



I/O Programming

김기정 (bangry313@gmail.com)

목차 (Table of Contents)

1. Java I/O 소개
2. Java I/O 프로그래밍



1. Java I/O 소개

1.1 Java I/O 개요

1.2 Java I/O API

1.1 Java I/O 개요

✓ 입력(Input) 정의

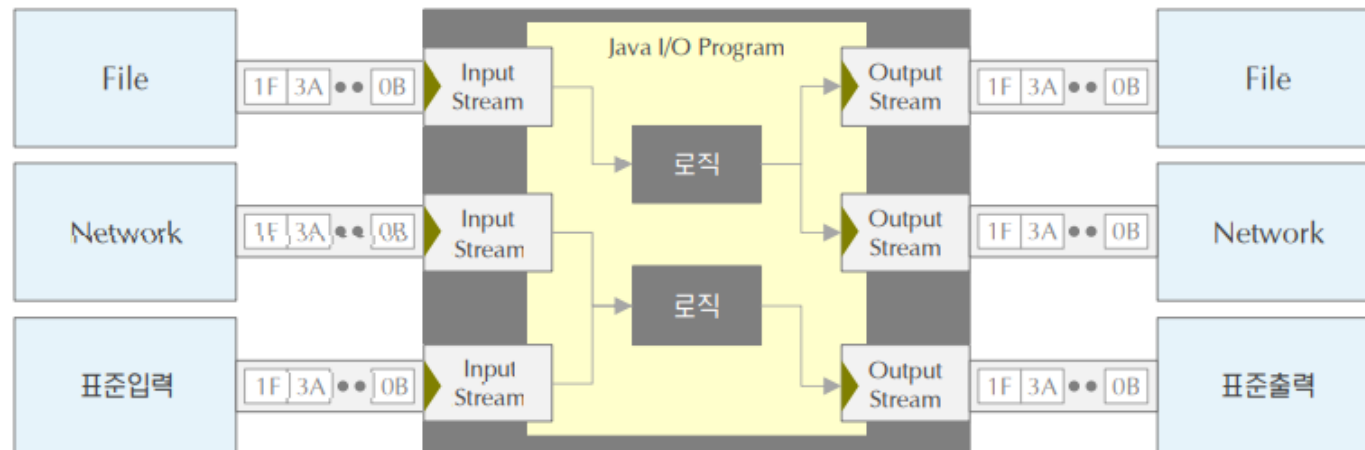
- 프로그램에서 특정 출발지(키보드, 파일, 메모리, 네트워크 등)로부터 데이터를 읽어 들이는 것을 입력이라 한다.

✓ 출력(Output) 정의

- 프로그램에서 특정 도착지(모니터, 파일, 메모리, 네트워크 등)로 데이터를 출력하는 것을 출력이라 한다.

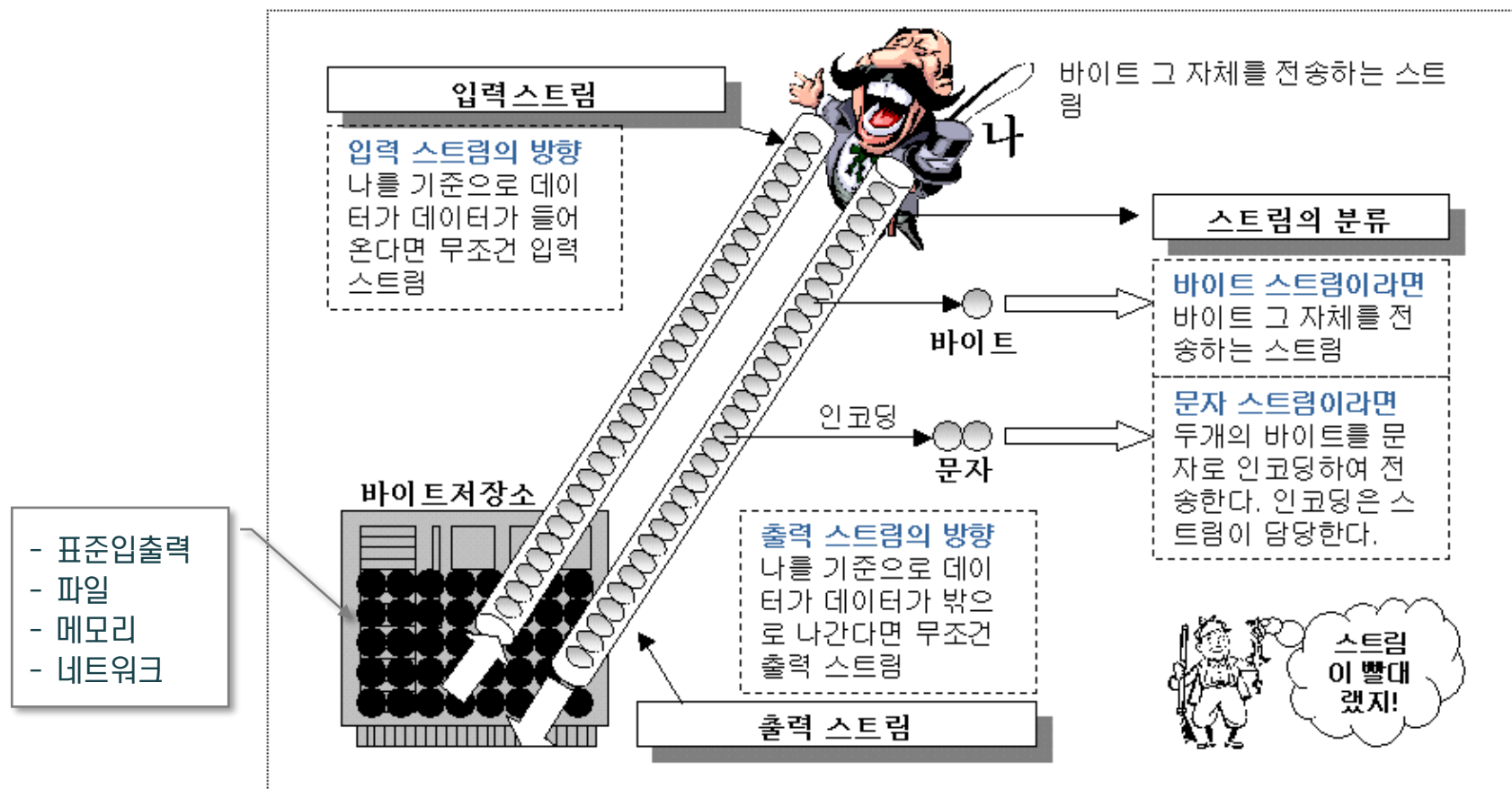
✓ Java는 키보드, 모니터, 파일, 네트워크 등으로 부터 데이터 입출력을 위해 Stream(시냇물) 개념을 추상화한 Stream API 를 사용한다.

- 프로그램 기준으로 데이터를 읽기 위해 입력 스트림, 데이터를 출력하기 위해 출력 스트림을 사용하며 프로그램이 다른 프로그램과 데이터를 입출력 하려면 양쪽 모두 입력 스트림과 출력스트림이 필요하다.
- 어떤 데이터를 입출력 하느냐에 따라 스트림은 2가지로 구분한다.
 - 바이트 스트림 : 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 입출력 할 때 사용
 - 문자 스트림 : 문자만 입출력 할 때 사용



1.2 Java I/O API (1/3) - 특징

- ✓ 입출력 대상에 상관 없이 **일관된 방법**으로 데이터를 읽고, 쓸 수 있다
- ✓ 시냇물이 **단방향** 이듯이 입력 스트림 클래스들과 출력 스트림 클래스들을 별도로 제공한다
- ✓ 스트림은 **FIFO**(First In First Out) 구조이다



1.2 Java I/O API (2/3) – 분류

✓ 입출력 방향에 따른 분류

- 입력스트림 - InputStream, FileInputStream 등
- 출력스트림 - OutputStream, FileOutputStream 등

✓ 입출력 데이터 종류에 따른 분류

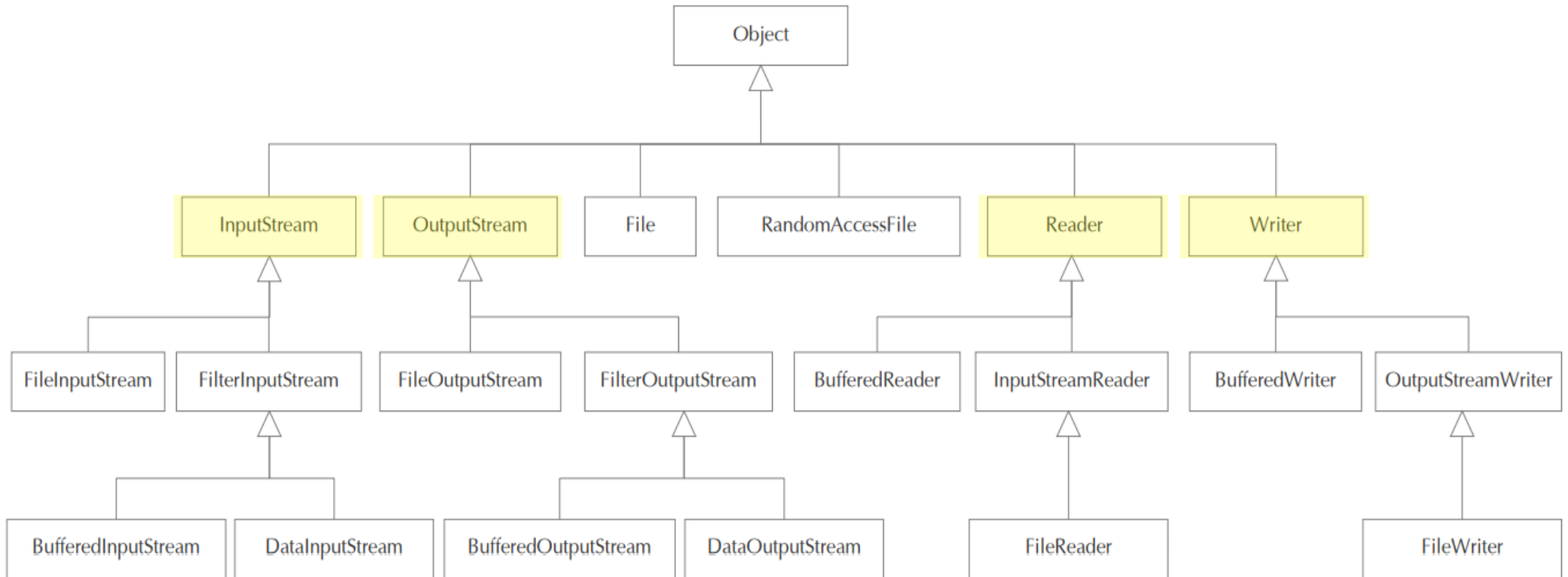
- 바이트(byte)스트림 - 데이터를 바이트 단위 읽고 쓰기
- 문자(character)스트림 - 바이트 데이터를 문자(인코딩/디코딩) 단위로 읽고 쓰기

✓ 입출력 스트림의 용도에 따른 분류

- 노드(Node)스트림 - 단순한 입출력만 담당
- 필터(Filter)스트림 - 스트림의 데이터를 조작

1.2 Java I/O API (3/3) – 구조

- ✓ 데이터 입출력과 관련된 라이브러리를 java.io 패키지에서 제공하고 있다.
- ✓ System 클래스에는 키보드와 모니터 같은 표준 입출력 장치를 다루는 입출력 메서드를 제공한다.
- ✓ 기본적인 입출력 스트림 클래스와 기능을 확장할 수 있는 필터 스트림 클래스가 존재한다.



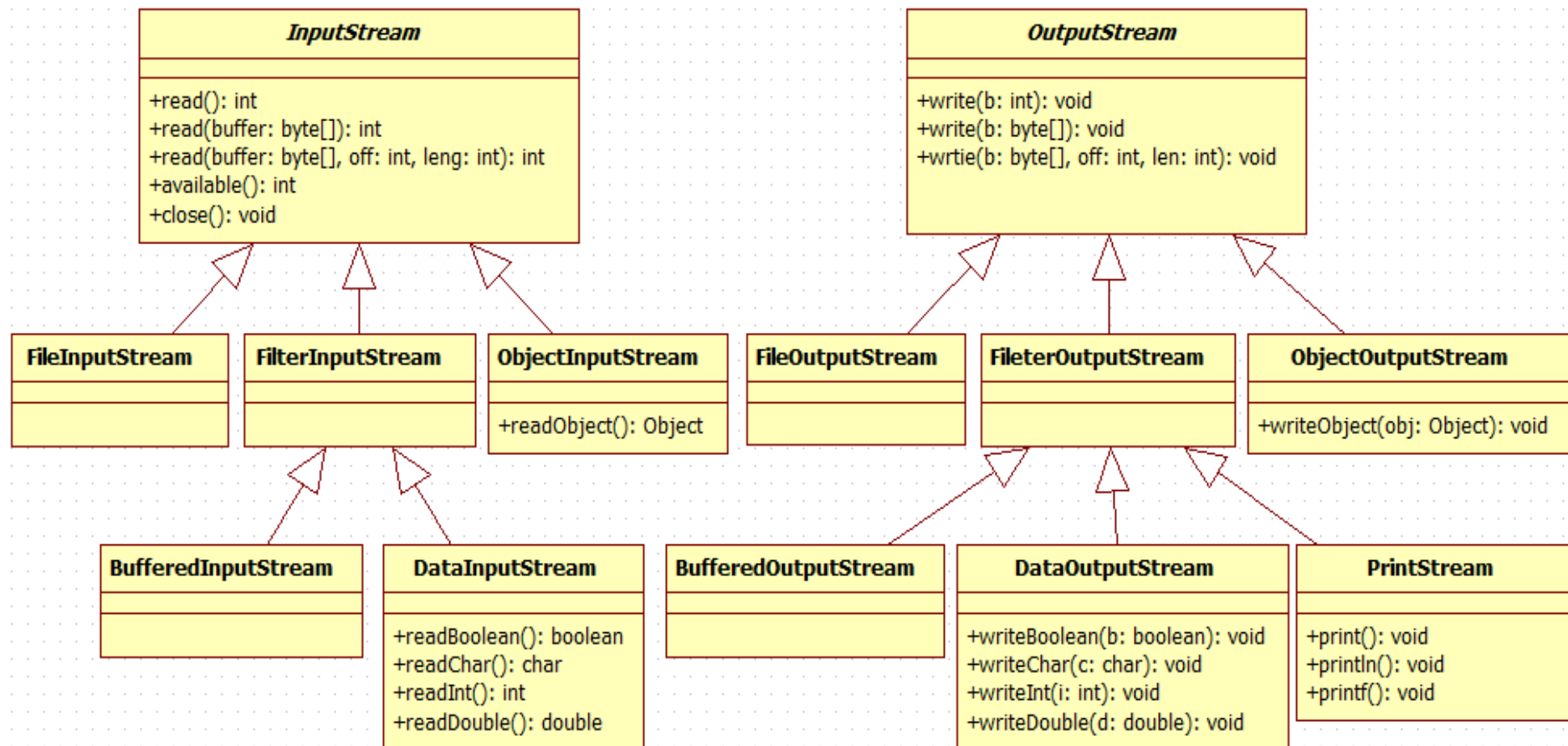


2. Java I/O 프로그래밍

- 2.1 바이트 스트림
- 2.2 문자 스트림
- 2.3 객체 직렬화와 객체 스트림

2.1 바이트 스트림 (1/15) - 개요

- ✓ 바이트 스트림 클래스는 1바이트 단위로 입출력을 처리하는 스트림 클래스이다.
- ✓ 최상위 클래스인 *InputStream*과 *OutputStream*은 추상 클래스이며, 이를 확장한 다양한 자식클래스들을 제공한다.
 - 파일, 바이트 배열, 부가적인 기능이 추가되어 있는 필터 스트림 클래스, 객체를 처리할 수 있는 입출력 스트림 클래스가 제공된다



2.1 바이트 스트림 (2/15) – InputStream / OutputStream

- ✓ *InputStream*과 *OutputStream* 클래스는 바이트 단위로 입출력을 처리하는 최상위 추상 클래스이다
- *InputStream*에는 모든 바이트 입력스트림이 기본적으로 제공해야 하는 메소드가 정의 되어 있다.
 - 입력을 위한 read() 메소드와, 읽을 수 있는 바이트 수를 반환하는 available() 메소드 있다.
 - *OutputStream*에는 모든 바이트 출력스트림이 기본적으로 제공해야 하는 메소드가 정의 되어 있다.
 - 출력을 위한 write() 메소드와, 버퍼에 남은 데이터를 출력하는 flush() 메소드가 있다 .
 - 스트림을 통한 입출력이 끝난 후에는 close() 메소드를 호출하여 스트림을 종료한다

InputStream 주요 메소드	설명
int available()	현재 읽을 수 있는 바이트 수를 반환합니다.
int read()	입력 스트림에서 한 바이트를 읽어 int 값으로 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int read(byte buf[])	입력 스트림에서 buf[] 크기만큼을 읽어 buf에 저장하고 읽은 바이트 수를 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int skip(long numBytes)	numBytes로 지정된 바이트를 무시하고 무시된 바이트 수를 반환합니다.

OutputStream 주요 메소드	설명
void flush()	버퍼에 남은 출력 스트림을 출력합니다.
void write(int i)	정수 i의 하위 8비트를 출력합니다.
void write(byte buf[])	buf 배열의 내용을 출력합니다.
void write(byte buf[], int index, int size)	buf 배열의 index 위치부터 size만큼의 바이트를 출력합니다.

2.1 바이트 스트림 (3/15) – FileInputStream / FileOutputStream

- ✓ FileInputStream과 FileOutputStream 클래스는 InputStream 또는 OutputStream을 상속한다
- ✓ FileInputStream은 파일을 바이트 단위로 읽고, FileOutputStream은 파일에 바이트 단위로 출력한다
- ✓ FileInputStream 또는 FileOutputStream은 파일이름 또는 File 객체로 스트림 객체를 생성할 수 있다
- ✓ FileOutputStream를 생성할 때, append 매개변수의 값을 true로 설정하면 기존 파일에 추가하여 출력한다



FileInputStream 생성자	설명
FileInputStream(String name)	name에 해당하는 파일로부터 바이트 단위로 읽어 들이는 스트림 객체를 생성합니다.
FileInputStream(File file)	file 객체로 지정한 파일로부터 바이트 단위로 읽어 들이는 스트림 객체를 생성합니다.

FileOutputStream 생성자	설명
FileOutputStream(String name)	name에 해당하는 파일에 대한 출력 스트림을 생성합니다.
FileOutputStream(String name, boolean append)	지정한 파일로 출력 스트림을 생성합니다. Append 변수 값이 true로 설정되면 기존 파일에 이어서 쓰게 됩니다.
FileOutputStream(File file)	File 객체로 지정된 파일에 대한 출력 스트림을 생성합니다.

2.1 바이트 스트림 (4/15) – File 클래스 (1/8)

- ✓ File 클래스는 스트림 클래스가 아니고 파일과 디렉토리를 다루는 클래스이다
- ✓ 파일의 기본 정보(크기, 변경날짜 등)를 제공하거나, 관리할 수 있는 기능을 제공한다
- ✓ 파일의 복사, 이름 변경 등의 조작을 할 경우에만 사용되며, 파일 데이터를 입출력 하기 위해서는 스트림 클래스를 사용한다

File 생성자	설명
File (String pathName)	주어진 경로명을 가지고 새로운 File 객체를 생성한다.
File (String parent, String child)	주어진 두개의 경로명을 가지고 새로운 File 객체를 생성한다. parent는 디렉토리이며, child는 디렉토리 또는 파일일 수 있다.
File (File parent, String child)	주어진 File 객체와 문자열을 이용하여 새로운 File 객체를 생성한다. parent 객체는 디렉토리 이며, child는 디렉토리 또는 파일일 수 있다.

2.1 바이트 스트림 (5/15) – File 클래스 (2/8)

✓ 주요 메소드

메소드	설명
String getName()	경로명이 나타내는 파일 또는 디렉터리의 이름을 얻는다. 이 이름은 경로명에 있는 마지막 이름이다.
String getParent()	경로명의 부모 경로명을 얻는다. 부모 경로명은 경로명에 있는 마지막 이름을 제외한 나머지 이름들을 포함하고 있다.
File getParentFile()	경로명의 부모 경로에 대한 File 객체를 얻는다. 부모 경로명은 경로명에 있는 마지막 이름을 제외한 나머지 이름들을 포함하고 있다.
String getPath()	경로명을 얻는다.
boolean isAbsolute()	절대경로명인지를 얻는다. 유닉스 시스템상에서는 prefix가 “/”를 포함하고 있을 경우, 윈도우 시스템상에서는 드라이브문자 또는 prefix가 “ www ”를 포함하고 있을 경우 절대경로이다.
String getAbsolutePath()	절대 경로명을 얻는다.
URL toURL()	경로명을 URL로 변환하여 얻는다.

2.1 바이트 스트림 (6/15) – File 클래스 (3/8)

✓ 주요 메소드

메소드	설명
boolean canRead()	파일이 읽기 가능한지 여부 얻는다.
boolean canWrite()	파일이 쓰기 가능한지 여부 얻는다.
boolean exists()	파일이 존재하는지 여부 얻는다.
boolean isDirectory()	디렉토리인지 여부 얻는다.
boolean isFile()	파일인지 여부 얻는다.
boolean isHidden()	숨겨진 파일인지 여부 얻는다.
long lastModified()	마지막으로 변경된 시간을 얻는다. 시간은 그리니찌표준시간(00:00:00 GMT, January 1, 1970)을 기준으로 경과된 밀리초를 나타낸다.
long length()	파일의 크기를 얻는다.
boolean createNewFile()	경로명에 해당하는 파일이 존재하지 않는 경우 빈 새로운 파일 생성.

2.1 바이트 스트림 (7/15) – File 클래스 (4/8)

✓ 주요 메소드

메소드	설명
<code>boolean delete()</code>	경로명이 가르키는 파일이나 디렉토리 삭제. 이때 비어있지 않은 디렉토리는 삭제할 수 없다.
<code>String[] list()</code>	경로명이 나타내는 디렉토리내의 파일과 디렉토리 이름에 대한 문자열 배열을 얻습니다.
<code>File[] listFiles()</code>	경로명이 나타내는 디렉토리내의 파일과 디렉토리 이름에 대한 파일 객체 배열을 얻습니다.
<code>boolean mkdir()</code>	경로명에 해당하는 디렉토리를 생성한다.
<code>boolean mkdirs()</code>	부모 디렉토리가 없으면 부모 디렉토리까지 생성한다.
<code>boolean renameTo(File file)</code>	파일의 이름을 변경한다.
<code>static File[] listRoots()</code>	사용 가능한 파일 시스템의 루트를 얻는다.

2.1 바이트 스트림 (8/15) – File 클래스 (5/8)

✓ 파일 정보 조회

```
1 public class FileInfoDisplay {
2
3     public static void main(String[] args) throws IOException {
4
5         System.out.println("user.dir : " + System.getProperty("user.dir"));
6
7         File file = new File("sample.txt");
8
9         System.out.println("getName() : " + file.getName());
10        System.out.println("getParent() : " + file.getParent());
11        System.out.println("getPath() : " + file.getPath());
12        System.out.println("getAbsolutePath() : " + file.getAbsolutePath());
13        System.out.println("getCanonicalPath() : " + file.getCanonicalPath());
14        System.out.println("length() : " + file.length() + " bytes");
15        System.out.println("isFile() : " + file.isFile());
16        System.out.println("isDirectory() : " + file.isDirectory());
17    }
18 }
```

FileInfoDisplay.java

코드설명

[Line5] 시스템 프로퍼티 user.dir 값을 조회하여 화면에 출력합니다.

[Line7] sample.txt 파일에 대한 객체를 생성합니다.

[Line 9~16] File 객체의 다양한 메소드를 실행하여 결과를 출력합니다.

실행결과

```
user.dir : C:\JavaIOExamples
getName() : sample.txt
getParent() : null
getPath() : sample.txt
getAbsolutePath() : C:\...\sample.txt
getCanonicalPath() : C:\...\sample.txt
length() : 18 bytes
isFile() : true
isDirectory() : false
```

2.1 바이트 스트림 (9/15) – File 클래스 (6/8)

✓ 파일 목록 조회

```
1 public class FileList {
2
3     public static void main(String[] args) {
4
5         File parentDir = new File("./");
6
7         if (parentDir.isDirectory()) {
8             File[] files = parentDir.listFiles();
9             for (File file : files) {
10
11                 String prefix = file.isDirectory() ? "[D] " : "[F] ";
12                 System.out.println(prefix + file.getName());
13             }
14         }
15     }
```

FileList.java

코드설명

[Line5] 지정한 경로를 나타내는 파일 객체를 생성합니다. (여기서는 현재 디렉토리)

[Line7] 지정한 경로가 디렉토리인지 여부를 체크합니다.

[Line8] 지정한 경로의 모든 파일을 파일객체 배열로 반환합니다.

[Line9~12] 반환 받은 파일객체 배열을 반복문을 사용하여 화면에 출력합니다. 이때, 디렉토리는 '[D]', 파일은 '[F]' 를 붙여줍니다.

실행결과

```
[F] .classpath
[F] .project
[D] .settings
[D] bin
[D] src
```

2.1 바이트 스트림 (10/15) – File 클래스 (7/8)

✓ 파일 생성과 삭제

```
1 public class FileCreateAndDelete {
2
3     public static void main(String[] args) throws IOException {
4
5         File file = new File("newFile.txt");
6
7         boolean isCreated = file.createNewFile();
8         System.out.println("new file is created : " + isCreated);
9         System.out.println("file.exists() : " + file.exists());
10
11         boolean isDeleted = file.delete();
12         System.out.println("file is deleted : " + isDeleted);
13         System.out.println("file.exists() : " + file.exists());
14     }
15 }
```

FileCreateAndDelete.java

코드설명

[Line5] newFile.txt 파일을 나타내는 파일 객체를 생성합니다.

[Line8] 새로운 파일을 생성합니다.

[Line13] 파일을 삭제합니다.

실행결과

```
new file is created : true
file.exists() : true
file is deleted : true
file.exists() : false
```


2.1 바이트 스트림 (11/15) – File 클래스 (8/8)

✓ 디렉토리 생성

```
1 public class DirectoryCreation {
2
3     public static void main(String[] args) throws IOException {
4
5         String baseDir = "./";
6
7         File dir1 = new File(baseDir + "dir1");
8         boolean isMade = dir1.mkdir();
9         System.out.println("new directory is made : " + isMade);
10
11         File dir2 = new File(baseDir + "parent/dir2");
12         isMade = dir2.mkdirs();
13         System.out.println("new directory is made : " + isMade);
14     }
15 }
```

DirectoryCreation.java

코드설명

[Line7] dir1 이라는 파일 객체를 생성합니다. File 객체는 파일이나 디렉토리 모두를 나타낼 수 있습니다.

[Line8] dir1 디렉토리를 생성합니다. 정상적으로 생성되었는지 여부를 isMade 변수에 할당합니다.

[Line11] 계층 구조를 가진 디렉토리를 생성합니다. mkdirs() 메소드는 부모 디렉토리가 없는 경우 부모 디렉토리까지 생성합니다.

실행결과

```
new directory is made : true
new directory is made : true
```

2.1 바이트 스트림 (12/15) – FilterInputStream/FilterOutputStream

- ✓ 노드 스트림은 바이트 단위로 데이터를 읽고, 쓰는 기능만 제공한다
- ✓ 노드 스트림에 특정 기능을 가진 필터 스트림을 연결해서 원하는 동작을 지원한다
 - 효율적인 입출력, 기본 자료형에 대한 입출력, 다중연결, 라인 넘버링 등

클래스	설명
FilterInputStream / FilterOutputStream	필터 스트림 클래스들의 최상위 추상 클래스
LineNumberInputStream	라인 번호 입력 필터 스트림 클래스
DataInputStream / DataOutputStream	자바 기본타입으로 읽고, 쓸 수 있는 필터 스트림 클래스
BufferedInputStream / BufferedOutputStream	스트림에 버퍼를 추가한 필터 스트림 클래스
PrintStream	각각의 자료형을 문자열로 출력 – 주로 디버깅 때 사용

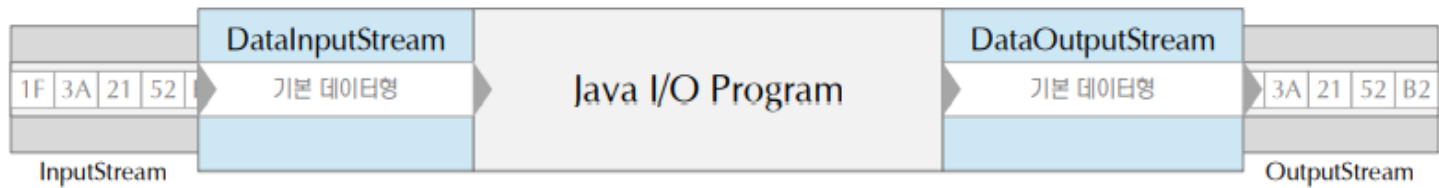
2.1 바이트 스트림 (13/15) – BufferedInputStream / BufferedOutputStream

- ✓ 입출력의 효율성을 높이기 위해 버퍼를 내장한 필터 스트림 클래스이다
- ✓ BufferedInputStream 클래스를 사용하면 읽기 동작이 있을 때 마다 목적지로부터 1바이트 씩 읽는 것이 아니라, 미리 버퍼(512byte)에 담아 놓 데이터를 읽어 들이므로 효율적이다
- ✓ BufferedOutputStream 클래스는 데이터를 출력할 때, 먼저 내장한 내부 버퍼에 출력이 되고, 버퍼가 꽉 차거나, flush(), close() 메소드가 호출될 때 내부 버퍼의 내용이 출력되게 된다

생성자	설명
BufferedInputStream(InputStream in)	주어진 바이트 입력 스트림에 대한 BufferedInputStream 객체 생성 내부 버퍼의 크기는 디폴트 8KB로 설정된다.
BufferedInputStream(InputStream in, int bufferSize)	주어진 바이트 입력 스트림에 대한 BufferedInputStream 객체 생성 내부 버퍼의 크기는 주어진 bufferSize로 설정된다.
BufferedOutputStream(OutputStream out)	주어진 바이트 출력 스트림에 대한 BufferedOutputStream 객체 생성 내부 버퍼의 크기는 디폴트 512 바이트로 설정된다.
BufferedOutputStream(OutputStream out, int bufferSize)	내부 버퍼의 크기는 주어진 bufferSize로 설정된다.

2.1 바이트 스트림 (14/15) – DataInputStream / DataOutputStream

- ✓ 바이트 단위의 입출력 기능 뿐만 아니라, 자바에서 제공하는 기본 데이터 타입별로 직접 읽고 쓸 수 있는 기능을 제공하는 필터 스트림이다



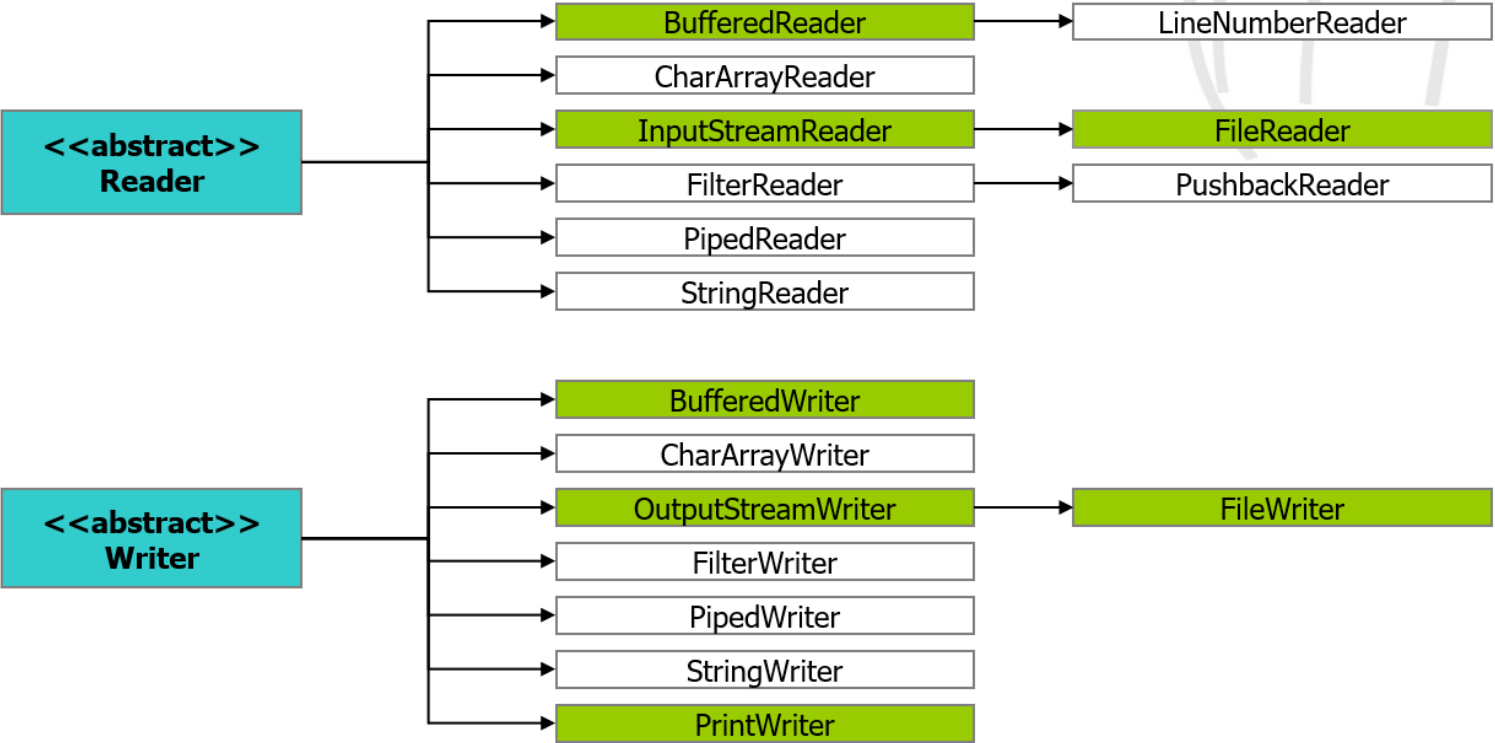
DataInputStream 메소드	설명	DataOutputStream 메소드	설명
boolean readBoolean()	스트림에서 boolean 값을 읽습니다.	void writeBoolean()	boolean 값을 스트림에 기록합니다.
byte readByte()	스트림에서 byte 값을 읽습니다.	void writeByte()	byte 값을 스트림에 기록합니다.
char readChar()	스트림에서 char 값을 읽습니다.	void writeChar()	char 값을 스트림에 기록합니다.
double readDouble()	스트림에서 double 값을 읽습니다.	void writeDouble()	double 값을 스트림에 기록합니다.
float readFloat()	스트림에서 float 값을 읽습니다.	void writeFloat()	float 값을 스트림에 기록합니다.
long readLong()	스트림에서 long 값을 읽습니다.	void writeLong()	long 값을 스트림에 기록합니다.
short readShort()	스트림에서 short 값을 읽습니다.	void writeShort()	short 값을 스트림에 기록합니다.
int readInt()	스트림에서 int 값을 읽습니다.	void writeInt()	int 값을 스트림에 기록합니다.
int readUTF()	스트림에서 UTF-8 문자열을 읽습니다.	void writeUTF()	UTF-8 문자열을 스트림에 기록합니다.

2.1 바이트 스트림 (15/15) – PrintStream

- ✓ System.out을 통해서 모니터(표준출력)로 출력하기 위해 주로 사용된다
 - 디버깅용으로 사용
- ✓ 기본 데이터 타입들을 OS의 문자 인코딩 방식에 맞춰 문자열로 변환하여 출력하는 여러 종류의 메소드를 제공하고 있다

2.2 문자 스트림 (1/6) – 개요

- ✓ 자바는 문자를 유니코드로 처리하기 때문에 유니코드 문자를 바이트 기반 입출력 클래스로 처리하는 것은 불편하다
- ✓ 텍스트를 입출력하기 위해 바이트 입출력 스트림인 `InputStream`과 `OutputStream`에 대응하는 문자 입출력 스트림으로 `Reader` 클래스와 `Writer` 클래스를 제공한다
 - 입출력 되는 단위가 문자인 것을 제외하고는 바이트 입출력 스트림과 사용법이 동일하다.
- ✓ `Reader`와 `Writer` 계열의 문자 기반 입출력 클래스를 사용하면, 바이트 대신 문자 단위로 입출력을 처리할 수 있다
- ✓ 모든 문자 기반 입출력 클래스는 추상 클래스인 `Reader`와 `Writer` 클래스를 상속받아 구현된다



2.2 문자 스트림 (2/6) – Reader, Writer 추상클래스

- ✓ Stream 계열 클래스가 바이트 단위로 처리하는 반면, Reader, Writer 추상클래스는 문자 단위로 입출력 한다
- ✓ Reader의 read() 메소드는 문자 단위로 데이터를 읽는다
- ✓ Reader는 차단(blocking) 없이 읽을 수 있는 데이터가 있는지 확인하는 ready() 메소드를 제공한다
- ✓ Writer는 문자 단위로 데이터를 쓰는 write() 메소드와, 체인형식으로 쓸 수 있는 append() 메소드를 제공한다

Reader 주요 메소드	설명
boolean ready()	스트림에 읽을 수 있는 데이터가 있는지 여부를 반환합니다.
int read()	입력 스트림에서 하나의 문자를 읽어 int 형으로 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
int read(char cbuf[])	입력 스트림에서 cbuf[] 크기만큼을 읽어 cbuf에 저장하고 읽은 문자 수를 반환합니다. 더 이상 읽을 내용이 없을 경우, -1을 반환합니다.
long skip(long n)	n으로 지정된 문자만큼을 무시하고, 실제로 무시된 문자의 수를 반환합니다.

Writer 주요 메소드	설명
void flush()	버퍼에 남은 출력 스트림을 출력합니다.
void write(int c)	하나의 문자를 출력합니다. (int 형의 하위 16비트만 사용합니다.)
void write(char cbuf[])	cbuf 배열의 내용을 출력합니다.
void write(char cbuf[], int off, int len)	cbuf 배열의 off 위치부터 len 만큼의 문자를 출력합니다.
Writer append(CharSequence csq)	csq 에 지정된 문자열을 출력합니다. (내부적으로는 write() 메소드 호출과 동일함) 출력이 끝나면 Writer 객체를 다시 리턴하여, 메소드 체인을 구성할 수 있게 합니다.

2.2 문자 스트림 (3/6) – InputStreamReader / OutputStreamWriter

- ✓ InputStreamReader와 OutputStreamWriter는 바이트 스트림을 문자 스트림으로 변환하는 입출력 클래스이다
- ✓ InputStreamReader 생성자를 통해, 바이트 스트림을 문자로 읽어들이기 때 사용할 문자 인코딩을 설정한다
- ✓ OutputStreamWriter 생성자를 통해, 문자를 바이트 스트림으로 출력할 때 사용할 문자 인코딩을 설정한다
- ✓ 문자 인코딩을 설정하지 않으면, 시스템의 디폴트 문자 인코딩을 사용한다

InputStreamReader 생성자	설명
InputStreamReader(InputStream in)	in 객체를 받아 Reader 객체를 생성합니다.
InputStreamReader(InputStream in, String charsetName)	in 객체와 charsetName 을 받아, 해당 문자 인코딩을 사용하는 Reader 객체를 생성합니다. 문자 인코딩 설정은 인코딩 이름을 나타내는 문자열이나 Charset, CharsetDecoder 객체를 사용할 수 있습니다.

OutputStreamWriter 생성자	설명
OutputStreamWriter(OutputStream out)	out 객체를 받아 Writer 객체를 생성합니다.
OutputStreamWriter (OutputStream out, String charsetName)	out 객체와 charsetName 을 받아, 해당 문자 인코딩을 사용하는 Reader 객체를 생성합니다. 문자 인코딩 설정은 인코딩 이름을 나타내는 문자열이나 Charset, CharsetEncoder 객체를 사용할 수 있습니다.

2.2 문자 스트림 (4/6) – FileReader / FileWriter

- ✓ FileReader 클래스는 파일을 문자 단위로 읽는 클래스로 InputStreamReader 클래스를 상속한다
- ✓ FileWriter 클래스는 파일에 문자단위로 쓰는 클래스로 OutputStreamWriter 클래스를 상속한다
- ✓ FileReader & FileWriter는 내부적으로는 파일 입출력 스트림을 Reader, Writer로 변환하여 처리한다
- ✓ FileReader & FileWriter는 문자로 구성된 텍스트 파일을 읽거나 저장하기에 적합하다

```
1 public class TextPrinter {
2     public static void main(String[] args) throws IOException {
3
4         System.out.print("Enter text filename : ");
5         String filename = new Scanner(System.in).nextLine();
6
7         FileReader reader = new FileReader(filename);
8         int ch = 0;
9         while ((ch = reader.read()) != -1) {
10             System.out.print((char) ch);
11         }
12
13         reader.close();
14     }
15 }
```

TextPrinter.java

코드설명

[Line5] 텍스트 파일명을 입력 받습니다.
[Line7] 파일에 대한 FileReader 객체를 생성합니다.
[Line9~11] 한 글자씩 읽습니다. 읽은 값이 EOF(-1)이 아닐 때 까지 반복하면서 읽은 값을 문자형으로 변환하여 화면에 출력합니다.
[Line13] 입력 스트림을 닫습니다.

실행결과

Enter text filename : poem.txt
서시 - 윤동주
죽는 날까지 하늘을 우러러
한점 부끄럼이 없기를
앞세에 이는 바람에도
나는 괴로워했다.
...

2.2 문자 스트림 (5/6) – BufferedReader / BufferedWriter

- ✓ 입력과 출력 작업이 빠른 속도로 일어날 때 병목현상이 발생할 수 있다
- ✓ BufferedReader와 BufferedWriter는 입출력 시 병목현상을 줄이기 위해서 내부적으로 버퍼를 사용한다
- ✓ 네트워크 프로그래밍의 경우 write() 호출 후, flush() 를 호출하여 버퍼에 남은 데이터를 전송해야 한다
 - flush() 는 버퍼에 남은 내용을 모두 출력한 후 버퍼를 비우는 메소드이다



```
1 public class BufferedTextPrinter {
2     public static void main(String[] args) throws IOException {
3
4         System.out.print("Enter text filename : ");
5         String filename = new Scanner(System.in).nextLine();
6
7         BufferedReader reader = new BufferedReader(new FileReader(filename));
8         String readLine = null;
9         while ((readLine = reader.readLine()) != null) {
10             System.out.println(readLine);
11         }
12
13         reader.close();
14     }
15 }
```

BufferedTextPrinter.java

코드설명

[Line5] 텍스트 파일명을 입력 받습니다.
[Line7] 파일에 대한 FileReader 객체를 생성합니다. 생성한 Reader 객체를 사용하여 BufferedReader 객체를 생성합니다.
[Line9~11] 한 라인씩 읽습니다. 읽은 문자열이 null (EOF)이 아닐 때 까지 반복하면서 읽은 문자열을 화면에 출력합니다.
[Line13] 입력 스트림을 닫습니다.

실행결과

Enter text filename : poem.txt
서시 - 윤동주
죽는 날까지 하늘을 우러러
한점 부끄럼이 없기를
웁새에 이는 바람에도
나는 괴로워했다.
...

2.2 문자 스트림 (6/6) – PrintWriter

- ✓ PrintWriter 클래스는 객체의 내용을 텍스트로 출력하는데 특화된 클래스이다
- ✓ PrintStream 클래스에서 제공하는 대부분의 출력 관련 메소드를 유사하게 제공한다
- ✓ 이 클래스의 모든 메소드는 예외를 발생시키지 않습니다. 예외발생 여부는 checkError() 메소드로 확인한다
- ✓ format()이나 printf() 메소드를 사용하면 다양한 형식으로 객체의 내용을 출력할 수 있다

```
1 public class GugudanPrinter {
2
3     public static void main(String[] args) throws FileNotFoundException {
4
5         System.out.print("Enter number : ");
6         int dan = new Scanner(System.in).nextInt();
7
8         File output = new File("gugudan_" + dan + ".txt");
9         PrintWriter writer = new PrintWriter(output);
10
11         for ( int i = 1; i <= 9; i++ ) {
12             writer.printf("%d * %d = %d\n", dan, i, (dan * i));
13         }
14
15         System.out.println("gugudan file saved.");
16         writer.close();
17     }
18 }
```

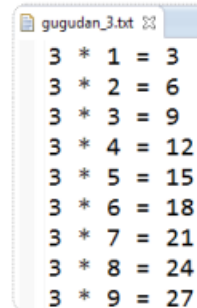
GugudanPrinter.java

코드설명

[Line6] 출력할 단을 입력합니다.
[Line8] 구구단을 출력할 파일 객체를 생성합니다.
[Line9] 파일에 텍스트로 출력하는 PrintWriter 객체를 생성합니다.
[Line11~13] 구구단 형식에 맞게 출력합니다.
[Line16] 출력 스트림을 닫습니다.

실행결과

Enter number : 3
gugudan file saved.

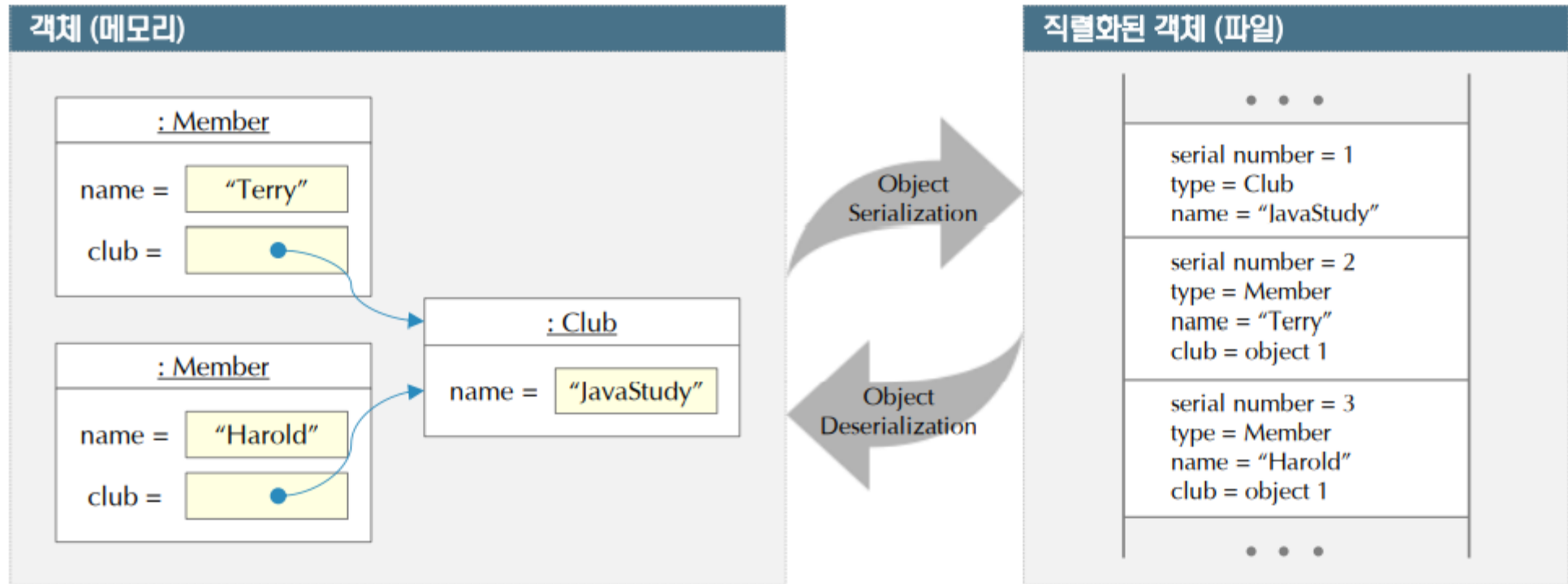


gugudan_3.txt

```
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
```

2.3 객체 직렬화와 객체 스트림 (1/2)

- ✓ 메모리상의 객체를 일련의 바이트들로 변환하는 것을 객체 직렬화(Serialization)라고 한다
- ✓ 객체를 직렬화하는 과정을 마샬링이라 하고, 역직렬화하여 객체로 변환하는 것을 언마샬링이라 한다
- ✓ 직렬화된 객체는 파일이나 네트워크로 전송할 수 있고, 전송된 데이터는 역직렬화를 통해 다시 객체로 변환된다
- ✓ `ObjectInputStream`과 `ObjectOutputStream` 클래스는 객체 직렬화 및 역직렬화를 위한 메소드를 제공한다



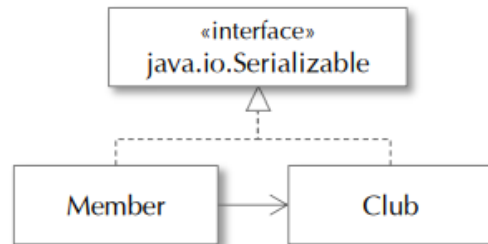
2.3 객체 직렬화와 객체 스트림 (2/2)

- ✓ 객체 직렬화는 기본 자료형이나 java.io.Serializable 인터페이스를 구현한 객체만 가능하다
- ✓ Serializable 인터페이스는 구현할 메소드가 없는 직렬화 대상을 나타내는 Mark 인터페이스이다
- ✓ ObjectInputStream의 readObject() 메소드는 입력 스트림으로부터 객체를 읽어 들인다
- ✓ ObjectOutputStream의 writeObject() 메소드는 출력 스트림으로 객체를 직렬화 하여 출력한다

```
1 Club club = new Club("JavaStudy");
2 Member member = new Member("Harold", club);
3
4 String filename = "object.out";
5
6 // save object to file
7 ObjectOutputStream oos =
8     new ObjectOutputStream(new FileOutputStream(filename));
9 oos.writeObject(member);
10
11 oos.close();
12
13 // read object from file
14 ObjectInputStream ois =
15     new ObjectInputStream(new FileInputStream(filename));
16 Member savedMember = (Member) ois.readObject();
17
18 System.out.println("member name : " + savedMember.getName());
19 System.out.println("club name : " + savedMember.getClub().getName());
20
21 ois.close();
```

ObjectInOut.java

객체 모델



코드설명

[Line1~2] 직렬화할 객체를 생성합니다.
[Line7~8] 객체를 파일로 출력하는 ObjectOutputStream 객체를 생성합니다.
[Line9] 객체를 파일로 저장합니다.
[Line11] 객체 출력 스트림을 닫습니다.
[Line14~15] 객체를 스트림으로 부터 조회하는 ObjectInputStream 객체를 생성합니다.
[Line16] 객체를 스트림에서 조회합니다.
[Line21] 객체 입력 스트림을 닫습니다.

실행결과

member name : Harold
club name : JavaStudy

End of Document

✓ Q&A



감사합니다...