SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Java

SAMSUNG SW ACADEMY FOR YOUTH

Programming Language

객체지향 프로그래밍

- 파일입출력

SAMSUNG SW ACADEMY FOR YOUTH

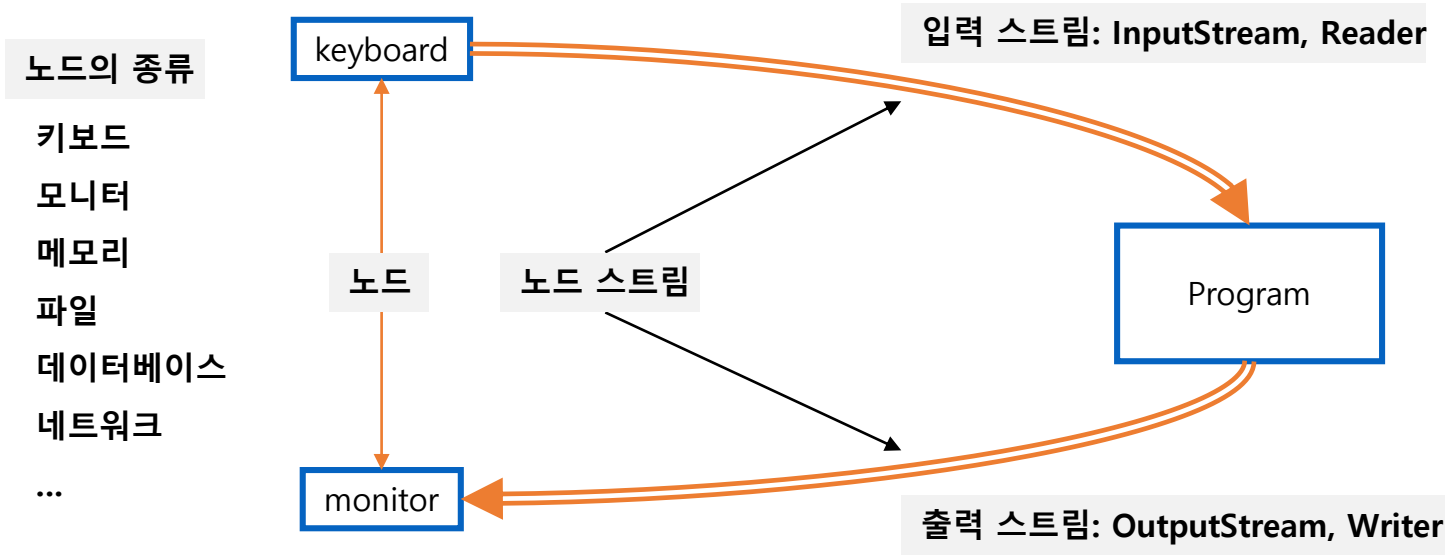
함께가요 미래로!
Enabling People

파일 입출력

입출력 스트림의 개요

I/O 와 Stream

- I/O : 데이터의 입력(input)과 출력(output)
- 데이터는 한쪽에서 주고 한쪽에서 받는 구조
  - 이때, 입력과 출력의 끝단 : 노드 (Node)
  - 두 노드를 연결하고 데이터를 전송할 수 있는 개념 : 스트림 (Stream)
  - 스트림은 단방향으로만 통신이 가능, 하나의 스트림으로 입력과 출력을 같이 처리할 수 없음



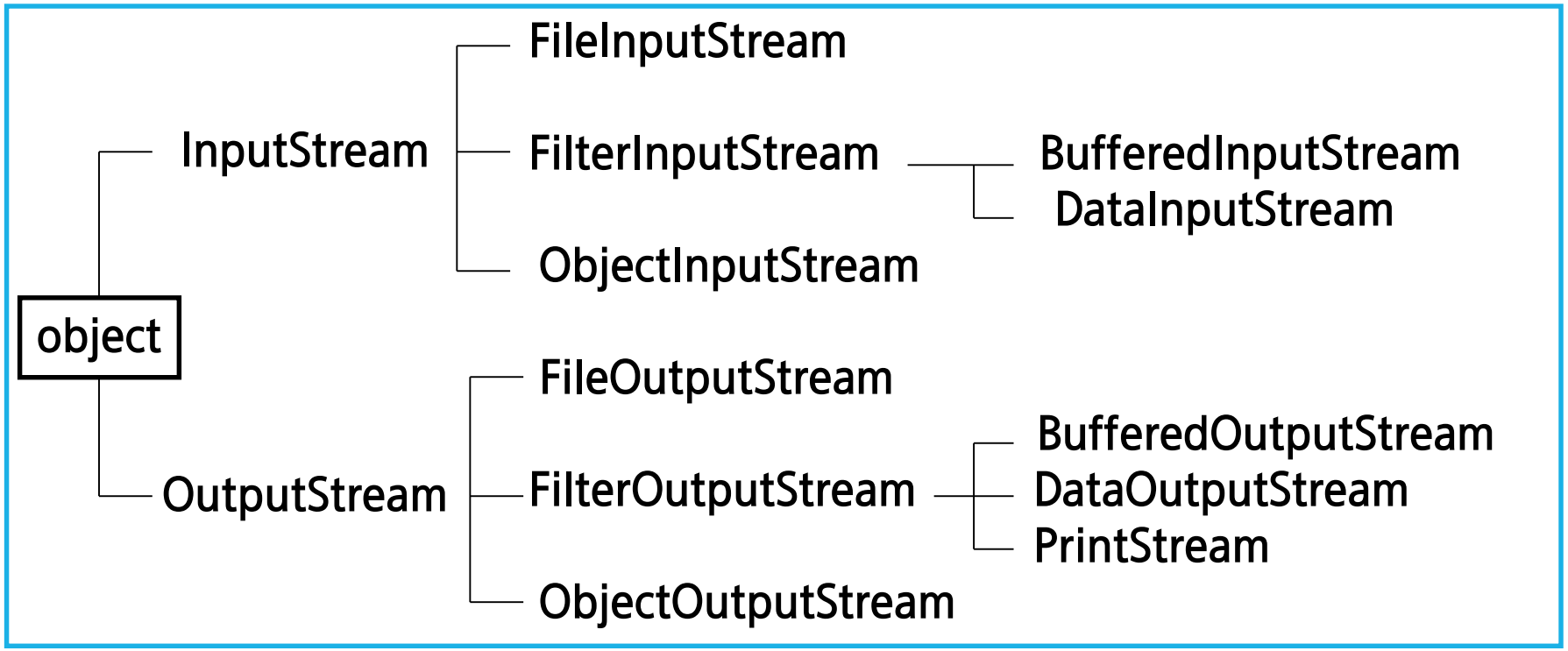
java.io 패키지의 개요

I/O 처리 단위

	byte	Char
입 력	InputStream	Reader
출 력	OutputStream	Writer

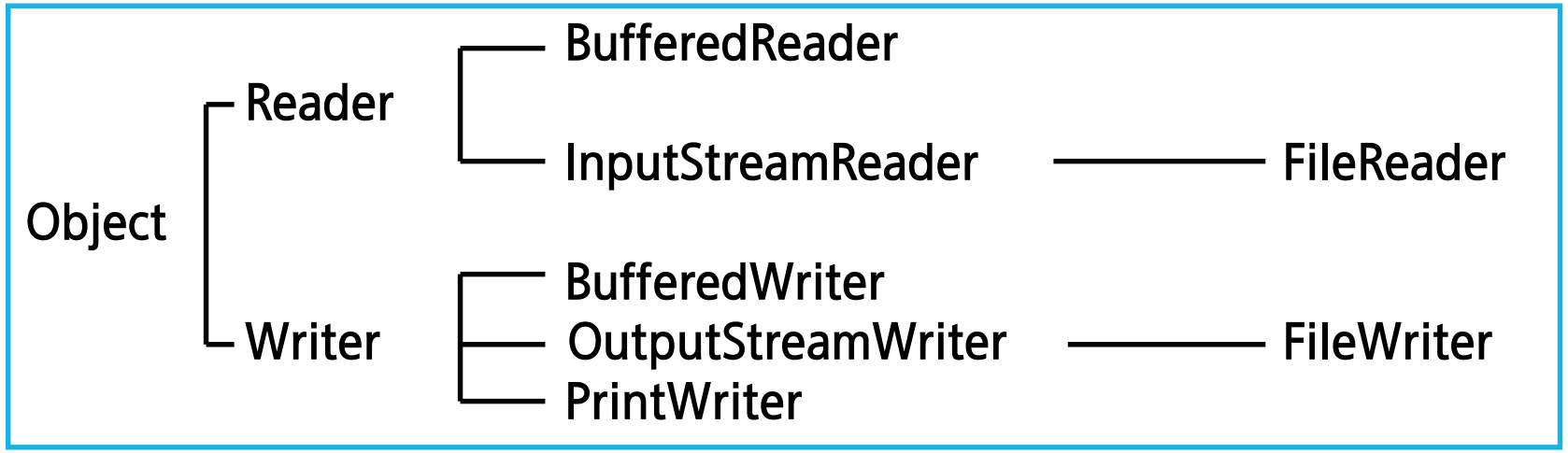
java.io 패키지의 개요

바이트 스트림



java.io 패키지의 개요

문자 스트림



InputStream의 주요 메서드

메서드 명	선언부와 설명
read()	public abstract int read() throws IOException
	byte 하나를 읽어서 int로 반환한다. 더 이상 읽을 값이 없으면 -1을 리턴한다.
	public int read(byte b[]) throws IOException
	데이터를 읽어서 b를 채우고 읽은 바이트의 개수를 리턴한다. 0이 리턴 되면 더 이상 읽을 값이 없는 상황이다.
	public int read(byte b[], int offset, int len) throws IOException
	최대 len 만큼 데이터를 읽어서 b의 offset부터 b에 저장하고 읽은 바이트 개수를 리턴한다. 따라서 len+offset은 b의 크기 이하여야 한다.
close()	public void close() throws IOException
	스트림을 종료해서 자원을 반납한다.

InputStream의 주요 메서드

- 바이트 한 개씩 읽음

```
private String data1 = "hi java world";
private void read1() {
    try (InputStream input = new ByteArrayInputStream(data1.getBytes())) {
        int read = -1;
        while ((read = input.read()) != -1) {
            System.out.printf("읽은 값: %d, 문자로: %c%n", read, read);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
읽은 값: 104, 문자로: h
읽은 값: 105, 문자로: i
읽은 값: 32, 문자로:
읽은 값: 106, 문자로: j
읽은 값: 97, 문자로: a
읽은 값: 118, 문자로: v
읽은 값: 97, 문자로: a
. . .
```

- Buffer 만큼 씩 읽음

```
private String data2 = "자바는 객체지향 언어입니다.";
private void read2() {
    byte[] buffer = new byte[10];
    try (InputStream input = new ByteArrayInputStream(data2.getBytes())) {
        int read = -1;
        while ((read = input.read(buffer)) > 0) {
            System.out.printf("읽은 개수: %d, 문자열: %s%n", read, new String(buffer));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Utf-8 한글은 한 글자가 3byte

```
읽은 개수: 10, 문자열: 자바는
읽은 개수: 10, 문자열: 객체지�
읽은 개수: 10, 문자열: ��  언어�
읽은 개수: 9, 문자열: ��니다.�
```

유니코드는 깨질 수 있음.

Reader의 주요 메서드

메서드 명	선언부와 설명
read()	public int read() throws IOException
	char 하나를 읽어서 int로 반환한다. 더 이상 읽을 값이 없으면 -1을 리턴한다.
	public int read(char cbuf[]) throws IOException
	데이터를 읽어서 cbuf를 채우고 읽은 문자의 개수를 리턴한다. 0이 리턴 되면 더 이상 읽을 값이 없는 상황이다.
	abstract public int read(char cbuf[], int off, int len) throws IOException
	최대 len 만큼 데이터를 읽어서 cbuf의 offset부터 cbuf에 저장하고 읽은 char 개수를 리턴한다. 따라서 len+off는 cbuf의 크기 이하여야 한다.
	public int read(java.nio.CharBuffer target) throws IOException
	데이터를 읽어서 target에 저장한다. target은 cbuf를 대체한다.
close()	public void close() throws IOException
	스트림을 종료해서 자원을 반납한다.

Reader의 주요 메서드

- Buffer 만큼 씩 읽음

```
private void read3() {
    char[] buffer = new char[10];
    try (Reader input = new CharReader(data2.toCharArray(), data2.length())) {
        int read = -1;
        while ((read = input.read(buffer)) > 0) {
            System.out.printf("읽은 개수: %d, 문자열: %s%n", read, new String(buffer, 0, read));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
읽은 개수: 10, 문자열: 자바는 객체지향 언
읽은 개수: 5, 문자열: 어입니다.
```

OutputStream의 주요 메서드

메서드 명	선언부와 설명
write()	public abstract void write(int b) throws IOException
	b의 내용을 byte로 출력한다.
	public void write(byte b[]) throws IOException
	b를 문자열로 변환해서 출력한다.
	public void write(byte b[], int off, int len) throws IOException
	b의 off 부터 off+len-1만큼을 문자열로 변환해서 출력한다.
close()	public void close() throws IOException
	스트림을 종료해서 자원을 반납한다. close()는 내부적으로 flush()를 호출한다.
flush()	public void flush() throws IOException
	버퍼가 있는 스트림에서 버퍼의 내용을 출력하고 버퍼를 비운다.

Writer의 주요 메서드

메서드 명	선언부와 설명
write()	public void write(int c) throws IOException
	b의 내용을 char로 출력한다.
	public void write(char cbuf[]) throws IOException
	cbuf를 문자열로 변환해서 출력한다.
	abstract public void write(char cbuf[], int off, int len) throws IOException
	cbuf의 off 부터 off+len-1만큼을 문자열로 변환해서 출력한다.
	public void write(String str) throws IOException
	str을 출력한다.
	public void write(String str, int off, int len) throws IOException
	str의 off 부터 off+len-1만큼을 출력한다.
append()	public Writer append(CharSequence csq) throws IOException
	csq를 출력하고 Writer를 리턴한다.
	public Writer append(CharSequence csq, int start, int end) throws IOException
	csq의 start부터 end 까지를 출력하고 Writer를 리턴한다.
close()	public void close() throws IOException
	스트림을 종료해서 자원을 반납한다. close()는 내부적으로 flush()를 호출한다.
flush()	abstract public void flush() throws IOException
	버퍼가 있는 스트림에서 버퍼의 내용을 출력하고 버퍼를 비운다.

File

- 가장 기본적인 입출력 장치 중 하나로 파일과 디렉터리를 다루는 클래스

메서드 명	선언부와 설명
File()	public File(String pathname)
	pathname에 해당하는 파일을 생성한다. 경로 없이 파일을 생성하면 애플리케이션을 시작한 경로가 된다.
	public File(String parent, String child)
	parent 경로 아래 child를 생성한다.
	public File(File parent, String child)
	parent 경로 아래 child를 생성한다.
	public File(URI uri)
	file로 시작하는 URI 객체를 이용해 파일을 생성한다.
createNewFile()	public boolean createNewFile() throws IOException
	새로운 물리적인 파일을 생성한다.
mkdir()	public boolean mkdir()
	새로운 디렉토리를 생성한다.
mkdirs()	public boolean mkdirs()
	경로상에 없는 모든 디렉토리를 생성한다.
delete()	public boolean delete()
	파일 또는 디렉토리를 삭제한다.

File

- 가장 기본적인 입출력 장치 중 하나로 파일과 디렉터리를 다루는 클래스

메서드 명	선언부와 설명
getName()	public String getName()
	파일의 이름을 리턴한다.
getPath()	public String getPath()
	파일의 경로를 리턴한다.
getAbsolutePath()	public String getAbsolutePath()
	파일의 절대 경로를 리턴한다.
getCanonicalPath()	public String getCanonicalPath() throws IOException
	파일의 정식 경로를 리턴한다. 정식 경로는 절대 경로이며 경로 내에 . 또는 .. 의 상대 경로 기호가 없는 경로이다.
isDirectory()	public boolean isDirectory()
	파일이 디렉토리인지 리턴한다.
isFile()	public boolean isFile()
	파일이 파일인지 리턴한다.
length()	public long length()
	파일의 길이를 리턴한다.
listFiles()	public File[] listFiles()
	파일이 디렉토리인 경우 자식 파일들을 File[] 형태로 리턴한다.

FileInputStream, FileOutputStream

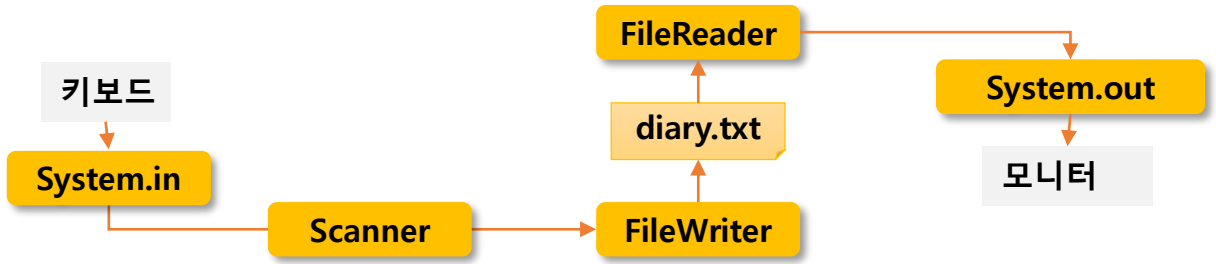
메서드 명	선언부와 설명
FileInputStream ()	public FileInputStream(String name) throws FileNotFoundException
	name 경로의 파일을 읽는 FileInputStream을 생성한다.
FileOutputStream()	public FileOutputStream(String name) throws FileNotFoundException
	name 경로의 파일에 출력하는 FileOutputStream을 생성한다.
	public FileOutputStream(String name, boolean append) throws FileNotFoundException
	name 경로의 파일에 출력하는 FileOutputStream을 생성한다. 기존에 파일이 있다면 뒤에 이어 쓴다.

String name 대신 File 객체 사용 가능

```
private long fileMove(int bufferSize) {
    long start = System.currentTimeMillis();
    File src = new File("c:\\ssafy\\eclipse-jee-2018-09-win32-x86_64.zip");
    File target = new File("c:\\Temp\\eclipse.zip");

    try (FileInputStream fin = new FileInputStream(src); FileOutputStream fout = new FileOutputStream(target)) {
        byte[] buffer = new byte[bufferSize];
        int read = 0;
        while ((read = fin.read(buffer)) > 0) {
            fout.write(buffer, 0, read);
        }
    } catch (IOException e) {        e.printStackTrace();    }

    return System.currentTimeMillis() - start;
}
```

FileReader, FileWriter

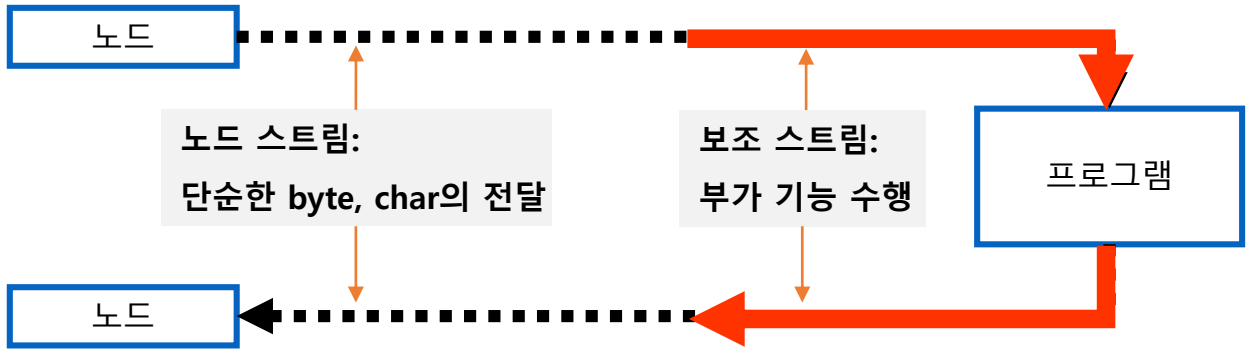
```
File target = new File("c:" + File.separator + "Temp" + File.separator + "diary.txt");

try (Scanner scanner = new Scanner(System.in);
     FileWriter writer = new FileWriter(target, true);
     FileReader reader = new FileReader(target);) {
     System.out.println("일기를 작성합니다. 그만두려면 x를 입력하세요.");
     writer.write("\n오늘 날짜: - " + new Date() + "\n");
     String str = null;
     while (true) {
          str = scanner.nextLine();
          if (str.equals("x")) {
               break;
          } else {
               writer.write(str + "\n");
          }
     }
     writer.flush();// 버퍼의 내용을 출력하고 비움

     char[] buffer = new char[10];
     int read = -1;
     while ((read = reader.read(buffer)) > 0) {
          System.out.print(String.valueOf(buffer, 0, read));
     }
} catch (IOException e) {
     e.printStackTrace();
}
```



보조 스트림의 개념과 종류

보조 스트림 : Filter Stream, Processing Stream

- 다른 스트림에 부가적인 기능을 제공하는 스트림



- 문자 set 변환
- Buffering
- 기본 데이터 형의 전송
- 객체 입출력

- 스트림 체이닝(Stream Chaining)
  - 필요에 따라 여러 보조 스트림을 연결해서 사용 가능

보조 스트림의 개념과 종류

보조 스트림의 종류

기능	byte 기반	char 기반
byte 스트림을 char 스트림으로 변환	InputStreamReader OutputStreamWriter	
버퍼링을 통한 속도 향상	BufferedInputStream BufferedOutputStream	BufferedReader BufferedWriter
객체 전송	ObjectInputStream ObjectOutputStream	

생성 : 이전 스트림을 생성자의 파라미터에 연결

```
new BufferedInputStream(System.in);

new DataInputStream(new BufferedInputStream(new FileInputStream());
```

종료 : 보조 스트림의 close()를 호출하면 노드 스트림의 close() 까지 호출 됨

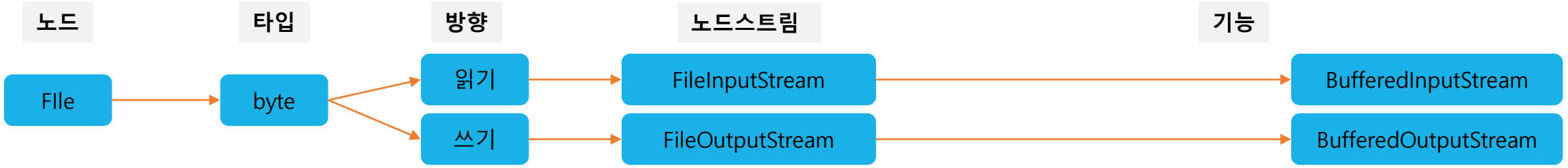
보조 스트림의 개념과 종류

사용할 스트림의 결정 과정

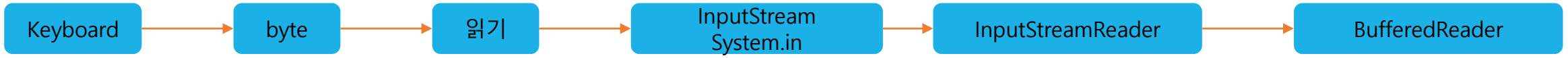
- 노드가 무엇인가 → 타입은 문자열인가? 바이트인가? → 방향이 무엇인가? → 추가 기능이 필요한가?

  노드 스트림 구성 / 보조 스트림 구성

- 어떤 파일을 빠른 속도로 이동시키고 싶다면?



- 키보드에서 유니코드 문자를 안전하고 빠르게 읽고 싶다면?



- 메모리의 객체를 파일로 저장하고 싶다면?

보조 스트림 활용

InputStreamReader & OutputStreamWriter

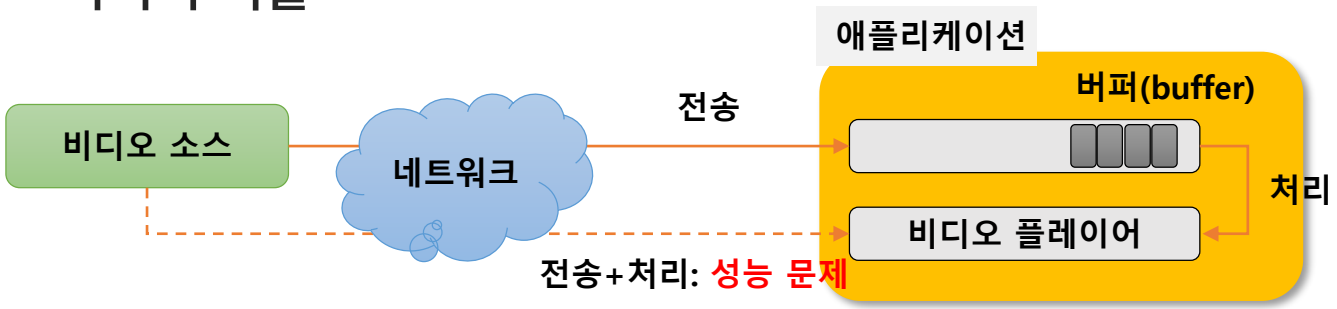
- byte기반 스트림을 char 기반으로 변경해주는 스트림
  - 문자열을 관리하기 위해서는 byte 단위보다 char 단위가 유리
  - 키보드에서 입력 (byte stream) 받은 데이터를 처리할 경우 등
- 변환 시 encoding 지정 가능

메서드 명	선언부와 설명
InputStreamReader()	public InputStreamReader(InputStream in)
	기본 캐릭터 셋을 이용해서 InputStreamReader를 생성한다.
	public InputStreamReader(InputStream in, String charsetName) throws UnsupportedEncodingException
	charsetName을 이용해 InputStreamReader를 생성한다.
	public InputStreamReader(InputStream in, Charset cs)
	cs를 이용해 InputStreamReader를 생성한다.
OutputStreamWriter()	public OutputStreamWriter(OutputStream out)
	기본 캐릭터 셋을 이용해 OutputStreamWriter를 생성한다.
	public OutputStreamWriter(OutputStream out, String charsetName) throws UnsupportedEncodingException
	charsetName을 이용해 OutputStreamWriter를 생성한다.
	public OutputStreamWriter(OutputStream out, Charset cs)
	cs를 이용해 OutputStreamWriter를 생성한다.

보조 스트림 활용

Buffered 계열

- 버퍼의 역할



- 스트림의 입/출력 효율을 높이기 위해 버퍼를 사용하는 스트림

메서드 명	선언부와 설명
BufferedInputStream()	public BufferedInputStream(InputStream in)
	in을 이용해 크기가 8192인 버퍼를 갖는 BufferedInputStream을 생성한다.
	public BufferedInputStream(InputStream in, int size)
	in을 이용해 크기가 size인 버퍼를 갖는 BufferedInputStream을 생성한다.
BufferedOutputStream()	public BufferedOutputStream(OutputStream out)
	out을 이용해 크기가 8192인 버퍼를 갖는 BufferedOutputStream을 생성한다.
	public BufferedOutputStream(OutputStream out, int size)
	out을 이용해 크기가 size인 버퍼를 갖는 BufferedOutputStream을 생성한다.

보조 스트림 활용

Buffered 계열

BufferedReader & BufferedWriter

메서드 명	선언부와 설명
BufferedReader()	public BufferedReader(Reader in)
	in을 이용해 크기가 8192인 버퍼를 갖는 BufferedReader를 생성한다.
	public BufferedReader(Reader in, int sz)
	in을 이용해 크기가 sz인 버퍼를 갖는 BufferedReader를 생성한다.
BufferedWriter()	public BufferedWriter(Writer out)
	out을 이용해 크기가 8192인 버퍼를 갖는 BufferedWriter를 생성한다.
	public BufferedWriter(Writer out, int sz)
	out을 이용해 크기가 sz인 버퍼를 갖는 BufferedWriter를 생성한다.

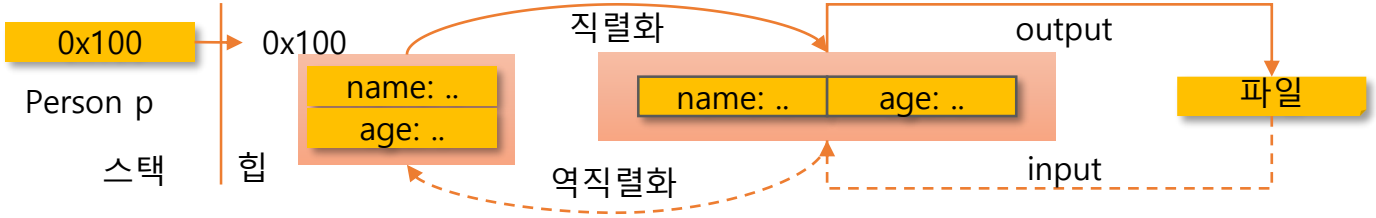
BufferedReader : readLine() → 줄 단위로 데이터를 읽어 들임

```
File src = new File("./.project");
try (BufferedReader br = new BufferedReader(new FileReader(src));) {
  String line = null;
  while ((line = br.readLine()) != null) {
    System.out.println(line);
  }
} catch (IOException e) {
  e.printStackTrace();
}
```

보조 스트림 활용

객체 직렬화(Serialization)

- 객체를 저장하거나 네트워크로 전송하기 위해 연속적인 데이터로 변환하는 것
- 반대의 경우는 역 직렬화 (deserialization)



- 직렬화 되기 위한 조건
  - Serializable 인터페이스를 구현할 것
  - 클래스의 모든 멤버가 Serializable 인터페이스를 구현해야 함
  - 직렬화에서 제외하려는 멤버는 transient 선언

```
class Person implements Serializable {
  private String name;
  private int age;

  private transient String ssn;
  private LoginInfo lInfo;
}
```

직렬화 필수 조건

직렬화 제외

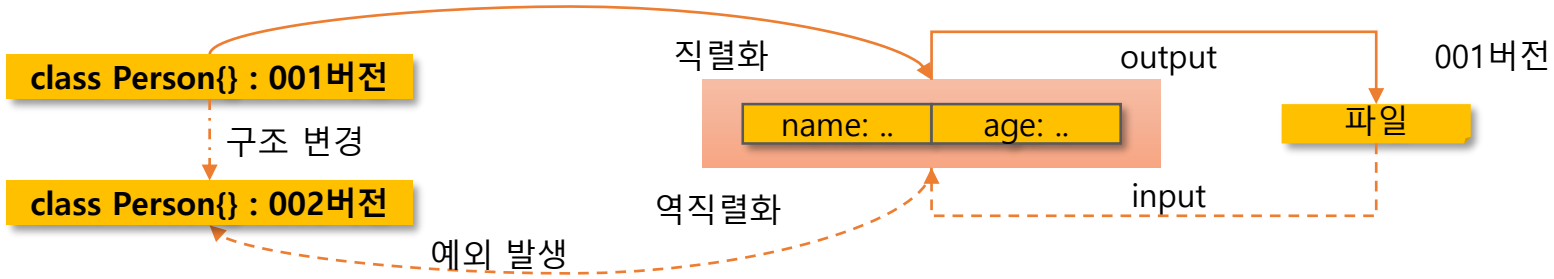
직렬화 필요

보조 스트림 활용

객체 직렬화(Serialization)

serialVersionUID

- 클래스의 변경 여부를 파악하기 위한 유일 키



- 직렬화 할 때의 UID와 역 직렬화 할 때의 UID가 다를 경우 예외 발생
- 직렬화 되는 객체에 UID가 설정되지 않았을 경우 컴파일러가 자동 생성
  (멤버 변경으로 인한 컴파일 시마다 변경 → InvalidClassException 초래)
- 직렬화 되는 객체에 대해서 serialVersionUID 설정 권장

ObjectInputStream, ObjectOutputStream

메서드 명	선언부와 설명
ObjectOutputStream ()	public ObjectOutputStream(OutputStream out) throws IOException
	out을 이용해 ObjectOutputStream 객체를 생성한다.
writeObject ()	public final void writeObject(Object obj) throws IOException
	obj를 직렬화해서 출력한다.

메서드 명	선언부와 설명
ObjectInputStream ()	public ObjectInputStream(InputStream in) throws IOException
	in을 이용해 ObjectInputStream 객체를 생성한다.
readObject ()	public final Object readObject() throws IOException, ClassNotFoundException
	직렬화된 데이터를 역직렬화해서 Object로 리턴한다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미