**[ 16 ] 자바 파일 입출력(i/o)**

목표 : ①자바 i/o stream 개념에 대해 안다. ②파일 입출력 API를 이용하여 text를 읽고 쓸 줄 안다. ③**File copy** 프로그램을 구현한다.

1. 입출력이란?

I/O란? 입력(Input)과 출력(Output)을 뜻 합니다. 프로그램에게 입력하는 것은 Input이며, 프로그램에서 어떤 것을 출력하는 것을 Output이라고 합니다.

앞에서 살펴본 ‘System.out.println("HelloWorld~~");’ 구문이 바로 Output 입니다.

앞으로 여러분은 어떤 데이터를 ‘읽는다’ 라고 하면 입력(Input)이라고 생각하시면 되고, 데이터를 ‘쓴다‘라고 하면 출력(Output)이라고 생각 하면 됩니다

1. Input : 파일 데이터를 읽는다. 키보드의 데이터를 읽는다. 네트워크상의 데이터를 읽는다(전송)
2. Output :파일에 데이터를 쓴다. 모니터에 데이터를 쓴다(출력). 네트워크상에 데이터를 쓴다(전송)
3. Java에서의 파일입출력 API
4. InputStream : 1byte단위 입력 API. 이미지, 동영상 등의 데이터에 주로 사용
5. OutputStream : 1byte단위 출력 API. 이미지, 동영상 등의 데이터에 주로 사용
6. Reader : 2byte단위 입력 API. 문자열에 주로 사용
7. Writer : 2byte단위 출력 API. 문자열에 주로 사용

|  |  |  |
| --- | --- | --- |
|  | 이미지동영상 등 데이터용(1byte단위) | 문자열용(2byte단위) |
| 입력 API | InputStream | Reader |
| 출력 API | OutputStream | Writer |

프로그램

도착지

출발지

입력스트림

출력스트림

출발지

1 .키보드

2. 파일

3. 네트워크

출력

1 .콘솔

2. 파일

3. 네트워크

1. BufferedReader / BufferedWriter

cf. 스트림(Stream)이란 데이터를 운반(입출력)하는데 사용되는 연결통로. 연속적인 데이터의 흐름을 물(stream)에 비유해서 붙여진 이름. 하나의 스트림으로 입출력을 동시에 수행할 수 없다(**단방향**통신). 입출력을 동시에 수행하려면 2개의 스트림이 필요하다

1. 파일로부터 데이터를 읽는 3단계 : 파일(연결통로)을 연다 -> 파일의 데이터를 읽는다(필요한 만큼 반복) -> 파일을 닫는다
2. 파일에 데이터를 쓰는 3단계 : 파일(연결통로)을 연다 -> 파일에 데이터를 쓴다 -> 파일을 닫는다
3. InputStream 사용법
   1. InputStream(추상) 클래스를 상속받은 여러 가지 API 하위 클래스 중의 하나를 이용해서 객체를 만든다. 또는 다른 클래스의 메소드에서 반환(리턴)되는 타입 객체를 얻는다.
   2. read(), read(byte[]) 두 개의 메소드를 이용하여 데이터를 읽는다.
      1. read() 1byte씩 읽는다. 속도가 느리다
      2. read(byte[]) byte[]만큼씩 읽는다. 속도가 빠르다.
   3. 예외 처리와 무조건 close() 실행
      1. I/O를 하면서 반드시 해야 하는 예외처리가 있습니다. IOException입니다. 반드시 하도록 컴파일러가 강요합니다
      2. I/O 작업 마지막은 close()로 외부 연결을 끝내야 합니다

//(1)파일을 연다(스트림객체생성) (2)데이터를 읽는다 (3)파일을 닫는다

**public** **class** Ex01\_inputStream {

**public** **static** **void** main(String[] args) {

InputStream is = **null**;

**try** {

is = **new** FileInputStream("txtFile/inTest.txt");//1.파일을 연다

// 2. 데이터를 읽는다(파일끝일때까지 1byte씩 반복적으로 읽는다)

**while**(**true**) {

**int** i = is.read(); // 1byte씩 읽기

**if**(i==-1) **break**; // 파일이 끝인지 여부

System.***out***.print((**char**)i + "("+i+")");

}

System.***out***.println("\nDONE");

} **catch** (FileNotFoundException e) {

System.***out***.println("예외메세지:파일못찾음"+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("예외메세지 : 못읽음 "+e.getMessage());

} **finally** {

// 3. 파일을 닫는다

**try** {

**if**(is!=**null**) is.close();

} **catch** (IOException e) { }

}

}

}

//(1)파일을 연다(스트림객체생성) (2)데이터를 읽는다 (3)파일을 닫는다

**public** **class** Ex02\_inputStreamByteArray {

**public** **static** **void** main(String[] args) {

InputStream is = **null**;

**try** {

is = **new** FileInputStream("txtFile/inTest.txt");//1.파일을 연다

// 2. 데이터를 읽는다(파일끝일때까지 1byte씩 반복적으로 읽는다)

**byte**[] bs = **new** **byte**[10];

**while**(**true**) {

**int** readByteCount = is.read(bs); // 10byte씩 읽기

**if**(readByteCount==-1) **break**; // 파일이 끝인지 여부

**for**(**byte** b : bs) {

System.***out***.print((**char**)b);

}

}

System.***out***.println("\nDONE");

} **catch** (FileNotFoundException e) {

System.***out***.println("예외메세지:파일못찾음"+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("예외메세지 : 못읽음 "+e.getMessage());

} **finally** {

// 3. 파일을 닫는다

**try** {

**if**(is!=**null**) is.close();

} **catch** (IOException e) { }

}

}

}

1. OutputStream 사용법
   1. OutputStream(추상) 클래스를 상속받은 여러가지 API 하위 클래스들 중의 하나를 이용해서 객체를 만든다. 또는 다른 클래스의 메소드에서 반환(리턴)되는 타입 객체를 얻는다.
   2. write()메소드를 이용해서 데이터를 읽으면 됩니다.
   3. write(int), write(byte[]), write(byte[], int, int) 세개의 메소드를 이용할 수 있습니다.
   4. 예외 처리와 무조건 close() 실행
      1. I/O를 하면서 반드시 해야 하는 예외처리가 있습니다. IOException입니다. 반드시 하도록 컴파일러가 강요합니다
      2. I/O 작업 마지막은 close()로 외부 연결을 끝내야 합니다

//1.파일을연다(스트림객체생성) 2.write한다 3.파일을닫는다(스트림객체.close)

**public** **class** Ex03\_outputStream {

**public** **static** **void** main(String[] args) {

OutputStream os = **null**;

**try** {

os = **new** FileOutputStream("txtFile/outTest.txt"); // (1)

**byte**[] bs = {'H','e','l','l','o'};

// for(int i=0 ; i<bs.length ; i++) { // (2)

// os.write(bs[i]);

// }

os.write(bs);

System.***out***.println("파일 출력 성공");

} **catch** (FileNotFoundException e) {

System.***out***.println("예외메세지:파일못찾음"+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("예외메세지 : 못읽음 "+e.getMessage());

} **finally** {

**try** {

**if**(os!=**null**) os.close(); // (3)

} **catch** (IOException e) { }

}//try-catch-finally

}//main

}

**public** **static** **void** main(String[] args) {

OutputStream os = **null**;

**try** {

os = **new** FileOutputStream("txtFile/outTest.txt"); // (1)

String str = "Hello, Java\n이 시국에 모두 건강하시길";

**byte**[] bs = str.getBytes();// 스트링을 byte배열로 바꾸는 함수

os.write(bs);

System.***out***.println("파일 출력 성공");

} **catch** (FileNotFoundException e) {

System.***out***.println("예외메세지:파일못찾음"+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("예외메세지 : 못읽음 "+e.getMessage());

} **finally** {

**try** {

**if**(os!=**null**) os.close(); // (3)

} **catch** (IOException e) { }

}// try-catch

}//main

1. 파일 읽고 쓰기
2. 읽고, 쓰기를 동시에 – 파일 복사
   1. 파일을 읽고, 다른 파일에 쓰고, 결국은 파일 복사 입니다.
   2. 작업순서 : InputStream, OutputStream 준비 > is로 읽어들인 데이터를 os으로 씀 > 외부연결 close()함.
   3. read() : read(), read(byte[]) 등 다양한 메소드 이용 가능
   4. write() : write(byte), write(byte[]), (write(byte[], int, int) 등 다양한 메소드 이용 가능

<1단계 file copy>

**public** **static** **void** main(String[] args) {

InputStream is = **null**;

OutputStream os = **null**;

**try** {

is = **new** FileInputStream("d:\\ai\\lecNote\\01\_java\\bts.jpg");

os = **new** FileOutputStream("d:/ai/lecNote/01\_java/bts\_copyed.jpg");

**int** cnt = 0;

**while**(**true**) {

**int** i = is.read();

**if**(i == -1) **break**;

os.write(i);

cnt++;

}

System.***out***.println(cnt+"번 while문 실행하며 힘들게 복사 성공");

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage());

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

}**finally** {

**try** {

**if**(os!=**null**) os.close();

**if**(is!=**null**) is.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}// try-catch

}//finally

}

<2단계 : filecopy 약간 수정 보완한 예제>

**public** **static** **void** main(String[] args) {

InputStream is = **null**;

OutputStream os = **null**;

**try** {

is = **new** FileInputStream("d:\\ai\\lecNote\\01\_java\\bts.jpg");

os = **new** FileOutputStream("d:/ai/lecNote/01\_java/bts\_copyed.jpg");

**int** cnt = 0;

**byte**[] bs = **new** **byte**[1024]; // 1킬로 byte

**while**(**true**) {

//int i = is.read();

**int** readByteCount = is.read(bs);

**if**(readByteCount == -1) **break**;

os.write(bs, 0, readByteCount);

cnt++;

}

System.***out***.println(cnt+"번 while문 실행하며 힘들게 복사 성공");

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage());

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

}**finally** {

**try** {

**if**(os!=**null**) os.close();

**if**(is!=**null**) is.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}// try-catch

}//finally

}

<3단계 더욱더 약간 수정 보완한 예제>

**public** **static** **void** main(String[] args) {

InputStream is = **null**;

OutputStream os = **null**;

**try** {

File file = **new** File("d:\\ai\\lecNote\\01\_java\\bts.jpg");

is = **new** FileInputStream(file);

os = **new** FileOutputStream("d:/ai/lecNote/01\_java/bts\_copyed.jpg ");

**int** cnt = 0;

**byte**[] bs = **new** **byte**[(**int**) file.length()]; // 1킬로 byte

**while**(**true**) {

//int i = is.read();

**int** readByteCount = is.read(bs);

**if**(readByteCount == -1) **break**;

os.write(bs, 0, readByteCount);

cnt++;

}

System.***out***.println(cnt+"번 while문 실행하며 힘들게 복사 성공");

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage());

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

}**finally** {

**try** {

**if**(os!=**null**) os.close();

**if**(is!=**null**) is.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}// try-catch

}//finally

}

1. 텍스트 읽고 쓰기
2. Reader ; 2byte 단위의 읽기

**public** **static** **void** main(String[] args) {

FileReader reader = **null**;

**try**{

reader = **new** FileReader("c:/gov/inTest.txt");

**while**(**true**){

**int** data = reader.read();

**if**(data == -1) **break**;

**char** ch = (**char**)data;

System.***out***.print(ch);

}

}**catch**(FileNotFoundException e){

System.***out***.println(e.getMessage());

}**catch**(IOException e){

System.***out***.println(e.getMessage());

}**finally** {

**if**(reader!= **null**)

**try** {

reader.close();

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

}

}

}

**public** **class** Ex01 {

**public** **static** **void** main(String[] args) {

Reader reader = **null**;

**try** {

reader = **new** FileReader("c:/mega\_IT/inTest.txt");

**char**[] cbuf = **new** **char**[6];

**while**(**true**) {

**int** cnt = reader.read(cbuf);

**if**(cnt==-1) **break**;

**for**(**int** idx=0 ; idx<cnt ; idx++)

System.***out***.print((**char**)cbuf[idx]);

}

}**catch** (Exception e) {

// **TODO**: handle exception

}**finally** {

**try** {

**if**(reader!=**null**) reader.close();

}**catch** (Exception e) {

// **TODO**: handle exception

}

}

}

}

1. BufferedReader, BufferedWriter 클래스

; 속도를 빨리 하기 위한 보조 스트림

**public** **class** Ex02\_filenameConsolePrint {

**public** **static** **void** main(String[] args) {

Reader reader = **null**;

BufferedReader br = **null**;

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("읽어올 파일명은 ? : "); // "abc.txt" => txtFile/abc.txt

String filename = scanner.next();

scanner.close();

File file = **new** File("txtFile/" + filename);

**if**(file.exists()) {

**try** {

reader = **new** FileReader(file); // 1.스트림 생성(기본스트림)

br = **new** BufferedReader(reader); // 기본스트림을 통해 보조스트림 생성

**while**(**true**) { // 2. 보조스트림을 통해 1줄씩 read

String linedata = br.readLine();

**if**(linedata == **null**) **break**;

System.***out***.println(linedata);

}

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage());

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

} **finally** {

**try** {

**if**(br!=**null**) br.close();

**if**(reader !=**null**) reader.close();

} **catch** (IOException e) {

System.***out***.println(e.getMessage());

}//close

}//try절

}**else** {

System.***out***.println("입력하신 파일은 존재하지 않습니다");

}//if

}//main

}

[ i/o에 앞서 File클래스] 파일 크기, 속성, 파일이름 정보를 갖고, 생성 및 삭제 메서드 포함

- 파일 안엔 디렉토리도 포함 : 생성, 디렉토리에 포함된 파일 리스트도 가져올 수 있다.

- 생성/삭제 메서드

File file = new File(“경로명/파일명”); 논리적인 파일이나 디렉토리

**exists();** 현재 파일이나 디렉토리가 있는지 여부

* 정보 메서드

**length();** 파일 크기