Java

입출력 스트림

- 데이터는 키보드를 통해 입력될 수도 있고, 파일 또는 타 프로그램으로부터 입력될 수도 있다.
- 반대로 모니터로 출력될 수도 있고, 파일에 저장되거나 타 프로그램으로 전송될 수 있다.
- 이것들을 총칭해서 데이터 입출력이라고 한다.

- Java는 입력 스트림과 출력 스트림을 통해 데이터를 입력하고 출력한다.
- 스트림(Stream)은 단방향으로 흐르는 것을 의미한다.

- 프로그램을 기준으로 데이터가 들어오면 입력 스트림, 데이터가 나가면 출력 스트림이 된다.
- 프로그램에서 다른 프로그램과 데이터를 교환하기 위해서는 양쪽 모두 입력 스트림과 출력 스트림이 필요하다.

입출력 스트림

- 어떤 데이터를 입출력하느냐에 따라서 스트림은 두 종류로 구분할 수 있다.
  - 바이트 스트림 : 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 입출력할 때 사용
  - 문자 스트림 : 문자만 입출력할 때 사용

- Java에서는 데이터 입출력과 관련된 라이브러리를 java.io 패키지에서 제공하고 있다.
- java.io 패키지는 아래와 같이 바이트 스트림과 문자 스트림을 구분해서 제공한다.

구분	바이트 스트림		문자 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	`InputStream`	`OutputStream`	`Reader`	`Writer`
하위 클래스	`XXXInputStream`	`XXXOutputStream`	`XXXReader`	`XXXWriter`

TIP

최상위 클래스를 상속 받는 자식 클래스는 접미사가 따라 붙는다.
예를 들어, 바이너리 파일의 입출력 스트림 클래스는 FileInputStream, FileOutputStream이다.

입출력 스트림 - 바이트 스트림 (출력)

- OutputStream은 바이트 출력 스트림의 최상위 클래스인데, 추상 클래스이다.
- 모든 바이트 출력 스트림 클래스는 OutputStream 클래스를 상속받아 만들어진다.
  - (FileOutputStream, PrintStream, BufferedOutputStream, DataOutputStream)

- OutputStream 클래스에는 모든 바이트 출력 스트림이 기본적으로 가져야 할 메소드가 정의되어 있다.

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

입출력 스트림 - 바이트 스트림 (출력)

```
public class WriteExample {
    public static void main(String[] args) {
        try {
            OutputStream os = new FileOutputStream("C:/Users/inkyu/output.txt");

            byte a = 65; byte b = 66; byte c = 67;
            os.write(a); os.write(b); os.write(c);

            byte[] arr = { a, b, c };
            os.write(arr);
            os.write(arr, 1, 1);

            os.flush(); // 내부 버퍼에서 잔류하는 모든 바이트를 출력하고 버퍼를 비움
            os.close(); // 출력 스트림이 사용한 메모리 해제
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

입출력 스트림 - 바이트 스트림 (입력)

- InputStream은 바이트 입력 스트림의 최상위 클래스인데, 추상 클래스이다.
- 모든 바이트 입력 스트림 클래스는 InputStream 클래스를 상속받아 만들어진다.
  - (FileInputStream, BufferedInputStream, DataInputStream)

- InputStream 클래스에는 모든 바이트 출력 스트림이 기본적으로 가져야 할 메소드가 정의되어 있다.

메소드	설명
read()	1byte를 읽은 후 읽은 바이트를 반환 (더 이상 읽을 수 없다면 -1을 반환)
read(byte[] b)	읽은 바이트를 매개값으로 주어진 배열에 저장 후 읽은 바이트 수를 반환
close()	입력 스트림을 닫고 사용 메모리 해제

입출력 스트림 - 바이트 스트림 (입력)

```
public class ReadExample {
    public static void main(String[] args) {
        try {
            InputStream is1 = new FileInputStream("C:/Users/inkyu/output.txt");

            while(true) {
                int data = is1.read();
                if(data == -1) break;
                System.out.println(data);
            }

            is1.close();

            System.out.println();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

입출력 스트림 - 바이트 스트림 (입력)

```
public class ReadExample {
    public static void main(String[] args) {
        try {
            // ...
            InputStream is2 = new FileInputStream("C:/Users/inkyu/output.txt");
            byte[] data = new byte[7];

            while(true) {
                int num = is2.read(data);
                if(num == -1) break;
            }

            for(byte b: data) {
                System.out.println(b);
            }
            is2.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

입출력 스트림 - 문자 스트림 (출력)

- Writer는 문자 출력 스트림의 최상위 클래스인데, 추상 클래스이다.
- 모든 문자 출력 스트림 클래스는 Writer 클래스를 상속받아 만들어진다.
  - (FileWriter, BufferedWriter, PrintWriter, OutputStreamWriter)

- Writer 클래스에는 모든 문자 출력 스트림이 기본적으로 가져야 할 메소드가 정의되어 있다.

메소드	설명
write(int c)	매개값으로 주어진 한 문자를 출력
write(char[] cbuf)	매개값으로 주어진 배열의 모든 문자를 출력
write(char[] cbuf, int off, int len)	매개값으로 주어진 배열 cbuf[off]부터 len개의 바이트를 출력
write(String str)	매개값으로 주어진 문자열을 출력
write(String str, int off, int len)	매개값으로 주어진 문자여에서 off 순번부터 len개의 바이트를 출력
flush()	출력 버퍼에 잔류하는 모든 바이트를 출력
close()	출력 스트림을 닫고 사용 메모리 해제

입출력 스트림 - 문자 스트림 (출력)

```
public class WriteExample2 {
    public static void main(String[] args) {
        try {
            Writer w = new FileWriter("C:/Users/inkyu/output.txt");

            char a = 'A';
            w.write(a);

            char[] arr = { 'B', 'C', 'D'};
            w.write(arr);

            w.write("EFG");

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

입출력 스트림 - 문자 스트림 (입력)

- Reader는 문자 입력 스트림의 최상위 클래스인데, 추상 클래스이다.
- 모든 문자 입력 스트림 클래스는 Reader 클래스를 상속받아 만들어진다.
  - (FileReader, BufferedReader, InputStreamReader)

- Reaer 클래스에는 모든 문자 입력 스트림이 기본적으로 가져야 할 메소드가 정의되어 있다.

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

입출력 스트림 - 문자 스트림 (입력)

```
public class ReadExample2 {
    public static void main(String[] args) {
        try {
            Reader r1 = new FileReader("C:/Users/Administrator/output.txt");

            while(true) {
                int data = r1.read();
                if(data == -1) break;
                System.out.println((char) data);
            }

            r1.close();

            System.out.println();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

입출력 스트림 - 문자 스트림 (입력)

```
public class ReadExample2 {
    public static void main(String[] args) {
        try {
            // ...
            Reader r2 = new FileReader("C:/Users/Administrator/output.txt");
            char[] data = new char[7];

            while(true) {
                int num = r2.read(data);
                if(num == -1) break;
            }

            for(char c: data) {
                System.out.println(c);
            }
            r2.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

입출력 스트림 - 보조 스트림

- 보조 스트림이란 다른 스트림과 연결되어 여러 가지 편리한 기능을 제공해주는 스트림을 말한다.
- 보조 스트림은 자체적으로 입출력을 수행할 수 없기 때문에 입출력 소스로부터 직접 생성된 입출력 스트림에 연결해서 사용해야 한다.

- 입출력 스트림에 보조 스트림을 연결하기 위해서는 보조 스트림 생성 시, 생성자 매개값으로 입출력 스트림을 제공하면 된다.

- 자주 사용되는 보조 스트림은 아래와 같다.

보조 스트림	기능
InputStreamReader	바이트 스트림을 문자 스트림을 변환
BufferedInputStream, BufferedOutputStream BufferedReader, BufferedWriter	입출력 성능 향상
DataInputStream, DataOutputStream	기본 타입 데이터 입출력
PrintStream, PrintWriter	줄바꿈 처리 및 형식화된 문자열 출력
ObjectInputStream, ObjectOutputStream	객체 입출력

입출력 스트림 - 성능 향상 스트림 (버퍼)

- CPU와 메모리가 아무리 뛰어나도 하드디스크의 입출력이 늦어지면, 프로그램의 실행 성능은 하드디스크 처리 속도에 맞춰진다.
- 네트워크로 데이터를 전송할 때도 느린 네트워크 환경이라면 컴퓨터 사양이 아무리 좋아도 메신저와 게임 속도는 느릴 수 밖에 없다.

- 프로그램이 입출력 소스와 직접 작업하지 않고 중간에 메모리 버퍼(Buffer)와 작업함으로써 실행 성능을 향상시킬 수 있다.

- 버퍼는 데이터가 쌓이기를 기다렸다가 꽉 차게 되면, 데이터를 한꺼번에 하드 디스크로 보낸다. 이로써 출력 횟수를 줄여준다.

```
BufferedInputStream bis = new BufferedInputStream(바이트입력스트림);
BufferedOutputStream bos = new BufferedOutputStream(바이트출력스트림);

BufferedReader br = new BufferedReader(문자입력스트림);
BufferedWriter bw = new BufferedWriter(문자출력스트림);
```

- 문자 입력 스트림 Reader에 BuffereReader를 연결하면 성능 향상 뿐만 아니라, readLine() 메소드도 제공한다.
  readLine() 메소드는 행 단위 문자열을 읽기 때문에 매우 편리하다.

입출력 스트림 - File과 Files 클래스

- java.io 패키지와 java.nio.file 패키지는 파일과 디렉토리 정보를 가지고 있는 File과 Files 클래스를 제공한다.
- Files는 Fiie을 개선한 클래스로, 보다 많은 기능을 가지고 있다.

- File 클래스로부터 File 객체를 생성하는 방법은 아래와 같다.

```
File file = new File("경로");
```

TIP

경로에서 사용하는 경로 구분자는 운영체제마다 다르다.
Windows : \\ 또는 /
MacOS, LINUX : \

```
// 파일 또는 디렉토리 존재 여부 반환
boolean isExist = file.exists();
```

- 만약 exists()의 값이 false이면, 새로운 파일 또는 폴더를 생성할 수 있다.

```
file.createNewFile();    // 새 파일 생성
file.mkdir();            // 새 폴더 생성
file.mkdirs();           // 경로 상 없는 모든 폴더 생성
```

입출력 스트림 - File과 Files 클래스

- 만약 exists()의 값이 true이면, 아래와 같은 메소드들을 사용할 수 있다.

```
file.delete();          // 파일 또는 폴더 삭제
file.canExecute();      // 실행 가능 여부 반환
file.canRead();         // 읽기 가능 여부 반환
file.canWrite();        // 편집 가능 여부 반환
file.getName();         // 파일명 반환
file.getParent();       // 부모 폴더 반환
file.getParentFile(); // 부모 폴더를 File 객체로 반환
file.getPath();         // 전체 경로 반환
file.isDirectory();     // 폴더 여부 반환
file.isFile();          // 파일 여부 반환
file.isHidden();        // 숨김 파일 여부 반환
file.lastModified();  // 마지막 수정일 반환
file.length();          // 파일 크기 반환
file.list();            // 폴더 내 모든 파일 및 폴더 반환
file.listFiles();       // 폴더 내 모든 파일 및 폴더를 File 객체 배열로 반환
```

입출력 스트림 - File과 Files 클래스

- Files 클래스는 정적 메소드로 구성되어 있기 때문에 File 클래스처럼 객체를 생성할 필요가 없다.
- Files의 정적 메소드는 운영체제 파일 시스템에게 파일 작업을 수행하도록 위임하여 동작한다.

메소드	설명
Files.copy(Path source, Path target)	파일 또는 디렉토리를 복사
Files.createDirectory(Path path)	새로운 디렉토리를 생성합니다.
Files.createFile(Path path)	새로운 파일을 생성합니다.
Files.delete(Path path)	파일 또는 디렉토리를 삭제합니다.
Files.exists(Path path)	파일 또는 디렉토리가 존재하는지 확인합니다.
Files.move(Path source, Path target)	파일 또는 디렉토리를 이동 또는 이름을 변경합니다.
Files.list(Path dir)	디렉토리의 파일 및 디렉토리 목록을 스트림으로 반환합니다.
Files.probeContentType(Path path)	파일의 MIME 타입을 반환합니다.
Files.size(Path path)	파일의 크기를 반환합니다.
Files.getLastModifiedTime(Path path)	파일의 마지막 수정 시간을 반환합니다.