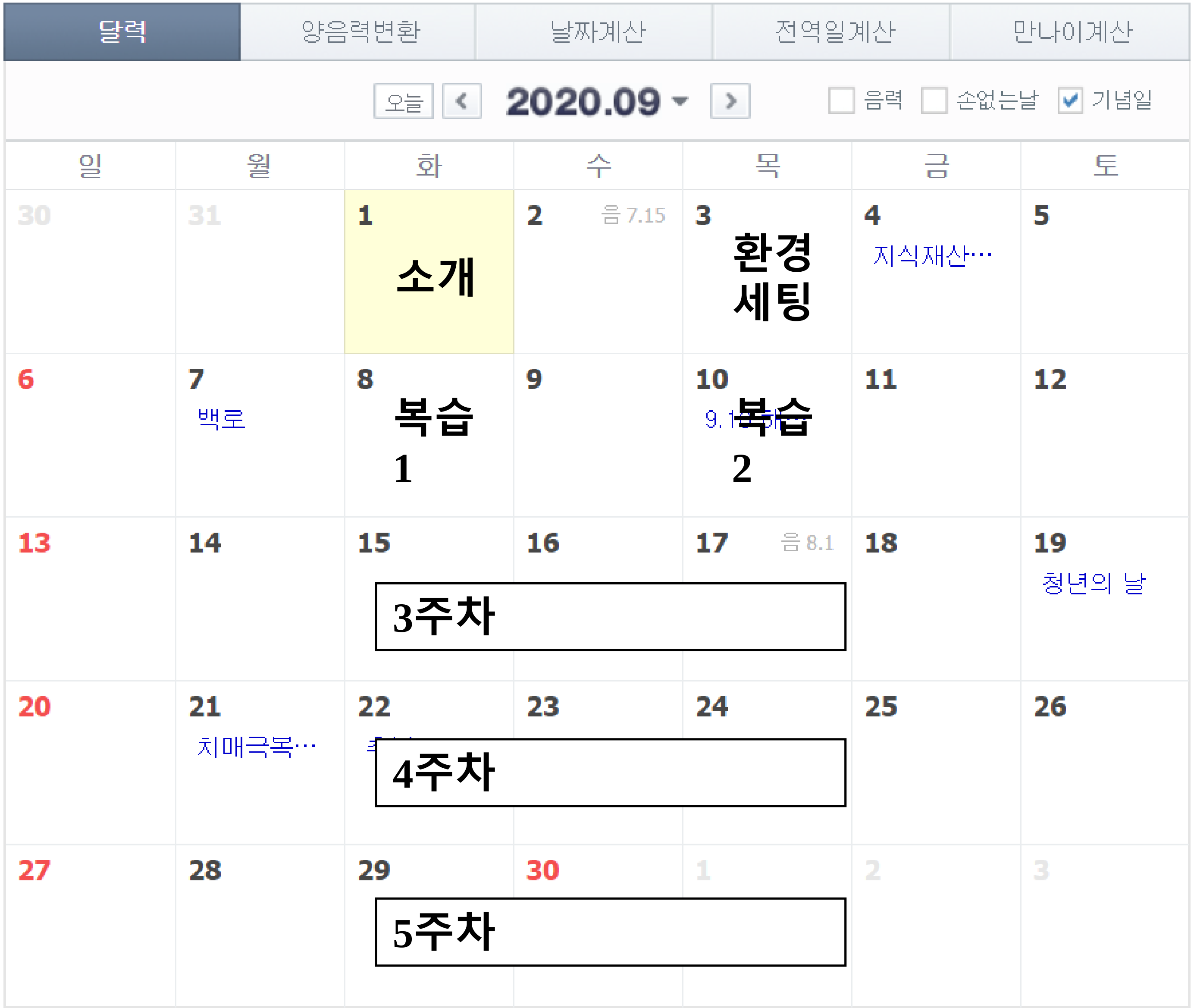
# Data Analysis

(Data Analysis Overview)

**Fall, 2020**

# Calendar

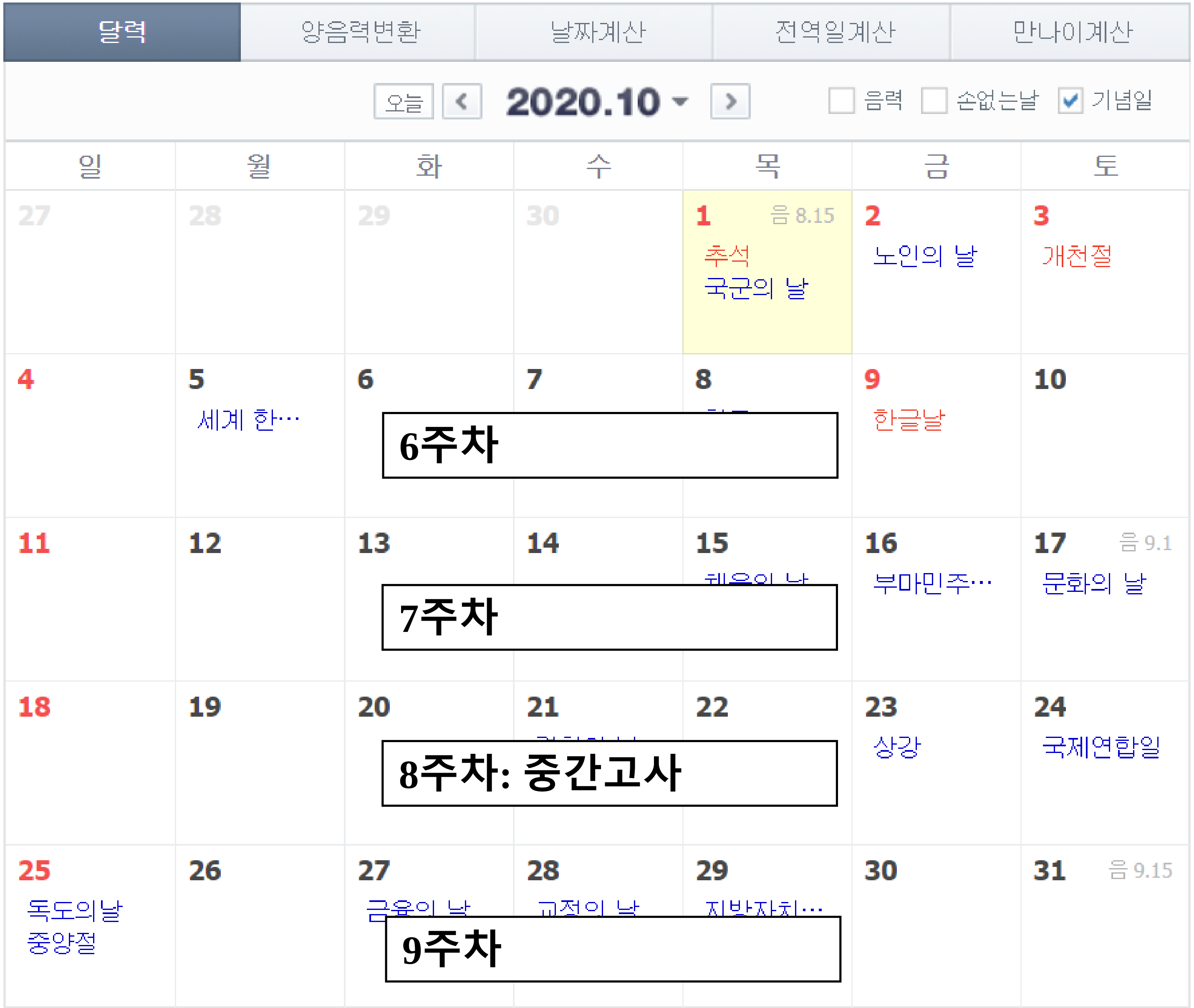


달력 | 양음력변환 | 날짜계산 | 전역일계산 | 만나이계산

오늘 ‹ 2020.09 ▾ › □ 음력 □ 손없는날 ☑ 기념일

| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| 30 | 31 | 1<br>소개 | 2 음 7.15 | 3<br>환경<br>세팅 | 4<br>지식재산… | 5 |
| 6 | 7<br>백로 | 8<br>복습<br>1 | 9 | 10<br>9. 1[illegible]<br>복습<br>2 | 11 | 12 |
| 13 | 14 | 15<br>3주차 | 16 | 17 음 8.1 | 18 | 19<br>청년의 날 |
| 20 | 21<br>치매극복… | 22<br>4주차 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29<br>5주차 | 30 | 1 | 2 | 3 |

# Calendar



| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| 27 | 28 | 29 | 30 | 1 음 8.15<br>추석<br>국군의 날 | 2<br>노인의 날 | 3<br>개천절 |
| 4 | 5<br>세계 한… | 6 | 7 | 8<br>한로 | 9<br>한글날 | 10 |
| 11 | 12 | 13 | 14 | 15<br>체육의 날 | 16<br>부마민주… | 17 음 9.1<br>문화의 날 |
| 18 | 19 | 20 | 21<br>경찰의 날 | 22 | 23<br>상강 | 24<br>국제연합일 |
| 25<br>독도의날<br>중양절 | 26 | 27<br>금융의 날 | 28<br>교정의 날 | 29<br>지방자치… | 30 | 31 음 9.15 |

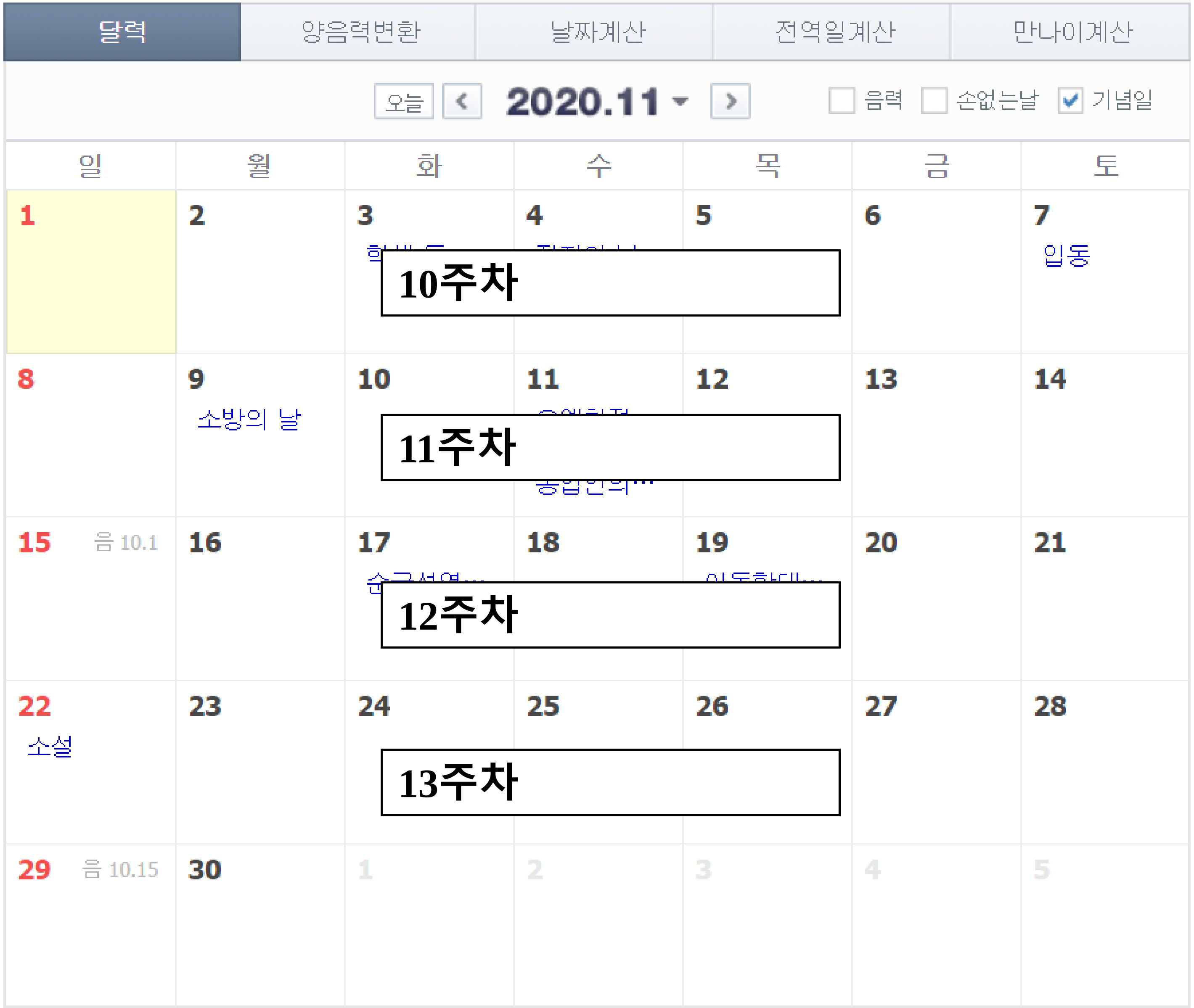
6주차

7주차

8주차: 중간고사

9주차

# Calendar



오늘 ‹ 2020.11 ▾ ›

□ 음력 □ 손없는날 ☑ 기념일

| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3<br>[illegible] | 4<br>[illegible] | 5 | 6 | 7<br>입동 |
| 8 | 9<br>소방의 날 | 10 | 11<br>[illegible]<br>농업인의… | 12 | 13 | 14 |
| 15 음 10.1 | 16 | 17<br>[illegible] | 18 | 19<br>[illegible] | 20 | 21 |
| 22<br>소설 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 음 10.15 | 30 | 1 | 2 | 3 | 4 | 5 |

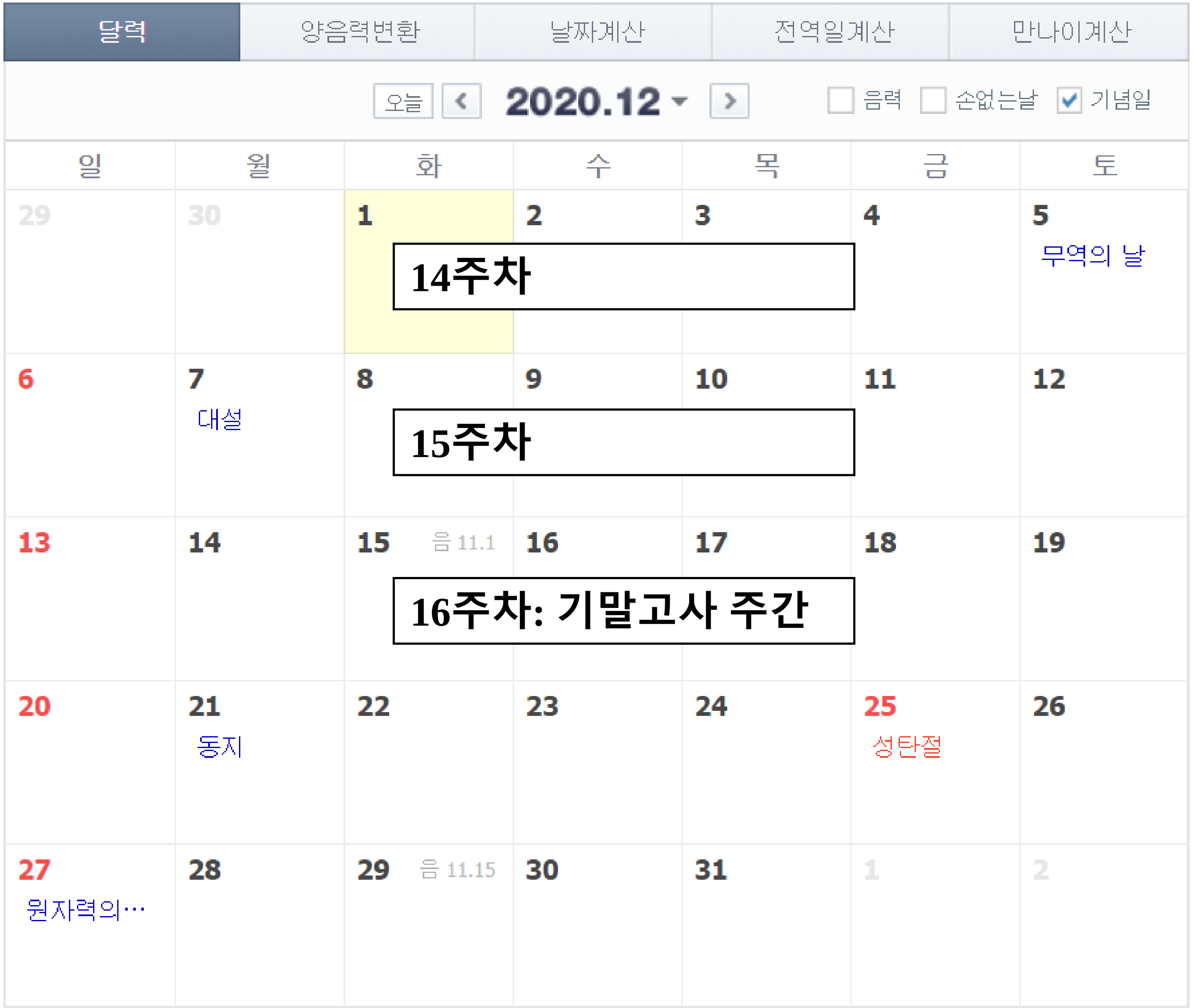
10주차

11주차

12주차

13주차

# Calendar



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

14주차

15주차

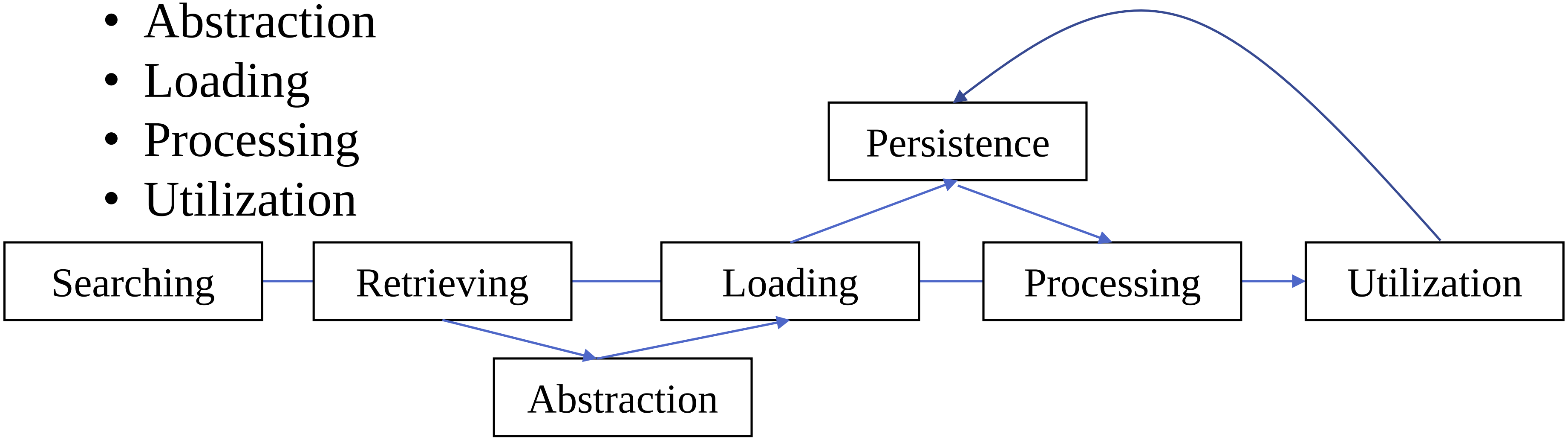
16주차: 기말고사 주간

# Table of Contents

- Overview of Data Analysis
- First Dataset
- File Input/Output

# Overview of Data Analysis

- Wiki says
  - “the collection and manipulation of items of data to produce meaningful information”
- One way to process data
  - Searching
  - Retrieving
  - Abstraction
  - Loading
  - Processing
  - Utilization



Searching

Retrieving

Abstraction

Loading

Persistence

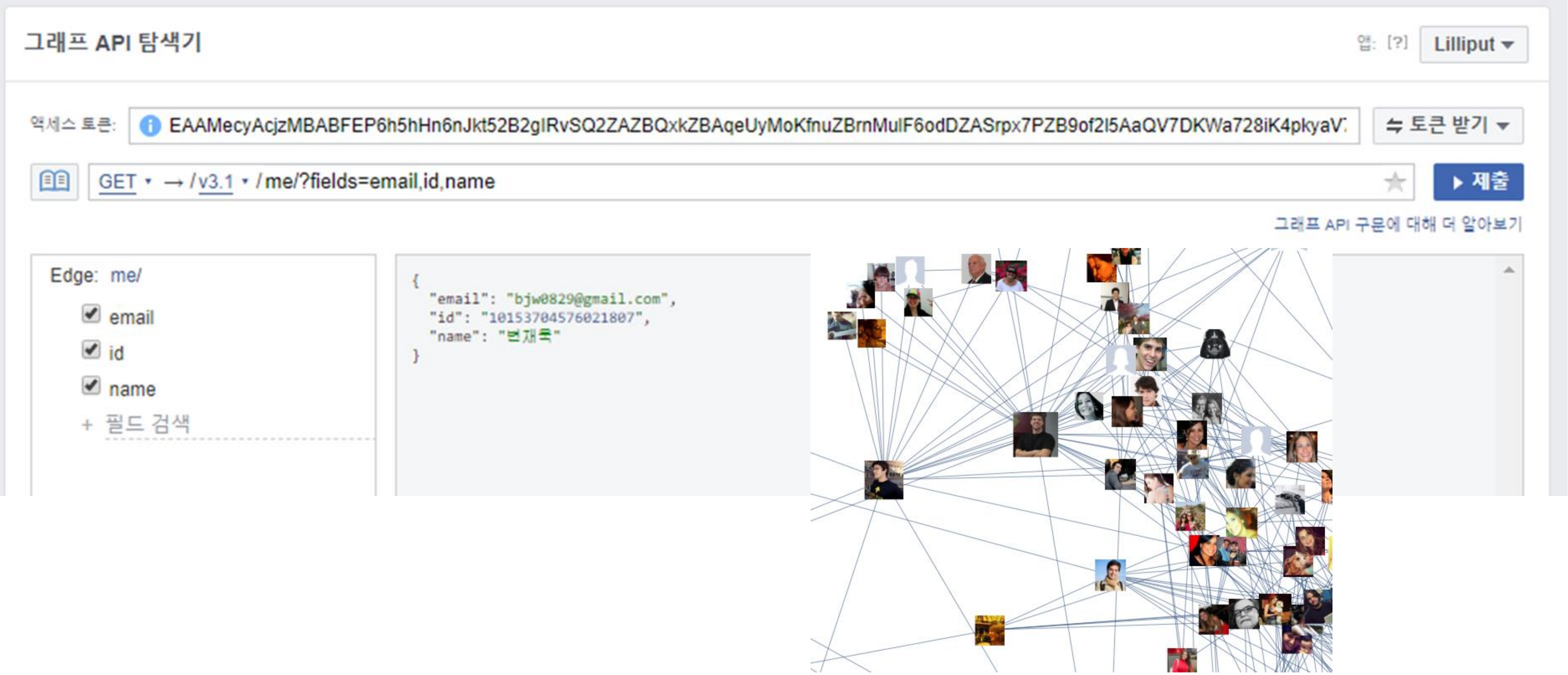
Processing

Utilization

- Visualization on Linux Kernel Development
  - https://github.com/torvalds/linux
  - https://www.youtube.com/watch?v=P_02QGsHzEQ

# Overview of Data Analysis: Retrieving Online Data

- Example: Facebook Graph API
  - https://developers.facebook.com/docs/graph-api/
  - REST API: [Facebook_API_Base_URL]/me/?fields=email,id,name



그래프 API 탐색기

앱: [?] Lilliput

액세스 토큰: EAAMecyAcjzMBABFEP6h5hHn6nJkt52B2gIRvSQ2ZAZBQxkZBAqeUyMoKfnuZBrnMulF6odDZASrpx7PZB9of2l5AaQV7DKWa728iK4pkyaV

GET → /v3.1 /me/?fields=email,id,name

그래프 API 구문에 대해 더 알아보기

Edge: me/

☑ email

☑ id

☑ name

+ 필드 검색

```
{
  "email": "bjw0829@gmail.com",
  "id": "10153704576021807",
  "name": "변재욱"
}
```

[Facebook Social Graph]

# Overview of Data Analysis: Retrieving Real-World events

- Example: The Live Social Semantics application… appeared in Percom Workshops 2010

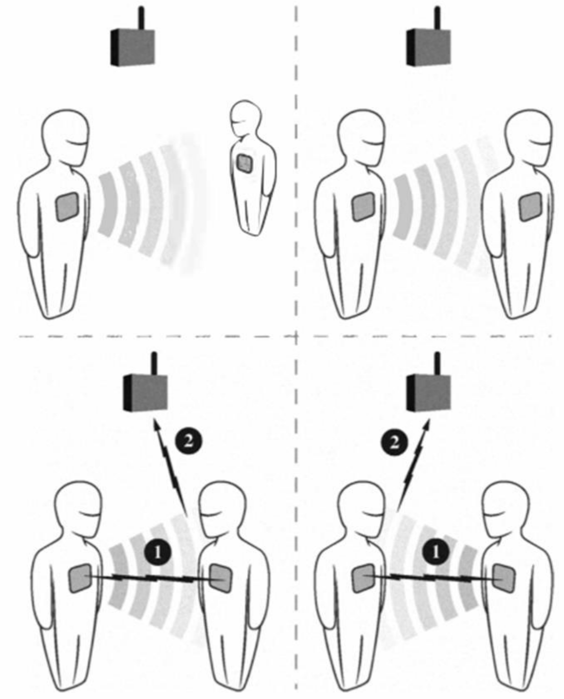


Figure 1. The SocioPatterns platform for distributed sensing of face-to-face proximity. Active RFID tags embedded in conference badges engage in ultra-low-power bidirectional packet exchange (1). Packet exchange is only possible (bottom panels) when two persons are at close range and facing each other, as the body blocks the exchange of low-power packets (top-right panel). Sustained face-to-face interactions are reported (2) to a data collection infrastructure.

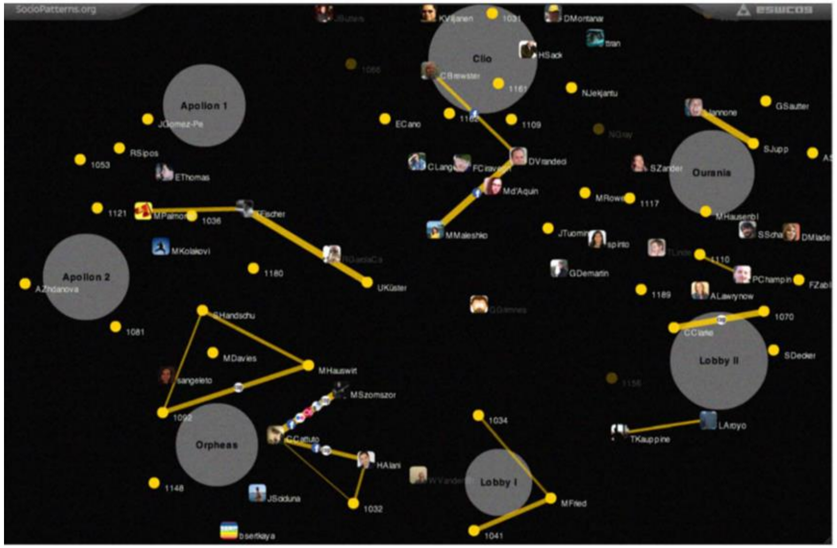


Figure 3. Screenshot of the spatial view grabbed during a session.

# Overview of Data Analysis: Open Datasets

- CRAWDAD
  - the Community Resource for Archiving Wireless Data At Dartmouth, a wireless network data resource for the research community. This archive has the capacity to store wireless trace data from many contributing locations, and staff to develop better tools for collecting, anonymizing, and analyzing the data. We work with community leaders to ensure that the archive meets the needs of the research community.
  - https://crawdad.org/index.html
- SNAP (Stanford Network Analysis Project)
  - A collection of more than 50 large network datasets from tens of thousands of nodes and edges to tens of millions of nodes and edges. In includes social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks.
  - https://snap.stanford.edu/data/

# Overview of Data Analysis: What we do today

- One way to process data
  - Searching: Search what kinds of data we will process
  - **Retrieving:**
    - Retrieving Online data
    - Sensing Real-world
    - **Read the file containing the dataset**
    - …
  - Abstraction: Model each line of the dataset into an instance of a class
  - Loading: Load a set of abstracted classes with a suitable collection
  - Processing: Let you know how to use essential methods in each collection
  - Utilization: **Make a file for the results**, visualization, etc.

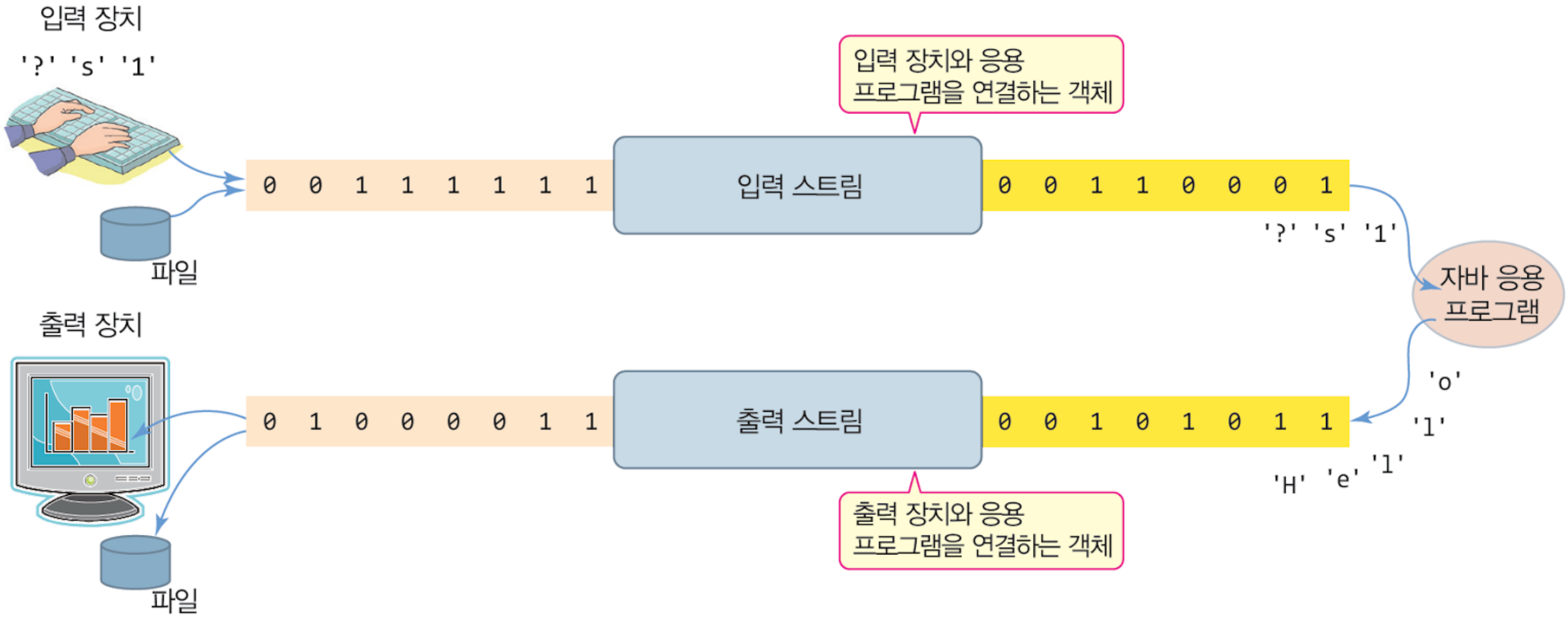
# First Dataset

- EU email communication network
    - https://snap.stanford.edu/data/email-EuAll.html
    - Email network of a large European Research Institution (directed edge means at least one email was sent between October 2003 and March 2005)
    - 265214 people send 420045 emails

# File Input/Output: Stream

- Stream input/output
  - Input/output with buffer

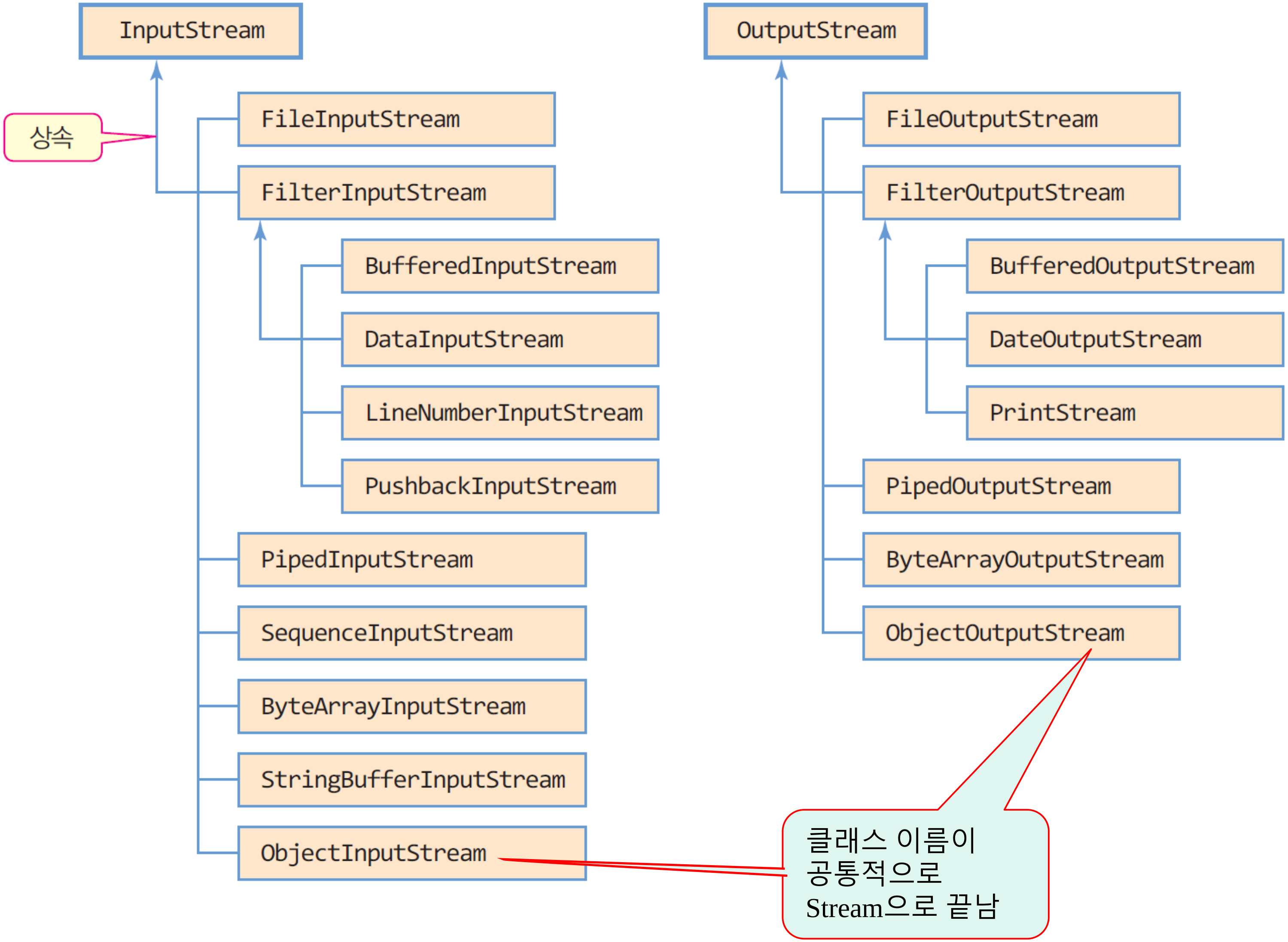
- Scanner(System.in);
- System.out.println(String str);



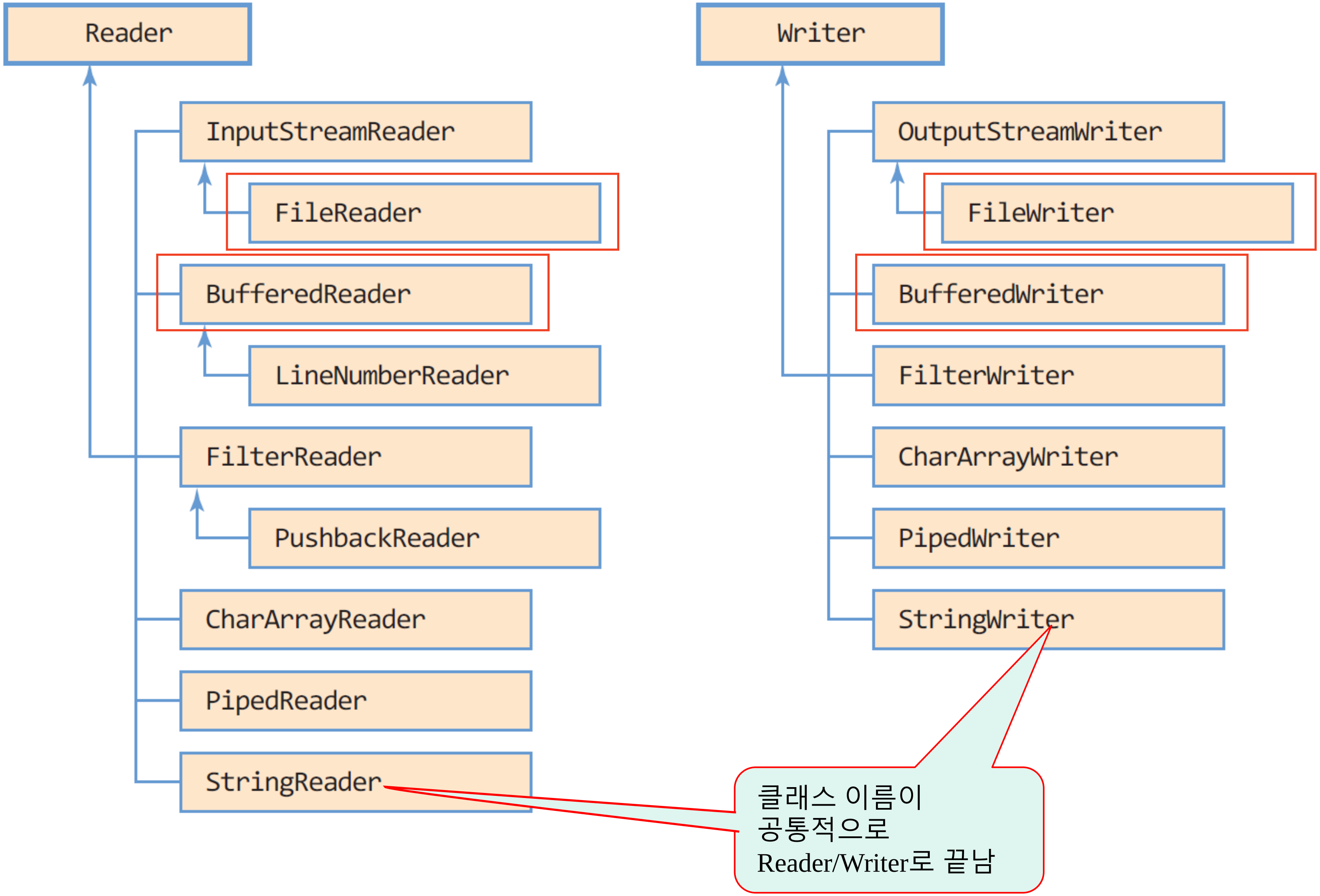
# File Input/Output: Stream

- Type of Stream
    - Byte Stream and Character Stream
        - Byte Stream
            - Usually used for processing binary data
                - e.g., image, audio, video
        - Character Stream
            - Used for processing text file
                - Cannot recognize binary data

# File Input/Output: Java Classes for Byte Stream



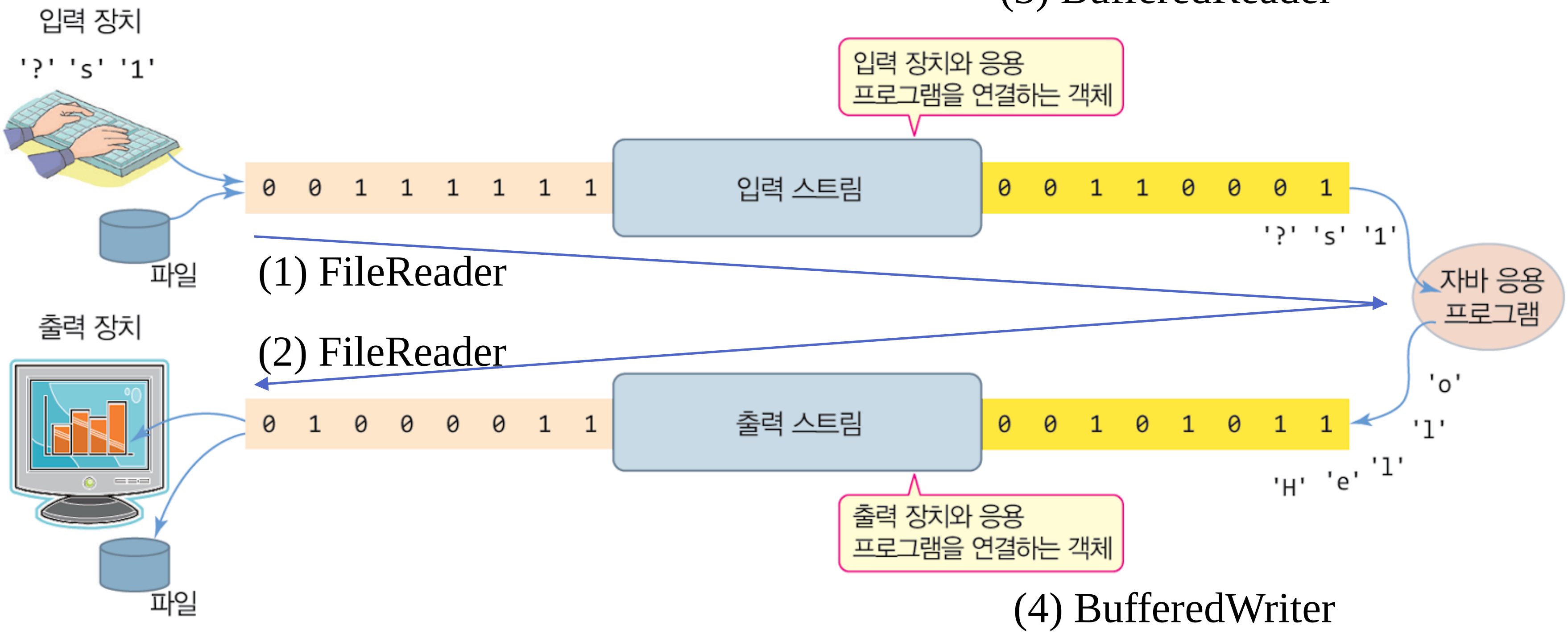
# File Input/Output: Java Classes for Character Stream



클래스 이름이
공통적으로
Reader/Writer로 끝남

# File Input/Output: Stream

- Stream input/output
  - Input/output with buffer



(3) BufferedReader

(1) FileReader

(2) FileReader

(4) BufferedWriter

# Practice #1: Read File using FileReader

- Use FileReader class
  - read(): read one character as an integer ( cast to char )
  - Need to be closed
  - Need to handle exceptions

# Practice #2: Write File using FileReader

- Use FileWriter class
  - write(String str): write *str* to a file
  - Need to be closed
  - Need to handle exceptions

# Practice #3: Copy File using FileReader/Writer

- Use FileReader and FileWriter class
  - use read() and write(char c)
  - Need to be closed
  - Need to handle exceptions

# Buffered Reader and Writer?

- For example
    - You have 100 million lines to write
    - FileWriter.write(String eachLine) will invoke a system call to write a line 100 million times

    - Buffered Writer
        - writes data
            - only when a buffer is filled || flush() or close() are invoked
        - Can reduce the number of the calls

# Practice #4: Copy file using BufferedReader and Writer

- Use BufferedReader and wrapping a FileReader
    - readLine(): read each line as String
- Use BufferedWriter and wrapping a FileWriter
    - write(String str): write *str* to a file (Note: insert new line yourself)

# Practice #5/6: Compute a time to complete a task

- System.currentTimeMillis();
  - Java Doc: “Returns the current time in milliseconds”

- long preTime = System.currentTimeMillis();

- long afterTime = System.currentTimeMillis();

- System.out.println(“Computation Time: “ + (afterTime – preTime));

# Practice #7: String.split()

- String[] String.split(String regex)
  - Splits this string around matches of the given **regular expression**.
  - https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

- String str = “a	b	c	d”	(delimiter = \t)
- String[] elem = str.split(“\t”);
  - elem[0] = a;
  - elem[1] = b;
  - elem[2] = c;
  - elem[3] = d;
- String str2 = “a b c d” (delimiter = *white space* “\\s”);
- String[] elem2 = str.split(“\\s”);
  - Identical result to elem;

# Summary

- Overview of Data Analysis
- First Dataset
- File Input/Output

- Next Class
  - Data modelling, collecting, and analyses