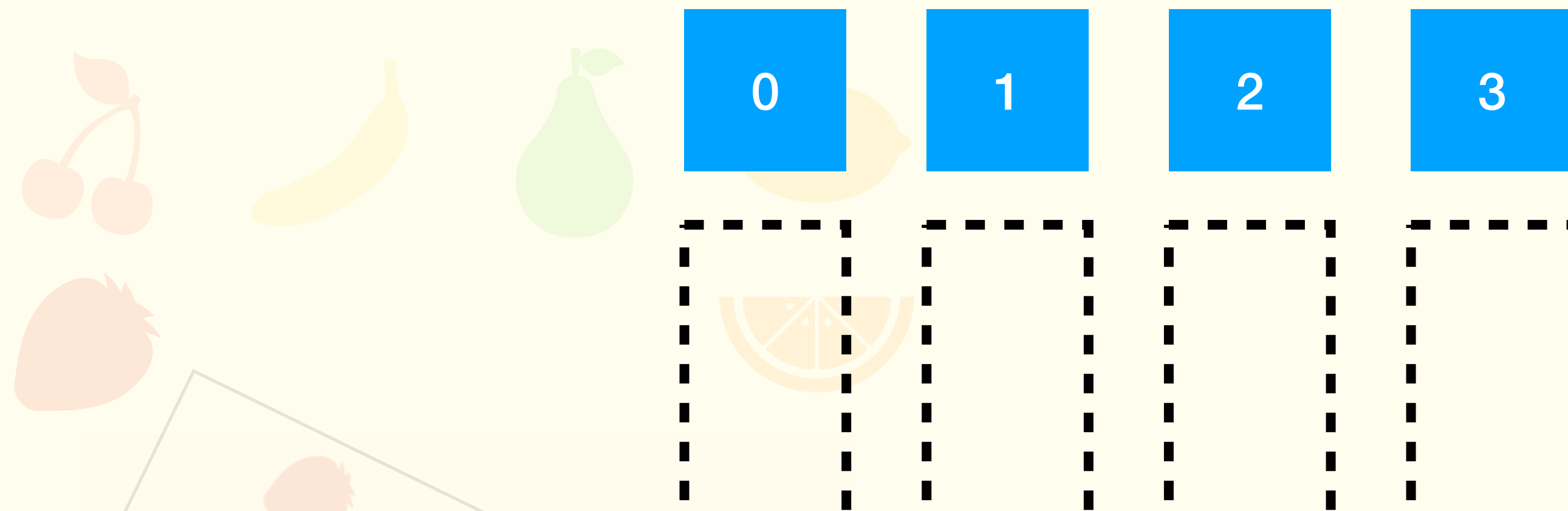
Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```

```
Set<Fruit>[] buckets = (Set<Fruit>[])new Set[Season.values().length];
```

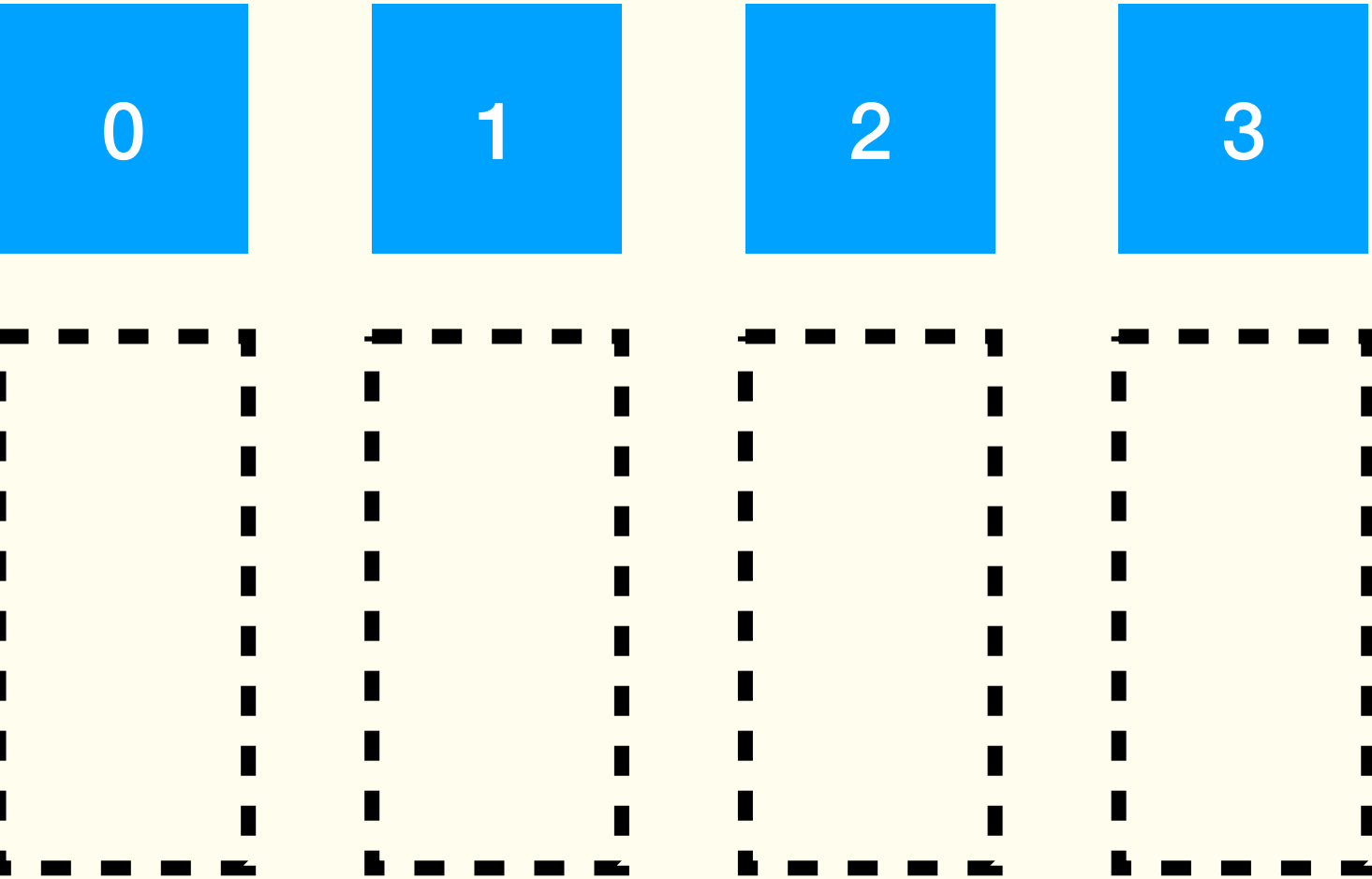


```
for( Season s : Season.values() ){  
    buckets[s.ordinal()] = new HashSet<>();  
}
```


Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

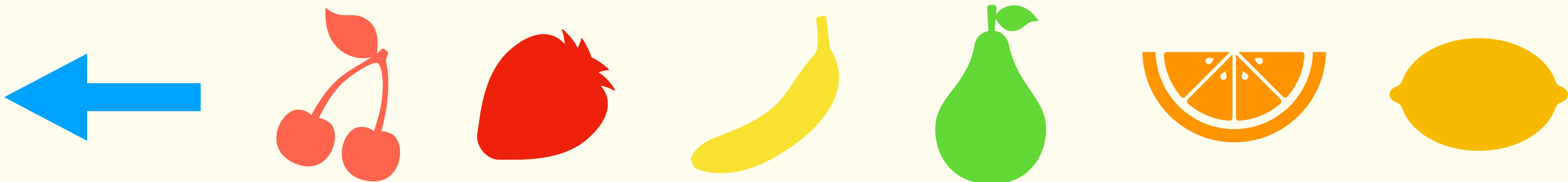
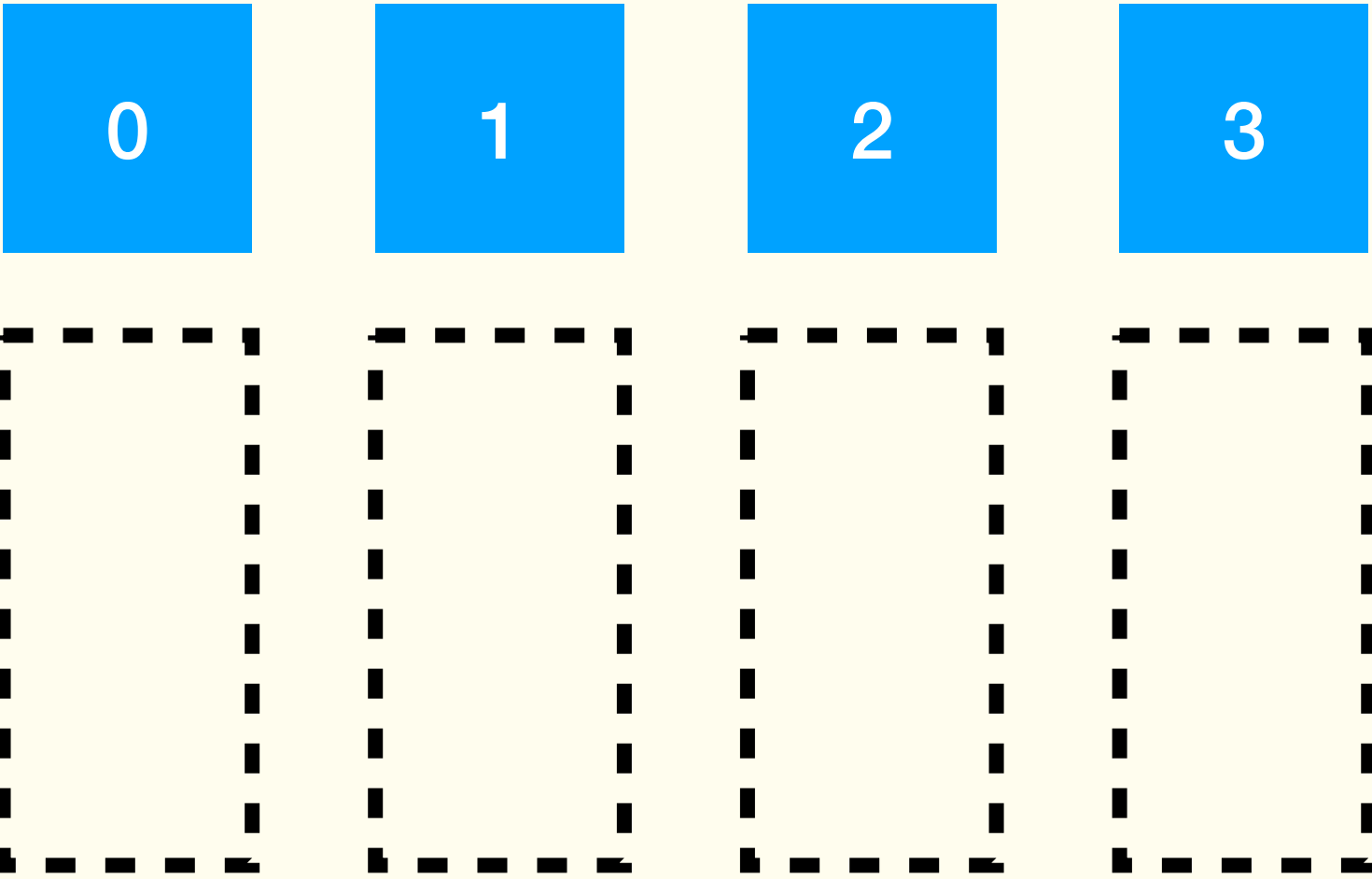
Set<Fruit>[] buckets =



Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =

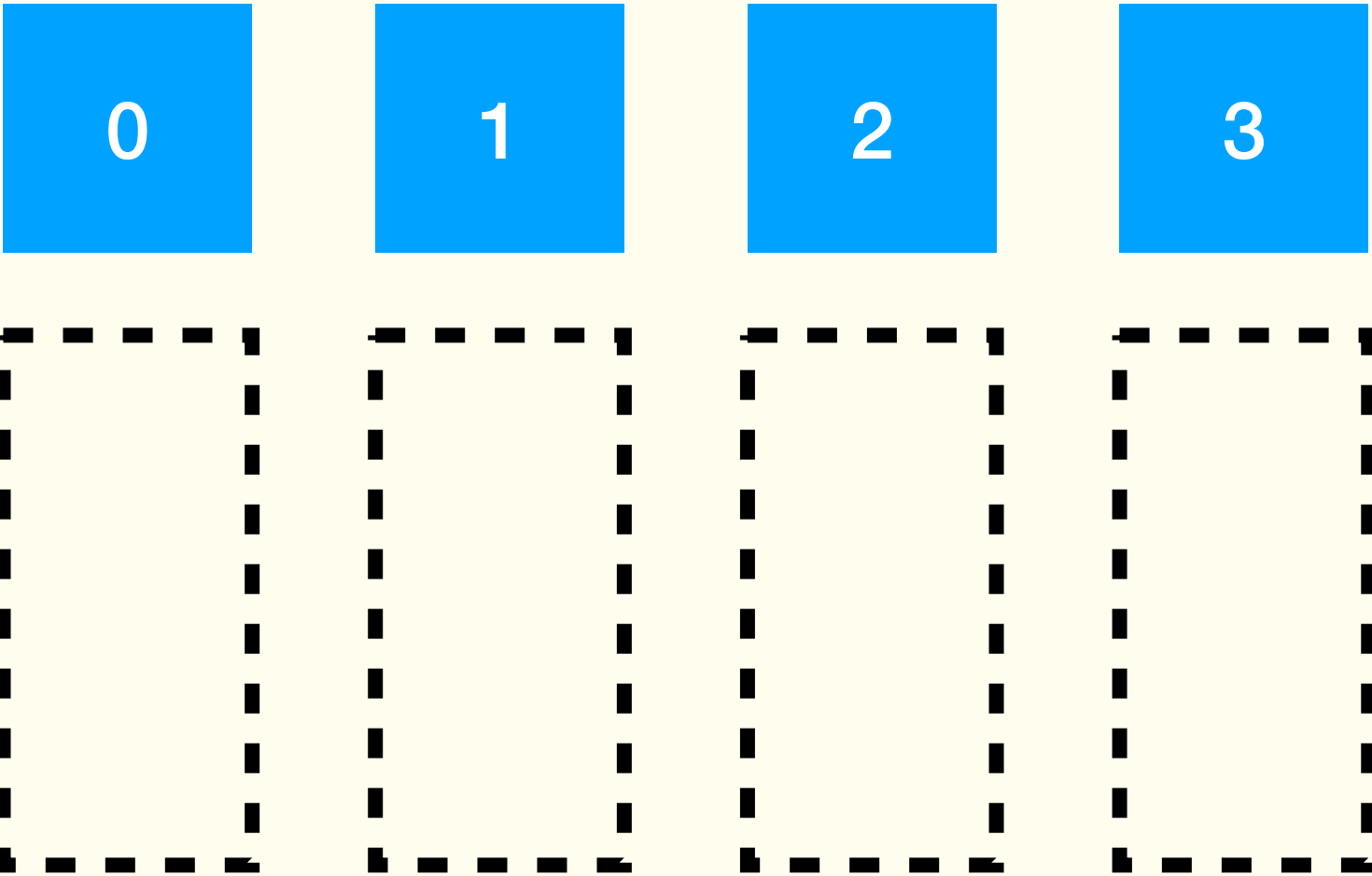


```
List<Fruit> fruits = List.of(  
    new Fruit("체리", Season.SPRING),  
    new Fruit("딸기", Season.SPRING),  
    new Fruit("바나나", Season.SUMMER),  
    ...  
);
```

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =



```
List<Fruit> fruits = List.of(  
    new Fruit("체리", Season.SPRING),  
    new Fruit("딸기", Season.SPRING),  
    new Fruit("바나나", Season.SUMMER),  
    ...  
);
```

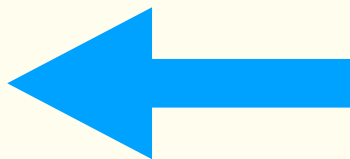
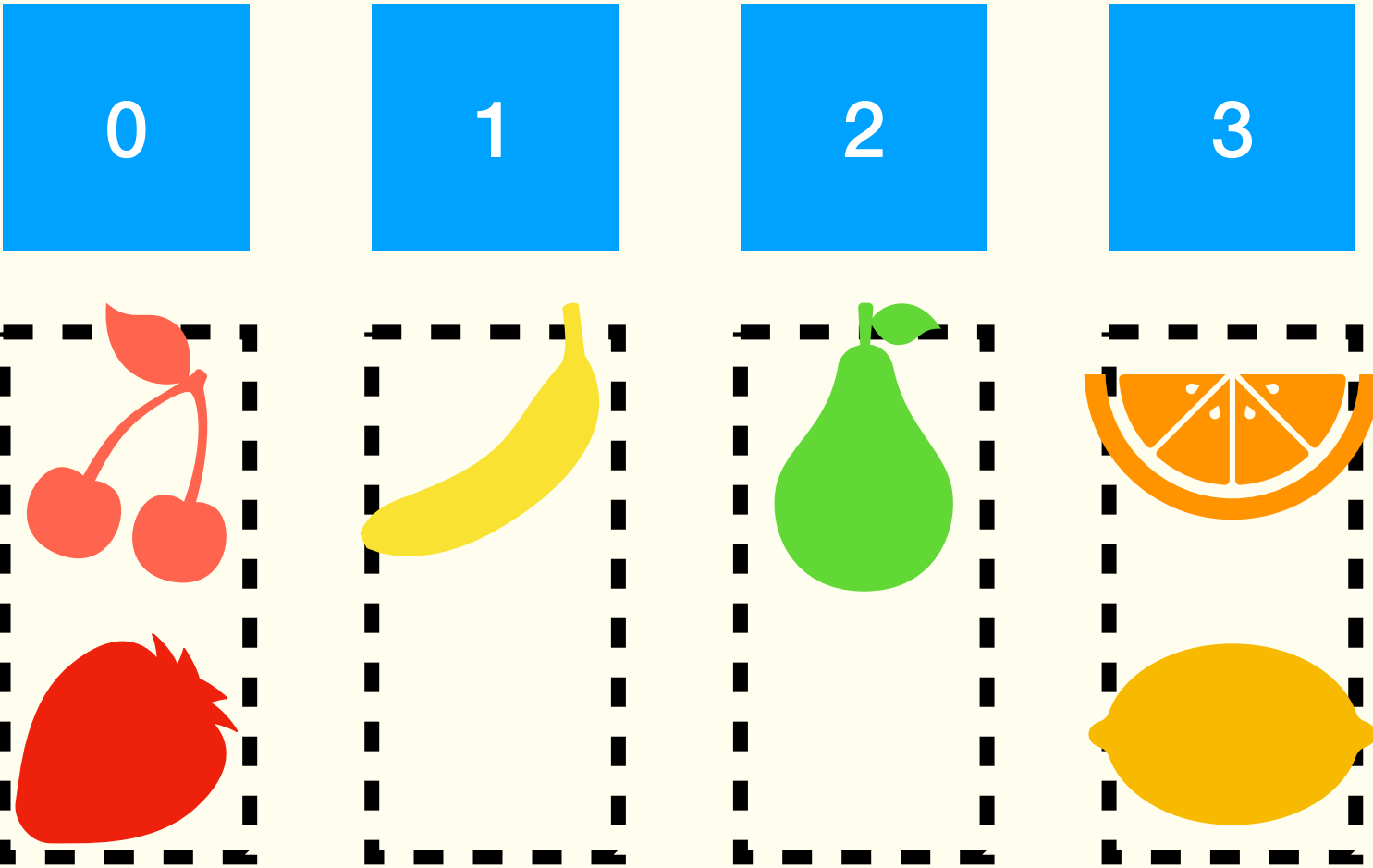


```
for (Fruit fruit : fruits) {  
    buckets[fruit.season().ordinal()].add(fruit);  
}
```

Ordinal() + 배열 인덱싱 예제

계절과일 배열 만들기

Set<Fruit>[] buckets =



```
List<Fruit> fruits = List.of(
    new Fruit("체리", Season.SPRING),
    new Fruit("딸기", Season.SPRING),
    new Fruit("바나나", Season.SUMMER),
    ...
)
```

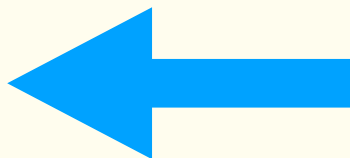
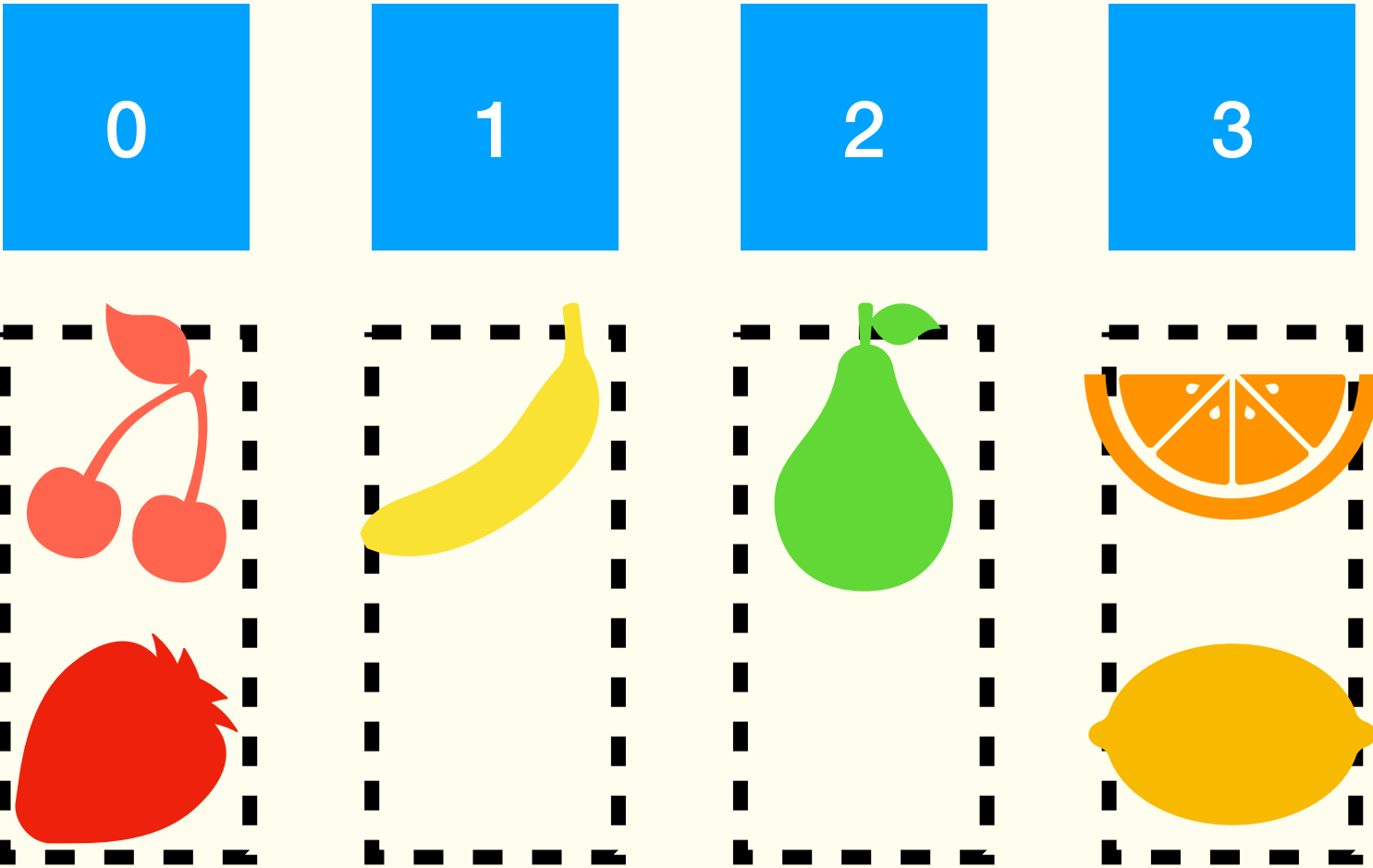
과일의 계절번호를 찾아서, 버킷에 과일 넣기

```
for (Fruit fruit : fruits) {
    buckets[fruit.season().ordinal()].add(fruit);
}
```

Ordinal() + 배열 인덱싱 예제

계절과일 배열 만들기

Set<Fruit>[] buckets =



```
List<Fruit> fruits = List.of(
    new Fruit("체리", Season.SPRING),
    new Fruit("딸기", Season.SPRING),
    new Fruit("바나나", Season.SUMMER),
    ...
);
```

과일의 계절번호를 찾아서, 버킷에 과일 넣기

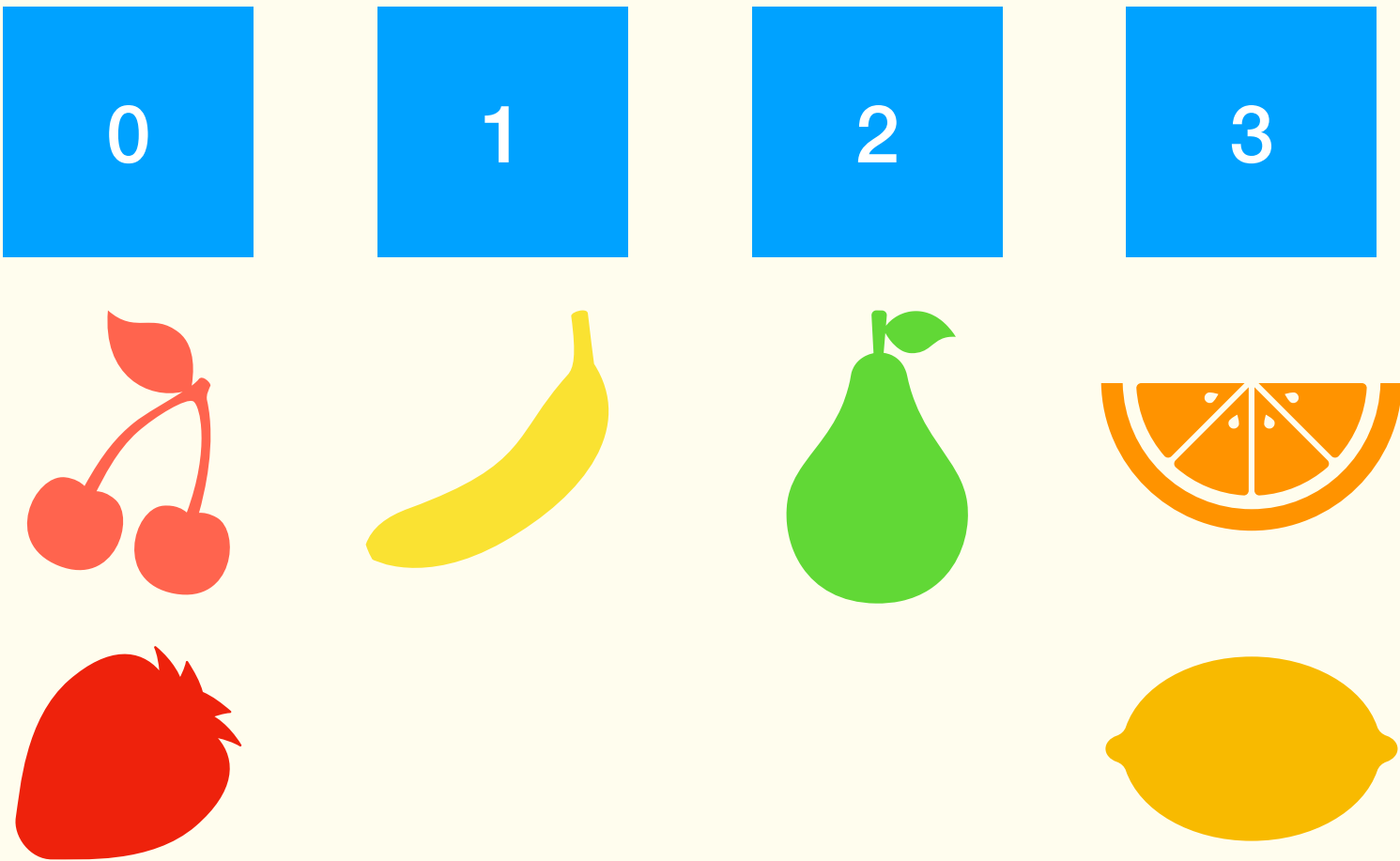
```
for (Fruit fruit : fruits) {
    buckets[fruit.season().ordinal()].add(fruit);
}
```

fruit.season().ordinal()이 무슨 계절인지?
코드만으로는 불명

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =



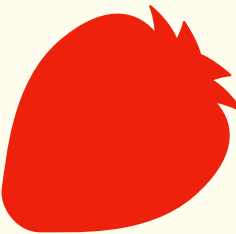
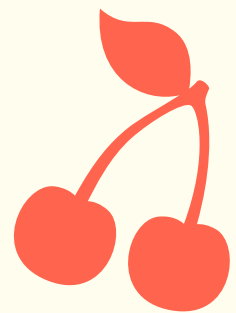
Season 이넘 클래스에
새로운 계절이 들어온다면..

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =

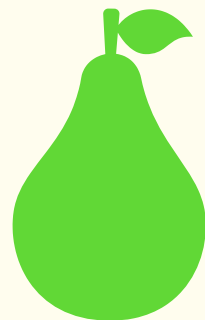
0



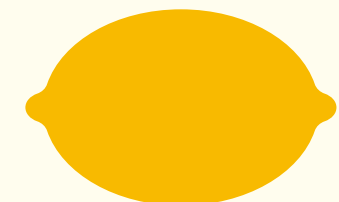
1



2



3

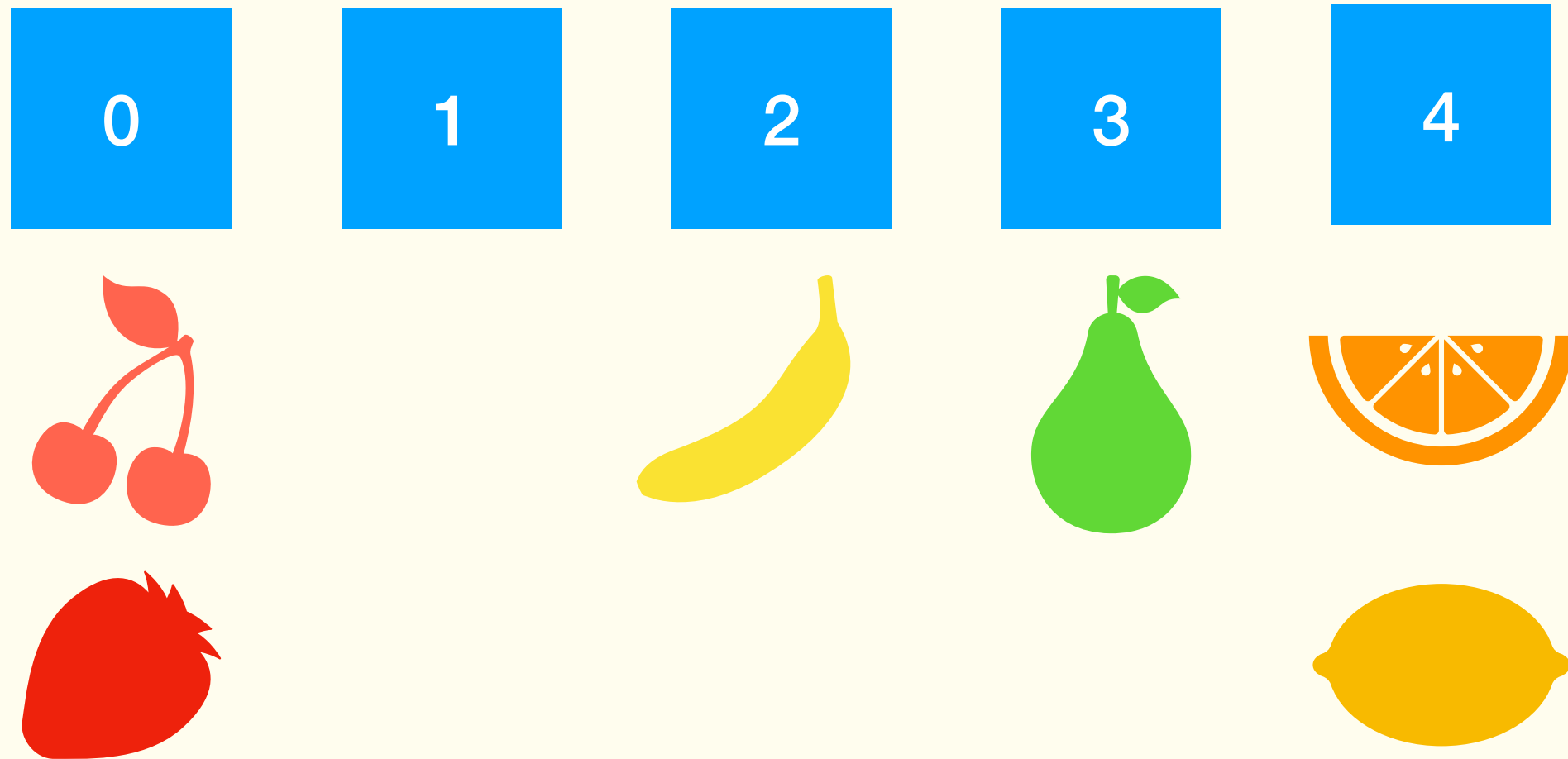


```
public enum Season {  
    SPRING,  
    PRE_SUMMER, //추가  
    SUMMER,  
    AUTUMN,  
    WINTER,  
}
```

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =

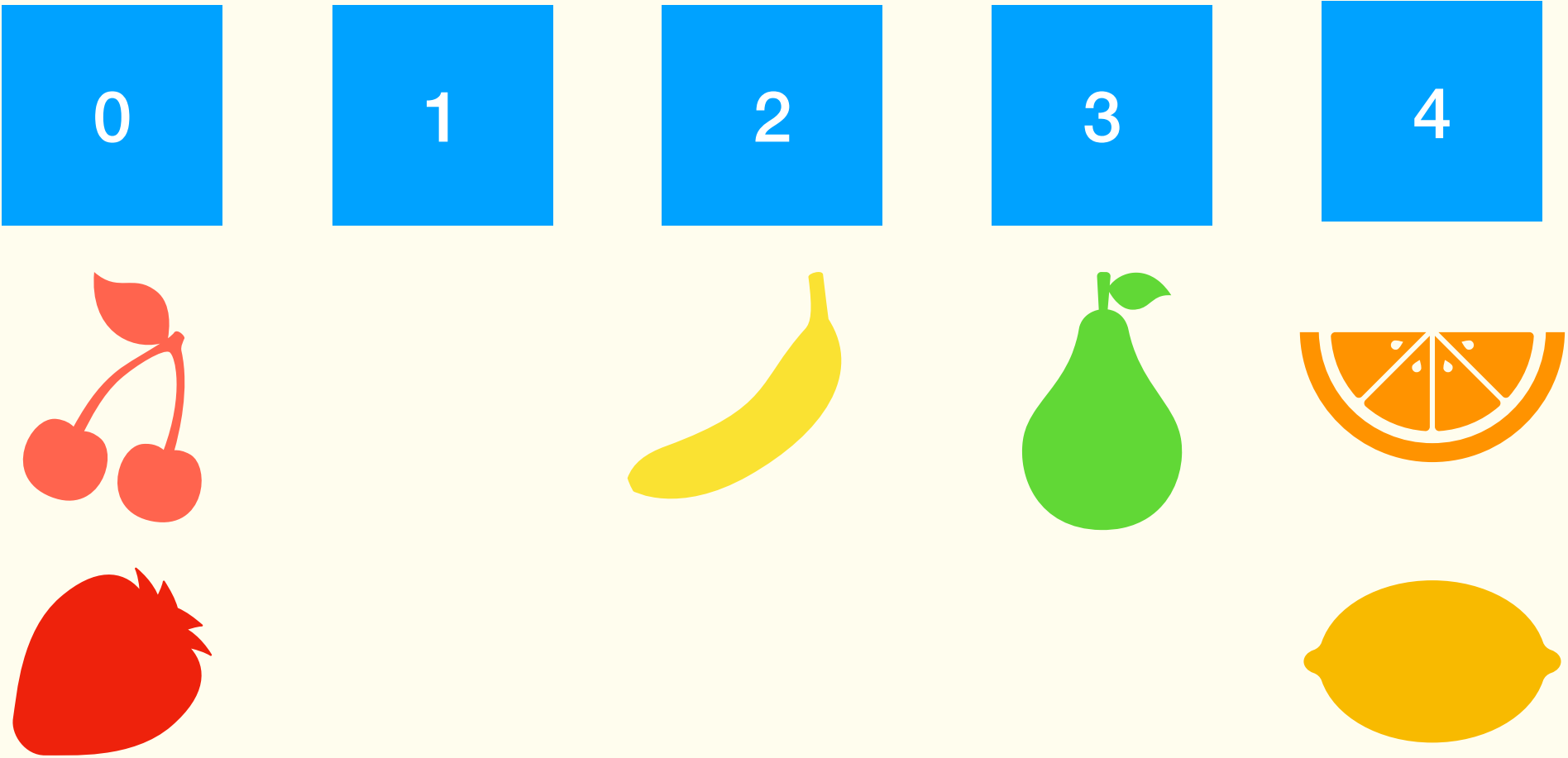


```
public enum Season {  
    SPRING,  
    PRE_SUMMER, //추가  
    SUMMER,  
    AUTUMN,  
    WINTER,  
}
```

Ordinal() + 배열 인덱싱 예제

제철과일 배열 만들기

Set<Fruit>[] buckets =



PRE_SUMMER 추가된 것을 잊어버리고,

Index 1 = 여름이라 생각하고 코딩한다면?

작동은 하지만.. 무슨 문제가 있나?

- **타입 안정성 부족** - Set<Fruit>[] 강제캐스팅
 - 런타임 ClassCastException 위험

```
new Set<Fruit>[4];
```

```
new Set<Fruit>[Season.values().length];
```

```
3 (Set<Fruit>[])new Set[Season.values().length];
```

비검사 경고 발생! ClassCastException 주의

작동은 하지만.. 무슨 문제가 있나?

- **가독성 저하** - 의미를 코드만으로 부족
 - 유지보수 난이도 상승

● ● ● 과일의 계절번호를 찾아서, 버킷에 과일 넣기

```
for (Fruit fruit : fruits) {  
    buckets[fruit.season().ordinal()].add(fruit);  
}
```

fruit.season().ordinal()이 무슨 계절인지?
코드만으로는 불명

작동은 하지만.. 무슨 문제가 있나?

- **정숫값 보증 불가** - 잘못된 인덱스도 컴파일 통과
 - 잘못된 인덱스도 컴파일 통과 -> 조용한 오작동
- **변경 취약** - 계절 순서 변경시, 매직넘버로 코딩된 코드 전체 인덱스 재배치
 - 잠복 버그

```
public enum Season {  
    SPRING,  
    PRE_SUMMER, //추가  
    SUMMER,  
    AUTUMN,  
    WINTER,  
}
```

02 EnumMap이 가져다주는 구조적 해결

“배열 성능은 유지하고, 안정성을 보장한다”

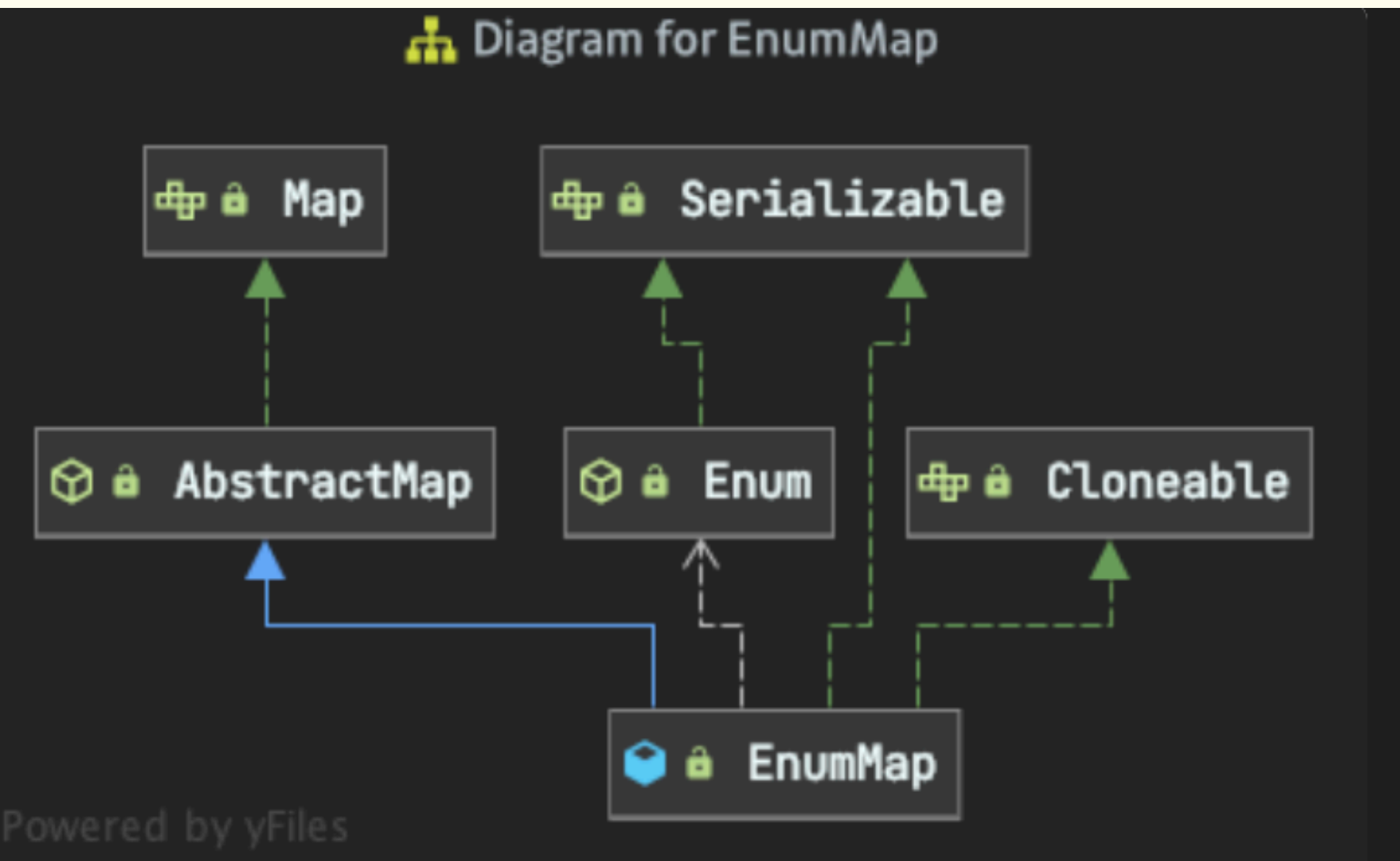
EnumMap 소개 및 목적

내부 구조 및 동작 원리

ordinal() 인덱싱 배열 vs EnumMap

EnumMap 소개 및 목적

“EnumMap은 ordinal() 의 위험을 모두 캡슐화한 실전형 솔루션이다.”



EnumMap의 특징

- EnumMap은 내부적으로 데이터를 저장할 때 배열을 사용한다.
- 기본 연산들이 모두 상수 시간 안에 실행된다.
- 캐싱을 통한 성능 향상을 위해 키 배열 또한 가지고 있다.
- equals 연산을 할 때 내부의 key와 value가 모두 일치하는지 전체 순회를 통해 확인한다.
- AbstractMap 클래스를 상속받으므로, Map 인터페이스 또한 상속 받는다.
- Thread-safe하지 않다. (필요하다면 Collections.synchronizedMap으로 wrap하는것을 권장한다.)

EnumMap vs HashMap

모든 연산을 다루기에는 내용이 많으니, put과 get 연산만 비교해보도록 하자.

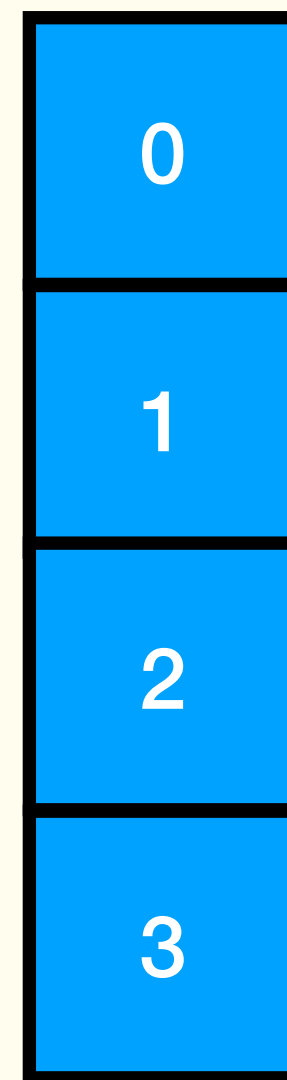
EnumMap 소개 및 목적

“EnumMap은 ordinal() 의 위험을 모두 캡슐화한 실전형 솔루션이다.”

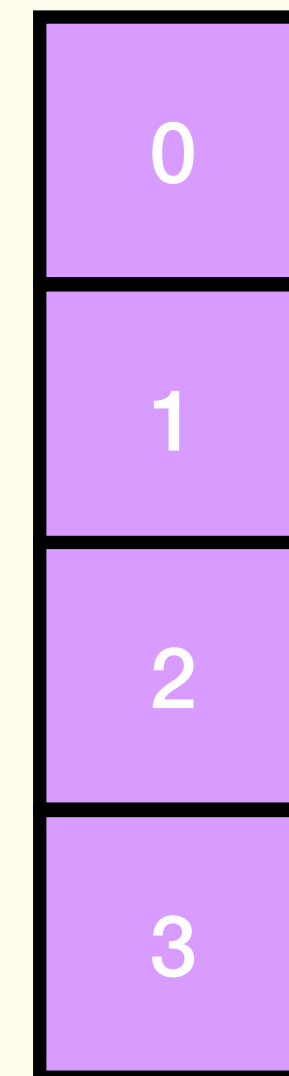
내부 구조 및 동작 원리: 초기화

```
public static void main(String[] args) {  
    /** 초기화 */  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
    Set<Fruit> fruits = Set.of(  
        new Fruit(name: "체리", Season.SPRING),  
        new Fruit(name: "딸기", Season.SPRING)  
    );  
}
```

Season[] keys



Object[] values :

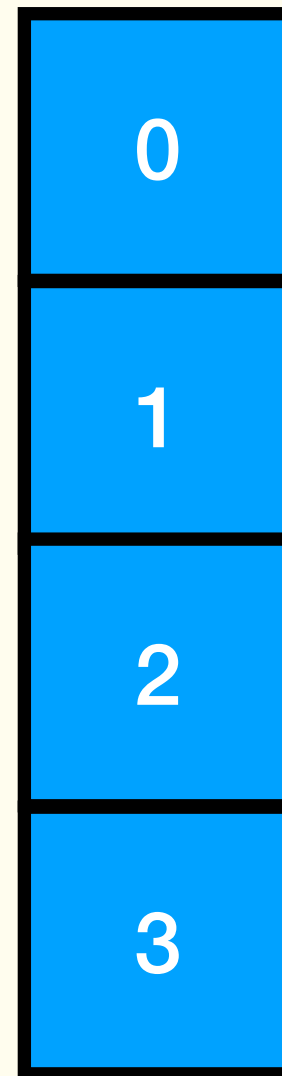


내부 구조 및 동작 원리: 초기화

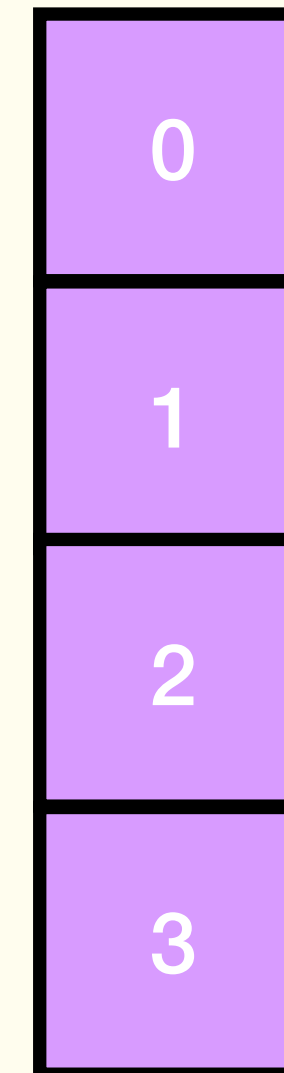
```
public enum Season {  
    SPRING, SUMMER,  
    AUTUMN, WINTER  
}
```

```
new EnumMap<>(Season.class);
```

Season[] keys



Object[] values :



내부 구조 및 동작 원리: 초기화

```
public enum Season {  
    SPRING, SUMMER,  
    AUTUMN, WINTER  
}
```

```
new EnumMap<>(Season.class);
```

Season[] keys

| | |
|-----------------------|---|
| <i>Season.SPRING</i> | 0 |
| <i>Season.SUMMMER</i> | 1 |
| <i>Season.AUTUMN</i> | 2 |
| <i>Season.WINTER</i> | 3 |

Object[] values :

| |
|---|
| 0 |
| 1 |
| 2 |
| 3 |

내부 구조 및 동작 원리: 초기화

```
public enum Season {  
    SPRING, SUMMER,  
    AUTUMN, WINTER  
}
```

```
new EnumMap<> (Season.class);
```

열거형 상수 전체를 복사 없이 얻음

리플렉션

Season[] keys

Season.SPRING

0

Season.SUMMMER

1

Season.AUTUMN

2

Season.WINTER

3

내부 구조 및 동작 원리: 초기화

1. 리플렉션이란??

리플렉션은 힙 영역에 로드 된 Class 타입의 객체를 통해, 접근 제어자 상관없이 원하는 클래스의 정보에 접근해서 조작할 수 있도록 지원하는 API이다.

- 조작할 수 있는 기능들을 나열해보면,
 - 필드 (목록) 가져오기
 - 메소드 (목록) 가져오기
 - 상위 클래스 가져오기
 - 인터페이스 (목록) 가져오기
 - 애노테이션 가져오기
 - 생성자 가져오기
 - 생성자를 통해 인스턴스 객체 생성하기
 - 등등...

내부 구조 및 동작 원리: 초기화

```
public enum Season {  
    SPRING, SUMMER,  
    AUTUMN, WINTER  
}
```

```
new EnumMap<> (Season.class);
```

열거형 상수 전체를 복사 없이 얻음

리플렉션

Season[] keys

Season.SPRING

0

Season.SUMMMER

1

Season.AUTUMN

2

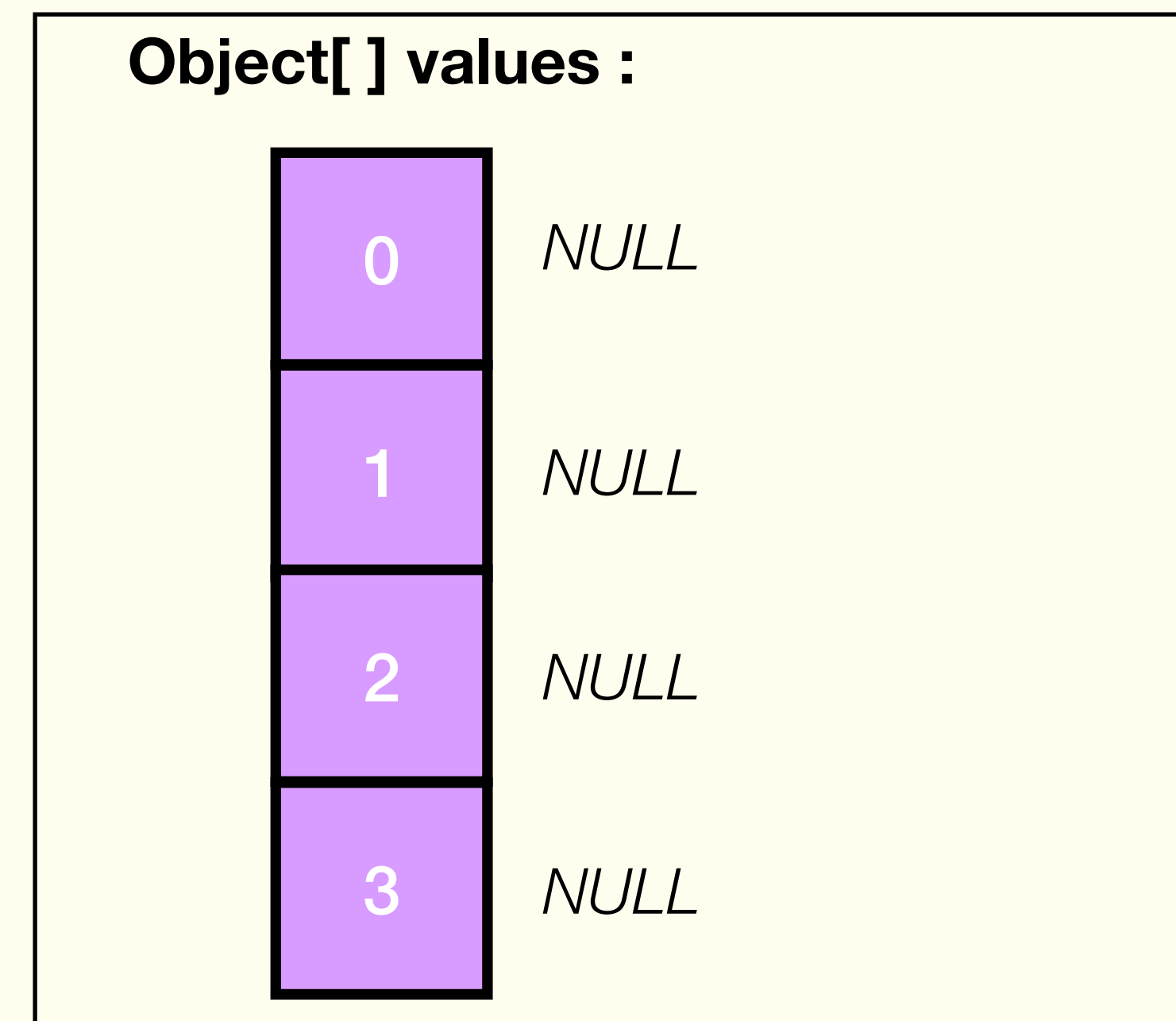
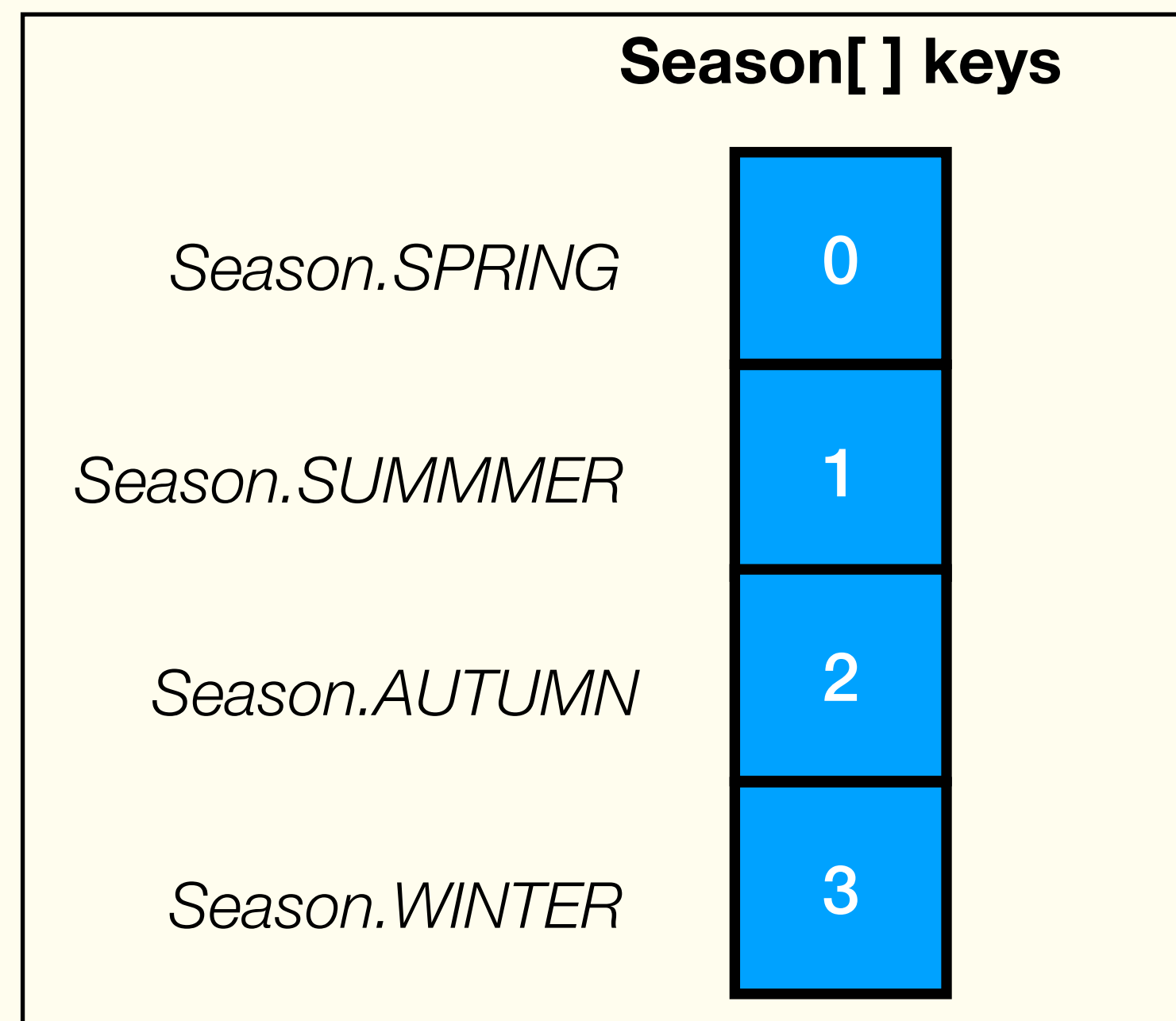
Season.WINTER

3

내부 구조 및 동작 원리: 초기화

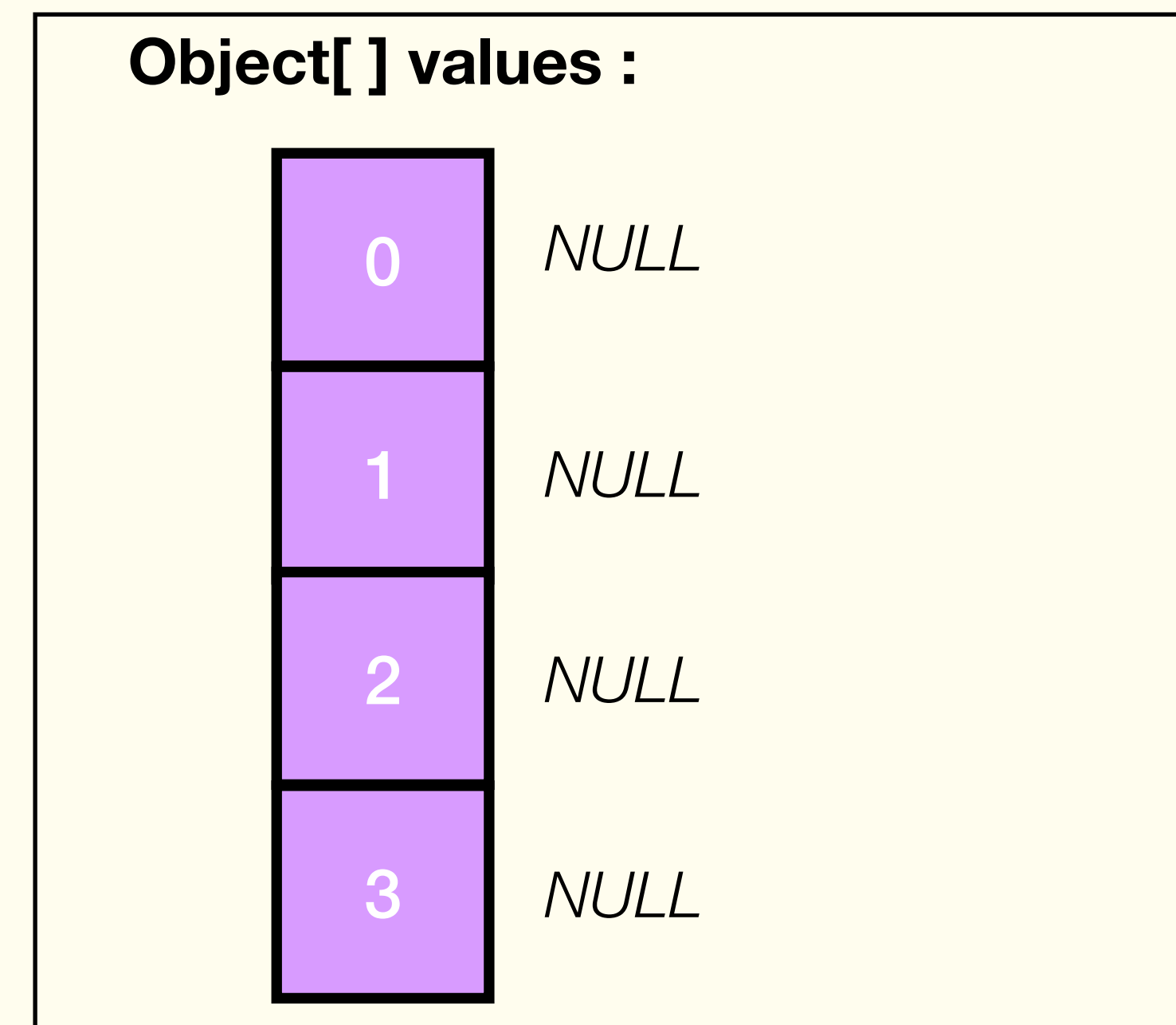
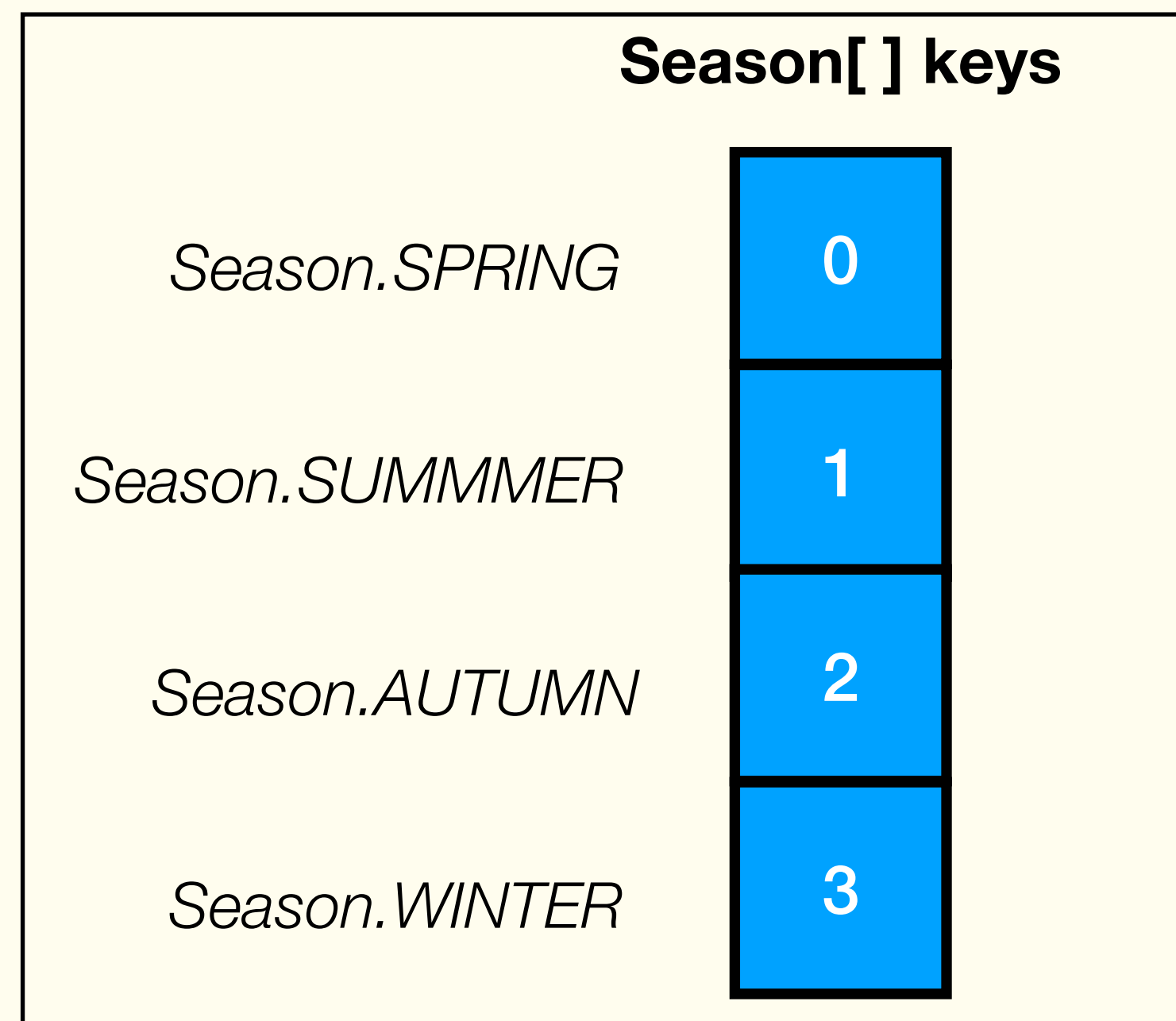
```
public enum Season {  
    SPRING, SUMMER,  
    AUTUMN, WINTER  
}
```

```
new EnumMap<>(Season.class);
```



내부 구조 및 동작 원리: 초기화

```
public static void main(String[] args) {  
    /** 초기화 */  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
}
```



내부 구조 및 동작 원리: put

```
public static void main(String[] args) {  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
    //  
    Set<Fruit> fruits = Set.of(  
        new Fruit(name: "체리", Season.SPRING),  
        new Fruit(name: "딸기", Season.SPRING)  
    );  
    //  
    enumMap.put(Season.SPRING, fruits);  
}
```


내부 구조 및 동작 원리: put

```
public static void main(String[] args) {  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
    //  
    Set<Fruit> fruits = Set.of(  
        new Fruit(name: "체리", Season.SPRING),  
        new Fruit(name: "딸기", Season.SPRING)  
    );  
    //  
    enumMap.put(Season.SPRING, fruits);  
}
```

내부 구조 및 동작 원리: put

```
enumMap.put(Season.SPRING, fruits);
```

내부 구조 및 동작 원리: put

```
enumMap.put(Season.SPRING, fruits);
```



`isValidKey()`

```
0 = Season.SPRING.ordinal();
```

내부 구조 및 동작 원리: put

```
enumMap.put(Season.SPRING, fruits);
```



isValidKey()

0 = Season.SPRING.ordinal();

Season[] keys

| | |
|----------------|---|
| Season.SPRING | 0 |
| Season.SUMMMER | 1 |
| Season.AUTUMN | 2 |
| Season.WINTER | 3 |

Object[] values :

| | |
|---|------|
| 0 | NULL |
| 1 | NULL |
| 2 | NULL |
| 3 | NULL |

내부 구조 및 동작 원리: put

```
enumMap.put(Season.SPRING, fruits);
```

↓ isValidKey()

```
0 = Season.SPRING.ordinal();
```



Object[] values :

| | |
|---|---------------|
| 0 | <i>fruits</i> |
| 1 | NULL |
| 2 | NULL |
| 3 | NULL |

내부 구조 및 동작 원리: get

```
public static void main(String[] args) {  
    /** 초기화 */  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
    Set<Fruit> fruits = Set.of(  
        new Fruit(name: "체리", Season.SPRING),  
        new Fruit(name: "딸기", Season.SPRING)  
    );  
    /** put */  
    enumMap.put(Season.SPRING, fruits);  
    /** get */  
    Set<Fruit> getFruits = enumMap.get(Season.SPRING);  
}
```

내부 구조 및 동작 원리: get

```
public static void main(String[] args) {  
    /** 초기화 */  
    EnumMap<Season, Set<Fruit>> enumMap = new EnumMap<>(Season.class);  
    Set<Fruit> fruits = Set.of(  
        new Fruit(name: "체리", Season.SPRING),  
        new Fruit(name: "딸기", Season.SPRING)  
    );  
    /** put */  
    enumMap.put(Season.SPRING, fruits);  
    /** get */  
    Set<Fruit> getFruits = enumMap.get(Season.SPRING);  
}
```


내부 구조 및 동작 원리: get

```
Set<Fruit> getFruits = enumMap.get(Season.SPRING);
```

↓ isValidKey()

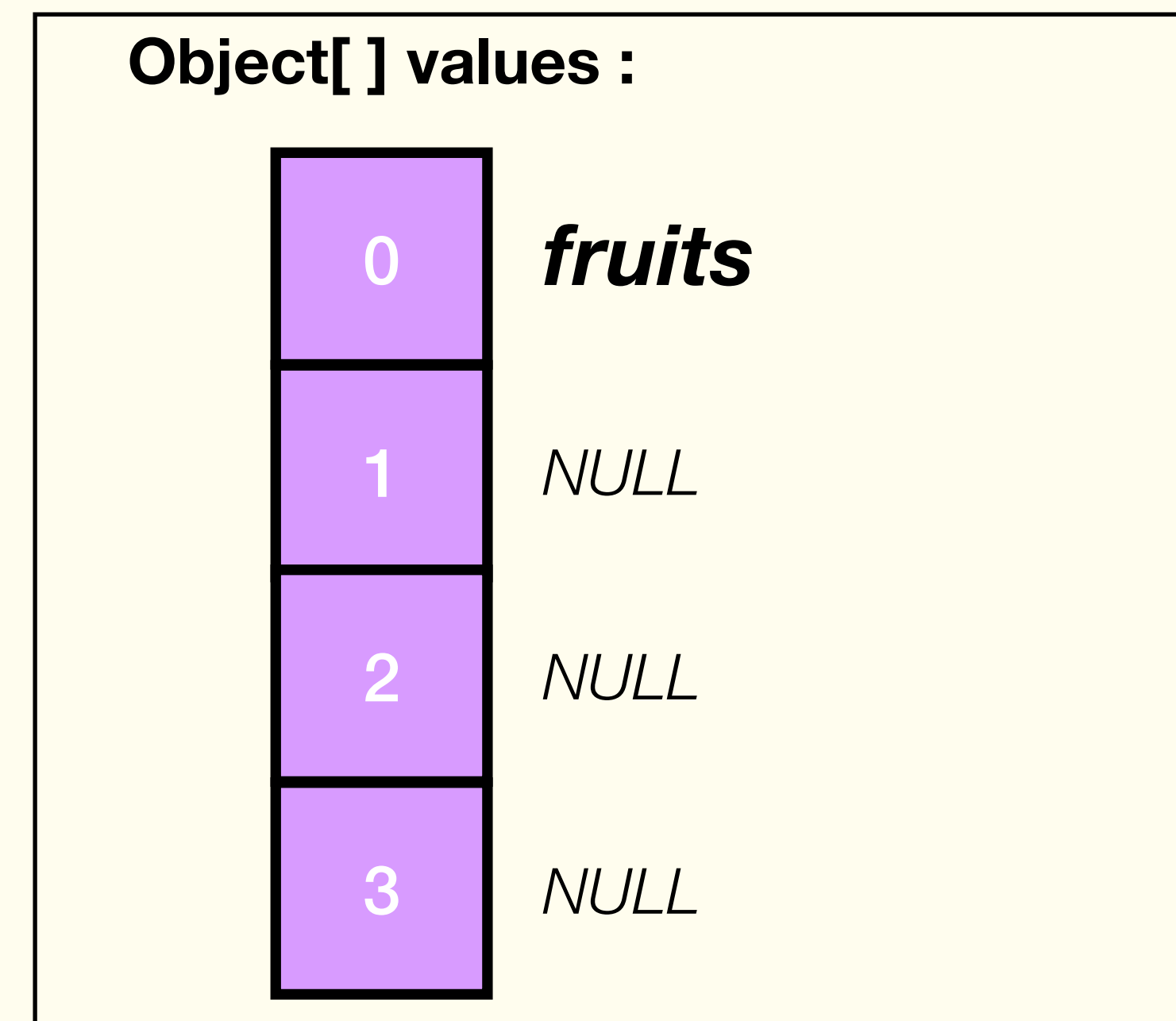
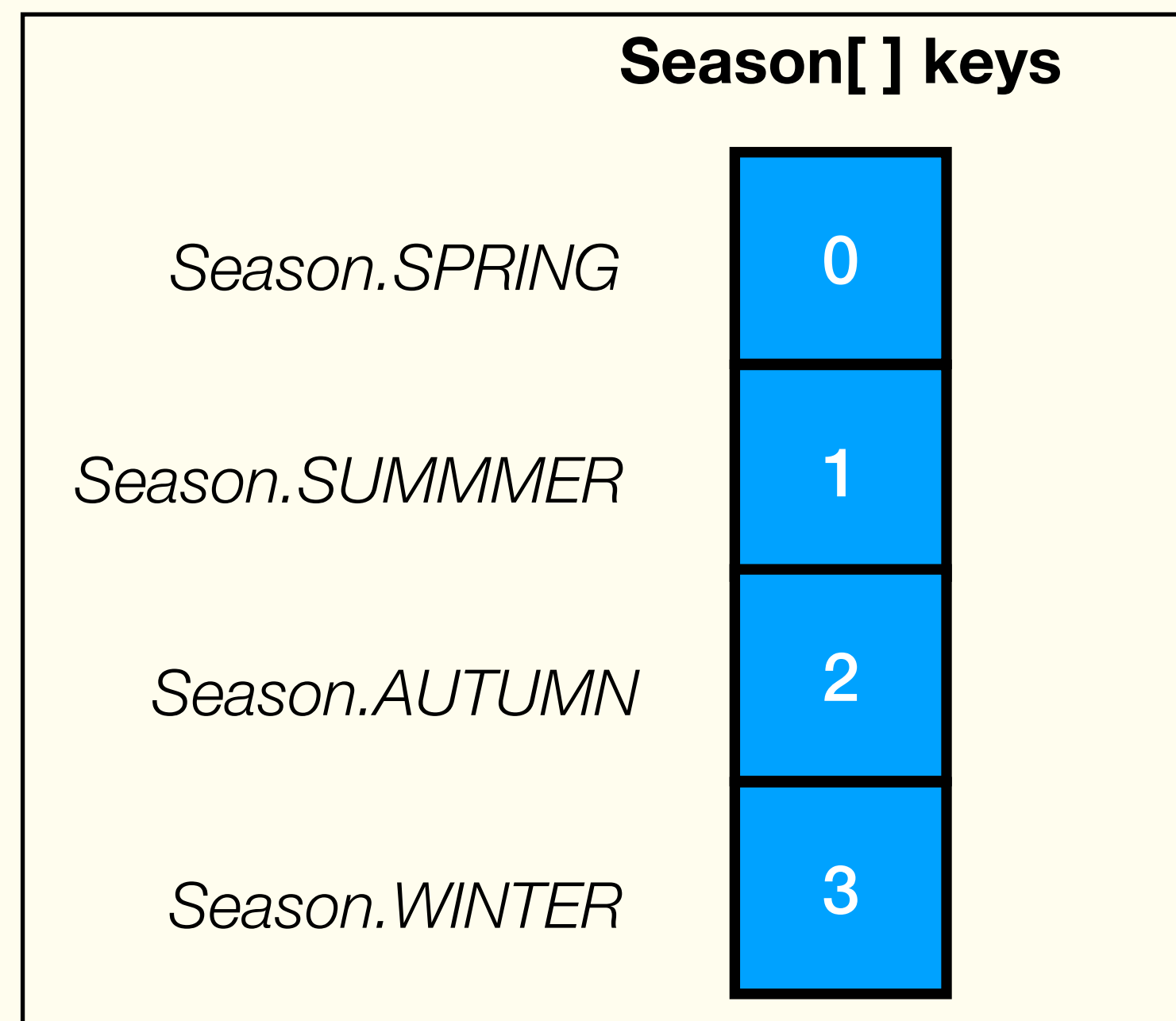
0 = Season.*SPRING*.ordinal();

내부 구조 및 동작 원리: get

```
Set<Fruit> getFruits = enumMap.get(Season.SPRING);
```

↓ isValidKey()

0 = Season.SPRING.ordinal();

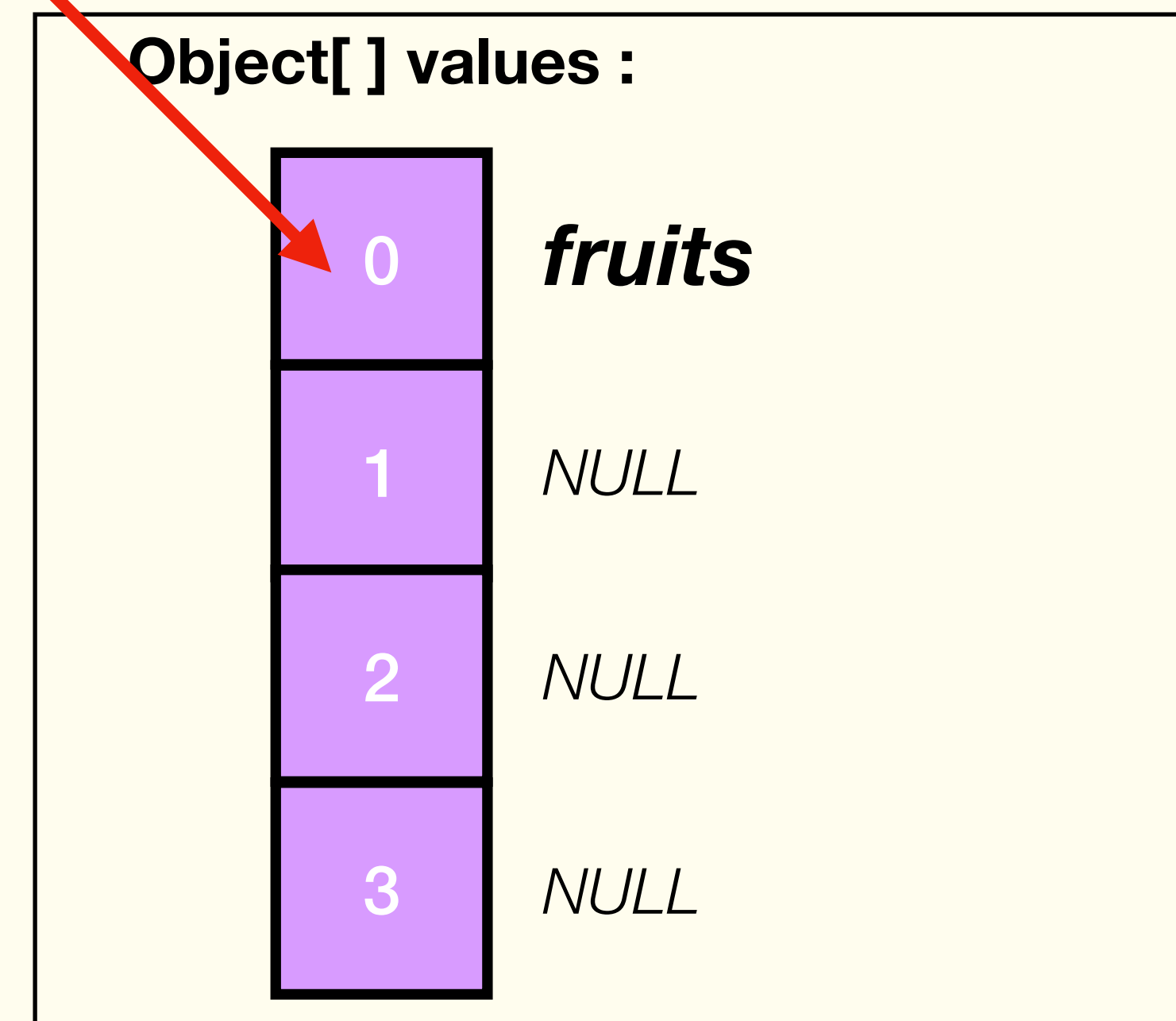
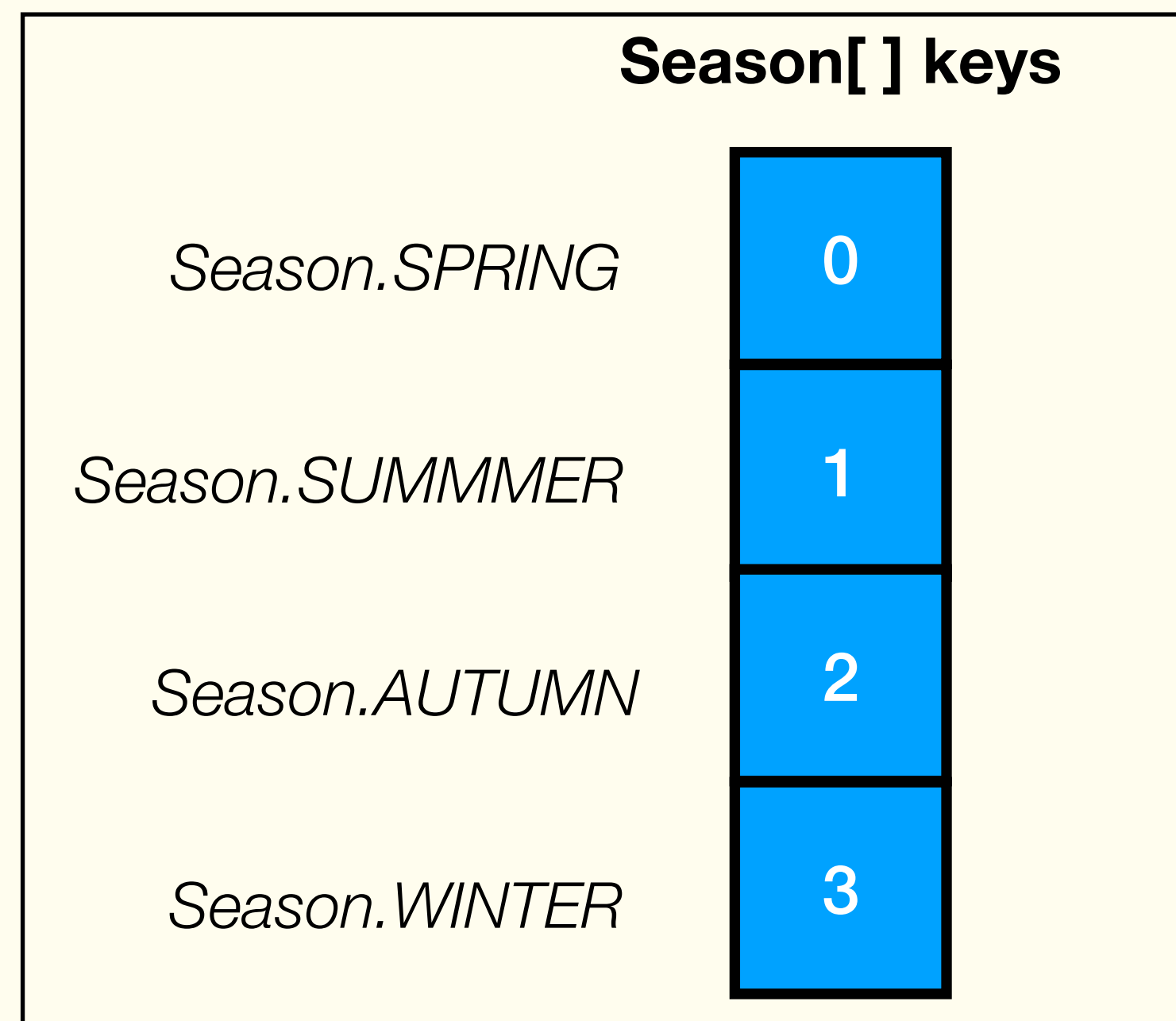


내부 구조 및 동작 원리: get

```
Set<Fruit> getFruits = enumMap.get(Season.SPRING);
```

isValidKey()

0 = Season.SPRING.ordinal();



02 EnumMap이 가져다주는 구조적 해결

“배열 성능은 유지하고, 안정성을 보장한다”


EnumMap 소개 및 목적

내부 구조 및 동작 원리

ordinal() 인덱싱 배열 vs EnumMap

ordinal() 인덱싱 배열 vs EnumMap

ordinal() 인덱싱 배열

 초기화

```
(Set<Fruit>[])new Set[Season.values().length];
```

- Set<Fruit>[] 강제캐스팅 -> 언체크 예외 문제

```
for (Fruit fruit : fruits) {  
    buckets[fruit.season().ordinal()].add(fruit);  
}
```

EnumMap

```
new EnumMap<>(Season.class);
```

- 파라미터로 Enum 타입 받음

```
enumMap.put(Season.SPRING, fruits);  
enumMap.get(Season.SPRING);
```

ordinal() 인덱싱 배열 vs EnumMap

ordinal() 인덱싱 배열

```
(Set<Fruit>[])new Set[Season.values().length];
```

put/get

```
for (Fruit fruit : fruits) {  
    buckets[fruit.season().ordinal()].add(fruit);  
}
```

- Enum의 ordinal()을 인덱스로 활용 -> 의미 미흡

EnumMap

```
new EnumMap<>(Season.class);
```

```
enumMap.put(Season.SPRING, fruits);
```

```
enumMap.get(Season.SPRING);
```

- Enum 상수명을 Key로 활용

ordinal() 인덱싱 배열 vs EnumMap

ordinal() 인덱싱 배열

```
new Set<Fruit>[Season.values().length];
```

```
for (Fruit fruit : fruits) {  
    buckets[fruit.season().ordinal()].add(fruit);  
}
```

EnumMap

```
new EnumMap<>(Season.class);
```

```
enumMap.put(Season.SPRING, fruits);
```

```
enumMap.get(Season.SPRING);
```

if 상수 순서 변경

- 매직넘버로 코딩된 코드 전체 인덱스 재배치

- 매직넘버 미사용



이펙티브자바 Ch6.

열거 타입과 애너테이션

Item 37.

Enum의 효율전략: Ordinal 인덱싱 대신 EnumMap을 사용하라

결론

EnumMap은 ordinal()의 위험을 모두 캡슐화한 실전형 솔루션이다.
가능한 enum 키 매핑은 EnumMap을 사용하자.

2025.05.19

김진수
