**[ 18 ] 자바 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 뜻합니다. 컴퓨터한테 입력하는 것은 Input이며, 컴퓨터가 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

4. 프로그램

출발지

1 .키보드

2. 파일

3. 네트워크

4. 프로그램

1. DataInputStream / DataOutputStream
2. BufferedReader / BufferedWriter
3. PrintWriter

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. 파일(스트림)을 연다 2.read한다 3.파일을닫는다(스트림객체.close)

**public** **class** Ex01inputStream {

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

InputStream is=**null**;

**try** {

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

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

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

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

System.***out***.println("데이터 : "+/\*(char)\*/i);

}

} **catch** (FileNotFoundException e) {

System.***out***.println("파일 못 찾음 : "+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("입출력 예외 : "+e.getMessage());

}**finally** {

**try** {

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

}**catch** (Exception e) {System.***out***.println(e.getMessage());}

}

}

}

**public** **class** Ex01inputStream-1 {

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

InputStream is=**null**;

**try** {

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

**byte**[] bs = **new** **byte**[3];

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

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

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

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

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

}

} **catch** (FileNotFoundException e) {

System.***out***.println("파일 못 찾음 : "+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("입출력 예외 : "+e.getMessage());

}**finally** {

**try** {

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

}**catch** (Exception e) {System.***out***.println(e.getMessage());}

}

}

}

**public** **class** Ex01inputStream-2 {

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

InputStream is=**null**;

**try** {

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

**byte**[] bs = **new** **byte**[5];

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

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

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

**for**(**int** idx=0 ; idx< readbyteCnt ;idx++) {

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

}

}

} **catch** (FileNotFoundException e) {

System.***out***.println("파일 못 찾음 : "+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("입출력 예외 : "+e.getMessage());

}**finally** {

**try** {

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

}**catch** (Exception e) {System.***out***.println(e.getMessage());}

}

}

}

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** Ex02outputStream {

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

OutputStream os = **null**;

**try** {

os = **new** FileOutputStream("c/mega\_IT/outTest.txt");

// byte[] bs = {'H','e','l','l','o',',',' ','J','a','v','a'};

String temp = "Hello, Java";

**byte**[] bs = temp.getBytes();

**for** (**int** idx = 0; idx < bs.length; idx++) {

System.***out***.println(bs[idx]);

os.write(bs[idx]);

//os.write((int)bs[idx]);

}

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage() + "파일 못 만들어");

} **catch** (IOException e) {

System.***out***.println(e.getMessage() + "쓰다가 예외");

} **finally** {

**try** {

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

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

}//try-catch-finall

}//main

}//class

**public** **class** Ex03OutputStream {

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

OutputStream os = **null**;

**try** {

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

//true : append / default는 false

String temp = "이번 겨울은 얼마나 추울려나\r\n감기조심하세요\r\n";

**byte**[] bs = temp.getBytes();

os.write(bs);

} **catch** (FileNotFoundException e) {

System.***out***.println(e.getMessage()+"파일 못 만들어");

} **catch** (IOException e) {

System.***out***.println(e.getMessage()+"쓰다가 예외");

} **finally** {

**try** {

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

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

}

}

}

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** **class** FileCopyStep1 {

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

InputStream is = **null**;

OutputStream os = **null**;

**try** {

is = **new** FileInputStream("c:\\mega\_IT\\sul.jpg");

os = **new** FileOutputStream("c:/mega\_IT/sulCopyed.jpg");

**int** cnt = 0;

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

cnt++;

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

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

os.write(i);

}

System.***out***.println("반복문 수행수 : "+cnt);

}**catch** (FileNotFoundException e) {

System.***out***.println("파일 못 찾음 : "+e.getMessage());

} **catch** (IOException e) {

System.***out***.println("읽기 쓰기 오류 : "+e.getMessage());

}**finally** {

**try** {

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

} **catch** (IOException e) { System.***out***.println(e.getMessage()); }

**try** {

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

} **catch** (IOException e) { System.***out***.println(e.getMessage()); }

}

}

}

<2단계 : filecopy 약간 수정 보완한 예제>

**public** **class** FileCopyEx2 {

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

InputStream is = **null**;

OutputStream os = **null**;

**try** {

is = **new** FileInputStream("C:\\gov\\inTest.txt");

os = **new** FileOutputStream("C:\\gov\\inCopy.txt");

**byte**[] bs = **new** **byte**[1024];

int count = 0;

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

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

System.out.println((++count)+”:”+ nReadByte);

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

os.write(bs, 0, nReadByte);

}

} **catch**(IOException e){

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

} **finally** {

**try**{

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

}**catch**(IOException e){

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

}

**try**{

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

}**catch**(IOException e){

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

}

}

}

}

<3단계 더욱더 약간 수정 보완한 예제>

**import** java.io.\*;

**public** **class** FileCopyEx2 {

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

InputStream is = **null**;

OutputStream os = **null**;

File file = **new** File("C:\\a01\_prog\\test.txt");

**try** {

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

os = **new** FileOutputStream("C:\\a01\_prog\\t.txt");

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

**int** i=0;

System.***out***.println((**int**) file.length()+"바이트 : "+bs.length);

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

System.***out***.println(++i);

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

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

os.write(bs, 0, count);

}

} **catch** (IOException e) {

e.printStackTrace();

} **finally**{

**try**{

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

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

}**catch** (Exception e) {System.***out***.println(e.getMessage());}

}

}

}

1. 조금 발전된, 속도를 빨리 하기 위한 Data 스트림 : DataInputStream, DataOutputStream

**import** java.io.DataOutputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.OutputStream;

**public** **class** Ex01DataOutputStream {

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

String dataFile = "c:/mega-IT/dataFile.dat";

OutputStream fos = **null**;

DataOutputStream dos = **null**;

**try** {

fos = **new** FileOutputStream(dataFile, **true**);

dos = **new** DataOutputStream(fos);

dos.writeUTF("Lee준석");

dos.writeInt(2);

dos.writeDouble(95.9);

}**catch** (IOException e) {

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

}**finally** {

**try**{

**if**(dos!=**null**){ dos.flush(); dos.close(); }

**if**(fos!=**null**){ fos.flush(); fos.close(); }

}**catch** (Exception ignore) { }

}

}

}

**import** java.io.DataInputStream;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**public** **class** Ex02DataInputStream {

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

String dataFile = "c:/mega-IT/dataFile.dat";

FileInputStream fis = **null**;

DataInputStream dis = **null**;

**try** {

fis = **new** FileInputStream(dataFile);

dis = **new** DataInputStream(fis);

String name; **double** score; **int** hak;

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

name = dis.readUTF();

grade = dis.readInt();

score = dis.readDouble();

System.***out***.println(name+"\t"+score+"\t"+ grade);

}

} **catch** (IOException e) {

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

} **finally** {

**try**{

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

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

}**catch** (Exception ignore) { }

}

}

}

< 오늘의 예제 >

**결과 파일 : Ex03\_ProductWrite.java, product.dat, Product.java, Ex04\_ProductList.java**

Ex03\_ProductWrite.java

product.dat저장 : Scanner를 활용하여 아래와 같은 재고데이터를 입력하여 데이터 파일에 저장한다(단, 재고 데이터는 Y/N를 입력할 때까지 입력한다)

Ex04\_ProductList.java

1단계 : 저장된 파일(product.dat)의 데이터를 불러 ArrayList에 넣는다

2단계 : ArrayList의 내용을 아래와 같이 출력하시오

Product.java (name, price, ps)

input data ex : 물건명(name) 가격(price) 개수(ps)

Apple 3000 2

Banana 4000 2

Mango 6000 3

/\* 출력 : 물건명 가격 재고갯수

apple 3000원 2개

banana 4000원 2개

mango 6000원 3개

이상 3가지 물품 입력됨 \*/

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

**import** java.io.\*;

**public** **class** BufferedEx2 {

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

FileReader reader = **null**;

BufferedReader br = **null**;

BufferedReader keyboard = **null**;

**try** {

keyboard = **new**

BufferedReader(**new** InputStreamReader(System.***in***));

System.***out***.print("읽어올 화일이름 입력해 : ");

String filename = keyboard.readLine();

reader = **new** FileReader(filename);

br = **new** BufferedReader(reader);

System.***out***.println(filename + "문서 내용은 다음과 같다★");

String linedata;

**while**((linedata = br.readLine())!=**null**){

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

}

} **catch** (FileNotFoundException e) {

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

} **catch** (IOException e2){

System.***out***.println(e2.getMessage());

} **finally** {

**try**{

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

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

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

}**catch**(Exception e){ }

}

}

}

1. Writer ; 2byte 단위의 쓰기

**public** **class** WriterEx {

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

FileWriter writer = **null**;

**try** {

writer = **new** FileWriter("c:/gov/outTest.txt", true);

/\*char[] arr = {'빼','앗','긴',' ','들', '\r',' ','\n', '광','복'};

for(char a : arr)

writer.write(a);\*/

String str = "빼앗긴 들\r\n광복";

writer.write(str);

}**catch**(IOException e){

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

} **finally** {

**try**{

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

}**catch**(Exception e){}

}

}

}

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

PrintWriter printWriter = **null**;

OutputStream os = **null**;

Writer writer = **null**;

**try** {

//printWriter = new PrintWriter("outTest.txt");

//writer = new FileWriter("outTest.txt");

//printWriter = new PrintWriter(writer);

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

printWriter = **new** PrintWriter(os);

printWriter.println("\r\n 성 적 표 ");

printWriter.printf("%5s : %3d %3d %5.1f\r\n", "홍길동",100,100,100.0);

printWriter.printf("%5s : %3d %3d %5.1f\r\n", "황보길동",99,100,99.5);

printWriter.printf("%5s : %3d %3d %5.1f\r\n", "김길동",90,90,90.0);

printWriter.printf("%5s : %3d %3d %5.1f\r\n", "신길동",80,81,80.5);

printWriter.printf("%1$tY년 %1$tm월 %1$td일 %1tH시 %1$tM분 %1$tS초", **new** Date());

}**catch** (Exception e) {

System.***out***.println(e.getMessage()+"예외");

}**finally** {

**try** {

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

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

}**catch** (Exception ignore) { }

}

}

[ 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

delete(); 파일 또는 디렉토리 삭제

* 정보 메서드

canExecute(); 실행할 수 있는 파일인지 여부

getName(); 파일이름

**getPath();** 전체 경로

isFile(), isDirectory(); 파일/디렉토리 인지 여부

**length();** 파일 크기

list(); 디렉토리인 경우 포함된 파일의 문자열 배열

listFiles(); 디렉토리인 경우 포함된 파일을 배열로

**public** **class** FileEx {

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

File dir = **new** File(".");

String[] strs = dir.list();

**for**(String s : strs)

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

File[] files = dir.listFiles();

System.***out***.println("파일명\t\t크기\t타입");

**for**(File f : files) {

**if**(f.getName().length()>=8) {

System.***out***.print(f.getName()+"\t");

}**else** {

System.***out***.print(f.getName()+"\t\t");

}

System.***out***.print(f.length()+"\t");

System.***out***.println((f.isDirectory()? "디렉토리":"파일")+"\t");

}

}

}

**public** **class** Ex2File {

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

File file = **new** File(".");

System.***out***.println("파일 상세 정보");

**try** {

System.***out***.println("절대 경로 : "+file.getAbsolutePath());

System.***out***.println("표준 경로 : "+file.getCanonicalPath());

Date thatTime = **new** Date(file.lastModified());

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy년 MM월 dd일 E ★a★ h시 m분 s초");

System.***out***.println("최종수정시점 : "+thatTime);

System.***out***.println("최종수정시점 : "+sdf.format(thatTime));

System.***out***.printf("최종수점시점 : %1$tY년 %1$tm월 %1$td일 (%1$ta) ★%1$tp★ %1$tl시 %1$tM분 %1$tS초\n", thatTime);

System.***out***.println("파일 크기 : "+file.length());

System.***out***.println("읽기 속성 : "+file.canRead());

System.***out***.println("쓰기 속성 : "+file.canWrite());

System.***out***.println("숨김 속성 : "+file.isHidden());

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

<오늘의 예제>

1. “N(n)”을 입력할 때까지 회원가입 정보(이름, 전화번호, 생일, 주소)를 ArrayList에 받는다

(단, 입력한 생일이 오늘이면 생일 축하 메시지 출력한다)

2. 가입한 정보를 콘솔(ArrayList)과 customer.txt파일에 출력

3. 제출파일명 (Customer.java, MainTest.java, customer.txt)을 제출

customer.txt 예시

홍길동 010-9999-9999 08-22생 서울시 용산구

김마동 010-8888-8888 12-18생 서울시 종로구

<Customer클래스 속성변수> name, phone, birth, address

김길동 010-9999-9999 12-16생 서울시 용산구

마길동 010-8888-8888 01-12생 서울시 종로구

<정리>

1. 입출력 스트림에 대한 설명 중 틀린 것은?
   1. 하나의 스트림으로 입력과 출력을 동시에 가능하다
   2. 프로그램을 기준으로 데이터가 들어오면 입력 스트림이다.
   3. 프로그램을 기준으로 데이터가 나가면 출력 스트림이다.
   4. 콘솔에 출력하거나, 파일에 저장하려면 출력 스트림을 사용하거나 Scanner를 이용해야 한다
2. InputStream과 Reader에 대한 설명으로 틀린 것은?
   1. 이미지 데이터는 InputStream 또는 Reader로 모두 읽는다.
   2. Reader의 read() 메소드는 1문자를 읽는다
   3. InputStream의 read() 메소드는 1바이트를 읽는다.
   4. InputStreamReader를 이용하면 InputStream을 Reader로 변환시킬 수 있다.

is.read(bs, 0, cnt)

1. InputStream의 read(byte[] b, int off, int len) 메소드에 대한 설명으로 틀린 것은?
   1. 메소드의 리턴값은 읽은 바이트 수이다.
   2. 첫번째 매개값 b에는 읽은 데이터가 저장된다.
   3. 두번째 매개값 off에는 첫번째 매개값 b에 데이터가 시작될 시작 인덱스이다
   4. 세번째 매개값 len은 첫번째 매개값 b에서 데이터가 저장된 마지막 인덱스이다