응용 제2장 : 컬렉션

데이터 구조에 따른 대표적인 컬렉션 (자료구조)

1) List : 순서 대로 쌓여있는 구조 (아이템의 중복 허용)
2) Map : 키(key)와 값(value)의 쌍으로 저장 (키의 중복 불가)
3) Set : 순서가 없는 집합 (중복 불가)

배열과 ArrayList 문법 비교


```java
// 배열 확보
String[] names = new String[3];

// 3인 추가
names[0] = "홍길동";
names[1] = "한석봉";
names[2] = "신사임당";
```

```java
System.out.println(names[1]);
```

```java
// ArrayList 확보
ArrayList<String> names = new ArrayList<String>();

// 3인 추가
names.add("홍길동");
names.add("한석봉");
names.add("신사임당");

System.out.println(names.get(1));
```

크기를 정해두지 않고 요소를 추가할 때 마다 크기가 커짐

컬렉션에서 기본형 취급 불가

컬렉션은 기본형(int, double, boolean 등)을 취급할 수 없다

```
ArrayList<int>
```

```
ArrayList<Integer> names = new ArrayList<Integer>();
names.add(10);
names.add(20);
names.add(30);
```

ArrayList 조작법

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

배열과 ArrayList 탐색 방법 비교


```
String[] names = new String[3];

for (int i = 0; i < names.length; i++) {
  System.out.println(names[i]);
}
```

```
ArrayList<String> names = new ArrayList<String>();

for (int i = 0; i < names.size(); i++) {
  System.out.println(names.get(i));
}
```

배열과 ArrayList 탐색 방법 비교 2

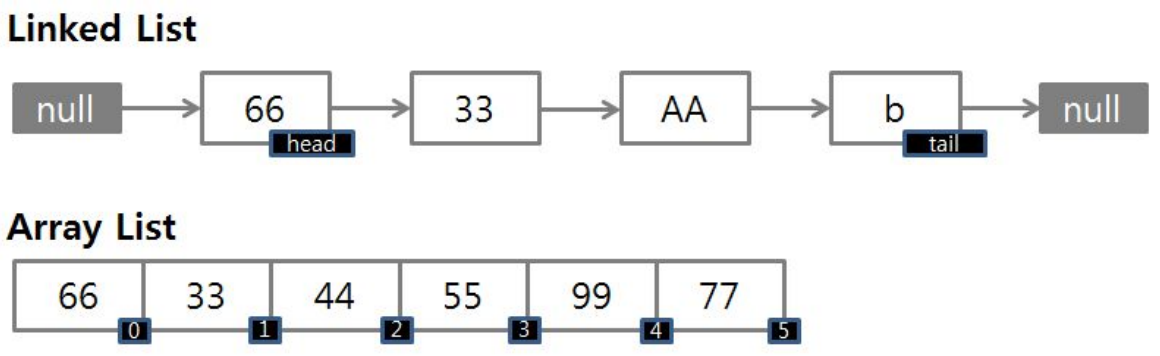

```
String[] names = new String[3];

for (String name: names) {
  System.out.println(name);
}
```

```
ArrayList<String> names = new ArrayList<String>();

for (String name: names) {
  System.out.println(name);
}
```

리스트의 요소를 하나씩 가리키는 객체

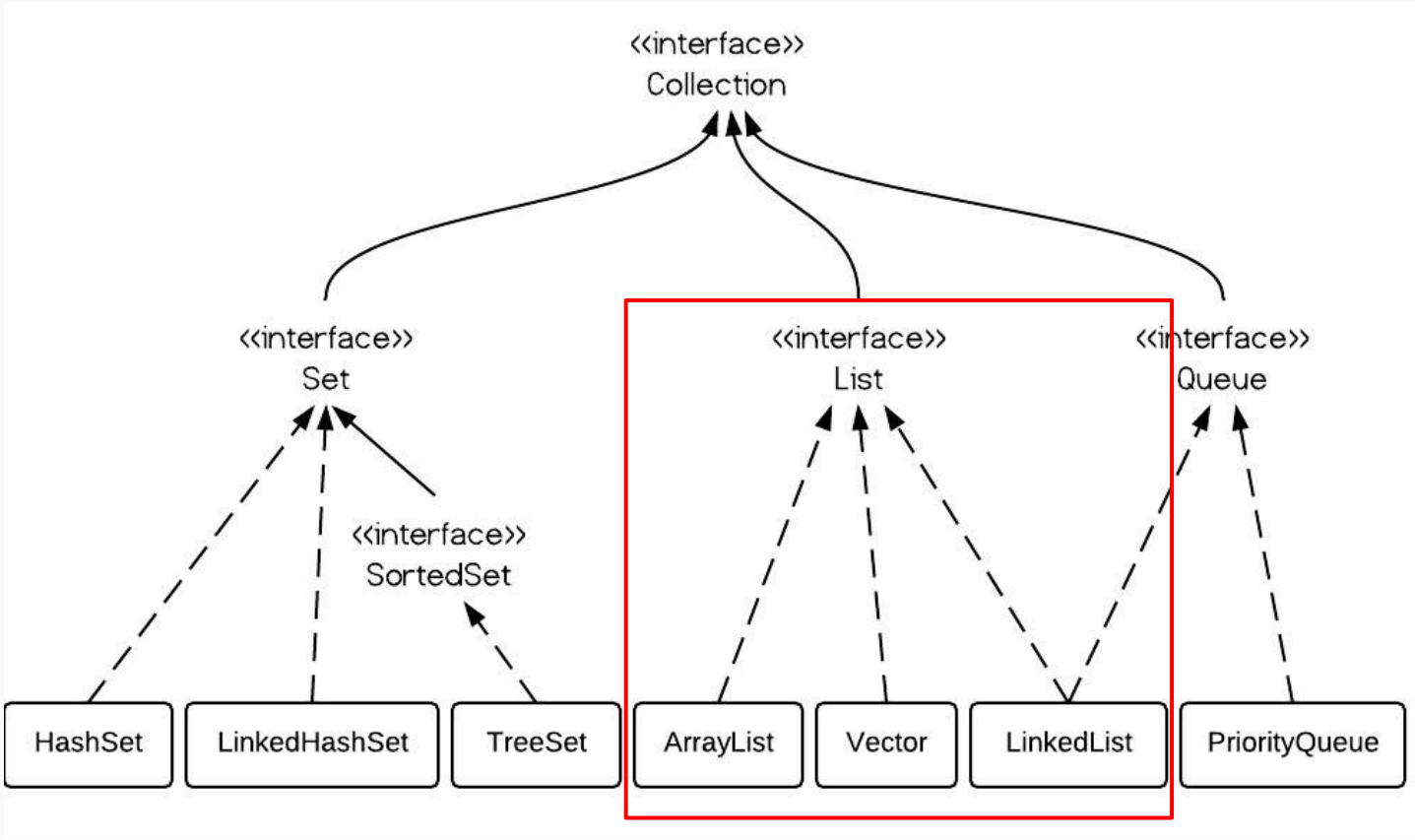
```
ArrayList<String> names = new ArrayList<>();
names.add("한석봉");
names.add("홍길동");
names.add("신사임당");

Iterator<String> it = names.iterator();

while(it.hasNext()) {
  String name = it.next();
  System.out.println(name);
}
```

각 요소는 다음 요소의 주소를 알고 있다.



삽입, 제거가 빠르다.
요소의 탐색은 느리다.

결국 List


«interface»
Collection
«interface»
Set
«interface»
SortedSet
«interface»
List
«interface»
Queue
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue

중복 값을 허용하지 않는 집합
get() 메서드는 제공하지 않기 때문에 반복이 필요하면 Iterator를 사용하거나 for each를 사용

```
Set<String> colors = new HashSet<>();

colors.add("red");
colors.add("green");
colors.add("blue");

colors.add("red");  // 중복 값

System.out.println(colors.size());  // 3
```

키(key) : 값(value) 의 쌍으로 이루어진 요소를 담는 자료구조
키의 중복은 허용되지 않음

```
Map<String, Integer> cities = new HashMap<>();
cities.put("서울시", 977);
cities.put("수원시", 124);
cities.put("부산시", 342);

int seoul = cities.get("서울시");
System.out.println("서울시 인구는 " + seoul + "만");
cities.remove("서울시");  // 삭제
cities.put("수원시", 130);  // 갱신
System.out.println("수원시 인구는 " + cities.get("수원시") + "만");
```

HashMap 에 저장된 값을 하나씩 얻기


```
for (String key: cities.keySet()) {
  int value = cities.get(key);
  System.out.println(key + " 인구는 " + value + "만");
}
```

HashMap은 순서를 보장하지 않기 때문에 매번 출력 결과의 순서가 다르게 표시 될 수 있음

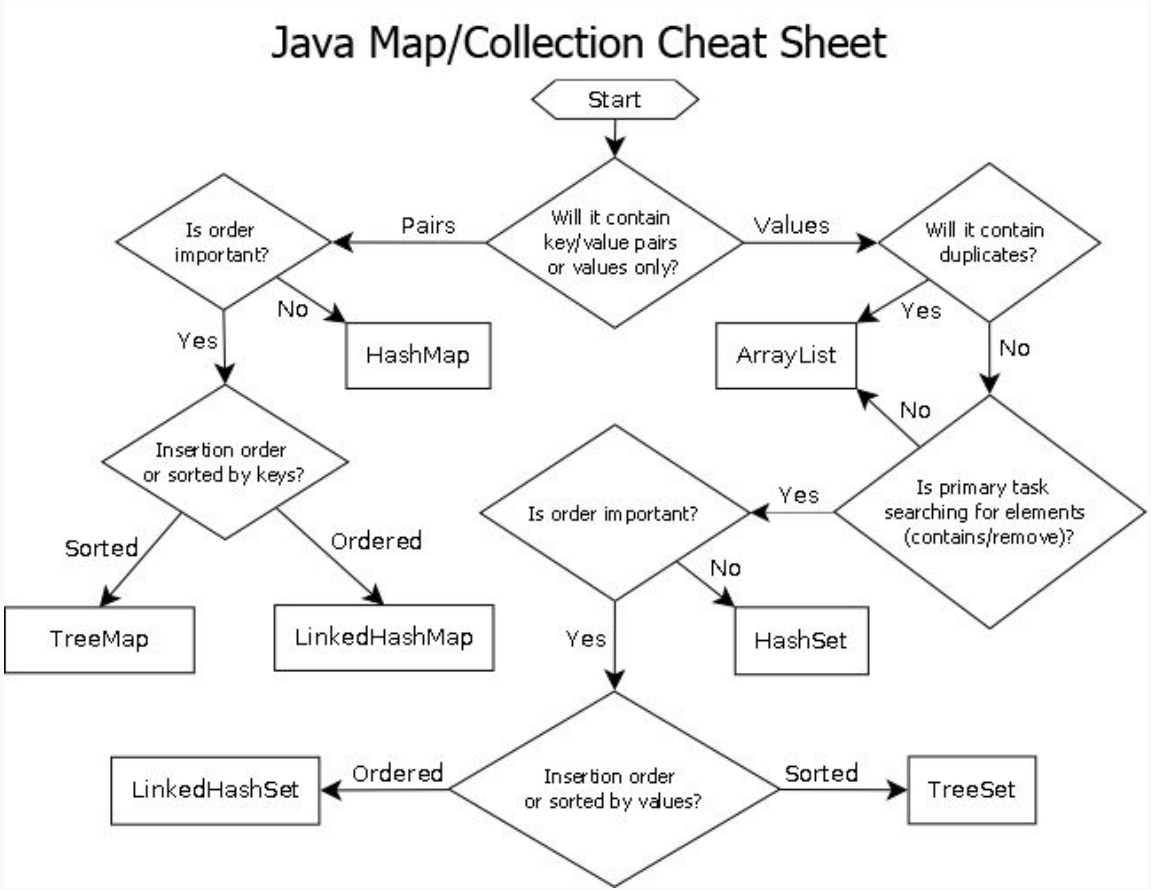
컬렉션 안에 컬렉션

Map<String, List<String>
List<List<Hero>>
…

요소의 참조에 대해


```
Hero hero = Hero();
hero.name = "홍길동";

List<Hero> heroList = new ArrayList<>();
heroList.add(hero);
hero.name = "한석봉";

System.out.println(heroList.get(0).name);    // ?
```

컬렉션 선택


Java Map/Collection Cheat Sheet
Start
Will it contain key/value pairs or values only?
Pairs
Values
Is order important?
No
HashMap
Yes
Insertion order or sorted by keys?
Sorted
Ordered
TreeMap
LinkedHashMap
Will it contain duplicates?
Yes
ArrayList
No
Is primary task searching for elements (contains/remove)?
Yes
Is order important?
No
HashSet
Yes
Insertion order or sorted by values?
Ordered
LinkedHashSet
Sorted
TreeSet

연습문제 2-1

다음 정보를 저장하기 좋은 컬렉션을 List, Set, Map 중 고르시오

1) 대한민국의 도시 이름 모음 (순서 상관 없음)
2) 10명 학생의 시험 점수
3) 대한민국의 도시별 인구수 (순서 상관 없음)

연습문제 2-2

다음을 수행하는 코드를 작성하시오.

1) 이름을 가지는 Person 클래스를 작성하시오. Person 은 반드시 이름을 포함해야 합니다.
2) 이름이 '홍길동', '한석봉' 인 Person 인스턴스를 생성하고, ArrayList에 담습니다.
3) ArrayList에 담긴 모든 Person 인스턴스의 이름을 표시하시오.

연습문제 2-3

연습문제 2-3 에서 작성한 Person 클래스로 생성한 '홍길동', '한석봉'의 나이를 각각 20, 25살 이라고 할 때, 이름과 나이를 쌍으로 적당한 컬렉션에 넣습니다.

그 다음, 컬렉션에 저장한 값을 하나씩 다음과 같이 출력합니다.

"홍길동의 나이는 20살"
"한석봉의 나이는 25살"