



It's Your Life

with





Collection

영어

↔

한국어



collection




×

kə'lekSHən

수집

sujib

'collection'의 번역

Collection 이라 쓰고



자료 구조라 읽는다



그게 원데 10덕아



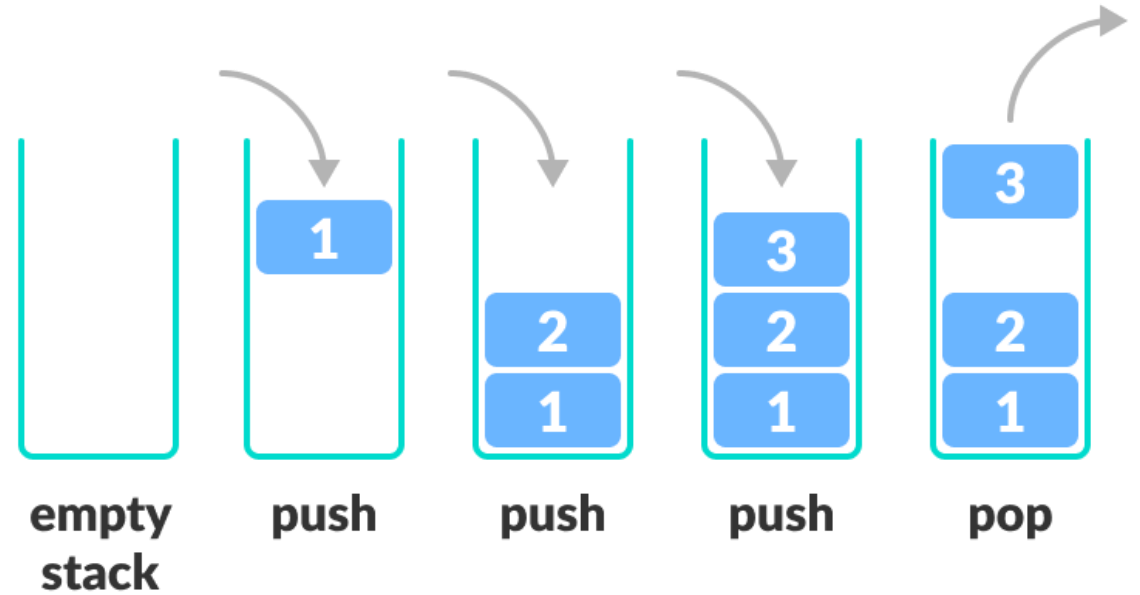
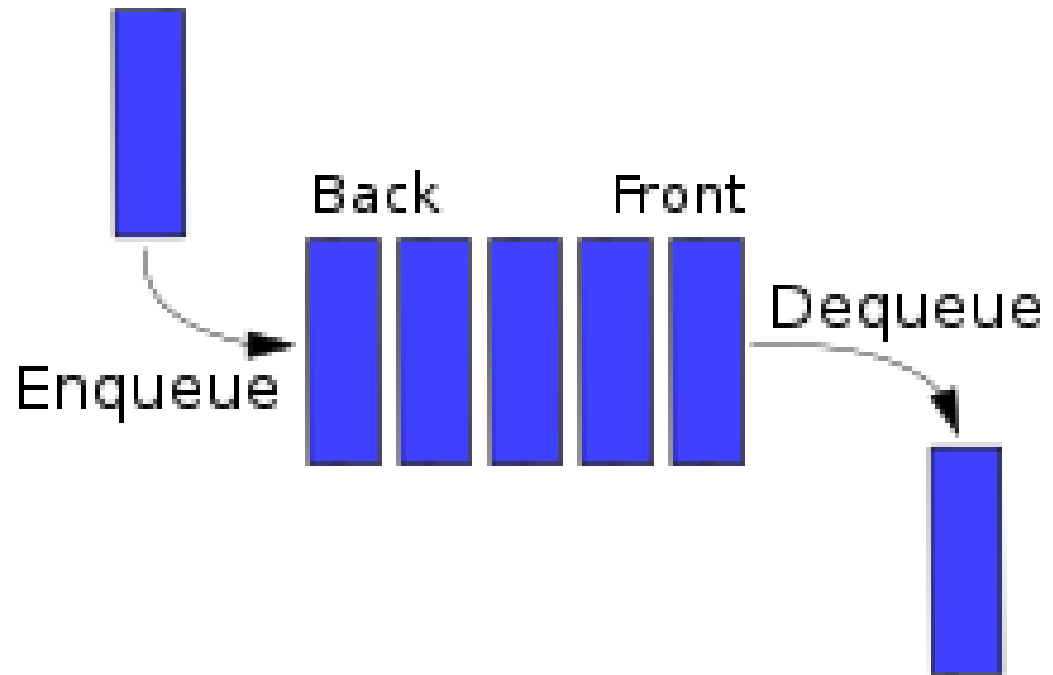
현실을 살아 제발 ㅋㅋㅋㅋ

여러분이 웹브라우저의 앞으로 가기 뒤로 가기
기능을 구현한다고 생각해 봅시다!





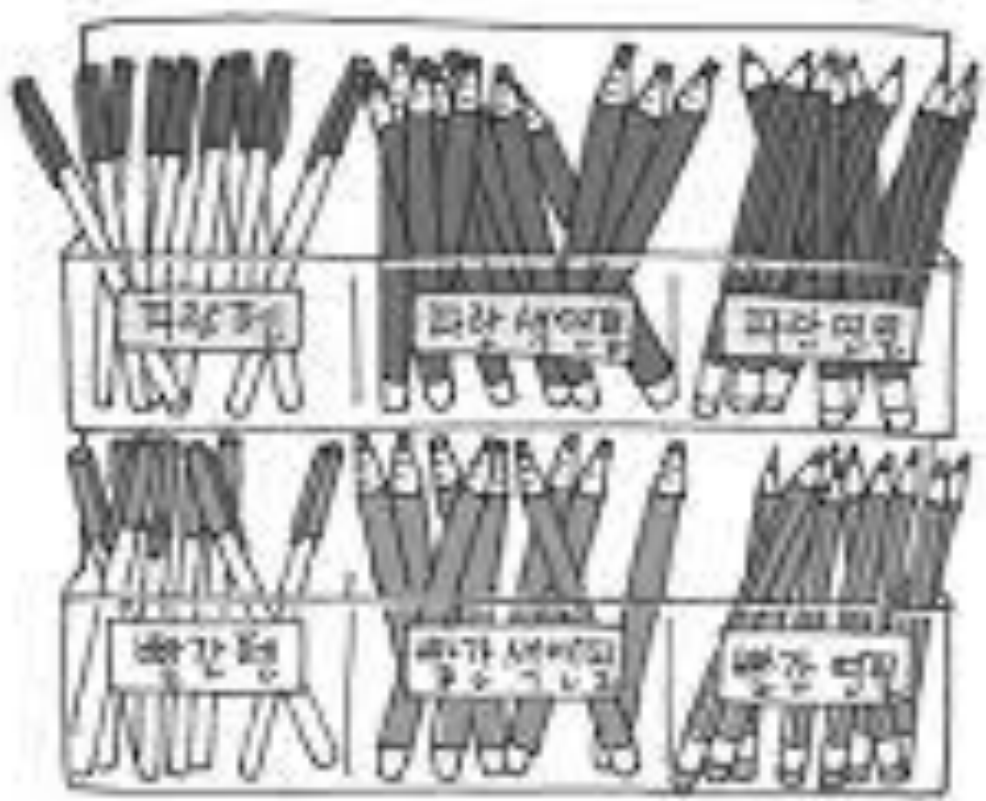
두 데이터 구조 중에서 어떤 방식이 더 유리 할까요!?



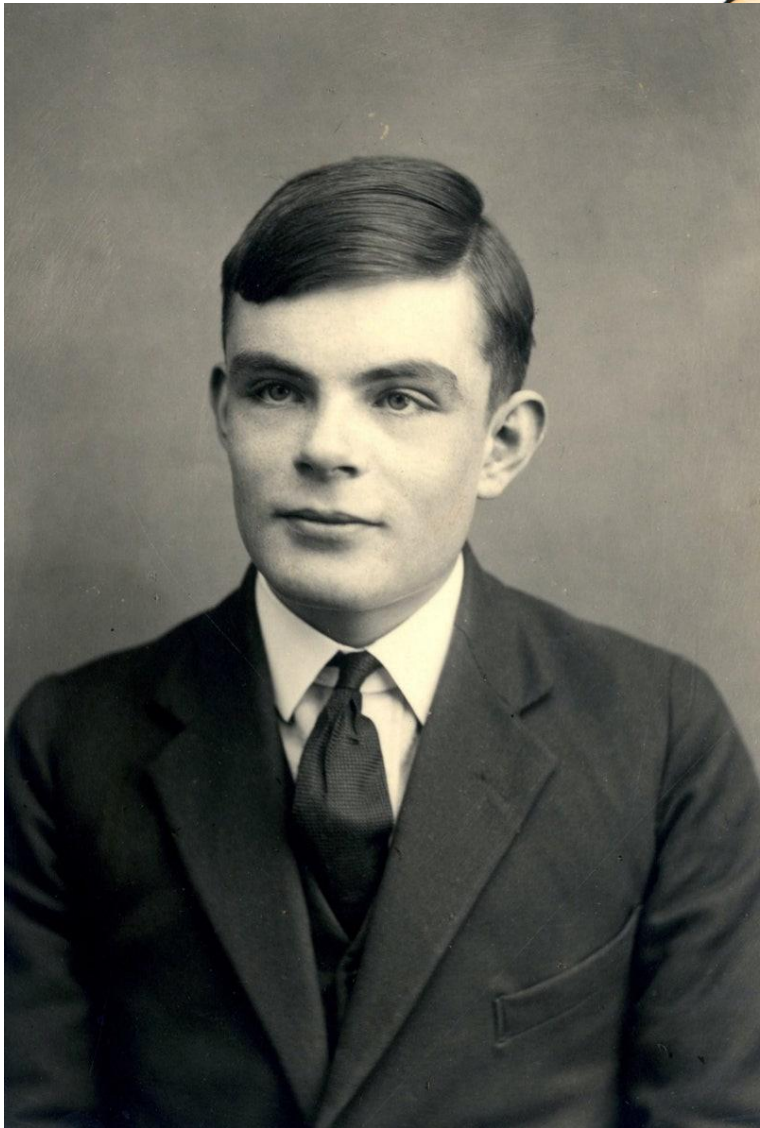




[나쁜 자료구조]



[좋은 자료구조]



진짜 사람 사는거 다 똑같다
백억부자도 어쩔 수 없음
쇼파 등받이로 사용하고 바닥에 앉는거 한국인 국룰임



오전 12:10 · 2023년 12월 28일



킹반인

다 똑같은 거 아님!?

현대 개발자

하지만 빨랐죠





Collection

3대장



Set

List

Queue



Set

Map

List

Stack

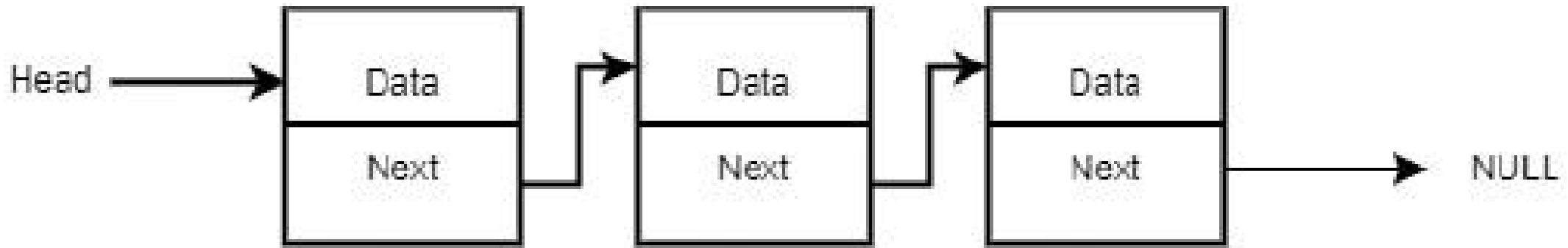
Queue



List



- 요소들이 순차적으로 저장 된 컬렉션

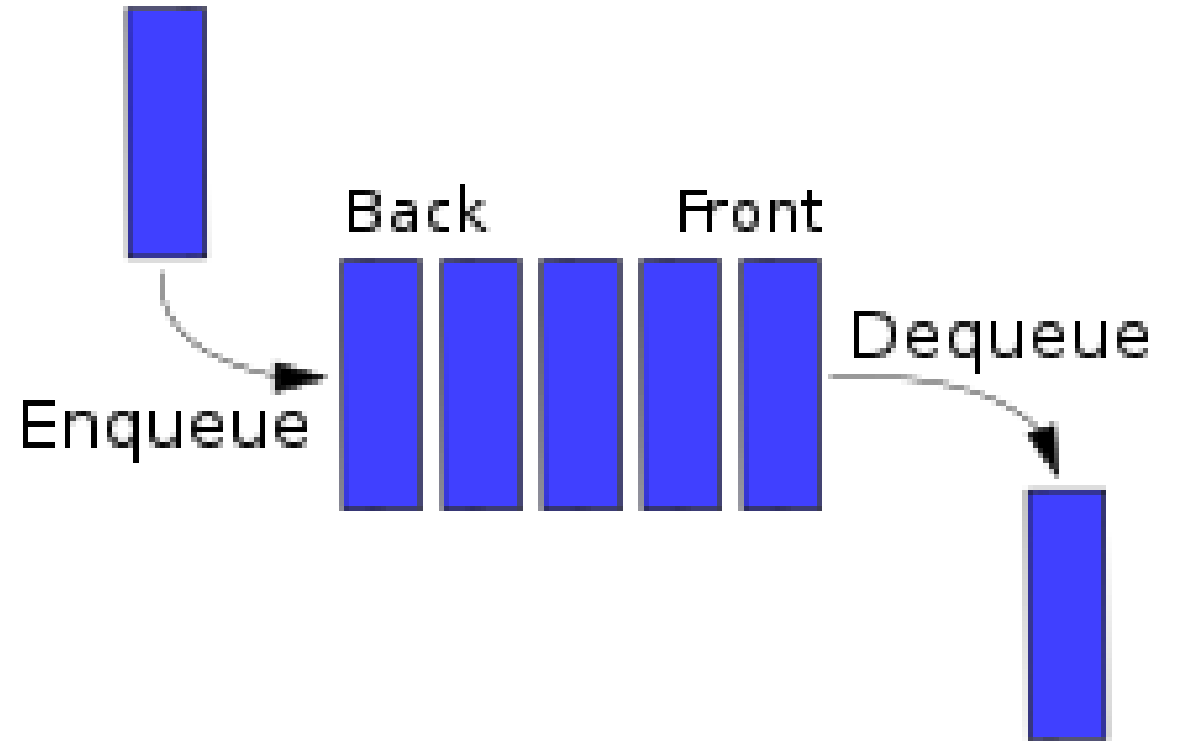


Single Linked List

Queue



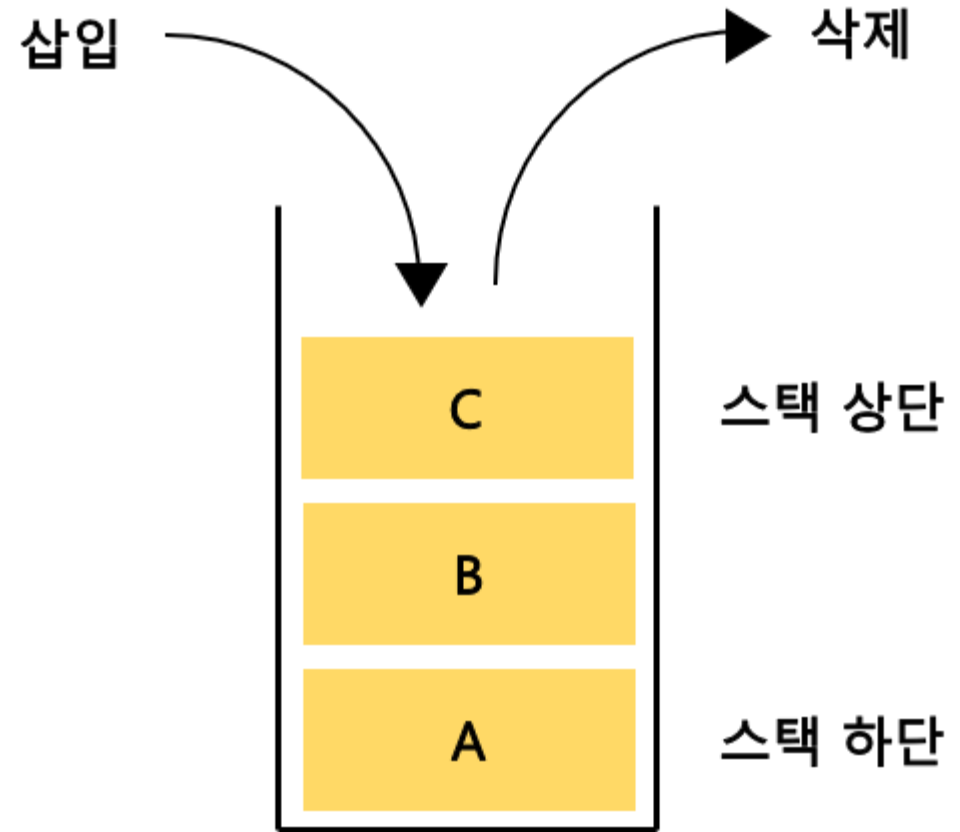
- 일렬로 정렬된 데이터
- 먼저 들어간 데이터가 먼저 나오는 특성(FIFO)을 가지며, 해당 특성이 유용한 경우에 자주 사용



Stack



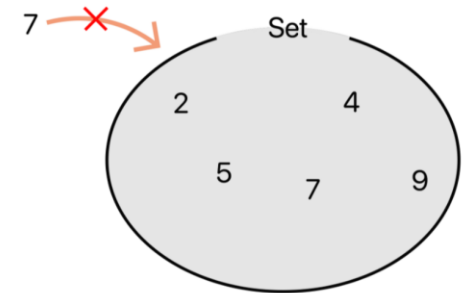
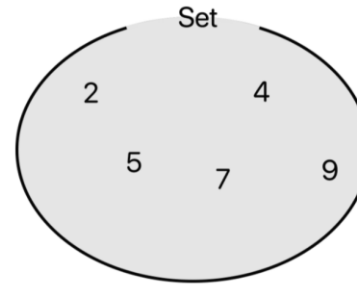
- 순차적으로 쌓이는 데이터
- 먼저 들어간 데이터가 가장 나중에 나오는 특성(FILO)을 가지며, 해당 특성이 유용한 경우에 자주 사용



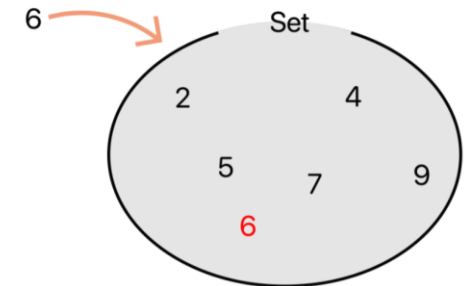
Set



- 중복을 허용하지 않는 요소들의 집합
- 데이터의 중복 체크 등에 자주 사용



7이 중복되므로 추가할 수 없음

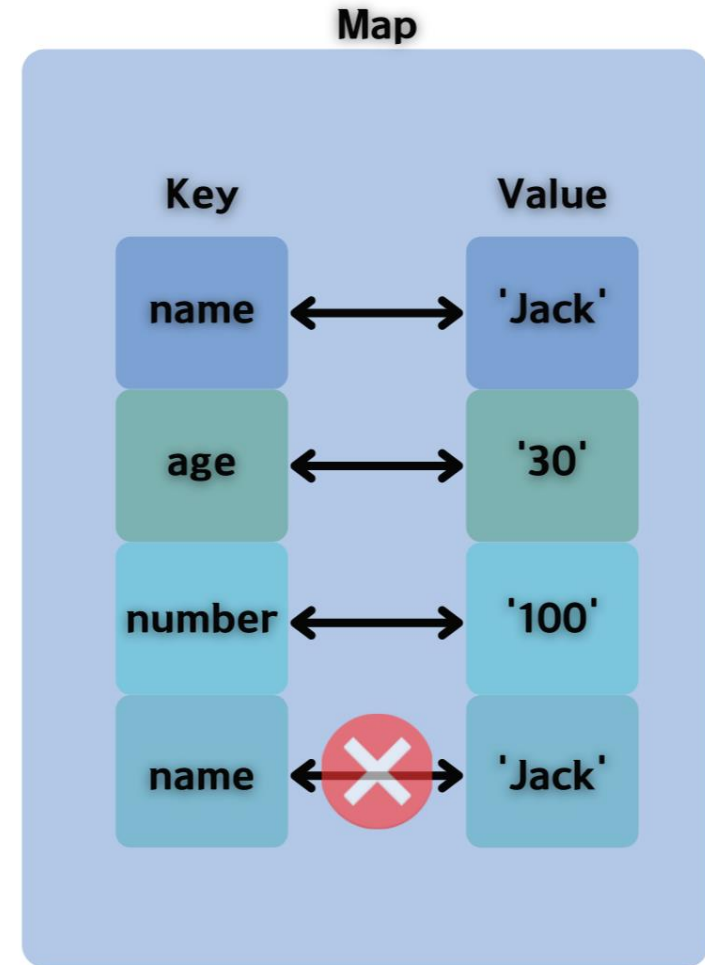


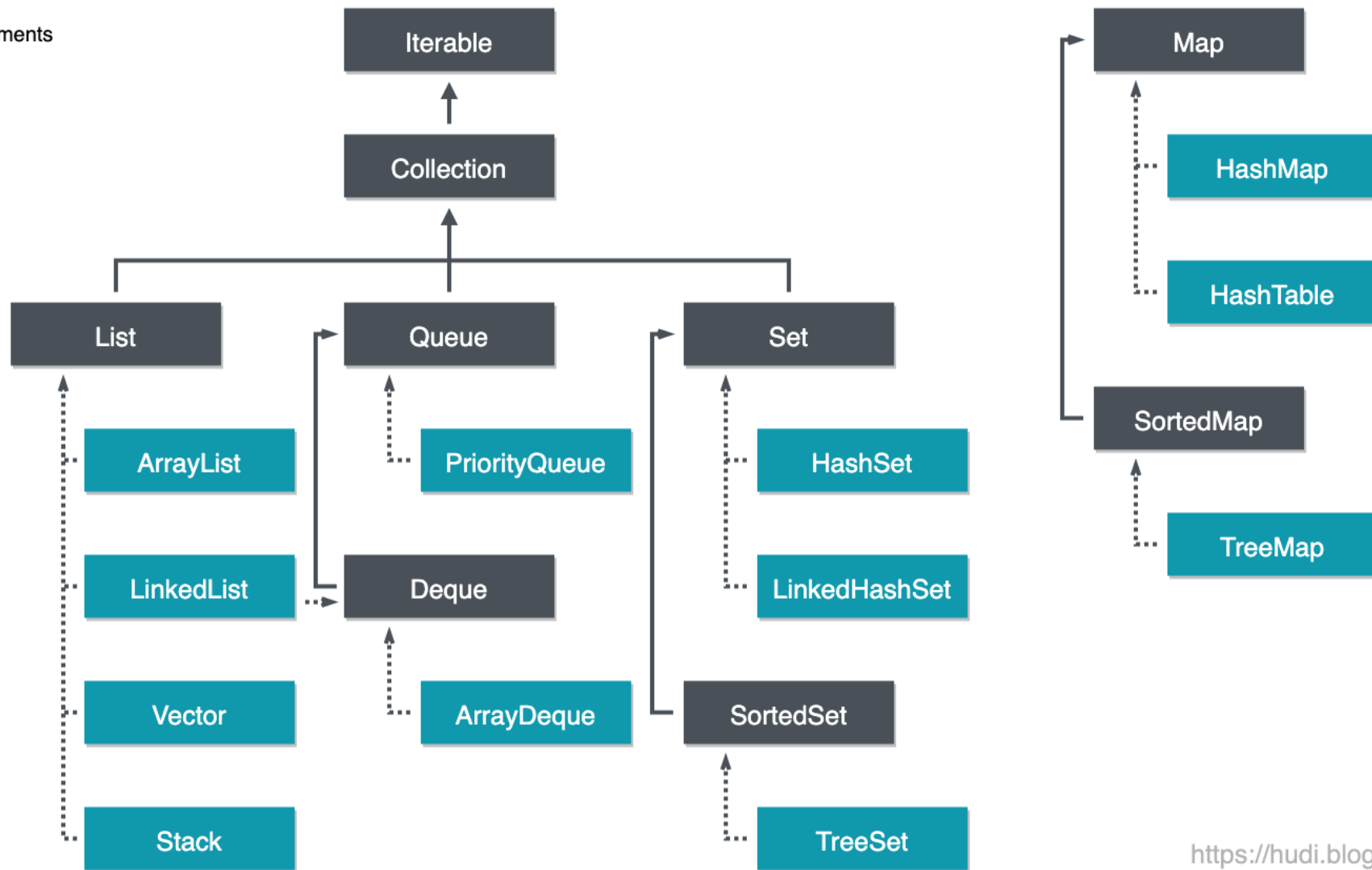
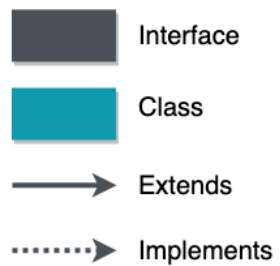
6은 중복되지 않으므로 추가 됨

Map

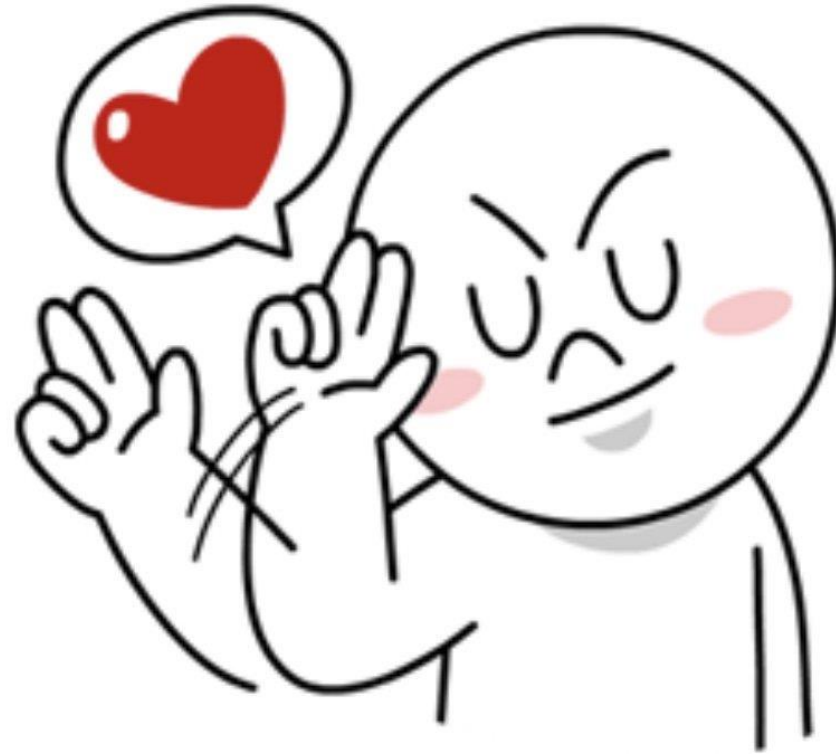


- key 와 value 의 쌍을 저장하는 데이터
- 빠른 검색 및 삽입, 삭제가 필요한 곳에서 주로 사용





지금까지 컬렉션에 대해서 알아보았습니다!





장난나랑 지금하냐?

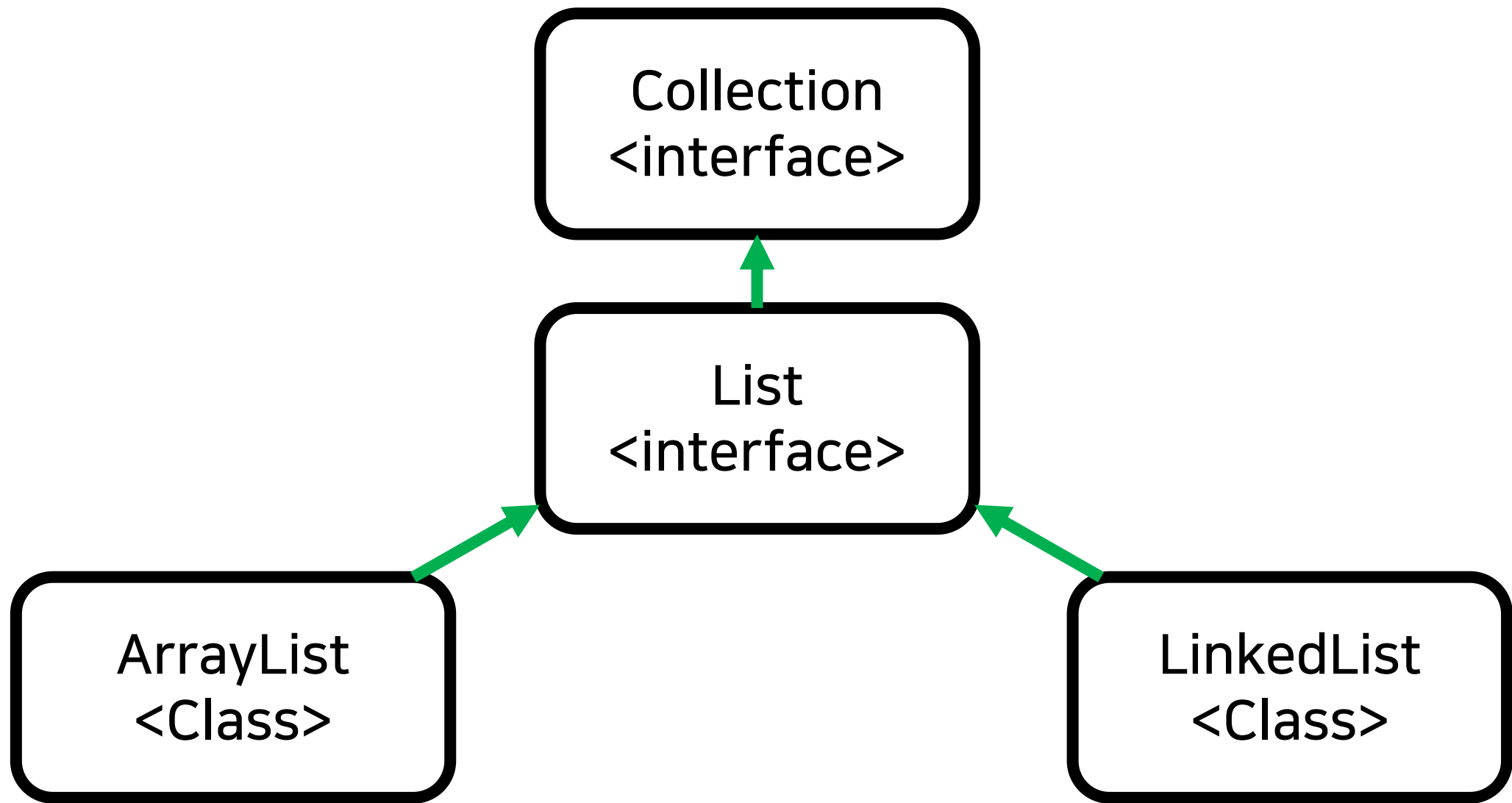




List



Single Linked List



List 인터페이스의 추상 메서드



- List 는 인터페이스 입니다!
- 즉, 해당 List 는 실제적 기능이 없으며 List 를 구현하는 구현체에게 List 이
기 위한 필수적인 메서드의 틀을 제공 합니다!

1. **add(E e):**

- 리스트의 끝에 지정된 요소를 추가합니다.
- 예시: ``list.add("element")``

2. **add(int index, E element):**

- 지정된 위치에 지정된 요소를 삽입합니다.
- 예시: ``list.add(1, "element")``

3. **boolean addAll(Collection<? extends E> c):**

- 지정된 컬렉션의 모든 요소를 리스트의 끝에 추가합니다.
- 예시: ``list.addAll(anotherList)``

4. **boolean addAll(int index, Collection<? extends E> c):**

- 지정된 컬렉션의 모든 요소를 리스트의 지정된 위치에 추가합니다.
- 예시: ``list.addAll(1, anotherList)``

5. **void clear():**

- 리스트의 모든 요소를 제거합니다.
- 예시: ``list.clear()``

6. **boolean contains(Object o):**

- 리스트에 지정된 요소가 포함되어 있는지 확인합니다.
- 예시: ``list.contains("element")``

7. **E get(int index):**

- 지정된 위치의 요소를 반환합니다.
- 예시: ``list.get(1)``

8. **int indexOf(Object o):**

- 리스트에서 지정된 요소의 첫 번째 발생 위치의 인덱스를 반환합니다.
- 예시: ``list.indexOf("element")``

9. **boolean isEmpty():**

- 리스트가 비어 있는지 확인합니다.
- 예시: ``list.isEmpty()``



14. **E remove(int index):**

- 지정된 위치의 요소를 제거하고, 제거된 요소를 반환합니다.
- 예시: ``list.remove(1)``

15. **boolean remove(Object o):**

- 리스트에서 지정된 요소의 첫 번째 발생을 제거합니다.
- 예시: ``list.remove("element")``

16. **boolean removeAll(Collection<?> c):**

- 리스트에서 지정된 컬렉션에 포함된 모든 요소를 제거합니다.
- 예시: ``list.removeAll(anotherList)``

19. **int size():**

- 리스트의 요소 수를 반환합니다.
- 예시: ``list.size()``

20. **List<E> subList(int fromIndex, int toIndex):**

- 리스트의 지정된 범위의 부분 리스트를 반환합니다.
- 예시: ``list.subList(1, 3)``

21. **Object[] toArray():**

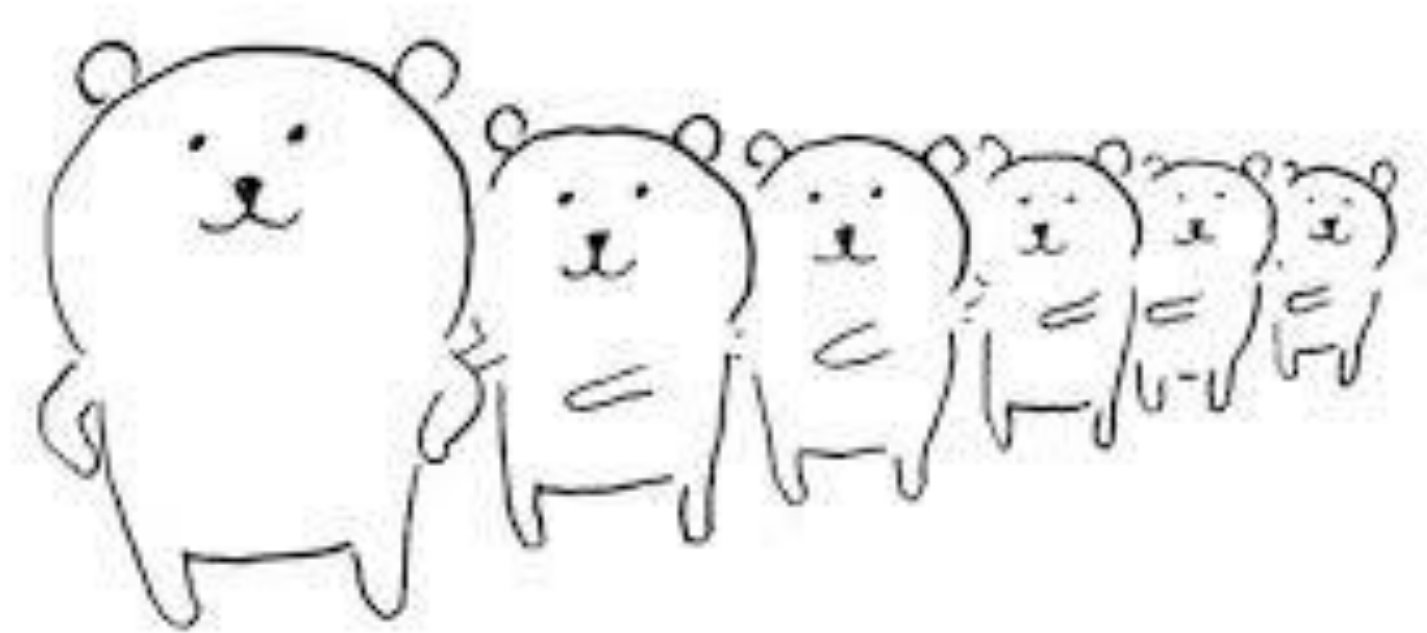
- 리스트의 모든 요소를 포함하는 배열을 반환합니다.
- 예시: ``Object[] array = list.toArray()``

22. **<T> T[] toArray(T[] a):**

- 리스트의 모든 요소를 지정된 배열에 저장하여 반환합니다.
- 예시: ``String[] array = list.toArray(new String[0])``



ArrayList





제25회 6.3 제주인의 날 기념
제10회 제주농아연 문화페스티벌 '울림'

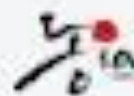
두번째 온라인퀴즈 3탄

몸으로 말해요

영상을 보고 퀴즈의 정답을 게시글 댓글에 남겨주세요!

추첨을 통해 기념품을 드립니다!

제시어는
속담입니다!



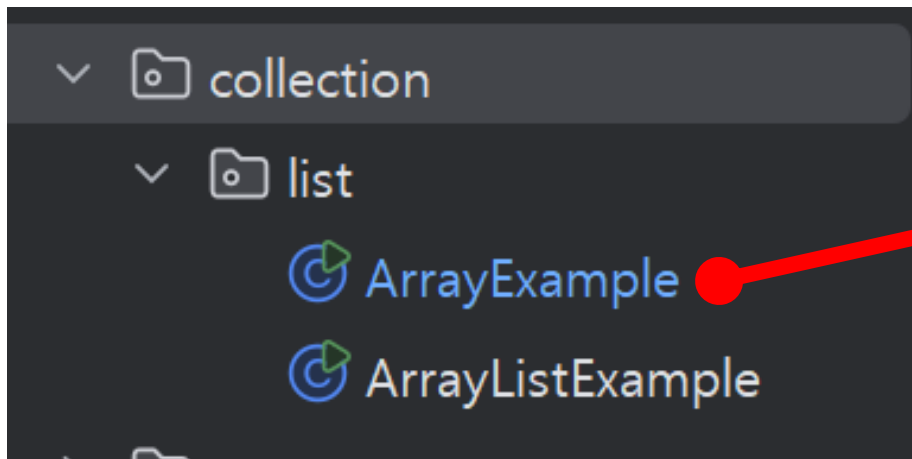




연산	시간 복잡도
<code>`get(index)`</code>	$O(1)$
<code>`set(index, element)`</code>	$O(1)$
<code>`add(element)`</code>	$O(1)$ 평균적으로
<code>`add(index, element)`</code>	$O(n)$
<code>`remove(index)`</code>	$O(n)$
<code>`remove(element)`</code>	$O(n)$
<code>`size()`</code>	$O(1)$
<code>`isEmpty()`</code>	$O(1)$

그래서 그거 왜 씹!?





collection 패키지 →
list 패키지 생성
+ ArrayExample 클래스 생성



```
public class ArrayExample {  👤 kdtTetz *  
    public static void main(String[] args) {  
        int[] arr = { 1, 2, 3, 4, 5 };  
  
        // 배열 제일 뒤에 6 을 추가해 봅시다!
```

배열 제일 뒤에 6을
추가해 봅시다!

자바 배열은 push 나 pop 같은 메서드를 지원 하나요!?

// 배열 제일 뒤에 6 을 추가해 봅시다!

```
int[] temp = new int[arr.length + 1];
```

```
for(int i = 0; i < arr.length; i++) {  
    temp[i] = arr[i];  
}
```

```
temp[temp.length - 1] = 6;  
arr = temp;
```

```
for (int i = 0; i < arr.length; i++) {  
    System.out.print(arr[i] + " ");  
}
```

기존 배열보다 크기가 1 큰
배열을 생성

기존 배열을 새 배열에 복사

새 배열의 마지막에 6 을 추가



// 배열 제일 뒤에 6 을 추가해 봅시다!

```
int[] temp = new int[arr.length + 1];
```

```
for(int i = 0; i < arr.length; i++) {  
    temp[i] = arr[i];  
}
```

```
temp[temp.length - 1] = 6;
```

```
arr = temp;
```

```
for (int i = 0; i < arr.length; i++) {  
    System.out.print(arr[i] + " ");  
}
```

기존 배열의 주소 정보를
가지고 있던 변수에
새로운 배열의 주소를 전달

배열 출력!





```
public class ArrayListExample {  👤 kdtTetz *  
    public static void main(String[] args) {  👤 kdtTetz *  
        ArrayList<Integer> list = new ArrayList<>();  
  
        list.add(1);  
        list.add(2);  
        list.add(3);  
        list.add(4);  
        list.add(5);  
  
        System.out.println("list = " + list);  
    }  
}
```

Integer 타입을 저장하는
ArrayList 컬렉션 선언

배열에 순서대로
1, 2, 3, 4, 5 를 삽입

배열 출력

```
list = [1, 2, 3, 4, 5]
```



```
System.out.println("list = " + list);
```

```
// 여기서 마지막에 6을 추가하려면!?
```

ArrayList 제일 뒤에
6을 추가해 봅시다!

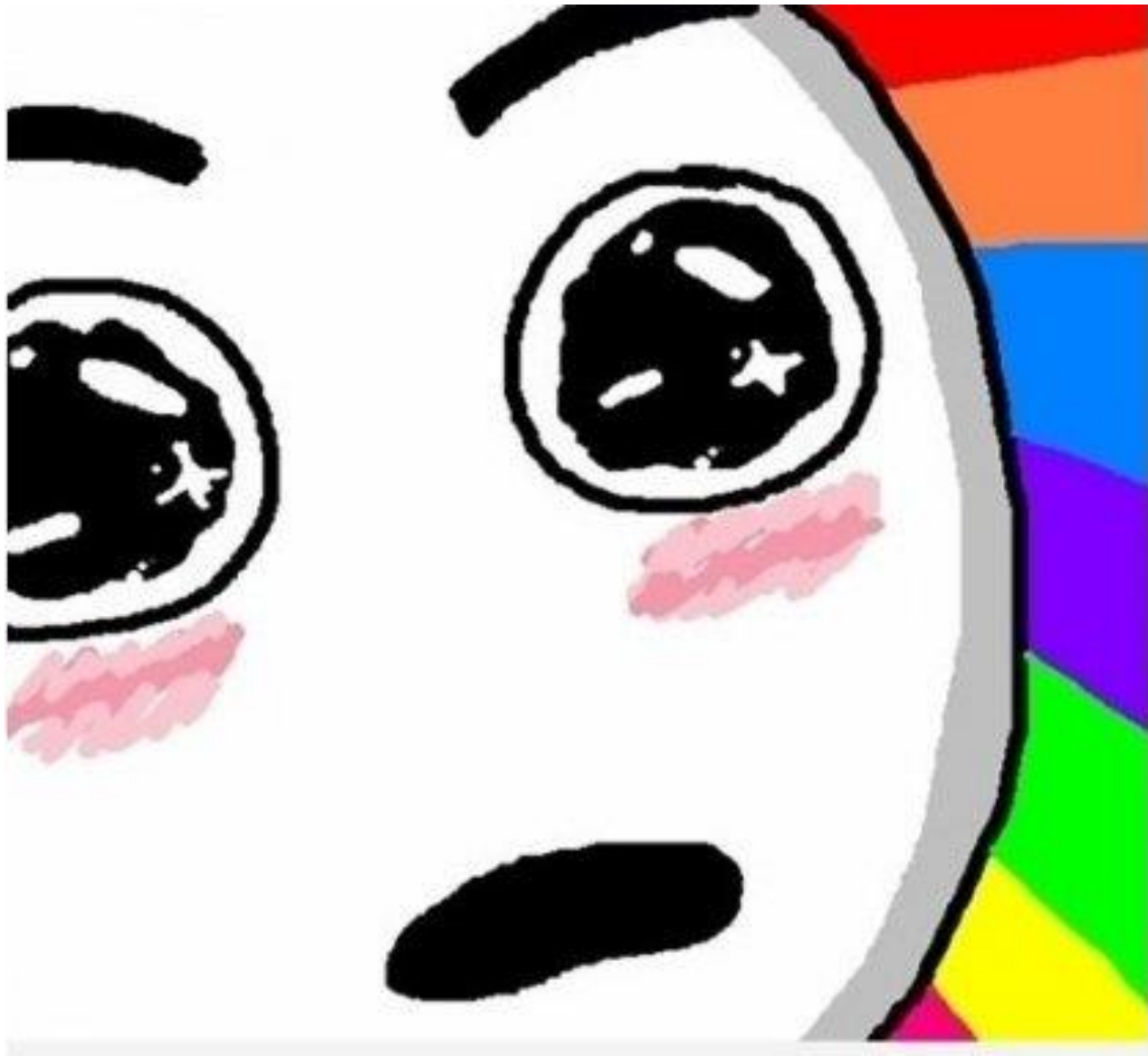
```
// 여기서 마지막에 6을 추가하려면!?
```

```
list.add(6);
```

```
System.out.println("list = " + list);
```

아까 배열 사용과는 달리
편하게 추가 가능!

```
list = [1, 2, 3, 4, 5, 6]
```





LinkedList





연산	시간 복잡도
<code>~get(index)~</code>	$O(n)$
<code>~set(index, element)~</code>	$O(n)$
<code>~add(element)~</code>	$O(1)$
<code>~add(index, element)~</code>	$O(n)$
<code>~remove(index)~</code>	$O(n)$
<code>~remove(element)~</code>	$O(n)$
<code>~size()~</code>	$O(1)$
<code>~isEmpty()~</code>	$O(1)$
<code>~contains(element)~</code>	$O(n)$



그래서 그거

어떻게 씹?

```
public class LinkedListExample { new *  
    public static void main(String[] args) { new *  
        LinkedList<Integer> linkedList = new LinkedList<>();  
  
        linkedList.add(1);  
        linkedList.add(2);  
        linkedList.add(3);  
        linkedList.add(4);  
        linkedList.add(5);  
  
        // 처음과 마지막에 데이터 추가  
        linkedList.addFirst(e: 0);  
        linkedList.addLast(e: 6);  
  
        // 특정 인덱스에 데이터 추가  
        linkedList.add(index: 3, element: 22);
```

자바에서 제공하는
LinkedList 사용

데이터 추가

처음과 마지막에 데이터 추가
시간은 오래 걸리나요!?

중간에 데이터 추가
시간은!?

```
// LinkedList 출력
System.out.println("LinkedList: " + linkedList);

// 데이터 검색
int firstElement = linkedList.getFirst(); // 첫 번째 데이터
int lastElement = linkedList.getLast(); // 마지막 데이터
int elementAtIndex = linkedList.get(3); // 인덱스 3의 데이터

System.out.println("First element: " + firstElement);
System.out.println("Last element: " + lastElement);
System.out.println("Element at index 3: " + elementAtIndex);
```

리스트 출력!

이중에서 시간이 오래 걸리는
작업은!?

```
// 데이터 삭제
LinkedList.removeFirst(); // 첫 번째 데이터 삭제
LinkedList.removeLast(); // 마지막 데이터 삭제
LinkedList.remove(index: 2); // 인덱스 2의 데이터 삭제

// LinkedList 출력
System.out.println("LinkedList after removals: " + linkedList);

// 데이터 포함 여부 확인
boolean contains30 = linkedList.contains(30);
System.out.println("Contains 30: " + contains30);
```

이중에서 시간이 오래 걸리는
작업은!?



List 는

배열 대신 씁니다!

조회가 빈번하면 ArrayList



삭제 추가가 빈번하면 LinkedList



Stack





그래서 그거

왜 씹?

순서를 반대로 만들 때, 혹은 짝을 맞추는 등등에 유용합니다!



<https://school.programmers.co.kr/learn/courses/30/lessons/12909>

```
public class StackReverse { new *  
    public static void main(String[] args) { new *  
        Stack<Character> stack = new Stack<>();  
        String input = "난장이장난";  
  
        // 문자열의 각 문자를 스택에 푸시  
        for (char ch : input.toCharArray()) {  
            stack.push(ch);  
        }  
    }  
}
```

스택 선언

뒤집을 문자열 선언

문자열을 배열로 변경하여
하나하나의 값을 스택에 넣기

```
// 스택에서 문자를 하나씩 팝하여 배열에 담기
char[] reversedArray = new char[input.length()];
int i = 0;
while (!stack.isEmpty()) {
    reversedArray[i++] = stack.pop();
}

// 배열을 문자열로 변환하여 반환
String reversed = new String(reversedArray);

System.out.println("Original String: " + input);
System.out.println("Reversed String: " + reversed);
```

스택이 완전히 빌 때까지
하나씩 값을 빼서 배열에 넣기

배열을 문자열로 변환

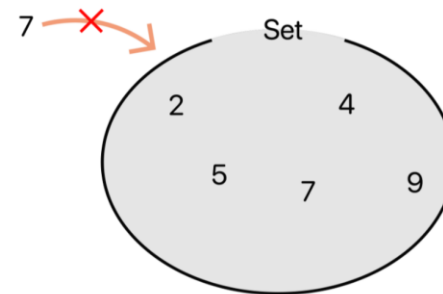
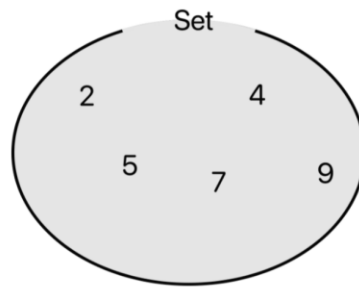


Queue

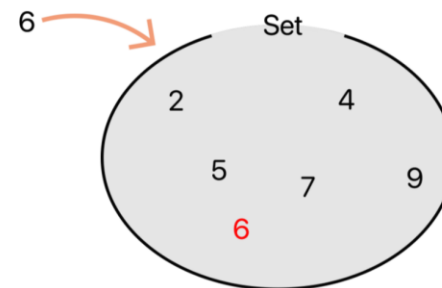




Set



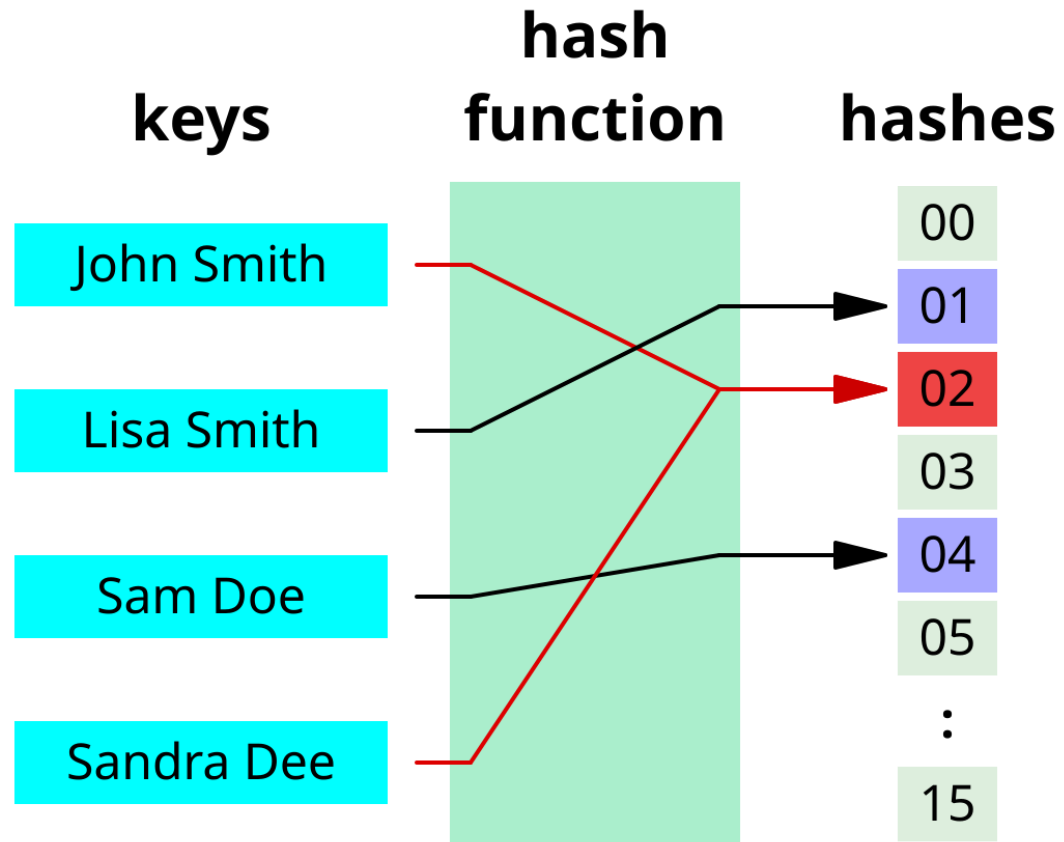
7이 중복되므로 추가할 수 없음



6은 중복되지 않으므로 추가 됨



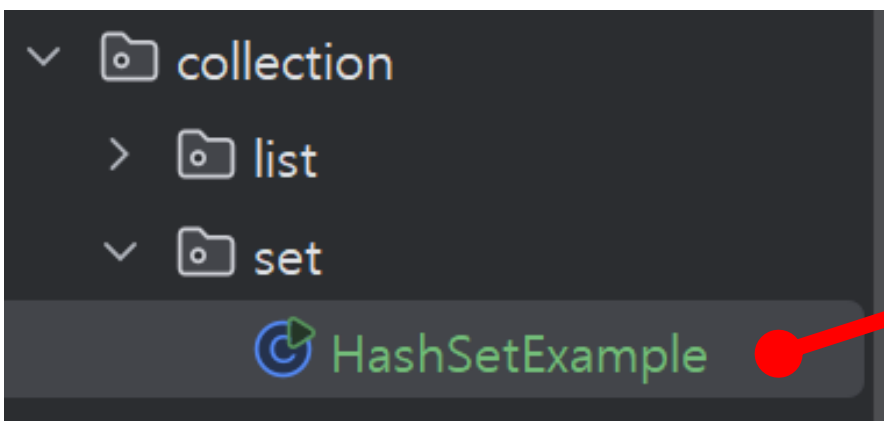
HashSet





그래서 그거

어떻게 씹?



HashSet 확인을 위한
set 패키지와 클래스 생성



```
public class HashSetExample { new *  
    public static void main(String[] args) { new *  
        // HashSet 선언  
        Set<Integer> set = new HashSet<>();  
  
        // 데이터 추가  
        set.add(10);  
        set.add(20);  
        set.add(30);  
        set.add(40);  
        set.add(50);  
  
        // 중복된 데이터 추가 시도  
        boolean isAdded = set.add(30); // 중복 요소 추가 시도  
        System.out.println("30 추가 시도 결과: " + isAdded);  
  
        // Set 출력  
        System.out.println("Set: " + set);  
    }  
}
```

정수 타입의 HashSet 생성

Set 의 특성으로 중복은 거절!

30 추가 시도 결과: false

Set: [50, 20, 40, 10, 30]



// 데이터 검색

```
boolean contains20 = set.contains(20);  
boolean contains60 = set.contains(60);  
System.out.println("Set에 20이 있는가?: " + contains20);  
System.out.println("Set에 60이 있는가?: " + contains60);
```

검색은 빠른 작업인가요?

// 데이터 삭제

```
boolean isRemoved = set.remove(20);  
System.out.println("20 삭제 시도 결과: " + isRemoved);  
System.out.println("Set after removal: " + set);
```

삭제는 빠른 작업인가요?

// Set의 크기 확인

```
int size = set.size();  
System.out.println("Set 크기: " + size);
```

Set에 20이 있는가?: true

Set에 60이 있는가?: false

20 삭제 시도 결과: true

Set after removal: [50, 40, 10, 30]

Set 크기: 4



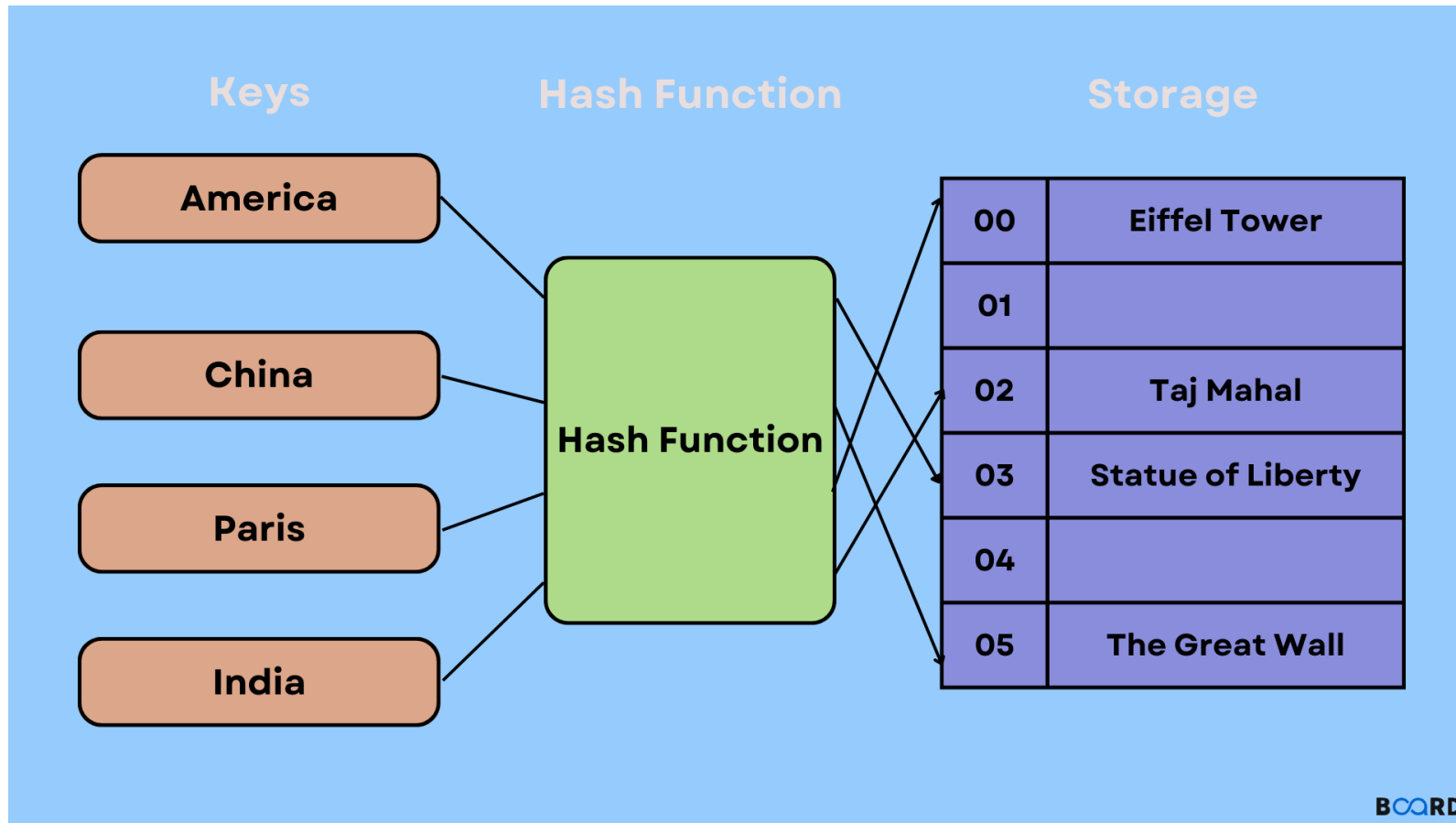


HashSet 은

중복 제거, 유일한 데이터 저장,
빠른 검색이 필요한 경우에 씁니다



HashMap



✓ collection

> list

✓ map

🎯 HashMapExample

HashMap 확인을 위한
map 패키지와 클래스 생성



```
public class HashMapExample { new *  
    public static void main(String[] args) { new *  
        // HashMap 선언  
        Map<String, Integer> hashMap = new HashMap<>();  
  
        // 데이터 추가 (put)  
        hashMap.put("사과", 10000);  
        hashMap.put("바나나", 2000);  
        hashMap.put("토마토", 500);  
        hashMap.put("수박", 20000);  
  
        // HashMap 출력  
        System.out.println("HashMap: " + hashMap);  
    }  
}
```

HashMap 컬렉션 생성

과일(Key)과
과일의 가격(Value)를
저장


```
// 특정 키의 값 가져오기 (get)
int appleCount = hashMap.get("사과");
System.out.println("사과 가격 : " + appleCount);
```

```
// 키가 존재하는지 확인 (containsKey)
boolean hasBanana = hashMap.containsKey("바나나");
System.out.println("바나나 가격 : " + hasBanana);
```

```
// 값이 존재하는지 확인 (containsValue)
boolean hasValue20000 = hashMap.containsValue(20000);
System.out.println("Contains value 20000: " + hasValue20000);
```

사과 가격 : 10000

바나나 가격 : true

Contains value 20000: true



```
// 데이터 삭제 (remove)
int removedValue = hashMap.remove(key: "수박");
System.out.println("Removed value: " + removedValue);
System.out.println("HashMap after removal: " + hashMap);

// HashMap의 크기 확인 (size)
int size = hashMap.size();
System.out.println("HashMap size: " + size);
```

```
Removed value: 20000
HashMap after removal: {토마토=500, 사과=10000, 바나나=2000}
HashMap size: 3
```



HashMap 은

키와 데이터를 쌍으로 저장하는 경우,
특정 데이터의 빈도를 세는 경우 등에
쓰면 좋습니다!