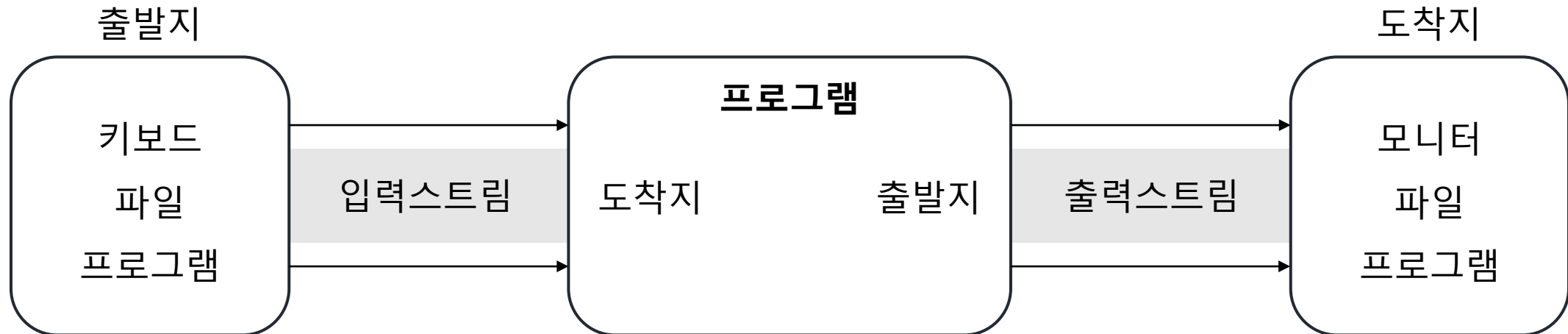


입출력(IO)

▶ 입출력(IO)

Input과 Output의 약자, 컴퓨터 내부 또는 외부 장치와 프로그램 간의 데이터를 주고 받는 것
장치와 입출력을 위해서는 하드웨어 장치에 직접 접근이 필요한데 다양한 매체에 존재하는 데이터들을
사용하기 위해 입출력 데이터를 처리할 공통적인 방법으로 스트림 이용



▶ 스트림(Stream)

입출력 장치에서 데이터를 읽고 쓰기 위해서 자바에서 제공하는 클래스

모든 스트림은 단방향이며 각각의 장치마다 연결할 수 있는 스트림 존재

하나의 스트림으로 입출력을 동시에 수행할 수 없으므로 동시에 수행하려면 2개의 스트림 필요

✓ 분류

구분	바이트 단위 처리		문자 단위 처리	
	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위클래스	XXXInputStream	XXXOutputStream	XXXReader	XXXWriter
데이터가 들어옴			데이터가 나감	

▶ 스트림 종류

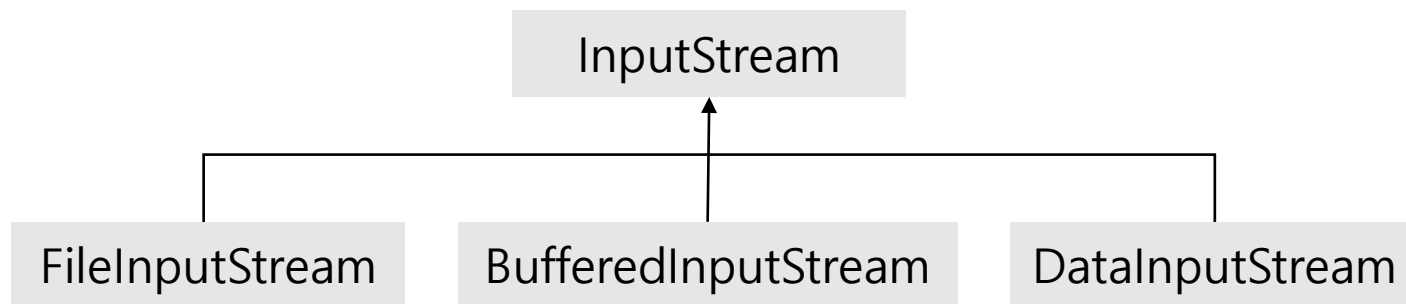
Reader	BufferedReader	LineNumberReader
	CharArrayReader	
	InputStreamReader	FileReader
	FilterReader	PushbackReader
	PipedReader	
	StringReader	
Writer	BufferedWriter	
	CharArrayWriter	
	OutputStreamWriter	FileWriter
	FilterWriter	
	PipedWriter	
	StringWriter	
	FilterWriter	

* 색이 있는 것은 기반 스트림, 색이 없는 것은 보조 스트림

InputStream	FileInputStream	
	PipedInputStream	
	FilterInputStream	LineNumberInputStream
		DataInputStream
		BufferedInputStream
		PushbackInputStream
	ByteArrayInputStream	
	SequenceInputStream	
	StringBufferedInputStream	
	ObjectInputStream	
OutputStream	FileOutputStream	
	PipedOutputStream	
	FilterOutputStream	DataOutputStream
		BufferedOutputStream
		PrintStream
	ByteArrayOutputStream	
	ObjectOutputStream	

▶ InputStream

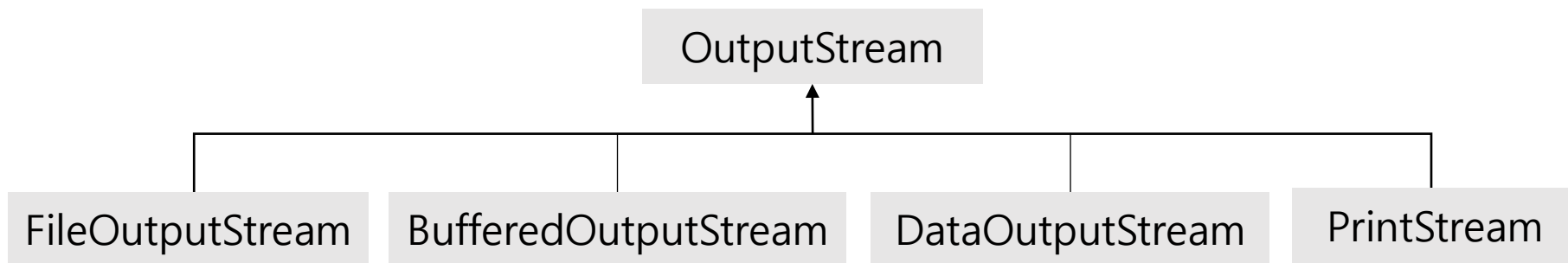
바이트 기반 입력 스트림의 최상위 클래스로 추상클래스임



리턴 타입	메소드	설명
int	read()	입력 스트림으로부터 1바이트를 읽고 읽은 바이트 리턴
int	read(byte[] b)	입력 스트림으로부터 읽은 바이트들을 매개 값으로 주어진 바이트 배열 b에 저장하고 실제로 읽은 바이트 수 리턴
int	read(byte[] b, int off, int len)	입력 스트림으로부터 len개의 바이트만큼 읽고 매개 값으로 주어진 바이트 배열 b[off]부터 len개까지를 저장, 그리고 실제로 읽은 바이트 수인 len개 리턴, 만약 len개를 모두 읽지 못 하면 실제로 읽은 바이트 수 리턴
void	close()	사용한 시스템 자원 반납 후 입력 스트림을 닫음

► OutputStream

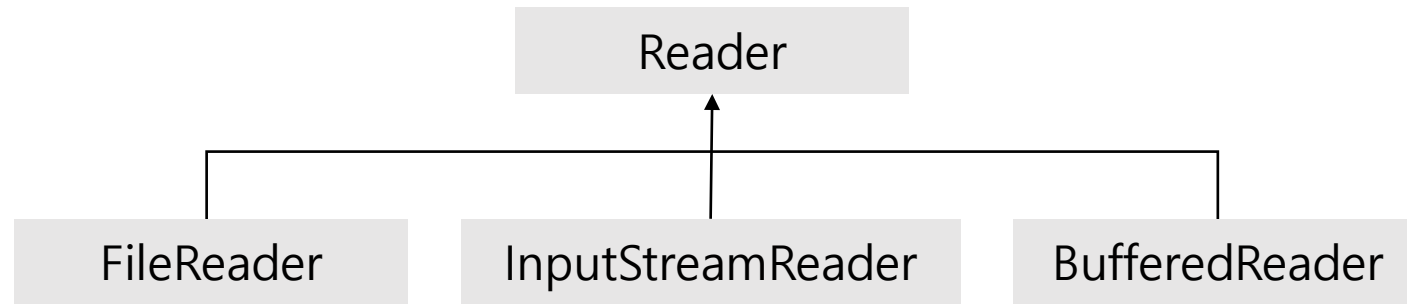
바이트 기반 출력 스트림의 최상위 클래스로 추상클래스임



리턴 타입	메소드	설명
void	<code>write(int b)</code>	출력 스트림으로 1바이트를 보냄
void	<code>write(byte[] b)</code>	출력 스트림에 매개 값으로 주어진 바이트 배열 b의 모든 바이트를 보냄
void	<code>write(byte[] b, int off, int len)</code>	출력 스트림에 매개 값으로 주어진 바이트 배열 b[off]부터 len개까지의 바이트를 보냄
void	<code>flush()</code>	버퍼에 잔류하는 모든 바이트 출력
void	<code>close()</code>	사용한 시스템 자원 반납 후 출력 스트림을 닫음

▶ Reader

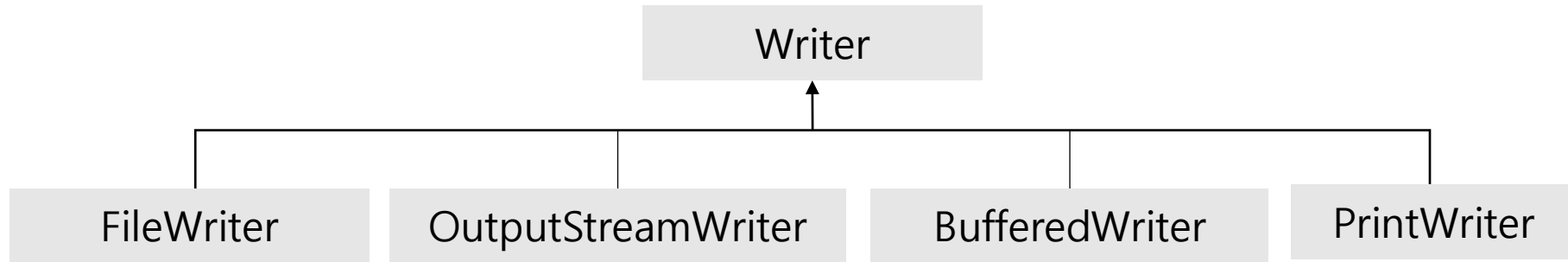
문자 기반 입력 스트림의 최상위 클래스로 추상클래스임



리턴 타입	메소드	설명
int	read()	입력 스트림으로부터 한 개의 문자를 읽고 리턴
int	read(char[] c)	입력 스트림으로부터 읽은 문자들을 매개 값으로 주어진 문자 배열 c에 저장하고 실제로 읽은 문자 수 리턴
int	read(char[] c, int off, int len)	입력 스트림으로부터 len개의 문자만큼 읽고 매개 값으로 주어진 문자배열 c[off]부터 len개까지 저장, 실제로 읽은 문자 수인 len개 리턴
void	close()	사용한 시스템 자원 반납 후 입력 스트림을 닫음

▶ Writer

문자 기반 출력 스트림의 최상위 클래스로 추상클래스임



리턴 타입	메소드	설명
void	<code>write(int c)</code>	출력 스트림으로 매개 값이 주어진 한 문자를 보냄
void	<code>write(char[] c)</code>	출력 스트림에 매개 값으로 주어진 문자 배열 <code>c</code> 의 모든 문자를 보냄
void	<code>write(char[] c, int off, int len)</code>	출력 스트림에 매개 값으로 주어진 문자 배열 <code>c[off]</code> 부터 <code>len</code> 개까지의 문자 보냄
void	<code>write(String str)</code>	출력 스트림에 매개 값으로 주어진 문자열을 보냄
void	<code>write(String str, int off, int len)</code>	출력 스트림에 매개 값으로 주어진 문자열 <code>off</code> 순번부터 <code>len</code> 개까지 문자 보냄
void	<code>flush()</code>	버퍼에 잔류하는 모든 문자열 출력
void	<code>close()</code>	사용한 시스템 자원 반납 후 출력 스트림을 닫음

▶ File 클래스

파일 시스템의 파일을 표현하는 클래스

파일 크기, 파일 속성, 파일 이름 등의 정보와 파일 생성 및 삭제 기능 제공

✓ File 객체 생성

```
File file = new File("파일 경로");
```

```
File file = new File("C:/data/test.txt");
```

▶ File 클래스

✓ 파일/디렉토리 생성 및 삭제 메소드

리턴 타입	메소드	설명
boolean	createNewFile()	새로운 파일 생성
boolean	mkdir()	새로운 디렉토리 생성
boolean	mkdirs()	경로 상에 없는 모든 디렉토리 생성
boolean	delete()	파일 또는 디렉토리 삭제

▶ File 클래스

✓ 파일/디렉토리 정보 리턴 메소드

리턴 타입	메소드	설명
boolean	canExcute()	실행할 수 있는 파일인지 여부
boolean	canRead()	읽을 수 있는 파일인지 여부
boolean	canWrite()	수정 및 저장할 수 있는 파일인지 여부
String	getName()	파일 이름 리턴
String	getParent()	부모 디렉토리 리턴
File	getParentFile()	부모 디렉토리를 File객체로 생성 후 리턴
String	getPath()	전체 경로 리턴
boolean	isDirectory()	디렉토리인지 여부
boolean	isFile()	파일인지 여부
boolean	isHidden()	숨김 파일인지 여부
long	lastModified()	마지막 수정 날짜 및 시간 리턴
long	length()	파일 크기 리턴

▶ File 클래스

✓ 파일/디렉토리 정보 리턴 메소드

리턴 타입	메소드	설명
String[]	list()	디렉토리 포함한 파일목록을 String배열로 리턴
String[]	list(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중 FilenameFilter에 맞는 것만 String배열로 리턴
File[]	listFiles()	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부 File 배열로 리턴
File[]	listFile(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중 FilenameFilter에 맞는 것만 File배열로 리턴

▶ FileInputStream

파일로부터 바이트 단위로 읽을 때 사용

그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 파일 읽기 가능

InputStream의 하위 클래스로 InputStream과 사용 방법 동일

✓ 객체 생성

FileInputStream 객체가 생성될 때 파일과 직접 연결 됨

만약 파일이 존재하지 않으면 FileNotFoundException이 발생하므로 예외처리 필수

```
FileInputStream fis = new FileInputStream("C:/data/test.txt");
```

▶ FileOutputStream

파일로부터 바이트 단위로 저장할 때 사용

그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 데이터를 파일로 저장

OutputStream의 하위 클래스로 OutputStream과 사용 방법 동일

✓ 객체 생성

FileOutputStream 객체가 생성될 때 파일과 직접 연결 됨

만약 파일이 존재하지 않으면 자동으로 생성하지만

이미 파일이 존재하는 경우 파일을 덮어쓰는 단점이 있음

```
FileOutputStream fos = new FileOutputStream("C:/data/test.txt");
```

만일 기존 파일에 이어서 계속 작성하고 싶다면 아래 예제처럼 객체 생성 시 가능

```
FileOutputStream fos = new FileOutputStream("C:/data/test.txt", true);
```

▶ FileReader

텍스트 파일로부터 문자 단위로 읽을 때 사용

텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽기 불가능

Reader의 하위 클래스로 Reader와 사용 방법 동일

✓ 객체 생성

FileReader객체가 생성될 때 파일과 직접 연결 됨

만약 파일이 존재하지 않으면 FileNotFoundException이 발생하므로 예외처리 필수

```
FileReader fr = new FileReader("C:/data/test.txt");
```

```
FileReader fr = new FileReader(new File("C:/data/test.txt"));
```

▶ FileWriter

텍스트 파일로부터 문자 단위로 저장할 때 사용
텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 저장 불가능
Writer의 하위 클래스로 Writer와 사용 방법 동일

✓ 객체 생성

FileWriter객체가 생성될 때 파일과 직접 연결 됨
만약 파일이 존재하지 않으면 자동으로 생성하지만
이미 파일이 존재하는 경우 파일을 덮어쓰는 단점이 있음

```
FileWriter fw = new FileWriter("C:/data/test.txt");
```

만일 기존 파일에 이어서 계속 작성하고 싶다면 아래 예제처럼 객체 생성 시 가능

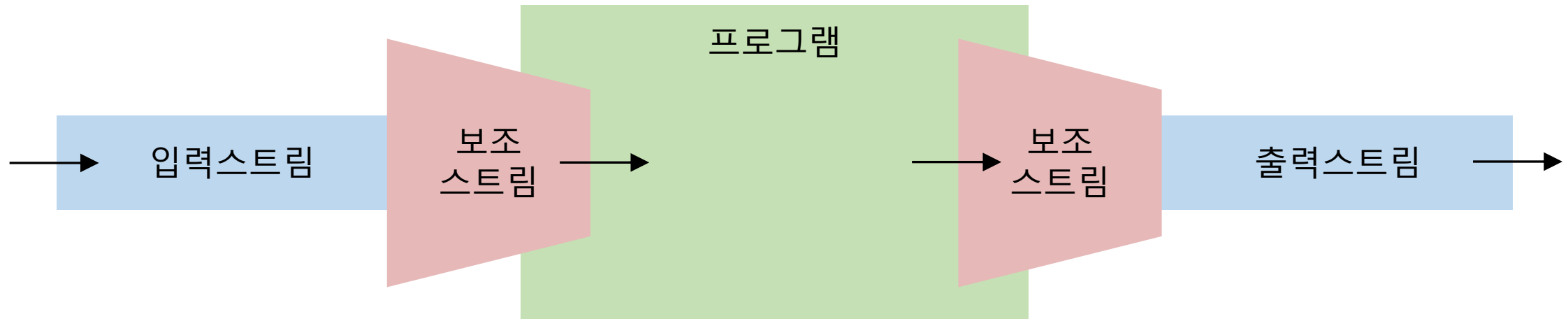
```
FileWriter fw = new FileWriter("C:/data/test.txt", true);
```


▶ 보조 스트림

스트림의 기능을 향상시키거나 새로운 기능을 추가하기 위해 사용

보조 스트림은 실제 데이터를 주고 받는 스트림이 아니기 때문에 입출력 처리 불가능

기반 스트림을 먼저 생성한 후 이를 이용하여 보조 스트림 생성



▶ 보조 스트림 종류

문자 변환(InputStreamReader/OutputStreamWriter),
입출력 성능(BufferedInputStream/BufferedOutputStream),
기본 데이터 타입 출력(DataInputStream, DataOutputStream),
객체 입출력(ObjectInputStream/ObjectOutputStream) 등의 기능을 제공하는 보조스트림이 있음

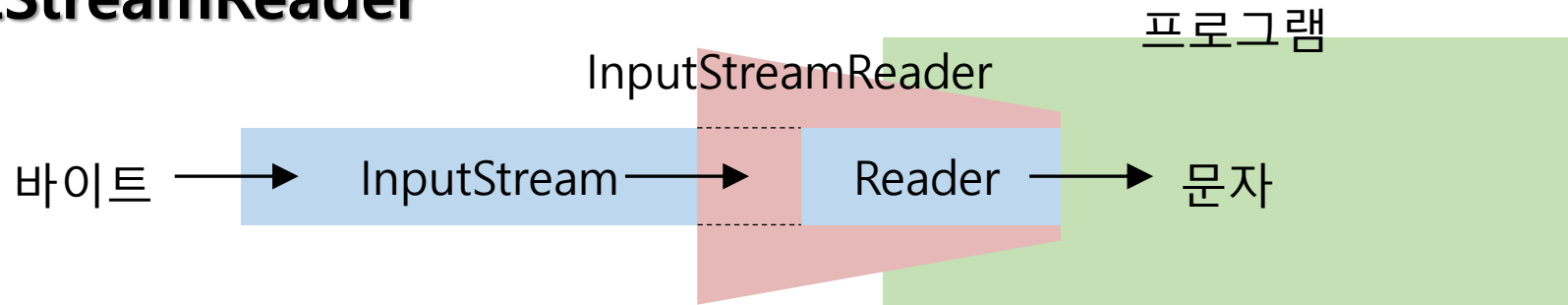
✓ 예시

```
FileInputStream fis = new FileInputStream("sample.txt"); //기반 스트림 생성  
BufferedInputStream bis = new BufferedInputStream(fis); //보조스트림 생성  
bis.read(); //보조스트림으로부터 데이터 읽어옴
```

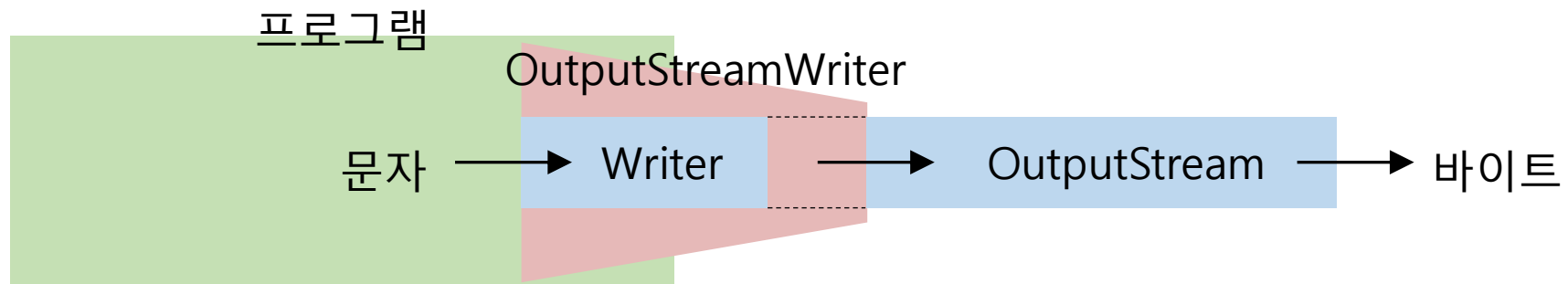
▶ 문자 변환 보조 스트림

소스 스트림이 바이트 기반 스트림이지만 데이터가 문자일 경우 사용
Reader와 Writer는 문자 단위로 입출력을 하기 때문에 데이터가 문자인 경우
바이트 기반 스트림보다 편리하게 사용 가능

✓ InputStreamReader



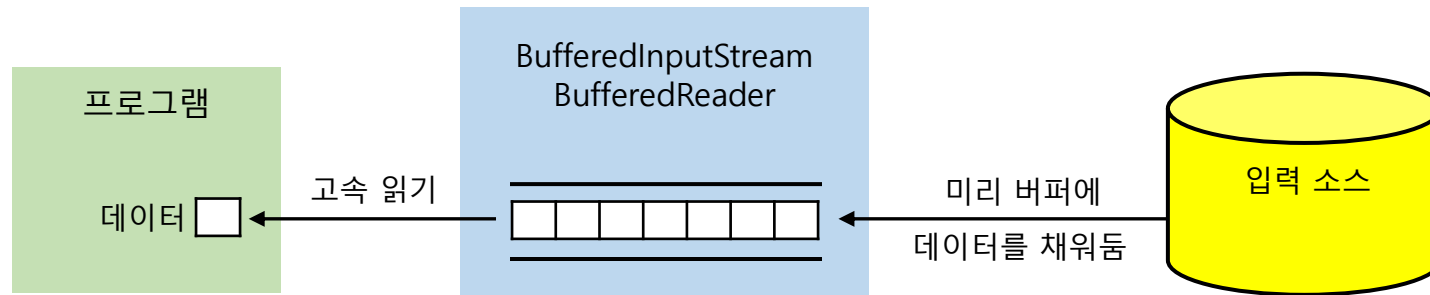
✓ OutputStreamWriter



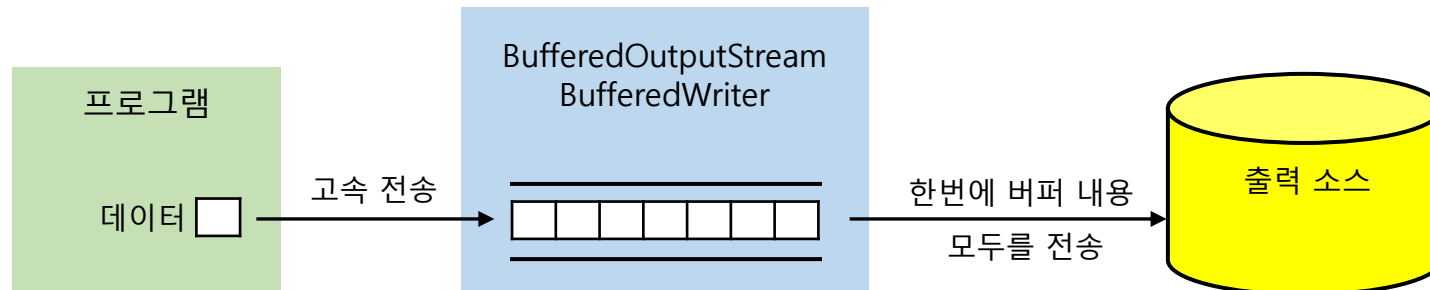
▶ 성능 향상 보조 스트림

느린 속도로 인해 입출력 성능에 영향을 미치는 입출력 소스를 이용하는 경우 사용
입출력 소스와 직접 작업하지 않고 버퍼에 데이터를 모아 한꺼번에 작업을 하여 실행 성능 향상
(입출력 횟수 줄임)

✓ BufferedInputStream/Reader

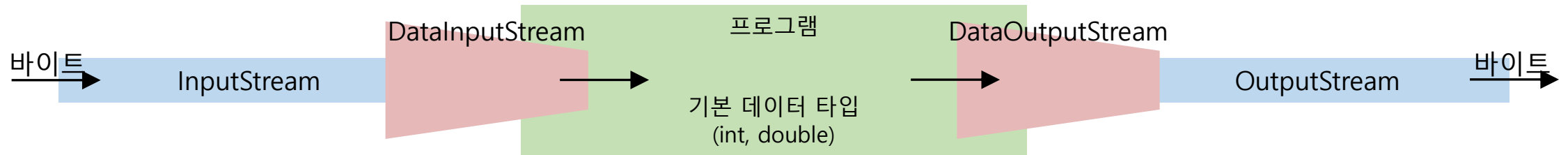


✓ BufferedOutputStream/Writer



▶ 기본 타입 입출력 보조 스트림

기본 자료형 별 데이터 읽고 쓰기가 가능하도록 기능 제공
단, 입력된 자료형의 순서와 출력될 자료형의 순서 일치

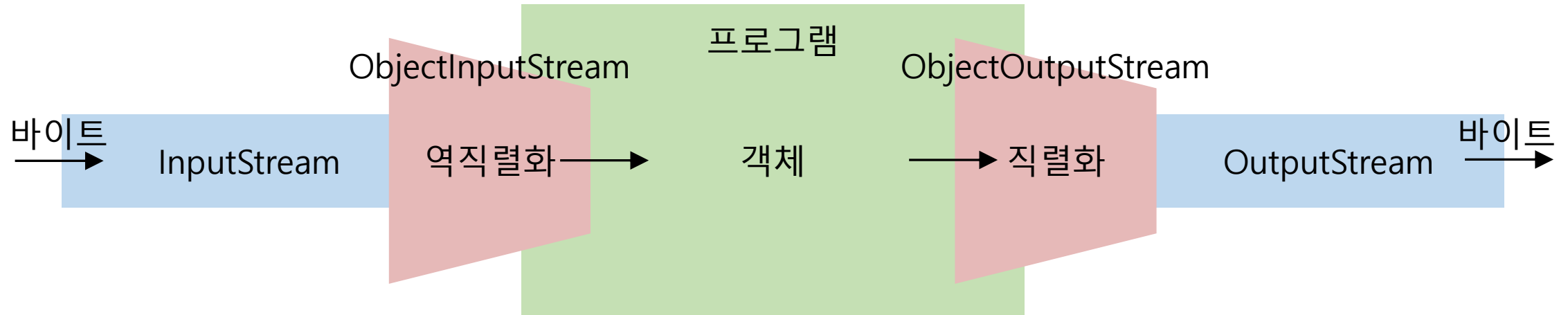


DataInputStream		DataOutputStream	
리턴 타입	메소드	리턴 타입	메소드
boolean	readBoolean()	void	writeBoolean(Boolean v)
byte	readByte()	void	writeByte(int v)
char	readChar()	void	writeChar(int v)
double	readDouble()	void	writeDouble(double v)
float	readFloat()	void	writeFloat(float v)
int	readInt()	void	writeInt(int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

▶ 객체 입출력 보조 스트림

객체를 파일 또는 네트워크로 입출력 할 수 있는 기능 제공

단, 객체는 문자가 아니므로 바이트 기반 스트림으로 데이터를 변경해주는 직렬화 필수



▶ 직렬화와 역직렬화

✓ 직렬화(Serialization)

Serializable 인터페이스를 implements하여 구현
객체 직렬화 시 private 필드를 포함한 모든 필드를 바이트로 변환하지만
transient 키워드를 사용한 필드는 직렬화에서 제외

✓ 역직렬화(Deserialization)

직렬화된 객체를 역직렬화할 때는 직렬화 했을 때와 같은 클래스 사용
단, 클래스 이름이 같더라도 클래스 내용이 변경된 경우 역직렬화 실패

✓ serialVersionUID 필드

직렬화한 클래스와 같은 클래스임을 알려주는 식별자 역할로 컴파일 시 JVM이 자동으로
serialVersionUID 정적 필드를 추가해줘 별도로 작성하지 않아도 오류는 나지 않지만
자동 생성 시 역직렬화에서 예상하지 못한 InvalidClassException을 유발할 수 있어 명시 권장

```
private static final long serialVersionUID = -6423919775137290062L;
```