HALLA
UNIVERSITY

Advanced Programming (2)

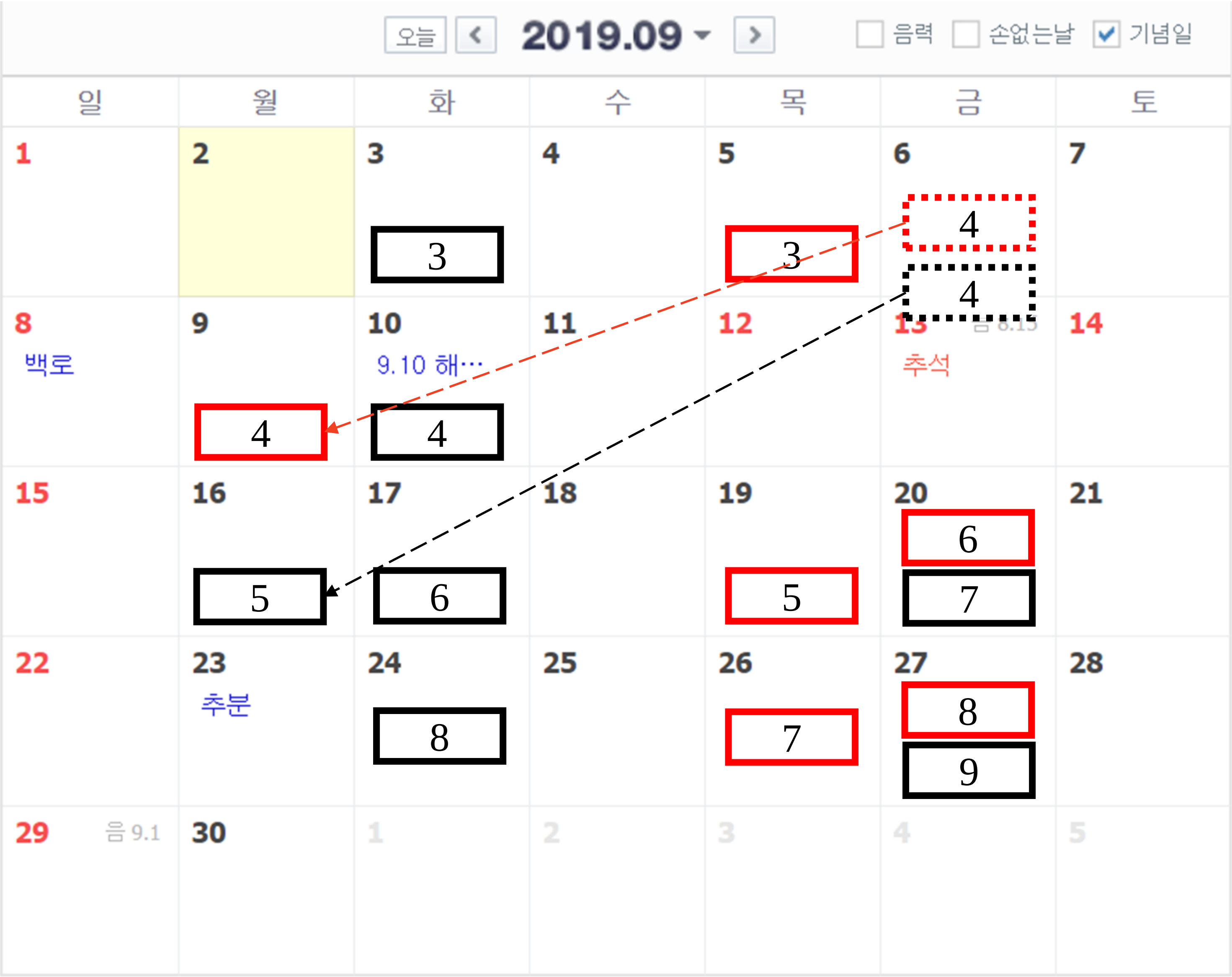
(File Input/Output)

Fall, 2019

Calendar

AP1

AP2



오늘 < 2019.09 > □ 음력 □ 손없는날 ☑ 기념일

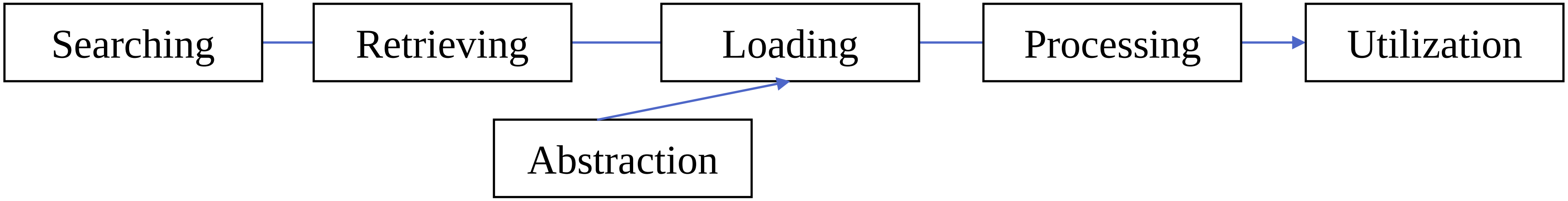
일	월	화	수	목	금	토
1	2	3 3	4	5 3	6 4 4	7
8 백로	9 4	10 9.10 해… 4	11	12	13 음 8.15 추석	14
15	16 5	17 6	18	19 5	20 6 7	21
22	23 추분	24 8	25	26 7	27 8 9	28
29 음 9.1	30	1	2	3	4	5

Table of Contents

- Data Processing
- Dataset Overview
- File Input/Output

Data Processing

- 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

Loading

Processing

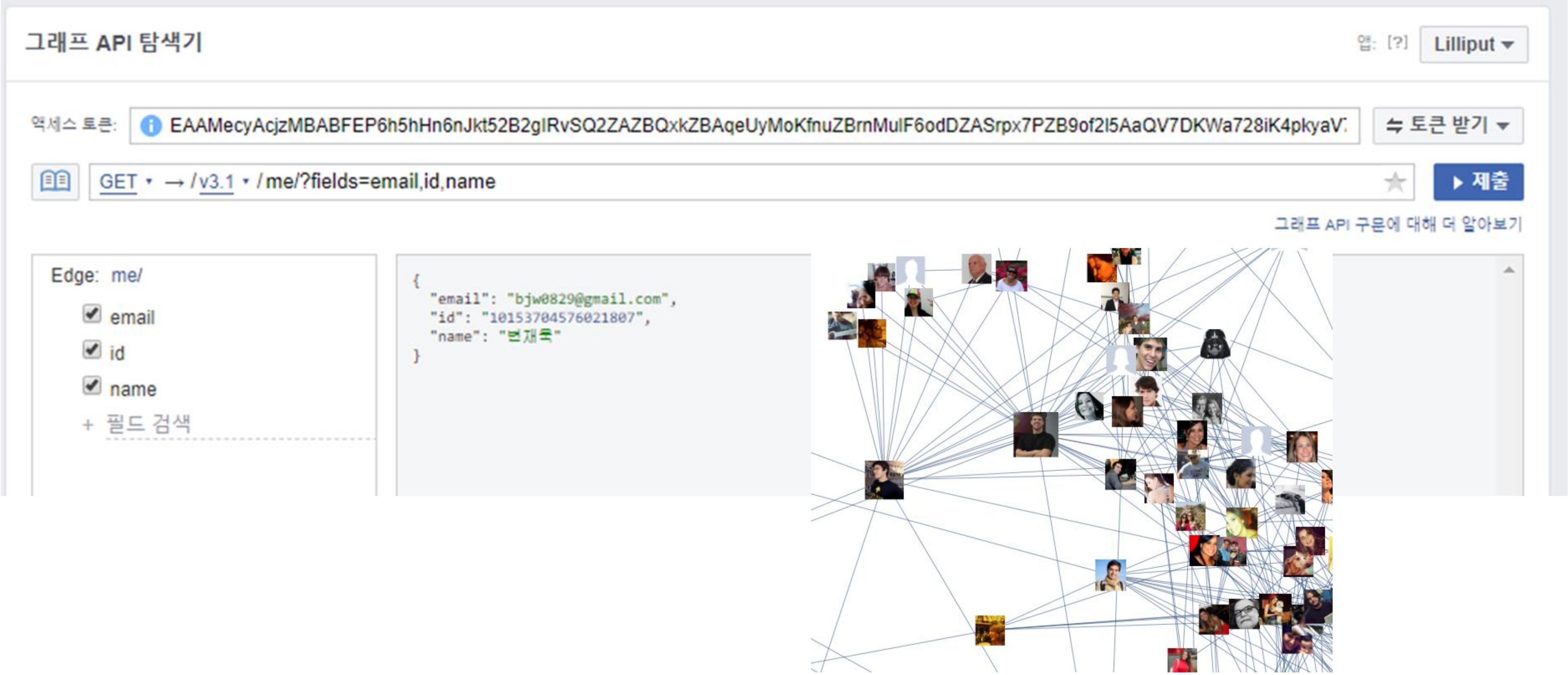
Utilization

Abstraction

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

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

액세스 토큰: EAAMecyAcjzMBABFEP6h5hHn6nJkt52B2gIRvSQ2ZAZBQxkZBAqeUyMoKfnuZBrnMulF6odDZASrpx7PZB9of2I5AaQV7DKWa728iK4pkyaV:

GET → /v3.1 / me/?fields=email,id,name

Edge: me/

☑ email
☑ id
☑ name

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

[Facebook Social Graph]

Retrieving Real-World events

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

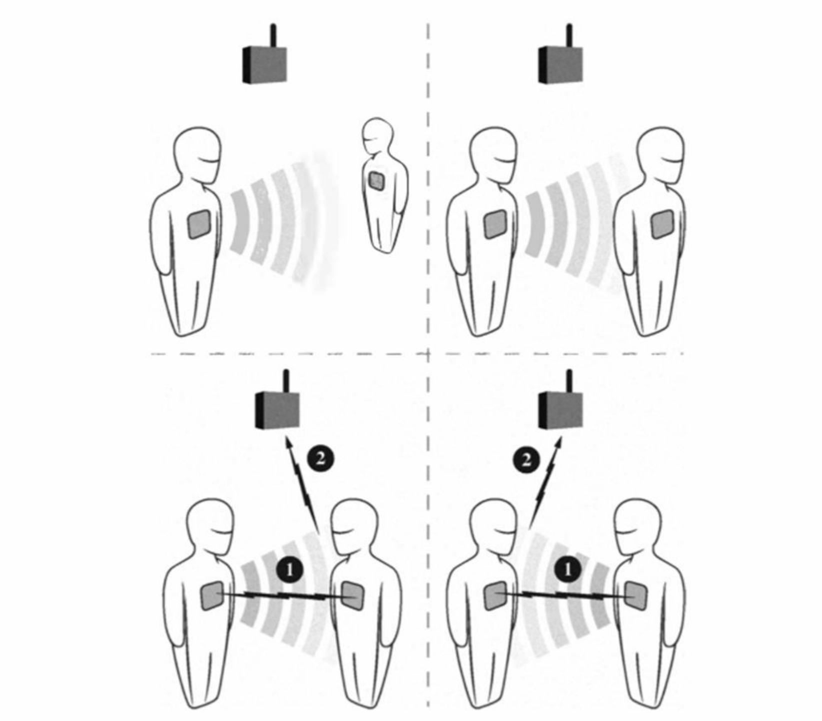


Figure 1. The SocioPatterns platform for distributed sensing of face-to-face proximity. Active RFID tags embedded in conference badges engage in ultra-low-power bidirectional packet exchange (1). Packet exchange is only possible (bottom panels) when two persons are at close range and facing each other, 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.

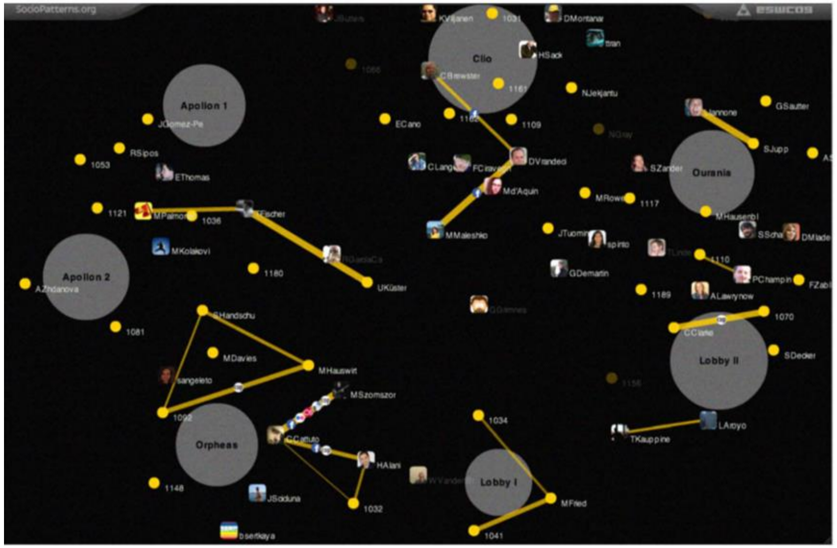


Figure 3. Screenshot of the spatial view grabbed during a session.

We are to use 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/

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.

Dataset

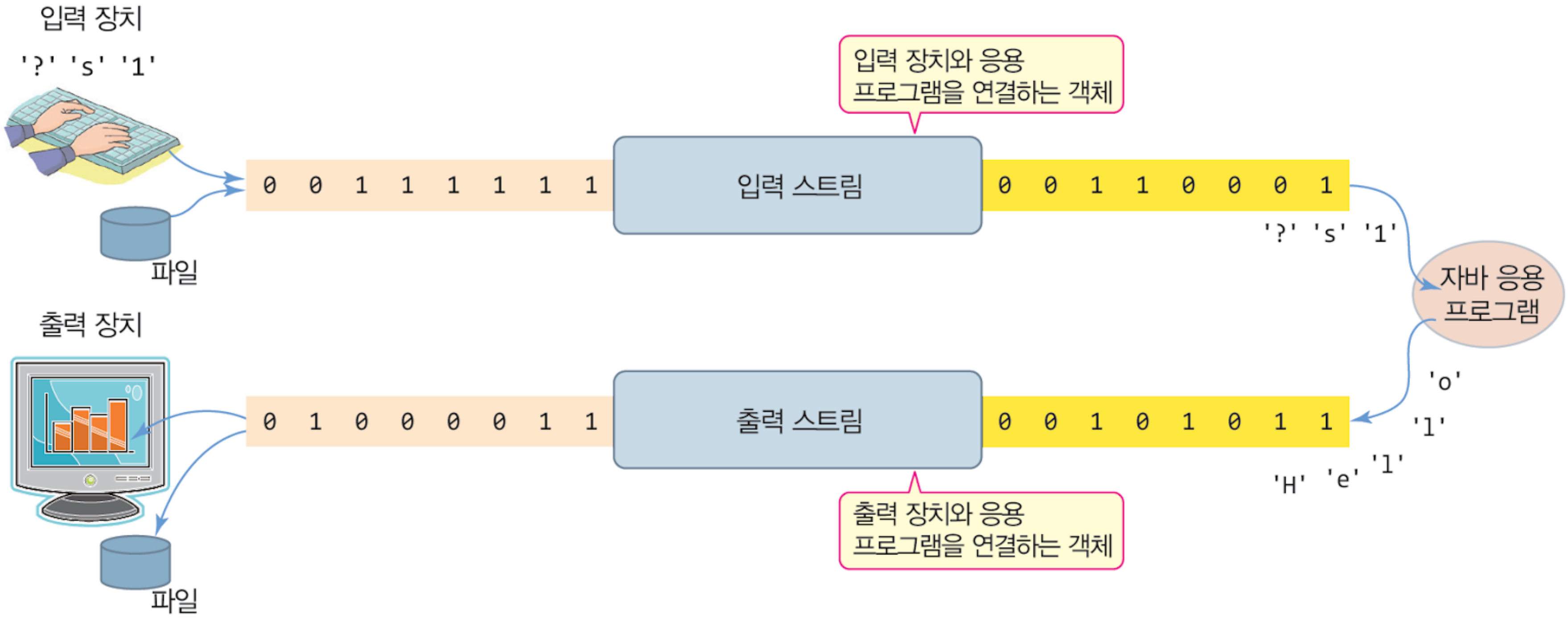
- EU email communication network
    - https://snap.stanford.edu/data/email-EuAll.html
    - Email network of a large European Research Institution (directed edge means at least one email was sent between October 2003 and March 2005)
    - 265214 people send 420045 emails

- Google Drive / Dataset / Email-EuAll.txt
    - The reduced data for practice

Stream

- Stream input/output
  - Input/output with buffer

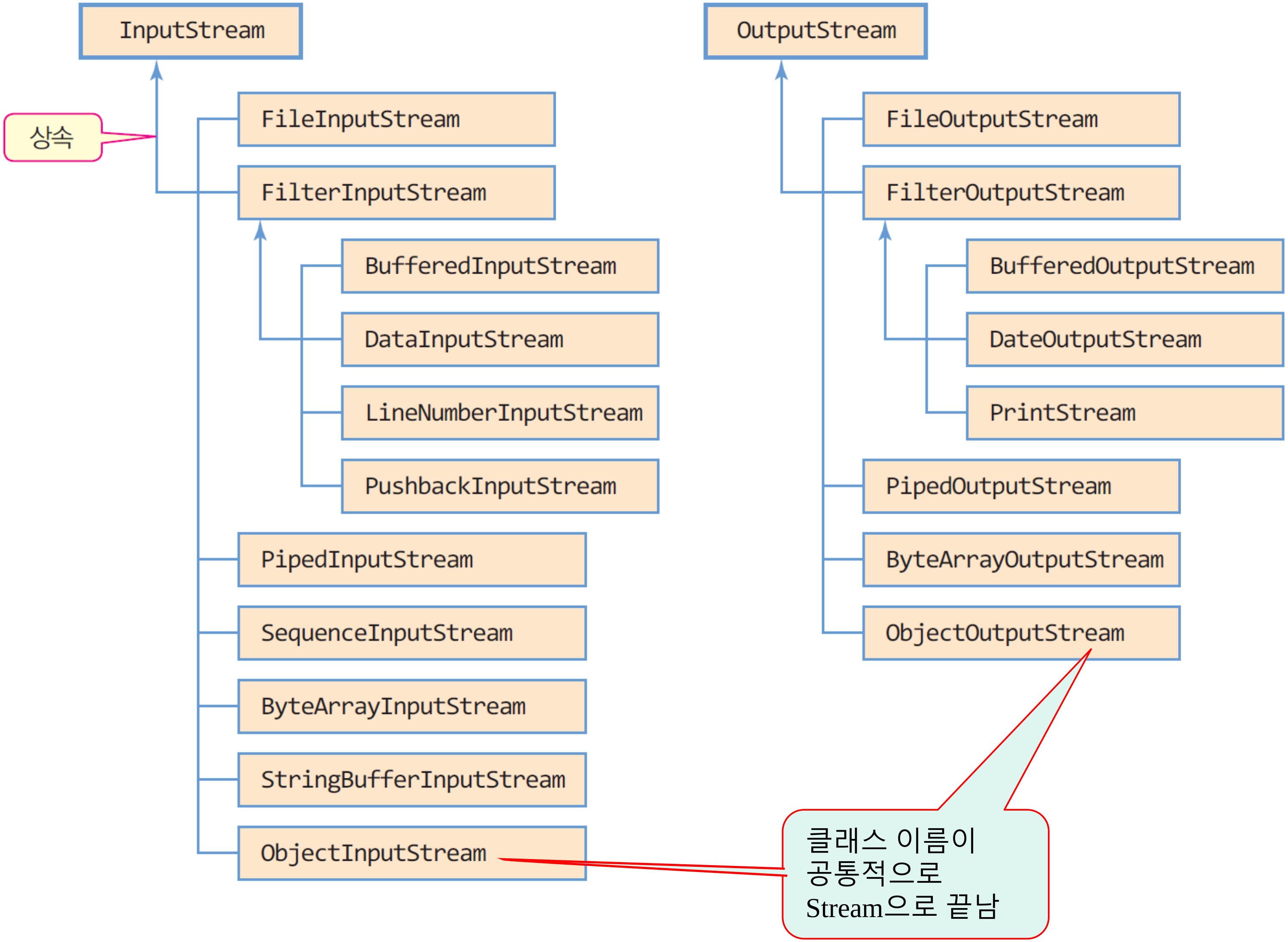
- Scanner(System.in);
- System.out.println(String str);



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

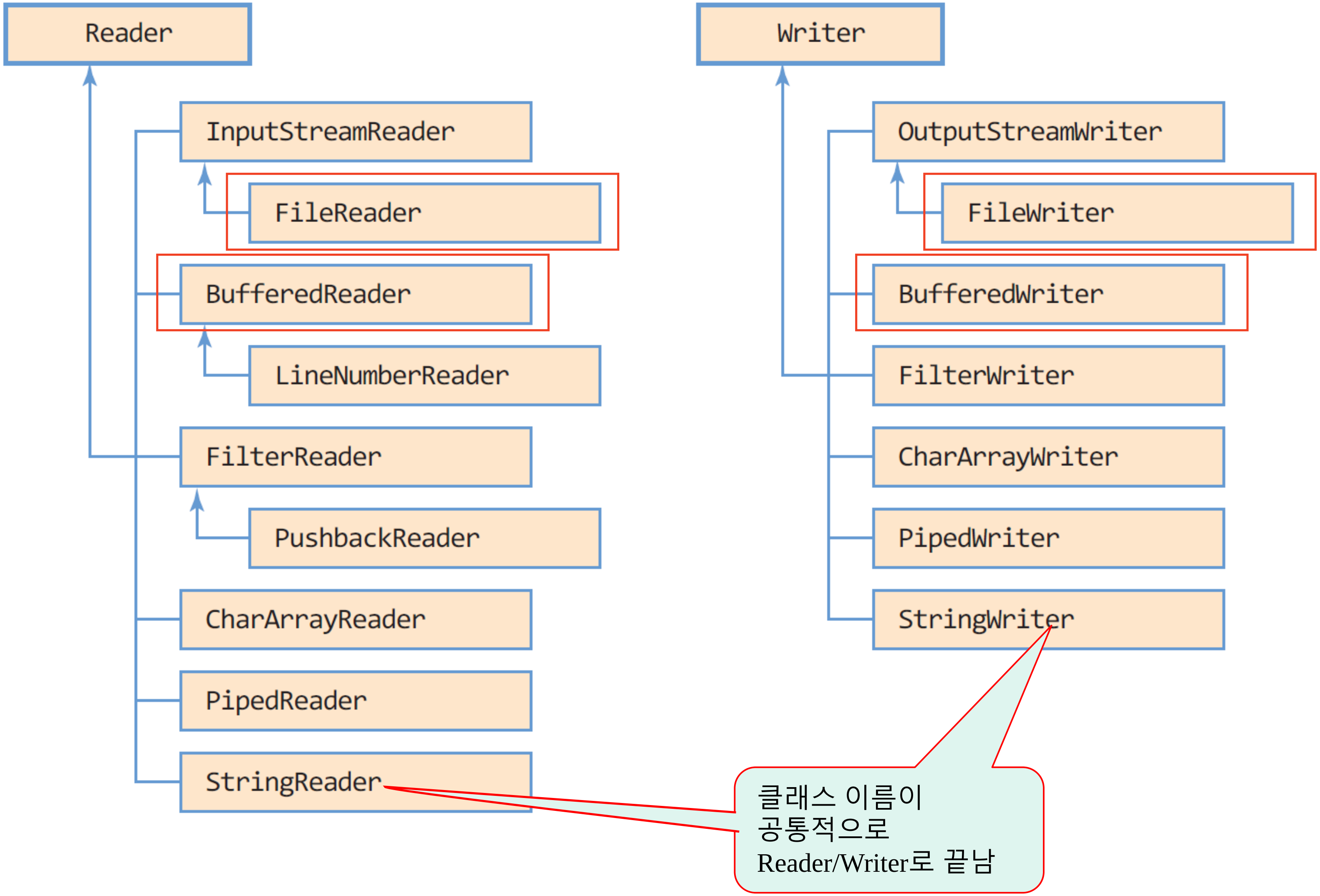
Java Classes for Byte Stream



상속

클래스 이름이
공통적으로
Stream으로 끝남

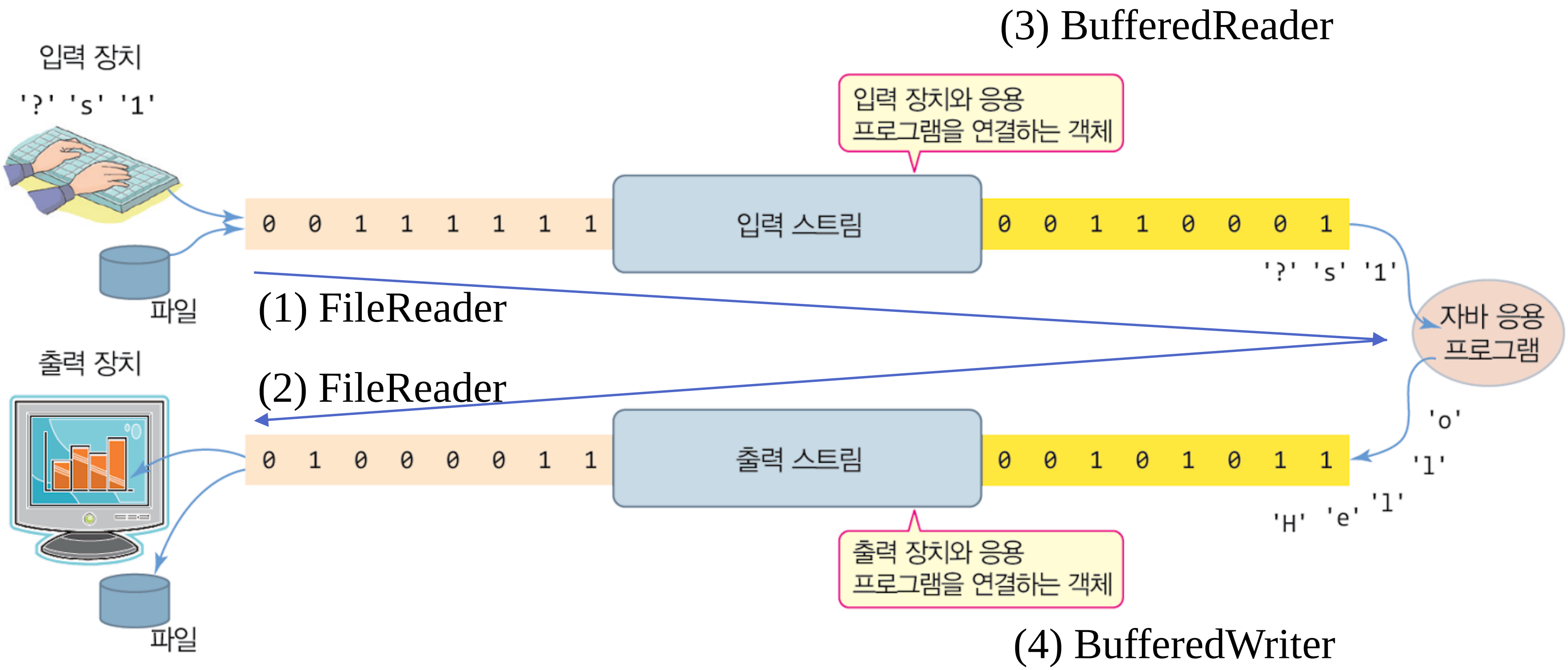
Java Classes for Character Stream



클래스 이름이
공통적으로
Reader/Writer로 끝남

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

- https://github.com/JaewookByun/h02404/blob/master/data_processing/src/main/java/kr/ac/halla/ice/advanced_programming/week3/Practice1.java ← Students do not have an access right to the file

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

- https://github.com/JaewookByun/h02404/blob/master/data_processing/src/main/java/kr/ac/halla/ice/advanced_programming/week3/Practice2.java

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

- https://github.com/JaewookByun/h02404/blob/master/data_processing/src/main/java/kr/ac/halla/ice/advanced_programming/week3/Practice3.java

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)

- https://github.com/JaewookByun/h02404/blob/master/data_processing/src/main/java/kr/ac/halla/ice/advanced_programming/week3/Practice4.java

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

- Data Processing
- Dataset Overview
- File Input/Output

- Next Class
  - Generics and Collection 1
- Next next class
  - In-class assignment 1