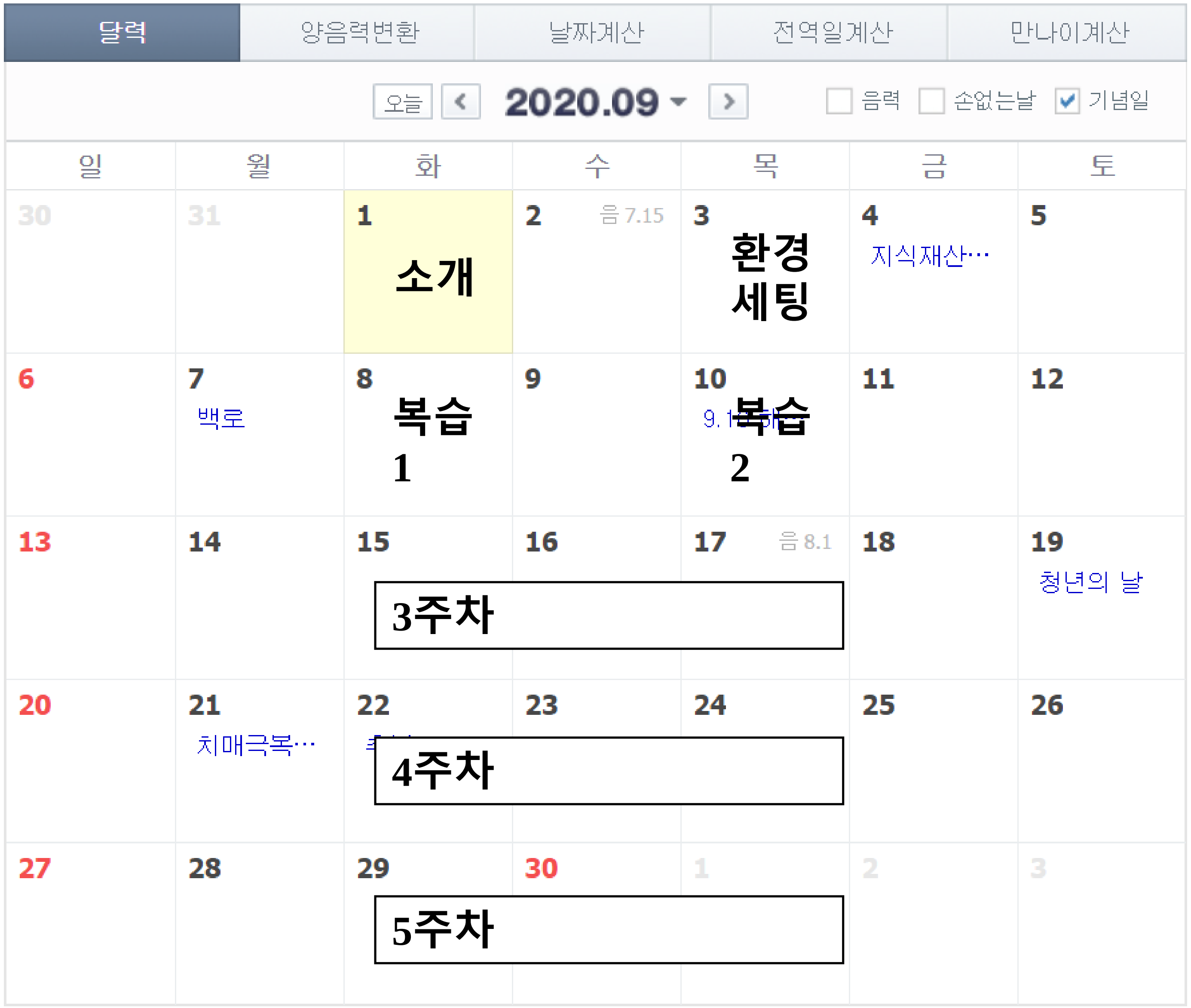
# Data Analysis

(Data modelling, collecting, and analyses 3)

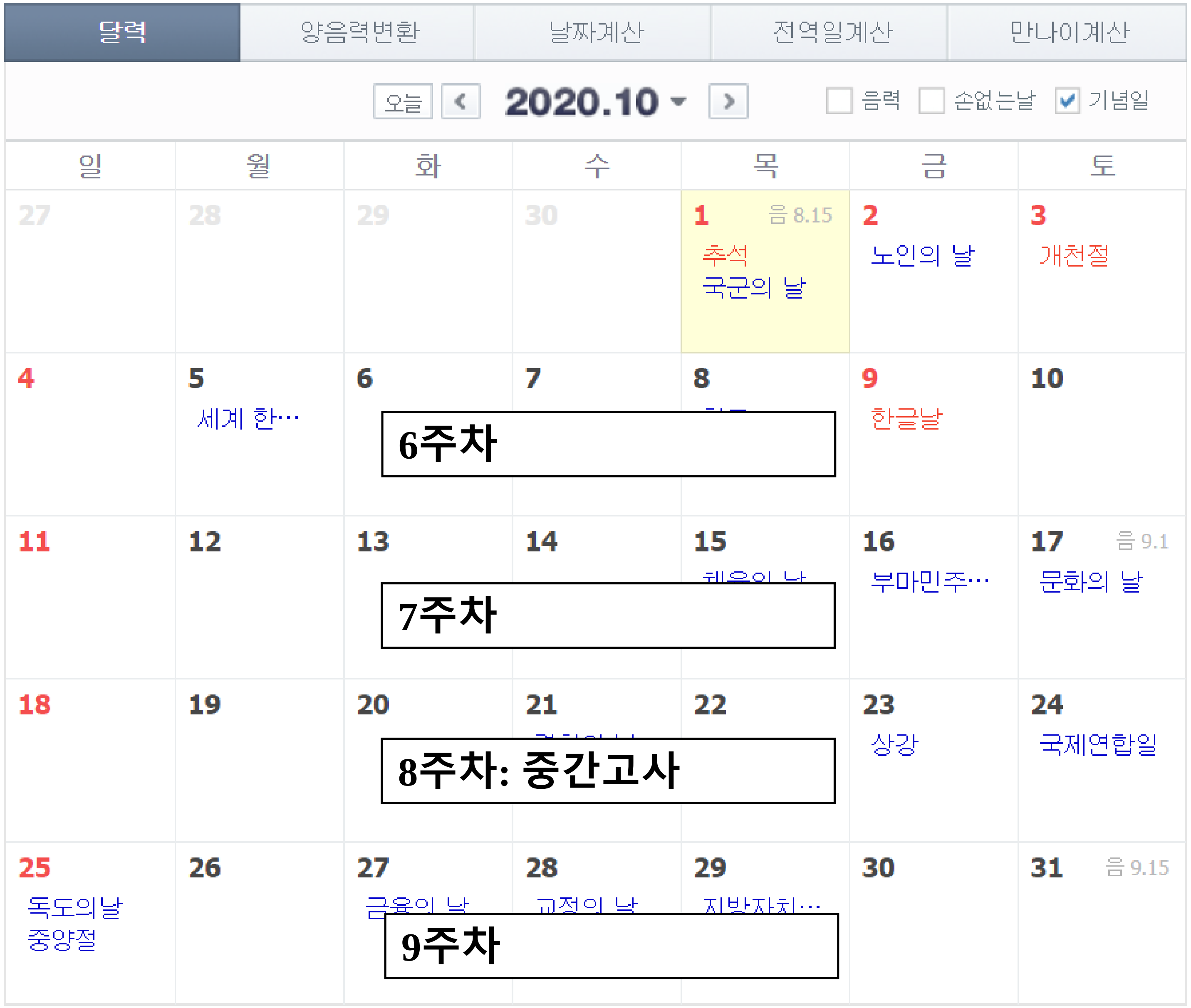
**Fall, 2020**

# Calendar



| 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[illegible] 복습 2 | 11 | 12 |
| 13 | 14 | 15 3주차 | 16 | 17 음 8.1 | 18 | 19 청년의 날 |
| 20 | 21 치매극복… | 22 [illegible] 4주차 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 5주차 | 30 | 1 | 2 | 3 |

# Calendar



| 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 한로 | 9 한글날 | 10 |
| 11 | 12 | 13 | 14 | 15 체육의 날 | 16 부마민주… | 17 음 9.1 문화의 날 |
| 18 | 19 | 20 | 21 경찰의 날 | 22 | 23 상강 | 24 국제연합일 |
| 25 독도의날 중양절 | 26 | 27 금융의 날 | 28 교정의 날 | 29 지방자치… | 30 | 31 음 9.15 |

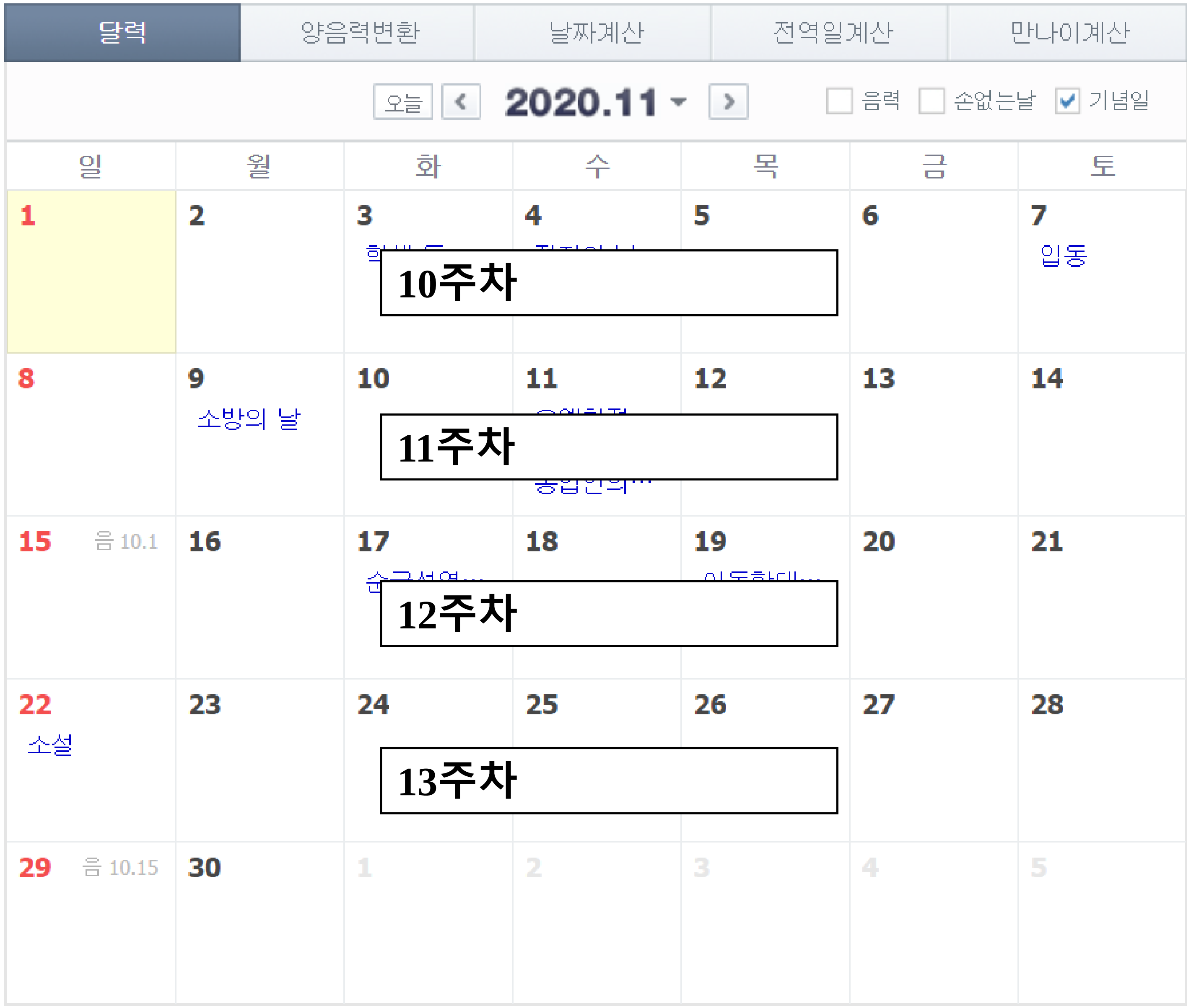
**6주차**

**7주차**

**8주차: 중간고사**

**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 |
| 8 | 9 소방의 날 | 10 | 11 농업인의··· | 12 | 13 | 14 |
| 15 음 10.1 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 소설 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 음 10.15 | 30 | 1 | 2 | 3 | 4 | 5 |

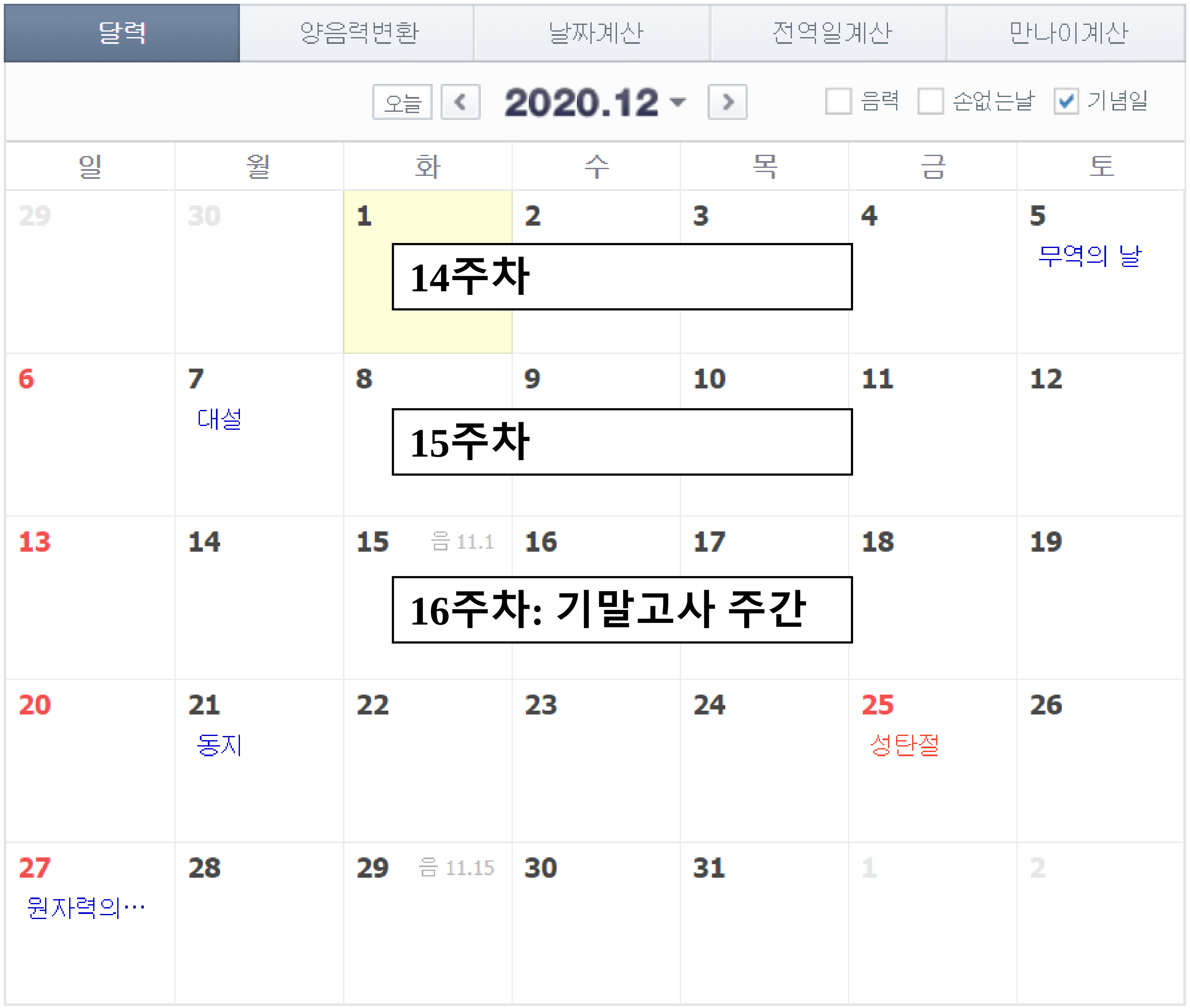
10주차

11주차

12주차

13주차

# Calendar

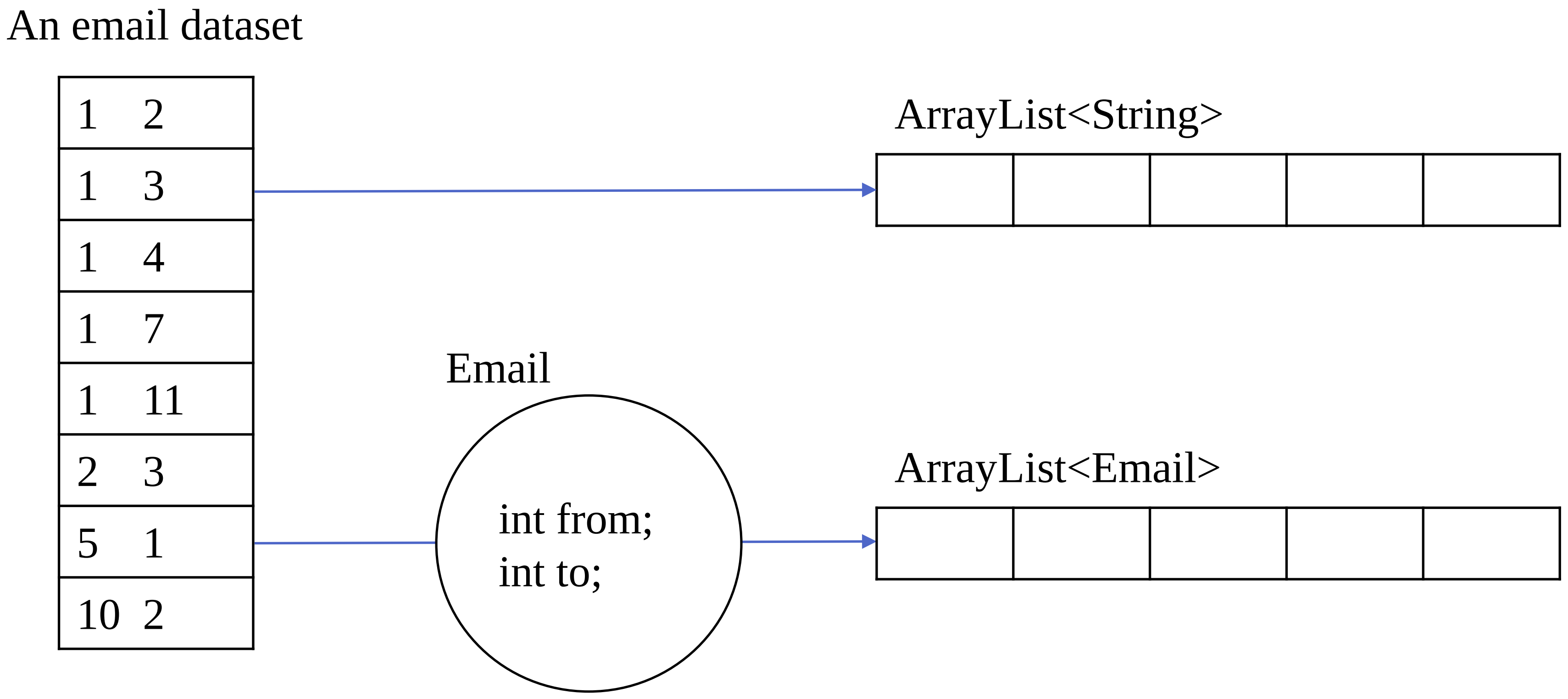


| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| 29 | 30 | 1 **14주차** | 2 | 3 | 4 | 5 무역의 날 |
| 6 | 7 대설 | 8 **15주차** | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 음 11.1 **16주차: 기말고사 주간** | 16 | 17 | 18 | 19 |
| 20 | 21 동지 | 22 | 23 | 24 | 25 성탄절 | 26 |
| 27 원자력의… | 28 | 29 음 11.15 | 30 | 31 | 1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

## Table of Contents

# Modelling

- Modelling a class for abstracting a dataset
  - Increasing the accessibility of the concept
  - Practice: get the maximum, minimum identifier



# Question #0

- Keep left identifiers in ArrayList with redundancy

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Remaining:** **1** 2 3 1 4 2 | | | | | | | | | |
| 1 | | | | | | | | | |
| **Remaining:** 2 3 1 4 2 | | | | | | | | | |
| 1 | 2 | | | | | | | | |
| **Remaining:** 3 1 4 2 | | | | | | | | | |
| 1 | 2 | 3 | | | | | | | |
| **Remaining:** 1 4 2 | | | | | | | | | |
| 1 | 2 | 3 | | | | | | | |
| **Remaining:** 4 2 | | | | | | | | | |
| 1 | 2 | 3 | 4 | | | | | | |
| **Remaining:** 2 | | | | | | | | | |
| 1 | 2 | 3 | 4 | | | | | | |

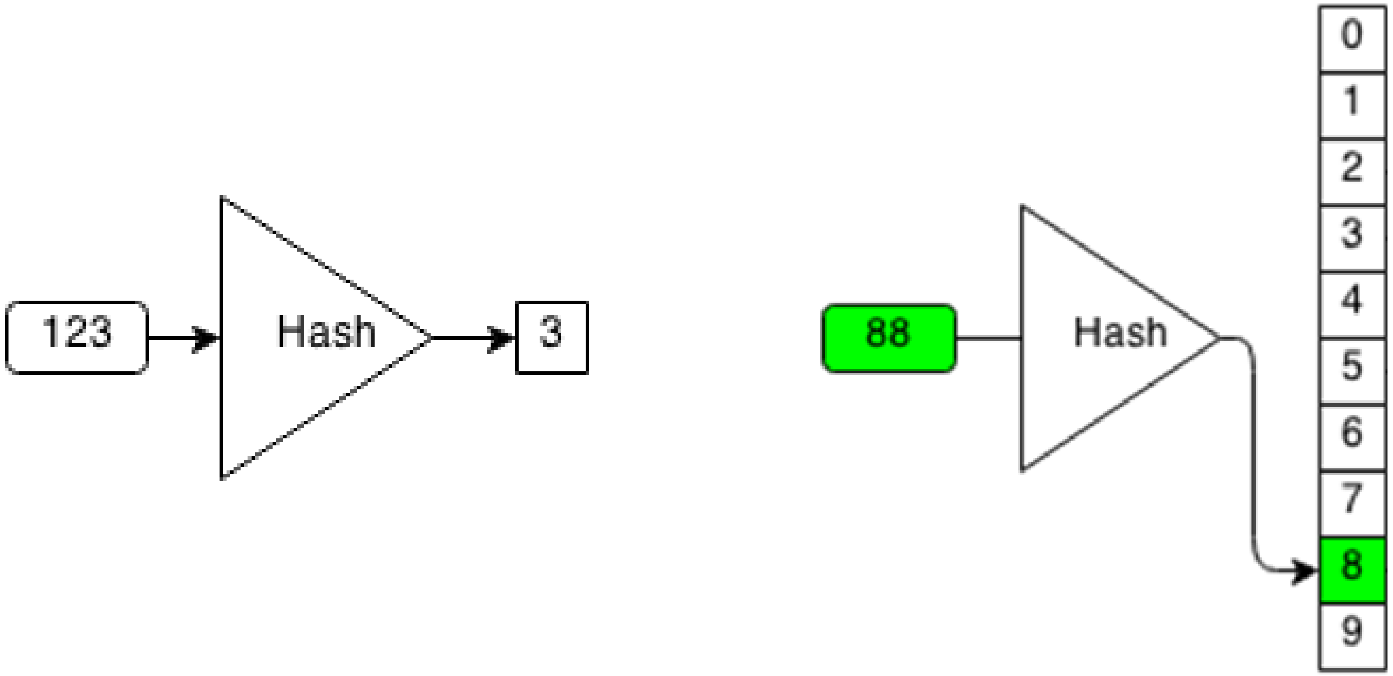
## Question #0

- Keep left identifiers in ArrayList without redundancy
  - See how the trend of the computation time for each line

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Remaining: 1** 2 3 1 4 2 | | | | | | | | | |
| 1 | | | | | | | | | |
| **Remaining:** 2 3 1 4 2 | | | | | | | | | |
| 1 | 2 | | | | | | | | |
| **Remaining:** 3 1 4 2 | | | | | | | | | |
| 1 | 2 | 3 | | | | | | | |
| **Remaining:** 1 4 2 | | | | | | | | | |
| 1 | 2 | 3 | | | | | | | |
| **Remaining:** 4 2 | | | | | | | | | |
| 1 | 2 | 3 | 4 | | | | | | |
| **Remaining:** 2 | | | | | | | | | |
| 1 | 2 | 3 | 4 | | | | | | |

# HashSet

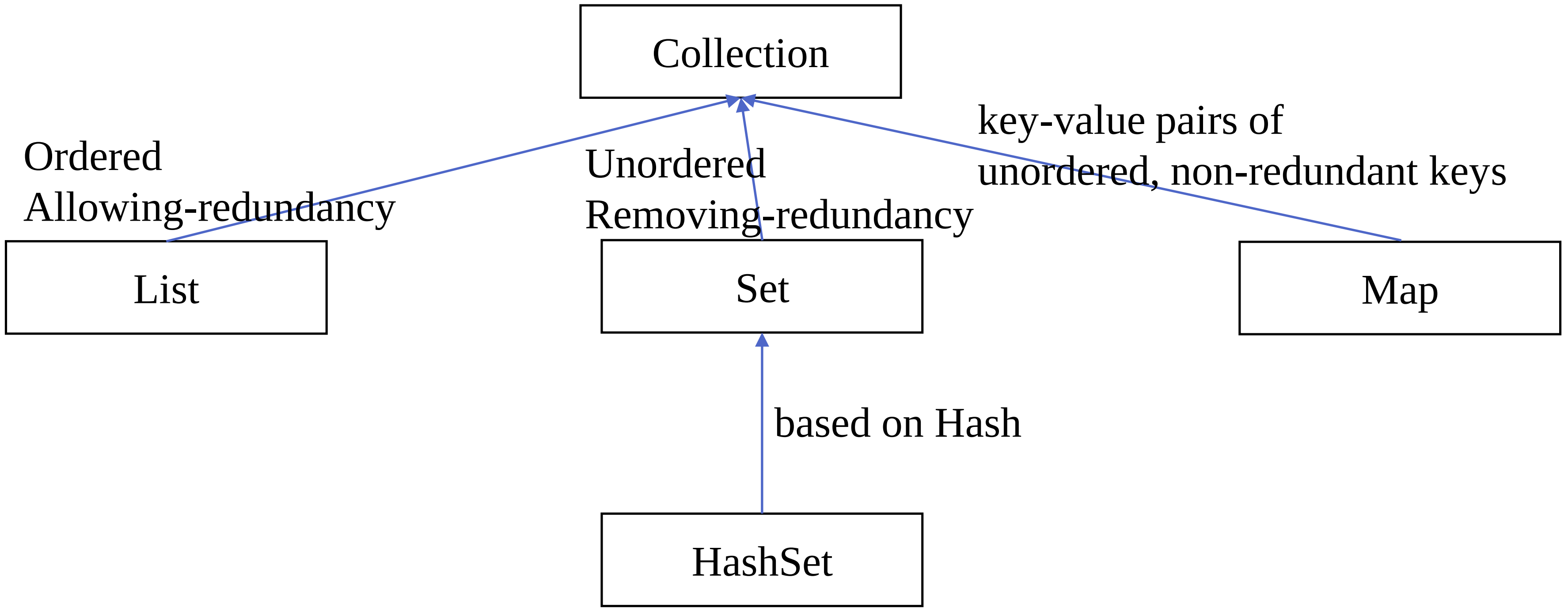
- HashSet consists of non-redundant elements
- Rethink the importance of 'Data Structure' and 'Algorithms'
- We can find an element of HashSet  O(1)



http://www.thecshandbook.com/Hash_Set

## Collection Framework: HashSet

- HashSet consists of unordered, non-redundant elements



http://www.thecshandbook.com/Hash_Set

# Collection Framework: HashSet

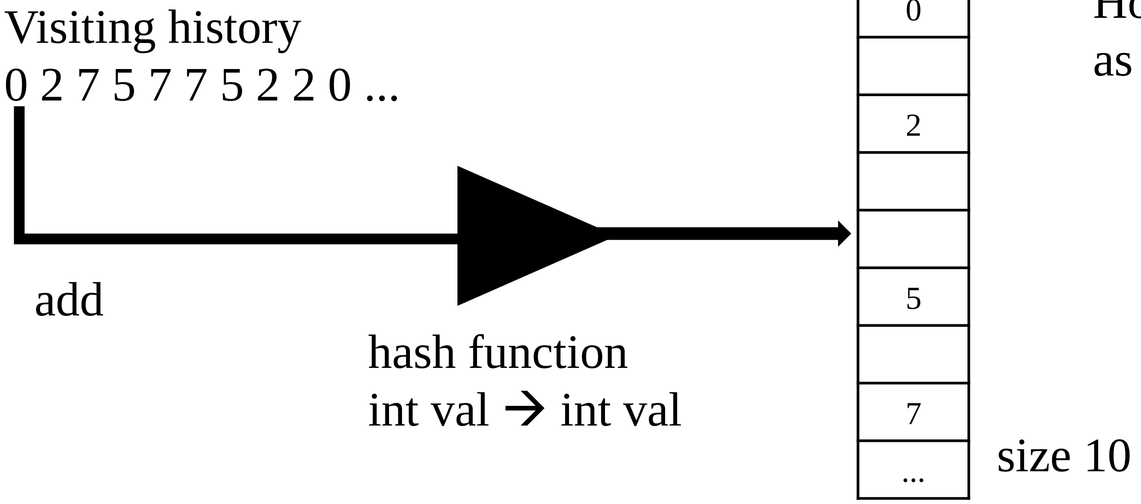
- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection

If there is a visiting history,
keeping its non-redundant visited people
by using ArrayList

| | | | |
|---|---|---|---|
| 0 | 2 | 7 | 5 |

A group of 10 people
each person has an ID from 0 to 9

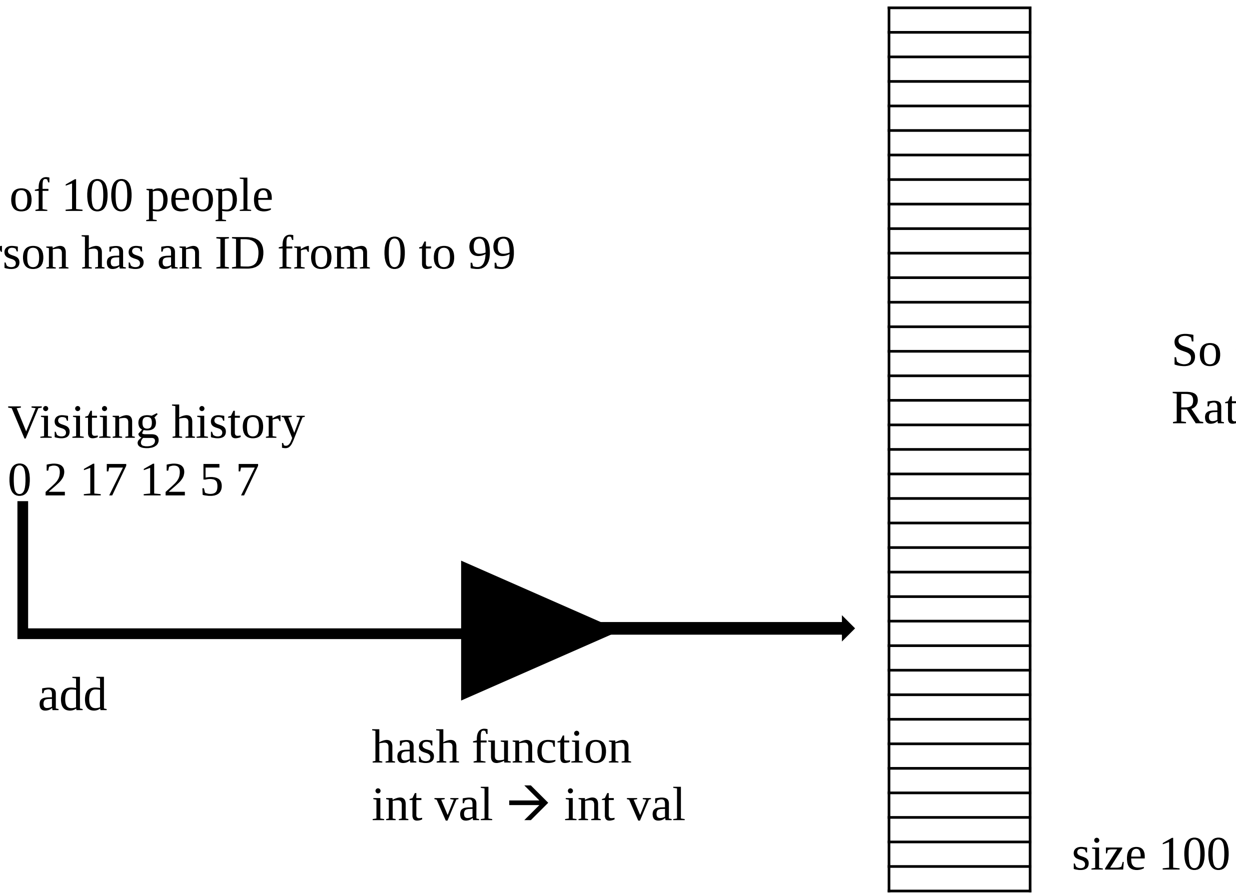
How about using the ID
as an index of array?



## Collection Framework: HashSet

- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection

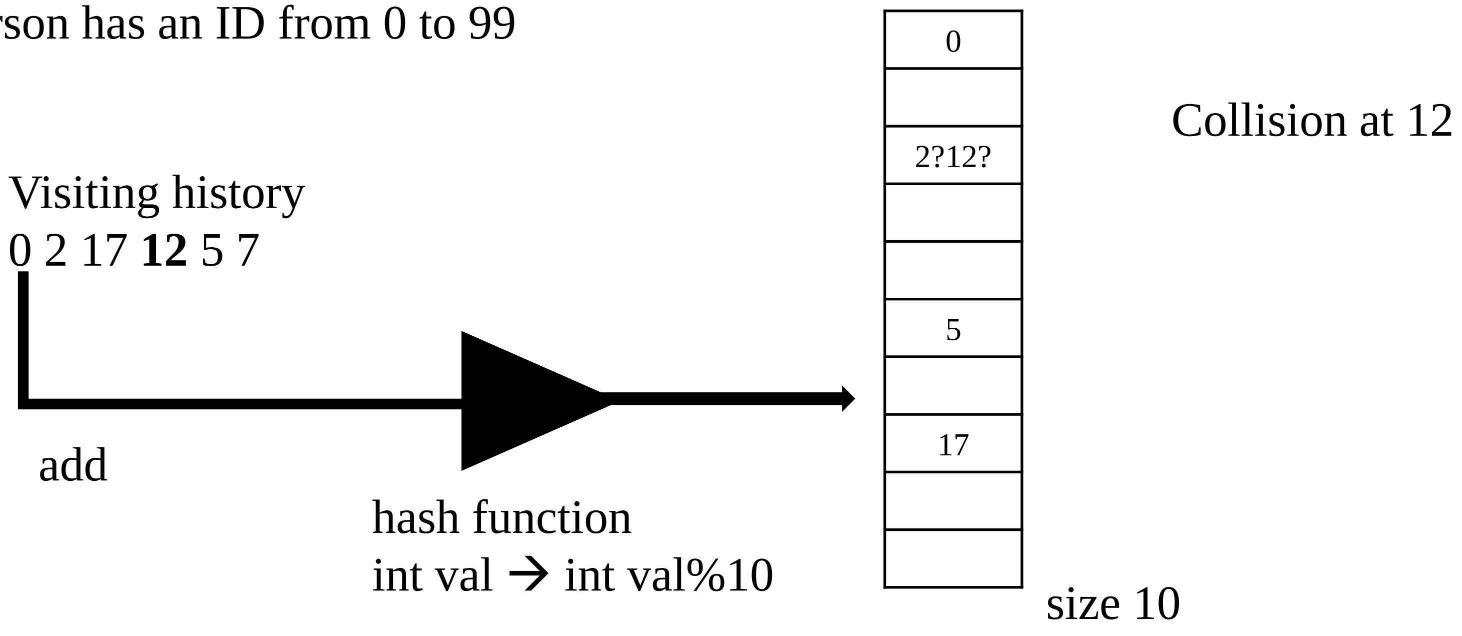
A group of 100 people
each person has an ID from 0 to 99



# Collection Framework: HashSet

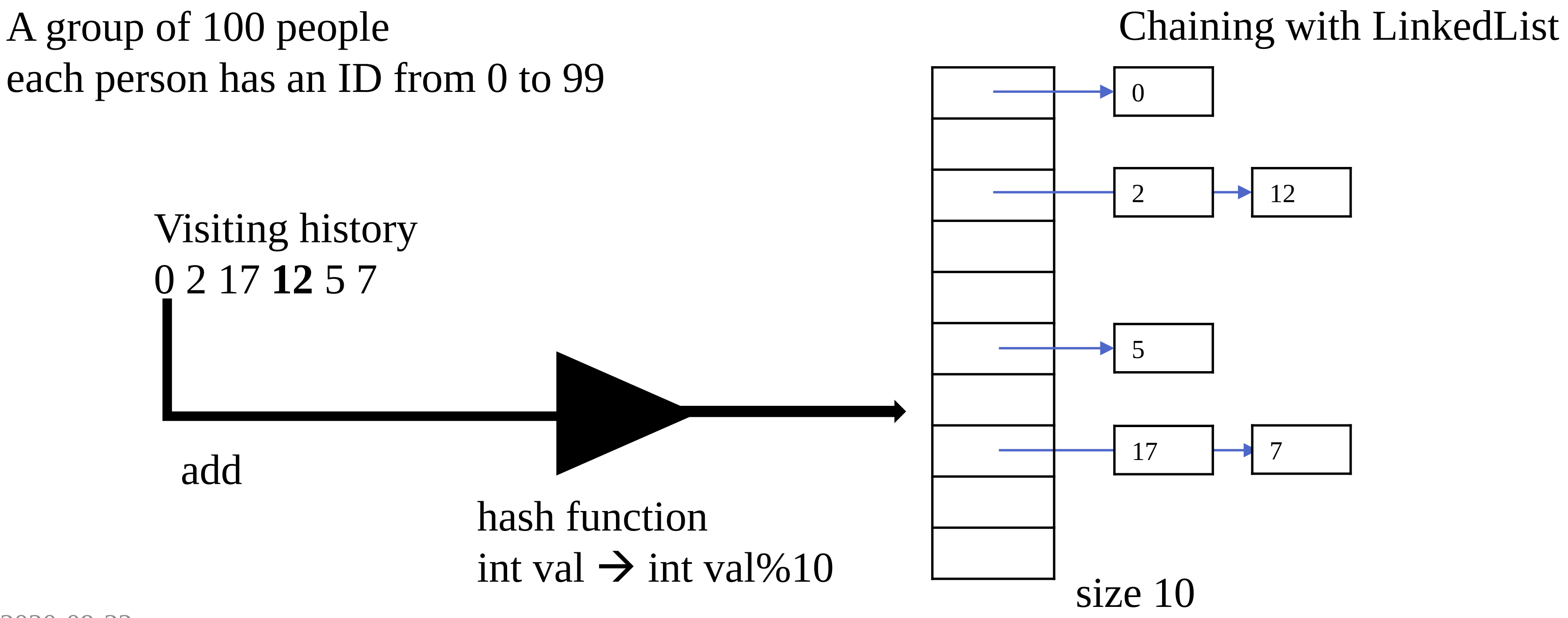
- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection

A group of 100 people
each person has an ID from 0 to 99



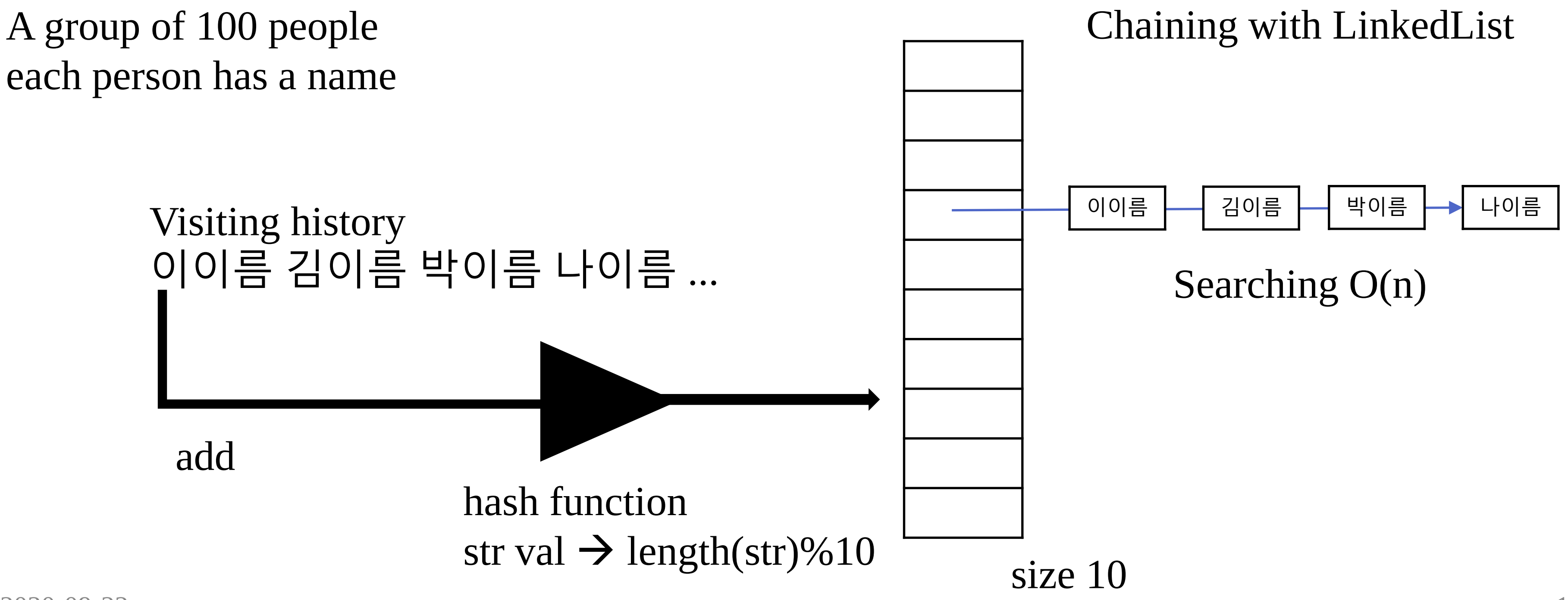
# Collection Framework: HashSet

- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection



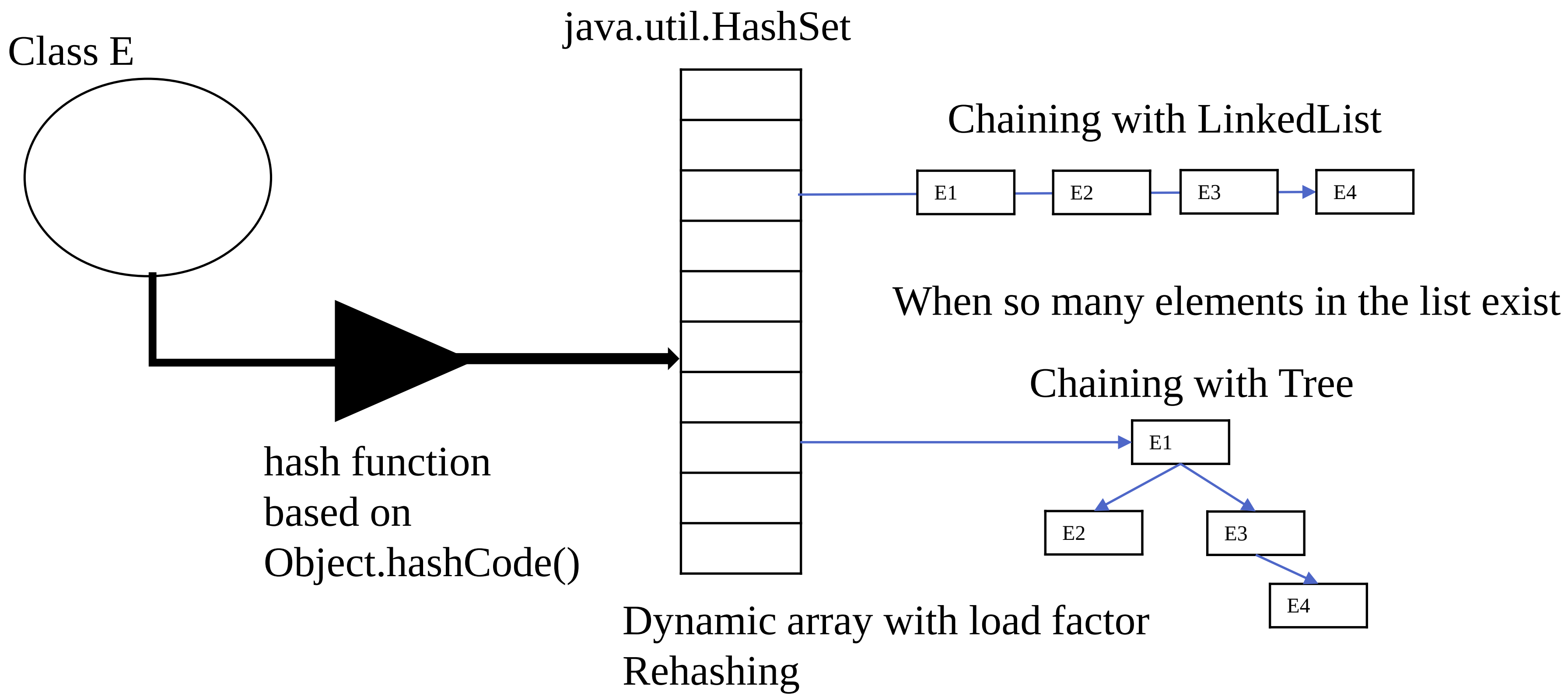
# Collection Framework: HashSet

- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection



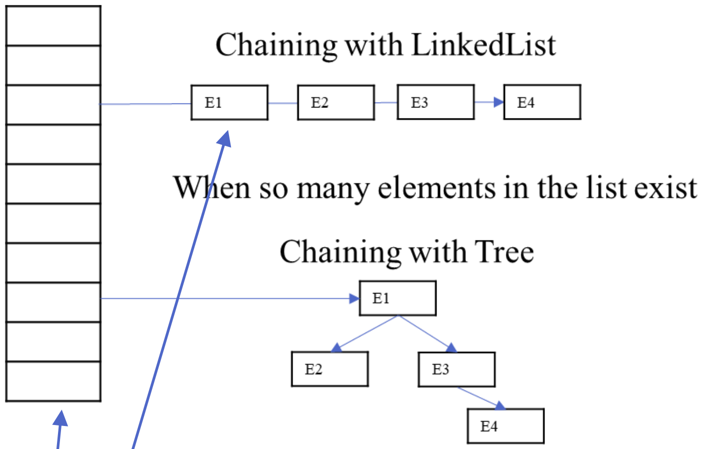
# Collection Framework: HashSet

- HashSet consists of unordered, non-redundant elements
- Basic idea
  - There is a function (element → integer ), called hash.
  - The result of hash function will be used for index of collection



# Collection Framework: HashSet

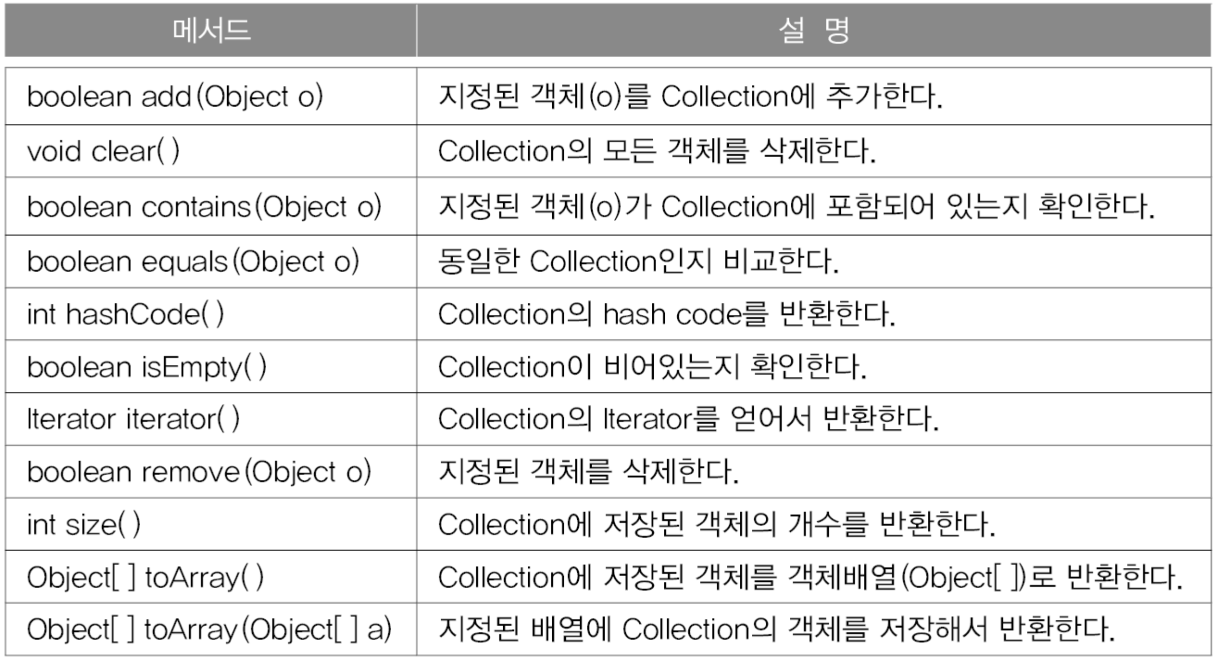
- HashSet
  - Unordered, Non-redundant elements E
  - Based on Hashing
  - CRUD by hash



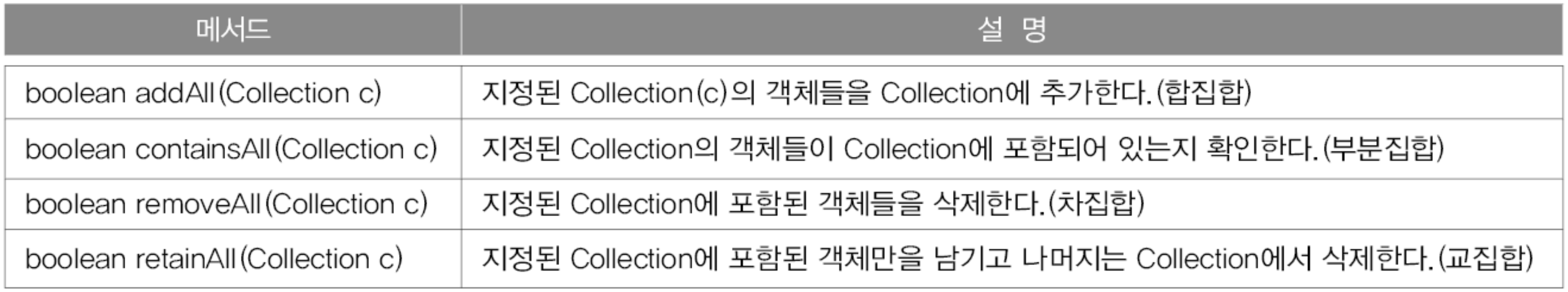
```
public class HashSet<E> extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
        private HashMap<E,Object> map;
                        transient Node<K,V>[] table;
                         static class Node<K,V> implements Map.Entry<K,V>
                        {
                                final int hash;
                                final K key;
                                V value;
                                Node<K,V> next;
                                   …
                        }
}
```

# Collection Framework: HashSet

- HashSet
  - implementing Set



| 메서드 | 설 명 |
|---|---|
| boolean add(Object o) | 지정된 객체(o)를 Collection에 추가한다. |
| void clear( ) | Collection의 모든 객체를 삭제한다. |
| boolean contains(Object o) | 지정된 객체(o)가 Collection에 포함되어 있는지 확인한다. |
| boolean equals(Object o) | 동일한 Collection인지 비교한다. |
| int hashCode( ) | Collection의 hash code를 반환한다. |
| boolean isEmpty( ) | Collection이 비어있는지 확인한다. |
| Iterator iterator( ) | Collection의 Iterator를 얻어서 반환한다. |
| boolean remove(Object o) | 지정된 객체를 삭제한다. |
| int size( ) | Collection에 저장된 객체의 개수를 반환한다. |
| Object[ ] toArray( ) | Collection에 저장된 객체를 객체배열(Object[ ])로 반환한다. |
| Object[ ] toArray(Object[ ] a) | 지정된 배열에 Collection의 객체를 저장해서 반환한다. |



| 메서드 | 설 명 |
|---|---|
| boolean addAll(Collection c) | 지정된 Collection(c)의 객체들을 Collection에 추가한다.(합집합) |
| boolean containsAll(Collection c) | 지정된 Collection의 객체들이 Collection에 포함되어 있는지 확인한다.(부분집합) |
| boolean removeAll(Collection c) | 지정된 Collection에 포함된 객체들을 삭제한다.(차집합) |
| boolean retainAll(Collection c) | 지정된 Collection에 포함된 객체만을 남기고 나머지는 Collection에서 삭제한다.(교집합) |

## Collection Framework: HashSet

- HashSet
    - Practice HashSet
        - CRUD
        - Generics
        - Iterator
        - hashCode
        - equals

## Question #1

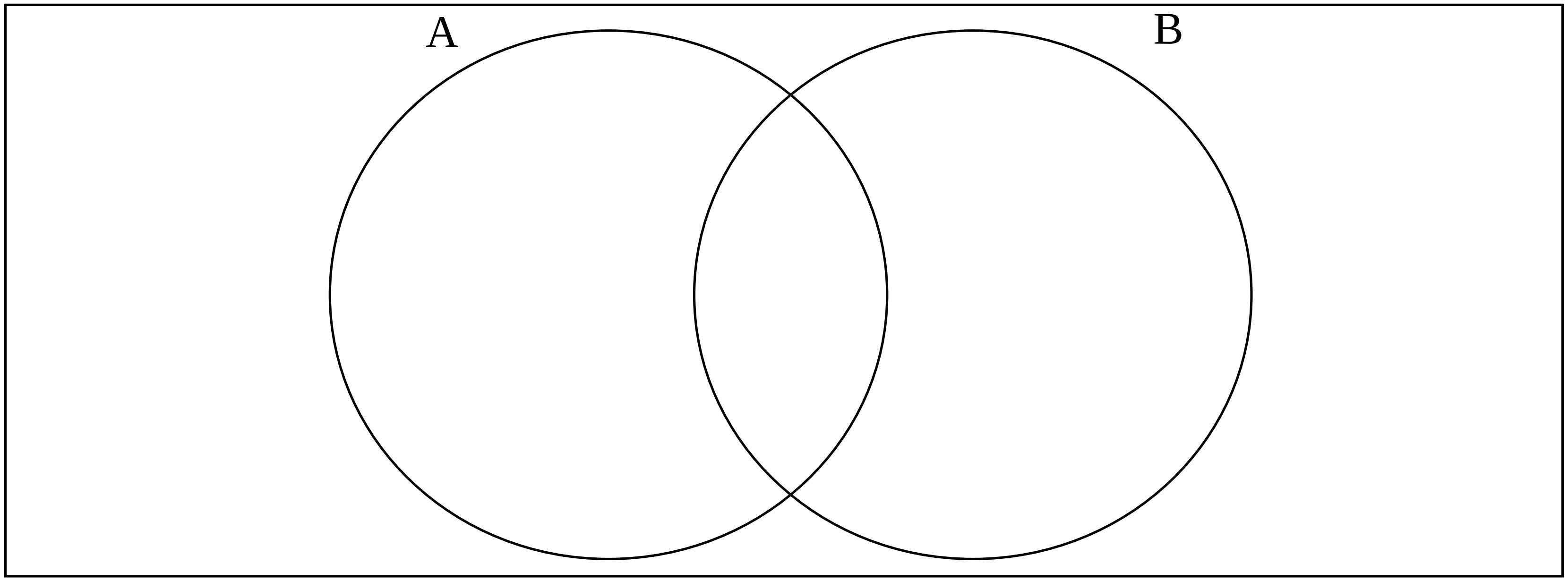
- Compute the number of identifiers (The redundancy is **not** allowed)
    - See how the trend of the computation time for computing each line

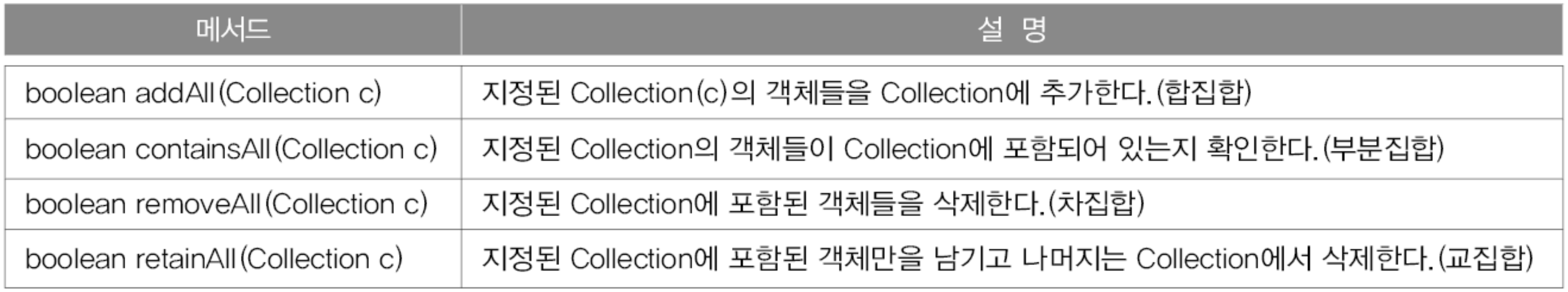
## Question #2

- Compute the number of non-redundant identifiers who only sending email(s)
- Compute the number of non-redundant identifiers who only receiving email(s)
- Compute the number of non-redundant identifiers who both sending and receiving email(s)
- Compute the number of non-redundant identifiers who attends an email network

# Collection Framework: HashSet

- HashSet





| 메서드 | 설 명 |
|---|---|
| boolean addAll(Collection c) | 지정된 Collection(c)의 객체들을 Collection에 추가한다.(합집합) |
| boolean containsAll(Collection c) | 지정된 Collection의 객체들이 Collection에 포함되어 있는지 확인한다.(부분집합) |
| boolean removeAll(Collection c) | 지정된 Collection에 포함된 객체들을 삭제한다.(차집합) |
| boolean retainAll(Collection c) | 지정된 Collection에 포함된 객체만을 남기고 나머지는 Collection에서 삭제한다.(교집합) |

## In-class assignment 2

- Implement your own MyHashSet implementing Set<E>
  - Implementing all the methods
  - Data Abstraction (example)
    - MyBucket<E>[]
  - The maximum size of hash set never changes for you since initialization
  - Solve Question #1 and #2 with your collection

```java
public class MyHashSet<E> implements Set<E>{

private MyBucket<E>[] bucketChain = null;

@SuppressWarnings("unchecked")
public MyHashSet(int capacity) {
bucketChain = new MyBucket[capacity];
}

@SuppressWarnings("rawtypes")
@Override
public int size() {
int cnt = 0;
for(MyBucket b: bucketChain) {
if(b != null)
cnt++;
}
return cnt;
}
```

```java
public class MyBucket<E> {
int hashCode;
ArrayList<E> bucketList;
}
```

## Question #3

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

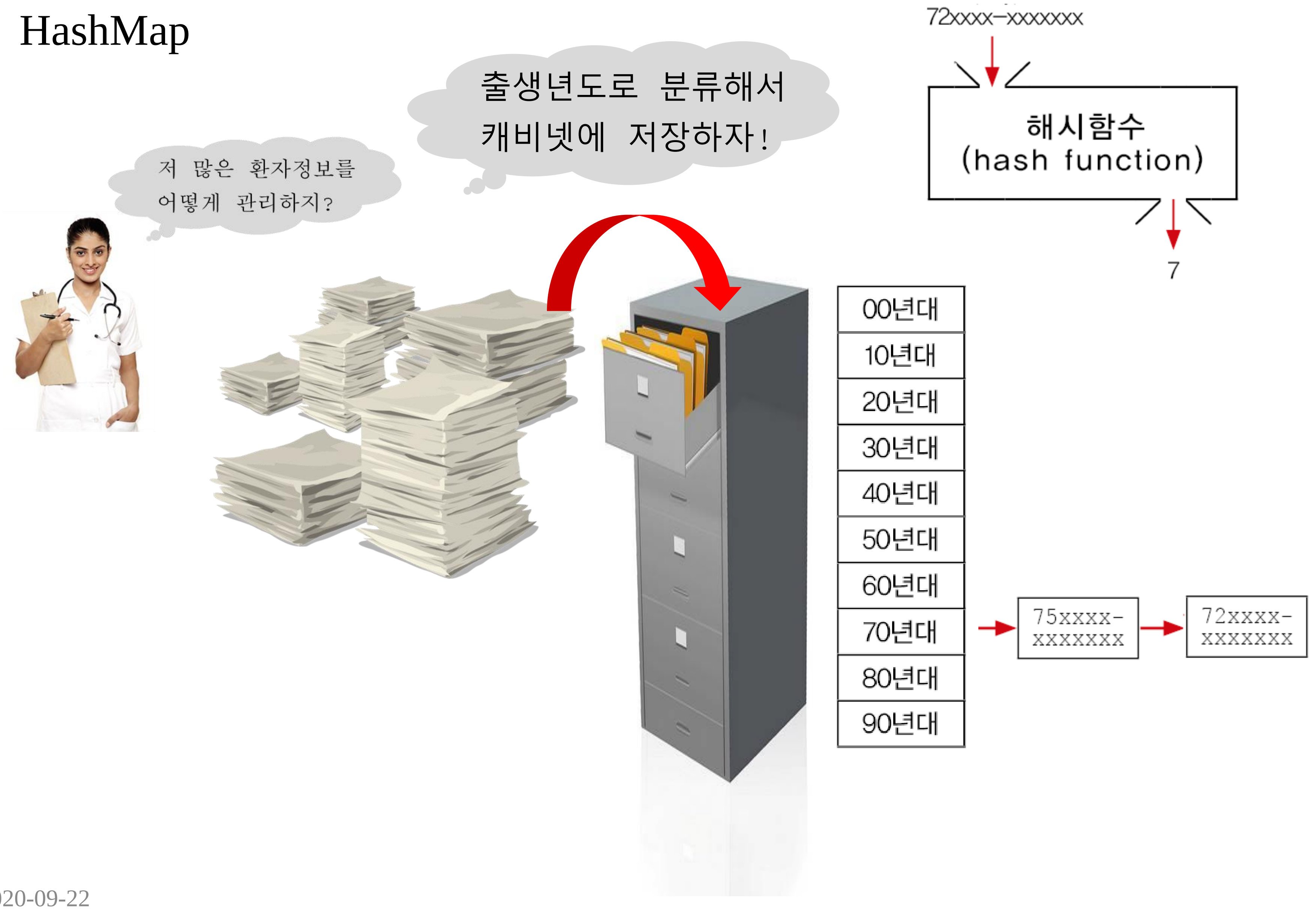
## Collection Framework: HashMap

- 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



## Summary

- Some Practice!
- Collecting, modelling, and analyses based on Hash-based collection

- Next Week
  - Collecting, modelling, and analyses based on Hash-based collection (Cont.)