Data Analysis

(Data modelling, collecting, and analyses 5)

Fall, 2020

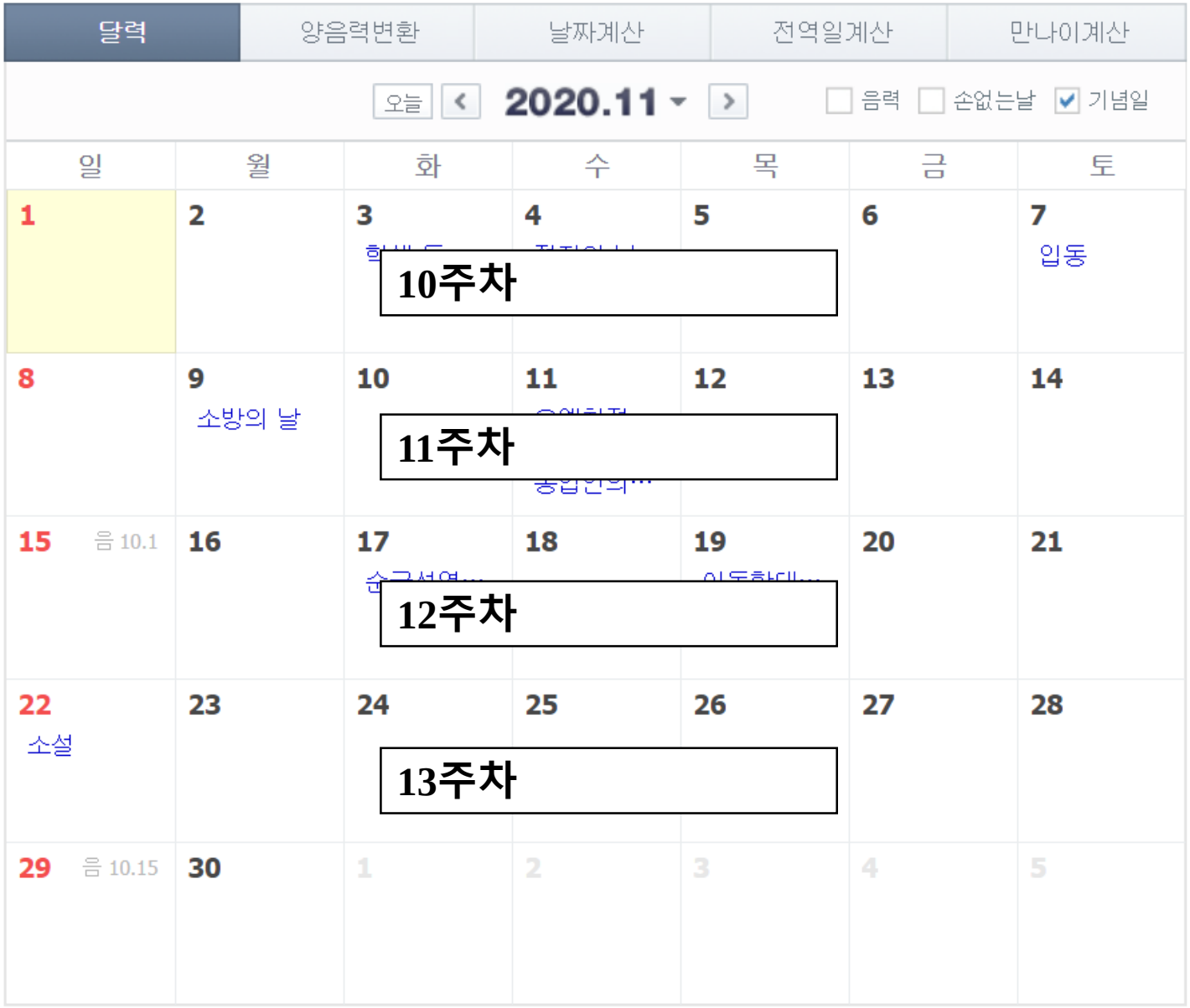
Calendar


달력
양음력변환
날짜계산
전역일계산
만나이계산
오늘
2020.09
음력
손없는날
기념일
일
월
화
수
목
금
토
30
31
1
소개
2
음 7.15
3
환경
세팅
4
지식재산…
5
6
7
백로
8
복습
1
9
10
9. 1
복습
2
11
12
13
14
15
16
17
음 8.1
18
19
청년의 날
3주차
20
21
치매극복…
22
23
24
25
26
4주차
27
28
29
30
1
2
3
5주차

Calendar


달력
양음력변환
날짜계산
전역일계산
만나이계산
오늘
2020.10
음력
손없는날
기념일
일
월
화
수
목
금
토
27
28
29
30
1 음 8.15
추석
국군의 날
2
노인의 날
3
개천절
4
5
세계 한…
6
7
8
9
한글날
10
6주차
11
12
13
14
15
16
부마민주…
17 음 9.1
문화의 날
7주차
18
19
20
21
22
23
상강
24
국제연합일
8주차: 중간고사
25
독도의날
중양절
26
27
28
29
지방자치…
30
31 음 9.15
9주차

Calendar


달력
양음력변환
날짜계산
전역일계산
만나이계산
오늘
2020.11
음력
손없는날
기념일
일
월
화
수
목
금
토
1
2
3
4
5
6
7
입동
10주차
8
9
소방의 날
10
11
12
13
14
11주차
15
음 10.1
16
17
18
19
20
21
12주차
22
소설
23
24
25
26
27
28
13주차
29
음 10.15
30
1
2
3
4
5

Calendar


달력
양음력변환
날짜계산
전역일계산
만나이계산
오늘
2020.12
음력
손없는날
기념일
일 월 화 수 목 금 토
29 30 1 2 3 4 5
무역의 날
14주차
6 7 8 9 10 11 12
대설
15주차
13 14 15 음 11.1 16 17 18 19
16주차: 기말고사 주간
20 21 22 23 24 25 26
동지
성탄절
27 28 29 음 11.15 30 31 1 2
원자력의…

Table of Contents

Question #0

- Compute the identifier where its occurrence is maximum in the dataset.
- 가장 많이 등장한 ID를 출력하시오.
  - e.g., Dataset
    - 1 2
    - 1 4
    - 1 8
    - 2 3
    - 5 8

    1 is seen 3 times

    2,8 … 2 times

    3,4,5 … 1 time

    So 1 is the answer

1 → 3
2 → 2
3 → 1
4 → 1
5 → 1
8 → 2

See a HashSet-based solution

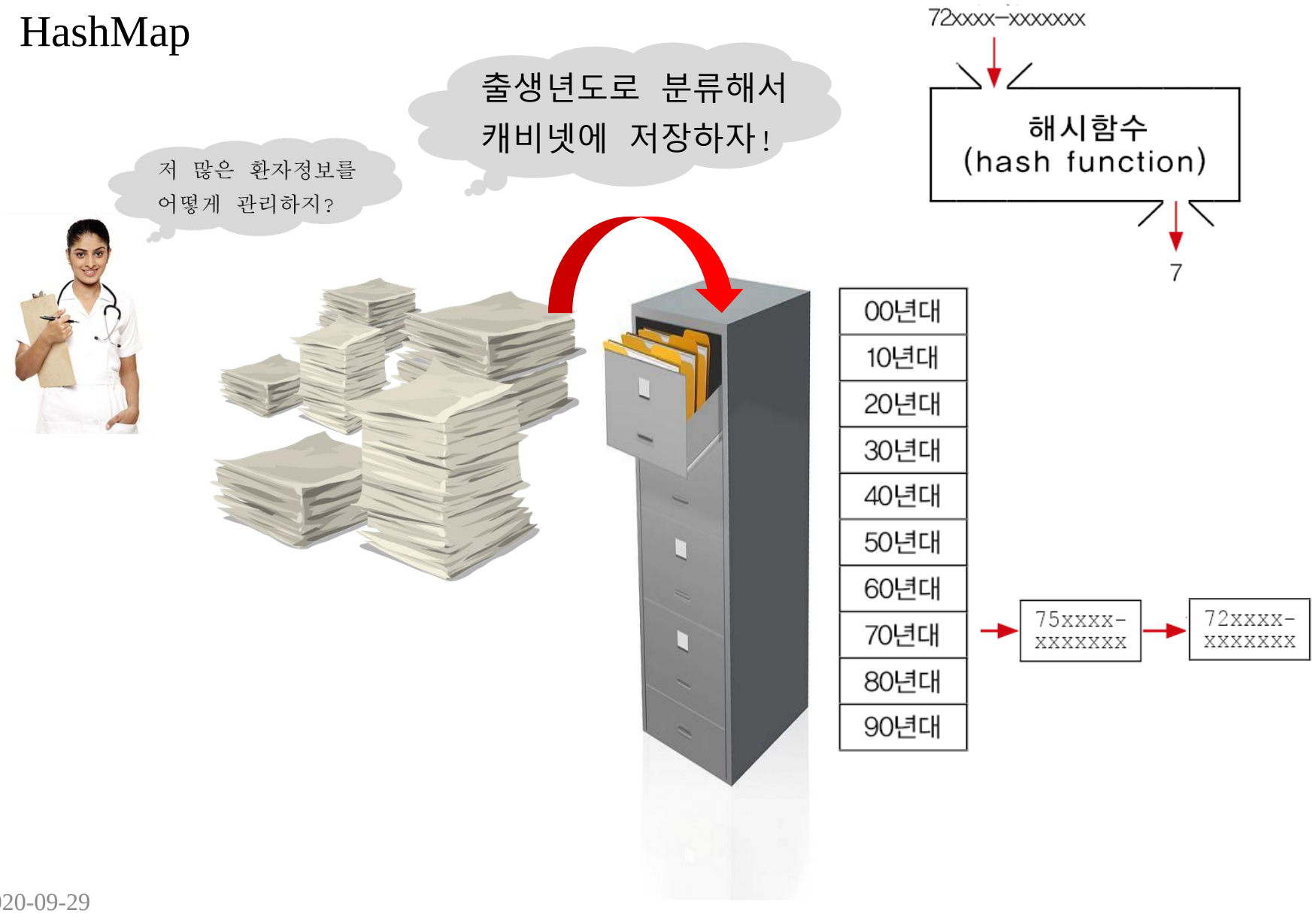
Collection Framework: HashMap

- HashMap
  - Key-value pairs of
  - Unordered, Non-redundant elements E
  - Based on Hashing
  - CRUD by hash
  - HashSet is just a special case of HashMap

```
public class HashSet<E> implements Set<E> {
        …
        private HashMap<E,Object> map;
        …
}
```

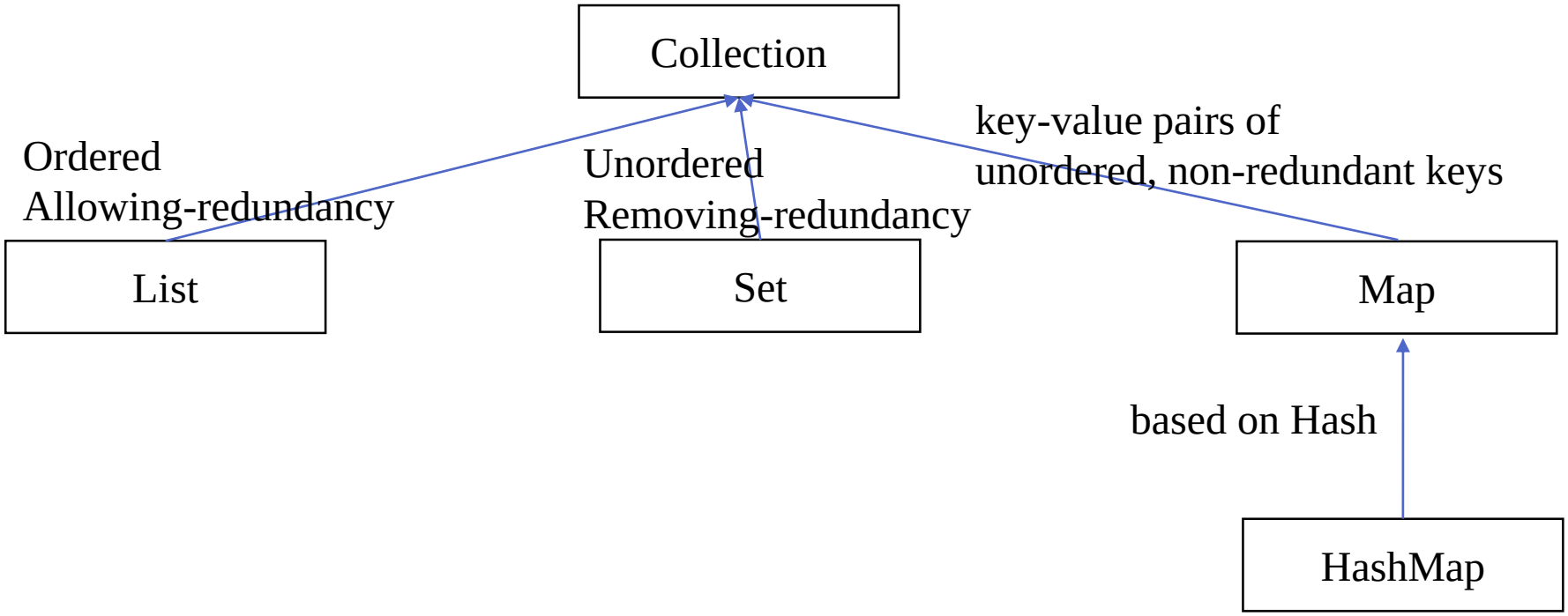
Collection Framework: HashMap

- HashMap



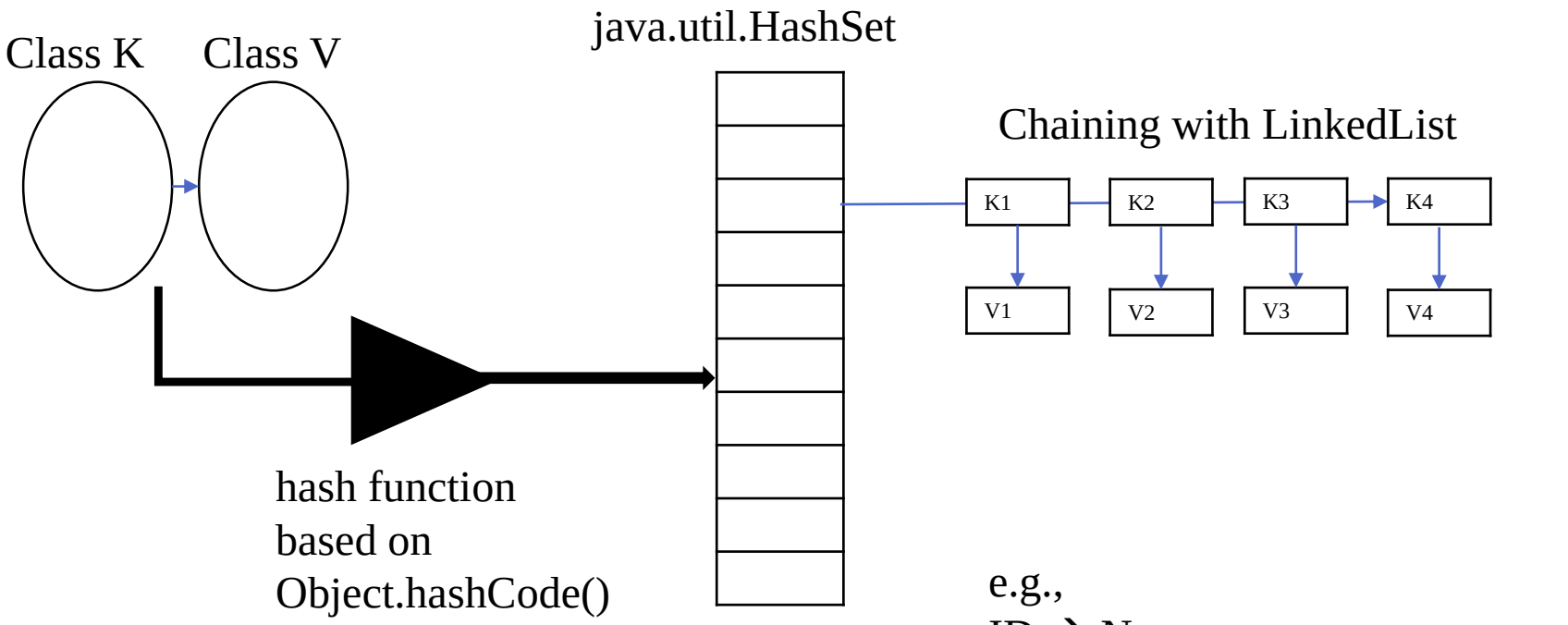
Collection Framework: HashMap

- HashSet consists of unordered, non-redundant elements



http://www.thecshandbook.com/Hash_Set

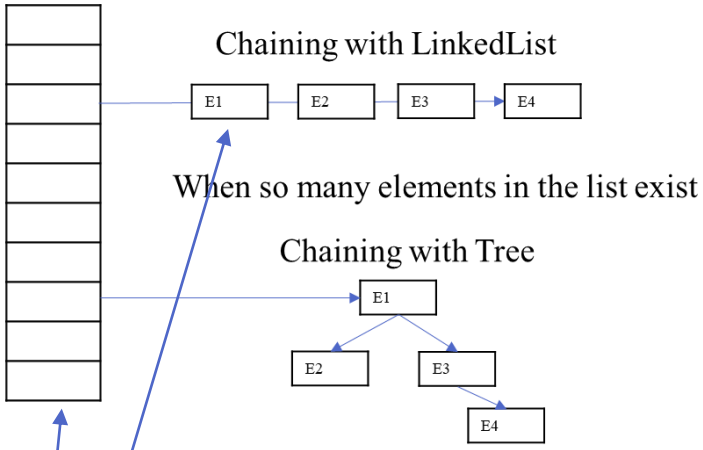
Collection Framework: HashMap



e.g.,
ID → Name
Phone Number → Device Name

Collection Framework: HashMap

- HashMap
  - Unordered, Non-redundant elements E
  - Based on Hashing
  - CRUD by hash




```
public class HashMap<K,V> extends AbstractMap<K,V>
    implements Map<K,V>, Cloneable, Serializable {

    transient Node<K,V>[] table;
     static class Node<K,V> implements
    Map.Entry<K,V> {
            final int hash;
            final K key;
            V value;
            Node<K,V> next;
              …
    }

}
```

Collection Framework: HashMap

- HashMap
  - implementing Map

메서드	설 명
void clear( )	Map의 모든 객체를 삭제한다.
boolean containsKey(Object key)	지정된 key객체와 일치하는 Map의 key객체가 있는지 확인한다.
boolean containsValue(Object value)	지정된 value객체와 일치하는 Map의 value객체가 있는지 확인한다.
Set entrySet( )	Map에 저장되어 있는 key-value쌍을 Map.Entry타입의 객체로 저장한 Set으로 반환한다.
boolean equals(Object o)	동일한 Map인지 비교한다.
Object get(Object key)	지정한 key객체에 대응하는 value객체를 찾아서 반환한다.
int hashCode( )	해시코드를 반환한다.
boolean isEmpty( )	Map이 비어있는지 확인한다.
Set keySet( )	Map에 저장된 모든 key객체를 반환한다.
Object put(Object key, Object value)	Map에 value객체를 key객체에 연결(mapping)하여 저장한다.
void putAll(Map t)	지정된 Map의 모든 key-value쌍을 추가한다.
Object remove(Object key)	지정한 key객체와 일치하는 key-value객체를 삭제한다.
int size( )	Map에 저장된 key-value쌍의 개수를 반환한다.
Collection values( )	Map에 저장된 모든 value객체를 반환한다.

Collection Framework: HashMap

- HashMap
  - Practice HashMap
    - CRUD
    - Generics
    - Iterator
    - hashCode
    - equals

Question #3

- Compute the identifier where its occurrence is maximum in the dataset.

- Keep all the key-value pairs of ID and its occurrence

Question #4

- Compute the number of people who received email(s) from ?.

Question #5

- Compute the number of people who received email(s) from ?.

HashMap<K,V>: K, V could be any class such as

HashMap<String, Double>

HashMap<String, HashSet<String>>

HashMap<String, HashMap<Double, Integer>>

HashMap<Integer, HashMap<Integer, HashSet<String>>>

Question #6

- Compute the number of people who send email(s) to ?.
- Compute the number of people who received email(s) from people who received email(s) from ?.
- Compute the number of people who send email(s) to people who send email(s) to ?.
- Compute the number of people [who received email(s) from people]* ?.
- Compute the number of people [who send email(s) to people]* ?.

Second Dataset

- Super User temporal network
  - http://snap.stanford.edu/data/sx-superuser.html
  - Syntax
    - Source Destination UNIX_EPOCH
  - A temporal network of interactions on the stack exchange
    - A source sends a message to a destination at a specific time

Question #7

- Compute the first unix epoch time
- Compute the second unix epoch time
- Compute the last unix epoch time
- Compute the second last unix epoch time

Temporal Data

- A collection of records where each record has valid at a specific time

Time	Temperature
1	36.5
2	36.7
3	37.5
4	36.9
5	…

Time	Source	Destination
1	1	2
2	1	3
3	3	5
4	3	7
5	5	9

- How to represent a time?
  - String: “2020-09-07”
  - String: “2020-09-07T15:50:32”
  - String: “2020/09/07T15:50:32”
  - String: “2020-09-07 15:50:32”
  - long: 1599461432863 milliseconds

Temporal Data

- Get Unix Epoch Time
  - java.lang.System.currentTimeMillis()
    - Returns the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
- Useful built-in class
  - Date
    - https://docs.oracle.com/javase/8/docs/api/java/util/Date.html
    - abstracts the date
      - Year
      - Month
      - Day of Month
      - Day of Week
      - Hour of Day
      - Minute
      - Second
      - Millisecond
  - SimpleDateFormat
    - You can parse and format the time
    - https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

Question #1

- Compute the time that occurs the most events
  - E.g.,

Source	Destination	Time
1	2	1
1	3	1
1	4	1
2	5	2
2	6	2
1	5	3

  - Answer: 1 (3 events occur)

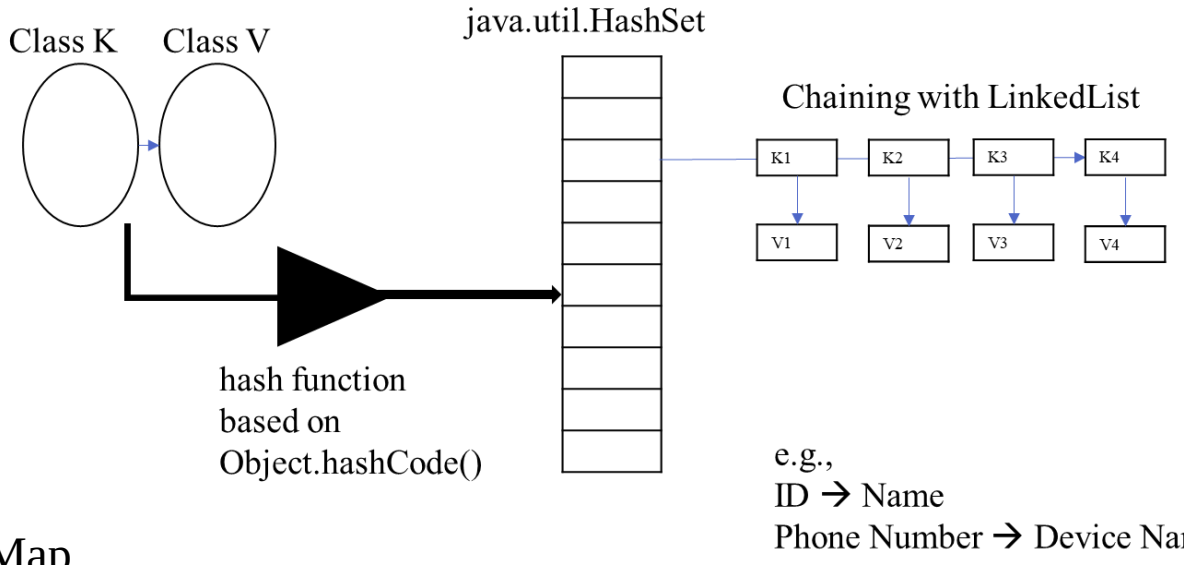
Question #2

- List up the events occurred at a time just next (chronologically) to the time that occurs the most events
  - E.g.,

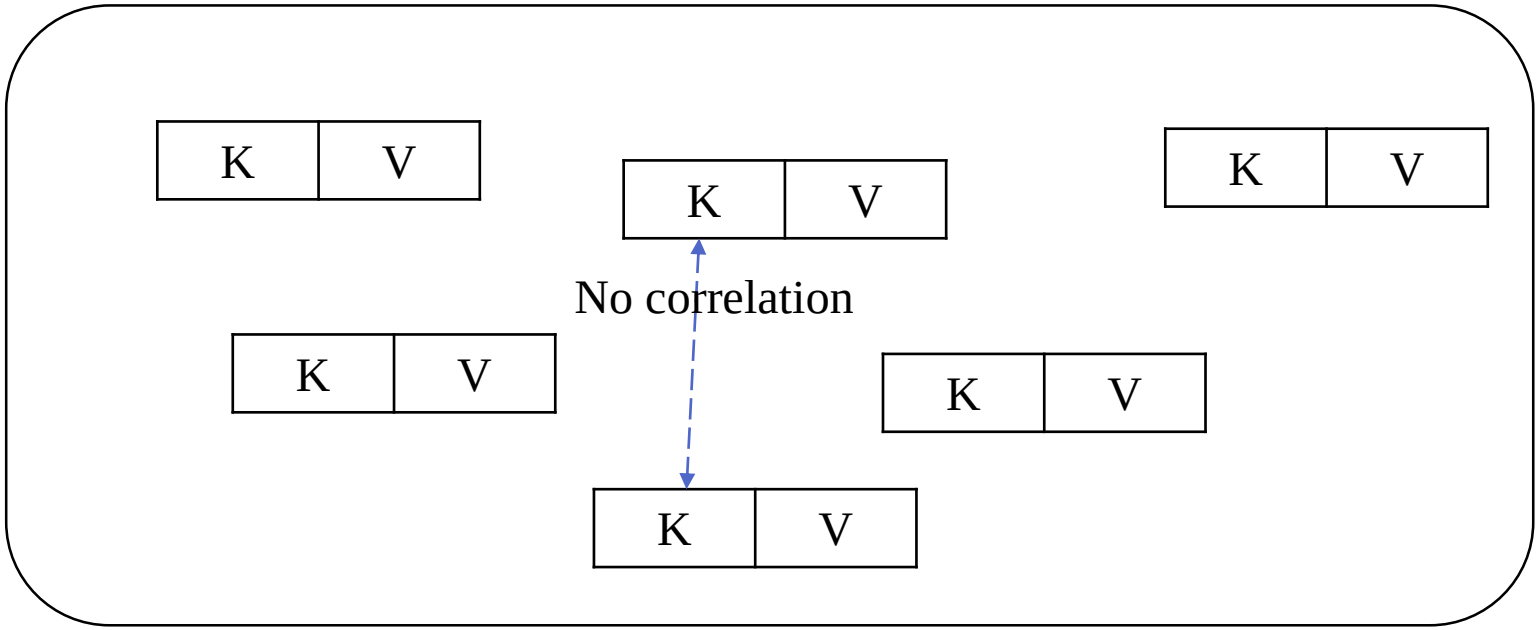
Source	Destination	Time
1	2	1
1	3	1
1	4	1
2	5	2
2	6	2
1	5	3

  - Answer: [2 5 **2**, 2 6 **2**] because at the time 1 the most events occurred
  - Think about how you find the answer

Collection Framework: HashMap

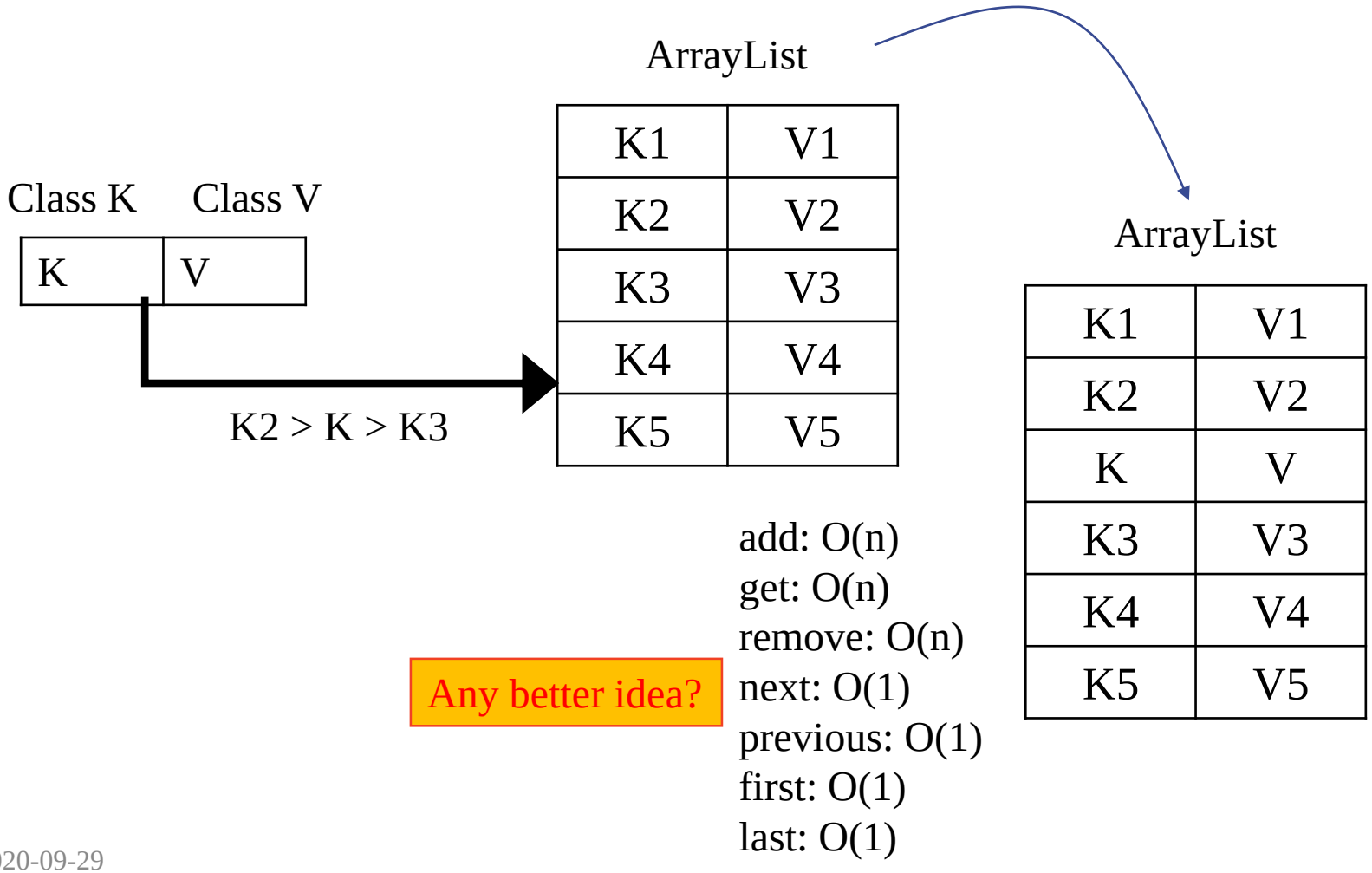


HashMap



Collection Framework(x): ArrayList-based Ordered Map

- How to give an order to key-value pairs?
  - Using ArrayList?

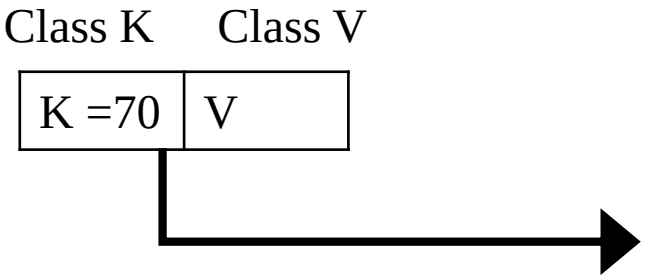


Collection Framework(x): BST-based Ordered Map

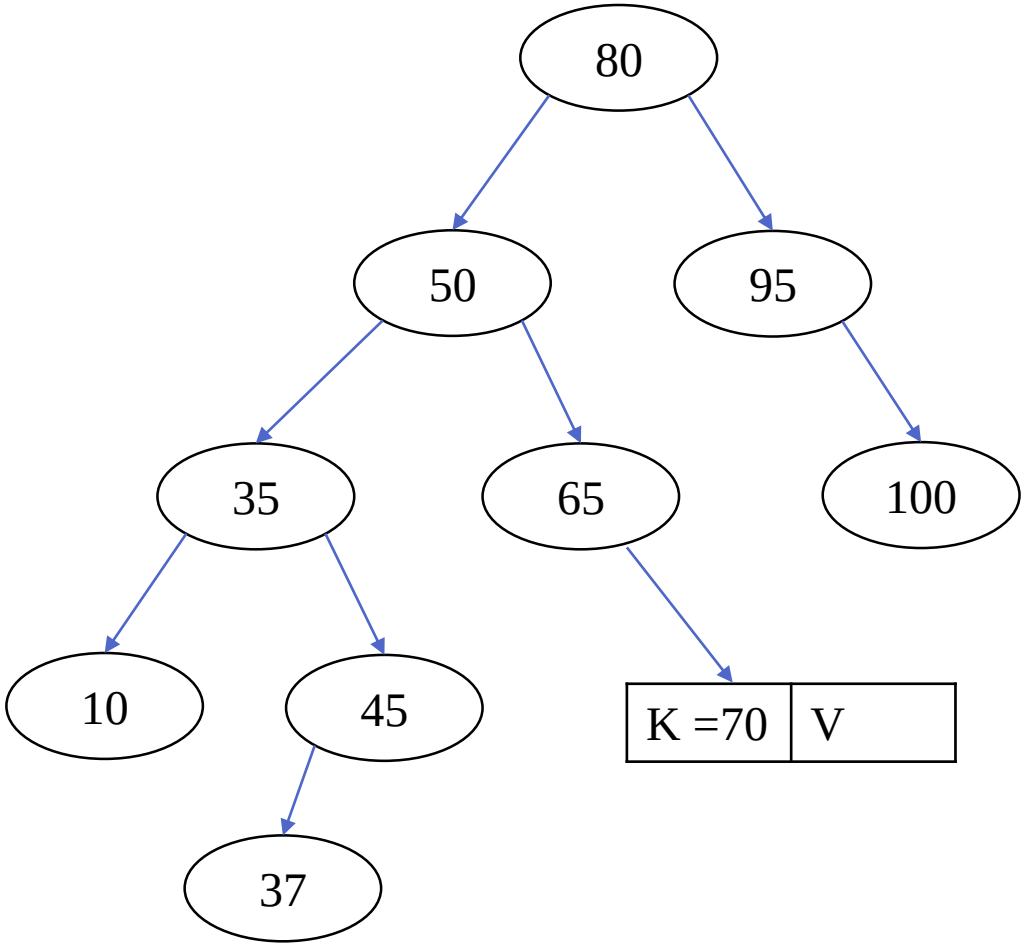
- Binary Search Tree
  - Simple idea
    - Prof. thinks one number from 1 to 50
    - While(true)
      - A student suggests one number
      - If (the number == answer)
        - Break;
      - Else if (the number > answer)
        - System.out.println(“UP”);
      - Else{
        - System.out.println(“DOWN”);

Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?



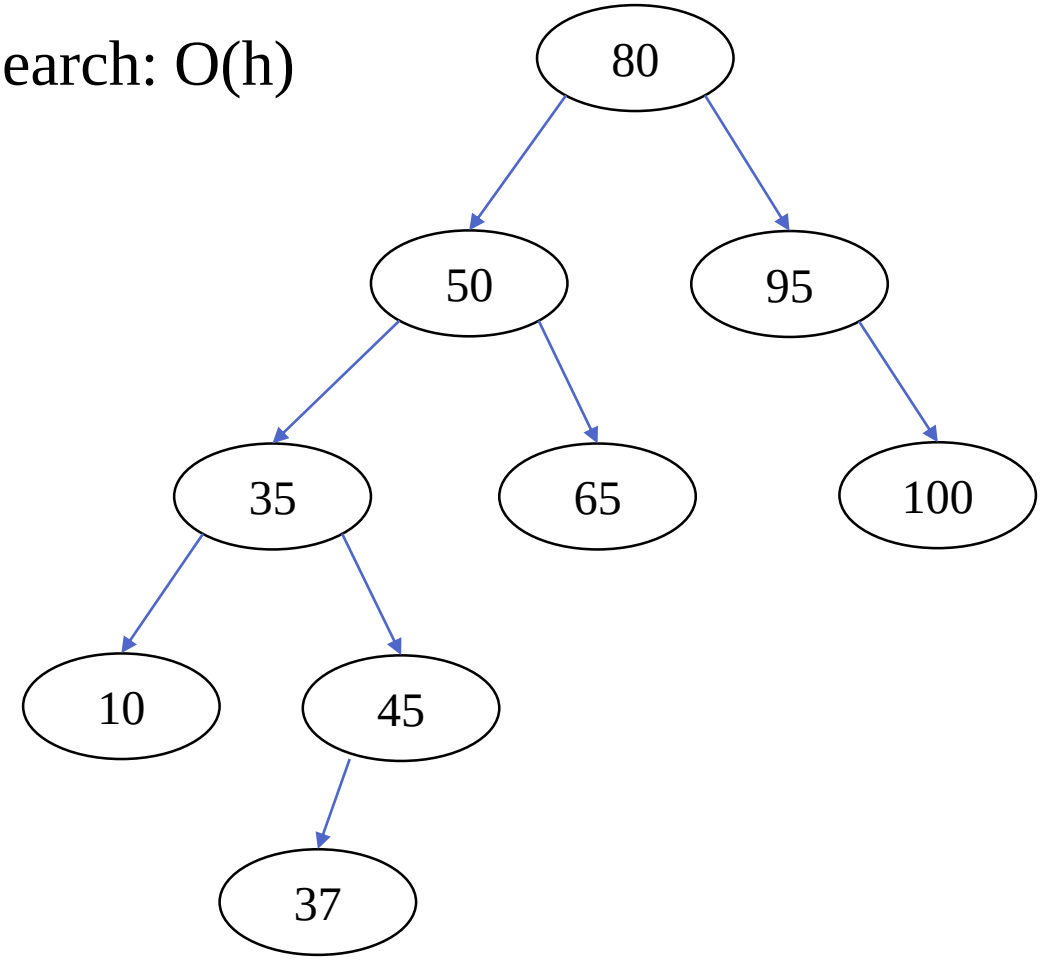
add: O(h)
get: O(h)
remove: O(h)
next: O(h)
previous: O(h)
first: O(h)
last: O(h)



Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

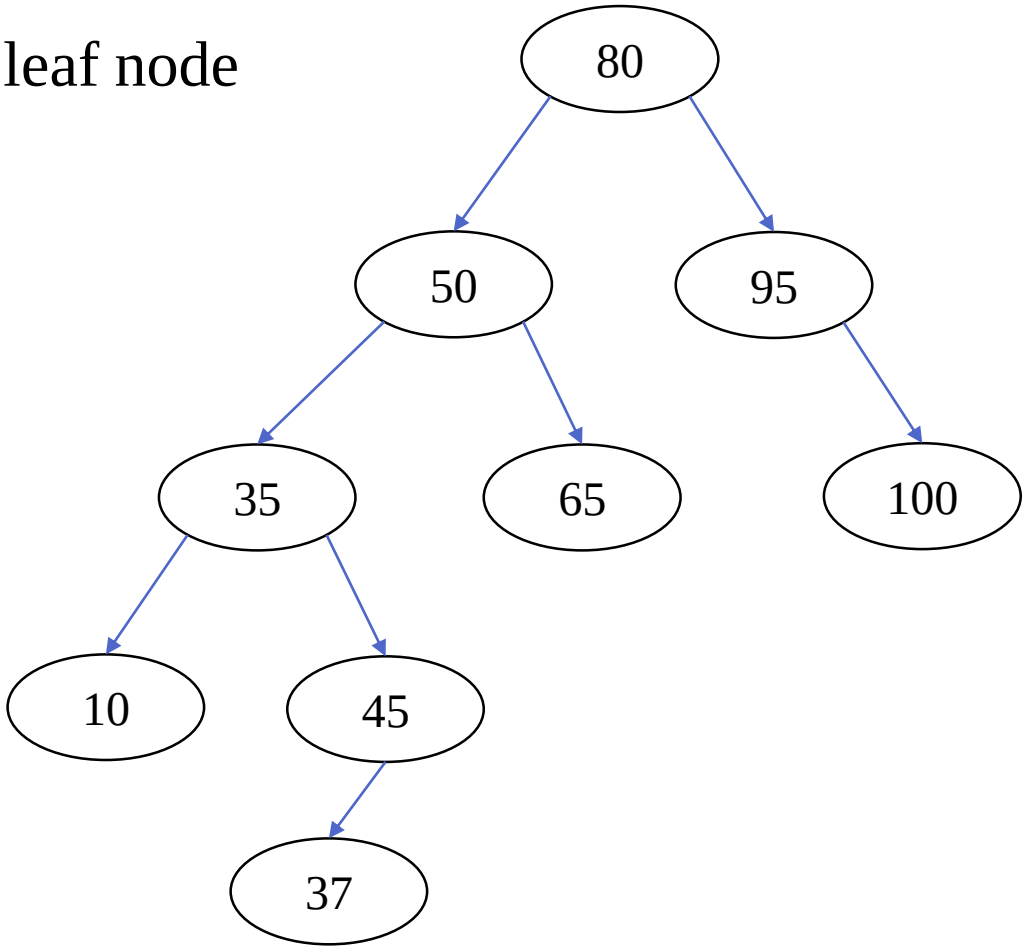
Search: O(h)



Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

Add: O(h)
= Search + add as a leaf node

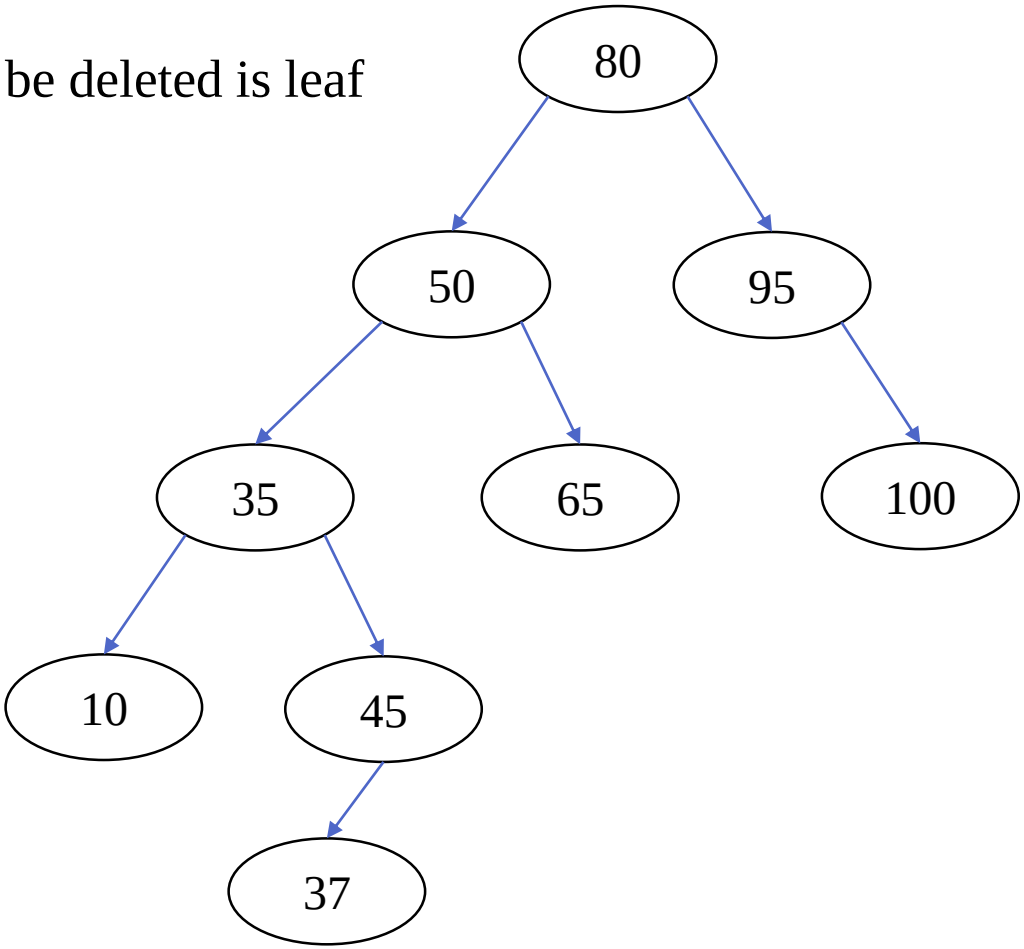


Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

Delete: O(h)

Search and if the key to be deleted is leaf just delete

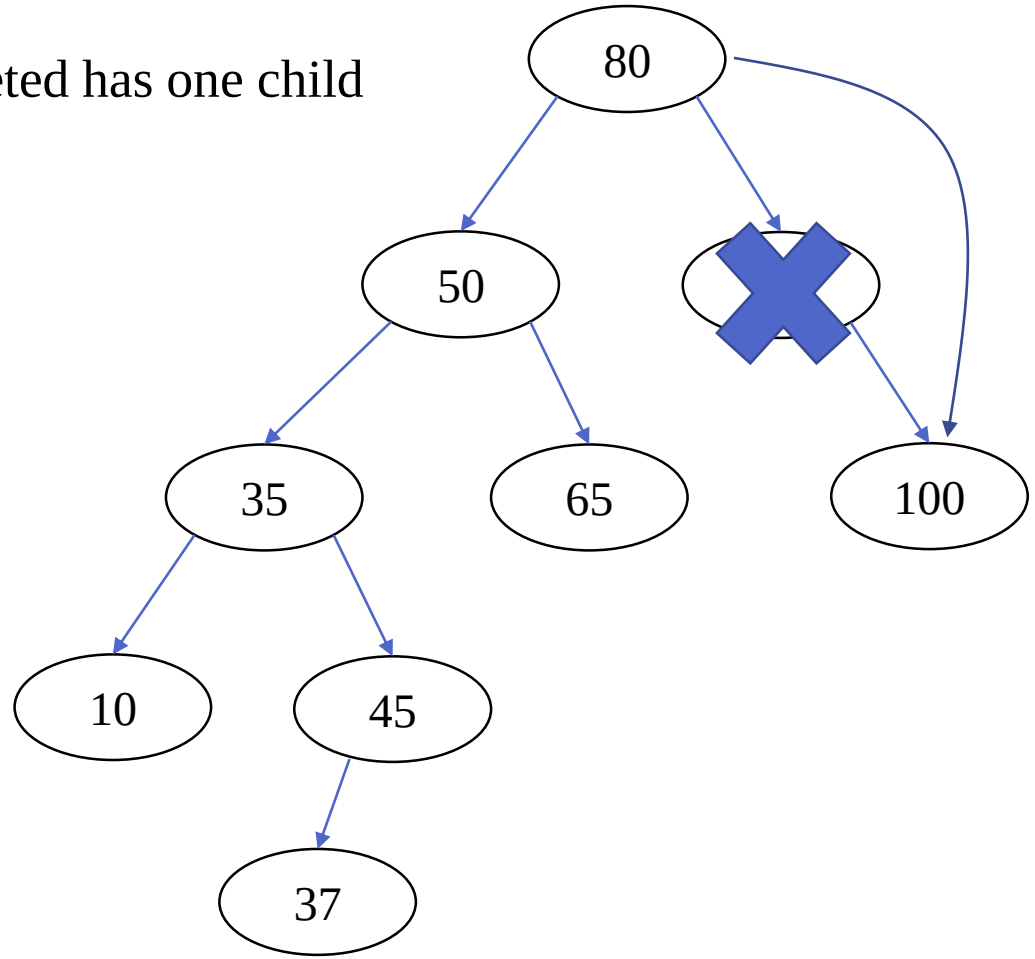


Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

Delete: O(h)

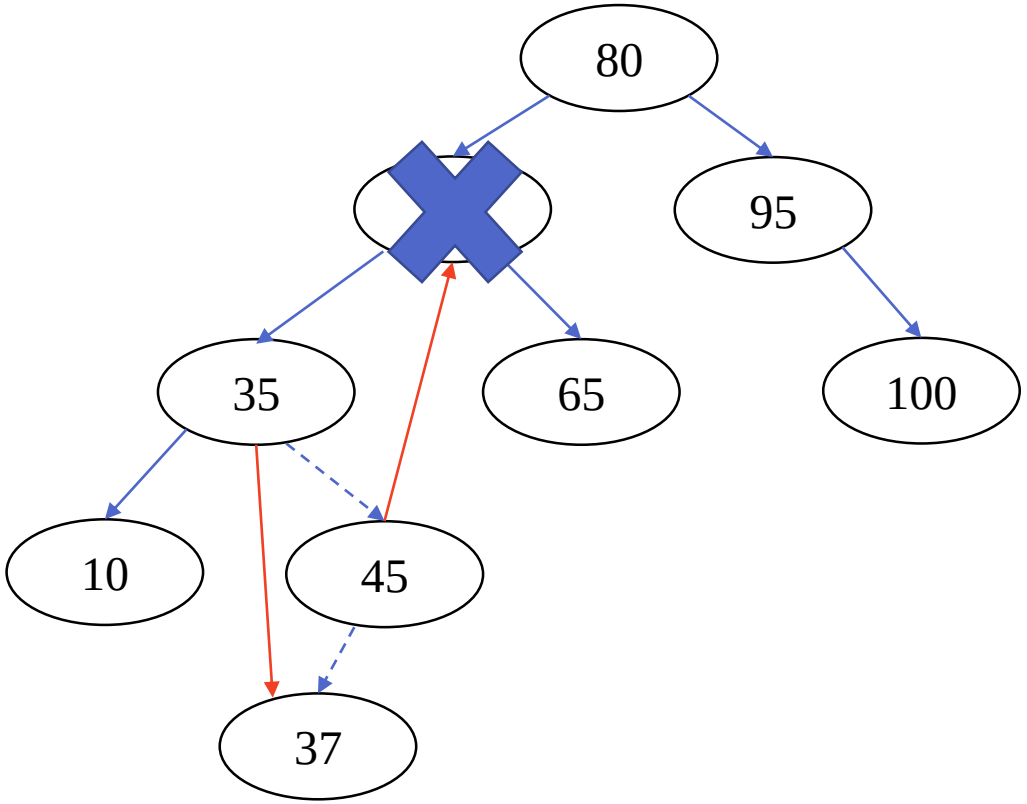
Search and if the key to be deleted has one child link its parent to its child



Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

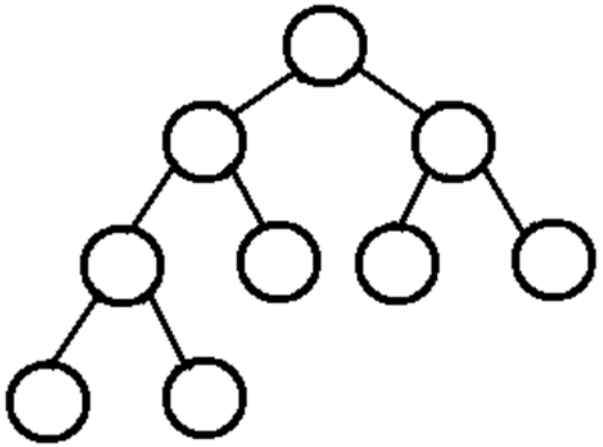
Delete: O(h) , Search and if the key to be deleted has two children
find its floor value and replace the key with the value
if the value has left child, link a parent of the floor value to the left child



Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?

add: O(h)
get: O(h)
remove: O(h)
next: O(h)
previous: O(h)
first: O(h)
last: O(h)



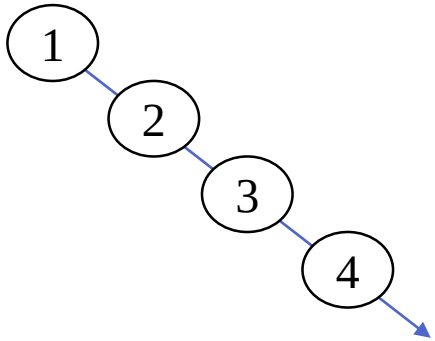
complete tree

- A complete binary tree, completely filled, with the possible exception of the bottom level
- $n = 1 + 2 + 4 + \ldots + 2^{h-1} + 2^h = \Sigma_0^h 2^m = 2^{h+1} - 1$
- $\log n = \log(2^{h+1} - 1)$
- Solving with respect to h, we obtain h = O(logn)

Collection Framework(x): BST-based Ordered Map

- How to give an order to key-value pairs?
  - Using Binary Search Tree (BST)?
    - Add 1 2 3 4 5 6 7 8 …

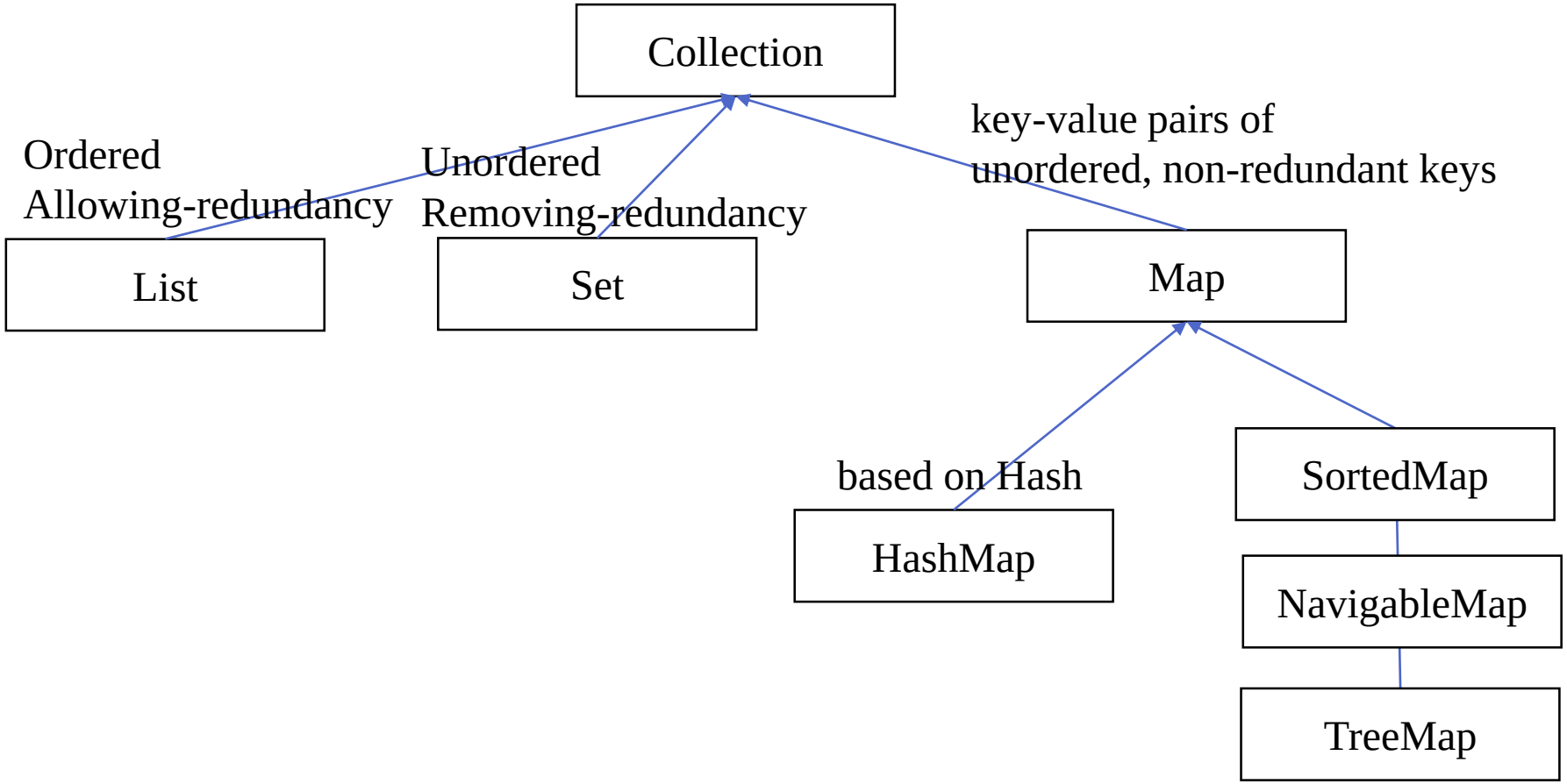
add: O(h)
get: O(h)
remove: O(h)
next: O(h)
previous: O(h)
first: O(h)
last: O(h)



    - This is linked list h = n, the worst case
  - Java TreeMap implements Red-black tree
    - Self-balanced tree: O(logn)

Collection Framework: TreeMap

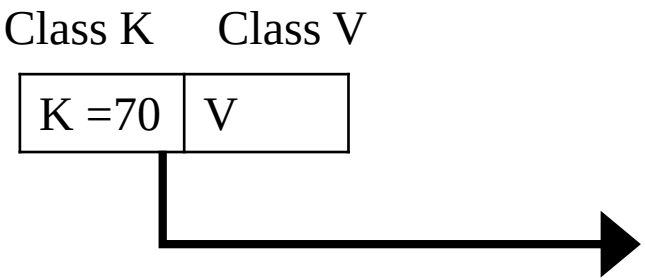
- HashSet consists of unordered, non-redundant elements



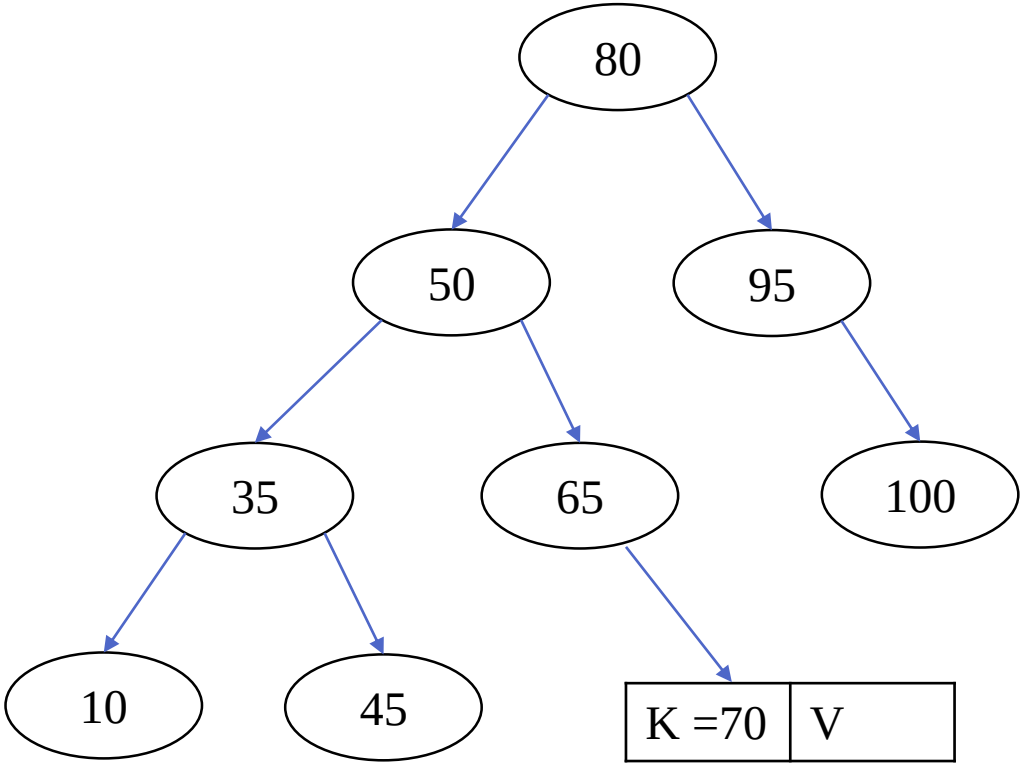
http://www.thecshandbook.com/Hash_Set

Collection Framework: TreeMap

- TreeMap implements Red-black, self-balanced binary search tree



add: O(h)
get: O(h)
remove: O(h)
next: O(h)
previous: O(h)
first: O(h)
last: O(h)



Collection Framework: TreeMap

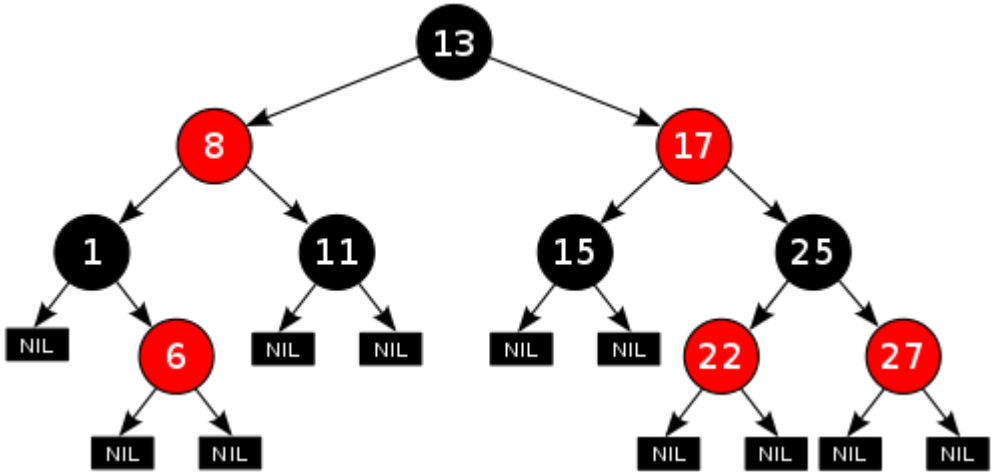
- TreeMap implements Red-black, self-balanced binary search tree

```
public class TreeMap<K,V> extends AbstractMap<K,V> implements NavigableMap<K,V>,
Cloneable, java.io.Serializable
{
    private transient Entry<K,V> root;

        …
}
```

```
        static final class Entry<K,V> implements Map.Entry<K,V> {
                K key;
                V value;
                Entry<K,V> left;
                Entry<K,V> right;
                Entry<K,V> parent;
                boolean color = BLACK;
        }
```

- Node is either black or red node
- Root is black
- Leaf(Nil) is black
- If a node is red, its children are black
- Every path from any node to its descendant nodes should have same number of black



Collection Framework: TreeMap

- TreeMap implements Red-black, self-balanced binary search tree

메서드	설명
TreeMap()	TreeMap객체를 생성한다.
TreeMap(Comparator c)	지정된 Comparator를 기준으로 정렬하는 TreeMap객체를 생성한다.
TreeMap(Map m)	주어진 Map에 저장된 모든 요소를 포함하는 TreeMap을 생성한다.
TreeMap(SortedMap m)	주어진 SortedMap에 저장된 모든 요소를 포함하는 TreeMap을 생성한다.
void clear()	TreeMap에 저장된 모든 객체를 제거한다.
Object clone()	현재 TreeMap을 복제해서 반환한다.
Comparator comparator()	TreeMap을 정렬기준이 되는 Comparator를 반환한다. Comparator가 지정되지 않았다면 null이 반환된다.
boolean containsKey(Object key)	TreeMap에 지정된 키(key)가 포함되어있는지 알려준다.
boolean containsValue(Object value)	TreeMap에 지정된 값(value)가 포함되어있는지 알려준다.
Set entrySet()	TreeMap에 저장된 키와 값을 엔트리(키와 값의 결합)의 형태로 Set에 저장해서 반환한다.
Object firstKey()	TreeMap에 저장된 첫번째 요소의 키를 반환한다.
Object get(Object key)	지정된 키(key)의 값(객체)을 반환한다.
SortedMap headMap(Object toKey)	TreeMap에 저장된 첫번째 요소부터 지정된 요소까지의 범위에 속한 모든 요소가 담긴 SortedMap을 반환한다.(toKey는 포함되지 않는다.)
Set keySet()	TreeMap에 저장된 모든 키가 저장된 Set을 반환한다.
boolean isEmpty()	TreeMap이 비어있는지 알려준다.
Object lastKey()	TreeMap에 저장된 마지막 요소의 키를 반환한다.
Object put(Object key, Object value)	지정된 키와 값을 TreeMap에 저장한다.
void putAll(Map map)	Map에 저장된 모든 요소를 TreeMap에 저장한다.
Object remove(Object key)	TreeMap에서 지정된 키로 저장된 값(객체)를 제거한다.
int size()	TreeMap에 저장된 요소의 개수를 반환한다.
SortedMap subMap(Object fromKey, Object toKey)	지정된 두 개의 키 사이에 있는 요소가 담긴 SortedMap을 반환한다.(toKey는 포함되지 않는다.)
SortedMap tailMap(Object fromKey)	지정된 키부터 마지막 요소의 범위에 속한 요소가 담긴 SortedMap을 반환한다.
Collection values()	TreeMap에 저장된 모든 값을 컬렉션의 형태로 반환한다.

https://dlsdn73.tistory.com/404

Collection Framework: TreeMap

- TreeMap
  - Practice
    - CRUD
    - Generics
    - Iterator
    - hashCode
    - equals

Question #3

- List up the events occurred at the very first time
- List up the events occurred at the very last time
- List up the events occurred at the second time
- List up the events occurred at the second last time

Question #4

- Get the time when the number of events valid at that time is maximum
- Get the number of events occurred at Monday
- Get the number of events occurred at 1st day of a month
- Get the number of events occurred at a specific year

Summary

- Collecting, modelling, and analyses based on Tree-based collection

- Next Week
  - Collecting, modelling, and analyses based on Tree-based collection (cont.)