JAVA 기초입문과정

CHAPTER15 스트림

Contents

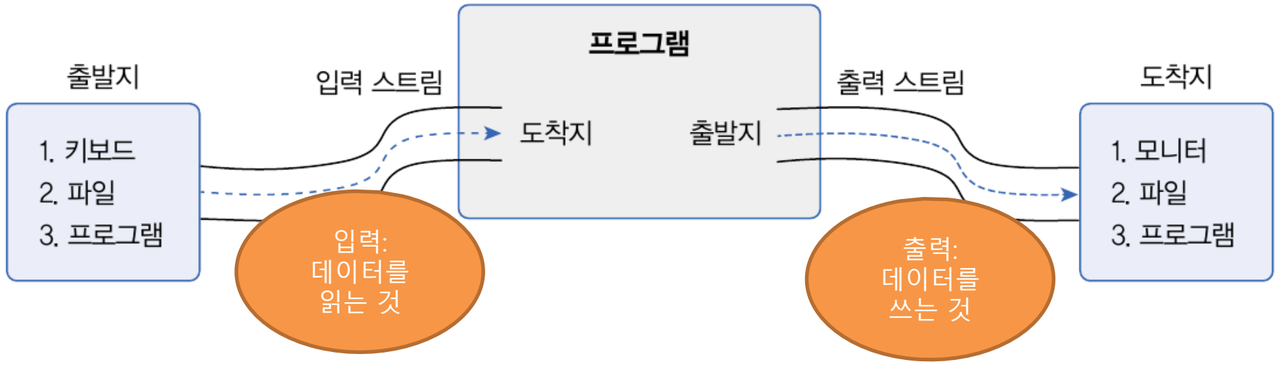

PART
01
“
입출력 스트림
”

PART 01

입출력 스트림

1. 입출력 스트림

데이터가 키보드, 다른 파일 또는 프로그램으로 부터 입력 되거나 모니터로 출력 및 다른 프로그램에 전송 되는것.



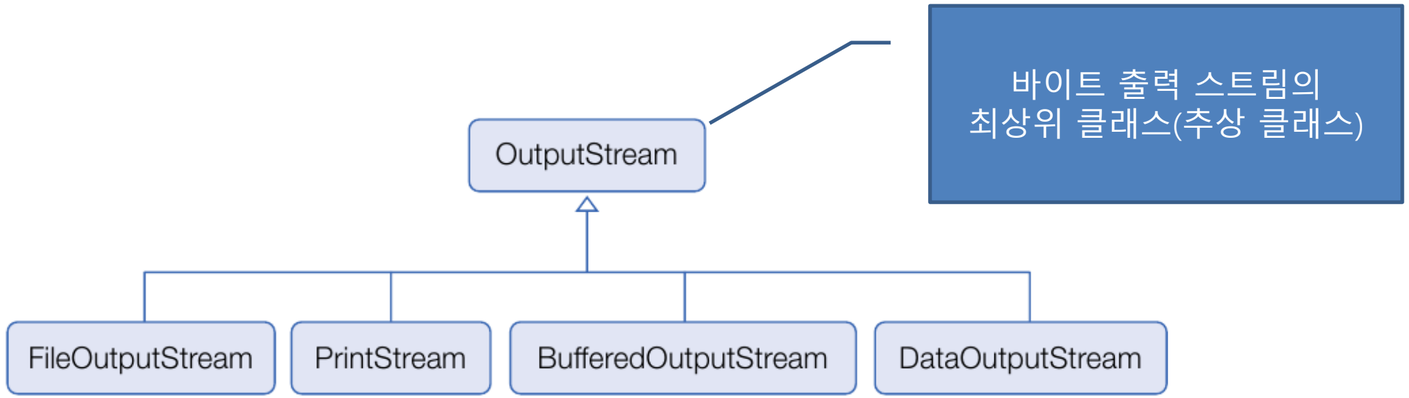
- 바이트 스트림: 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 입출력 할때 사용.
- 문자 스트림: 문자만 입출력 할 때 사용.(ex: txt파일)

어떤 데이터를 입출력 하냐에 따라 나눠진다.

[JAVA.IO 패키지에서 사용]

구분	바이트 스트림		문자 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXXInputStream (FileInputStream)	XXXOutputStream (FileOutputStream)	XXXReader (FileReader)	XXXWriter (FileWriter)

2. 바이트 출력 스트림(OutputStream)



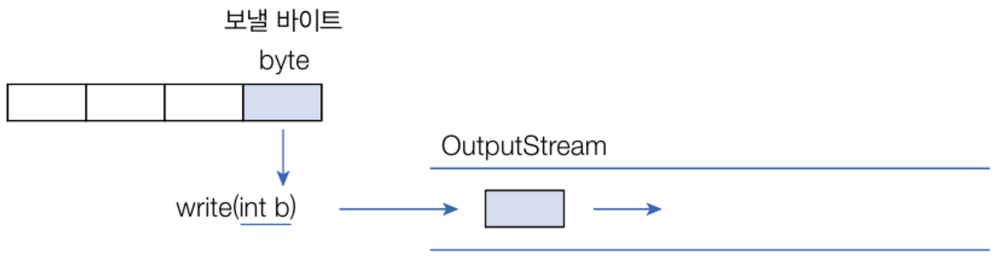
[OutputStream 주요 메소드]

리턴 타입	메소드	설명
void	wrtie(int b)	1byte를 출력
void	write(byte[ ] b)	매개값으로 주어진 배열 b의 모든 바이트를 출력
void	write(byte[ ] b, int off, int len)	매개값으로 주어진 배열 b[off]부터 len개의 바이트를 출력
void	flush( )	출력 버퍼에 잔류하는 모든 바이트를 출력
void	close( )	출력 스트림을 닫고 사용 메모리 해제

- OutStream은 내부에 작은 버퍼를 가지고 있어 버퍼가 다 차면 순서대로 출력 -> flush()는 이런 바이트를 비우는 역할

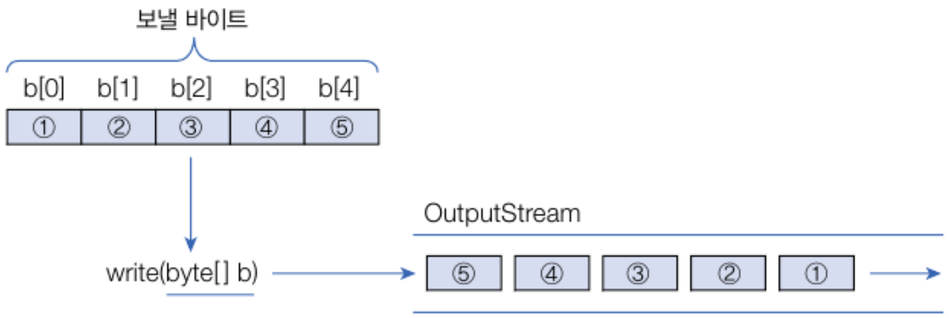
2. 바이트 출력 스트림(OutputStream)

1. 1바이트 출력 write(int b)



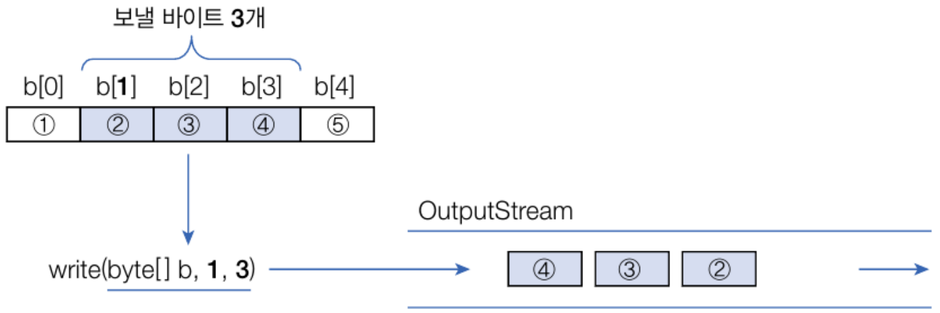
- 매개변수가 int지만 4byte 씩 출력하지 않고 int에 담아 1byte씩 출력

2. 바이트 배열 출력(배열 전체) write(byte[] b)



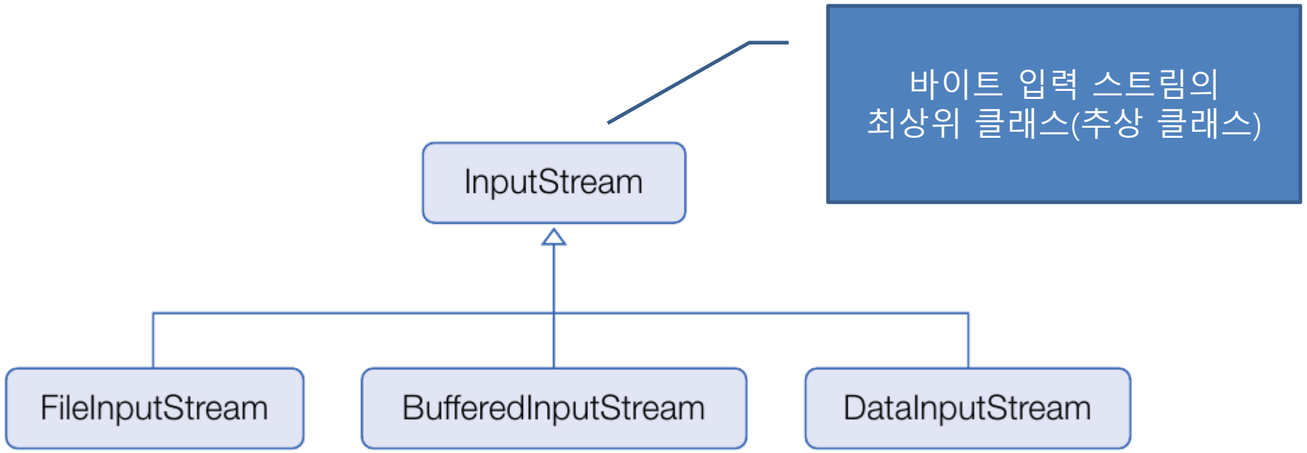
- 매개 값으로 주어진 모든 배열의 바이트를 통째로 출력

3. 바이트 배열 출력(배열의 일부분) write(byte[] b, int off, int len)



- 매개 값으로 주어진 일부분을 출력

3. 바이트 입력 스트림(InputStream)



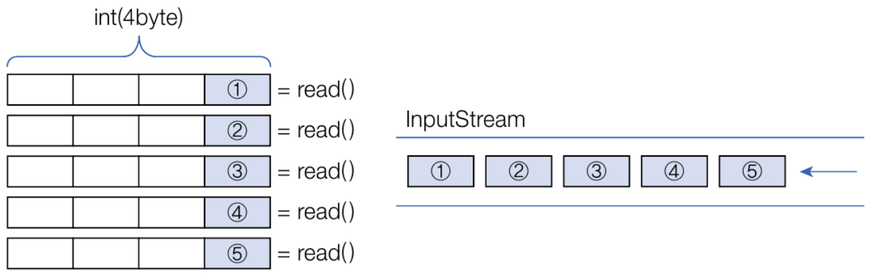
[InputStream 주요 메소드]

리턴 타입	메소드	설명
int	read( )	1byte를 읽은 후 읽은 바이트를 리턴
int	read(byte[ ] b)	읽은 바이트를 매개값으로 주어진 배열에 저장 후 읽은 바이트 수를 리턴
void	close( )	입력 스트림을 닫고 사용 메모리 해제

3. 바이트 입력 스트림(InputStream)

1. 1바이트 입력

read(int b)



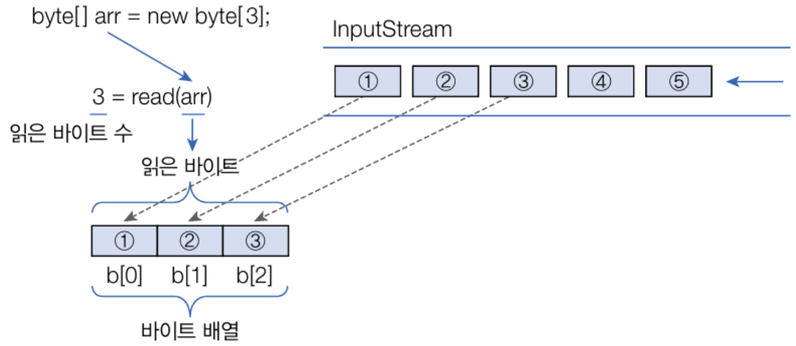
- 매개변수가 int지만 4byte 씩 입력하지 않고 int에 담아 1byte씩 입력
- 더 이상 읽을 수 없다면 -1을 리턴

많은 양의 데이터를 읽을 때는
read(byte[] b)로 읽는 것이 좋다->
read()메소드는 100개의 바이트가 들어오면
100번을 반복해서 읽음

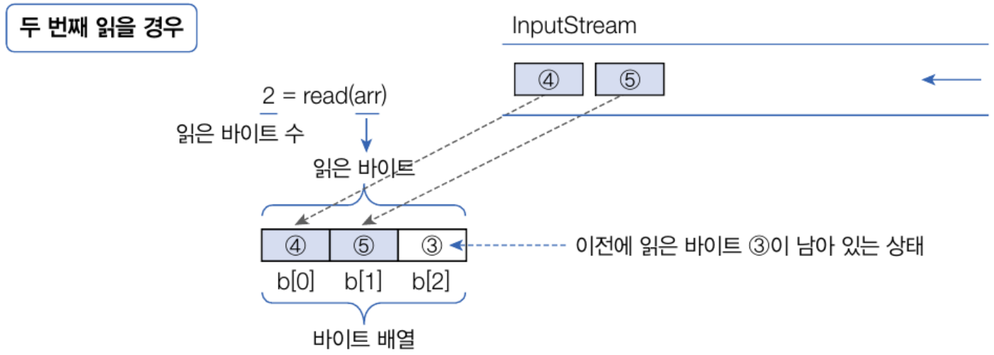
2. 바이트 배열 입력(배열 전체)

read(byte[] b)

첫 번째 읽을 경우

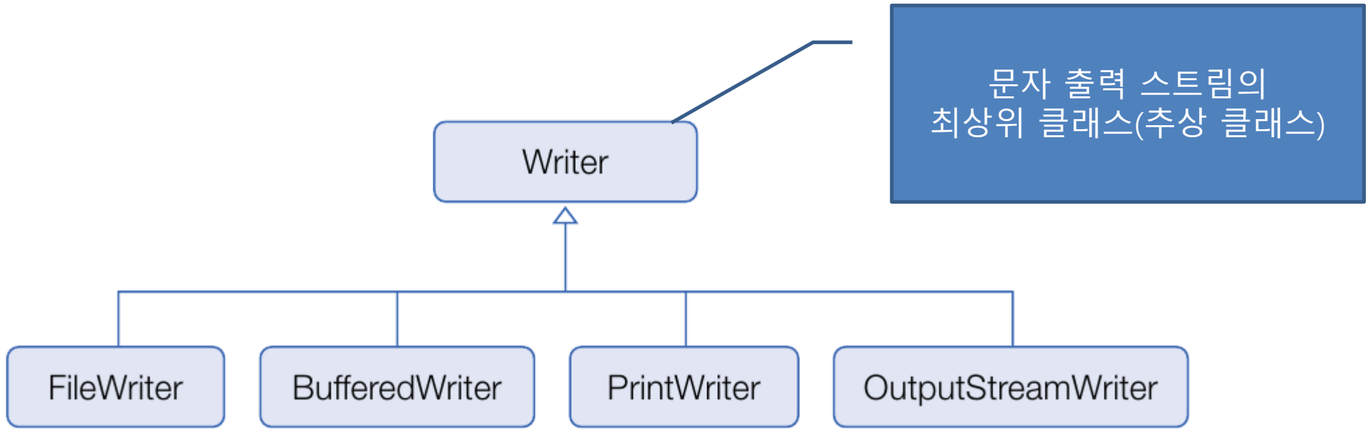


두 번째 읽을 경우



- 입력 스트림으로 부터 주어진 길이 만큼 읽고 배열에 저장한 다음 읽은 바이트수를 리턴
- 더 이상 읽을 수 없다면 -1을 리턴

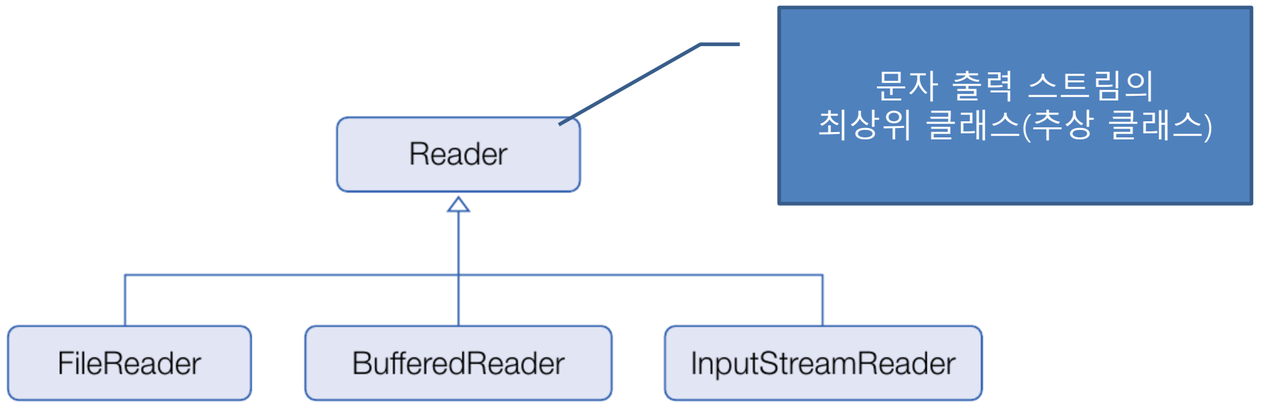
4. 문자 쓰기(Writer)



[Writer 주요 메소드]

리턴 타입	메소드	설명
void	wrtie(int c)	매개값으로 주어진 한 문자를 출력
void	write(char[ ] cbuf)	매개값으로 주어진 배열의 모든 문자를 출력
void	write(char[ ] cbuf, int off, int len)	매개값으로 주어진 배열에서 cbuf[off]부터 len개까지의 문자를 출력
void	write(String str)	매개값으로 주어진 문자열을 출력
void	write(String str, int off, int len)	매개값으로 주어진 문자열에서 off 순번부터 len개까지의 문자를 출력
void	flush( )	버퍼에 잔류하는 모든 문자를 출력
void	close( )	출력 스트림을 닫고 사용 메모리를 해제

5. 문자 읽기(Reader)



[Reader 주요 메소드]

메소드		설명
int	read( )	1개의 문자를 읽고 리턴
int	read(char[ ] cbuf)	읽은 문자들을 매개값으로 주어진 문자 배열에 저장하고 읽은 문자 수를 리턴
void	close( )	입력 스트림을 닫고, 사용 메모리 해제


PART
02
“
보조 스트림
”

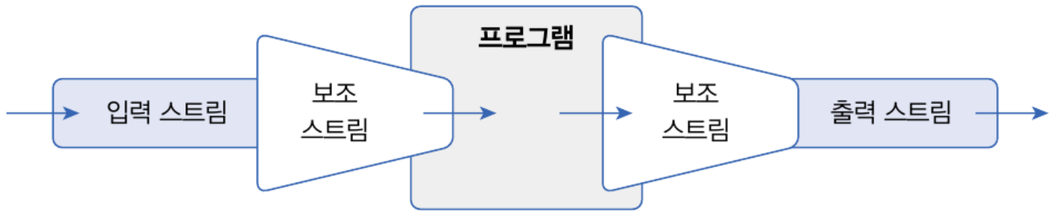
PART 02

보조 스트림

1. 보조 스트림

- 다른 스트림과 연결되어 여러 가지 편리한 기능을 제공해주는 스트림.
- 자체적으로 입출력을 수행할 수 없으므로 다른 입출력 스트림에 연결해서 사용.

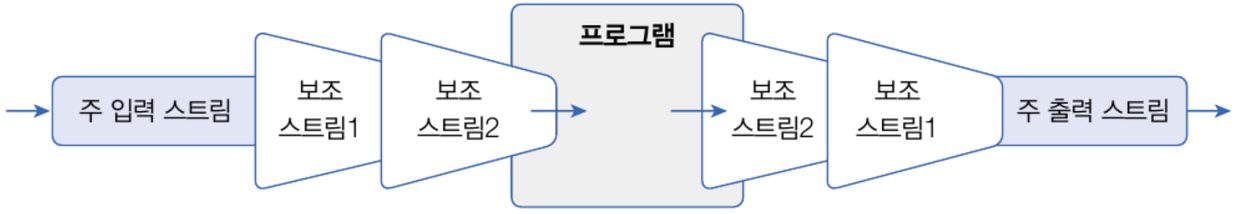
- **일반 보조스트림 연결**



보조스트림 변수 = new 보조스트림(입출력스트림)

```
InputStream is = new FileInputStream("...");
InputStreamReader reader = new InputStreamReader(is);
```

- **보조스트림과 또 다른 보조스트림 연결(스트림 체인)**



보조스트림2 변수 = new 보조스트림2(보조스트림1)

```
InputStream is = new FileInputStream("...");
InputStreamReader reader = new InputStreamReader(is);
BufferedReader br = new BufferedReader(reader);
```

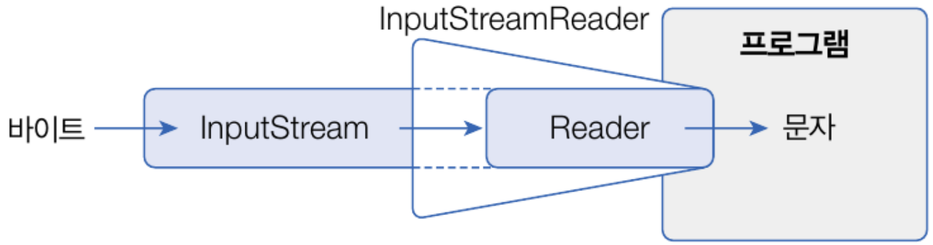
보조 스트림

2. 문자 변환 스트림

- 바이트 스트림에서 입출력할 데이터가 문자라면 문자 스트림(Reader, Writer)으로 변환에서 사용.

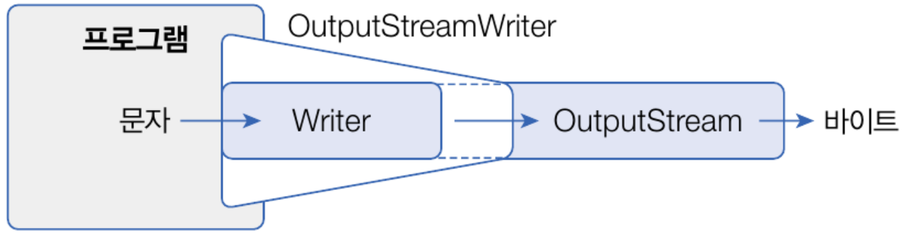
  - 문자로 바로 입출력하는 편리함
  - 문자셋 종류 지정 가능

- **InputStream -> Reader**




```
InputStream is = new FileInputStream("C:/Temp/test.txt");
InputStreamReader reader = new InputStreamReader(is);
```

- **Writer -> OutputStream**



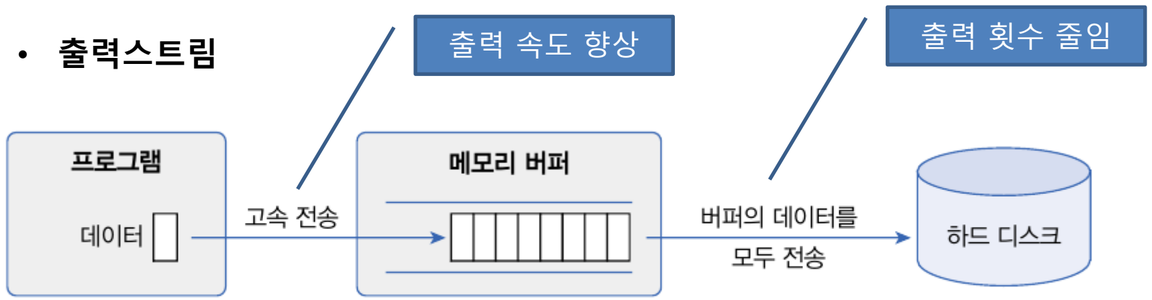

```
InputStream is = new FileInputStream("C:/Temp/test.txt");
InputStreamReader reader = new InputStreamReader(is);
```

보조 스트림

3. 성능 향상 스트림

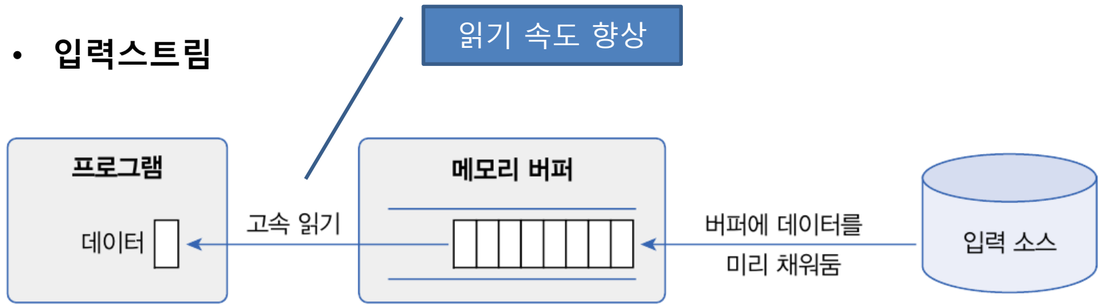
- CPU와 메모리 성능이 뛰어나도 하드 디스크의 입출력이 늦어지면 프로그램의 실행 성능은 하드디스크에 맞춰짐.
- 해결책: 프로그램이 입출력 소스와 직접 작업하지 않고 중간에 메모리 버퍼와 함께 작업.

- **출력스트림**



BufferedInputStream bis = new BufferedInputStream(바이트 입력 스트림);
BufferedOutputStream bos = new BufferedOutputStream(바이트 출력 스트림);

- **입력스트림**

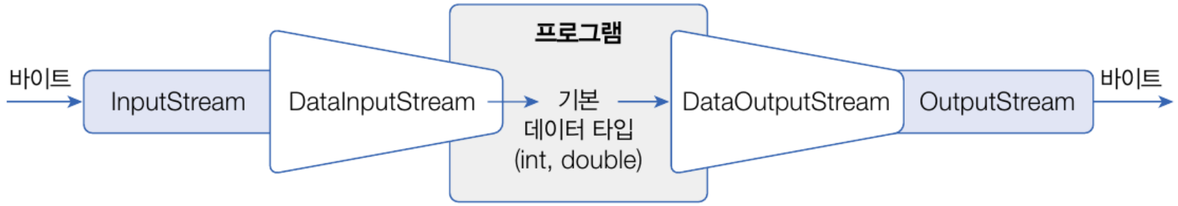


BufferedReader br = new BufferedReader(문자 입력 스트림);
BufferedWriter bw = new BufferedWriter(문자 출력 스트림);

보조 스트림

4. 기본 타입 스트림

- 바이트 스트림에 DataInputStream과 DataOutputStream 보조스트림을 연결하면 기본 타입(primitive type)들을 출력할 수 있다.



DataInputStream dis = **new DataInputStream(바이트 입력 스트림);**
DataOutputStrem dos = **new DataOutputStrem(바이트 출력 스트림);**

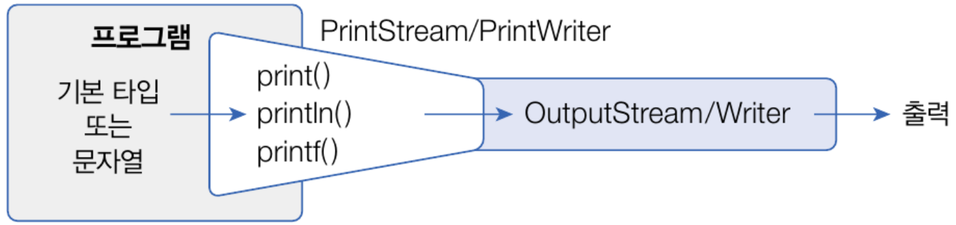
- 데이터 타입의 크기가 모두 다르므로 입출력 할 때 순서가 같아야 한다.

DataInputStream		DataOutputStream	
boolean	readBoolean( )	void	writeBoolean(boolean v)
byte	readByte( )	void	writeByte(int v)
char	readChar( )	void	writeChar(int v)
double	readDouble( )	void	writeDouble(double v)
float	readFloat( )	void	writeFloat(float v)
int	readInt( )	void	writeInt(int v)
long	readLong( )	void	writeLong(long v)
short	readShort( )	void	writeShort(int v)
String	readUTF( )	void	writeUTF(String str)

보조 스트림

5. 프린트 스트림

- 프린터와 유사하게 출력하는 print() ,println() 메소드를 가지고 있는 스트림



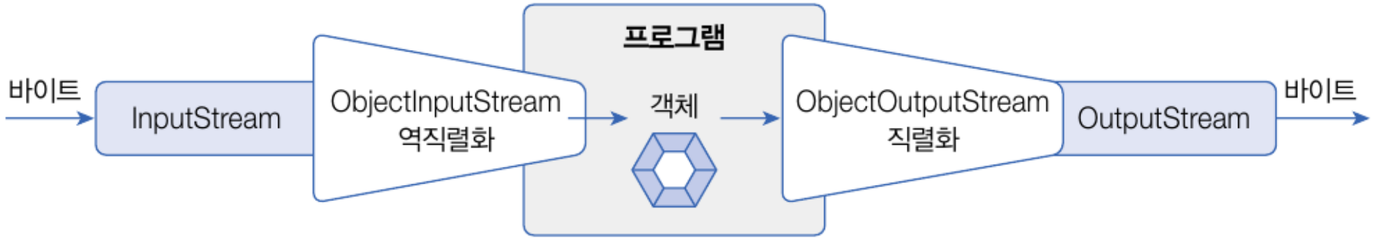
PrintStream ps = new PrintStream(바이트 출력 스트림);
PrintWriter pw = new PrintWriter(문자 출력 스트림);

PrintStream / PrintWriter			
void	print(boolean b)	void	println(boolean b)
void	print(char c)	void	println(char c)
void	print(double d)	void	println(double d)
void	print(float f)	void	println(float f)
void	print(int i)	void	println(int i)
void	print(long l)	void	println(long l)
void	print(Object obj)	void	println(Object obj)
void	print(String s)	void	println(String s)
		void	println( )

보조 스트림

6. 객체 스트림

- 메모리에 생성된 객체를 파일 또는 네트워크로 출력할 수 있다.
- 직렬화: 객체를 출력하기 위해 필드값을 일렬로 늘어선 바이트로 변경하는 것.
- 역직렬화: 직렬화된 바이트를 객체의 필드값으로 복원하는 것.



ObjectInputStream ois = new ObjectInputStream(바이트 입력 스트림);
ObjectOutputStream oos = new ObjectOutputStream(바이트 출력 스트림);

- 객체 직렬화(필드 값 -> 바이트)

```
oos.writeObject(객체);
```

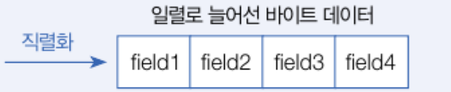
- 역 직렬화(바이트 -> 필드 값)

```
객체타입 변수 = (객체타입) ois.readObject();
```

6. 객체 스트림

- 자바는 Serializable 인터페이스를 구현한 클래스만 직렬화 할 수 있도록 제한한다.
- Serializable 인터페이스는 멤버가 없는 빈 인터페이스이지만, 객체를 직렬화할 수 있다고 표시하는 역할을 한다.
- 객체가 직렬화될 때 인스턴스 필드 값은 직렬 대상, 정적 필드값과 transient로 선언된 필드 값은 직렬화에서 제외되므로 출력되지 않는다.

```
public class XXX implements Serializable {
  public int field1;
  protected int field2;
  int field3;
  private int field4;
  public static int field5;    //정적 필드는 직렬화에서 제외
  transient int field6;        //transient로 선언된 필드는 직렬화에서 제외
}
```



- 직렬화시 사용된 클래스와 역직렬화할 때 사용한 클래스는 기본적으로 동일한 클래스여야 한다.
- serialVersionUID 상수: 클래스 내용이 달라도 직렬화된 필드를 공통으로 포함하고 있으면 역직렬화할 수 있다.

```
public class Member implements
  Serializable {
  int field1;
  int field2;
}
```

```
public class Member implements
  Serializable {
  int field1;
  int field2;
  int field3;
}
```

```
public class Member implements
  Serializable {
  static final long
    serialVersionUID = 1;
  int field1;
  int field2;
}
```

```
public class Member implements
  Serializable {
  static final long
    serialVersionUID = 1;
  int field1;
  int field2;
  int field3;
}
```


PART
03
“
File과 Files 클래스
”

PART 03

File과 Files 클래스

1. File클래스

- File 클래스는 파일과 디렉토리의 접근 권한, 생성된 시간, 마지막 수정 일자, 크기, 경로 등의 정보를 얻을 수 있으며 새로운 파일과 디렉토리 생성 및 삭제, 이름 변경 등의 조작 메소드를 가지고 있다.
- Files는 File을 개선한 기능으로 좀 더 많은 기능을 가지고 있다.

```
File file = new File("경로");
```

- 경로 구분자
  -윈도우: ₩₩, / 둘다 사용
  -맥OS, 리눅스: / 사용

```
File file = new File("C:/Temp/file.txt");
File file = new File("C:\\Temp\\file.txt");
```

- 파일 디렉토리가 있는지 실제로 확인해야 한다.

```
boolean isExist = file.exists();   //파일이나 폴더가 존재한다면 true를 리턴
```

True

리턴 타입	메소드	설명
boolean	delete( )	파일 또는 디렉토리 삭제
boolean	canExecute( )	실행할 수 있는 파일인지 여부
boolean	canRead( )	읽을 수 있는 파일인지 여부
boolean	canWrite( )	수정 및 저장할 수 있는 파일인지 여부
String	getName( )	파일의 이름을 리턴
String	getParent( )	부모 디렉토리를 리턴
File	getParentFile( )	부모 디렉토리를 File 객체로 생성 후 리턴
String	getPath( )	전체 경로를 리턴
boolean	isDirectory( )	디렉토리인지 여부
boolean	isFile( )	파일인지 여부
boolean	isHidden( )	숨김 파일인지 여부
long	lastModified( )	마지막 수정 날짜 및 시간을 리턴
long	length( )	파일의 크기 리턴
String[]	list( )	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 String 배열로 리턴
String[]	list(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중에 FilenameFilter에 맞는 것만 String 배열로 리턴
File[]	listFiles( )	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 File 배열로 리턴
File[]	listFiles(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중에 FilenameFilter에 맞는 것만 File 배열로 리턴

False

리턴 타입	메소드	설명
boolean	createNewFile( )	새로운 파일을 생성
booelan	mkdir( )	새로운 디렉토리를 생성
boolean	mkdirs( )	경로상에 없는 모든 디렉토리를 생성

File과 Files 클래스

2. Files클래스

- Files는 정적메소드로 구성되어 있기 때문에 File 클래스처럼 객체로 만들 필요는 없다.

기능	관련 메소드
복사	copy
생성	createDirectories, createDirectory, createFile, createLink, createSymbolicLink, createTempDirectory, createTempFile
이동	move
삭제	delete, deleteIfExists
존재, 검색, 비교	exists, notExists, find, mismatch
속성	getLastModifiedTime, getOwner, getPosixFilePermissions, isDirectory, isExecutable, isHidden, isReadable, isSymbolicLink, isWritable, size, setAttribute, setLastModifiedTime, setOwner, setPosixFilePermissions, probeContentType
디렉토리 탐색	list, newDirectoryStream, walk
데이터 입출력	newInputStream, newOutputStream, newBufferedReader, newBufferedWriter, readAllBytes, lines, readAllLines, readString, readSymbolicLink, write, writeString

Thank You!