

4000번의 도전
드디어 찾아낸 반응 단백질



2012 인텔 국제과학경진대회 고든무어상 수상
(Intel ISEF 2012, Gordon E. Moore Award)



AN ANTONIO SPURS
처음에는 막막한 느낌이지만 시간이 지나면 감이 오거든요.



처음 배울때 감이 안나는게 뭐가 있을까요?



프로그래밍은 천재들만 할 수 있는것이 아니에요. 의지력이 필요할 뿐입니다.



Java 프로그래밍 수업내용 정리 및 실습 과제

책을 보거나 인터넷을 찾아보면서 배우고 거기에 저만의 방식을 추가했어요.



JACK ANDRAKA
IS CREATING A REVOLUTION



저는 여러분에게 컴퓨터를 통해 어떻게
세상을 바꿀 수 있는진 얘기하고 싶습니다

그러나 사실, 구글이나 위키피디아 등을 이용해
알고 싶은 모든 것을 찾을 수 있고

GOOGLE, WIKIPEDIA
AND STACKOVERFLOW

2020.04.24.금

B반 송명훈



15세 소년의 유일한 발명 도구는 “더 좋은 진단 키트를 만들고 싶었다.”
인터넷(internet)
- 잭 안드라카

응용 제 2장 컬렉션

1. 수업 내용 정리
2. 실습 화면 캡처
3. 연습문제 2-1
4. 연습문제 2-2
5. 연습문제 2-3



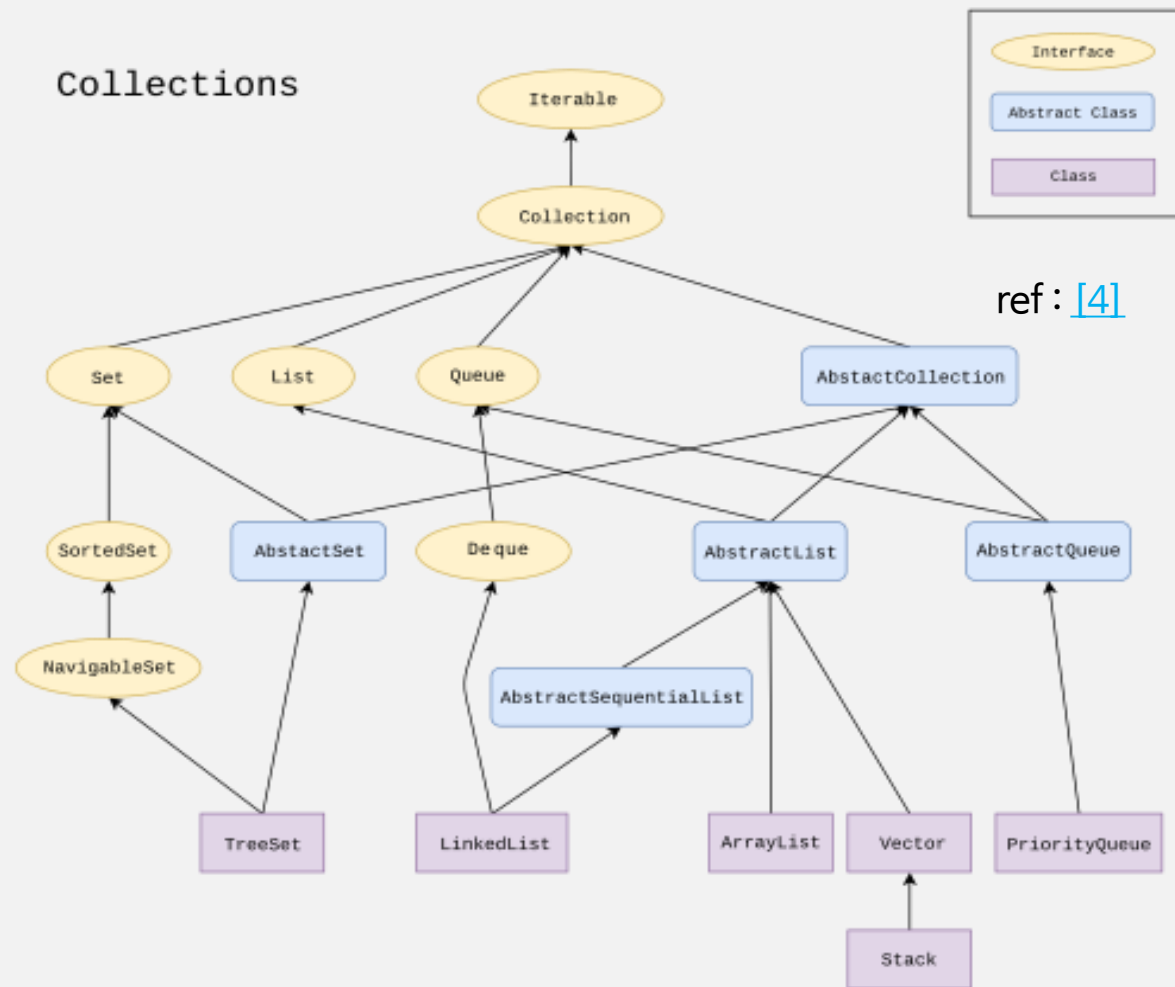
1. 수업 내용 정리

1. 수업 내용 정리 [Collection]

Collection

: Container라고도 부르며 [1] 여러 원소들을 담을 수 있는 자료 구조를 구현한 아이템 [2]

1. Java에서의 Collection은 배열처럼 다른 객체를 저장하고 다루는 것을 목적으로 하는 객체 [3]
2. 배열과 달리 Collection에는 요소의 일부를 반환하고 변환 및 정렬 등의 메서드를 포함 [3]하며, 정적 메모리가 아닌 동적 메모리 할당을 하게 된다. 즉, `new int[4]`으로 생성하게 되면 4개 공간 밖에 못 쓰고 미리 선언을 통해 4개의 공간을 만들어야 하지만, Collection은 계속 필요한 만큼 공간을 추가할 수 있다. [2]



[1] [https://en.wikipedia.org/wiki/Collection_\(abstract_data_type\)](https://en.wikipedia.org/wiki/Collection_(abstract_data_type))
[2] <https://www.crocus.co.kr/1553>

[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>
[4] https://en.wikipedia.org/wiki/Java_collections_framework

1. 수업 내용 정리 [Collection]

Java Collection Frameworks (JCF)

: Java에서 데이터를 저장하는 기본적인 자료구조들을 한 곳에 모아 관리하고 편하게 사용하기 위해서 제공하는 것.

1. Java에서 일반적으로 재사용되는 자료 구조 구현체 (Collection 객체)들을 구현하는 interface와 class들의 집합 [\[4\]](#)
2. 데이터를 저장하는 자료 구조와 데이터를 처리하는 알고리즘을 구조화하여 클래스로 구현해 놓은 것
3. JCF는 Frameworks이지만 Library 방식으로 작동 [\[4\]](#)
4. Primitive type(int, long 등)을 요소의 type으로 지정 불가. But primitive type 에 대응하는 Wrapper class를 사용하여 int, long 등을 요소로 사용 가능 [\[3\]](#)

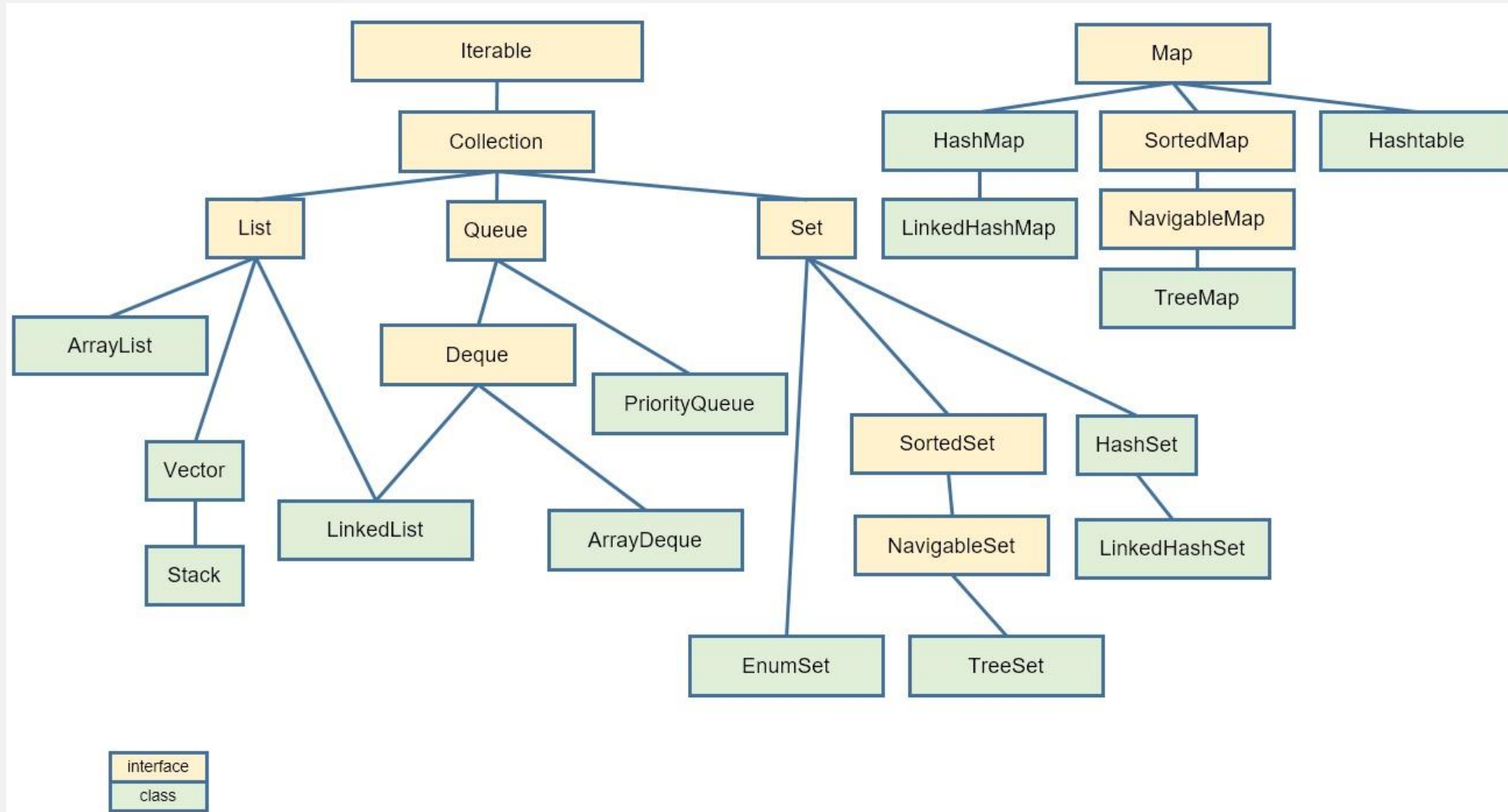
Ex] int ⇨ java.lang.Integer, long ⇨ java.lang.Long 등

[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>

[4] https://en.wikipedia.org/wiki/Java_collections_framework

1. 수업 내용 정리 [Collection]

※ abstract class 상속이 생략된 Java collection frameworks 구조 [3]



[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>

1. 수업 내용 정리 [Collection]

※ 자료 구조 별 일반 특징 비교 (대분류) [\[3\]](#)

Class or Interface	Super Interfaces	Notes	중복 허용	Null 허용	멤버접근
Collection	Iterable	전체 Collection의 상위 인터페이스.	-	-	iterator
List	Collection	원하는 순서로 Element 삽입가능. 각 요소는 Index 번호를 부여 받는다.	Yes	-	iterator 또는 숫자 Index로 임의 접근
Set	Collection	중복 Element 불가능. 그러므로 쉽게 여부 중복확인 가능. 특정 순서(Order) 정할 수 없음.	No	-	iterator
Queue	Collection	Output으로 나올 Element만 기본적으로 접근 가능하다.	Yes	-	Peeking or Polling or iterator
PriorityQueue	Queue	가장 우선순위가 높은 Element가 Head First가 된다.	Yes	No	Peeking or Polling
Deque	Queue	양 끝단에서 모두 삽입 / 삭제 가능	Yes	-	Peeking or Polling or iterator
Map	해당 없음	Key / Value로 구성된다.	키 : No 값 : Yes	키 : 하나의 (null) 키 허용 값 : Yes	통상 Key를 통해 접근 혹은 (keySet ; entrySet ; values) View를 제공

[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>

1. 수업 내용 정리 [Collection]

※ 주요 자료 구조 별 특징 비교 (실 구현 Class) [\[3\]](#)

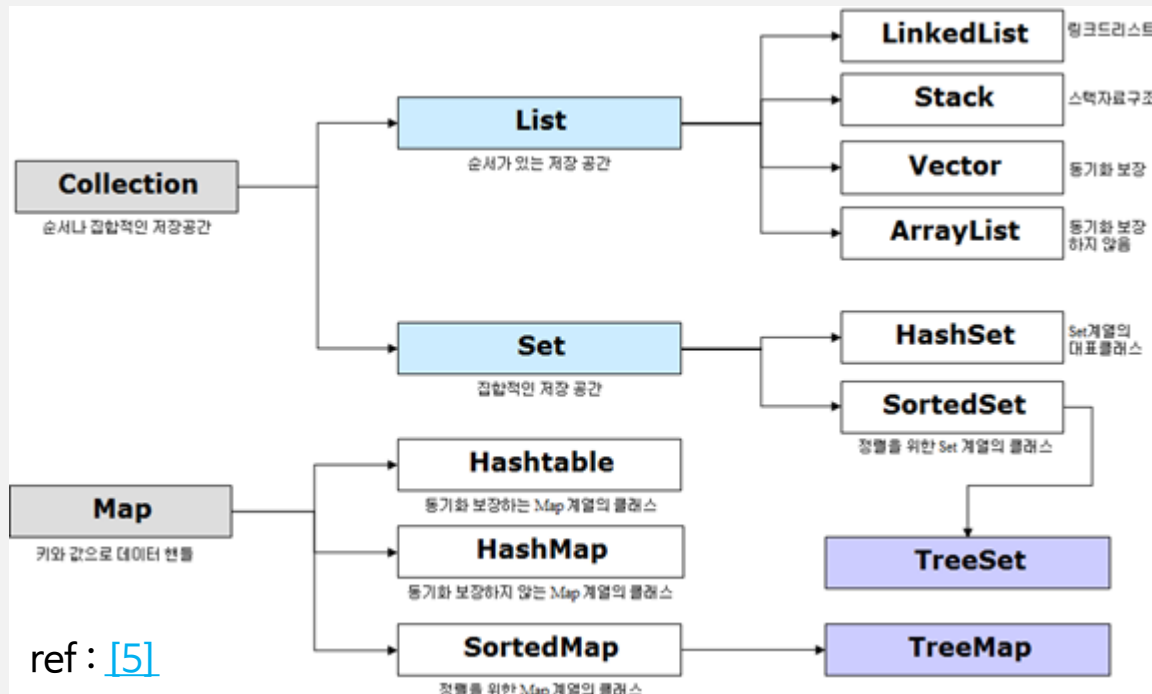
Class	Base class	Base interfaces	중복 허용	순서 (Order) 존재	정렬 여부	Thread-safe
ArrayList	AbstractList	List	Yes	Yes	No	No
HashMap	AbstractMap	Map	No	No	No	No
HashSet	AbstractSet	Set	No	No	No	No
Hashtable	Dictionary	Map	No	No	No	Yes
LinkedHashMap	HashMap	Map	No	Yes	No	No
LinkedHashSet	HashSet	Set	No	Yes	No	No
LinkedList	AbstractSequentialList	List;Deque	Yes	Yes	No	No
TreeMap	AbstractMap	Map;NavigableMap;SortedMap	No	Yes	Yes	No
TreeSet	AbstractSet	Set;NavigableSet;SortedSet	No	Yes	Yes	No
Vector	AbstractList	List	Yes	Yes	No	Yes

[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>

1. 수업 내용 정리 [Collection]

데이터 구조에 따른 대표적인 Collection

- List : 순서대로 쌓여 있는 구조 (아이템의 중복 허용)
- Map : 키(key)와 값(value)의 쌍으로 저장
(key의 중복 불가)
- Set : 순서가 없는 집합 (중복 불가)



- List와 Set 인터페이스는 모두 Collection 인터페이스를 상속받지만, 구조상의 차이로 인해 Map 인터페이스는 별도로 정의 [\[6\]](#)
- 따라서 List 인터페이스와 Set 인터페이스의 공통된 부분을 Collection 인터페이스에서 정의 [\[6\]](#)
- Map은 Collection 인터페이스를 상속하지 않음에도 불구하고, 일반적으로 Collection을 이야기 할 때 항상 포함 [\[3\]](#)

※ 각 인터페이스의 특징 [\[7\]](#)

인터페이스	구현 클래스	특 징
List	LinkedList Stack Vector ArrayList	<ul style="list-style-type: none"> • 순서가 있는 데이터의 집합. • 데이터의 중복을 허용함.
Set	HashSet TreeSet	<ul style="list-style-type: none"> • 순서를 유지하지 않는 데이터의 집합. • 데이터의 중복을 허용하지 않음
Map	HashMap TreeMap HashTable Properties	<ul style="list-style-type: none"> • 키(key)와 값(value)의 쌍으로 이루어진 데이터의 집합 • 순서 유지 X, key 중복 X, 값 중복 O

[3] <http://blog.breakingthat.com/2018/05/07/java-collection-%EA%B0%9C%EC%9A%94-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0/>, [5] <https://gangnam-american.tistory.com/41>, [6] http://tcpschool.com/java/java_collectionFramework_concept, [7] <https://hackersstudy.tistory.com/26>

1. 수업 내용 정리 [Collection]

Collection 인터페이스

- 모든 Collection의 상위 인터페이스로서 collection들이 가지고 있는 핵심 메서드를 선언. [\[7\]](#)
- List와 Set 인터페이스의 많은 공통된 부분을 Collection 인터페이스에서 정의하고, 두 인터페이스는 그 것을 상속받는다. [\[6\]](#)
- 따라서 Collection 인터페이스는 컬렉션을 다루는데 가장 기본적인 동작들을 정의하고, 그것을 메서드로 제공한다. [\[6\]](#)
- Collection 인터페이스에서 제공하는 주요 메서드 [\[6\]](#)

메서드	설명
boolean add(E e)	해당 컬렉션(collection)에 전달된 요소를 추가함. (선택적 기능)
void clear()	해당 컬렉션의 모든 요소를 제거함. (선택적 기능)
boolean contains(Object o)	해당 컬렉션이 전달된 객체를 포함하고 있는지를 확인함.
boolean equals(Object o)	해당 컬렉션과 전달된 객체가 같은지를 확인함.
boolean isEmpty()	해당 컬렉션이 비어 있는지를 확인함.
Iterator<E> iterator()	해당 컬렉션의 반복자(iterator)를 반환함.
boolean remove(Object o)	해당 컬렉션에서 전달된 객체를 제거함. (선택적 기능)
int size()	해당 컬렉션의 요소의 총 개수를 반환함.
Object[] toArray()	해당 컬렉션의 모든 요소를 Object 타입의 배열로 반환함.

[6] http://tcpschool.com/java/java_collectionFramework_concept

[7] <https://hackersstudy.tistory.com/26>

1. 수업 내용 정리 [Collection]

List 인터페이스

- Collection 인터페이스를 확장한 자료형으로 요소들의 순서를 저장하여 색인(Index)를 사용해 특정 위치에 요소를 삽입하거나 접근할 수 있으며 중복 요소 허용 [\[7\]](#)

이름	설 명
ArrayList	<ul style="list-style-type: none">상당히 빠르고 크기를 마음대로 조절할 수 있는 배열 형태의 List단방향 포인터 구조로 자료에 대한 순차적인 접근에 강점이 있음데이터의 삽입, 삭제 시 많은 연산이 필요한 단점이 있다.
Vector	<ul style="list-style-type: none">ArrayList의 구형 버전이며, 모든 메서드가 동기화 되어 있음잘 쓰이지 않는다
LinkedList	<ul style="list-style-type: none">양방향 포인터 구조로 데이터의 삽입, 삭제가 빈번할 경우 빠른 성능을 보장스택, 큐, 양방향 큐 등을 만들기 위한 용도로 쓰임

1. 수업 내용 정리 [Collection]

Set 인터페이스 : 집합을 정의하며 요소의 중복을 허용하지 않는다. 상위 메서드만 사용한다. [\[7\]](#)

이름	설 명
HashSet	<ul style="list-style-type: none">가장 빠른 임의 접근 속도순서 지정 불가, 순서를 전혀 예측 할 수 없음
LinkedHashSet	<ul style="list-style-type: none">추가된 순서, 또는 가장 최근에 접근한 순서대로 접근 가능
TreeSet	<ul style="list-style-type: none">정렬된 순서대로 보관하며 정렬 방법을 지정할 수 있음

Map 인터페이스 : Key와 Value의 쌍으로 연관되어 저장하는 객체 [\[7\]](#)

이름	설 명
HashMap	<ul style="list-style-type: none">Map 인터페이스를 구현하기 위해 HashTable을 사용한 클래스중복을 허용하지 않고 순서를 보장하지 않음키와 값으로 null 허용
HashTable	<ul style="list-style-type: none">HashMap 보다는 느리지만 동기화가 지원키와 값으로 null이 허용되지 않음
TreeMap	<ul style="list-style-type: none">이진검색트리의 형태로 키와 값의 쌍으로 이루어진 데이터를 저장정렬된 순서로 키/값 쌍을 저장하므로 빠른 검색 가능저장 시 정렬(오름차순)하므로 저장시간이 다소 오래걸림
LinkedHashMap	<ul style="list-style-type: none">기본적으로 HashMap을 상속받아 HashMap과 매우 흡사Map에 있는 엔트리들의 연결리스트를 유지하므로 입력한 순서대로 반복가능

1. 수업 내용 정리 [Collection]

각 클래스의 메소드 링크

- [Collection](#)
- [List](#)
- [Map](#)
- [ArrayList](#)
- [Vector](#)
- [LinkedList](#)
- [HashSet](#)
- [LinkedHashSet](#)
- [TreeSet](#)
- [HashMap](#)
- [Hashtable](#)
- [TreeMap](#)
- [LinkedHashMap](#)



2. 실습 화면 캡처

2. 실습 화면 캡처 [Collection]

배열과 ArrayList 문법 비교

```
public class Main {  
    public static void main(String[] args) {  
        // 배열 확보  
        String[] names = new String[3];  
        // 3인 추가  
        names[0] = "홍길동";  
        names[1] = "한석봉";  
        names[2] = "신사임당";  
  
        System.out.println(names[1]);  
    }  
}
```

Main ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe"
한석봉

Process finished with exit code 0

```
public class Main {  
    public static void main(String[] args) {  
        // ArrayList 확보  
        ArrayList<String> names = new ArrayList<>();  
        // 3인 추가  
        names.add("홍길동");  
        names.add("한석봉");  
        names.add("신사임당");  
  
        System.out.println(names.get(1));  
    }  
}
```

크기를 정해두지 않고
요소를 추가할 때마다
크기가 커짐

Main ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-j
한석봉

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

Collection에서 Primitive type 취급 불가

```
public class Main {  
    public static void main(String[] args) {  
        ArrayList<Integer> names = new ArrayList<>();  
  
        names.add(10);  
        names.add(20);  
        names.add(30);  
  
        System.out.println(names.get(1));  
    }  
}
```

```
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-ja  
20
```

```
Process finished with exit code 0
```

java.lang 패키지에서 제공하는 Wrapper Class 활용

- int ⇔ Integer
- long ⇔ Long
- boolean ⇔ Boolean
- char ⇔ Character
- short ⇔ Short
- byte ⇔ Byte 등

2. 실습 화면 캡처 [Collection]

배열과 ArrayList 탐색 방법 비교

```
public class Main {  
    public static void main(String[] args) {  
        String[] names = new String[3];  
        names[0] = "홍길동";  
        names[1] = "한석봉";  
        names[2] = "신사임당";  
        for (int i = 0; i < names.length ; i++) {  
            System.out.println(names[i]);  
        }  
    }  
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "  
홍길동  
한석봉  
신사임당
```

Process finished with exit code 0

```
public class Main {  
    public static void main(String[] args) {  
        ArrayList<String> names = new ArrayList<>();  
        names.add("홍길동");  
        names.add("한석봉");  
        names.add("신사임당");  
        for (int i = 0; i < names.size() ; i++) {  
            System.out.println(names.get(i));  
        }  
    }  
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-j  
홍길동  
한석봉  
신사임당
```

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

배열과 ArrayList 탐색 방법 비교 2

```
public class Main {  
    public static void main(String[] args) {  
        String[] names = new String[3];  
        names[0] = "홍길동";  
        names[1] = "한석봉";  
        names[2] = "신사임당";  
        for (String name : names) {  
            System.out.println(name);  
        }  
    }  
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.e  
홍길동  
한석봉  
신사임당
```

Process finished with exit code 0

```
public class Main {  
    public static void main(String[] args) {  
        ArrayList<String> names = new ArrayList<>();  
        names.add("홍길동");  
        names.add("한석봉");  
        names.add("신사임당");  
        for (String name : names) {  
            System.out.println(name);  
        }  
    }  
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-j  
홍길동  
한석봉  
신사임당
```

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

Iterator : List의 요소를 하나씩 가리키는 객체

```
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("홍길동");
        names.add("한석봉");
        names.add("신사임당");

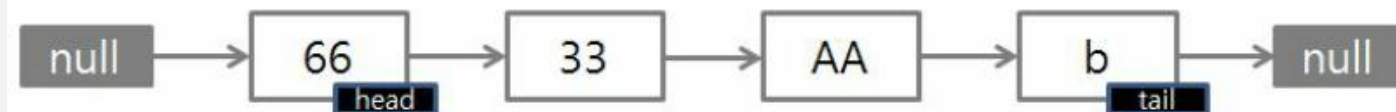
        Iterator<String> it = names.iterator();
        while(it.hasNext()) {
            String name = it.next();
            System.out.println(name);
        }
    }
}
```

```
Main x
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-jav
홍길동
한석봉
신사임당

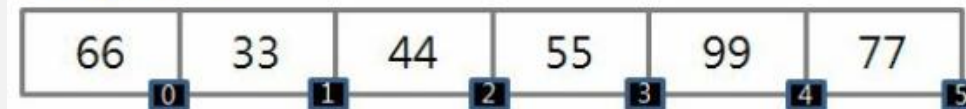
Process finished with exit code 0
```

LinkedList : 다음 각 요소의 주소를 알고 있음

Linked List



Array List



삽입, 제거가 빠르다 - 참조만 새로 이어주면 됨

요소의 탐색은 느리다 - 요소마다 주소를 찾아가야 되므로

2. 실습 화면 캡처 [Collection]

HashSet

중복 값을 허용하지 않는 집합

get() 메서드는 제공 않기 때문에 반복이 필요하다면

Iterator를 사용하거나 for each를 사용

```
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        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
    }
}
```

Main ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe"

3

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

HashMap

- 키(key) : 값(value)의 쌍으로 이루어진 요소를 담는 자료구조
- 키의 중복은 불허

```
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        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("서울시의 인구는 " + seoul + "만");
        cities.remove(key: "서울시"); // 삭제
        cities.put("수원시", 130); // 갱신
        System.out.println("수원시의 인구는 " + cities.get("수원시") + "만");
    }
}
```

Main ×

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program File
서울시의 인구는 977만
수원시의 인구는 130만
```

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

- HashMap에 저장된 값을 하나씩 얻기
HashMap은 순서를 보장하지 않기 때문에
매번 출력 결과의 순서가 다르게 표시될 수 있음

```
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<String, Integer> cities = new HashMap<>();
        cities.put("서울시", 977);
        cities.put("수원시", 124);
        cities.put("부산시", 342);

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

Main ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\Java\jdk-13.0.2\bin\java.exe" -Djava.class.path=C:\Program Files\Java\jdk-13.0.2\bin\java.exe

부산시 인구는 342만
수원시 인구는 124만
서울시 인구는 977만

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]

- Collection의 응용

Map<String, List<String>>

List<List<Hero>>

...

- 요소의 참조에 대해

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        Hero hero = new Hero();
        hero.name = "홍길동";

        List<Hero> heroList = new ArrayList<>();
        heroList.add(hero);
        hero.name = "한석봉";
        System.out.println(heroList.get(0).name);
    }
}
```

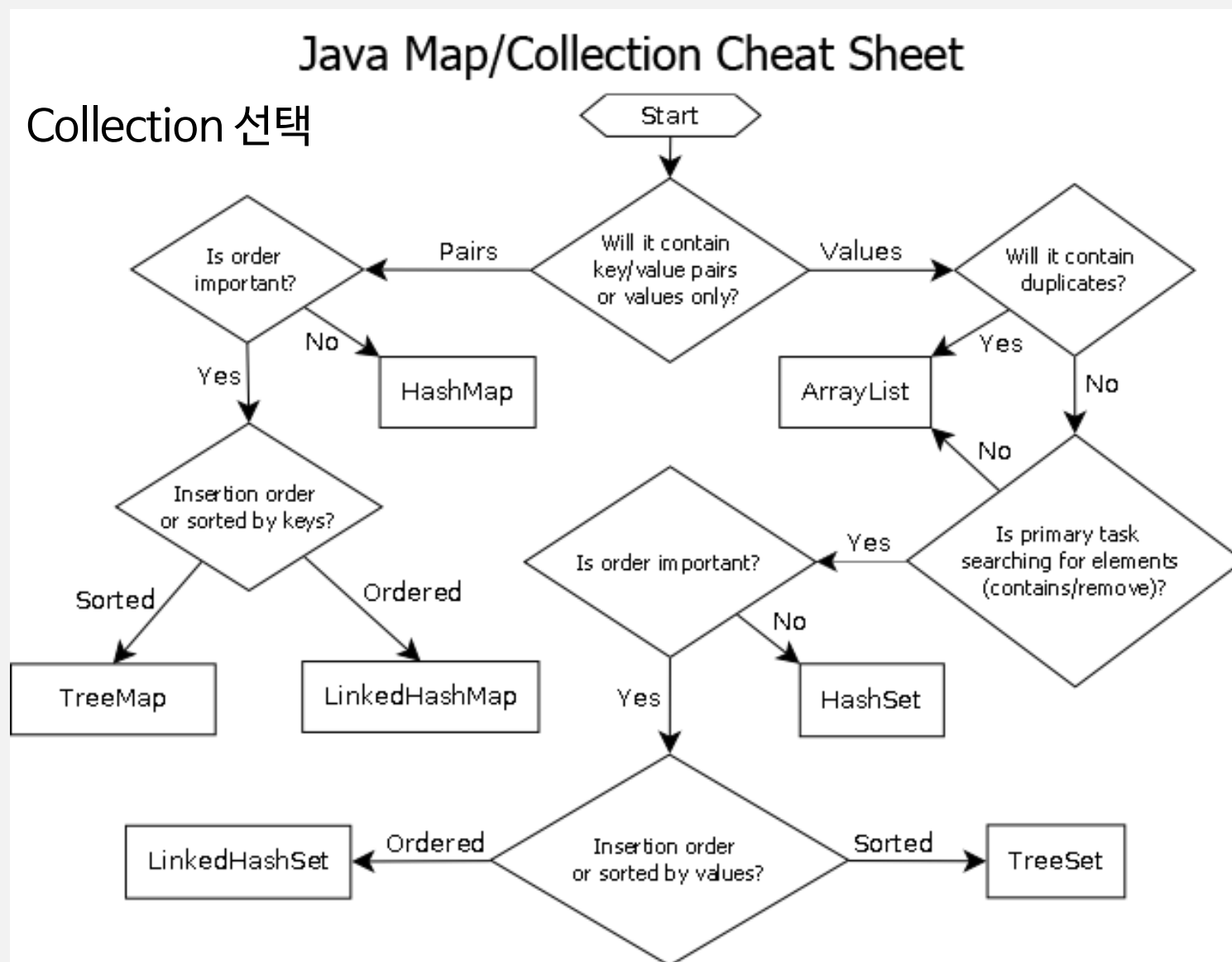
Main ×

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe"

한석봉

Process finished with exit code 0

2. 실습 화면 캡처 [Collection]





3. 연습문제 2-1

3. 연습문제 2-1

2-1. 문제 : 다음 정보를 저장하기 좋은 컬렉션을 List, Set, Map 중 고르시오

1) 대한민국의 도시 이름 모음 (순서 상관 없음)

순서 무관, 중복 불허 이므로 Set

2) 10명 학생의 시험 점수 (이중 List)

```
List<List<String>> studentGrade = new ArrayList<>();
```

```
List<String> name = new ArrayList<>();
```

```
List<String> grade = new ArrayList<>();
```

```
studentGrade.add(name);
```

```
studentGrade.add(grade);
```

3) 대한민국의 도시 별 인구수 (순서 상관 없음)

순서 무관이므로 Map<도시이름, 인구수>



4. 연습문제 2-2

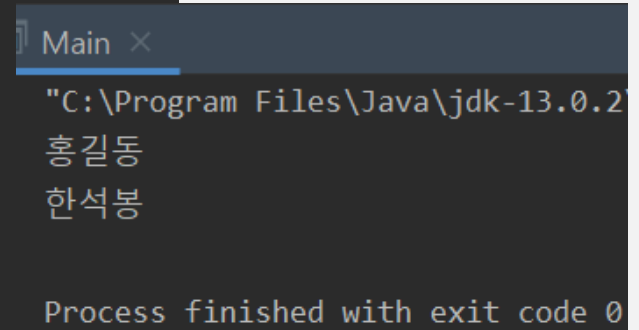
4. 연습문제 2-2

2-2. 문제 : 다음을 수행하는 코드를 작성하시오

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

```
public class Person {  
    private String name;  
    public Person(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Person gilDong = new Person( name: "홍길동");  
        Person seokBong = new Person( name: "한석봉");  
        List<Person> people = new ArrayList<>();  
        people.add(gilDong);  
        people.add(seokBong);  
        for (Person person : people) {  
            if(person instanceof Person) {  
                System.out.println(person.getName());  
            } else {  
                System.out.println("Error.ElementTypeUnsuitable");  
            }  
        }  
    }  
}
```



```
Main ×  
"C:\Program Files\Java\jdk-13.0.2"  
홍길동  
한석봉  
Process finished with exit code 0
```



5. 연습문제 2-3

5. 연습문제 2-3

2-3. 문제

- 연습문제 2-3에서 작성한 Person 클래스로 생성한 '홍길동', '한석봉'의 나이를 각각 20살, 25살이라고 할 때, 이름과 나이를 쌍으로 적당한 컬렉션에 넣습니다.
- 그 다음, 컬렉션에 저장한 값을 하나씩 다음과 같이 출력합니다.
"홍길동의 나이는 20살"
"한석봉의 나이는 25살"

5. 연습문제 2-3

```
public class Main {  
    public static void main(String[] args) {  
        Person gilDong = new Person( name: "홍길동");  
        Person seokBong = new Person( name: "한석봉");  
        List<Person> people = new ArrayList<>();  
        people.add(gilDong);  
        people.add(seokBong);  
        // Map 자료형의 personInfo 인스턴스 생성  
        Map<String, Integer> personInfo = new HashMap<>();  
        // personInfo에 앞에서 생성한 Person의 인스턴스에서 호출한 getName() 값 저장  
        personInfo.put(people.get(0).getName(), 20);  
        personInfo.put(people.get(1).getName(), 25);  
        // StringBuilder를 이용한 문자열 합성 및 출력  
        for (int i = 0; i < personInfo.size(); i++) {  
            StringBuilder sb = new StringBuilder();  
            sb.append(people.get(i).getName()).append("의 나이는 ");  
            sb.append(personInfo.get(people.get(i).getName())).append("살");  
            String str = sb.toString();  
            System.out.println(str);  
        }  
    }  
}
```

```
Main ×  
"C:\Program Files\Java\jdk-13.0.2\  
홍길동의 나이는 20살  
한석봉의 나이는 25살  
|  
Process finished with exit code 0
```

Person 클래스는
연습문제 2-2에서
정의한 그대로 사용