

API-IO(Input스트림-Output스트림)



스트림이란?



스트림은 데이터의 흐름(흐르는 통로)를 말합니다.

- 데이터가 들어오면 입력 스트림 입니다
- 데이터가 나가면 출력 스트림 입니다
- 스트림 클래스는 크게 두 종류로 구분됩니다. (byte기반, 문자 기반)
- **바이트 기반 스트림** - 그림, 문자 등 모든 종류의 데이터를 보낼 수 있습니다
- **문자 기반 스트림** - 오직 문자만 보낼 수 있도록 특화되어 있습니다

스트림은 byte기반, 문자 기반으로 나뉜다

바이트 기반 최상위 클래스

문자 기반 최상위 클래스

	Byte Streams	Character Streams
Source Streams	InputStream	Reader
Sink Streams	OutputStream	Writer



- 모든 바이트 기반 입력 스트림은 이 클래스를 상속받아 만들어지며 하위 클래스들은 접미사로 **InputStream**이 붙습니다.
- 모든 바이트 기반 출력 스트림 클래스는 이 클래스를 상속받아 만들어지며 하위 클래스들은 접미사로 **OutputStream**이 붙습니다.

바이트 기반 하위 클래스 예시

문자 기반 하위 클래스 예시

	Byte Streams	Character Streams
File	FileInputStream FileOutputStream	FileReader FileWriter
Memory : Array	ByteArrayInputStream ByteArrayOutputStream	CharArrayReader CharArrayWriter
Memory : String	-	StringReader StringWriter
Pipe	PipedInputStream PipedOutputStream	PipedReader PipedWriter

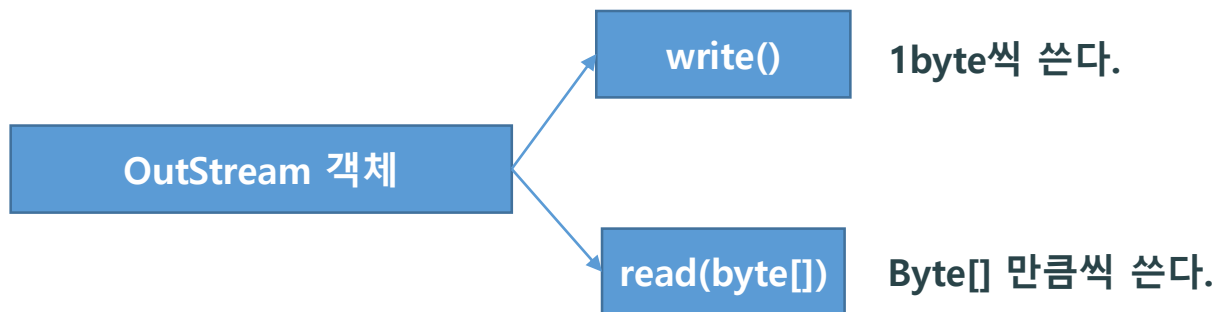
바이트 단위로 쓰는 OutputStream

OutputStream 클래스

- OutputStream은 바이트 기반 출력 스트림의 **최상위 클래스**로 **추상 클래스**입니다.

OutputStream 주요 메서드

1. write(byte b): 출력 스트림으로 1바이트를 내보냄.
2. write(byte[] b): 출력 스트림으로 주어진 바이트 배열 b의 모든 바이트들을 내보냄.



OutputStream예제

```
Scanner scan = new Scanner(System.in);
System.out.print("파일명을 입력하세요: ");
String name = scan.next();
```

```
OutputStream fos = null;
```

```
try {
    //fileOutputStream(파일을쓸경로)
    fos = new FileOutputStream("C:\\Users\\Park\\Desktop\\eclipse\\file\\" + name + ".txt");

    System.out.print("문장 입력: ");
    scan.nextLine();
    String str = scan.nextLine();

    byte[] bs = str.getBytes();

    fos.write(bs);
    System.out.println("파일이 정상적으로 저장되었습니다.");
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        fos.close();
        scan.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
```

바이트단위로 읽는 InputStream

InputStream

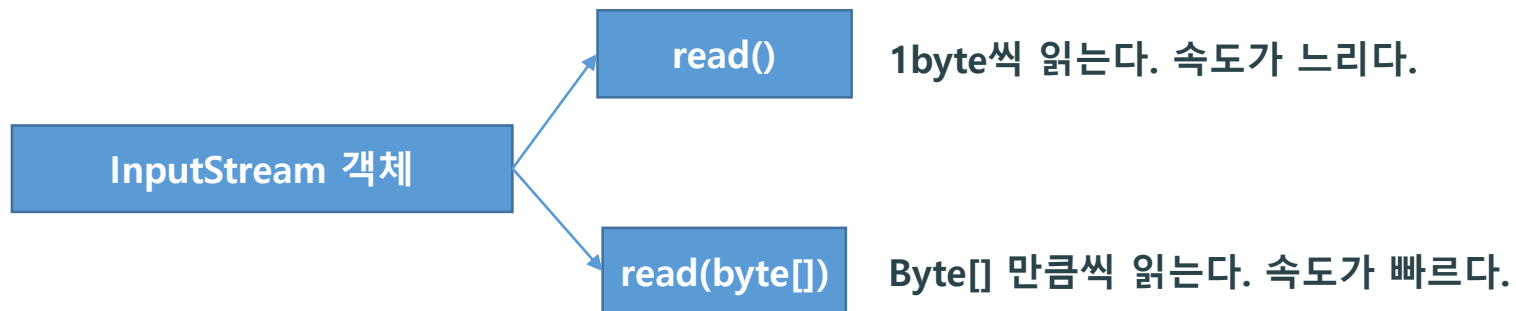
- InputStream 클래스는 바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스입니다.

InputStream 사용방법

- InputStream (추상)클래스를 이용해서 객체를 만든다. 하위 클래스로 구현 시킵니다.
- 한글의 경우 2바이트 이기 때문에 한글이 깨지는 현상이 발생합니다.

InputStream 클래스 주요 메서드

1. **read()**: 입력 스트림으로부터 1바이트를 읽고 읽은 바이트를 리턴.
2. **read(byte[] b)**: 입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열에 저장하고 실제로 읽은 바이트 수를 리턴.
3. **close()**: 스트림을 더 이상 사용하지 않을경우 사용한 시스템 자원을 풀어준다.



InputStream예제

```
InputStream fis = null;

try {
    //FileInputStream(읽을 파일의 경로)
    fis = new FileInputStream("C:\\Users\\Park\\Desktop\\eclipse\\file\\test.txt");
    while(true) {
        int data = fis.read();
        System.out.print((char)data);

        if(data == -1)
            break;
    }

} catch (FileNotFoundException e) {
    System.out.println("해당 경로의 파일을 찾을 수 없습니다.");
} catch (IOException e) {
    System.out.println("파일을 읽을 수 없습니다.");
} finally {
    try {
        fis.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

문자 기반 스트림 클래스

문자 기반 스트림 클래스

- Input/OutputStream은 1바이트 단위로 입/출력 동작을 수행하지만 **한글의 경우 2바이트**이기 때문에 바이트 단위 **처리시 글자가 깨질 수 있습니다.**
- 이러한 문제점을 없애기 위해 자바에서는 문자나 문자열을 다룰 때 유니코드 방식을 제공하고 있습니다.
- 유니코드를 사용하는 입/출력 클래스를 **Reader와 Writer**라고 부르며, 이들을 사용하면 자동으로 바이트가 **유니코드화 되므로 한글 같은 2바이트 문자도 정상적으로 처리**할 수 있습니다.

바이트 기반 최상위 클래스

문자 기반 최상위 클래스

	Byte Streams	Character Streams
Source Streams	InputStream	Reader
Sink Streams	OutputStream	Writer

↓

	Byte Streams	Character Streams
File	FileInputStream FileOutputStream	FileReader FileWriter
Memory : Array	ByteArrayInputStream ByteArrayOutputStream	CharArrayReader CharArrayWriter
Memory : String	-	StringReader StringWriter
Pipe	PipedInputStream PipedOutputStream	PipedReader PipedWriter



문자 기반으로 쓰는 Writer클래스

Writer

- 이 클래스는 텍스트 파일을 프로그램으로 쓸 때 사용하는 문자 기반 스트림입니다.

Writer 사용법

- `writer` (추상)클래스를 이용해서 객체를 만든다. 하위 클래스로 구현 시킵니다.
- 2바이트 문자도 정상적으로 처리할 수 있습니다

Writer 주요 메서드

- `InputStream`클래스와 거의 동일



문자 기반으로 읽는 Reader클래스

Reader

- 이 클래스는 텍스트 파일을 프로그램으로 읽을 때 사용하는 문자 기반 스트림입니다.
- 문자 단위로 읽고 쓰기 때문에 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽거나 쓸 수 없습니다.

Reader사용법

- Reader (추상)클래스를 이용해서 객체를 만든다. 하위 클래스로 구현 시킵니다.
- 2바이트 문자도 정상적으로 처리할 수 있습니다

Reader주요 메서드

- OutputStream클래스와 거의 동일

Byte단위 vs 문자 단위

InputStream, Reader & OutputStream, Writer

1byte 단위

InputStream
OutputStream

↓
이미지, 동영상등의 데이터에
주로 사용

2byte 단위

Reader
Writer

↓
문자열에
주로 사용

API-IO(입출력 성능향상 보조 스트림)



입출력 성능향상 스트림 BufferedWriter


BufferedWriter

- 이 클래스는 텍스트 파일을 프로그램으로 읽을 때 사용하는 문자 기반 스트림입니다.
- 데이터를 읽고 버퍼에 저장한 후 한번에 쓰는 형태로 사용되기 때문에 속도가 빠름

BufferedWriter 사용법

- 생성자의 매개변수(문자출력스트림 클래스) 를 전달합니다

```
Writer rt = new FileWriter("출파일경로");  
BufferedWriter bw = new BufferedWriter(rt);
```



BufferedWriter 주요 메서드

- Writer클래스와 거의 동일
- flush() : 버퍼를 비운다

주의할점

버퍼가 가득찼을 때만 출력을 해내기 때문에 flush()를 호출하여 잔류하는 데이터를 모두 보내야합니다

입출력 성능향상 스트림 BufferedReader


BufferedReader

- 이 클래스는 텍스트 파일을 프로그램으로 읽을 때 사용하는 문자 기반 스트림입니다.
- 데이터를 읽고 버퍼에 저장한 후 한번에 읽는 형태로 사용되기 때문에 속도가 빠름

BufferedReader사용법

- 생성자의 매개변수(문자출력스트림 클래스) 를 전달합니다

```
Reader rd = new FileReader("파일경로");  
BufferedReader bf = new BufferedReader(rd);
```



BufferedReader주요 메서드

- Reader클래스와 거의 동일
- **readLine()** : 한 줄을 한꺼번에 읽어 들입니다



BufferedReader 예제

```
try {  
  
    Reader rd = new FileReader( " 읽을파일경로");  
    BufferedReader bf = new BufferedReader(rd);  
  
    String str = "";  
    while( (str = bf.readLine()) != null) {  
  
        System.out.println(str);  
    }  
  
} catch (Exception e) {  
    e.printStackTrace();  
}
```



Chapter 21

수고하셨습니다