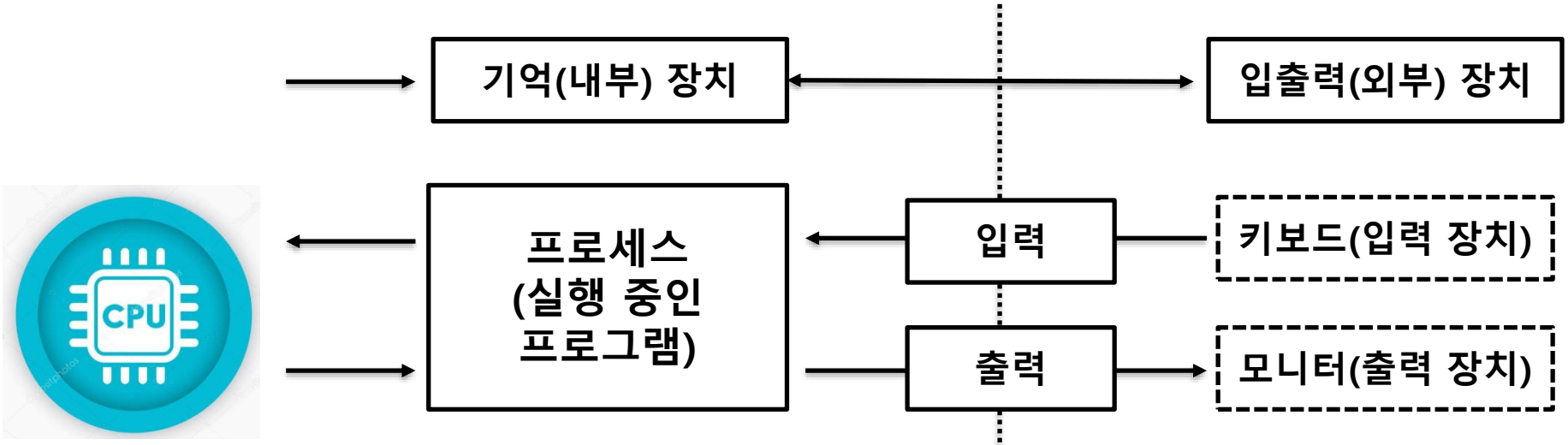

JAVA
입출력

입출력

1. 스트림이란?
2. 스트림의 분류
3. 형식 입출력
4. 명령어행에서 입출력
5. 파일 입출력

스트림(stream)

입출력이란 외부장치에서 컴퓨터의 주기억 장치로 데이터가 이동하는 과정인 입력과 반대로 컴퓨터의 주기억 장치에서 외부 장치로 데이터가 이동하는 과정인 출력을 줄인 말이다. 즉 두 대상 간에 이루어지는 데이터 이동이라 하겠다. 외부 장치로는 키보드와 모니터가 대표적이다.



메모리에서 외부 장치로 데이터를 내보내는 것을 출력(output), 외부 장치에서 메모리로 데이터를 받아들이는 것을 입력(input)이라고 한다.

스트림(stream)

스트림(stream)은 데이터의 흐름을 형성해 주는 통로를 의미한다. 순서가 있는 데이터의 연속적인 흐름을 의미한다.

스트림은 장치와 프로그램 사이를 연결하는 가상의 파이프(pipe)로 보면 된다. 이 파이프를 통해 바이트들이 순서대로 이동하게 된다. 스트림은 소프트웨어적으로 구현된 논리적인 장치라고 생각하면 이해하기 쉽다.

스트림
- 입력스트림 프로그램으로 데이터를 읽어 들이는 스트림
- 출력스트림 프로그램으로부터 데이터를 내보내는 스트림

스트림(stream)

프로그램이 데이터를 입력받을 때에는 입력 스트림이라 부르고, 프로그램이 데이터를 보낼 때에는 출력 스트림이라고 부른다.

스트림은 데이터의 입출력을 도와주는 중간 매개체이다. 데이터의 흐름이나 통로로 보면 이해하기 쉽다. 자바는 이러한 스트림을 통해서 입출력을 수행한다. **하나의 스트림은 하나의 방향만 가능하다.** 따라서 입력과 출력을 동시에 하려면 입력 스트림과 출력 스트림이 각각 필요하다.

스트림(stream)

test.txt파일에 저장된 데이터를 읽어들이기 위한 스트림을 형성한다고 가정해보자. 다음 한 문장으로 간단히 스트림을 형성할 수 있다.

```
FileReader in = new FileReader("test.txt");
```

- 스트림의 형성이라는 것이 결국은 인스턴스의 생성이다.
- FileReader은 입력 스트림의 형성을 위한 클래스이다.

  그런데 그 대상이 파일이다. 즉 파일과의 입력 스트림 형성을 위한 클래스이다. 파일에 저장되어 있는 데이터를 읽어 들이는 통로가 형성되는 셈이다.

스트림의 특징

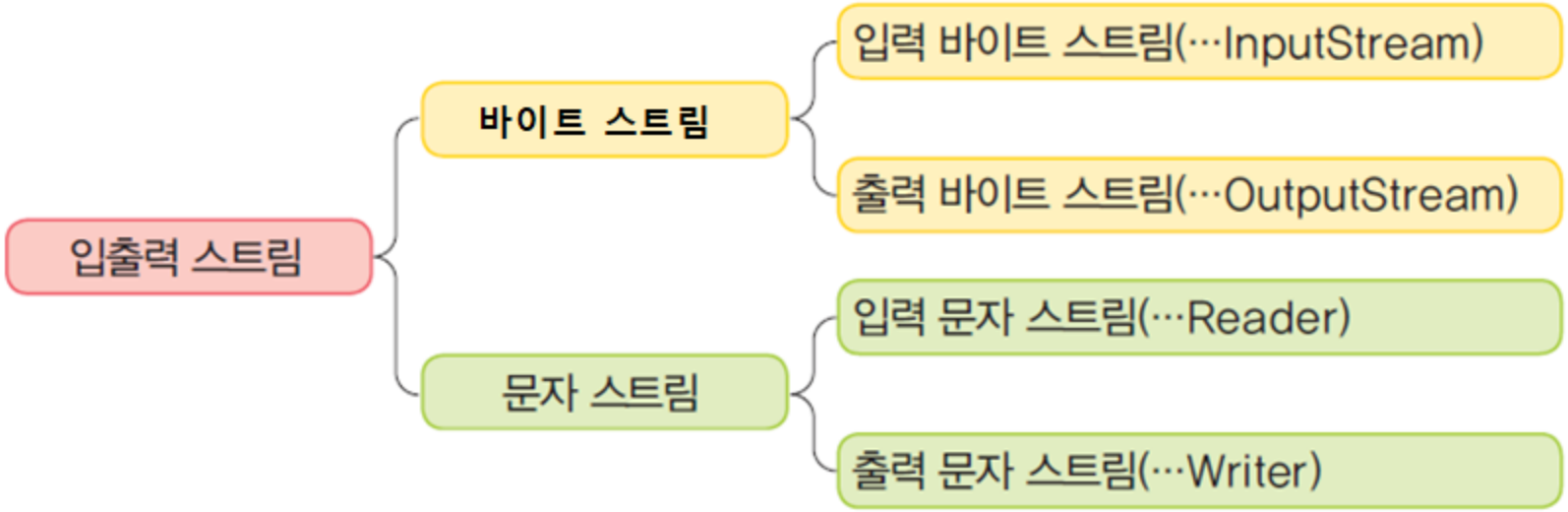
스트림의 특징은 다음과 같다.

- 스트림은 FIFO(First In First Out) 구조를 갖는다.
  먼저 입력된 데이터가 먼저 출력되는 형태이며, 이 때문에 순차적 접근만 가능하다.
- 스트림은 단방향이므로 하나의 스트림으로 입출력을 동시에 수행하지 못한다. 양방향의 전송이 필요하면 출력 스트림과 입력 스트림 각각이 필요하다

스트림의 분류 #1

java.io 패키지에 존재하며 데이터의 종류에 따라 크게 2가지로 나눈다.

- 바이트 스트림(byte stream) : 바이트 단위로 읽고 쓰기 위한 클래스.(그림, 멀티미디어, 문자 등 모든 종류의 데이터를 받고 보내는 것 가능하다.)
- 문자 스트림(character stream) :문자 단위(유니코드)로 읽고 쓰기 위한 클래스.(문자만 받고 보낼 수 있도록 특화)



스트림의 분류 #1

- 바이트 스트림(byte stream)은 데이터를 1바이트(8비트) 단위로 전송한다.
  - ✓ 바이트 스트림 클래스들은 추상 클래스인 InputStream와 OutputStream 클래스가 최상위 클래스이다.
  - ✓ 바이트 스트림 클래스 이름에는InputStream(입력)과 OutputStream(출력)이 붙는다.
  - ✓ 파일과 객체, 기본형의 입출력에 사용한다.
- 문자 스트림(character stream)은 데이터를 2바이트 단위로 전송한다.
  - ✓ 이들은 모두 추상 클래스인 Reader와 Writer 클래스가 최상위 클래스이다.
  - ✓ 문자 스트림 클래스 이름에는 Reader(입력)와 Writer(출력)가 붙는다.
  - ✓ 문자만 입출력 할 수 있다.

스트림의 분류 #2

기능에 따라 크게 2가지로 나눈다.

- 1차 스트림 : 실제 데이터를 주고받는 입출력 스트림을 직접 생성한다. 기본 기능을 담당하여 대상과 직접 데이터를 주고 받는다.

- 2차 스트림 : 1차 스트림에 새로운 기능을 추가해 확장한 스트림이다. 1차 스트림이나 다른 2차 스트림에 붙어서 효율성을 높여준다. 자체만으로는 입출력을 수행하지 못한다.

스트림의 분류 #2

스트림 클래스들은 소스나 목적지에 데이터를 입출력하는 데이터 싱크(Data Sink Class)와 데이터를 가공하는 데이터 처리 클래스(Data Processing Class)로 분류할 수 있다.

데이터 싱크 스트림

데이터 싱크 스트림을 표로 정리하면 다음과 같다.

소스/목적지	문자 스트림	바이트 스트림
메모리	CharArrayReader	ByteArrayInputStream
	CharArrayWriter	ByteArrayOutputStream
	StringReader	StringBufferInputStream
	StringWriter	
파일	FileReader	FileInputStream
	FileWriter	FileOutputStream

데이터 처리 스트림

내용	문자스트림	바이트 스트림	설명
버퍼링	BufferedReader	BufferedInputStream	효율성을 위하여 입출력 시에 버퍼링을 한다.
	BufferedWriter	BufferedOutputStream	
필터링	FilterReader	FilterInputStream	필터링을 위한 추상 클래스들
	FilterWriter	FilterOutputStream	
문자와 바이트 변환	InputStreamReader		문자스트림와 바이트 스트림 간의 연결 역할을 한다
	OutputStreamWriter		

데이터 처리 스트림

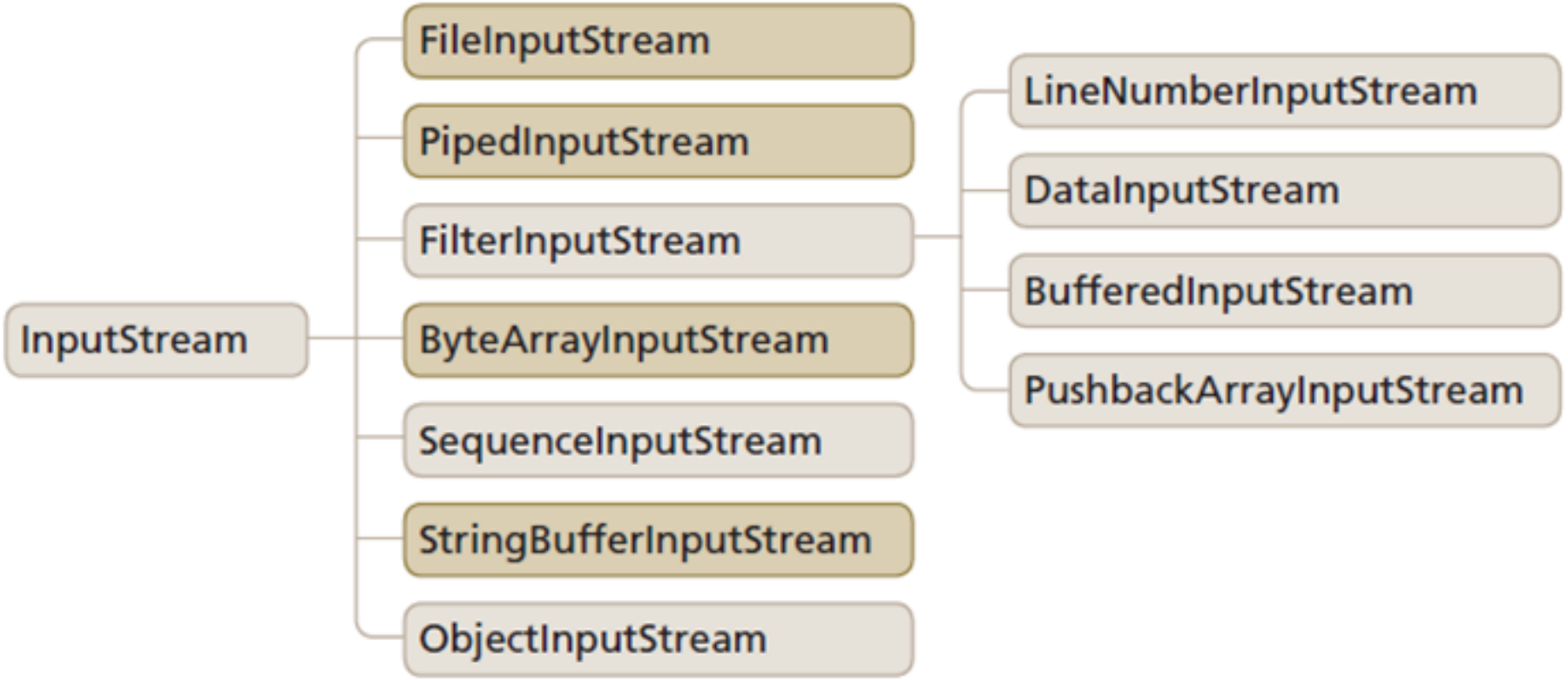
내용	문자스트림	바이트 스트림	설명
결합		SequenceInputStream	여러 개의 입력 스트림을 하나의 입력 스트림으로 결합
객체 직렬화		ObjectInputStream	객체를 직렬화하는데 사용
		ObjectOutputStream	
데이터변환		DataInputStream	기초 자료형 데이터를 읽거나 쓴다
		DataOutputStream	
문장 번호 세기		LineNumberInputStream	입력시 줄 수를 센다
출력	PrintWriter	PrintStream	사용이 편리한 출력메소드제공

중간 점검 문제

1. 문자 스트림과 바이트 스트림의 차이점은 무엇인가?
2. 데이터 씽크 스트림과 데이터 처리 스트림의 차이점은 무엇인가?

바이트 스트림

바이트 스트림은 바이트 단위로 입출력을 수행한다. 모든 바이트 스트림은 InputStream과 OutputStream에서 파생된다.



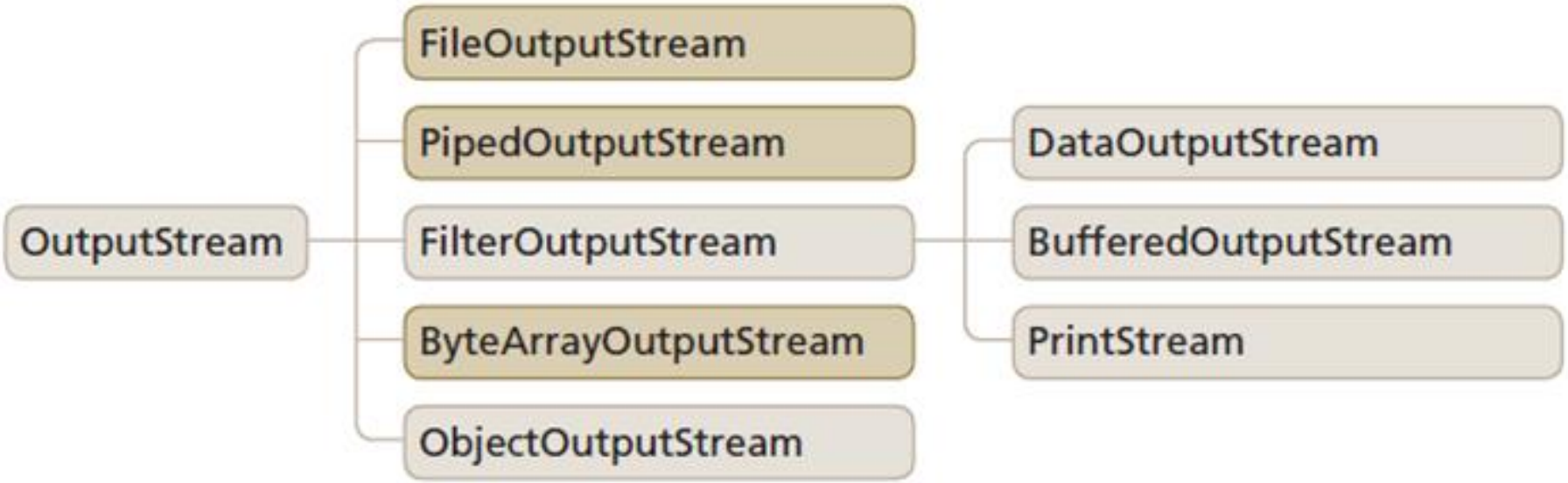
InputStream 클래스

- **InputStream 클래스에서 파생된 클래스**

클래스	설명
InputStream	바이트 입력 스트림에서 사용하는 추상 클래스
FileInputStream	파일에서 데이터를 바이트 단위로 읽어들이는 클래스
FilterInputStream	필터가 있는 바이트 입력 스트림에서 사용하는 추상 클래스
DataInputStream	바이트 단위가 아니라 기본 자료형인 데이터를 바이트 스트림을 읽어들이는 클래스.
BufferedInputStream	버퍼가 있는 바이트 입력 스트림에서 사용하는 클래스. 버퍼로 인해 성능이 향상된다.
ObjectInputStream	객체 단위로 직력화된 데이터를 역직렬화하여 읽어들이는 클래스

바이트 스트림

바이트 스트림은 바이트 단위로 입출력을 수행한다. 모든 바이트 스트림은 InputStream과 OutputStream에서 파생된다.



OutputStream 클래스

- **OutputStream 클래스에서 파생된 클래스**

클래스	설명
OutputStream	바이트 출력 스트림에서 사용하는 추상 클래스
FileOutputStream	데이터를 바이트 단위로 파일에 저장하는 클래스
FilterOutputStream	필터가 있는 바이트 출력 스트림에서 사용하는 추상 클래스
DataOutputStream	바이트 스트림을 바이트 단위가 아니라 기본 자료형인 데이터로 내보내는 클래스.
BufferedOutputStream	버퍼가 있는 바이트 출력 스트림에서 사용하는 클래스. 버퍼로 인해 성능이 향상된다.
ObjectOutputStream	바이트를 객체 단위로 직렬화하여 저장하는 클래스

InputStream과 OutputStream

InputStream 바이트 입력 스트림의 최상위 클래스로 추상 클래스이다.

```
abstract int read()
```

메소드	설명
int read()	1바이트의 데이터를 읽어서 반환하는 메소드이다. 더 이상 읽어들일 데이터가 존재하지 않는 상황에서 반환되는 것은 -1이다.
int read(byte[] buf)	입력스트림을 통해서 읽어 들여진 데이터들이 배열에 저장된다. 실제 읽어들인 데이터와 바이트 크기를 반환한다. 더 이상 읽을 데이터가 존재하지 않으면 -1이 반환된다.
int read(byte[] buf, int off, int len)	입력스트림으로부터 len개의 바이트만큼 읽고 매개변수인 바이트 배열 buf[off]부터 len개까지 저장한다. 그리고 실제로 읽은 바이트 수인 len개를 리턴한다. 만약 len개를 읽지 못하면 실제로 읽은 바이트 수를 리턴한다.

InputStream과 OutputStream

OutputStream은 바이트 출력 스트림의 최상위 클래스로 추상 클래스이다.

```
abstract void write(int data)
```

메소드	설명
void write(int data)	인자로 전달된 1바이트 데이터를 출력 스트림을 통해 목적지로 전달하는 메소드이다.
void write(byte[] buf)	buf 배열을 출력 스트림을 통해 목적지로 전달하는 메소드이다.
void write(byte[] buf, int off, int len)	매개변수 buf로 전달된 배열을 대상으로 off의 인덱스 위치부터 시작해서 len 바이트를 출력스트림을 통해서 전송하는 메소드이다

FileInputStream과 FileOutputStream

이들 클래스의 입출력 대상은 파일이다. 즉 FileInputStream 클래스는 파일에서 바이트를 읽고 FileOutputStream 클래스는 파일에다가 바이트를 쓴다.

- FileInputStream의 생성자

FileInputStream(File file) File 객체에 연결된 파일로부터 바이트를 읽는 객체를 생성한다.
FileInputStream(String name) 파일의 경로 이름이 name인 파일과 연결된 입력 담당 객체를 생성한다

```
FileInputStream in = new FileInputStream("input.txt");
```

- FileInputStream 객체가 생성될 때 파일과 직접 연결
- 만약 파일이 존재하지 않으면 FileNotFoundException발생
- try-catch문으로 예외 처리

FileInputStream과FileOutputStream

- FileOutputStream의 생성자

FileOutputStream(File file) File 객체가 나타내는 파일에 바이트를 쓰는 객체를 생성한다.
FileOutputStream(String name) 파일의 경로 이름이 name인 파일과 연결된 출력 담당 객체를 생성한다
FileOutputStream(String name, append) append가 true이면 파일의 끝에 바이트를 추가하고 false이면 바이트를 기존의 바이트 위에 겹쳐서 쓴다.

```
FileOutputStream in = new FileOutputStream("output.txt", false);
```

- 파일이 이미 존재할 경우, 데이터를 출력하게 되면 **파일을 덮어쓰기가 기본값으로 설정**되어 있다.

FileInputStream과FileOutputStream

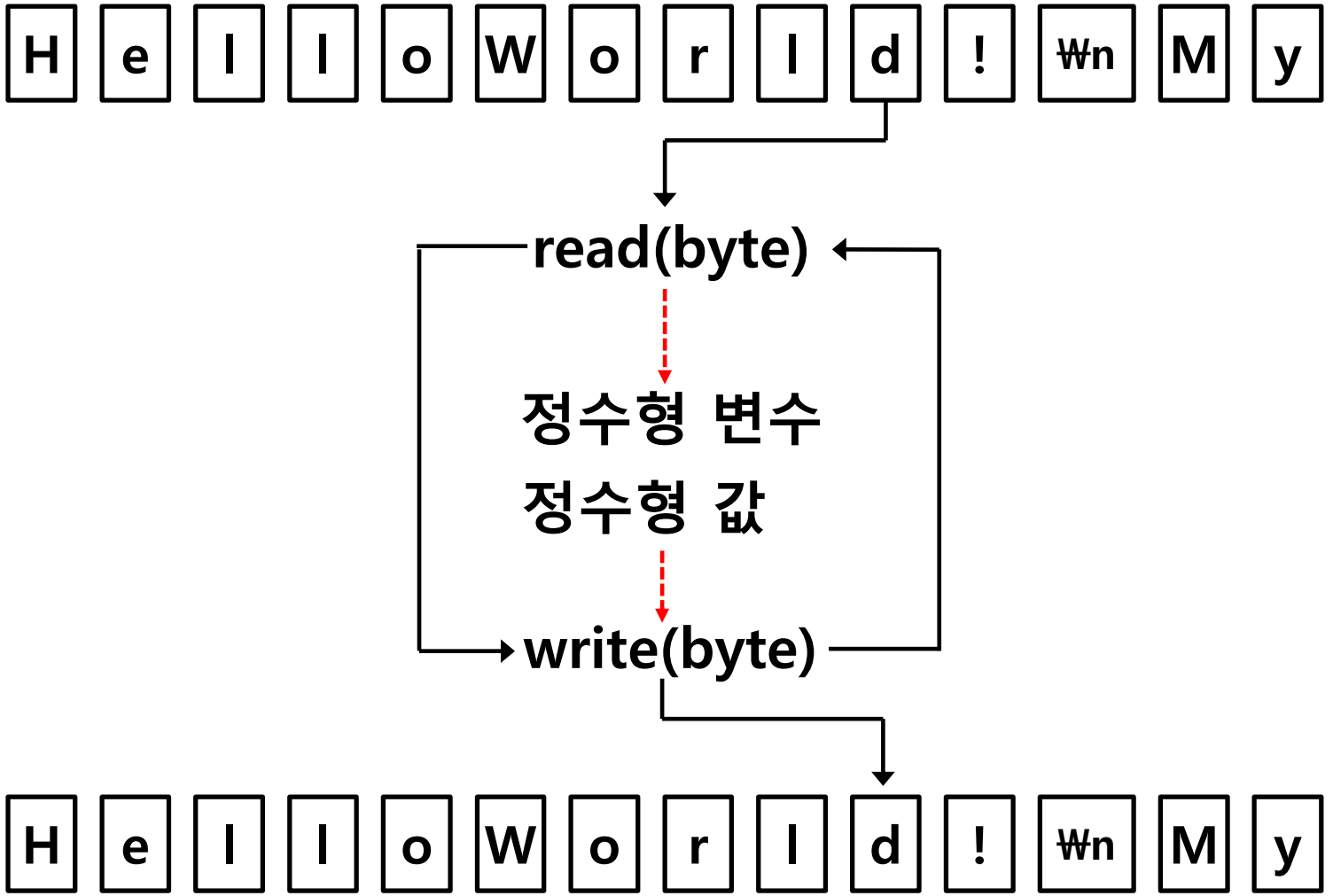
```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class InputOutputStreamExample {
   public static void main(String[] args){
      FileInputStream fis = null;
      FileOutputStream fos = null;
         try{
            fis = new FileInputStream("C:/Temp/intest.txt");
            fos = new FileOutputStream("C:/Temp/outtest.txt");
            int data = -1;
            while((data = fis.read()) != -1) {
                    fos.write(data);
            }
          }
```

FileInputStream과FileOutputStream

```
        }catch(IOException io){
            System.out.println("파일 입출력에 문제가 발생하여
                        더 이상의 작업을 진행할 수 없습니다.");
        }catch(Exception e){
            System.out.println("오류 발생으로 더 이상의 작업을
                        진행할 수 없습니다.");
        }finally{
            try{
                    if(fis!=null) fis.close();
                    if(fos!=null) fos.close();
            }catch(Exception e){
                    e.printStackTrace();
            }
        }
    }
}
```

FileInputStream과FileOutputStream

다음은 이러한 과정을 도식화한 그림이다.



자동 리소스 닫기

- try-with-resources
  - 예외 발생 여부와 상관 없음
  - try-with-resources 문장은 문장의 끝에서 리소스들이 자동으로 닫혀지게 한다. 즉 사용했던 리소스 객체의 close() 메소드 호출해 리소스 닫음
  - try-with-resources 문장은 Java SE 7버전부터 추가되었다.
  - 리소스 객체
    - 각종 입출력스트림 등
    - java.lang.AutoCloseable 인터페이스 구현하고 있어야 함

```
try(리소스자료형1 참조변수 = new 리소스자료형1();
    리소스자료형2 참조변수 = new 리소스자료형2()){
}
```

try-with-resources

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class InputOutputStreamExample {
   public static void main(String[] args){
      try (FileInputStream fis = new FileInputStream("C:/Temp/intest.txt");
           FileOutputStream fos =
                                   new FileOutputStream("C:/Temp/outtest.txt")) {

            int data = -1;
            while((data = fis.read()) != -1) {
                  fos.write(data);
            }
      } catch(Exception e) {
               e.printStackTrace();
  }
}
```

보조 스트림(필터 스트림)

- 필터 스트림(보조 스트림)
  - 다른 스트림과 연결 되어 여러 가지 편리한 기능을 제공해주는 스트림
    - ✓ 문자 변환, 입출력 성능 향상, 기본 데이터 타입 입출력, 객체 입출력 등의 기능을 제공

- 필터 스트림 생성

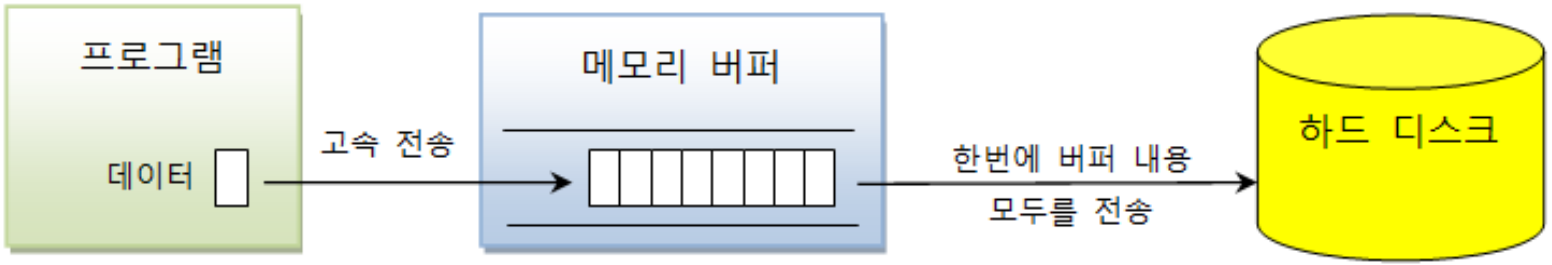
```
필터 스트림 변수 = new 필터스트림(연결스트림);
```

- 필터 스트림 체인 : 다른 보조 스트림과 연결되어 역할 수행

BufferedInputStream과 BufferedOutputStream

- 입출력 성능에 영향을 미치는 입출력 소스
  - 하드 디스크
  - 느린 네트워크
- 버퍼를 이용한 해결

  입출력 소스와 직접 작업하지 않고 버퍼(buffer)와 작업
  - 실행 성능 향상



- 프로그램은 쓰기 속도 향상
- 버퍼 차게 되면 데이터를 한꺼번에 하드 디스크로 보내 출력 횟수를 줄여줌

BufferedInputStream과 BufferedOutputStream

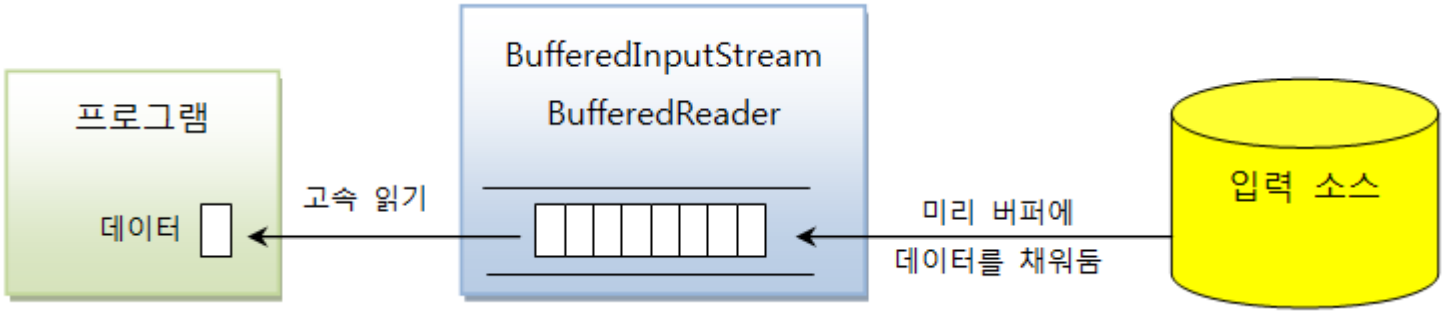
BufferedInputStream과 BufferedOutputStream는 버퍼링의 기능을 제공하는 필터 스트림들이다. 버퍼의 크기를 지정하지 않으면 디폴트의 버퍼 크기가 8192byte이다.

버퍼 입력 스트림은 버퍼(임시 저장소)라고 알려진 메모리 영역에서 데이터를 읽는다. 버퍼가 비었을때만 디스크나 네트워크에서 읽는다. 비슷하게 버퍼 출력 스트림은 데이터를 버퍼에다 쓴다. 버퍼가 가득찼을 경우에만 디스크나 네트워크에 쓴다. BufferedInputStream과 BufferedOutputStream 클래스는 버퍼 기능을 제공한다.

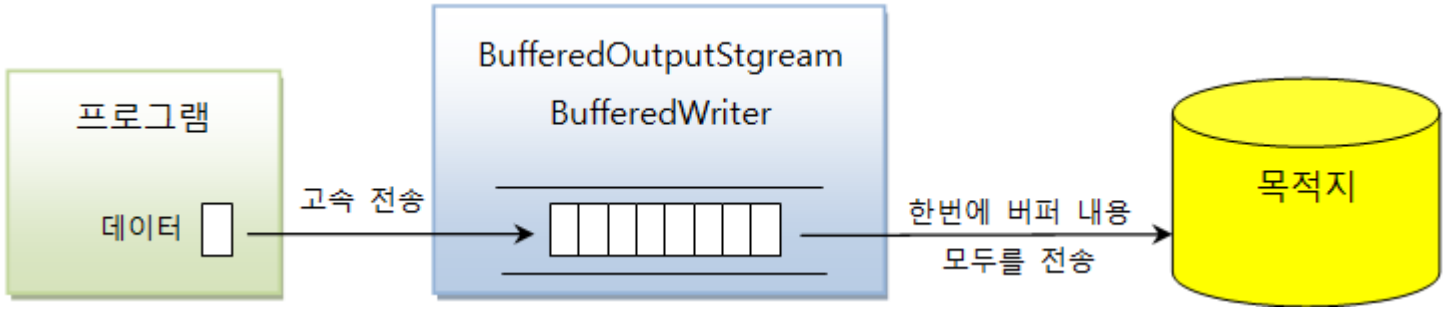
BufferedInputStream과 BufferedOutputStream

- BufferedInputStream과 BufferedOutputStream 생성자

BufferedInputStream(InputStream in)
입력 스트림 in을 감싸는 버퍼 입력 스트림 객체를 생성한다.



BufferedOutputStream(OutputStream out)
출력 스트림 out을 감싸는 버퍼 출력 스트림 객체를 생성한다



BufferedInputStream과 BufferedOutputStream

버퍼가 없는 스트림을 버퍼가 있는 스트림으로 변경하려면 버퍼가 없는 스트림 객체를 버퍼 스트림 클래스의 생성자로 전달하면 된다.

```
FileInputStream fis = new FileInputStream("data.txt"));
BufferedInputStream bis = new BufferedInputStream(fis);

BufferedInputStream bis =
        new BufferedInputStream(new FileInputStream("data.txt"));
```

```
FileOutputStream fos = new FileOutputStream("data.txt"));
BufferedOutputStream bos = new BufferedOutputStream (fos);

BufferedOutputStream bos =
      new BufferedOutputStream(new FileOutputStream("data.txt"));
```

BufferedInputStream과 BufferedOutputStream

어떤 경우에는 버퍼가 다 채워지지 않았어도 버퍼를 쓰는 것이 필요하다. 이것을 비우기(flushing)이라고 한다. 버퍼된 출력 클래스(PrintWriter)에는 자동 비우기 기능이 있다.

버퍼를 수동으로 비우기 위해서는 flush() 메소드를 호출한다. 버퍼가 꽉차지 않아도 출력 스트림을 통해서 파일에 저장해야 할 데이터가 존재한다면 flush() 메소드를 호출해야 한다.

DataInputStream 과 DataOutputStream

필터 스트림 클래스는 입력 스트림에 의해서 읽힌 데이터의 형태를 다양하게 구성하는 기능의 스트림이다.

- DataInputStream : 바이트 단위의 데이터를 기본 자료형으로 변환하는 필터 스트림
- DataOutputStream : 기본 자료형의 데이터를 바이트 단위로 분리해서 출력스트림으로 전송하는 필터 스트림

DataInputStream과 DataOutputStream 클래스는 기초 자료형 단위로 데이터를 읽고 쓸 수 있다.

DataInputStream 과 DataOutputStream

- DataInputStream과 DataOutputStream의 생성자

DataInputStream(InputStream in) InputStream 객체를 인수(in)로 전달받아 DataInputStream 객체를 생성한다.
DataOutputStream(OutputStream out) OutputStream 객체를 인수(out)로 전달받아 DataOutputStrea 객체를 생성한다.

DataInputStream 과 DataOutputStream

- DataInputStream 클래스의 메서드(throws IOException)

메서드	설명
boolean readBoolean()	스트림으로부터 boolean형 데이터를 반환
byte readByte()	스트림으로부터 byte형 데이터를 반환
char readChar()	스트림으로부터 char형 데이터를 반환
double readDouble()	스트림으로부터 double형 데이터를 반환
float readFloat()	스트림으로부터 float형 데이터를 반환
long readLong()	스트림으로부터 long형 데이터를 반환
short readShort()	스트림으로부터 short형 데이터를 반환
int readInt()	스트림으로부터 int형 데이터를 반환
void readFully(byte[] buf)	스트림으로부터 바이트배열인 buf 크기만큼의 바이트를 읽어 배열buf[]에 저장
String readUTF()	UTF 인코딩 값을 얻어 문자열로 반환

DataInputStream 과 DataOutputStream

- DataOutputStream 클래스의 메서드(throws IOException)

메서드	설명
void writeBoolean(boolean b)	인수로 전달된 boolean형 b를 1바이트만 읽어서 출력스트림에 출력
void writeByte(int i)	인수로 전달된 int형 값을 하위 1바이트만 읽어서 출력스트림에 출력
void writeBytes(String s)	인수로 전달된 문자열 중 하위 1바이트만 읽어서 출력 스트림에 출력한다. String형 문자열을 바이트 순으로 출력 스트림에 출력
void writeChar(int i)	인수로 전달된 i를 char형(2byte) 값으로 상위 바이트 먼저 출력 스트림에 출력
void writeDouble(double d)	double 클래스의 doubleToBits()를 사용하여 int형으로 변환한다. 그런 다음 long형을 8바이트 값으로 상위 바이트 먼저 출력 스트림에 출력

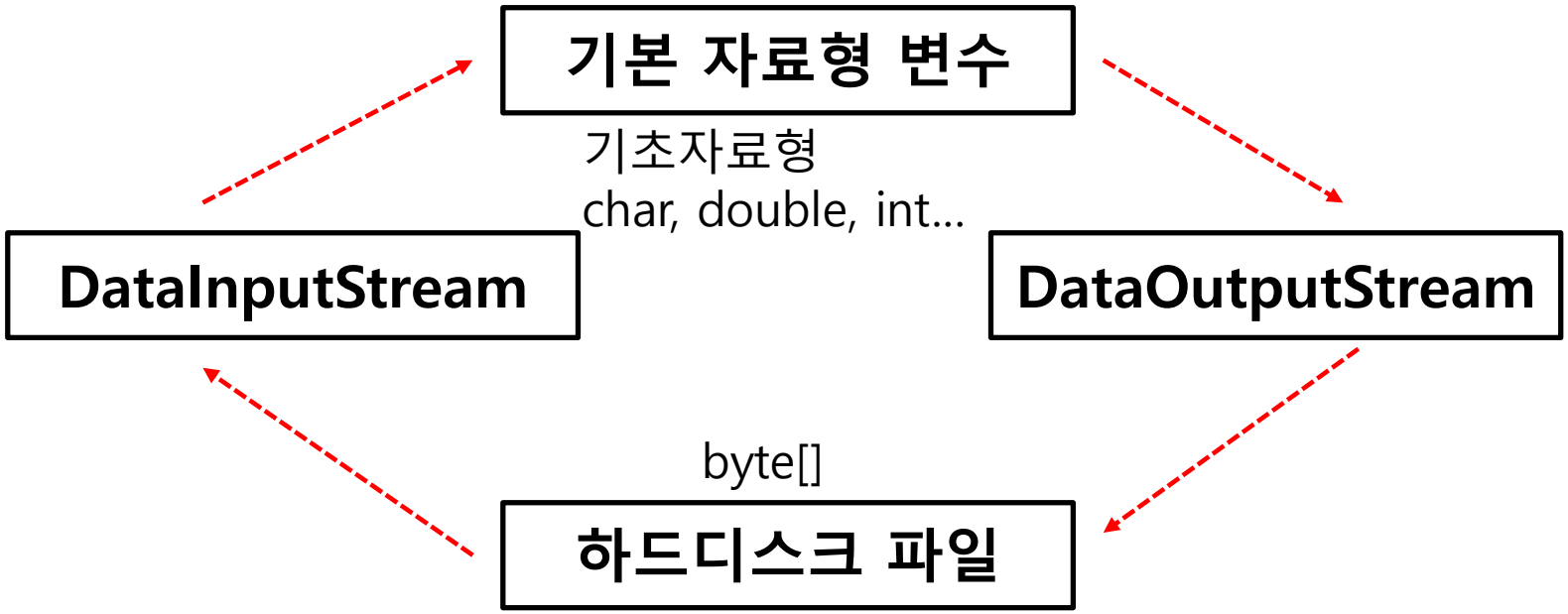
DataInputStream 과 DataOutputStream

- DataOutputStream 클래스의 메서드(throws IOException)

메서드	설명
void writeFloat(float f)	float 클래스의 floatToBits()를 사용하여 int 형으로 변환한다. 그런 다음 long형을 8바이트 값으로 상위 바이트 먼저 출력 스트림에 출력
void writeInt(int i)	인수로 전달된 int형 값을 상위 바이트 먼저 출력 스트림에 출력
void writeLong(long l)	인수로 전달된 값을 long형(8byte)값을 상위 바이트 먼저 출력 스트림에 출력
void writeShort(short s)	인수로 전달된 값을 short형(4byte)값을 상위 바이트 먼저 출력 스트림에 출력
void writeUTF(String s)	인수로 전달된 문자열을 UTF-8 인코딩 처리해서 출력 스트림에 출력

DataInputStream 과 DataOutputStream

FileInputStream와 FileOutputStream 클래스는 바이트 단위의 데이터만 입출력 할 수 있다. 하지만 데이터 스트림을 필터로 적용하면 자바 기본 자료형으로 데이터를 입력하고 출력할 수 있다.



ObjectInputStream과 ObjectOutputStream

DataStream이 기초적인 자료형의 입출력을 지원하는 것처럼 ObjectStream은 객체의 입출력을 지원한다. 이 클래스를 이용하면 객체를 파일에 저장할 수도 있고 또 반대로 파일에 저장된 객체를 읽어 들일 수 있다.

객체를 저장한다고 하면 도대체 어떤 것이 저장되는 것인가? 하나의 객체는 인스턴스 변수(필드)와 메소드를 가지고 있다. 객체를 저장할 때 객체마다 메소드를 저장할 필요는 없다. 객체마다 달라지는 것은 인스턴스 변수이다. 인스턴스 변수만 저장하면 객체의 현재 상태를 저장하는 것이 된다.

직렬화라는 것은 클래스의 구성을 파일이나 네트워크로 전달하기 위해 형태를 재구성하는 것이다.

ObjectInputStream과 ObjectOutputStream

객체를 파일에 저장하려면 객체가 가진 데이터(필드값)들을 일렬로 늘어선 연속적인 바이트로 변환하는 절차가 필요하다. 이것을 직렬화(serialization)라고 한다.

직렬화란 객체를 순차적인 바이트로 표현한 데이터를 의미한다. 어떤 클래스가 직렬화를 지원하려면 Serializable이라는 인터페이스를 구현하면 된다. 객체가 직렬화된 데이터를 읽어서 자신의 상태(객체)로 복구(복원)하는 것을 역직렬화(deserialization)라고 한다.

ObjectInputStream과 ObjectOutputStream

자바는 Serializable 인터페이스를 구현한 클래스만 직렬화할 수 있도록 제한하고 있다. 객체를 직렬화할 때 private 필드를 포함한 모든 필드를 바이트로 변환해도 좋다는 표시 역할을 한다. 필드 선언에 transient가 붙어 있을 경우 직렬화가 되지 않는다.

```
public class 클래스명 implements Serializable { ... }
```

ObjectInputStream과 ObjectOutputStream

객체를 입출력할 수 있는 두 개의 스트림은 ObjectInputStream과 ObjectOutputStream을 제공한다.

- ObjectOutputStream : 바이트 출력 스트림과 연결되어 객체를 직렬화하는 역할을 한다.

ObjectOutputStream(OutputStream out) 인수로 전달된 출력 스트림(out)을 직렬화하기 위해 ObjectOutputStream의 객체를 생성한다.

- ObjectInputStream : 바이트 입력 스트림과 연결되어 객체로 역직렬화하는 역할을 한다.

ObjectInputStream(InputStream in) 인수로 전달된 입력 스트림(in)을 역직렬화하기 위해서 ObjectInputStream 객체를 생성한다.

ObjectInputStream과 ObjectOutputStream

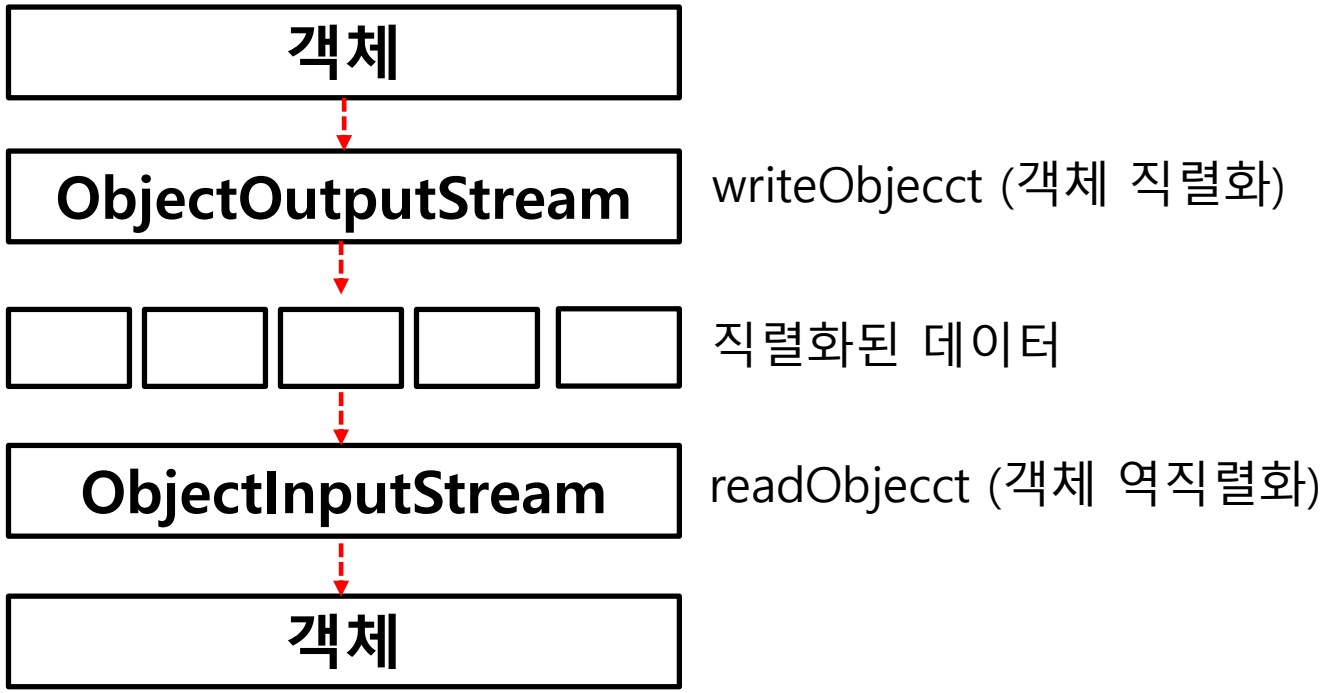
- ObjectOutputStream으로 객체를 직렬화하기 위해서는 writeObject() 메서드를 사용한다.

- ObjectInputStream의 readObject() 메서드는 입력 스트림에서 읽은 바이트를 역직렬화해서 객체로 생성한다. readObject() 메서드의 리턴 타입은 Object타입이기 때문에 객체 원래 타입으로 변환해야 한다.

ObjectInputStream과 ObjectOutputStream

- 객체를 전송하려면 다음의 세 단계를 거친다.

① 객체를 바이트 단위로 나눈다. (객체 직렬화)
② 직렬화되어 나누어진 데이터를 순서대로 전송한다.
③ 전송받은 데이터를 원래대로 복원한다. (객체 역직렬화)



ObjectInputStream과 ObjectOutputStream

직렬화된 객체를 역직렬화 했을 때와 같은 클래스를 사용해야 한다. 클래스의 이름이 같다라도 클래스의 내용이 변경되면, 역직렬화는 실패하게 그때 serialVersionUID가 다르다는 예외가 발생한다.

직렬화 당시 클래스	역직렬화 당시 클래스

class A
implements Serializable{
private int no;
private String name;
…
}
	class A
	implements Serializable{
	private int no;
	private String name;
	private int age;
	…
	}

serialVersionUID는 같은 클래스임을 알려주는 식별자 역할을 하는데, Serializable 인터페이스를 구현한 클래스를 컴파일하면 자동적으로 serialVersionUID가 추가된다.

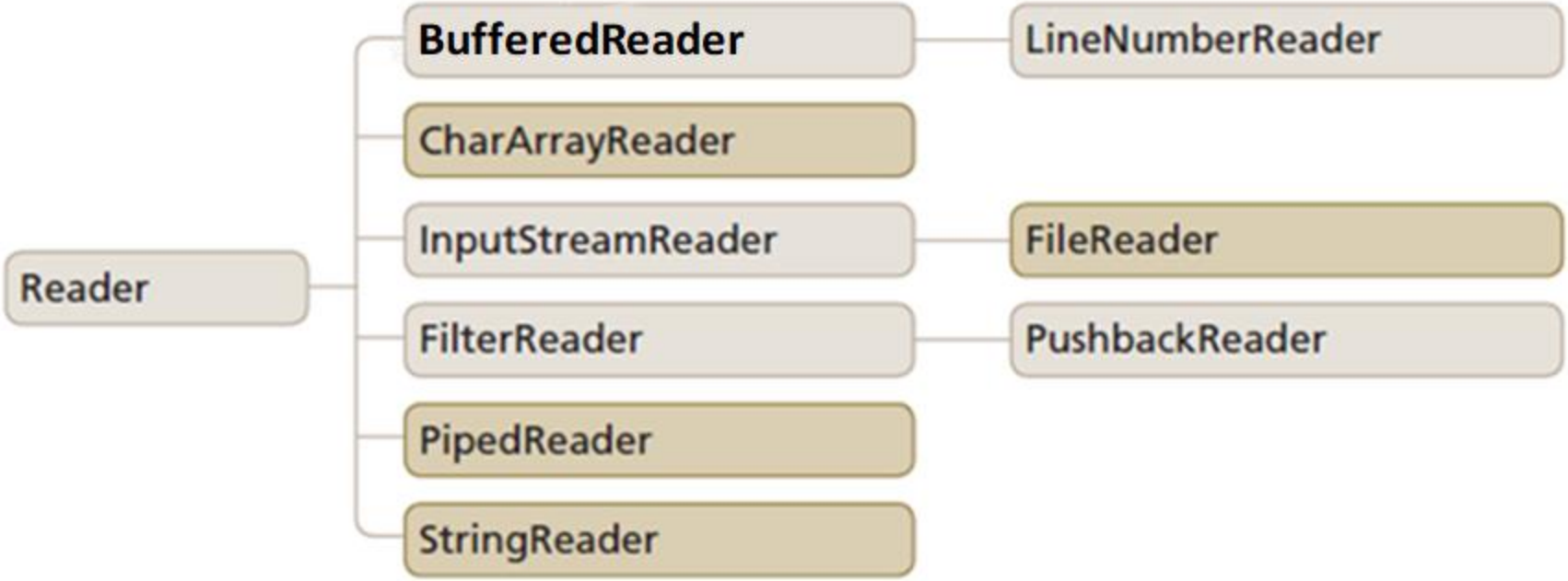
중간 점검 문제

1. 파일 base.txt에서 바이트 형태로 버퍼를 사용하여서 데이터를 읽는 스트림을 생성하여 보라.

2. double형의 데이터를 저장하였다가 다시 읽으려면 어떤 스트림 클래스가 적합한가?

문자 스트림

문자 스트림은 입출력 단위가 문자가 된다. 자바는 유니코드를 기반으로 문자를 표현한다. 이렇듯 정해진 규칙을 기준으로 문자를 수의 형태로 표현하는 것을 가리켜 인코딩이라 한다.



Reader와 Writer 클래스

- **Reader 클래스에서 파생된 클래스**

클래스	설명
Reader	문자 입력 스트림을 위한 최상위 클래스이며 추상 클래스
BufferedReader	문자 버퍼 기반의 입력 스트림을 처리
InputSreamReader	바이트 스트림을 문자 스트림으로 변환
FileReader	파일에서 바이트를 읽어 들여 문자 스트림으로 변환

문자 스트림

문자 스트림은 입출력 단위가 문자가 된다.

Reader와 Writer 클래스

- **Writer 클래스에서 파생된 클래스**

클래스	설명
Writer	문자 출력 스트림을 위한 최상위 클래스이며 추상 클래스
BufferedWriter	문자 버퍼 출력 스트림을 생성
OutputStreamWriter	문자 스트림을 바이트 스트림으로 변환하는 스트림을 생성
FileWriter	문자 데이터를 바이트로 변환한 출력 스트림

Reader와 Writer 클래스

Reader는 문자 입력 스트림의 최상위 클래스로 추상 클래스이다. Writer 클래스는 문자 출력 스트림의 최상위 클래스로 추상 클래스이다.

- Reader 클래스의 메소드

메소드	설명
int read()	파일로부터 읽어 들인 문자 하나를 반환한다. 읽을 값이 없으면 -1을 반환
int read(char[] cbuf)	buf의 크기만큼 데이터를 읽어서 buf에 저장하고 읽은 바이트 수를 반환
int read(char[] cbuf, int offset, int length)	최대 len의 개수만큼 문자를 읽어 들여서, cbuf로 전달된 배열의 인덱스 위치 off에서부터 문자를 저장한다. 그리고 실제로 읽어들인 문자의 수를 반환한다. 더 이상 읽어 들일 문자가 존재하지 않다면 -1을 반환한다.

Reader와 Writer 클래스

메소드	설명
void mark(int readAheadLimit)	스트림의 현재 위치를 표시해 놓는다.
boolean markSupported()	마크 기능이 지원되는지 여부를 반환한다.
abstract void close()	스트림을 닫고 모든 자원을 반납한다.
boolean ready()	스트림이 읽을 준비가 되었는지 여부를 반환한다.
void reset()	스트림을 리셋한다.
long skip(long n)	n개의 문자를 건너뛴다.

Reader와 Writer 클래스

- Writer 클래스의 메소드

메소드	설명
int write(int c)	파일에 하나의 문자를 저장한다.
void write(char[] cbuf)	cbuf[]에 저장된 문자들을 출력한다.
void write(char[] cbuf, int off, int len)	buf로 전달된 배열의 인덱스 위치 off에서부터 len개의 문자를 파일에 저장한다.
void write(String str)	출력 스트림으로 주어진 문자열을 전부 보낸다.
void write(String str, int off, int len)	출력 스트림으로 주어진 문자열 off 순번부터 len개까지의 문자를 보낸다.

Reader와 Writer 클래스

- Writer 클래스의 메소드

메소드	설명
abstract void flush()	스트림 버퍼의 데이터를 모두 출력한다.
abstract void close()	스트림을 닫고 모든 자원을 반납한다.
Writer append(char c)	문자 c를 스트림에 추가한다.
void mark(int readAheadLimit)	스트림의 현재 위치를 표시해 놓는다.

- 문자 출력 스트림은 내부에 작은 버퍼가 있어서 데이터가 출력되기 전에 버퍼에 쌓여있다가 순서대로 출력된다. flush() 메서드는 버퍼에 잔류하고 있는 데이터를 모두 출력시키고 버퍼를 비우는 역할을 한다.

FileReader와 FileWriter

FileReader와 FileWriter 클래스는 파일로부터 문자를 읽고 쓰는데 사용된다. 모든 문자 스트림은 Reader와 Writer로부터 파생된다.

- FileReader 클래스의 생성자

FileReader(File file) 주어진 file을 이용하여 새로운 FileReader 객체를 생성한다.
FileReader(String fileName) 주어진 파일의 이름을 이용하여 새로운 FileReader 객체를 생성한다.

FileReader와 FileWriter

- FileWriter 클래스의 생성자

FileWriter(File file) 주어진 file을 이용하여 새로운 FileWriter 객체를 생성한다.
FileWriter(String fileName) 주어진 파일의 이름을 이용하여 새로운 FileWriter 객체를 생성한다.
FileWriter(String filename, boolean append) append가 true이면 주어진 파일의 끝에 텍스트를 추가하고, append가 false이면 파일 처음부터 다시 텍스트를 저장한다.

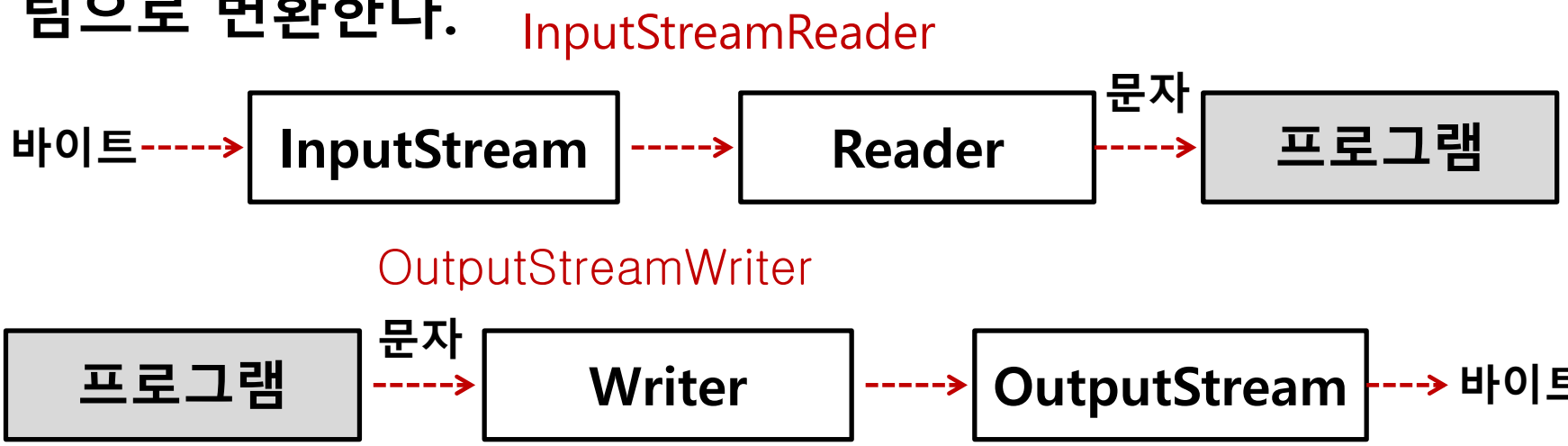
인코딩

인코딩(encoding)은 응용 프로그램의 데이터를 스트림 형식으로 변환해 보조 기억 장치나 네트워크에서 사용 가능하도록 하는 작업이다.

ANSI(미국 표준)	한글 2byte, 영어(숫자) 1byte
EUC-KR	한글 2byte, 영어(숫자) 1byte
UTF-8	한글 3byte, 영어(숫자) 1byte

InputStreamReader와 OutputStreamWriter

바이트 스트림과 문자 스트림을 연결하는 클래스이다. 즉 InputStreamReader(바이트 스트림을 문자 스트림 변환)는 바이트 입력 스트림에 연결되어 문자 입력 스트림으로 변환한다. OutputStreamWriter(문자 스트림을 바이트 스트림 변환)는 바이트 출력 스트림에 연결되어 문자 출력 스트림으로 변환한다.



InputStreamReader와 OutputStreamWriter

효율성을 위하여 InputStreamReader를 BufferedReader로 감싸는 것이 좋다.

InputStreamReader(InputStream file)

디폴트 문자 집합을 이용하여서 InputStreamReader 객체를 생성한다.

```
BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
```

InputStreamReader와 OutputStreamWriter

```
public class InputStreamReaderExample {
public static void main(String[] args) throws Exception{
    InputStream is = System.in;
    Reader reader = new InputStreamReader(is);

    int readCharNo;
    String data ;
    char[] cbuf = new char[100];
    while ((readCharNo=reader.read(cbuf)) != -1) {
        data = new String(cbuf, 0, readCharNo-2);
        System.out.println(data);
    }
    reader.close();
   }
}
```

InputStreamReader와 OutputStreamWriter

효율성을 위하여 BufferedWriter로 감싸는 것이 좋다.

OutputStreamWriter(OutputStream out)

지정된 문자 집합을 사용하는 OutputStreamWriter 객체를 생성한다

```
BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(System.out));
```

InputStreamReader와 OutputStreamWriter

```
public class OutputStreamWriterExample {
public static void main(String[] args) throws Exception{
    FileOutputStream fos = new
                        FileOutputStream("C:/Temp/file.txt");
    Writer writer = new OutputStreamWriter(fos);

    String data = "바이트 출력 스트림을 문자 출력
                    스트림으로 변환";
    writer.write(data);
    writer.close();
    fos.close();
    System.out.println("파일 저장이 끝났습니다.");
}
}
```

형식 입출력

특정 형식으로 데이터를 입출력하는 것이 필요하다.

- Scanner 클래스는 입력을 각각의 토큰으로 분리하는 작업도 가능하다. Scanner는 자바의 기초적인 자료형으로도 변환할 수 있다.

- PrintStream과 PrintWriter 클래스는 형식을 가진 출력을 지원하는 스트림 클래스이다. PrintStream은 바이트 입력을 받아서 텍스트 형태로 출력하는 클래스이고 PrintWriter는 문자 입력을 받아서 텍스트 형태로 출력한다.

형식 입출력

PrintStream과 PrintWriter는 모든 기초 자료형을 형식화 시켜서 출력할 수 있는 메소드들을 제공한다.

- print()와 println() 메소드는 각각의 값을 표준화된 방법으로 형식화한다.
- format()과 printf() 메소드는 형식 제어 문자열에 기반하여 거의 모든 값들을 세밀하게 형식화할 수 있다. 자바 1.5 버전부터는 printf() 메소드가 제공된다.

형식 입출력

형식 지정자는 %로 시작하여 생성되는 출력 형식의 종류를 지정하는 글자로 끝난다.

- d는 정수를 십진수로 출력한다.
- f는 부동소수점을 십진수로 출력한다.
- x는 정수를 16진수로 출력한다.
- s는 문자열로 출력한다.
- n은 정확한 줄바꿈을 위해 사용한다. (₩n은 항상 라인피트 문자만을 생성한다.)

표준 스트림

기본적으로 키보드에서 읽으며 모니터로 출력한다. 또한 파일에 대한 입출력과 프로그램 사이의 입출력도 지원한다. 다음은 표준 입출력을 위한 스트림 객체들이다.

System.in(표준 입력) : 콘솔로부터 데이터를 입력받는데 사용한다.

System.out(표준 출력) : 콘솔로부터 데이터를 출력하는데 사용한다.

System.err(표준 오류) : 콘솔로부터 오류를 출력하는데 사용한다.

표준 출력과 표준 오류는 모두 출력을 위한 것이다.

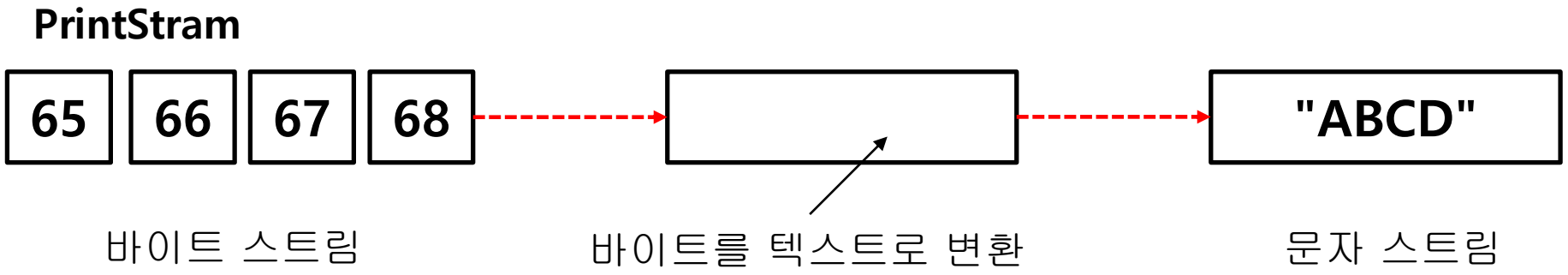
표준 스트림

표준 스트림은 바이트 스트림으로 정의된다.

System.out 과 System.err는 PrintStram의 객체로 정의된다. 비록 기술적으로는 바이트 스트림이지만 PrintStram은 문자 스트림의 특징을 가지고 있다.

반면 System.in은 문자 스트림 특징을 전혀 갖지 않는 바이트 스트림이다. 표준을 문자 스트림으로 사용하기 위해서는 System.in을 InputStreamReader로 둘러 싸야 한다.

```
InputStreamReader isr = new InputStreamReader(System.in);
```



File 클래스

File 클래스는 파일을 조작하고 검사하는 코드를 쉽게 작성하게 해주는 클래스이다.

```
File  file = new File("data.txt");
```

파일에 대한 여러 가지 메소드를 제공한다

메소드	설명
boolean delete()	파일을 삭제한다.
boolean exists()	파일의 존재 여부를 출력한다.
String getAbsolutePath()	파일의 절대 경로를 반환한다.
String getName()	파일의 이름을 반환한다.

File 클래스

파일에 대한 여러 가지 메소드를 제공한다.

메소드	설명
boolean isDirectory()	디렉토리이면 참(true)을 반환한다
boolean isFile()	파일이면 참(true)을 반환한다
long length()	파일 길이를 반환한다.
String[] list()	디렉토리 안에 포함된 파일과 디렉토리를 반환한다.
boolean mkdir()	디렉토리를 생성한다
boolean renameTo (File dest)	파일 이름을 변경한다.

임의 접근 파일

임의 접근 파일은 파일에 비순차적인 접근을 가능하게 한다. 파일 안에서 임의의 위치로 가려면 RandomAccessFile 클래스를 사용하여야 하면 된다. 파일이 오픈되면 read()나 write()를 사용할 수 있다. RandomAccessFile은 파일이 처음으로 생성되면 파일에서 현재 위치를 나타내는 파일 포인터는 0으로 설정된다. read()나 write() 메소드가 호출되면 읽혀지거나 쓰여진 바이트만큼 파일 포인터가 변경된다.

모드	설명
r	읽기 전용 파일. 쓰기 작업을 하면 IOException 발생
rw	읽기, 쓰기 기능 파일. 파일이 존재하지 않으면 생성.

임의 접근 파일

메소드	설명
int skipBytes(int)	지정된 바이트만큼 파일 포인터를 앞쪽으로 이동한다.
void seek(long)	지정된 바이트 위치로 파일 포인터를 설정한다
long getFilePointer()	파일 포인터의 현재 위치를 반환한다.


Thank You