<입출력>

1. 입출력

- 입력(Input)과 출력(Output) 처리
- java.io 패키지의 클래스들로 지원함

  단, java 5버전부터 java.util.Scanner가 추가되었음 → 값의 자료형 별 입력이 편해짐

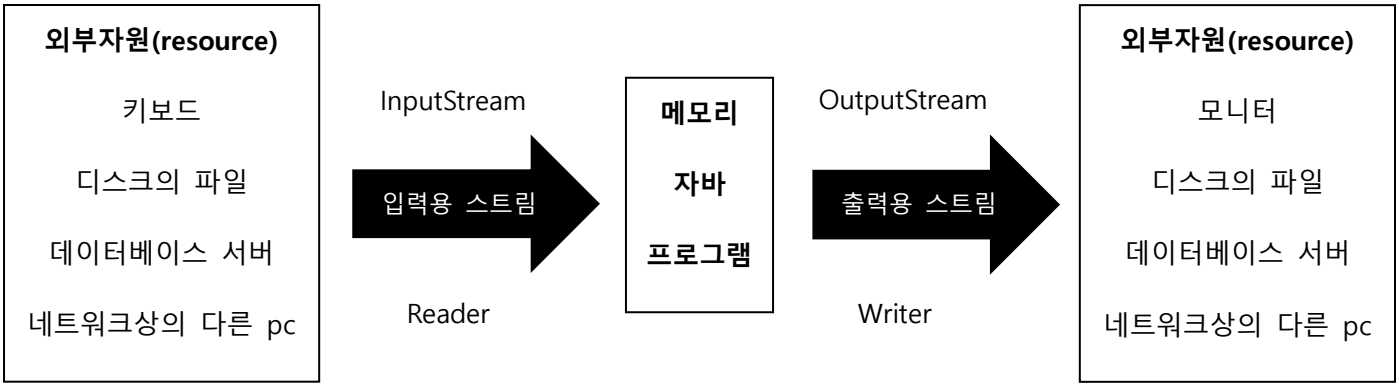
2. 스트림

- 입출력에는 stream(통로)이 사용됨
- stream은 단방향임(일방통행) → 입출력 동시에 하고 싶으면 입력용, 출력용과 같이 총 2개를 사용해야됨
- stream(통로)는 시스템(os)가 관리하고 있음

  → 외부 자원과의 입출력이 필요할 경우, 시스템으로부터 스트림을 얻어와서 사용하고

  사용이 끝나면 반드시 시스템으로 스트림을 반납해야 함(close)

3. 입출력 처리 방식



- **자바에서 입/출력을 하려면, 외부자원과 스트림부터 연결시키고, 그 다음에 읽거나 쓰기를 함**

  → 스트림 클래스에 대한 객체 생성 : 외부자원과의 통로 만들기

  → 위에서 만든 레퍼런스.읽기용메소드() 또는 레퍼런스.쓰기용메소드() 호출

  * 읽기용 메소드 : 버퍼에서 읽어내서 변수에 기록하는 메소드 공통으로 read()메소드 사용함

  * 쓰기용 메소드 : 변수 값을 버퍼에 보내서 내보내는 메소드 공통으로 write()메소드 사용함

4. 자바에서의 스트림 크게 2가지 구분

➔ 바이트(byte) 스트림 : 통로로 움직이는 데이터 크기가 1바이트임

(최상위 클래스 : InputStream / OutputStream)

➔ 문자(character) 스트림 : 통로로 움직이는 데이터 크기가 2바이트임 (문자 1개씩 움직임)

(최상위 클래스 : Reader / Writer)

5. 기반 스트림과 보조 스트림의 차이

➔ 기반 스트림 : 어떤 대상을 가지고 데이터 입출력을 할 지 선정하는 스트림 클래스

➔ 보조 스트림 : 기반 스트림의 성능을 보완한 클래스들 (속도 향상, 기능 추가, 변환 작업 등등..)

→ 보조 스트림들은 단독으로 사용(객체 생성) 못함 !!

→ 기반 스트림 클래스에 대한 객체를 사용해서 객체 생성하도록 되어있음

→ 하나의 기반 스트림을 대상으로 여러 개의 보조 스트림을 달 수 있다.

***** 표만 알면 된다!! ****

	바이트	**외부자원 대상**	**문자**
기반 스트림	**FileInputStream / FileOutputStream** ByteArrayInputStream / ByteArrayOutputStream PipedInputStream / PipedOutputStream StringBufferInputStream / StringBufferOutputStream	파일 메모리 프로세스 메모리	**FileReader / FileWriter** CharArrayReader / CharArrayWriter PipedReader / PipedWriter StringReader / StringWriter
보조 스트림	BufferedInputStream / BufferedOutputStream **DataInputStream / DataOutputStream** **ObjectInputStream / ObjectOutputStream**		**BufferedReader / BufferedWriter** **InputStreamReader / OutputStreamWriter**
최상위 클래스	**InputStream / OutputStream**		**Reader / Writer**

* docs 확인해보면 어떤 클래스가 기반 스트림이고 어떤 클래스가 보조 스트림인지 알 수 있다.

→ 각 클래스의 매개변수 생성자를 확인해보면..

6. 기반 스트림, 보조스트림 사용법

➔ **기반 스트림 사용법**

내용 : 외부자원 ------> 기반 스트림 -------> 메모리

코드 : 기반스트림 레퍼런스 = new 기반스트림 (외부자원) ;

실 예 : FileInputStream fis = new FileInputStream(new File("test.txt"));

➔ **보조 스트림 사용법**

내용 : 외부자원 ----> 기반 스트림 ------> 보조 스트림1 -----> 보조 스트림2 -----> 메모리

코드 : 보조스트림2 레퍼런스 = new 보조스트림2 (new 보조스트림1 (new 기반스트림 (외부자원)));

실 예 : BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

*** 보통 BufferedReader를 사용하는 이유는 BufferedReader의 readLine()메소드 때문이다.**

*** 표준 입출력 (System.in, System.out, System.err)**

- 콘솔(console, 화면)을 통한 데이터의 입출력을 말함.
- JVM이 시작되면서 자동적으로 생성되는 스트림 → 스트림을 열고 닫을 필요가 없다.
- 보통 단독으로 사용하지 않는다.

7. DataInputStream / DataOutputStream 보조스트림

- 바이트 스트림만 제공됨
- FileInputStream / FileOutputStream만을 가지고 데이터를 저장하고 불러올 때는 어떤 값을 넣어도 1byte 단위로 저장되고 불러옴

➔ Data Input/Output Stream 보조스트림을 가지고 작업을 하게 되면 1byte단위가 아닌 기본자료형(8가지)과 String 참조자료형 크기 단위로 저장되고 불러올 수 있음

➔ 값이 저장될 때 데이터형을 같이 저장시킴 → 파일을 열어보면 깨지는 걸 확인 할 수 있음

(인코딩이 안 돼서)

➔ 해당 자료형에 맞춰 write~()메소드 , read~()메소드를 통해 저장하고 불러옴

8. ObjectInputStream / ObjectOutputStream 보조스트림

- 바이트 스트림만 제공됨
- 객체가 가진 정보를 객체 상태로 파일에 기록하거나, 파일에서 읽어와서 바로 객체에 저장할 때 사용하는 스트림
- **반드시 직렬화 처리된 클래스의 객체만 사용할 수 있음**

- 객체 입출력에 사용되는 메소드

  출력 시 : writeObject(객체) >> IOException 처리해야 함

  입력 시 : readObject() >> IOException, ClassNotFoundException 2개를 처리해야 함

  >> 반환형이 Object이므로 해당 클래스 타입으로 형 변환해 됨

*** 직렬화 (java.io.Serializable)**

- 객체 클래스에 적용함 → 클래스가 객체 생성하고, 그 객체를 Object 입/출력에 사용할 경우, 바이트 스트림이므로 객체 상태로 그대로 스트림으로 전송 할 수 없음
- **직렬화가 적용된 클래스 일 때는 바이트 스트림 크기에 맞춰서, 객체가 가진 필드들을 기록된 순서대로 바이트 단위로 길게 연속으로 나열 처리를 함 ➔ 직렬화라고 함**
- 직렬화 처리된 객체 정보가 바이트 스트림을 거쳐 전송됨
- **읽어 들일 때는 역 직렬화가 자동으로 처리됨 ➔ 바이트 단위로 나열된 값들을 다시 각 자료형 별 필드 값으로 바꾸는 것**

➔ 직렬화 처리 방법 : 직렬화를 적용할 클래스 헤더에 java.io.Serializable 인터페이스 상속 처리함

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

*** 마크업(태그) 인터페이스 ***

- 추상메소드를 가지고 있지 않는 인터페이스 → 표시용 인터페이스