Programming in JAVA.

Prof. 박은종
Edit. 김중일

Collection & Map

Array
배열

같은 형의 데이터 타입을
메모리에 저장하는
선형적 자료구조.

논리적 구조와
물리적 구조가
동일합니다

Array
배열

n0	n1	n2	n3	n4	...
가	나	다	라	마	...

*JDK
- ArrayList
- Vector

Linked List

데이터와 링크로 구성

Linked List

배열과 비교하면?

* JDK
-LinkedList

Stack

일반적인 의미

- 쌓다.
- 더미.

Stack

자료구조에서의 의미

- 선형 자료구조
- LIFO (Last In First Out)

// 나중에 들어온 분이 먼저 나가는 구조.

*JDK
- Stack

Stack

Push?

*스택의 저장공간이 넘치면 Overflow현상이 일어날 수 있습니다.

Stack

Pop?

Stack

Peek?

- 스택의 맨 위에 있는 원소를 반환

```
//스택에서 제거하지는 않음.
//일종의 get().
```

Queue

일반적인 의미

- 대기열.

Queue

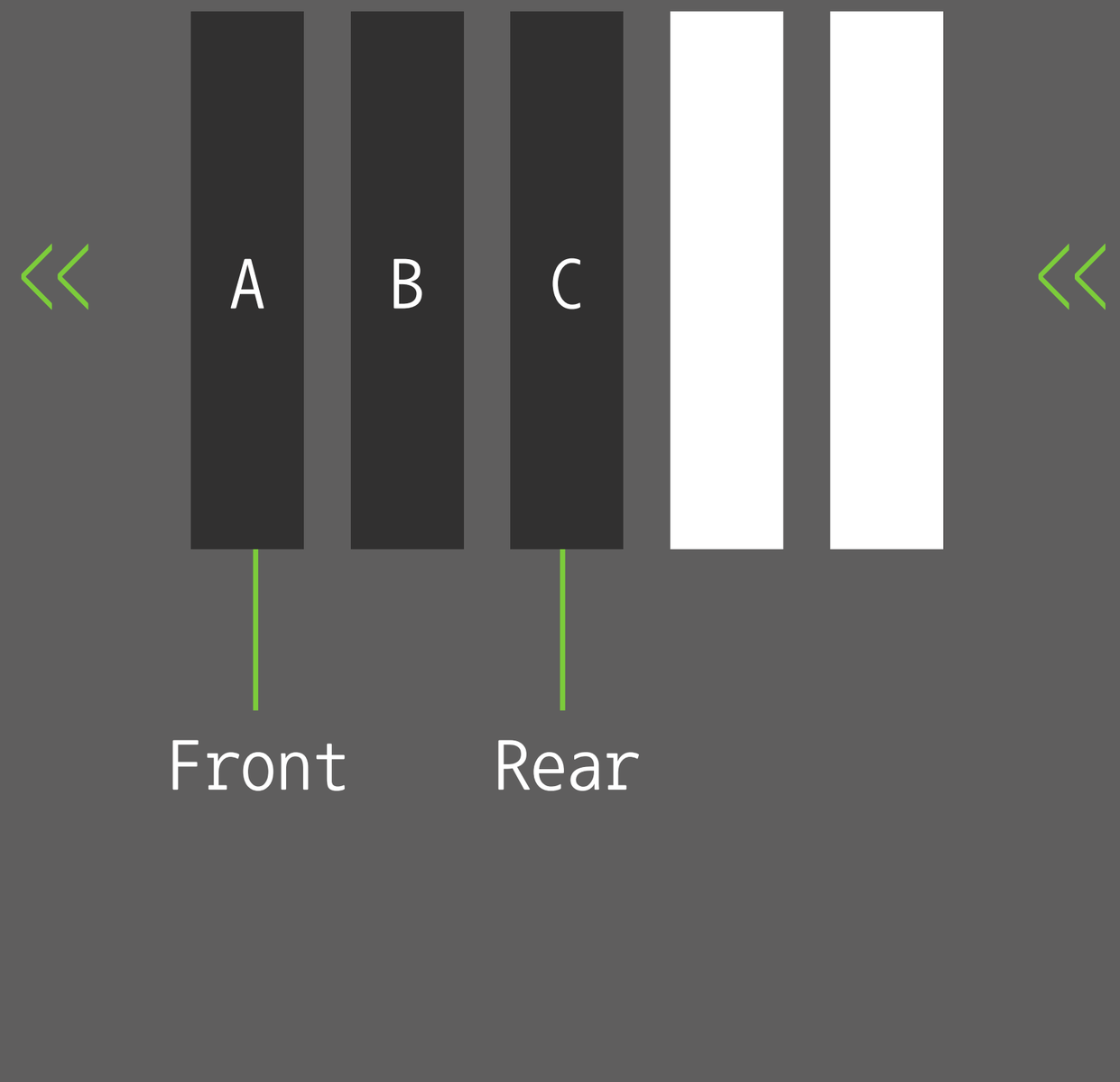
자료구조에서의 의미

- FIFO (First In First Out)

// 먼저 들어온 분이 먼저 나가는 구조.

*JDK
- ArrayList
- Vector
- LinkedList

Queue



Tree

Binary Search Tree
를 왜 쓰나요?

자료의 ‘검색’

Tree

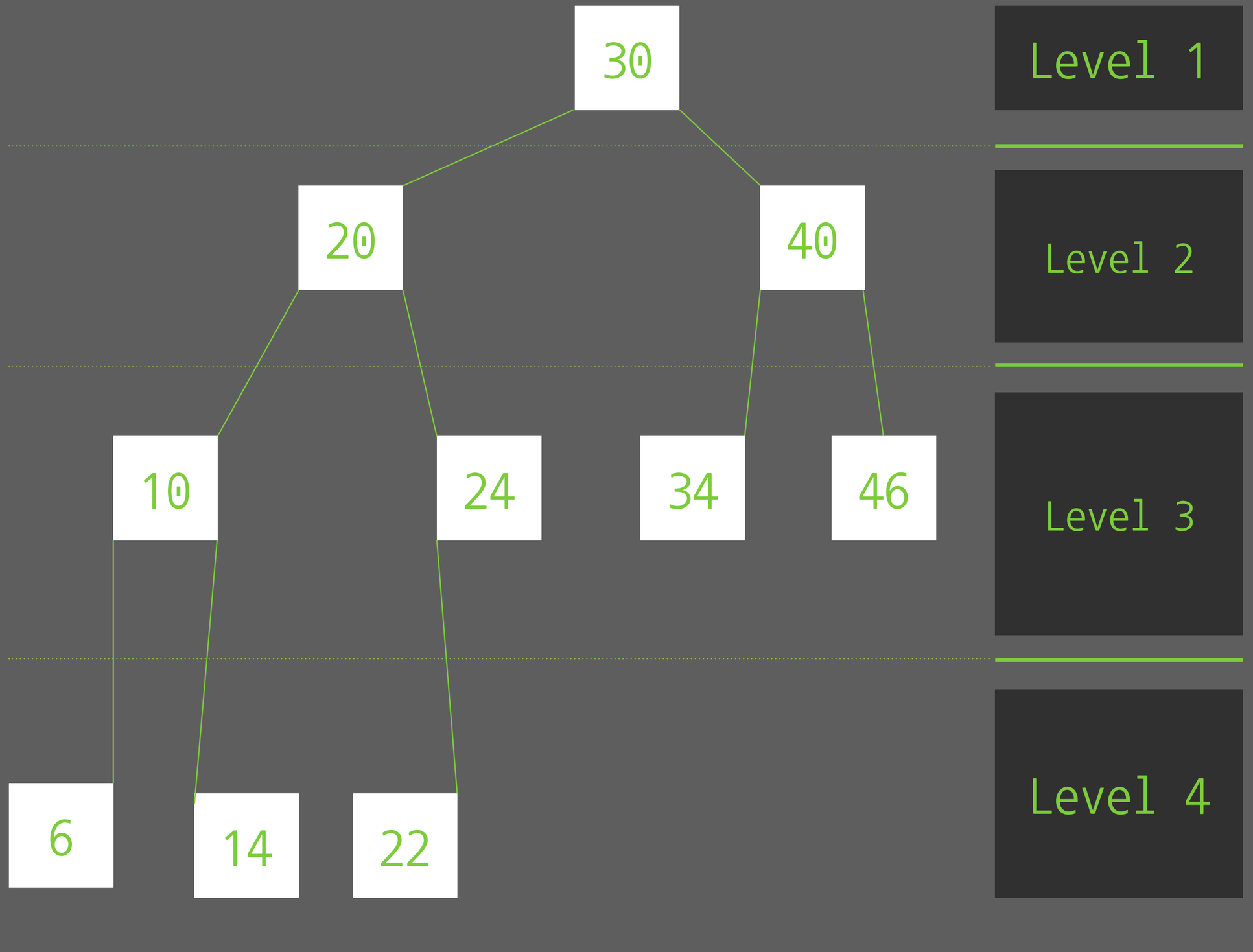
Binary Search Tree 의 특성

1. 유일한 키 값.
2. 루트 노드의 키 값 기준

왼쪽 서브트리 키 값
오른쪽 서브트리 키 값

*JDK
- Treeset, TreeMap(Red-Black Tree)

Tree



Hashing

해싱?

산술 연산을 이용한
검색 방식

Hashing

평균 시간 복잡도

=

자료가 N개 일때,

O(1)

*JDK
- HashMap
- HashSet

Hashing

검색 키
(KEY)

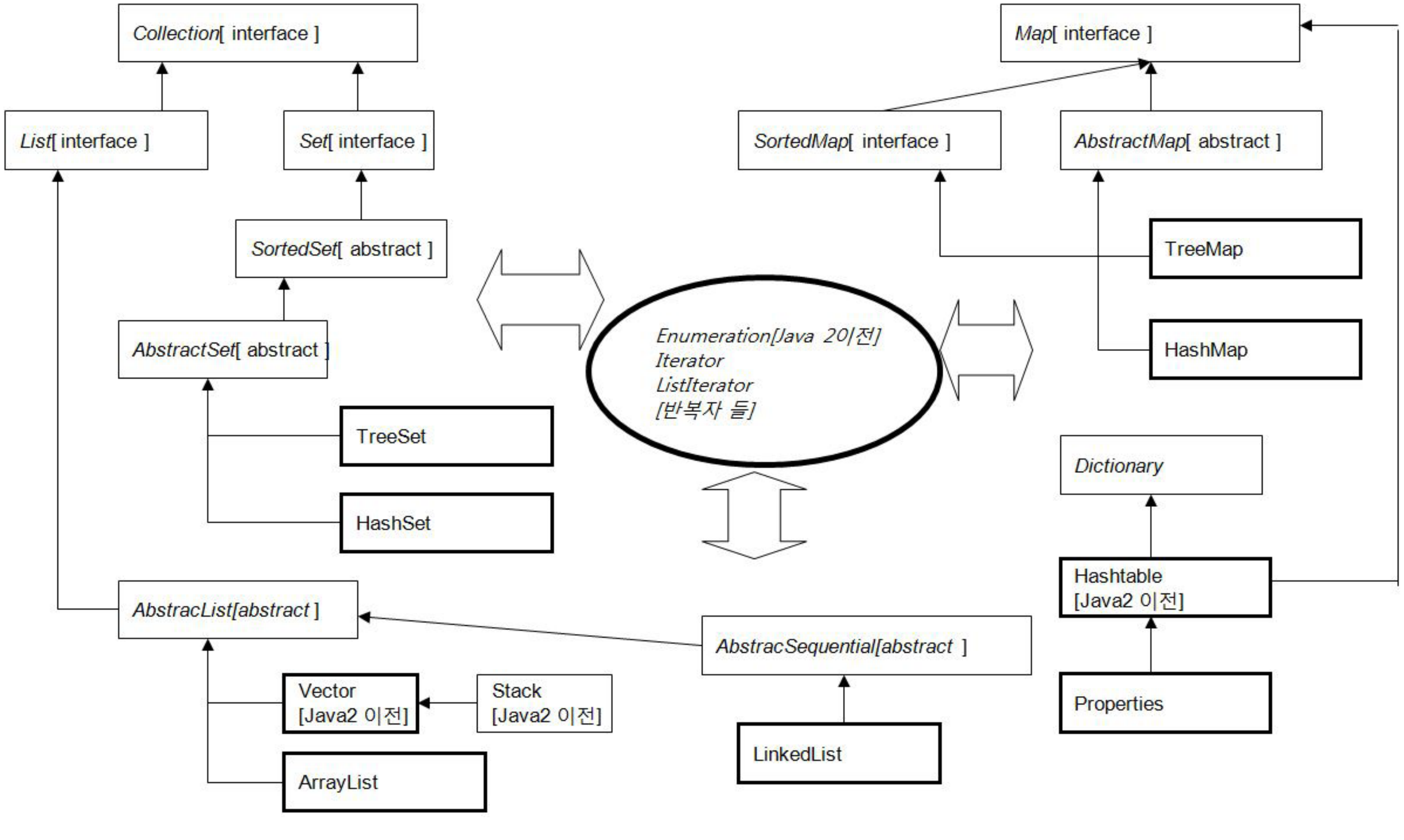
해싱 함수
(Hashing Function)

검색 자료의 주소
(Address)

Hash
Table

0	
1	
2	
3	
4	
5	
6	
⋮	
n	

Collection & Map


Collection[interface]
Map[interface]
List[interface]
Set[interface]
SortedMap[interface]
AbstractMap[abstract]
SortedSet[abstract]
TreeMap
AbstractSet[abstract]
HashMap
Enumeration[Java 20|전]
Iterator
ListIterator
[반복자 들]
TreeSet
Dictionary
HashSet
Hashtable
[Java2 이전]
AbstracList[abstract]
AbstracSequential[abstract]
Vector
[Java2 이전]
Stack
[Java2 이전]
Properties
LinkedList
ArrayList

Collection & Map

클래스에는
Collection 종류의 클래스와
Map 종류의 클래스가 있습니다.

결국,
자료구조적인
기능의 클래스

Collection & Map

인터페이스에도 Collection 종류와 Map 종류가 있습니다.

Collection과 Map은 인터페이스이기 때문에
메서드의 프로토 타입만 존재합니다.

Collection을 구현해서 사용하면
집합적인 저장공간으로서의 기능을 구현한
클래스가 됩니다.

반면에, Map을 구현해서 사용하면
검색적인 저장공간으로서의 기능을 구현한
클래스가 됩니다.

Collection

Collection 인터페이스의 특징

이것을 구현한 클래스들은
모두 집합적인
'저장공간'으로서의
기능을 가지게 됩니다.

* capacity(용기의 크기) vs size(내용물의 갯수)

Collection

Collection 인터페이스를 구현하고 있는 클래스

Stack,
Vector,
LinkedList,
TreeSet,
HashSet

Collection

Collection 인터페이스의 데이터 삽입/삭제를 위한 추상 메서드

```
boolean add(Object o);
boolean remove(Object o);
```

Collection

Collection 인터페이스의 데이터 확인을 위한 추상 메서드

```
boolean isEmpty();
boolean contains(Object o);
int size();
iterator();
```

java.
util.
List
?

java.util.List?

= 순서가 있는 Collection

/ List 인터페이스를 구현

/ 데이터를 중복해서 포함가능

ArrayList,
Vector,
LinkedList

java.
util.
List
?

ArrayList vs Vector

Vector는 기본적으로
동기화가 보장됩니다.

ArrayList가 Vector보다는
최적화 된 클래스 입니다.

java.
util.
List
?

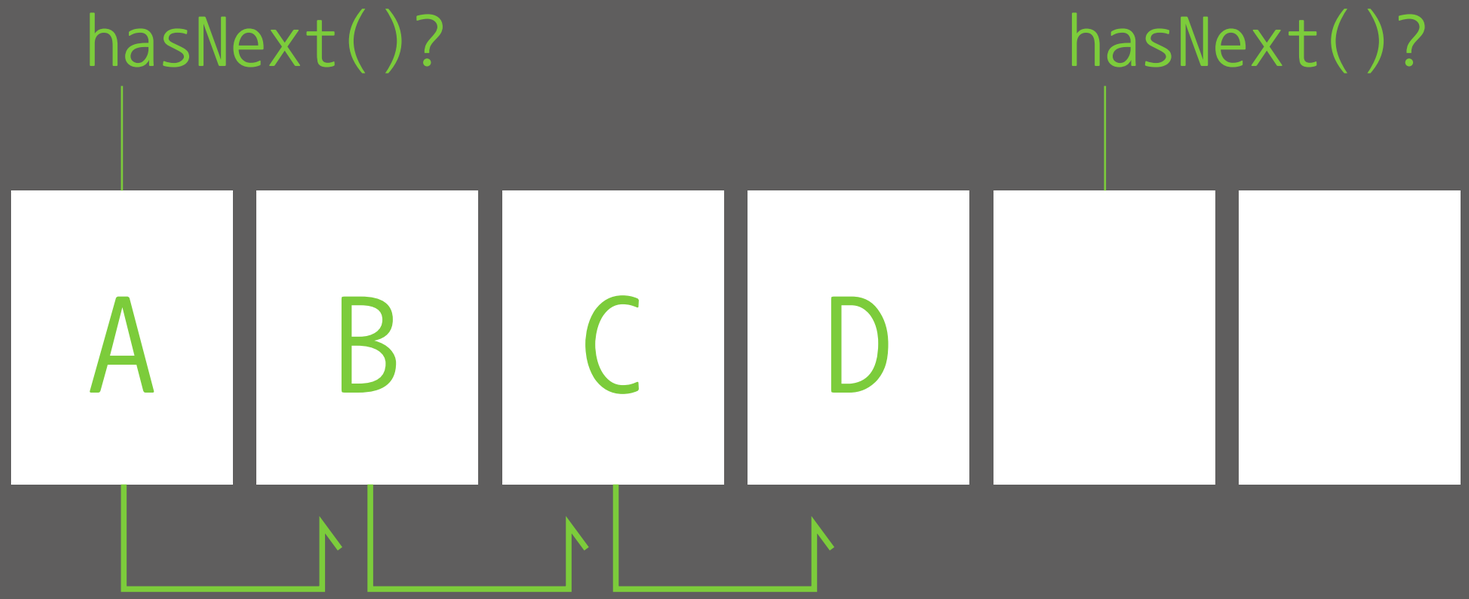
ArrayList vs LinkedList

ArrayList는 배열로
구현되어 있습니다.
그래서 검색속도가 빠릅니다.

LinkedList는 Data의 입출력이
자주 일어나는 경우에
사용합니다.

java.
util.
List
?

Iterator




```
boolean hasNext(); // 현재 위치에 element가 있는가?

Object next();     // 현재 위치에 있는 element를
                   return하고 다음으로
```

java.
util.
Set
?

java.util.Set?

Collection의
하위 인터페이스.

중복된 개체를 불허함.
(null도 불가)

java.
util.
Set
?

HashSet

Hashing 알고리즘으로 구현
빠르게 검색

O(1)

java.
util.
Set
?

TreeSet

/Red-Black Tree로 구현
/정렬된 결과로 출력됨
/내부적으로 TreeMap 사용
/정렬된 Collection을 제공
(Sorted Collection)
/항목을 Iteration하는 경우 자동적으로
정렬된 상태로 제공됨
/항목이 추가되거나 삭제되어도 항상
Tree가 유지됨.

O(logN)

java.
util.
Set
?

TreeSet

객체가 insert될 때
객체의 정렬된 상태를
유지해야 하므로
트리셋에 추가되는 항목은
Comparator 혹은
Comparable을 구현해야함.

java.
util.
Set
?

TreeSet

Comparable 인터페이스

```
public int compareTo(Object o)
```

Comparator 인터페이스

```
public int compare(Object o1, Object o2)
// TreeSet 생성자에 Comparator 객체를
인자로 넣어주어야함.
```

Map

Map인터페이스의 특징

Collection과 달리
Map은 검색적인 개념을
담고 있다.

Map

Map으로 구현된 클래스

Attributes,
HashMap,
Hashtable,
Properties,
TreeMap

Map

Map 인터페이스의 데이터 삽입 삭제를 위한 추상 메서드

```
Object put(Object key, Object value);
Object remove(Obejct key);
Object get(Obejct key);
```

Map

Map 인터페이스의 데이터 확인을 위한 추상 메서드

```
boolean isEmpty();
boolean containsKey(Object key);
boolean contatinsValue(Object value);
int size();
```

Hash Map 예제

Hashing 알고리즘으로 구현됨
key-value 쌍으로 저장
key는 중복 될 수 없다.

TreeMap
key-value로 저장
key 값으로 정렬 된다.

Callback Function

Callback Function?

시스템에 의해
‘호출’되는 함수

ex)

Wdiget의 EventHandler

Windows Programming의 MessageHandler

Callback Function

어떤 상황에서
Call이 되어야 하는지
등록(register)되어 있고,
그 상황이 발생했을 때
System에 의해 호출되는
함수.

*프로그래머가 하지 않습니다.

Callback Function

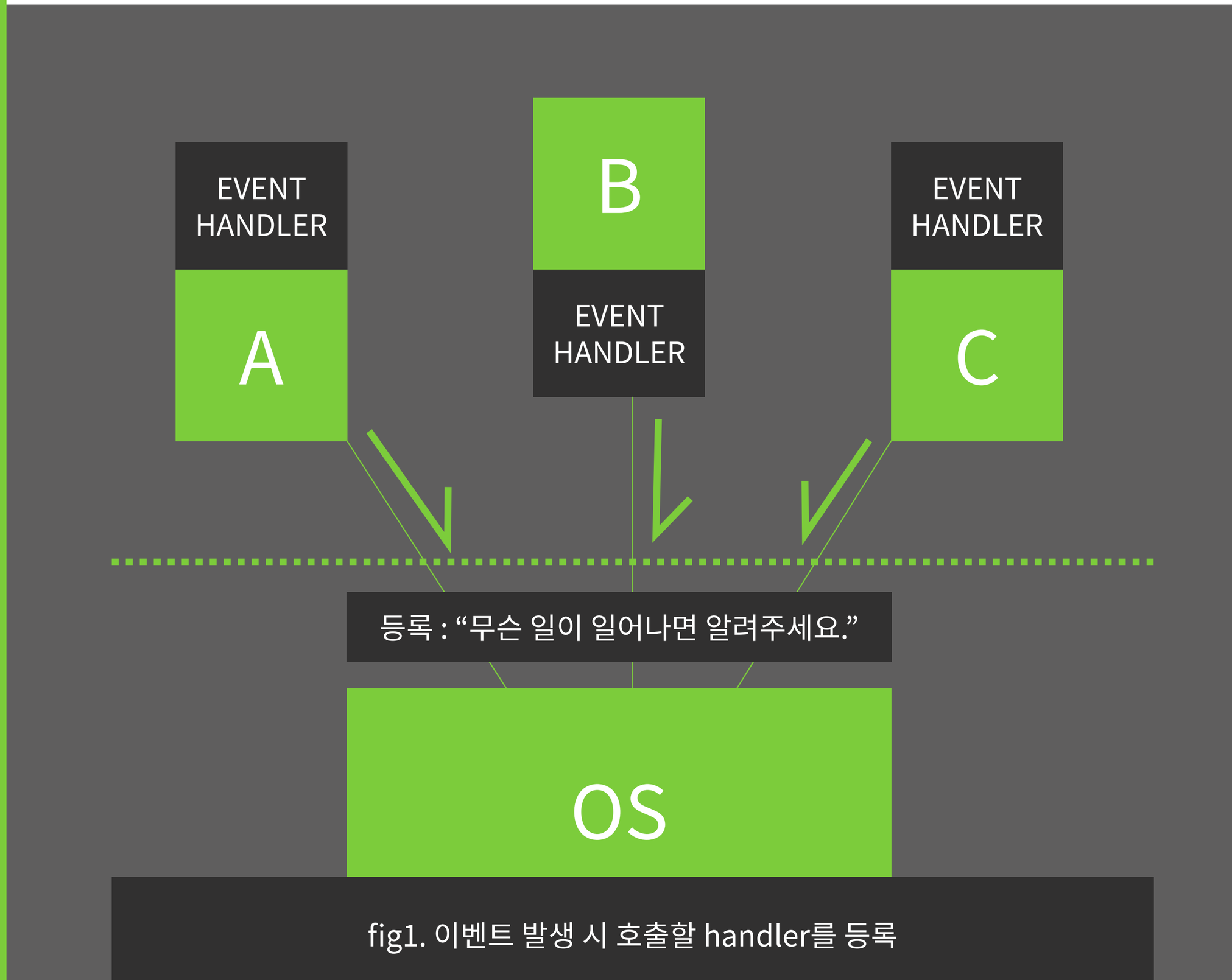


fig1. 이벤트 발생 시 호출할 handler를 등록

Callback Function

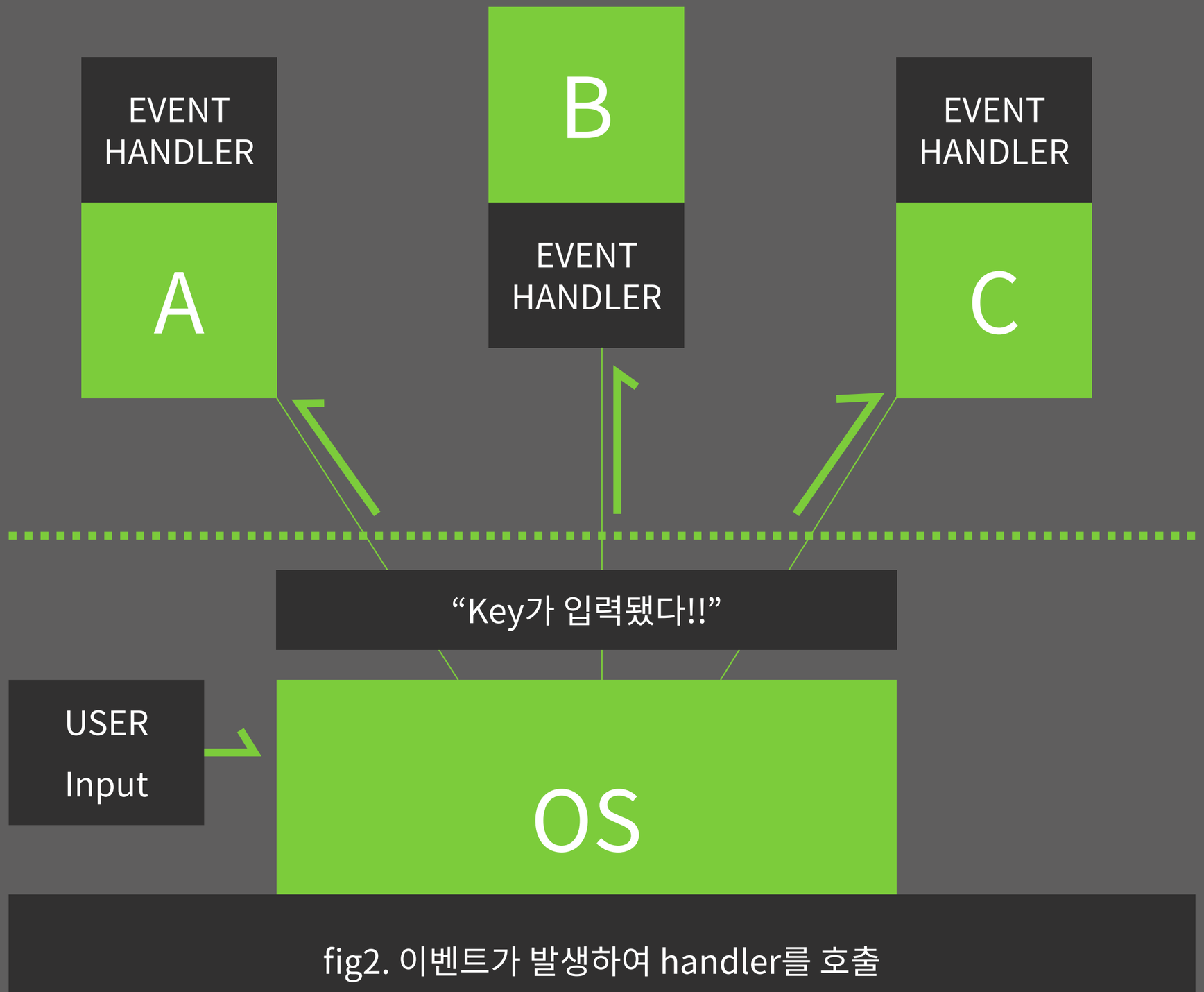


fig2. 이벤트가 발생하여 handler를 호출