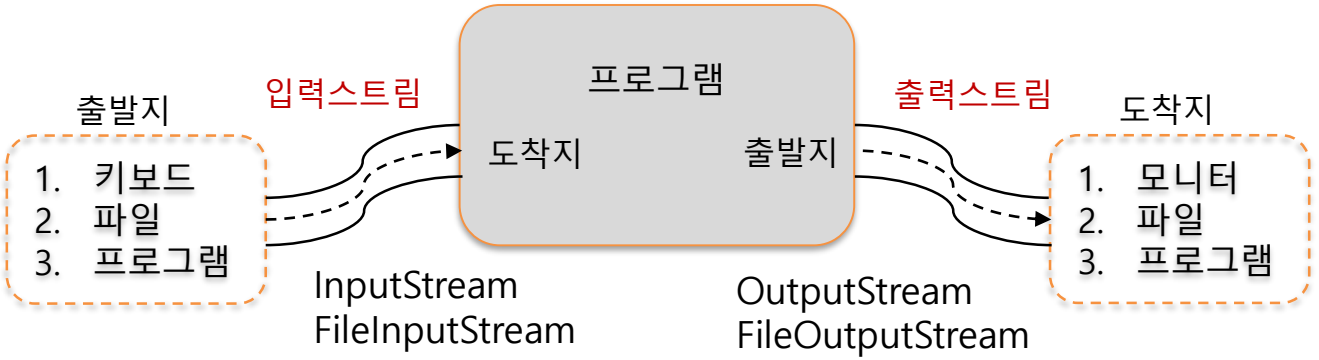
12장. 파일 입출력

file IO

입, 출력 스트림

- **스트림**이란? 자료흐름이 물의 흐름과 같다는 뜻이다. 입출력 장치는 매우 다양하기 때문에 프로그램 호환성이 떨어짐
- 입출력 장치와 무관하고 일관성 있게 프로그램을 구현할 수 있도록 일종의 가상통로인 스트림을 제공
- 자료를 읽어 들이려는 소스(source)와 자료를 쓰려는 대상(target)에 따라 각각 다른 스트림 클래스를 제공
  - ✓ 입력 스트림 – 어떤 동영상을 재생하기 위해 동영상 파일에서 자료를 읽을때 사용함
  - ✓ 출력 스트림 – 편집 화면에 사용자가 쓴 글을 파일에 저장할 때는 출력 스트림 사용함

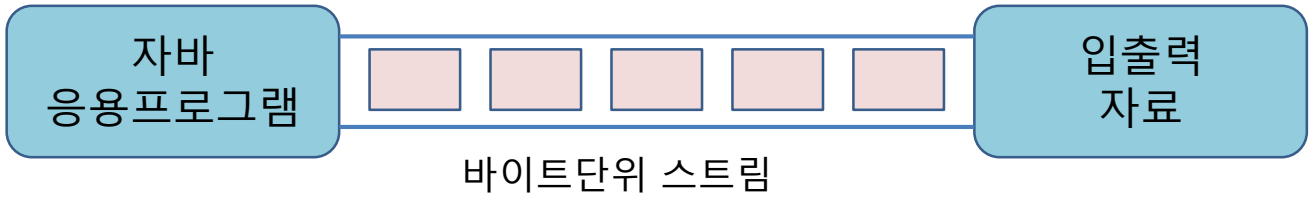


입, 출력 스트림

바이트 단위 스트림과 문자 단위 스트림

- 바이트 단위 스트림 - 그림, 동영상, 음악 파일등 대부분 파일은 바이트 단위로 읽거나 쓴다.
- 문자 단위 스트림 – 문자만 받고 보낼 수 있도록 특화되어 있다.

구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스	FileInputStream	FileOutputStream	FileReader	FileWriter



바이트단위 스트림

문자 단위 스트림

표준 입출력

표준 입출력

- **System 클래스**는 3개의 변수를 가지고 있는데, System.out은 표준 출력용, System.in은 표준 입력용 , 빨간색으로 오류를 표시할 때는 System.err을 사용한다.
- out, in, err 모두 정적 변수이다.
- 그 외 java.util 패키지에 있는 **Scanner 클래스** – 문자, 정수, 실수 등을 읽을 수 있다.

자료형	변수 이름	설명
static PrintStream	out	표준 출력 스트림
static InputStream	in	표준 입력 스트림
static OutputStream	err	표준 오류 출력 스트림

표준 입출력

- **System.in으로 화면에서 문자 1개 입력 받기**

```
public class SystemInTest1 {

    public static void main(String[] args) {

        System.out.println("한 문자를 입력하고 [Enter]를 누르세요");

        try {
            int data = System.in.read();
            System.out.println((char)data);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
한 문자를 입력하고 [Enter]를 누르세요
b
b
```

표준 입출력

- System.in으로 화면에서 문자 여러개 입력 받기

```
public class SystemInTest2 {
    public static void main(String[] args) {

        System.out.println("여러 개의 문자를 입력하고 [Enter]를 누르세요.");

        try {
            int data;
            while((data = System.in.read()) != -1) {
                System.out.print((char)data);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        /*while(true) {
            try {
                int data = System.in.read();
                if(data == -1) break;
                System.out.print((char)data);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }*/
    }
}
```

```
여러 개의 문자를 입력하고 [Enter]를 누르세요.
java
java
```

문자 단위 스트림 – Writer

- **Writer 클래스**
  - 주요 하위 클래스

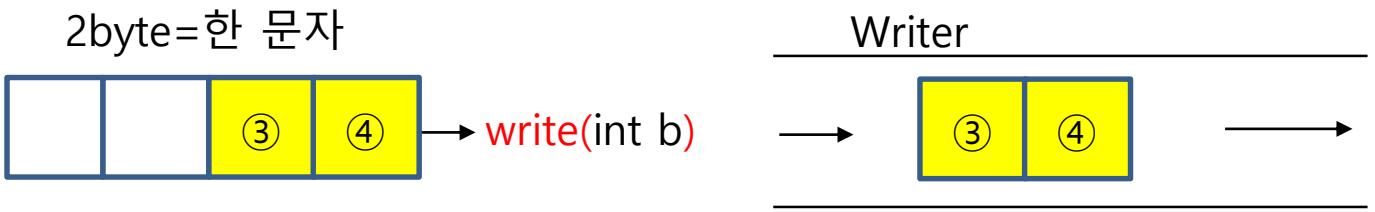
스트림 클래스	설명
FileWriter	문자 단위로 파일에 자료를 씁니다.
OutputStreamWriter	파일에 바이트 단위로 출력한 자료를 문자로 변환해 주는 보조 스트림이다.
BufferedWriter	문자로 쓸 때 배열을 제공하여 한꺼번에 쓸 수 있는 기능을 제공해 주는 보조 스트림이다.

- 주요 메소드

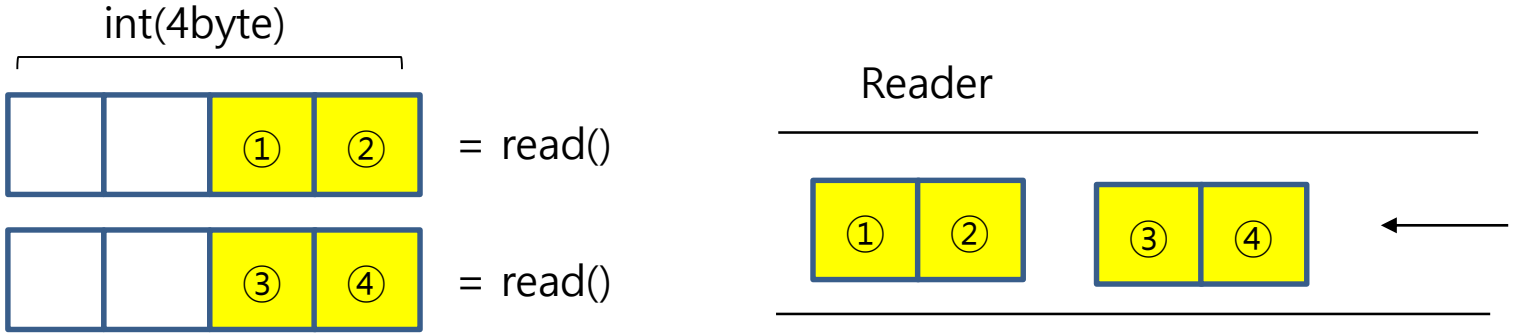
리턴타입	메소드	설명
void	write(int c)	한 문자를 파일에 출력한다.
void	write(char[] b)	문자 배열 buf의 내용을 파일에 출력한다.
void	flush()	파일에 출력하기 전에 자료가 있는 공간(출력버퍼)을 비워 출력
void	close()	출력 스트림과 연결된 대상 리소스를 닫는다.

문자 단위 스트림 – Writer

- **FileWriter 클래스로 문자 쓰기**



- **FileReader 클래스로 읽기**



문자 단위 스트림 – Writer

- **FileWriter 클래스로 문자 쓰기**

✓ **파일을 이용한 입출력 과정**

1. 파일을 연다(open) -> 파일 경로에 파일 생성
2. 파일을 쓴다. -> 문자(열)를 쓴다(저장)
3. 파일을 닫는다. -> writer.close()

문자 단위 스트림 – Writer

- 문자 쓰기

```
public class FileWriterTest {
    public static void main(String[] args) {

        try {
            //파일 열기(file 디렉터리는 미리 생성하고, test.txt는 없으면 생성됨)
            Writer writer = new FileWriter("c:/file/text.txt");

            //한 개의 문자 쓰기
            writer.write('A');
            writer.write('b');

            //숫자 쓰기 - 아스키 코드로 저장
            writer.write(49);

            //배열을 활용하여 문자 쓰기
            char[] arr = {'C', 'D', 'E'};
            writer.write(arr);
```

문자 단위 스트림 – Writer

- 문자 쓰기

```
            writer.write('\n');   //줄바꿈

            //문자열 쓰기
            writer.write("apple");
            writer.write("좋아요");

            writer.flush();   //버퍼의 데이터 강제 쓰기

            //파일 닫기
            writer.close();

            System.out.println("파일 쓰기 완료!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
Ab1CDE
apple좋아요
```

문자 단위 스트림 – Reader

- **Reader 클래스**
  - 주요 하위 클래스

스트림 클래스	설명
FileReader	파일에서 문자 단위(2바이트)로 읽는 스트림 클래스이다.
InputStreamReader	바이트 단위로 읽은 자료를 문자로 변환해 주는 보조 스트림
BufferedReader	문자로 읽을 때 배열을 제공하여 한꺼번에 읽을 수 있는 기능을 제공해 주는 보조 스트림이다.

- 주요 메소드

리턴타입	메소드	설명
int	read()	파일로부터 한 문자를 읽는다.
int	read(char[] buf)	파일로부터 buf 배열에 문자를 읽는다.
void	close()	스트림과 연결된 파일 리소스를 닫는다.

문자 단위 스트림 – Reader

- **FileReader 클래스로 문자 쓰기**

✓ **파일을 이용한 입출력 과정**

1. 파일을 연다(open) -> 파일 경로에 파일을 연다
2. 파일을 읽는다. -> 문자(열)을 읽는다.
3. 파일을 닫는다. -> reader.close()

문자 단위 스트림 – Reader

- FileReader 클래스로 읽기

```
public class FileReaderTest {
    public static void main(String[] args) {

        Reader reader = null;
        try {
            //문자로 읽기
            reader = new FileReader("c:/file/text.txt");
            while(true) {
                int data = reader.read();
                if(data == -1) break;
                System.out.print((char)data);
            }
            reader.close();
            System.out.println();
```

문자 단위 스트림 – Reader

- 파일에서 읽기

```
            //문자 배열로 읽기
            reader = new FileReader("c:/file/text.txt");
            char[] data = new char[100];
            while(true) {
                int num = reader.read(data);
                if(num == -1) break;
                for(int i = 0; i < num; i++) {
                    System.out.print(data[i]);
                }
            }
            reader.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
Ab1CDE
apple좋아요
Ab1CDE
apple좋아요
```

문자 단위 스트림 – FileWriter

- 파일에 추가로 쓰기

```
public class FileWriterTest2 {
    public static void main(String[] args) {

        try {
            // 데이터 추가(append)로 쓰기 - true 모드
            Writer writer = new FileWriter("c:/file/text.txt", true);

            writer.write("\n행운을 빌어요!!");

            writer.close();

            System.out.println("쓰기 완료!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
Ab1CDE
apple좋아요
행운을 빌어요!!
```

File 클래스

- File 클래스

  디렉토리 생성, **파일 생성**, **파일 정보 조회**, **파일 삭제** 등의 기본적인 파일 시스템 작업을 수행하는 클래스이다

```java
public class FileTest {

    public static void main(String[] args) throws IOException {

        //파일 쓰기 전 디렉터리 존재 유무 확인
        File dir = new File("c:/file2");
        if (!dir.exists())
            dir.mkdirs();    //디렉터리 생성

        File file = new File("C:/file2/new_file.txt");
        file.createNewFile();

        System.out.println(file.isFile());
        System.out.println(file.isDirectory());
        System.out.println(file.getName());
        System.out.println(file.getPath());

        //file.delete();  //파일 삭제
    }
}
```

```
true
false
new_file.txt
C:\file2\new_file.txt
```

바이트 단위 스트림 – OutputStream

➢ OutputStream 클래스

- 주요 하위 클래스

스트림 클래스	설명
FileOutputStream	바이트 단위로 파일에 자료를 씁니다.
BufferedOutputStream	기반 스트림에서 자료를 쓸때 추가 기능을 제공하는 보조 스트림의 상위 클래스이다.

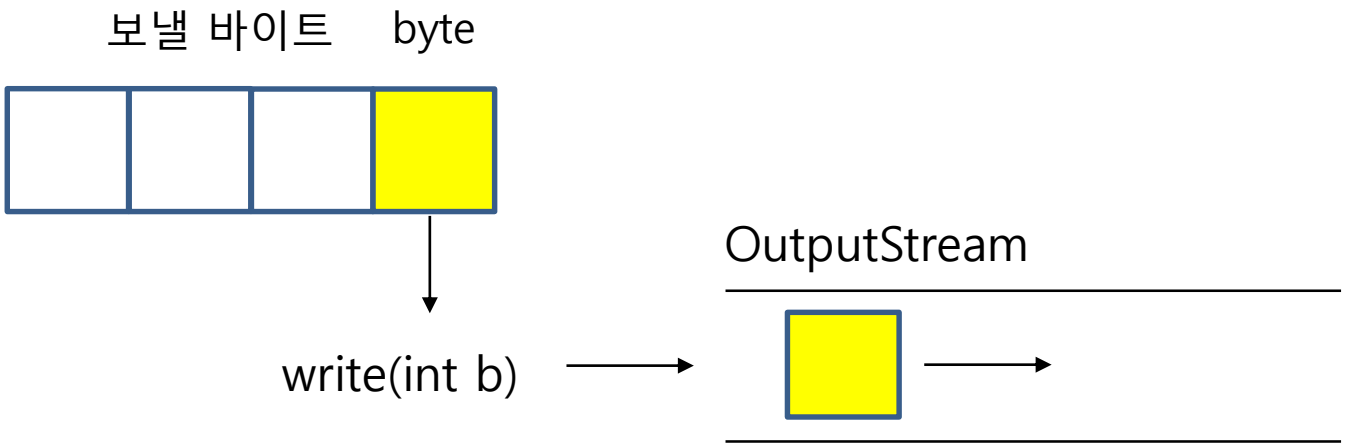
- 주요 메소드

리턴타입	메소드	설명
void	write(int b)	한 바이트를 출력한다.
void	write(byte[ ] b)	b[ ] 배열에 있는 자료를 출력한다.
void	write(byte b[ ], int off, int len)	b[ ] 배열에 off 위치부터 len개수 만큼 출력
void	close()	출력 스트림과 연결된 대상 리소스를 닫는다.

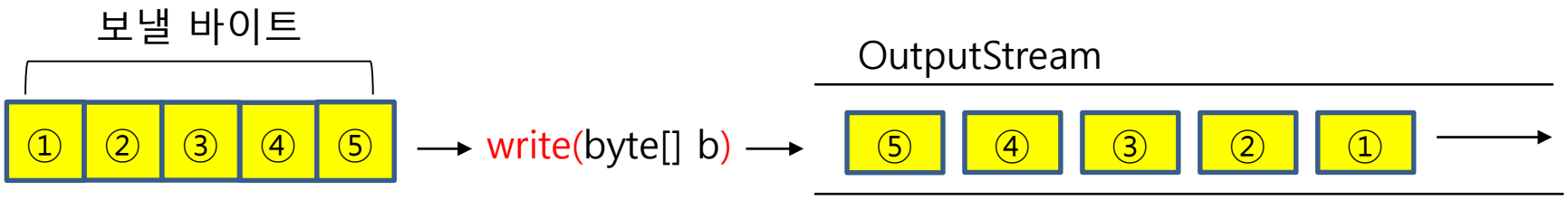
FileOutputStream

- **파일에 바이트 자료 쓰기**

write(int b) 메서드는 매개값 int(4byte)에서 끝 1byte만 출력한다.



- **write(byte[] b) – 파일에 바이트 배열로 쓰기**



FileOutputStream

- 파일에 바이트 자료 쓰기

```
public class OutputStreamTest1 {
    public static void main(String[] args) {
        try {
            //바이트 출력 스트림 생성
            OutputStream os = new FileOutputStream("C:/file/file1.bin");
            byte a = 10;
            byte b = 20;

            os.write(a); // 바이트 값 10 (0x0A) 기록
            os.write(b); // 바이트 값 20 (0x14) 기록

            //배열로 쓰기
            byte[] array = {65, 66, 67};
            os.write(array);
            //os.write(new byte[]{65, 66, 67});  // "ABC" (ASCII 코드)

            os.flush();
            os.close();
            System.out.println("쓰기 완료!!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

FileOutputStream

- 파일에 바이트 자료 쓰기

```
public class OutputStreamTest2 {

    public static void main(String[] args) {
        //try ~ with ~ resource문 : close() 하지 않아도 됨
        try(OutputStream os = new FileOutputStream("c:/file/file2.bin")){
            //특정 데이터 보내기
            byte[] array = {48, 49, 50, 51, 52};

            os.write(array, 1, 3); //1번 인덱스 부터 3개 쓰기

            //문자열로 변환하여 쓰기
            String text = "\nHava a nice day!";
            byte[] textBytes = text.getBytes();
            os.write(textBytes);

            os.flush(); //버퍼에 강제 쓰기
            System.out.println("쓰기 완료!");
        }catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

바이트 단위 스트림 – InputStream

➢ InputStream

- 주요 하위 클래스

스트림 클래스	설명
FileInputStream	파일에서 바이트 단위로 자료를 읽는다.
BufferedInputStream	기반 스트림에서 자료를 읽을때 추가 기능을 제공하는 보조 스트림의 상위 클래스이다.

- 주요 메소드

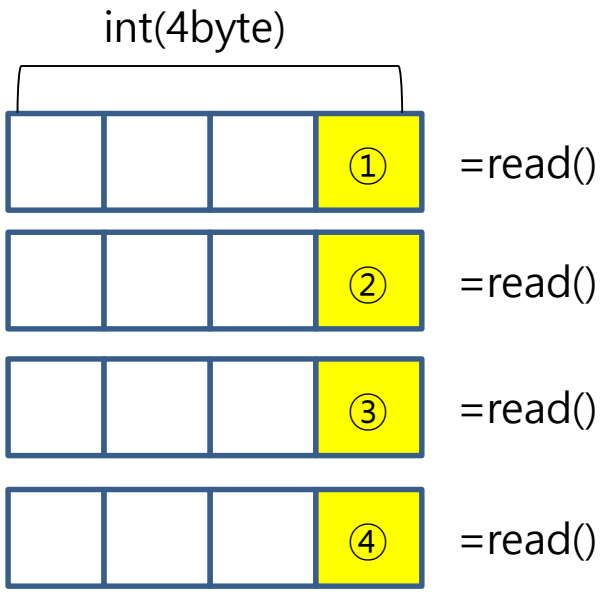
리턴타입	메소드	설명
int	read()	입력 스트림으로부터 1바이트를 읽고 읽은 바이트를 리턴한다.
int	read(byte[] b)	입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열 b에 저장하고 실제로 읽은 바이트 수를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

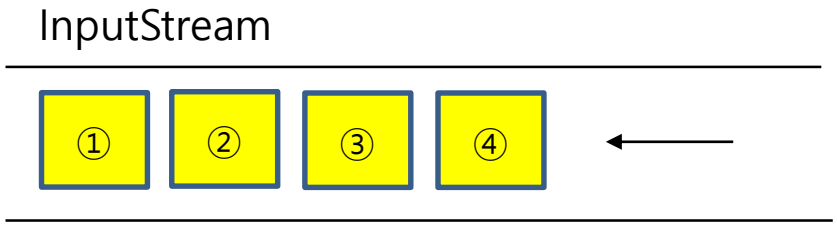
FileInputStream

➢ 파일에서 자료 읽기 – 1byte

read() 메서드는 입력 스트림으로부터 1byte를 읽고 int(4byte) 타입으로 리턴한다.

따라서 리턴된 4byte 중 끝 1byte에만 데이터가 들어있음





FileInputStream

- 파일에서 자료 읽기

```
public class InputStreamTest1 {

    public static void main(String[] args) {

        try {
            InputStream is = new FileInputStream("C:/file/file1.bin");
            int data;
            while((data = is.read()) != -1) {
                System.out.println(data);
                //System.out.println((char)data);
            }
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

FileInputStream

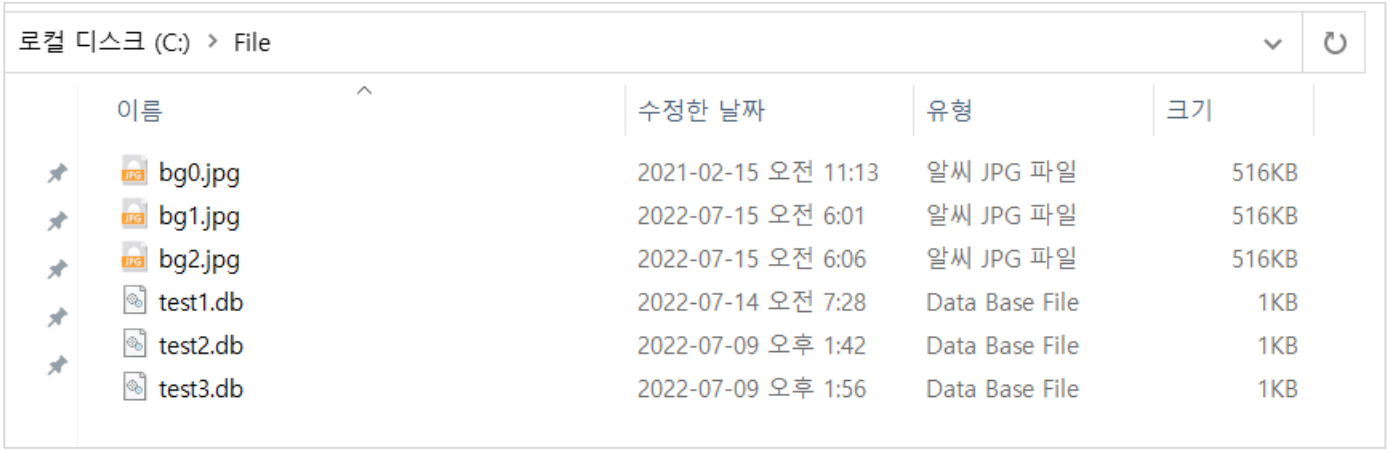
- 파일에서 자료 읽기

```
public class InputStreamTest2 {

    public static void main(String[] args) {

        try(InputStream is = new FileInputStream("C:/file/file2.bin")){
            byte[] data = new byte[100];

            while(true) {
                int num = is.read(data);
                if(num == -1) break;
                for(int i = 0; i < num; i++) {
                    System.out.print((char)data[i]);
                }
            }
        }catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
123
Hava a nice day!
```

이미지 파일 복사

➢ **FileInputStream과 FileOutputStream을 이용하여 파일 복사하기**



이미지 파일 복사

➢ **FileInputStream과 FileOutputStream을 이용하여 파일 복사하기**

```
public class FileCopyTest1 {

    public static void main(String[] args) {

        String originFile = "c:/file/img1.jpg";  //원본 파일
        String copyFile = "c:/file/img2.jpg";    //복사 파일

        try(InputStream is = new FileInputStream(originFile);
            OutputStream os = new FileOutputStream(copyFile)){

            while(true) {
                int num = is.read();  //파일 읽기
                if(num == -1) break;
                os.write(num);        //파일 쓰기
            }
            os.flush();

            System.out.println("복사 완료!");
        }catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

이미지 파일 복사

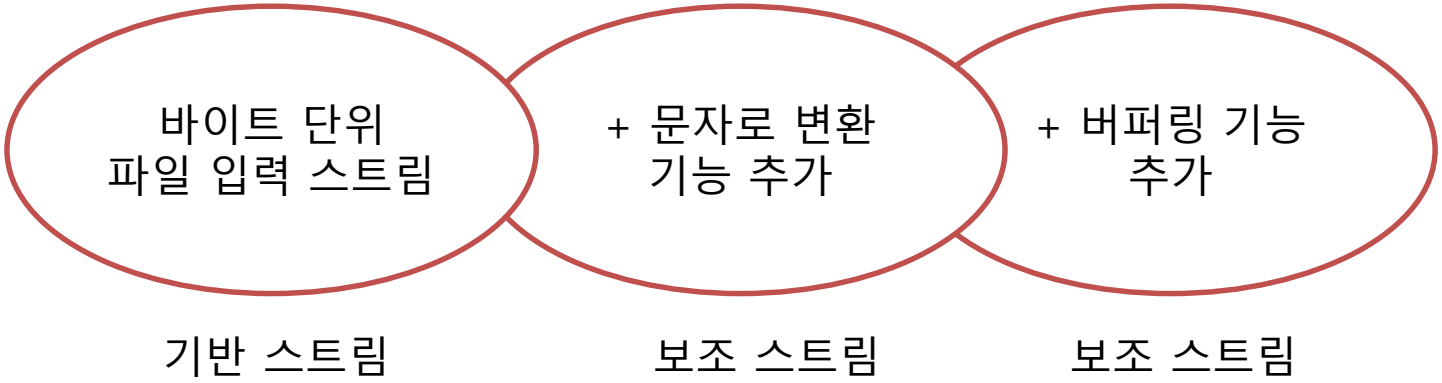
```java
public class FileCopyTest2 {

    public static void main(String[] args) {
        String originFile = "c:/file/img1.jpg";  //원본 파일
        String copyFile = "c:/file/img2.jpg";    //복사 파일
        long start, end;

        try(InputStream is = new FileInputStream(originFile);
            OutputStream os = new FileOutputStream(copyFile)){
            start = System.currentTimeMillis();

            while(true) {
                int num = is.read();  //최대 1024바이트를 읽고 배열에 저장
                if(num == -1) break;
                os.write(num);  //읽은 바이트 수만큼 출력
            }
            os.flush();
            end = System.currentTimeMillis();
            System.out.println("복사 소요 시간: " + (end-start) + "ms");
        }catch(IOException e) {
            e.printStackTrace();
        }
    }
```

```
복사 소요 시간: 1901ms
```

보조 스트림

보조 스트림

- 다른 스트림과 연결되어 여러 가지 편리한 기능을 제공해주는 스트림을 말한다.
- 자체적으로 입출력을 수행할 수 없기 때문에 입출력 소스로부터 직접 생성된 입출력 스트림에 연결해서 사용함
- 생성자의 매개변수로 **기반스트림**을 가짐



보조 스트림 – InputStreamReader

InputStreamReader와 OutputStreamWriter

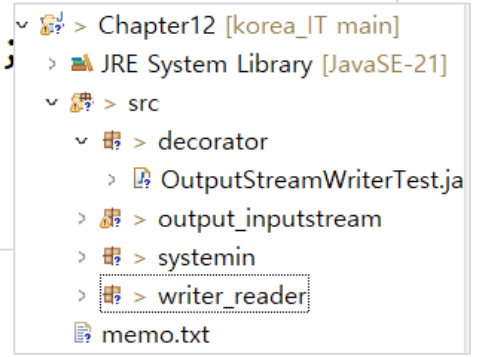
- 바이트 자료만 입력되는 스트림에서 문자로 변환해 준다.
- **System.in** 이나 네트워크 **socket** 통신을 할때 쓰인다.
- 입출력 기능이 없으므로 다른 입출력 스트림을 포함한다.

생성자	설명
InputStreamReader (FileInputStream in)	생성자의 매개변수로 FileInputStream을 받는다.
OutputStreamWriter (FileOutputStream out)	생성자의 매개변수로 FileOutputStream을 받는다.

보조 스트림 – InputStreamReader

- **OutputStreamWriter & InputStreamReader**
  - ✓ 바이트 자료만 출력되는 스트림에서 문자로 변환해 파일을 생성하고 읽기
  - ✓ 파일 쓰기 후 프로젝트 우클릭 -> Refresh(F5) -> memo.txt 표시

```
public class OutputStreamWriterTest {

    public static void main(String[] args) {
        //보조 스트림은 기반 스트림을 생성자로 함
        //문자 쓰기
        try(OutputStreamWriter osw =
                new OutputStreamWriter(new FileOutputStream("memo.txt"))){
            osw.write("오늘도 좋은 하루 되세요~");
            System.out.println("test.txt 파일을 열어 보세요~");
        }catch(IOException e) {
            System.out.println(e);
        }
```



보조 스트림 – InputStreamReader

- **OutputStreamWriter & InputStreamReader**

  바이트 자료만 출력되는 스트림에서 문자로 변환해 파일을 생성하고 읽기

```
        //문자 읽기
        try(InputStreamReader isr =
                new InputStreamReader(new FileInputStream("memo.txt"))){
            int data;
            while((data = isr.read()) != -1) {
                System.out.print((char)data);
            }
        }catch(IOException e) {
            System.out.println(e);
        }
    }
}
```

보조 스트림 – Buffered 스트림

Buffered 스트림

- 입출력이 한 바이트나 문자 단위로 이루어지면 그만큼 프로그램 수행 속도가 느려진다.
- Buffered 스트림은 내부적으로 8,192바이트 크기의 배열을 가지고 있으며 더 빠르게 입출력을 수행하는 버퍼링 기능을 제공한다.

스트림 클래스	설명
BufferedInputStream (InputStream in)	바이트 단위로 읽는 스트림에 버퍼링 기능을 제공
BufferedOutputStream (FileOutputStream out)	바이트 단위로 출력하는 스트림에 버퍼링 기능을 제공
BufferedReader	문자 단위로 읽는 스트림에 버퍼링 기능을 제공
BufferedWriter	문자 단위로 출력하는 스트림에 버퍼링 기능을 제공

보조 스트림 – Buffered 스트림

- **BufferedInputStream과 BufferedOutputStream으로 파일 복사하기**

```
public class BufferedStreamTest {

    public static void main(String[] args) {
        long start = 0;
        long end = 0;
        String originFile = "C:/javaDev/day12/feature.png";
        String copyFile = "C:/ncsTest/feature3.png";

        try(FileInputStream fis = new FileInputStream(originFile);
            FileOutputStream fos = new FileOutputStream(copyFile);
            BufferedInputStream bis = new BufferedInputStream(fis);
            BufferedOutputStream bos = new BufferedOutputStream(fos)){
            start = System.currentTimeMillis();
            int i;
            while((i=bis.read()) != -1) {
                bos.write(i);
            }
            end = System.currentTimeMillis();
        }catch(IOException e) {
            e.printStackTrace();
        }
        System.out.println("파일 복사 소요시간 : " + (end-start) + "milliseconds");
    }
}
```

```
파일 복사 소요시간 : 23 millseconds
```

보조 스트림 – Buffered 스트림

- **BufferedReader**
  - 문자 입력 스트림 Reader에 BuffredReader를 연결하면 행 단위로 문자열을 읽는 매우 편리한 readLine() 메소드를 제공한다.
  - readLine()은 enter 키 이전의 모든 문자열을 읽고 리턴한다.

readLine

```
public String readLine() throws IOException
```

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), a carriage return followed immediately by a line feed, or by reaching the end-of-file (EOF).

```
오늘도 좋은 하루되세요!!
감사합니다.
여름 장마비가 옵니다.
```

보조 스트림 – Buffered 스트림

```
public class WriteExample2 {
    public static void main(String[] args) {
        try {
            Writer writer = new FileWriter("message.txt");

            //문자열 출력
            String message = "오늘도 좋은 하루 되세요!\n행운을 빌어요!\n감사합니다.";
            writer.write(message);

            writer.flush();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

보조 스트림 – Buffered 스트림

```
public class ReadLineTest {

    public static void main(String[] args) {
        //FileReader에 BufferedReader 보조 스트림 연결
        try {
            BufferedReader br = new BufferedReader(
                    new FileReader("message.txt"));
            /*while(true) {
                String data = br.readLine(); //1행을 읽음
                if(data == null) break;
                System.out.println(data);
            }*/
            String data;
            while((data = br.readLine()) != null) { //읽을 행이 없을때까지 반복
                System.out.println(data);
            }
            br.close();
        }catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

보조 스트림 – BufferedReader

split()를 함수를 사용하여 문자열을 배열로 만들어 랜덤하게 출력하기

```
public class ReadLineTest3 {
    public static void main(String[] args) {
        //FileReader에 BufferedReader 보조 스트림 연결
        try(BufferedReader br = new BufferedReader(
                new FileReader("word.txt"))) {
            String data;
            String[] word = null;
            while((data = br.readLine()) != null) { //읽을 행이 없을때까지 반복
                //System.out.println(data);
                word = data.split(" ");  //공백 문자를 구분기호로 배열 요소화 됨
            }
            System.out.println(Arrays.toString(word)); //배열 객체 출력
            System.out.println(word[0]); //0번 인덱스 객체

            //랜덤 출력
            int rand = (int)(Math.random()*word.length);
            System.out.println(rand);
            System.out.println(word[rand]);
        }catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
[ant, bear, cow, chicken, eagle, elephant, fox, horse, monkey, penguin, tiger]
ant
9
penguin
```

보조 스트림 – BufferedReader

- 영어 타자 연습 게임

```java
public class EnglishTypingGame {

    public static void main(String[] args) {
        /*
           영어 단어 게임 - 총 10 문제 출제
           단어는 외부 파일을 읽어옴
           게임 소요 시간 측정
         */
        Scanner sc = new Scanner(System.in);
        try(FileReader fr = new FileReader("word.txt");
            BufferedReader br = new BufferedReader(fr)){

            String[] word = null;
            String data = null;
            while(true) {
                data = br.readLine();
                if(data == null) break;
                word = data.split(" ");
            }
            //System.out.println(word[0]); //ant 확인
```

word.txt

```
1 ant bear chicken cow cat dog dove horse monkey penguin
```

```
타자 연습 게임 - 준비되면 [Enter]

문제 1
horse
horse
통과!!
문제 2
penguin
penguin
통과!!
문제 3
dog
```

보조 스트림 – BufferedReader

```
        int n = 1;  //문제 번호
        long start = 0, end = 0;
        System.out.println("타자 연습 게임 - 준비되면 [Enter]");
        sc.nextLine();
        start = System.currentTimeMillis();
        while(n <= 10) {
            System.out.println("문제 " + n);
            int rand = (int)(Math.random()*word.length);
            String question = word[rand];
            System.out.println(question);  //문제 표시

            String answer = sc.nextLine(); //답변 입력

            if(answer.equals(question)) {
                System.out.println("통과!!");
                n++;   //문제1 증가
            }else {
                System.out.println("오타! 다시 도전!");
            }
        }
        end = System.currentTimeMillis();
        //float형 변환 - 소수까지 출력
        System.out.println("게임 소요시간 " + (float)(end-start)/1000 + "초");
        sc.close();
    }catch(IOException e) {
        e.printStackTrace();
    }
}
```

보조스트림 – DataStream

- **DataInputStream과 DataOutputStream**

  지금까지의 공부한 스트림은 사람이 읽고 쓰는 텍스트 또는 이미지 형식의 자료를 다루었으나, DataStream은 메모리에 저장된 0, 1상태를 그대로 읽거나 쓴다.
  즉, 자료형의 크기가 그대로 보존된다.

DataInputStream

메서드	설명
byte readByte()	1바이트를 읽어 반환
char readChar()	한 문자를 읽어 반환
int readInt()	4바이트를 읽어 정수값 반환
double readDouble()	8바이트를 읽어 실수값 반환
String readUTF()	문자열을 읽어 반환

DataOutputStream

메서드	설명
void writeByte(int v)	1바이트의 자료 쓰기
void writeChar(int v)	2바이트의 자료 쓰기
void writeInt(int v)	4바이트의 자료 쓰기
void writeDouble(double v)	8바이트의 자료 쓰기
void readUTF(String str)	문자열을 쓰기

DataStream

```
public class DataInputOutputStreamTest {
    public static void main(String[] args) {
        try {
            //DataInputStream 클래스의 dis 객체 생성
            //fos 객체를 기반으로 DataOutputStream dos 객체 생성
            FileOutputStream fos = new FileOutputStream("data.db");
            DataOutputStream dos = new DataOutputStream(fos);

            //기본 타입 출력
            dos.writeInt(1);
            dos.writeUTF("우영우");
            dos.writeDouble(95.5);

            dos.writeInt(2);
            dos.writeUTF("장그래");
            dos.writeDouble(85.5);

            dos.flush();
            dos.close();
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

DataStream

```
    try {
        //DataInputStream 클래스의 fis 객체 생성
        //fis 객체를 기반으로 DataInputStream dis 객체 생성
        FileInputStream fis = new FileInputStream("data.db");
        DataInputStream dis = new DataInputStream(fis);

        for(int i=0; i<2; i++) { //데이터 수만큼 반복
            //기본 타입 읽기
            int order = dis.readInt();
            String name = dis.readUTF();
            double score = dis.readDouble();
            System.out.println(order + " : " + name + " : " + score);
        }

        dis.close(); fis.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
1 : 우영우 : 95.5
2 : 장그래 : 85.5
```

직렬화(Serialization)

◆ Serialization(직렬화) – 객체 스트림

- 인스턴스의 어느 순간 상태를 그대로 저장하거나, 네트워크를 통해 전송하기 위해 연속 스트림(필드값을 일렬로 늘어선 바이트로 변경)으로 만드는 것을 직렬화라 한다.
- 역직렬화는 저장된 내용이나 전송받은 내용을 다시 복원하는 것이다.
- 보조 스트림인 **ObjectInputStream**과 **ObjectOutputStream** 사용한다.
- 주요 메서드로는 writeObject()와 readObject()가 있다.
- serialVersinUID를 사용하여 버전 관리 (객체를 역직렬화할때 직렬화할때의 클래스 상태가 다르면 오류가 발생.)

생성자	설명
ObjectInputStream(InputStream in)	InputStream을 생성자의 매개변수로 받아 ObjectInputStream을 생성합니다.
ObjectOutputStream(OutputStream out)	OutputStream을 생성자의 매개변수로 받아 ObjectOutputStream을 생성합니다.

직렬화(Serialization)

◆ 객체의 역직렬화

```
public class ObjectInputOutputStreamTest {
    public static void main(String[] args) {
        //객체를 역직렬화해서 파일에 쓰기
        try(FileOutputStream fos = new FileOutputStream("object.dat");
            ObjectOutputStream oos = new ObjectOutputStream(fos)){
            //객체 생성
            Member m1 = new Member("sky123", "김하늘");
            Product p1 = new Product("스마트폰", 1200000);
            int[] number = {1, 2, 3, 4};

            //객체를 역직렬화해서 파일에 저장
            oos.writeObject(m1);
            oos.writeObject(p1);
            oos.writeObject(number);

            oos.flush();
        }catch(Exception e) {
            e.printStackTrace();
        }
```

직렬화(Serialization)

```
    //객체를 역직렬화해서 파일에서 읽기
    try(FileInputStream fis = new FileInputStream("object.dat");
            ObjectInputStream ois = new ObjectInputStream(fis)){

        //파일을 읽고 역직렬화해서 객체로 복원
        Member m2 = (Member)ois.readObject();
        Product p2 = (Product)ois.readObject();
        int[] number2 = (int[])ois.readObject();

        //복원된 객체 내용 확인
        System.out.println(m2);
        System.out.println(p2);
        System.out.println(Arrays.toString(number2));
    }catch(Exception e) {
        e.printStackTrace();
    }
}
```

```
sky123, 김하늘
스마트폰, 1200000
[1, 2, 3, 4]
```

직렬화(Serialization)

◆ 객체의 직렬화

```
public class Member implements Serializable{

    private static final long serialVersionUID = 1234L;

    private String id;
    private String name;

    public Member(String id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return id + ", " + name;
    }
}
```

직렬화(Serialization)

◆ 객체의 직렬화

```
public class Product implements Serializable{

    private static final long serialVersionUID = 1002L;

    private String name;
    private int price;

    public Product(String name, int price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return name + ", " + price;
    }
}
```

직렬화(Serialization)

◆ Serialization 예제

```
package iostream.serialization;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

class Person implements Serializable{
    private static final long serialVersionUID = 12345L;

    String name;
    String job;

    public Person() {}

    public Person(String name, String job) {
        this.name = name;
        this.job = job;
    }

    public String toString() {
        return name + "," + job;
    }
}
```

Serializable 인터페이스 구현

```
java.io.NotSerializableException: iostream.serialization.Person
        at java.base/java.io.ObjectOutputStream.writeObject0(Ob
        at java.base/java.io.ObjectOutputStream.writeObject(Obje
```

직렬화(Serialization)

```java
public class SerializationTest {
    public static void main(String[] args) {
        //직렬화
        Person personSon = new Person("손정의", "대표이사");
        Person personJang = new Person("장그래", "부장");

        try(FileOutputStream fos = new FileOutputStream("serial.out");
            ObjectOutputStream oos = new ObjectOutputStream(fos)){
            oos.writeObject(personSon);  //객체를 파일에 씀
            oos.writeObject(personJang);
        }catch(Exception e) {
            e.printStackTrace();
        }

        //복원 - 역직렬화
        try(FileInputStream fis = new FileInputStream("serial.out");
            ObjectInputStream ois = new ObjectInputStream(fis)){
            Person p1 = (Person) ois.readObject(); //Object형 -> Person 변환
            Person p2 = (Person) ois.readObject();

            System.out.println(p1);
            System.out.println(p2);
        }catch(Exception e) {
            e.printStackTrace();
```

```
ы ☐sr iostream.serialization.Person
  09☐ ☐L ☐jobt ☐Ljava/lang/String;L
```

```
손정의,대표이사
장그래,부장
```