

Java IO (Input/Output)

파일과 문자셋(Charset)

파일과 문자셋(Charset) - **자바의 파일**

파일과 문자셋(Charset) - 자바의 파일

1 File 객체의 생성

- File 생성자

주의. File 객체는 실제 파일의 존재여부와는 상관 없음

(파일이 없는 경우 사용하려고 하는 시점에서 FileNotFoundException 발생)

파일 또는
폴더를
가리키는 객체

3

File(String pathname)	pathname 위치를 가리키는 파일 객체 생성
File(File parent, String child)	parent 폴더에 child 파일을 가리키는 파일 객체 생성
File(String parent, String child)	parent 폴더에 child 파일을 가리키는 파일 객체 생성
File(URI uri)	uri 위치를 가리키는 파일 객체 생성

4 - 실제 파일의 생성

파일 또는 폴더의 존재여부 확인

boolean exists()

경로 위치에 파일 생성

boolean createNewFile()

경로 위치에 폴더 생성

boolean mkdir()

예시

```
//#1-1. 파일객체 생성
File newFile = new File("C:/temp/newFile.txt");
//#1-2. 파일이 없는 경우 실제 파일 생성
if(!newFile.exists()) newFile.createNewFile();
```

5

Windows NTFS 파일시스템의 경우 권한문제로
코드로 C드라이브 루트에 파일쓰기 불가
해결책:
- 이클립스 관리자 권한으로 실행하면 동작 가능
- 가능한 C드라이브 root는 지정하지 말자

파일과 문자셋(Charset) - 자바의 파일

1 🖱️ File 경로 표시

2

- **File의 구분자(separator)** : System별 File 구분자 가져오기 → File의 정적 필드 **File.separator**

3

Windows에서는
두개 다 동작

- Windows 파일 구분자 : 역슬래시 "W"

"W"

단, W의 경우 문자열 " " 안에 사용시
제어문자로 인식하여 "WW"와 같이 표기

- Mac 파일 구분자 : 슬래시 "/"

C:WabcWbcd.txt

Windows

String path = "C:WWabcWWbcd.txt"

4

Mac

String path = "C:/abc/bcd.txt"

공통사용 가능

String path = "C:" + **File.separator** + "abc" + **File.separator** + "bcd.txt"

파일과 문자셋(Charset) - 자바의 파일

👉 File 경로 표시

1 - File의 절대경로 vs. 상대경로

File의 절대경로

- 드라이브명(C:, D: 등)부터 특정위치까지 절대적인 경로를 표기하는 방식

2

예시

//#. 절대경로

```
File newFile1 = new File("C:/abc/newFile11.txt");
```

```
File newFile2 = new File("C:/abc/bcd/newFile12.txt");
```

→ C:\abc\newFile11.txt

→ C:\abc\bcd\newFile12.txt

File의 상대경로

- 현재 작업폴더(working directory) 위치를 기준으로 상대적인 경로를 표기하는 방식

3

```
System.out.println(System.getProperty("user.dir"));
```

ex) 현재의 작업 위치가 C:/abc인 경우

4

예시

//#. 상대경로

```
File newFile1 = new File("newFile21.txt");
```

```
File newFile2 = new File("bcd/newFile22.txt");
```

→ C:\abc\newFile21.txt

→ C:\abc\bcd\newFile22.txt

파일과 문자셋(Charset) - 자바의 파일

👉 File 클래스 객체 생성 및 절대경로와 상대경로

예시

```
public static void main(String[] args) throws IOException {  
  
    1 // #1-0. C 드라이브내에 temp 폴더가 없는 경우 생성  
    File tempDir = new File("C:/temp");  
    if(!tempDir.exists()) tempDir.mkdir(); // temp 폴더가 없는 temp 폴더 생성  
  
    2 // #1-1. 파일객체 생성  
    File newFile = new File("C:/temp/newFile.txt");  
  
    3 // #1-2. 파일이 없는 경우 실제 파일 생성  
    if(!newFile.exists()) newFile.createNewFile(); // temp 폴더가 없는 경우 예외 발생  
  
    4 // #2. 파일 구분자  
    File newFile2 = new File("C:\\temp\\newFile.txt");  
    File newFile3 = new File("C:" + File.separator + "temp" + File.separator + "newFile.txt");  
    File newFile4 = new File("C:/temp/newFile.txt");  
  
    System.out.println(newFile2.exists());  
    System.out.println(newFile3.exists());  
    System.out.println(newFile4.exists());  
}
```

true
true
true

파일과 문자셋(Charset) - 자바의 파일

👉 **File** 클래스 객체 생성 및 절대경로와 상대경로

1

TIP

파일의 절대 경로 출력하기 구하기
String getAbsolutePath()

예시

2

//#3-1. 절대경로

```
File newFile5 = new File("C:/abc/newFile.txt");  
File newFile6 = new File("C:/abc/bcd/newFile.txt");  
System.out.println(newFile5.getAbsolutePath());  
System.out.println(newFile6.getAbsolutePath());
```

C:\abc\newFile.txt
C:\abc\bcd\newFile.txt

3

//#3-2. 상대경로

```
System.out.println(System.getProperty("user.dir")); //현재 작업 위치
```

4

```
File newFile7 = new File("newFile1.txt");  
File newFile8 = new File("bcd/newFile2.txt");  
System.out.println(newFile7.getAbsolutePath());  
System.out.println(newFile8.getAbsolutePath());
```

D:\java_exam_book2\Part04
D:\java_exam_book2\Part04\newFile1.txt
D:\java_exam_book2\Part04\bcd\newFile2.txt

}

파일과 문자셋(Charset) - 자바의 파일

File 클래스의 주요 메서드

1

String getAbsolutePath()	파일의 절대 경로를 문자열로 리턴
boolean isDirectory()	폴더 여부를 참/거짓으로 리턴
boolean isFile()	파일 여부를 참/거짓으로 리턴
String getName()	파일의 이름을 문자열로 리턴
String getParent()	부모 폴더의 이름을 문자열로 리턴
String[] list()	경로내의 폴더와 파일이름을 문자열 배열로 리턴
File[] listFiles()	경로내의 폴더와 파일이름을 파일 객체 배열로 리턴
boolean mkdir()	해당 경로에 폴더 생성 (하위 폴더만 생성가능)
boolean mkdirs()	존재하지 않는 경로상의 모든 폴더 생성

파일과 문자셋(Charset) - 자바의 파일

👉 File 클래스의 주요 메서드

예시

1

```
public static void main(String[] args) throws IOException {  
    //#. C 드라이브내에 temp 폴더가 없는 경우 생성  
    File tempDir = new File("C:/temp"); if(!tempDir.exists()) tempDir.mkdir();  
    //#. 파일객체 생성  
    File file = new File("C:/Windows");
```

2

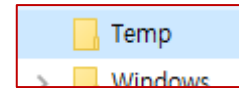
```
    //#. 파일 메서드  
    System.out.println("절대경로: "+file.getAbsolutePath());  
    System.out.println("폴더(?): "+file.isDirectory());  
    System.out.println("파일(?): "+file.isFile());  
    System.out.println("파일이름: "+file.getName()); //파일 또는 폴더이름  
    System.out.println("부모폴더: "+file.getParent());
```

5

```
    File newfile1 = new File("C:/temp/abc");  
    System.out.println(newfile1.mkdir()); //true : (이미 폴더가 있는 경우 false)  
    File newfile2 = new File("C:/temp/bcd/cde");  
    System.out.println(newfile2.mkdir()); //false  
    System.out.println(newfile2.mkdirs()); //true : (이미 폴더가 있는 경우 false)  
  
    File[] fnames = file.listFiles();  
    for(File fname : fnames)  
        System.out.println((fname.isDirectory()?"폴더: ":"파일:")+fname.getName());  
}
```

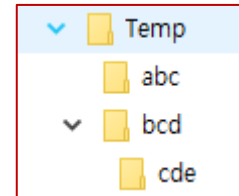
3

절대경로: C:\Windows
폴더(?): true
파일(?): false
파일이름: Windows
부모폴더: C:\



4

true
false
true



6

폴더: addins
폴더: appcompat
폴더: Application Data
폴더: apppatch
폴더: AppReadiness
폴더: assembly
폴더: bcastdvr
파일: bfsvc.exe
폴더: BitLockerDiscoveryVolumeContents

The End

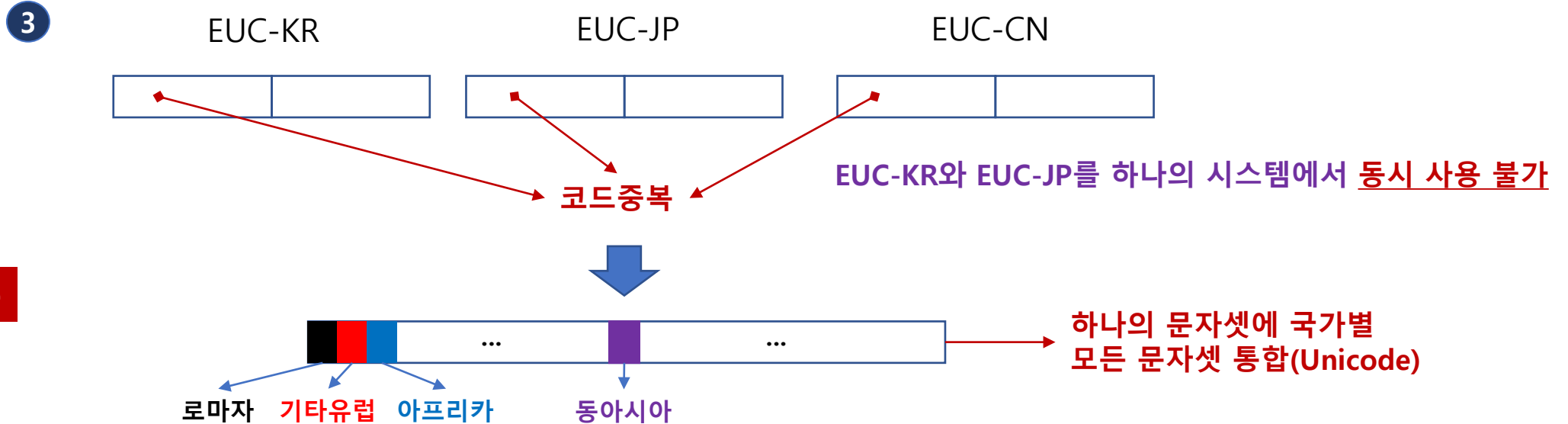
파일과 문자셋(Charset) - 자바의 문자셋(Charset)

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

1 🖱️ 아스키(ASCII) vs. 유니코드(Unicode)

- ### 2 ASCII
- 미국정보교환표준부호(**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
 - 영문 알파벳, 숫자, 특수기호, 제어코드로 구성
 - 7bit 정보포함 (실제 8bit) (MSB bit:0 → 표준 ASCII 코드, MSB : 1 → 나라별 코드 첨가 (16개 버전))

한국어, 일본어, 중국어의 문자 표현은?



파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 한글(영문/한자) 전용 문자셋 : **EUC-KR** vs. **MS949**

2

종성이 없는 경우

초성(19)x중성(21)개x중성(27+1)=11,172개

EUC-KR

1

- **KS 완성형** : 초기의 한글완성형 문자셋 (한글문자 11,172자 중 2,350자만 표기) : (8,822 글자누락) ex. 뽕 (공식명칭 : KS C 5601-1987)
- **EUC-KR** : KS X 1001 한국 산업 규격으로 지정된 한국어 문자 집합
KS완성형 + ASCII로 구성 : 즉, 한글 2,350자 표현 가능 (한자 4,888자 포함)
국가 표준으로 한글 웹페이지 표준 문자셋으로 사용
- ACSII 대응 문자는 1byte

MS949

3

- Windows에서 사용되는 한글완성형 표기 (2byte) (cf. Mac : UTF-8) : 영문은 모두 1 byte
- EUC-KR에 누락된 8,822자를 포함한 Microsoft에서 도입한 한글 기본 문자셋
(즉, KS완성형 + ASCII + 누락된 8,822자)
- EUC-KR과 하위호환성을 가짐
- 비표준으로 한글 웹페이지를 만드는 경우 EUC-KR 문자셋 사용
- ACSII 대응 문자는 1byte

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 한글(영문/한자) 전용 문자셋 : **EUC-KR** vs. **MS949**

TIP

1

문자열.getBytes(문자셋)
→ 문자셋을 기준으로 문자열을 byte[]로 분해해라.

new String(byte[], 문자셋)
→ 문자셋을 기준으로 byte[]을 문자열로 조합해라.

2

```
byte[] b1 = "a".getBytes("EUC-KR");  
byte[] b2 = "a".getBytes("MS949");
```

```
System.out.println(b1.length); //1  
System.out.println(b2.length); //1
```

```
System.out.println(new String(b1, "EUC-KR")); //a  
System.out.println(new String(b2, "MS949")); //a  
System.out.println();
```

1
1
a
a

3

```
byte[] b1 = "가".getBytes("EUC-KR");  
byte[] b2 = "가".getBytes("MS949");
```

```
System.out.println(b1.length); //2  
System.out.println(b2.length); //2
```

```
System.out.println(new String(b1, "EUC-KR")); //가  
System.out.println(new String(b2, "MS949")); //가  
System.out.println();
```

2
2
가
가

4

```
byte[] b3 = "뵤".getBytes("EUC-KR");  
byte[] b4 = "뵤".getBytes("MS949");
```

```
System.out.println(b3.length); //1  
System.out.println(b4.length); //2
```

```
System.out.println(new String(b3, "EUC-KR")); //?  
System.out.println(new String(b4, "MS949")); //뵤  
System.out.println();
```

1
2
?
뵤

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 대표적인 유니코드 문자셋 : **UTF-16** vs. **UTF-8**

UTF-16

1

- 고정 길이 문자 인코딩 방식(2byte) : 영문 및 한글 동일
- 자바에서의 char 자료형 저장을 위해 사용되는 방식 (char : 2byte)
- 저장 **문자열 앞**에 Little Endian/Big Endian 방식의 구분을 위한 **2byte (0xFEFF)** BOM(Byte Order Mark) 코드 삽입

2

"abc".getBytes("UTF-16"); → FE FF 00 61 00 62 00 63 → 8 byte

"가나다".getBytes("UTF-16"); → FE FF AC 00 B0 98 B2 E4 → 8 byte

UTF-8

3

- 가변 길이 문자 인코딩 방식(1byte~4byte)
- 대부분의 **웹서버**(Apache, IIS, NginX 등), **데이터베이스**(MySQL 등), **리눅스**, **Mac** 시스템의 기본 인코딩 방식
- 유니코드 한 문자를 나타내기 위해 1 byte~4byte까지를 사용
(4 byte로 표현되는 문자는 모두 기본 다국어 평면(BMP) 바깥의 유니코드 문자로 거의 사용 안됨)
- 아스키 코드 해당 문자는 1 byte, 한글은 3 byte로 표현 (U+AC00(가)~U+D7A3(힉))

4

"abc".getBytes("UTF-8") → 61 62 63 → 3 byte

"가나다".getBytes("UTF-8") → EA B0 80 EB 82 98 EB 8B A4 → 9 byte

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 대표적인 유니코드 문자셋 : **UTF-16** vs. **UTF-8**

byte[] **b1** = "abc".getBytes("UTF-16");
byte[] **b2** = "abc".getBytes("UTF-8");

System.out.println(b1.length); //8
System.out.println(b2.length); //3

for(byte b : **b1**)
System.out.printf("%02X ", b);
System.out.println();

for(byte b : **b2**)
System.out.printf("%02X ", b);
System.out.println();

System.out.println(new String(**b1**, "UTF-16"));
System.out.println(new String(**b2**, "UTF-8"));
System.out.println();

8
3
FE FF 00 61 00 62 00 63
61 62 63
abc
abc

byte[] **b3** = "가나다".getBytes("UTF-16");
byte[] **b4** = "가나다".getBytes("UTF-8");

System.out.println(b3.length); //8
System.out.println(b4.length); //9

for(byte b : **b3**)
System.out.printf("%02X ", b);
System.out.println();

for(byte b : **b4**)
System.out.printf("%02X ", b);
System.out.println();

System.out.println(new String(**b3**, "UTF-16"));
System.out.println(new String(**b4**, "UTF-8"));
System.out.println();

8
9
FE FF AC 00 B0 98 B2 E4
EA B0 80 EB 82 98 EB 8B A4
가나다
가나다

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 자바의 문자셋(Charset) ← `java.nio.charset.Charset` 클래스로 정의

- Charset 객체 생성 : 2가지 정적 메서드 사용

1

`static Charset defaultCharset()`

현재 설정되어 있는 디폴트 문자셋 리턴
(최소 파일단위까지 지정가능 /
일반적으로 프로젝트 또는 워크스페이스 단위로 설정)
미설정시 (**Windows JVM: MS949**, **Mac JVM: UTF-8**)

`static Charset forName(String charsetName)`

매개변수로 넘어온 charsetName의 문자셋 리턴
지원하지 않는 문자셋의 경우 `UnsupportedCharsetException` 실행
예외 발생

2

```
Charset cs1 = Charset.defaultCharset(); // = x-windows-949
Charset cs2 = Charset.forName("MS949"); // = x-windows-949
Charset cs3 = Charset.forName("UTF-8"); // UTF-8
```

→ 확장완성형

3

참고. JVM에서의 문자셋 지원여부 확인 : 정적메서드 `isSupported(...)`

`static boolean`

`isSupported(String charsetName)`

4

```
System.out.println(Charset.isSupported("MS949"));
System.out.println(Charset.isSupported("UTF-8"));
```

true
true

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 자바의 문자셋(Charset) ← `java.nio.charset.Charset` 클래스로 정의

```
Charset cs1 = Charset.defaultCharset();  
Charset cs2 = Charset.forName("MS949");  
Charset cs3 = Charset.forName("UTF-8");  
  
System.out.println(cs1);  
System.out.println(cs2);  
System.out.println(cs3);  
  
System.out.println(Charset.isSupported("MS949"));  
System.out.println(Charset.isSupported("UTF-8"));
```



Text file encoding

☒ Default (inherited from container: MS949)

☐ Other: MS949



```
x-windows-949  
x-windows-949  
UTF-8  
true  
true
```



Text file encoding

☐ Default (inherited from container: MS949)

☒ Other: UTF-8



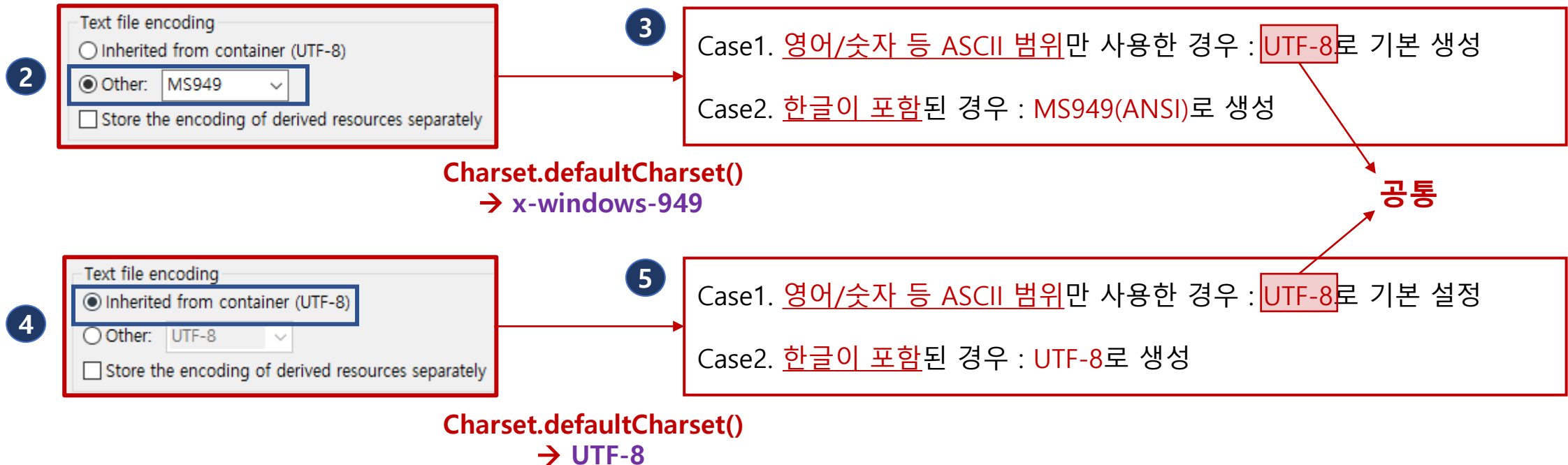
```
UTF-8  
x-windows-949  
UTF-8  
true  
true
```

x-windows-949

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 자바의 **문자셋(Charset)** ← `java.nio.charset.Charset` 클래스로 정의

1 - 자바 코드로 **파일 생성** / 이클립스 new file 생성 파일 **문자셋**

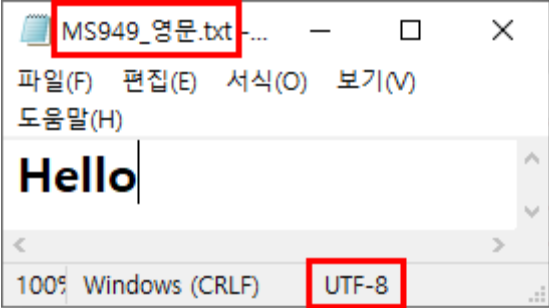


파일과 문자셋(Charset) - 자바의 Charset (문자셋)

☞ 자바의 **문자셋(Charset)** ← `java.nio.charset.Charset` 클래스로 정의

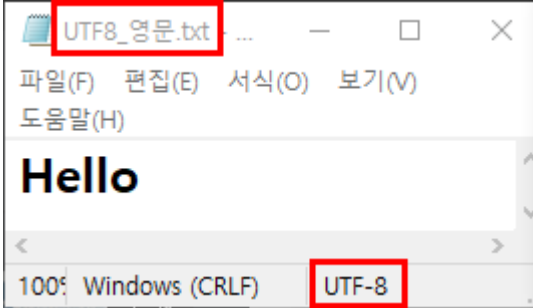
- 자바 코드로 **파일 생성** / 이클립스 new file 생성 파일 **문자셋**

1

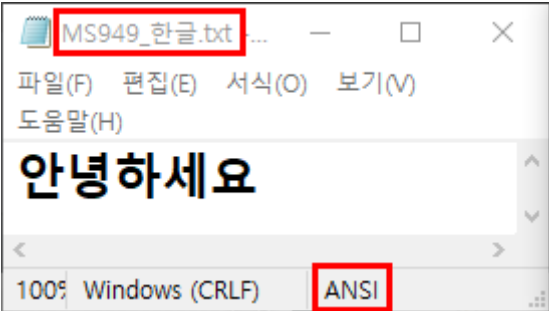


MS949_영문.txt ...
파일(F) 편집(E) 서식(O) 보기(V)
도움말(H)
Hello
100% Windows (CRLF) UTF-8

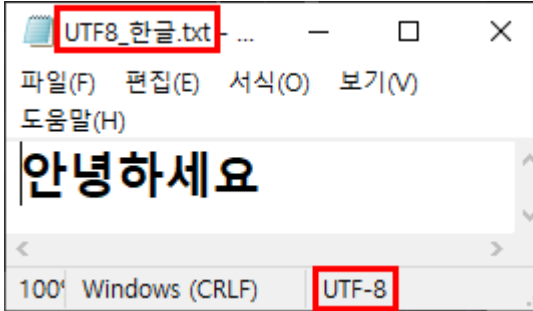
2



UTF8_영문.txt ...
파일(F) 편집(E) 서식(O) 보기(V)
도움말(H)
Hello
100% Windows (CRLF) UTF-8



MS949_한글.txt ...
파일(F) 편집(E) 서식(O) 보기(V)
도움말(H)
안녕하세요
100% Windows (CRLF) ANSI



UTF8_한글.txt ...
파일(F) 편집(E) 서식(O) 보기(V)
도움말(H)
안녕하세요
100% Windows (CRLF) UTF-8

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

👉 **명시적 문자셋 지정**이 필요한 경우

1 - Case 1. **문자열 → byte[]** 로 변환하는 경우 **String 클래스 인스턴스 메서드 getBytes(..)를 이용한 문자열 → byte[]**

: 어떤 문자셋을 사용하느냐에 따라 2byte 또는 3byte로 쪼개어 byte[]로 변환

2	byte[] getBytes()	문자열을 디폴트 문자셋(charset)을 이용하여 byte[]로 변환
	byte[] getBytes(Charset charset)	문자열을 매개변수 charset 문자셋(charset)을 이용하여 byte[]로 변환
	byte[] getBytes(String charsetName)	문자열을 매개변수 charsetName 이름의 문자셋(charset)을 이용하여 byte[]로 변환

3 - Case 2. **byte[] → 문자열**로 변환하는 경우 **String 생성자를 이용한 byte[] → 문자열**

: 어떤 문자셋을 사용하느냐에 따라 2byte 또는 3byte를 묶어 문자로 변환 필요

4	String(byte[] bytes)	매개변수 bytes을 디폴트 문자셋(charset)을 이용하여 문자열로 변환
	String(byte[] bytes, Charset charset)	매개변수 bytes을 매개변수 charset 문자셋을 이용하여 문자열 변환
	String(byte[] bytes, String charsetName)	매개변수 bytes을 매개변수 charsetName 문자셋을 이용하여 문자열 변환
	String(byte bytes[], int offset, int length, Charset charset)	매개변수 bytes의 offset 위치에서 부터 length개를 읽어와 매개변수 charset 문자셋을 이용하여 문자열 변환
	String(byte bytes[], int offset, int length, String charsetName)	매개변수 bytes의 offset 위치에서 부터 length개를 읽어와 매개변수 charsetName 문자셋을 이용하여 문자열 변환

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

👉 명시적 문자셋 지정이 필요한 경우

예시

1

```
//#1-1. String의 getBytes()를 이용한 byte[] --> 문자열  
byte[] array1 = "안녕".getBytes();  
byte[] array2 = "땡큐".getBytes(Charset.defaultCharset());  
byte[] array3 = "베리".getBytes(Charset.forName("MS949"));  
byte[] array4 = "감사".getBytes("UTF-8");
```

2

```
System.out.println(array1.length);  
System.out.println(array2.length);  
System.out.println(array3.length);  
System.out.println(array4.length);
```



Text file encoding

☒ Default (inherited from container: MS949)

☐ Other: MS949



4
4
4
6



Text file encoding

☐ Default (inherited from container: MS949)

☒ Other: UTF-8



6
6
4
6

파일과 문자셋(Charset) - 자바의 Charset (문자셋)

👉 명시적 문자셋 지정이 필요한 경우

예시

```
1 // #1-2. String 생성자를 이용한 문자열 --> byte[]  
String str1 = new String(array1);  
String str2 = new String(array2, Charset.defaultCharset());  
String str3 = new String(array3, Charset.forName("MS949"));  
String str4 = new String(array4, "UTF-8");  
  
2 String str5 = new String(array3, "UTF-8");  
String str6 = new String(array4, "MS949");  
  
3 System.out.println(str1); // defaultCharset-> defaultCharset  
System.out.println(str2); // defaultCharset-> defaultCharset  
System.out.println(str3); // MS949-> MS949  
System.out.println(str4); // UTF-8-> UTF-8  
System.out.println(str5); // MS949-> UTF-8 (깨짐)  
System.out.println(str6); // UTF-8-> MS949 (깨짐)
```

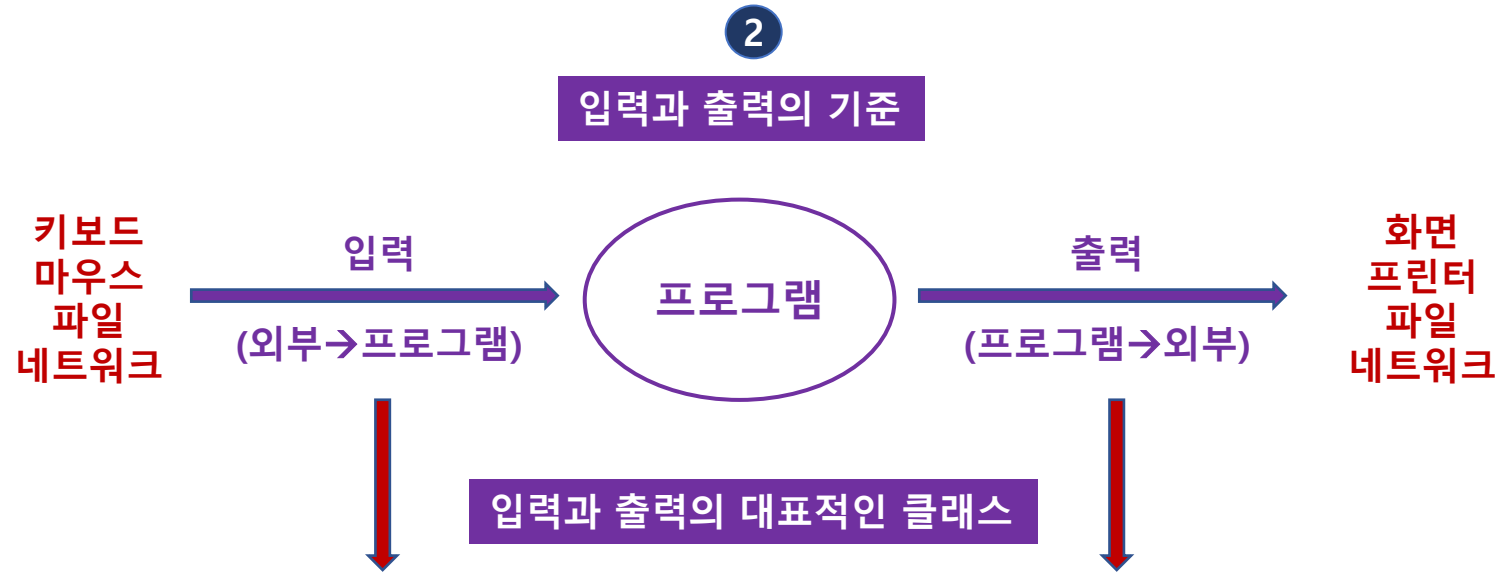
안녕
땡큐
베리
감사
????
媛맷꿀

The End

Java IO(Input/Output)의 개념

Java IO(Input/Output)의 개념

1 🖱️ Java IO : **입력(Input)**과 **출력(Output)**으로 구성



3

byte 단위의 입력	➡	InputStream		OutputStream	⬅	byte 단위의 출력
-------------	---	-------------	--	--------------	---	-------------

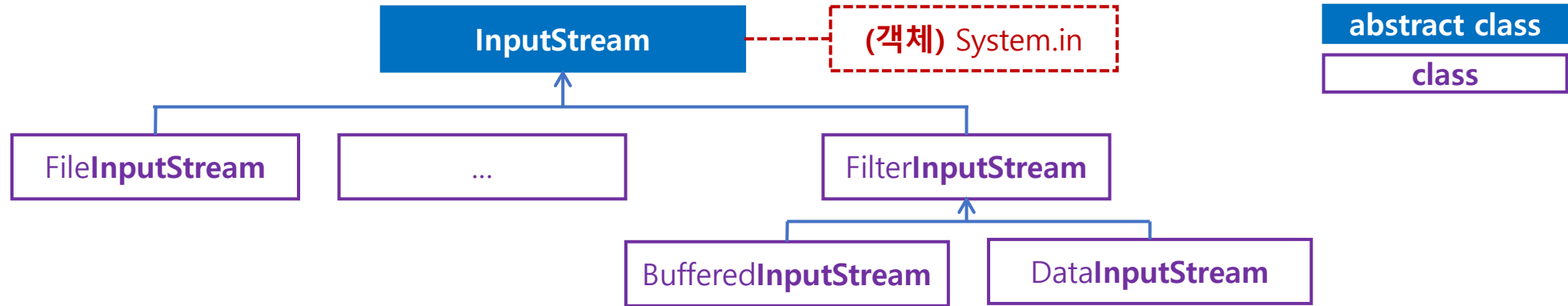
4

char 단위의 입력	➡	Reader		Writer	⬅	char 단위의 출력
-------------	---	--------	--	--------	---	-------------

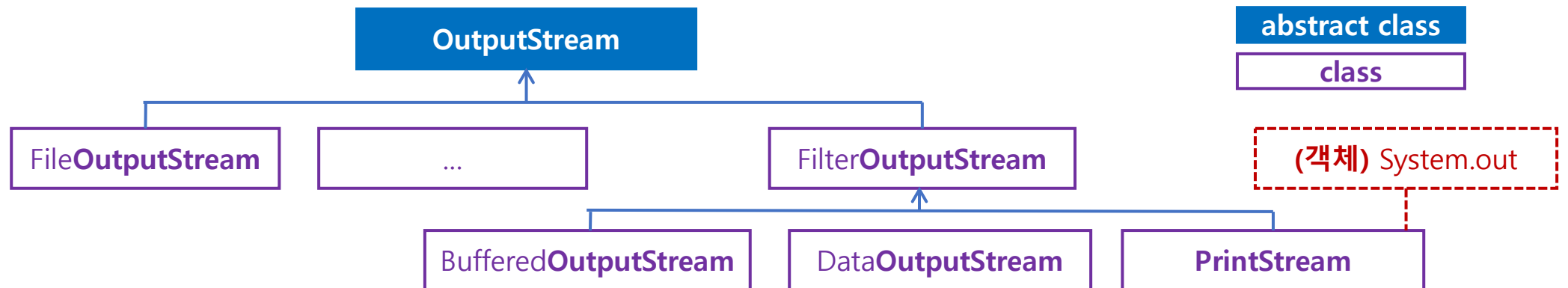
byte 단위의 입출력 (InputStream/OutputStream)

byte 단위의 입출력 (InputStream/OutputStream)

- 1 🖱️ InputStream ← **byte** 단위 **입력**을 수행하는 추상클래스



- 2 🖱️ OutputStream ← **byte** 단위 **출력**을 수행하는 추상클래스



byte 단위의 입출력 (InputStream/OutputStream)

3

InputStream ← byte 단위 입력을 수행하는 추상클래스

1

InputStream의 주요 메서드

실제 읽은 데이터는 1byte
int(4byte)의 마지막 byte 위치에 저장
(→ 읽은 데이터가 있는 경우 항상 + 값 리턴)
(더 이상 읽을 데이터 없는 경우 -)

2

int available()

InputStream의 남은 바이트 수를 리턴

abstract int read()

int(4byte)의 하위 1byte에 읽은 데이터를 저장하여 리턴 (추상 메서드)

int read(byte[] b)

읽은 데이터를 byte[] b의 0번째 위치부터 저장, 읽은 바이트 수를 리턴

int read(byte[] b, int off, int len)

length 개수만큼 읽은 데이터를 byte[] b의 offset 위치부터 저장

void close()

InputStream의 자원 반환

내부에서
이 메서드
호출

4

Q1

read() 메서드 하나만 추상 메서드이니까 겹데기만 아래처럼 overriding한 MyInputStream 클래스를 생성하고 나머지 완성된 메서드만(read(byte[] b), read(byte[] b, int off, int len)을 가지고 읽으면 될까? **NO!!**

```
class MyInputStream extends InputStream{
    @override
    int read() throws IOException { }
}
```

다른 read(..) 메서드에서 추상메서드 read()를 내부적으로 사용
속도를 위해 read()메서드는 **JNI를 이용하여 오버라이딩**

5

native int read();

편의를 위해 이미 JNI로 read()를 오버라이딩 한 하위클래스를
사용하여 InputStream 객체 생성 (**FileInputStream** 등)

byte 단위의 입출력 (InputStream/OutputStream)

OutputStream ← byte 단위 출력을 수행하는 추상클래스

1 OutputStream의 주요 메서드

내부에서 이 메서드 호출	2	void flush()	메모리 버퍼에 저장된 output stream 내보내기 (실제 출력 수행)
		abstract void write (int b)	int(4byte)의 하위 1byte를 output 버퍼에 출력 (추상 메서드)
		void write(byte[] b)	매개변수로 넘겨진 byte[] b의 0번째 위치부터 메모리버퍼에 출력
		void write(byte[] b, int off, int len)	byte[]의 offset 위치에서부터 length개수를 읽어와 출력
		void close()	OutputStream의 자원 반환

3 Q1 write(int b) 메서드 하나만 추상 메서드이니까 겹데기만 아래처럼 overriding한 MyOutputStream 클래스를 생성하고 나머지 완성된 메서드만(write(byte[] b), write (byte[] b, int off, int len)을 가지고 읽으면 될까? **NO!!**

```
class MyOutputStream extends OutputStream{
    @override
    void write(int b){ }
}
```

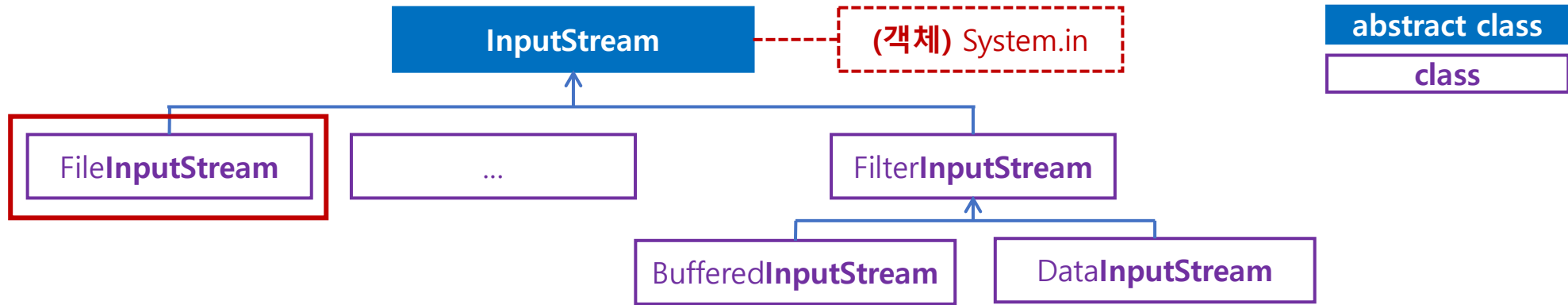
다른 write(..) 메서드에서 추상메서드 write(int b)를 내부적으로 사용
속도를 위해 write(int b)메서드는 **JNI를 이용하여 오버라이딩**

4 native void write();

편의를 위해 이미 JNI로 write()를 오버라이딩 한 하위클래스를
사용하여 OutputStream 객체 생성

The End

#1. FileInputStream/FileOutputStream으로
InputStream/OutputStream 객체 생성



#1-1. **FileInputStream**으로 **InputStream** 객체 생성

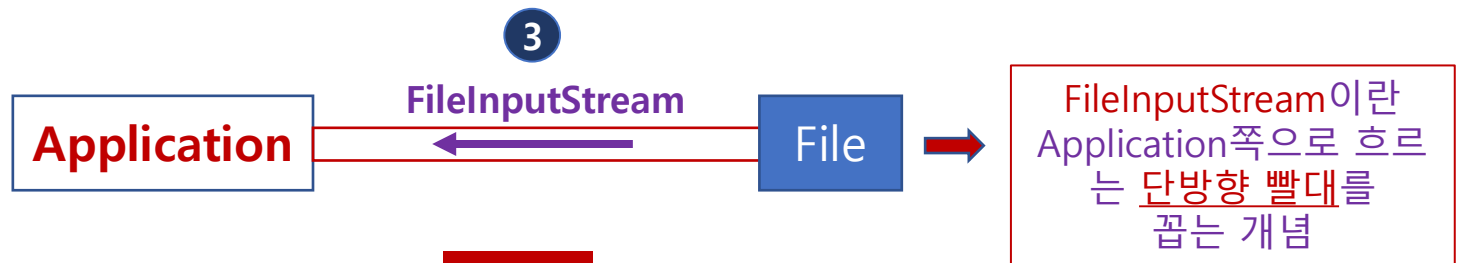
byte 단위의 입출력 (InputStream/OutputStream)

1 🖱️ FileInputStream ← File의 내용을 byte 단위로 데이터를 읽는 InputStream을 상속한 클래스

2 - FileInputStream 생성자

FileInputStream(<u>File</u> file)	매개변수로 넘어온 file을 읽기 위한 InputStream 생성
FileInputStream(<u>String</u> name)	매개변수로 넘어온 name 위치의 파일을 읽기 위한 InputStream 생성

- FileInputStream 객체 생성



예시 1

4

```
//#1. 파일객체 생성
File inFile = new File("infile.txt");
//#2. FileInputStream 객체 생성
InputStream fis = new FileInputStream(inFile);
```

예시 2

5

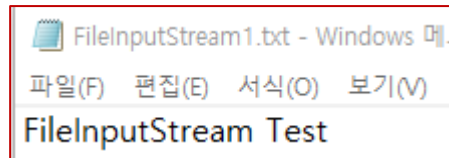
```
//#1. 파일경로로 바로 FileInputStream 객체 생성
InputStream fis = new FileInputStream("infile.txt");
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1 - int available(), void close()

2



예시

3

```
//입력파일 생성
File inFile = new File("src/pack02_javaio/sec02_files/FileInputStream1.txt");

//InputStream 생성
InputStream is = new FileInputStream(inFile);

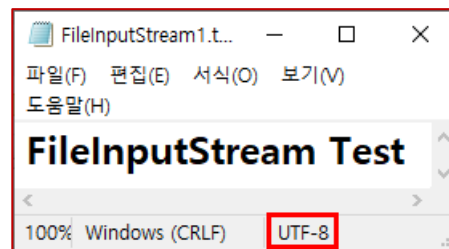
int data;
while((data=is.read())!=-1) {
    System.out.println("읽은 데이터 : " + (char)data + " 남은 바이트수: " + is.available());
}

//InputStream 자원반납
is.close();
```

파일의 끝을 나타냄 ★

읽은 데이터 :	F	남은 바이트수 :	19
읽은 데이터 :	i	남은 바이트수 :	18
읽은 데이터 :	l	남은 바이트수 :	17
읽은 데이터 :	e	남은 바이트수 :	16
읽은 데이터 :	I	남은 바이트수 :	15
읽은 데이터 :	n	남은 바이트수 :	14
읽은 데이터 :	p	남은 바이트수 :	13
읽은 데이터 :	u	남은 바이트수 :	12
읽은 데이터 :	t	남은 바이트수 :	11
읽은 데이터 :	S	남은 바이트수 :	10
읽은 데이터 :	t	남은 바이트수 :	9
읽은 데이터 :	r	남은 바이트수 :	8
읽은 데이터 :	e	남은 바이트수 :	7
읽은 데이터 :	a	남은 바이트수 :	6
읽은 데이터 :	m	남은 바이트수 :	5
읽은 데이터 :		남은 바이트수 :	4
읽은 데이터 :	T	남은 바이트수 :	3
읽은 데이터 :	e	남은 바이트수 :	2
읽은 데이터 :	s	남은 바이트수 :	1
읽은 데이터 :	t	남은 바이트수 :	0

1



2 파일: **영문** → 읽기방식: **영문**(byte)

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1 - **int read()**, **int read(byte[])**, **int read(byte[], int offset, int length)**

예시

파일: 영문 → 읽기방식: 영문(byte)

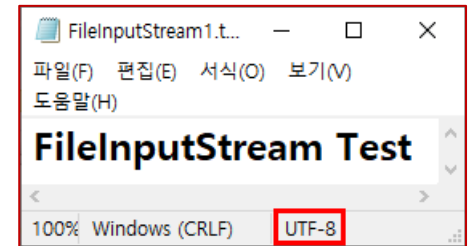
```
//입력파일 생성
File inFile = new File("src/pack02_javaio/sec02_files/FileInputStream1.txt ");

//#1. 1-byte 단위 읽기
InputStream is1 = new FileInputStream(inFile);

int data;
while((data=is1.read())!= -1 ) {
    System.out.print((char)data);
}

System.out.println(); System.out.println();
```

2



FileInputStream Test

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1 - `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

예시

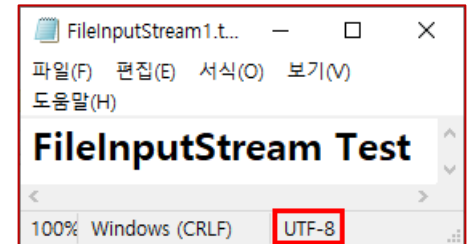
파일: 영문 → 읽기방식: 영문(byte)

##2-1. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)

```
byte[] byteArray1 = new byte[9];  
InputStream is2 = new FileInputStream(inFile);
```

```
int count1;  
while((count1 = is2.read(byteArray1)) != -1) {  
    for(int i=0; i<count1; i++)  
        System.out.print((char)byteArray1[i]);  
    System.out.println(" : count="+count1);  
}
```

```
System.out.println(); System.out.println();
```



3

```
FileInput : count=9  
Stream Te : count=9  
st : count=2
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1 - `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

예시

파일: 영문 → 읽기방식: 영문(byte)

2

```
//#3-1. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)
InputStream is3 = new FileInputStream(inFile);

int offset=3; int length=6;
byte[] byteArray2 = new byte[9]; //offset+length
int count2 = is3.read(byteArray2, offset, length); //offset:3 length:6

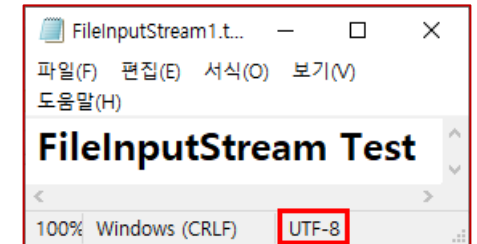
for(int i=0; i<offset+count2; i++)
    System.out.print((char)byteArray2[i]);

//#InputStream 자원 반납
is1.close();
is2.close();
is3.close();
```

Tip

3

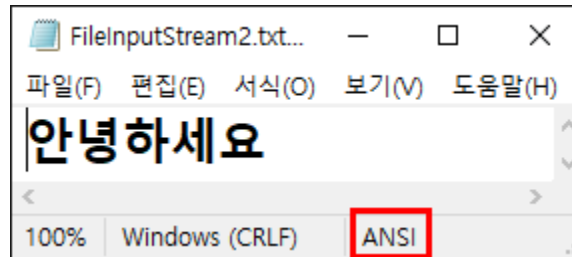
파일은 항상 0번 위치부터 읽거나 쓰기 가능
(RandomAccessFile 제외)



4

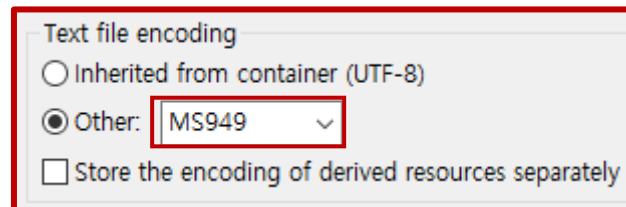
FileIn

1



2 파일: 한글 → 읽기방식: 영문(byte)→ 문자열(String) 변환

3 Charset.defaultCharset() → MS949



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

2 - `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

직접 한글읽기 불가능

String 생성자를 이용하여 `byte[]` → `String`로 변환하기

예시

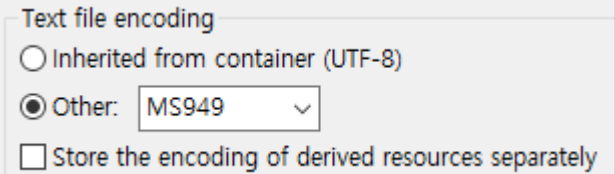
파일: 한글 → 읽기방식: 영문(`byte`) → 문자열(`String`) 변환

```
//#2-2. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)
byte[] byteArray1 = new byte[8];
InputStream is2 = new FileInputStream(inFile);

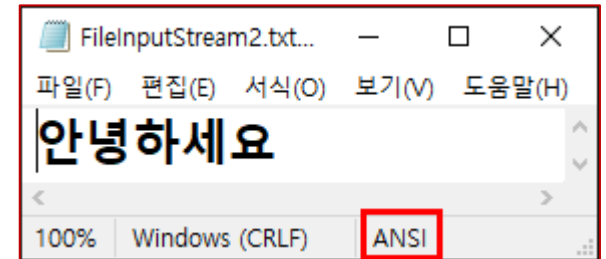
int count1;

while((count1 = is2.read(byteArray1)) != -1) {
    String str = new String(byteArray1, 0, count1, Charset.forName("MS949"));
    System.out.print(str);
    System.out.println(" : count="+count1);
}

System.out.println(); System.out.println();
```



1



4

안녕하세 : count=8
요 : count=2

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1

- `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

직접 한글읽기 불가능

String 생성자를 이용하여 `byte[]` → `String`로 변환하기

예시

파일: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

///`3-2. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)`
`InputStream is3 = new FileInputStream(inFile);`

`int offset=2; int length=6;`
`byte[] byteArray2 = new byte[8]; //offset+length`
`int count2 = is3.read(byteArray2, offset, length); //offset:2 length:6`

2

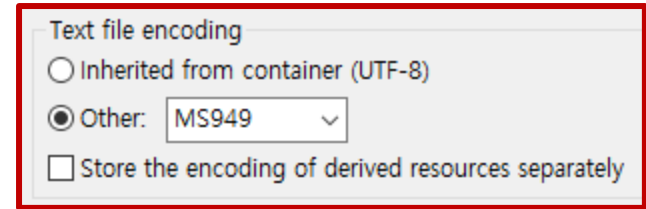
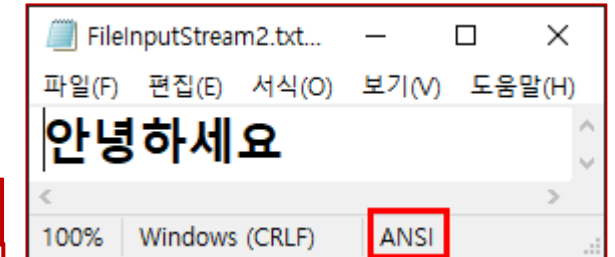
`String str = new String(byteArray2, 0, offset+count2, Charset.defaultCharset());`
`String str = new String(byteArray2, offset, count2, Charset.defaultCharset()); //안녕하`
`System.out.println(str);`

///`#InputStream 자원 반납`
`is2.close();`
`is3.close();`

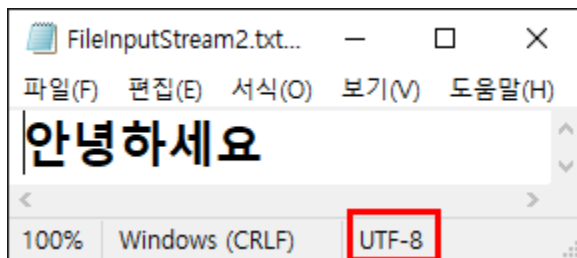
4

3

안녕하

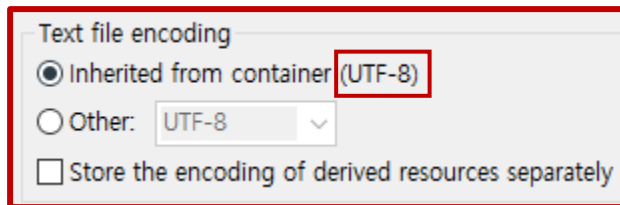


1



2 파일: 한글 → 읽기방식: 영문(byte)→ 문자열(String) 변환

3 Charset.defaultCharset() → UTF-8



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

2 - `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

직접 한글읽기 불가능

String 생성자를 이용하여 `byte[]` → `String`로 변환하기

예시

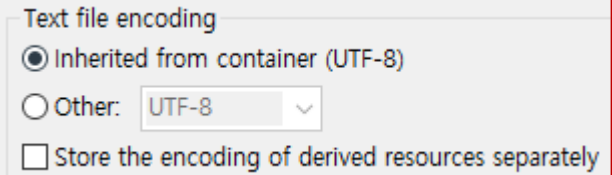
파일: 한글 → 읽기방식: 영문(`byte`) → 문자열(`String`) 변환

```
//#2-2. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)
byte[] byteArray1 = new byte[9];
InputStream is2 = new FileInputStream(inFile);

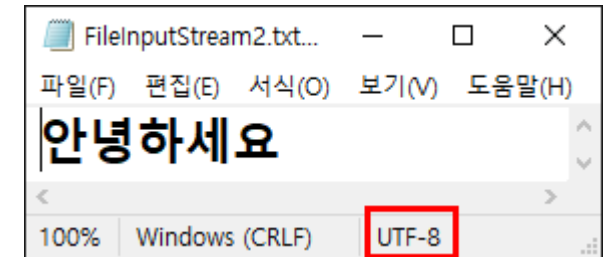
int count1;

while((count1 = is2.read(byteArray1)) != -1) {
    String str = new String(byteArray1, 0, count1, Charset.forName("UTF-8"));
    System.out.print(str);
    System.out.println(" : count=" + count1);
}

System.out.println(); System.out.println();
```



1



4

안녕하 : count = 9
세요 : count = 6

byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)InputStream 메서드

1

- `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

직접 한글읽기 불가능

String 생성자를 이용하여 `byte[]` → `String`로 변환하기

예시

파일: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

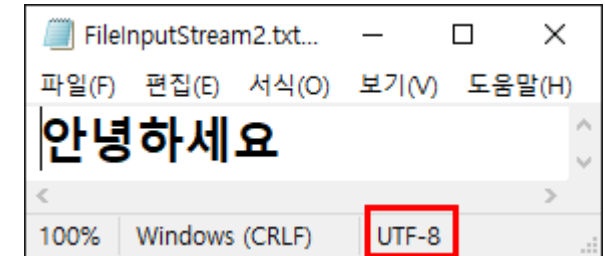
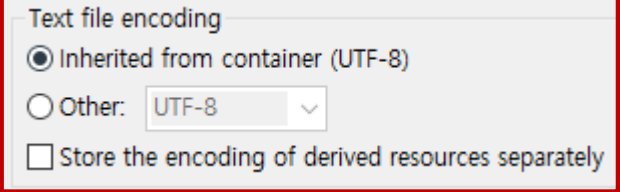
///`3-2. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)`
`InputStream is3 = new FileInputStream(inFile);`

`int offset=3; int length=6;`
`byte[] byteArray2 = new byte[9]; //offset+length`
`int count2 = is3.read(byteArray2, offset, length); //offset:3 length:6`

2

`String str = new String(byteArray2, 0, offset+count2, Charset.defaultCharset());`
`//String str = new String(byteArray2, offset, count2, Charset.defaultCharset()); //안녕`
`System.out.println(str);`

///`#InputStream 자원 반납`
`is2.close();`
`is3.close();`

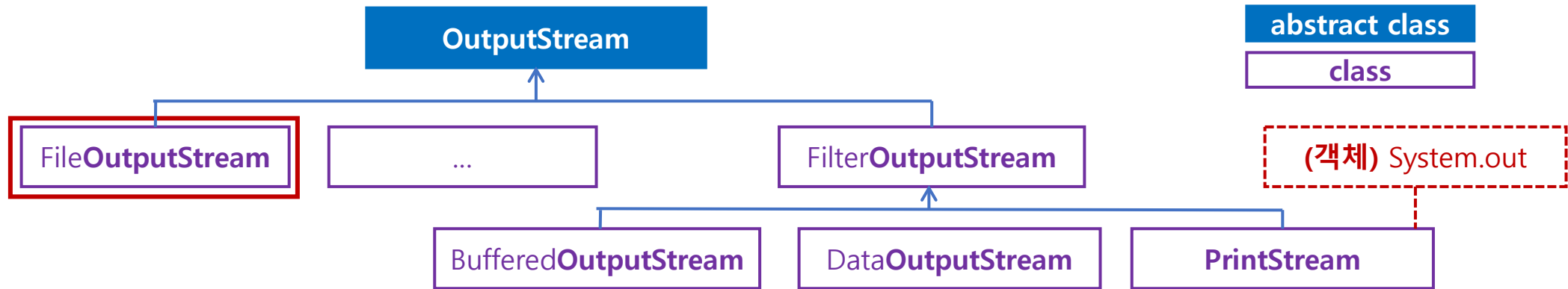


3

안녕

4

The End



#1-2. **FileOutputStream**으로 **OutputStream** 객체 생성

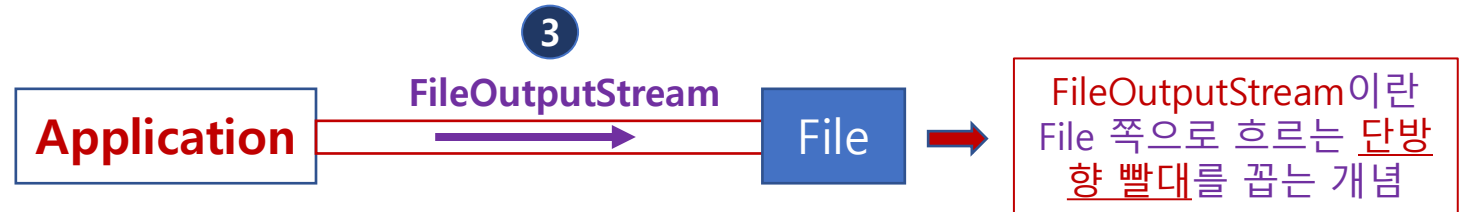
byte 단위의 입출력 (InputStream/OutputStream)

- 1 🖱️ FileOutputStream ← File에 byte 단위로 데이터를 쓰는 OutputStream을 상속한 클래스

- FileOutputStream 생성자

2	FileOutputStream(File file)	매개변수로 넘어온 file을 쓰기 위한 OutputStream 생성, append=true인 경우 <u>이어쓰기</u> , append=false인 경우 새로 <u>덮어쓰기</u> (default = false)
	FileOutputStream(File file, boolean append)	
	FileOutputStream(String name)	매개변수로 넘어온 name 위치의 파일을 쓰기 위한 OutputStream 생성 append=true인 경우 <u>이어쓰기</u> , append=false인 경우 <u>덮어쓰기</u> (default = false)
	FileOutputStream(String name, boolean append)	

- FileOutputStream 객체 생성



예시 1

4

```
//#1. 파일객체 생성
File outFile1 = new File("outfile1.txt");
File outFile2 = new File("outfile2.txt");
//#2. FileOutputStream 객체 생성
OutputStream fos1 = new FileOutputStream(outFile1);
OutputStream fos2 = new FileOutputStream(outFile2, true);
```

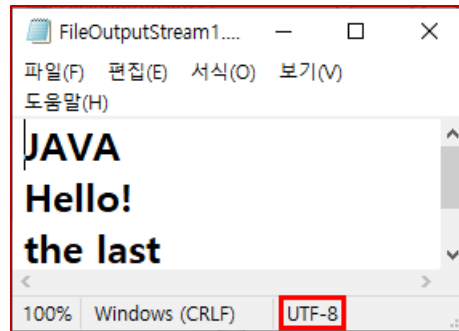
예시 2

5

```
//#1. 파일경로로 바로 FileOutputStream 객체 생성
OutputStream fos3 = new FileOutputStream("outfile1.txt");
OutputStream fos4 = new FileOutputStream("outfile2.txt", true);
```

① 쓰기방식: **영문**(byte) → 파일: **영문**(byte)

②



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)OutputStream 메서드의 활용

- 1 - **void write(int b), void flush(), void close()**

예시

쓰기방식: 영문(byte) → 파일: 영문(byte)

//입력파일 생성

```
File outFile = new File("src/pack02_javaio/sec02_files/FileOutputStream1.txt");  
if(!outFile.exists()) outFile.createNewFile(); //파일을 쓰는 경우에는 생략가능 (자동 생성)
```

//#1. 1-byte 단위 쓰기

```
OutputStream os1 = new FileOutputStream(outFile); //덮어쓰기
```

```
os1.write('J');
```

```
os1.write('A');
```

```
os1.write('V');
```

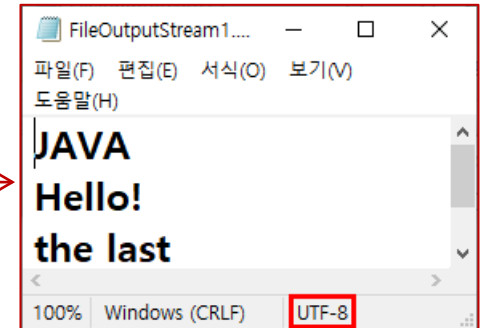
```
os1.write('A');
```

```
os1.write("Wr"); //13
```

```
os1.write("Wn"); //10 → 코드에서는 Wn만으로도 개행 가능
```

```
os1.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os1.close();
```

단, 윈도우 콘솔에서 enter 입력시 2byte(WrWn) 입력됨



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)OutputStream 메서드의 활용

- 1 - **void write(byte[] b)**, **void flush()**, **void close()**

예시

쓰기방식 : 영문(byte) → 파일: 영문(byte)

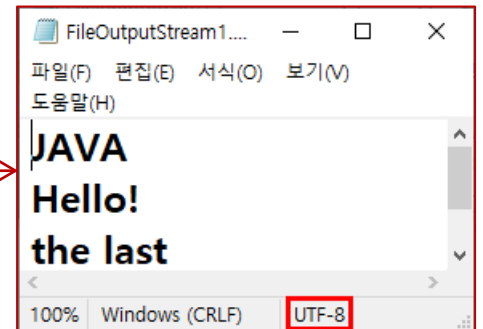
```
//#2. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝가지를 출력)  
OutputStream os2 = new FileOutputStream(outFile, true); //내용 연결
```

```
byte[] byteArray1 = "Hello!".getBytes(); → String→byte[]
```

```
os2.write(byteArray1);  
os2.write('Wn');
```

```
os2.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os2.close();
```

3



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)OutputStream 메서드의 활용

- 1 - **void write(byte[] b, int off, int len), void flush(), void close()**

예시

쓰기방식 : 영문(byte) → 파일: 영문(byte)

//#3. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)

OutputStream os3 = new FileOutputStream(outFile, **true**); //내용 연결

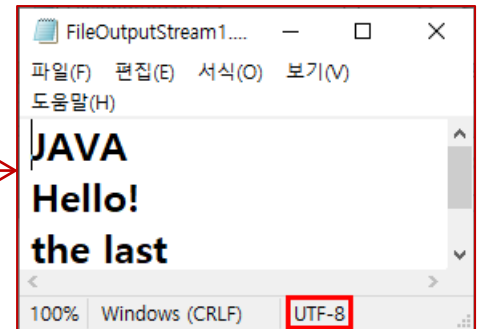
byte[] byteArray2 = "Better **the last** smile than the first laughter."**.getBytes();**

os3.**write(byteArray2, 7, 8);**

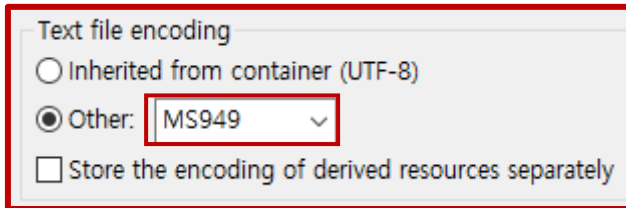
os3.**flush();** //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능
os3.close();

String→byte[]

3

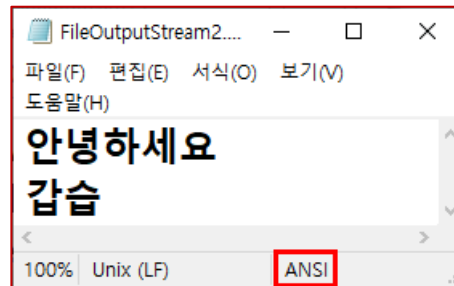


1 Charset.defaultCharset() → MS949



2 문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

3

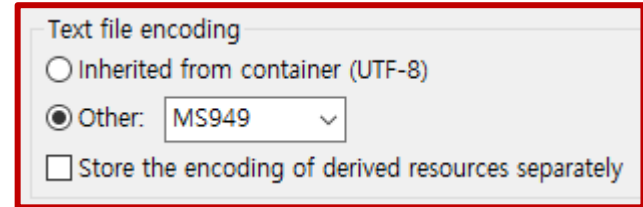


byte 단위의 입출력 (InputStream/OutputStream)

3

☞ (File)OutputStream 메서드의 활용

1 - **void write(byte[] b), void flush(), void close()**



String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

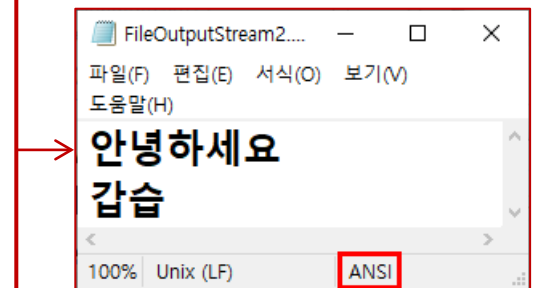
//#2. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝까지를 출력)

```
byte[] byteArray1 = "안녕하세요".getBytes();
```

```
OutputStream os2 = new FileOutputStream(outFile, false); //덮어쓰기  
os2.write(byteArray1);  
os2.write('\n');
```

```
os2.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os2.close();
```

4



byte 단위의 입출력 (InputStream/OutputStream)

☞ (File)OutputStream 메서드의 활용

1 - `void write(byte[] b, int offset, int length), void flush(), void close()`

String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

2

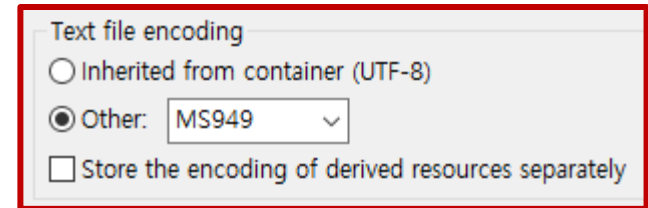
///`3`. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)

```
byte[] byteArray2 = "반갑습니다.".getBytes(Charset.defaultCharset());
```

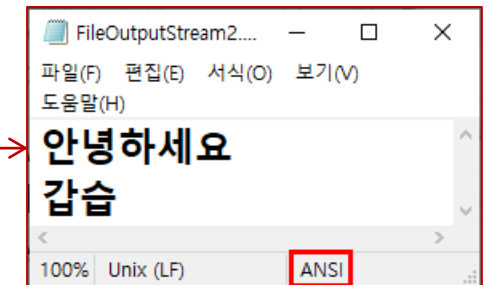
```
OutputStream os3 = new FileOutputStream(outFile, true); //내용 연결  
os3.write(byteArray2, 2, 4);
```

```
os3.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os3.close();
```

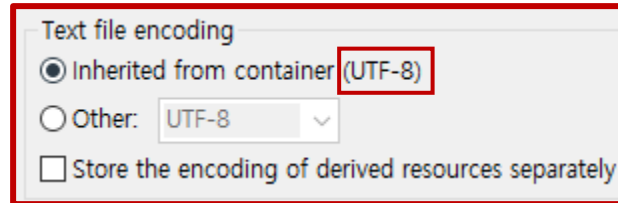
3



4

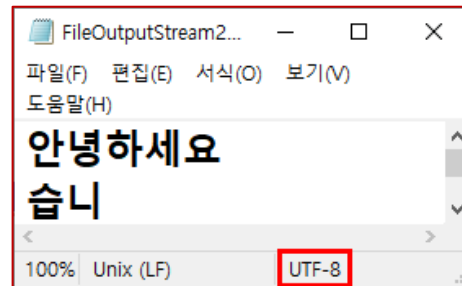


① Charset.defaultCharset() → UTF-8



② 문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

③

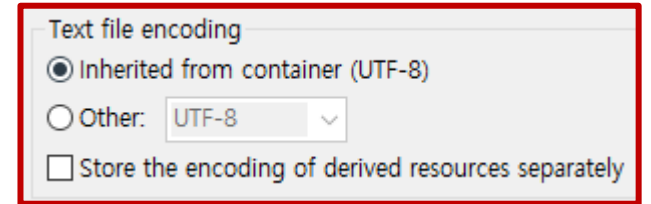


byte 단위의 입출력 (InputStream/OutputStream)

3

☞ (File)OutputStream 메서드의 활용

1 - **void write(byte[] b), void flush(), void close()**



String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

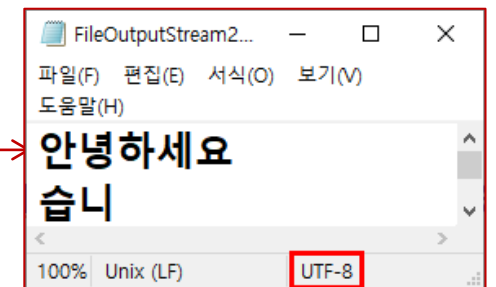
//#2. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝까지를 출력)

```
byte[] byteArray1 = "안녕하세요".getBytes();
```

```
OutputStream os2 = new FileOutputStream(outFile, false); //덮어쓰기  
os2.write(byteArray1);  
os2.write('\n');
```

```
os2.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os2.close();
```

4

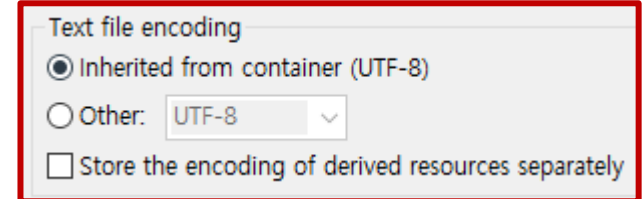


byte 단위의 입출력 (InputStream/OutputStream)

3

☞ (File)OutputStream 메서드의 활용

1 - `void write(byte[] b, int offset, int length), void flush(), void close()`



String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 파일: 한글

2

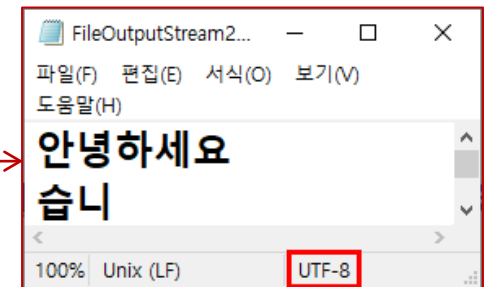
##3. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)

```
byte[] byteArray2 = "반갑습니다.".getBytes(Charset.defaultCharset());
```

```
OutputStream os3 = new FileOutputStream(outFile, true); //내용 연결  
os3.write(byteArray2, 6, 6);
```

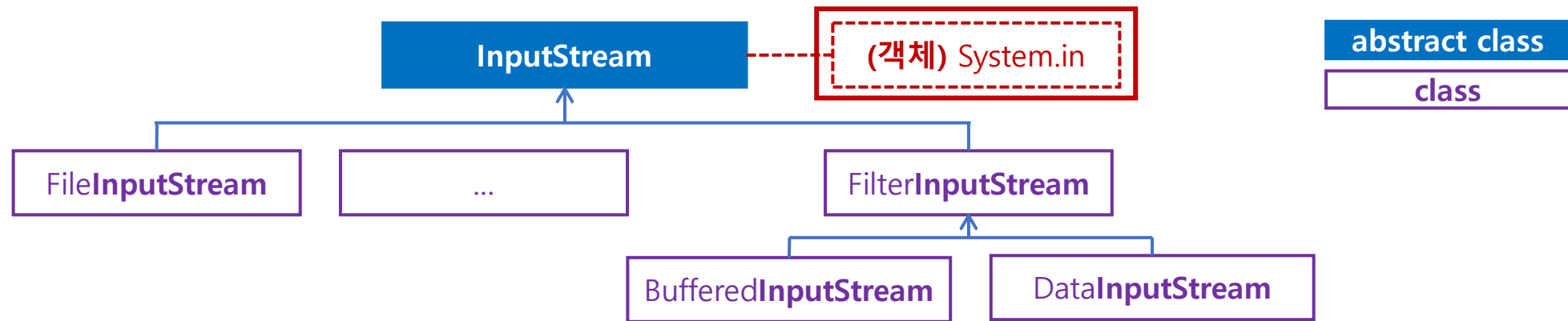
```
os3.flush(); //FileOutputStream은 내부적으로 메모리 버퍼를 사용하지 않아 생략해도 가능  
os3.close();
```

4



The End

#2. (콘솔입출력 객체) Java API가 제공하는 **System.in / System.out**으로 InputStream/OutputStream 사용



#2-1. (콘솔입력객체) Java API가 제공하는 **System.in** 객체로 **InputStream** 사용

byte 단위의 입출력 (InputStream/OutputStream)

1 🖱️ System.in ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

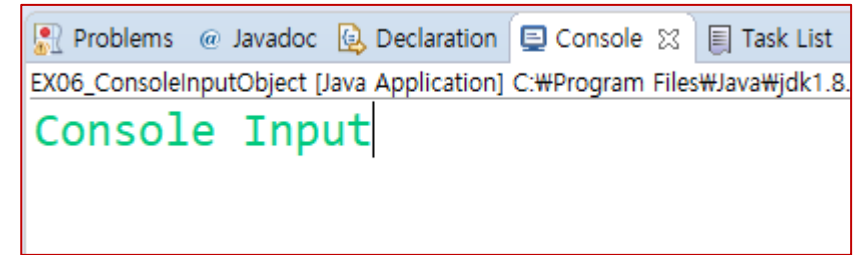
- **System.in** 콘솔 입력의 특징

2 - Console 입력이 InputStream으로 전달되는 시점 → **엔터(Enter) 입력**

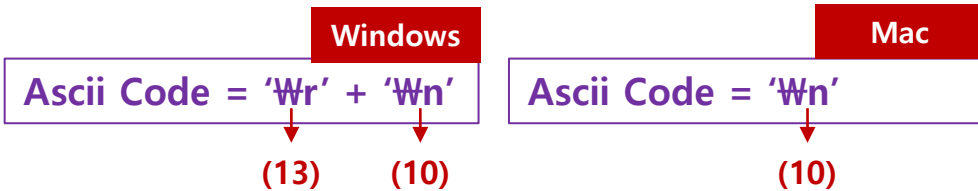
→ 즉, 한 줄 단위로만 입력처리

3 - Java API에서 콘솔 입력용으로 **하나의 객체**를 생성하여 제공

→ **close()**로 자원 해제하면 이후 콘솔입력 불가



4 - **엔터 입력시점**을 알아내는 방법 → **ASCII 코드값** 확인



예시 1

5

```
##case1. Windows
while ((data = is.read()) != 'Wr')
    System.out.println(data);
##case2. Windows
while ((data = is.read()) != 13)
    System.out.println(data);
```

여전히 버퍼에는 'Wn'이 남아 있어
연속 입력을 하는 경우
in.read()로 buffer를 비워야 함

6

예시 2

7

```
##case1. Mac
while ((data = is.read()) != 'Wn')
    System.out.println(data);
##case2. Mac
while ((data = is.read()) != 10)
    System.out.println(data);
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용

← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1

- int available(), void close()

예시

```
//InputStream 생성
InputStream is = System.in;
int data;
while((data=is.read())!=Wr) {
    System.out.println("읽은 데이터 : " + (char)data + " 남은 바이트수: " + is.available());
}
System.out.println(data); //Wr(13)
System.out.println(is.read()); //Wn(10)

//InputStream 자원반납
//is.close();
```

원도우에서 엔터(Enter)의 시작점

2

자원 반납을 하는 경우 이후에는 콘솔입력 불가
일반적으로 System.in은 자원반납 하지 않음

3

```
Hello
읽은 데이터 : H 남은 바이트수: 6
읽은 데이터 : e 남은 바이트수: 5
읽은 데이터 : l 남은 바이트수: 4
읽은 데이터 : l 남은 바이트수: 3
읽은 데이터 : o 남은 바이트수: 2
13
10
```


1 `Hello`
`Hello`

2 콘솔: **영문** → 읽기방식: **영문**(byte)

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

- 1 - **int read()**, **int read(byte[])**, **int read(byte[], int offset, int length)**

예시

콘솔: 영문 → 읽기방식: 영문(byte)

2

```
//InputStream 객체 생성
InputStream is = System.in;

//#1. 1-byte 단위 읽기
int data;
while((data=is.read())!='\r') {
    System.out.print((char)data);
}
is.read(); //'\n';
System.out.println();
```

3

Hello
Hello

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - `int read()`, `int read(byte[])`, `int read(byte[], int offset, int length)`

예시

콘솔: 영문 → 읽기방식: 영문(byte)

//InputStream 객체 생성

InputStream is = **System.in**;

2 //2-1. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)

byte[] byteArray1 = new byte[100];

int count1 = is.**read(byteArray1)**;

for (int i = 0; i < count1; i++)

System.out.print((char) byteArray1[i]);

System.out.println(" : count=" + count1);

System.out.println();

콘솔입력은 한 줄단위로 입력되기 때문에
일반적으로 반복문이 아닌 크기가 큰 배열로 입력처리

abcdef
abcdef
: count=8

3
엔터(Wr+Wn)로
한줄 개행
즉, count1-2인 경우 개행 생략

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - int read(), int read(byte[]), int read(byte[], int offset, int length)

예시

콘솔: 영문 → 읽기방식: 영문(byte)

2

```
//InputStream 객체 생성
InputStream is = System.in;

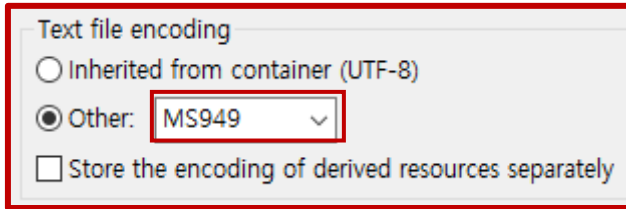
//#3-1. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)
int offset=3;
int length=5;
byte[] byteArray2 = new byte[8]; //offset+length

int count2 = is.read(byteArray2, offset, length);//offset:3 length:5
for (int i = 0; i < offset+count2; i++)
    System.out.print((char) byteArray2[i]);
System.out.println(" : count=" + count2);
```

3

Hello GoodBye!!
Hello : count=5

1 Charset.defaultCharset() → MS949



2

안녕하세요
안녕하세요

3 콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - int read(), int read(byte[]), int read(byte[], int offset, int length)

String 생성자를 이용하여 byte[]→String로 변환하기

예시

콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

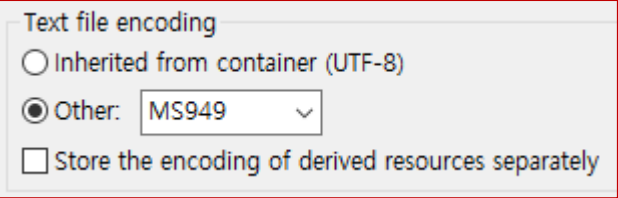
3

```
//InputStream 객체 생성  
InputStream is = System.in;
```

2

```
//#2-2. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)  
byte[] byteArray1 = new byte[100];  
int count1 = is.read(byteArray1);
```

```
String str = new String(byteArray1, 0, count1, Charset.forName("MS949"));  
System.out.println(str);
```



안녕하세요
안녕하세요

4

엔터(Wr+Wn)로
한줄 개행
즉, count1-2인 경우 개행 생략

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - int read(), int read(byte[]), int read(byte[], int offset, int length)

String 생성자를 이용하여 byte[] → String 로 변환하기

예시

콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

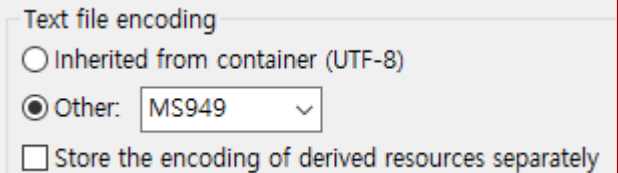
2

```
//InputStream 객체 생성
InputStream is = System.in;

//#3-2. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)
int offset=2;
int length=4;
byte[] byteArray2 = new byte[6]; //offset+length

int count2 = is.read(byteArray2, offset, length); //offset:2 length:4
String str2 = new String(byteArray2, 0, offset+count2, Charset.defaultCharset());
//String str2 = new String(byteArray2, offset, count2, Charset.defaultCharset()); //반갑
System.out.println(str2);
```

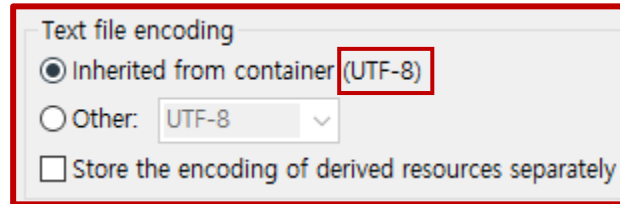
3



4

반갑습니다.
반갑

1 Charset.defaultCharset() → UTF-8



2

안녕하세요
안녕하세요

3 콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - int read(), int read(byte[]), int read(byte[], int offset, int length)

String 생성자를 이용하여 byte[]→String로 변환하기

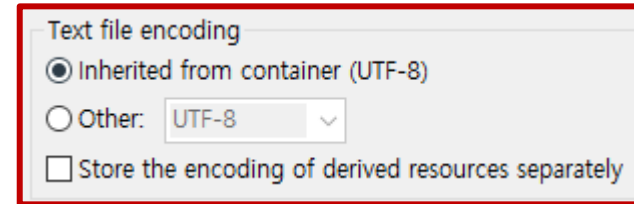
예시

콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

```
//InputStream 객체 생성
InputStream is = System.in;

//2-2. n-byte 단위 읽기 (byte[]의 처음 위치에서 부터 읽은 데이터 저장)
byte[] byteArray1 = new byte[100];
int count1 = is.read(byteArray1);

String str = new String(byteArray1, 0, count1, Charset.forName("UTF-8"));
System.out.println(str);
```



안녕하세요
안녕하세요

엔터(Wr+Wn)로
한줄 개행
즉, count1-2인 경우 개행 생략

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.in(InputStream) 객체의 활용 ← 자바 API에서 제공하는 콘솔입력을 위한 InputStream 객체

1 - int read(), int read(byte[]), int read(byte[], int offset, int length)

String 생성자를 이용하여 byte[] → String 로 변환하기

예시

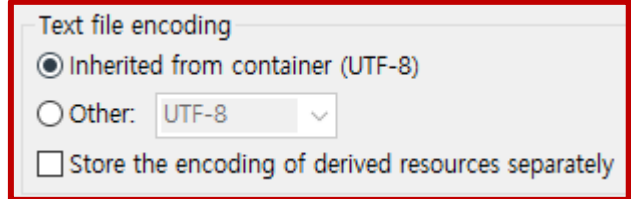
콘솔: 한글 → 읽기방식: 영문(byte) → 문자열(String) 변환

```
//InputStream 객체 생성
InputStream is = System.in;

//3-2. n-byte 단위 입력 (length만큼의 길이를 읽어와 byte[]의 offset 위치에서 부터 저장)
int offset=3;
int length=6;
byte[] byteArray2 = new byte[9]; //offset+length

int count2 = is.read(byteArray2, offset, length); //offset:2 length:4
String str2 = new String(byteArray2, 0, offset+count2, Charset.defaultCharset());
//String str2 = new String(byteArray2, offset, count2, Charset.defaultCharset()); //반갑
System.out.println(str2);
```

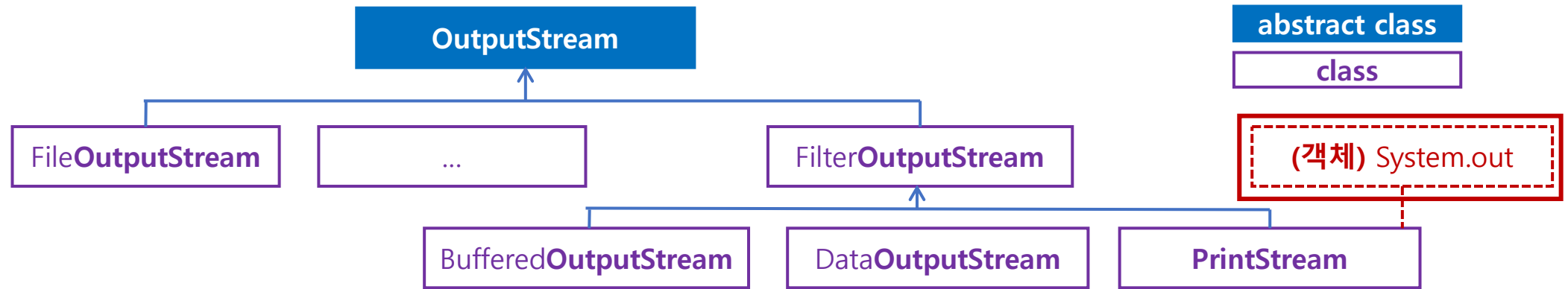
3



4

반갑습니다.
반갑

The End



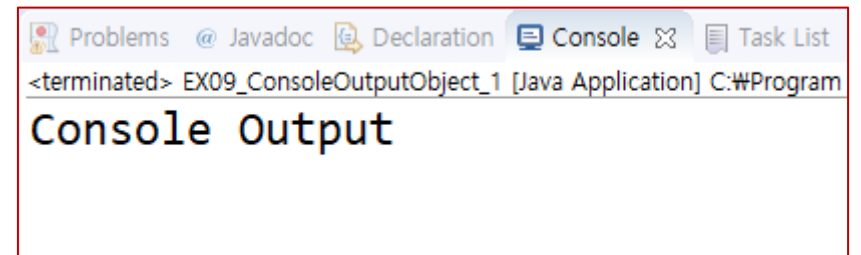
#2-2. (콘솔출력객체) Java API가 제공하는 **System.out** 객체로 **OutputStream(PrintStream)** 사용

byte 단위의 입출력 (InputStream/OutputStream)

- 1 🖱️ System.out ← 자바 API에서 제공하는 콘솔출력을 위한 OutputStream(PrintStream) 객체

- System.out 콘솔 출력의 특징

- 2 - Java API에서 콘솔 출력용으로 하나의 객체를 생성하여 제공
→ close()로 자원해제하면 이후 콘솔출력 불가
- 3 - 출력시 엔터(개행)의 표현은 Wr+Wn, Wn 모두 가능
- 4 - write() 메서드는 버퍼에 쓰기를 수행 + flush() 메서드는 버퍼의 내용을 콘솔로 출력 (반드시 flush() 사용하여야 함)



예시 1

```
///write() + flush()  
OutputStream os = System.out;  
os.write('A'); //출력내용 → 버퍼  
os.flush(); //버퍼 → 콘솔
```

출력값
동일 (A)

6

예시 2

```
///write() + flush()  
OutputStream os = System.out;  
os.write(65); //출력내용 → 버퍼  
os.flush(); //버퍼 → 콘솔
```

① 쓰기방식: **영문**(byte) → 콘솔: **영문**(byte)

②

```
JAVA
Hello!
the last
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.out ← 자바 API에서 제공하는 콘솔출력을 위한 OutputStream(PrintStream) 객체

1 - **void write(int b), void flush(), void close()**

예시

쓰기방식: **영문**(byte) → 콘솔: **영문**(byte)

```
//# OutputStream 생성(콘솔)
OutputStream os1 = System.out;

//#1. 1-byte 단위 쓰기
os1.write('J');
os1.write('A');
os1.write('V');
os1.write('A');
os1.write(13); //carriage return : \r
os1.write(10); //line feed : \n

os1.flush();
```

3

JAVA

TIP

다양한 flush()의 생략조건

- 내부적으로 버퍼를 사용하지 않는 경우
- write(int)를 사용하는 경우 write('\n')이 포함되는 경우 자동 flush();
- write(byte[], ...)의 경우 자동 flush()

➔ 굳이 구분하지 말고 write() 다음에는 flush() 를 쓰는 버릇을 들이자!!!

byte 단위의 입출력 (InputStream/OutputStream)

☞ System.out ← 자바 API에서 제공하는 콘솔출력을 위한 OutputStream(PrintStream) 객체

1 - **void write(byte[] b), void write(byte[], int offset, int length), void flush(), void close()**

예시

쓰기방식: 영문(byte) → 콘솔: 영문(byte)

```
///  
OutputStream 생성(콘솔)  
OutputStream os = System.out;
```

```
///  
#2-1. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝까지 출력)  
byte[] byteArray1 = "Hello!".getBytes();  
os.write(byteArray1);  
os.write('Wn');  
os.flush();
```

```
///  
#3-1. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)  
byte[] byteArray2 = "Better the last smile than the first laughter.".getBytes();  
os.write(byteArray2, 7, 8);  
os.flush();
```

3

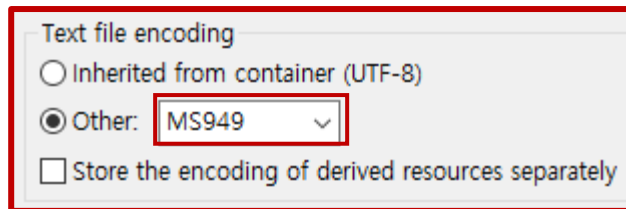
Hello!
the last

① 문자열(String) : 한글 → 쓰기방식: 영문(byte[])→ 콘솔: 한글 변환

②

안녕하세요
습니

③ Charset.defaultCharset() → MS949



byte 단위의 입출력 (InputStream/OutputStream)

☞ System.out ← 자바 API에서 제공하는 콘솔출력을 위한 OutputStream(PrintStream) 객체

1 - **void write(byte[] b), void write(byte[], int offset, int length), void flush(), void close()**

String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

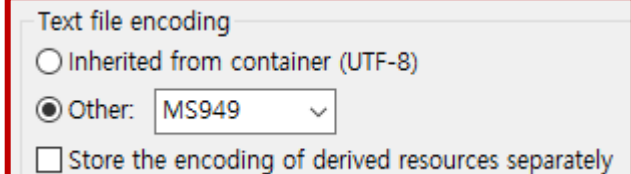
문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 콘솔: 한글 변환

```
## OutputStream 생성(콘솔)
OutputStream os = System.out;

##2-2. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝가지를 출력)
byte[] byteArray1 = "안녕하세요".getBytes(Charset.forName("MS949"));
os.write(byteArray1);
os.write('\n');
os.flush();

##3-2. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)
byte[] byteArray2 = "반갑습니다".getBytes(Charset.defaultCharset());
os.write(byteArray2, 4, 4);
os.flush();
```

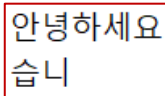
2



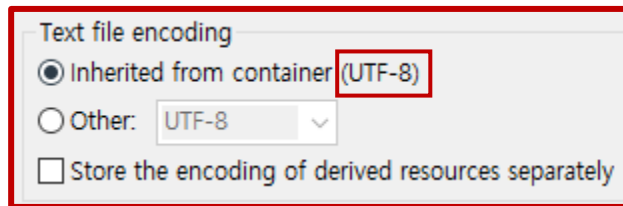
3

안녕하세요
습니

① 문자열(String) : 한글 → 쓰기방식: 영문(byte[])→ 콘솔: 한글 변환

② 

③ Charset.defaultCharset() → UTF-8



byte 단위의 입출력 (InputStream/OutputStream)

☞ System.out ← 자바 API에서 제공하는 콘솔출력을 위한 OutputStream(PrintStream) 객체

1 - **void write(byte[] b), void write(byte[], int offset, int length), void flush(), void close()**

String의 `getBytes(Charset c)` 메서드를 사용하여 String → byte[]로 변환하기

예시

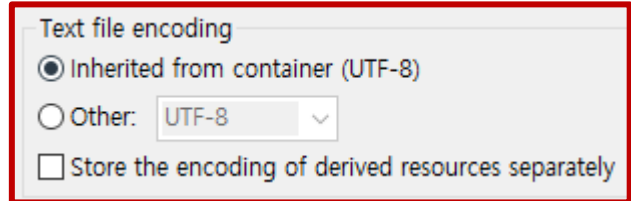
문자열(String) : 한글 → 쓰기방식: 영문(byte[]) → 콘솔: 한글 변환

```
//# OutputStream 생성(콘솔)
OutputStream os = System.out;

//#2-2. n-byte 단위 쓰기 (byte[]의 처음 위치에서 부터 끝가지를 출력)
byte[] byteArray1 = "안녕하세요".getBytes(Charset.forName("UTF-8"));
os.write(byteArray1);
os.write('\n');
os.flush();

//#3-2. n-byte 단위 쓰기 (byte[]의 offset 위치에서부터 length개수를 읽어와 출력)
byte[] byteArray2 = "반갑습니다".getBytes(Charset.defaultCharset());
os.write(byteArray2, 6, 6);
os.flush();
```

2



3

안녕하세요
습니

The End

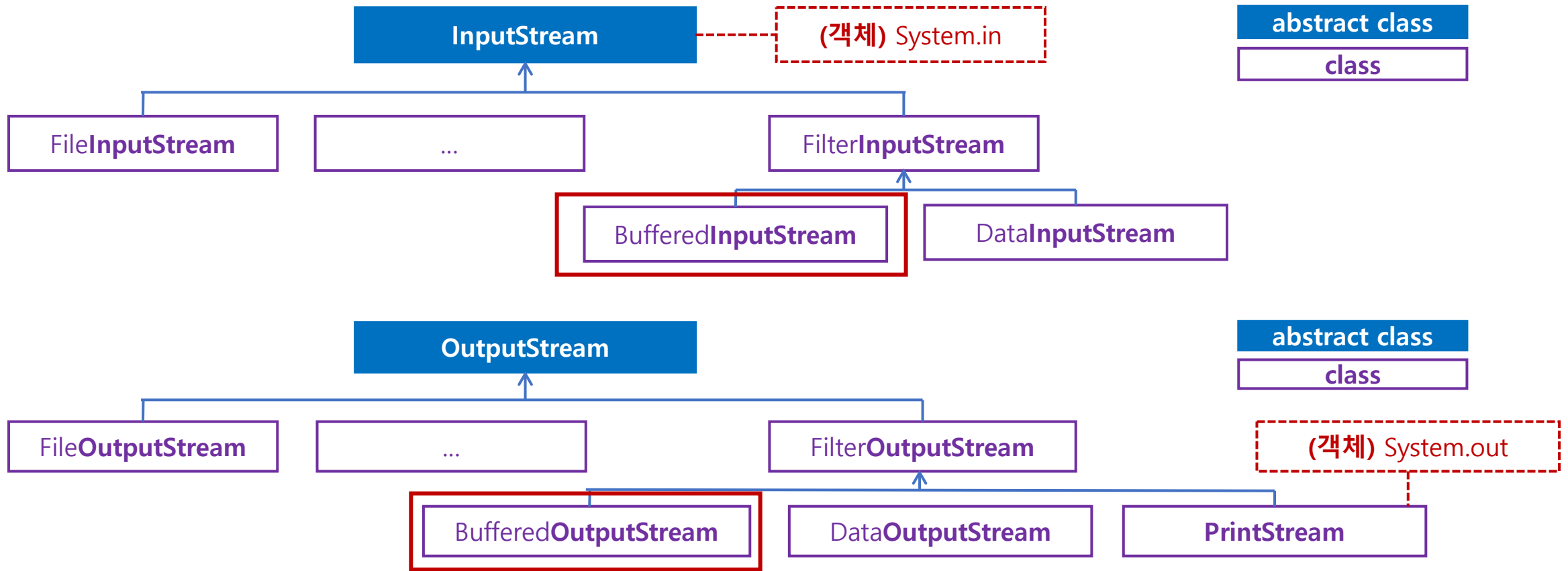
#3. 입출력의 Filtering (FilterInputStream/FilterOutputStream)

byte 단위의 입출력 (InputStream/OutputStream)

👉 FiterStream의 개념

← 입출력 스트림의 형태 특징의 변경





#3. 입출력의 Filtering (FilterInputStream/FilterOutputStream)

#3-1. **BufferedInputStream/BufferedOutputStream**을 이용한 속도 향상

byte 단위의 입출력 (InputStream/OutputStream)

☞ BufferedInputStream/BufferedOutputStream ← ① 입출력과정에서 메모리 버퍼를 사용함으로써 속도 향상

② - BufferedInputStream 생성자

```
BufferedInputStream(InputStream in)  
BufferedInputStream(InputStream in, int size)
```

Default Buffer Size 사용

예시

```
//# BufferedInputStream 객체 생성  
File orgfile = new File("src/pack02_javaio/sec04_files/mycat_origin.jpg");  
InputStream is = new FileInputStream(orgfile);  
BufferedInputStream bis = new BufferedInputStream(is);
```

직접 Buffer Size 지정

③ - BufferedOutputStream 생성자

```
BufferedOutputStream(OutputStream in)  
BufferedOutputStream(OutputStream in, int size)
```

예시

```
//# BufferedOutputStream 객체 생성  
File outfile = new File("src/pack02_javaio/sec04_files/mycat_origin.jpg");  
OutputStream os = new FileOutputStream(outfile);  
BufferedOutputStream bos = new BufferedOutputStream(os);
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ BufferedInputStream/BufferedOutputStream ← 입출력과정에서 메모리 버퍼를 사용함으로써 **속도 향상**

1 - Buffered(InputStream/OutputStream) : 파일 복사하기

예시

```
//#파일 생성
File orgfile = new File("src/pack02_javaio/sec04_files/mycat_origin.jpg");
File copyfile1 = new File("src/pack02_javaio/sec04_files/mycat_copy1.jpg");
File copyfile2 = new File("src/pack02_javaio/sec04_files/mycat_copy2.jpg");
```

//#Buffered(InputStream/OutputStream)을 사용하지 않은 경우

```
long start, end, time1, time2;
start = System.nanoTime();
try(InputStream is = new FileInputStream(orgfile);
    OutputStream os = new FileOutputStream(copyfile1));{
```

```
    int data;
    while((data = is.read())!=-1) { os.write(data); }
    os.flush();
```

```
}catch(IOException e) {}
end = System.nanoTime();
time1 = end-start;
System.out.println("Without BufferedXXXStream : " + (time1));
```

mycat_origin.jpg



Without BufferedXXXStream : 2324488500
With BufferedXXXStream : 13148700
Ratio of with and without : 176

mycat_copy1.jpg



byte 단위의 입출력 (InputStream/OutputStream)

☞ BufferedInputStream/BufferedOutputStream ← 입출력과정에서 메모리 버퍼를 사용함으로써 속도 향상

- Buffered(Input/Output)Stream : 파일 복사하기

예시

//#Buffered(Input/Output)Stream을 사용한 경우

start = System.nanoTime();

try(InputStream is = new FileInputStream(orgfile);

BufferedInputStream bis = new BufferedInputStream(is);

OutputStream os = new FileOutputStream(copyfile2);

BufferedOutputStream bos = new BufferedOutputStream(os);{

int data;

while((data = **bis.read()**)!=-1) { **bos.write(data)**; }

bos.flush();

}catch(IOException e) {}

end = System.nanoTime();

time2 = end-start;

System.out.println("With BufferedXXXStream : " + (time2));

//사용한 경우와 사용하지 않은 경우의 비

System.out.println("Ratio of with and without : " + (**time1/time2**));

mycat_origin.jpg

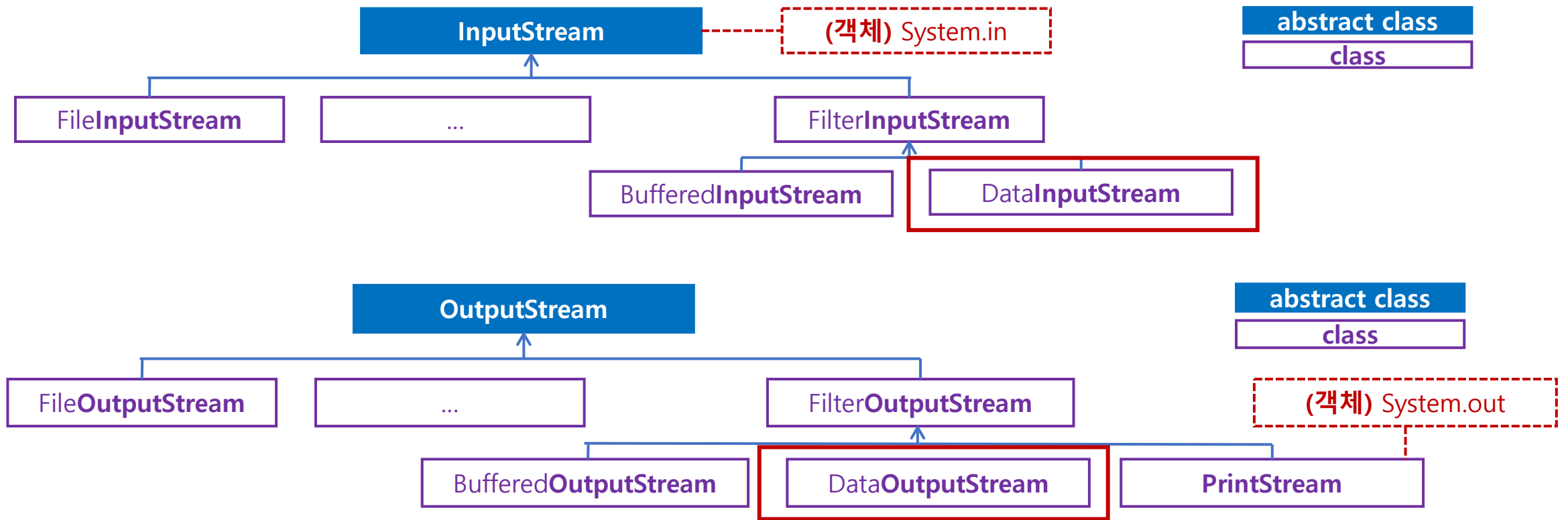


4

Without BufferedXXXStream : 2324488500
With BufferedXXXStream : 13148700
Ratio of with and without : 176

mycat_copy2.jpg





#3. 입출력의 Filtering (`FilterInputStream`/`FilterOutputStream`)
#3-2. **`DataInputStream`/`DataOutputStream`**을 이용한 데이터 타입 다양화

byte 단위의 입출력 (InputStream/OutputStream)

☞ DataInputStream/DataOutputStream

- DataInputStream 생성자

```
DataInputStream(InputStream in)
```

2

예시

```
//# DataInputStream 객체 생성  
File datafile = new File("src/pack02_javaio/sec04_files/file1.data");  
InputStream is = new FileInputStream(datafile);  
DataInputStream dis = new DataInputStream(is);
```

1 입출력과정에서 다양한

데이터타입(int, long, float, double, UTF8 ...)으로
입력 및 출력 수행

- DataOutputStream 생성자

```
DataOutputStream(OutputStream in)
```

3

예시

```
//# DataOutputStream 객체 생성  
File datafile = new File("src/pack02_javaio/sec04_files/file1.data");  
OutputStream os = new FileOutputStream(datafile);  
DataOutputStream dos = new DataOutputStream(os);
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ DataInputStream/DataOutputStream

입출력과정에서 다양한

데이터타입(int, long, float, double, UTF8 ...)으로
입력 및 출력 수행

1

- DataInputStream 메서드

```
int    read(byte[] b)
int    read(byte[] b, int off, int len)
```

```
boolean readBoolean()
byte    readByte()
char    readChar()
short   readShort()
int     readInt()
long    readLong()
float   readFloat()
double  readDouble()
```

```
String readUTF()
```

2

- DataOutputStream 메서드

```
void    write(int b)
void    write(byte[] b, int off, int len)
```

```
void    writeBoolean(boolean v)
void    writeByte(int v)
void    writeChar(int v)
void    writeShort(int v)
void    writeInt(int v)
void    writeLong(long v)
void    writeFloat(float v)
void    writeDouble(double v)
```

```
void    writeUTF(String str)
void    writeBytes(String s)
```

추가메서드

byte 단위의 입출력 (InputStream/OutputStream)

☞ DataInputStream/DataOutputStream



입출력과정에서 다양한
데이터타입(int, long, float, double, UTF8 ...)으로
입력 및 출력 수행

- Data(Input/Output)Stream : 다양한 타입으로 입력 및 출력

예시

//파일 생성

```
File dataFile = new File("src/pack02_javaio/sec04_files/file1.data");  
if(!dataFile.exists()) dataFile.createNewFile();
```

//데이터 쓰기 (FilterStream = DataOutputStream)

```
try(OutputStream os = new FileOutputStream(dataFile);  
    DataOutputStream dos = new DataOutputStream(os);) {
```

```
    dos.writeInt(35);
```

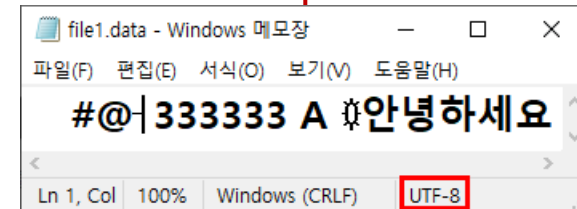
```
    dos.writeDouble(5.8);
```

```
    dos.writeChar('A');
```

```
    dos.writeUTF("안녕하세요"); //디폴트 Charset과 관계없이 항상 UTF8로 생성  
    dos.flush();
```

```
}catch(IOException e) {}
```

4



3

1

2

byte 단위의 입출력 (InputStream/OutputStream)

☞ DataInputStream/DataOutputStream

입출력과정에서 다양한
데이터타입(int, long, float, double, UTF8 ...)으로
입력 및 출력 수행

- Data(InputStream/OutputStream) : 다양한 타입으로 입력 및 출력

예시

//파일 생성

```
File dataFile = new File("src/pack02_javaio/sec04_files/file1.data");
```

//데이터 읽기 (FilterStream = DataInputStream)

```
try(InputStream is = new FileInputStream(dataFile);  
    DataInputStream dis = new DataInputStream(is)){
```

```
    System.out.println(dis.readInt()); //-> 35
```

```
    System.out.println(dis.readDouble()); //-> 5.8
```

```
    System.out.println(dis.readChar()); //-> A
```

```
    System.out.println(dis.readUTF()); //-> 안녕하세요
```

```
}catch(IOException e) {}
```

1

2

3

35
5.8
A
안녕하세요

The End

#3. 입출력의 Filtering (FilterInputStream/FilterOutputStream)

#3-3. 필터조합 **CombineFilters** :

Buffered(Input/Output)Stream + Data(Input/Output) Stream

byte 단위의 입출력 (InputStream/OutputStream)

2

☞ FilterStream의 연결 가능 (BufferedXXXStream+DataXXXStream)

← 향상된 속도로
다양한 타입의 입출력 수행

1 - Buffered(Input/Output)Stream + Data(Input/Output)Stream

이외에도 다양한 조합이 가능

예시

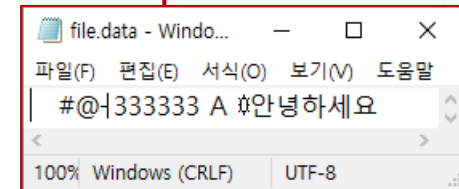
```
//파일 생성
File dataFile = new File("src/pack02_javaio/sec04_files/file2.data");

//데이터 쓰기 (FilterStream = BufferedOutputStream + DataOutputStream)
try(OutputStream os = new FileOutputStream(dataFile);
    BufferedOutputStream bos = new BufferedOutputStream(os);
    DataOutputStream dos = new DataOutputStream(bos);) {

    dos.writeInt(35);
    dos.writeDouble(5.8);
    dos.writeChar('A');
    dos.writeUTF("안녕하세요");
    dos.flush();

} catch(IOException e) {}
```

4



byte 단위의 입출력 (InputStream/OutputStream)

☞ FilterStream의 연결 가능 (BufferedXXXStream+DataXXXStream)

← 향상된 속도로
다양한 타입의 입출력 수행

- **Buffered(Input/Output)Stream + Data(Input/Output)Stream**

이외에도 다양한 조합이 가능

예시

//파일 생성

```
File dataFile = new File("src/pack02_javaio/sec04_files/file2.data");  
if(!dataFile.exists()) dataFile.createNewFile();
```

//데이터 읽기 (FilterStream = BufferedInputStream + DataInputStream)

```
try(InputStream is = new FileInputStream(dataFile);
```

```
    BufferedInputStream bis = new BufferedInputStream(is);
```

```
    DataInputStream dis = new DataInputStream(bis);){
```

```
        System.out.println(dis.readInt()); //-> 35
```

```
        System.out.println(dis.readDouble()); //-> 5.8
```

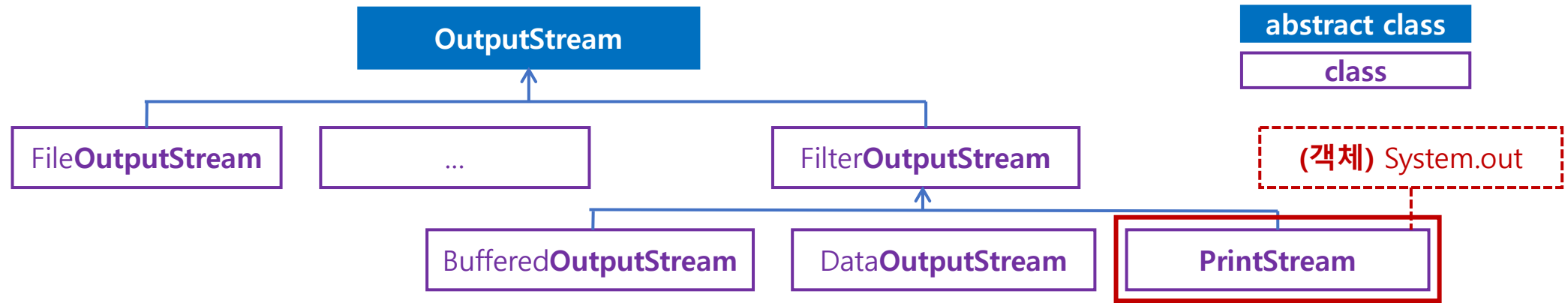
```
        System.out.println(dis.readChar()); //-> A
```

```
        System.out.println(dis.readUTF()); //-> 안녕하세요
```

```
}catch(IOException e) {}
```

3

35
5.8
A
안녕하세요



#3. 입출력의 Filtering (FilterInputStream/FilterOutputStream)
#3-4. 다양한 출력에 특화된 PrintStream

byte 단위의 입출력 (InputStream/OutputStream)

☞ PrintStream

1

다양한 타입의 출력에 특화된 클래스
자동 flush()기능 제공 (flush() 필요없음)
System.out의 객체 타입

- PrintStream의 생성자

CASE#1. 출력할 파일을 매개변수로 직접 받는 경우

2

```
PrintStream(File file)
PrintStream(String fileName)
```

3



다른 FilterStream은 InputStream 또는
OutputStream을 생성자 매개변수로 받음

CASE#2. OutputStream을 매개변수로 받는 경우

4

```
PrintStream(OutputStream out)
PrintStream(OutputStream out, boolean autoFlush)
```

byte 단위의 입출력 (InputStream/OutputStream)

☞ PrintStream

다양한 타입의 출력에 특화된 클래스
자동 flush()기능 제공 (flush() 필요없음)
System.out의 객체 타입

- PrintStream의 대표적 메서드

1

void println()

2

```
void print(boolean b)
void print(char c)
void print(int i)
void print(long l)
void print(float f)
void print(double d)
void print(String s)

void print(Object obj)
```

3

```
void println(boolean b)
void println(char c)
void println(int i)
void println(long l)
void println(float f)
void println(double d)
void println(String s)

void println(Object obj)
```

4

연속호출 가능

→ PrintStream printf(String format, Object... args)

byte 단위의 입출력 (InputStream/OutputStream)

👉 PrintStream

다양한 타입의 출력에 특화된 클래스
자동 flush()기능 제공 (flush() 필요없음)
System.out의 객체 타입

- PrintStream : 다양한 타입으로 File/콘솔 출력

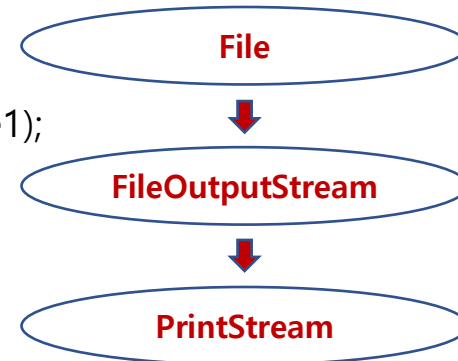
예시

1

```
//#File 객체 생성 및 OutputStream 객체 생성 (PrintStream)
File outFile1 = new File("src/pack02_javaio/sec04_files/PrintStream1.txt");
File outFile2 = new File("src/pack02_javaio/sec04_files/PrintStream2.txt");
```

2

```
//#1. PrintStream(FileOutputStream(File))
try(OutputStream os1 = new FileOutputStream(outFile1);
    PrintStream ps = new PrintStream(os1)){
    ps.println(5.8);
    ps.print(3 + " 안녕 " + 12345 + "\n");
    ps.printf("%d ", 7).printf("%s %f", "안녕", 5.8);
    ps.println();
} catch(IOException e) { }
```



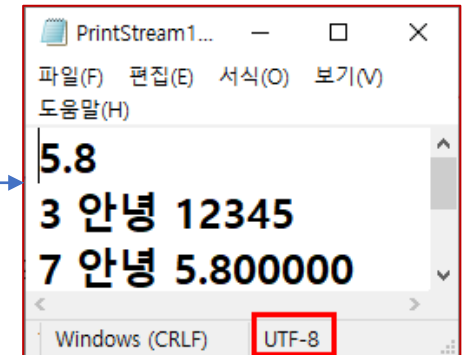
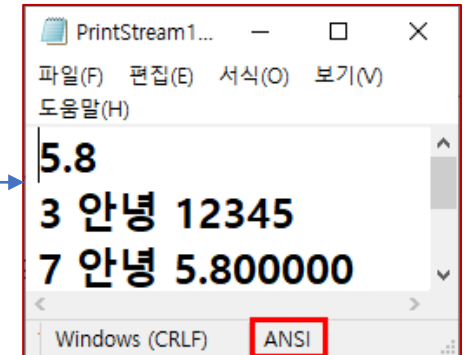
Default
Charset

MS949

3

Default
Charset

UTF-8



byte 단위의 입출력 (InputStream/OutputStream)

👉 PrintStream

다양한 타입의 출력에 특화된 클래스
자동 flush()기능 제공 (flush() 필요없음)
System.out의 객체 타입

- PrintStream : 다양한 타입으로 File/콘솔 출력

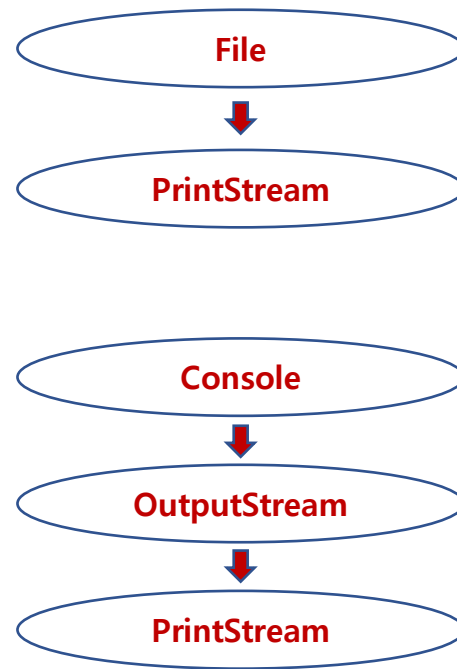
예시

1

```
//#2. PrintStream(File)
try(PrintStream ps = new PrintStream(outFile2)){
    ps.println(5.8);
    ps.print(3 + " 안녕 " + 12345 + " \n");
    ps.printf("%d ", 7).printf("%s %f", "안녕", 5.8);
    ps.println();
}catch(IOException e) { }
```

2

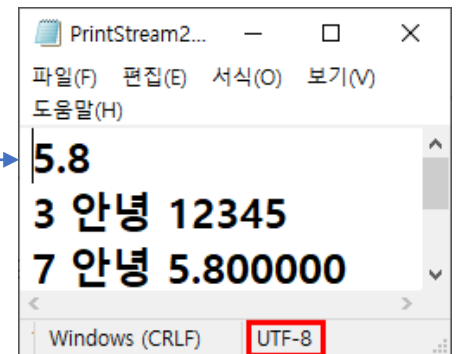
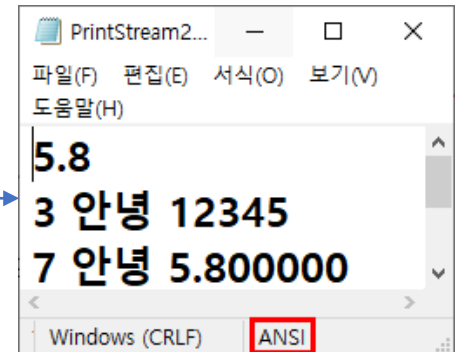
```
//#3. PrintStream = System.out
try(OutputStream os2 = System.out;
    PrintStream ps = new PrintStream(os2)){
    ps.println(5.8);
    ps.print(3 + " 안녕 " + 12345 + " \n");
    ps.printf("%d ", 7).printf("%s %f", "안녕", 5.8);
    ps.println();
}catch(IOException e) { }
```



Default
Charset
MS949

3

Default
Charset
UTF-8



4

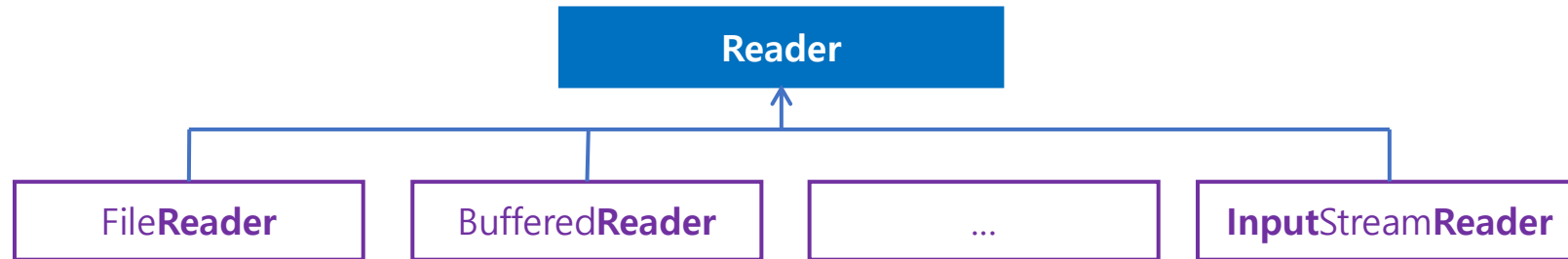
5.8
3 안녕 12345
7 안녕 5.800000

The End

char 단위의 입출력 (Reader/Writer)

char 단위의 입출력 (Reader/Writer)

1 🖱 Reader ← **char** 단위 **입력**을 수행하는 추상클래스



abstract class

class

2 🖱 Writer ← **char** 단위 **출력**을 **수행**하는 추상클래스



abstract class

class

char 단위의 입출력 (Reader/Writer)

☞ Reader

← char 단위 입력을 수행하는 추상클래스

1

Reader의 주요 메서드

내부에서
이 메서드
호출

int skip(long n)	n개의 char 스킵하기 (실제 스킵된 char 수 리턴)
int read()	int(4byte)의 하위 2byte에 읽은 데이터를 저장하여 리턴
int read(char[] cbuf)	읽은 데이터를 char[] cbuf에 저장하며 읽은 char수를 리턴
abstract int read(char[] cbuf, int off, int len)	length 개수만큼 읽은 데이터를 char[] cbuf의 offset 위치부터 저장 (추상 메서드)
abstract void close()	Reader의 자원 반환

2

Q1

read(...) 메서드 하나만 추상 메서드이니까 껍데기만 아래처럼 overriding한 MyReader 클래스를 생성하고 나머지 완성된 메서드만(read(), read(char[] cbuf)을 가지고 읽으면 될까? **NO!!**

```
class MyReader extends Reader{
    @override
    int read(char[] cbuf, int off, int len){ }
}
```

다른 read(..) 메서드에서 추상메서드 read(...)를 내부적으로 사용
속도를 위해 read(...)메서드는 **JNI를 이용하여 오버라이딩**

3

native int read(...) {...}

편의를 위해 이미 JNI로 read(...)를 오버라이딩 한 하위클래스를
사용하여 Reader객체 생성

char 단위의 입출력 (Reader/Writer)

☞ Writer

← char 단위 출력을 수행하는 추상클래스

1

Writer의 주요 메서드

내부에서
이 메서드
호출

abstract void flush()	메모리 버퍼에 저장된 데이터 내보내기 (실제 출력 수행) (추상 메서드)
void write(int c)	int(4byte)의 하위 2byte 를 메모리버퍼에 출력
void write(char[] cbuf)	매개변수로 넘겨진 char[] cbuf 데이터를 메모리버퍼에 출력
void write(String str)	매개변수로 넘겨진 String 값을 메모리버퍼에 출력
void write(String str, int off, int len)	str의 offset 위치에서부터 length개수를 읽어와 메모리버퍼에 출력
abstract void write(char[] cbuf, int off, int len)	char[]의 offset 위치에서부터 length개수를 읽어와 출력 (추상 메서드)
abstract void close()	Writer의 자원 반환 (추상 메서드)

2

Q1

write(...) 메서드 하나만 추상 메서드이니까 겹데기만 아래처럼 overriding한 MyWriter 클래스를 생성하고 나머지 완성된 메서드만(write(char c), write(char[] b), write (String str) 등을 가지고 읽으면 될까? **NO!!**

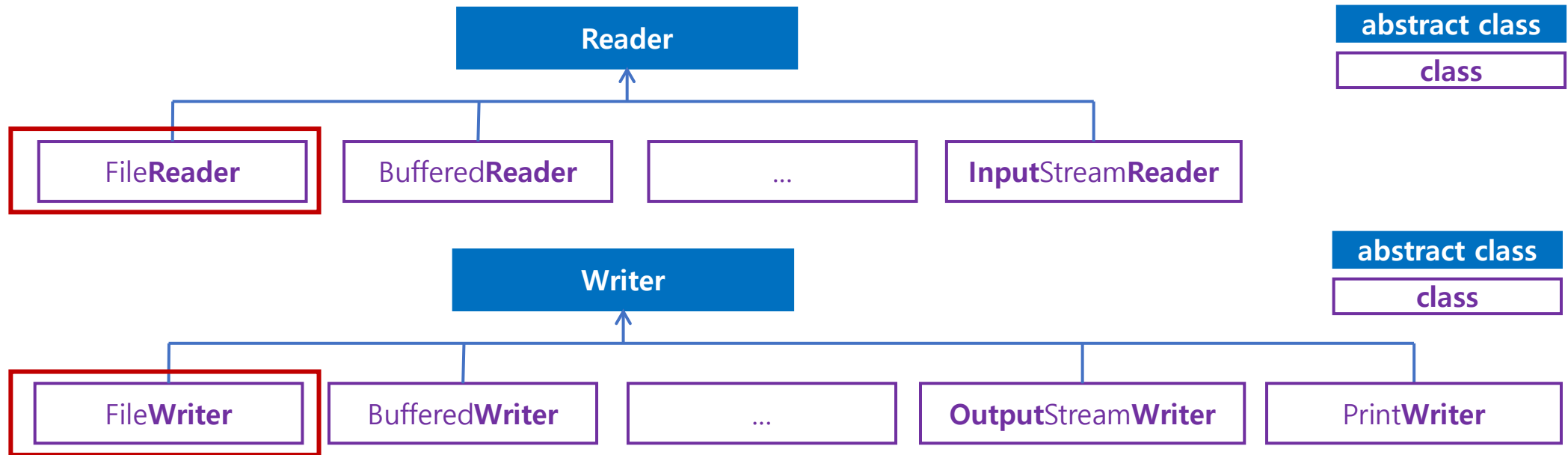
```
class MyWriter extends Writer {  
    @override  
    void write(...) { }  
}
```

다른 write(..) 메서드에서 추상메서드 write(...)를 내부적으로 사용
속도를 위해 abstract write(...)메서드는 **JNI를 이용하여 오버라이딩**

3

native void write(...) {...}

편의를 위해 이미 JNI로 write(...)를 오버라이딩 한 하위클래스를
사용하여 Writer 객체 생성



#1. **FileReader/FileWriter**로 Reader/Writer 객체 생성

char 단위의 입출력 (Reader/Writer)

1 FileReader/FileWriter ← char 단위로 File 입출력 수행

- FileReader 생성자

2

```
FileReader(File file)  
FileReader(String fileName)
```

예시

```
///  
File readerwriterFile = new File("src/pack02_javaio/sec05_files/ReaderWriterFile.txt");  
Reader reader = new FileReader(readerwriterFile);
```

- FileWriter 생성자

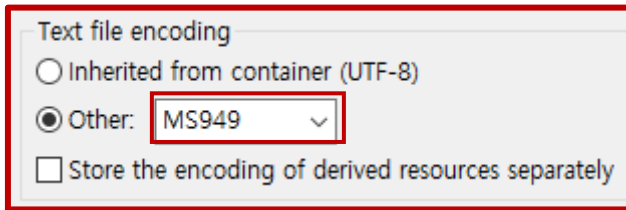
3

```
FileWriter(File file)  
FileWriter(File file, boolean append)  
FileWriter(String fileName)  
FileWriter(String fileName, boolean append)
```

예시

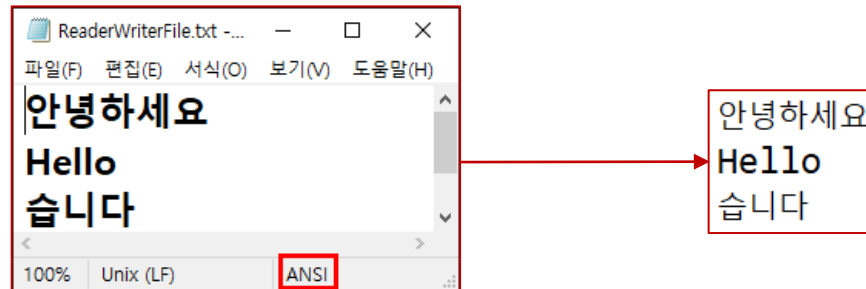
```
///  
File readerwriterFile = new File("src/pack02_javaio/sec05_files/ReaderWriterFile.txt");  
Writer writer = new FileWriter(readerwriterFile);
```


1 Charset.defaultCharset() → MS949



2 ANSI 모드 텍스트 파일 → FileWriter/FileReader → char 단위 텍스트 읽기

3



char 단위의 입출력 (Reader/Writer)

👉 FileReader/FileWriter

← char 단위로 File 입출력 수행

- **FileWriter**로 파일쓰기 + **FileReader**로 데이터 읽기

예시

//#.파일 객체 선언

```
File readerwriterFile = new File("src/pack02_javaio/sec05_files/ReaderWriterFile.txt");
```

1 //#1. FileWriter를 이용한 파일 쓰기 (MS949 모드)

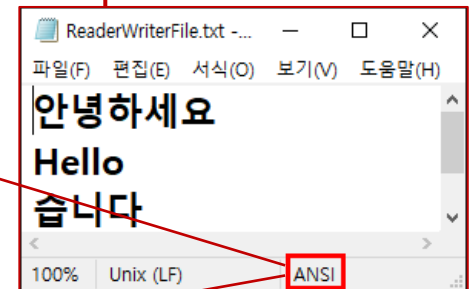
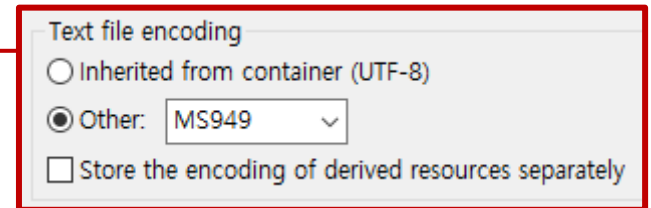
```
try(Writer writer = new FileWriter(readerwriterFile)){  
    writer.write("안녕하세요\n".toCharArray());  
    writer.write("Hello");  
    writer.write('\n');  
    writer.write('\n');  
    writer.write("반갑습니다", 2, 3);  
    writer.flush();  
}
```

String → char[]

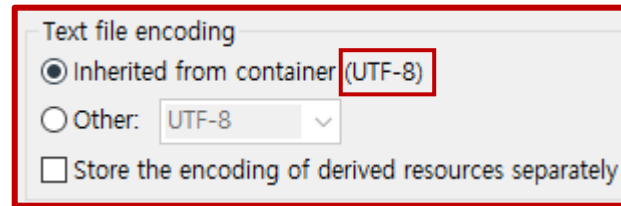
```
catch(IOException e) {}
```

3 //#2. FileReader를 이용한 파일 읽기 (MS949 모드)

```
try(Reader reader = new FileReader(readerwriterFile)){  
    int data;  
    while((data=reader.read())!=-1) {  
        System.out.print((char)data);  
    }  
}  
catch(IOException e) {}
```

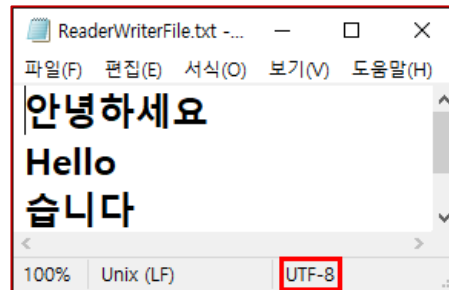


1 Charset.defaultCharset() → UTF-8



2 UTF-8 모드 텍스트 파일 → FileWriter/FileReader → char 단위 텍스트 읽기

3



안녕하세요
Hello
습니다

char 단위의 입출력 (Reader/Writer)

👉 FileReader/FileWriter

← char 단위로 File 입출력 수행

- **FileWriter**로 파일쓰기 + **FileReader**로 데이터 읽기

예시

//#.파일 객체 선언

```
File readerwriterFile = new File("src/pack02_javaio/sec05_files/ReaderWriterFile.txt");
```

1 //#1. **FileWriter**를 이용한 파일 쓰기 (**UTF-8 모드**)

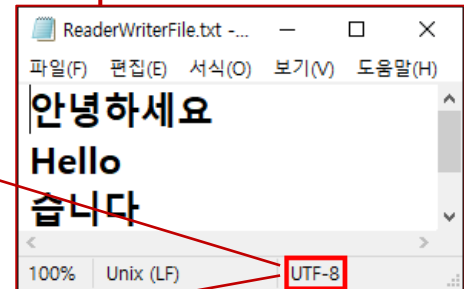
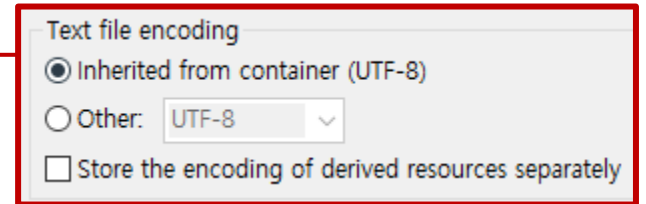
```
try(Writer writer = new FileWriter(readerwriterFile)){  
    writer.write("안녕하세요\n".toCharArray());  
    writer.write("Hello");  
    writer.write('\r');  
    writer.write('\n');  
    writer.write("반갑습니다", 2, 3);  
    writer.flush();  
}
```

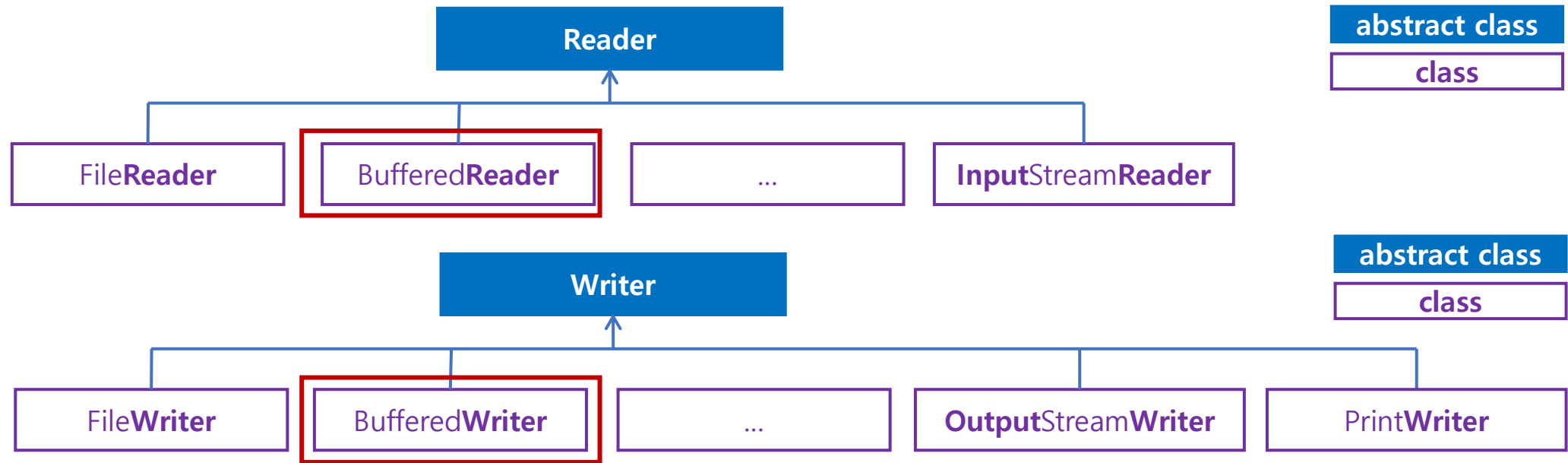
String → char[]

```
catch(IOException e) {}
```

3 //#2. **FileReader**를 이용한 파일 읽기 (**UTF-8 모드**)

```
try(Reader reader = new FileReader(readerwriterFile)){  
    int data;  
    while((data=reader.read())!=-1) {  
        System.out.print((char)data);  
    }  
}  
catch(IOException e) {}
```





#2. **BufferedReader/BufferedWriter**로 속도 개선

char 단위의 입출력 (Reader/Writer)

4

BufferedReader의 추가 메서드

String readLine()

한 줄씩 읽어 String으로 리턴

👉 BufferedReader/BufferedWriter

1

입출력과정에서 메모리 버퍼를
사용함으로써 속도 향상

- BufferedReader 생성자

2

```
BufferedReader(Reader in)  
BufferedReader(Reader in, int size)
```

Default Buffer Size 사용

예시

```
///  
File BuffredXXXFile = new File("src/pack02_javaio/sec05_files/ BuffredXXXFile.txt");  
Reader reader = new FileReader(BuffredXXXFile);  
BufferedReader br = new BufferedReader(reader);
```

- BufferedWriter 생성자

3

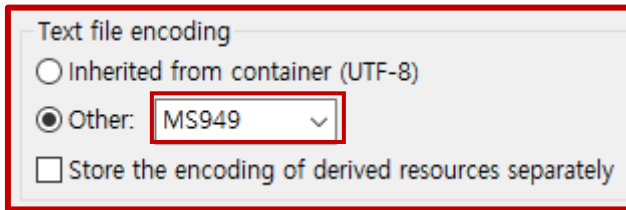
```
BufferedWriter(Writer out)  
BufferedWriter(Writer out, int size)
```

예시

```
///  
File BuffredXXXFile = new File("src/pack02_javaio/sec05_files/BuffredXXXFile.txt");  
Writer writer = new FileWriter(BuffredXXXFile);  
BufferedWriter bw = new BufferedWriter(writer);
```

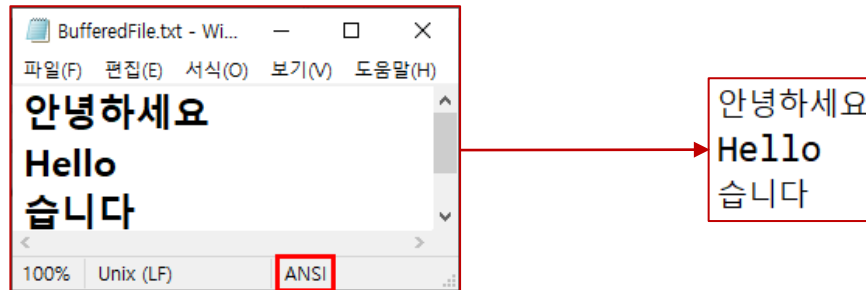
직접 Buffer Size 지정

1 Charset.defaultCharset() → MS949



2 MS949 모드 텍스트 파일 → FileWriter/FileReader → BufferedWriter/BufferedReader → char 단위 텍스트 읽기

3



char 단위의 입출력 (Reader/Writer)

👉 BufferedReader/BufferedWriter

← 입출력과정에서 메모리 버퍼를
사용함으로써 **속도 향상**

- BufferedWriter/BufferedReader를 활용한 쓰기 읽기

예시

//#.파일 객체 선언

```
File buffredFile = new File("src/pack02_javaio/sec05_files/BufredFile.txt");
```

1

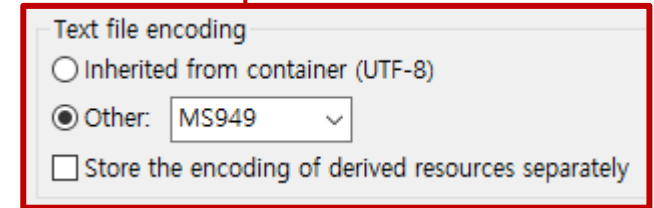
//#1. FileWriter/BufferedWriter를 이용한 파일 쓰기 (**MS949** 모드)

```
try(Writer writer = new FileWriter(buffredFile);  
    BufferedWriter bw = new BufferedWriter(writer);{  
    bw.write("안녕하세요\n".toCharArray());  
    bw.write("Hello");  
    bw.write('\n');  
    bw.write("반갑습니다", 2, 3);  
    bw.flush();  
} catch(IOException e) {}
```

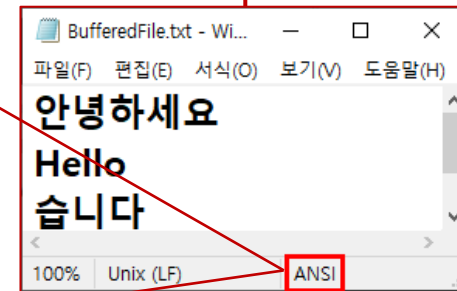
3

//#2. FileReader/BufferedReader를 이용한 파일 읽기 (**MS949** 모드)

```
try(Reader reader = new FileReader(buffredFile);  
    BufferedReader br = new BufferedReader(reader);{  
    String data;  
    while((data=br.readLine())!=null) {  
        System.out.println(data);  
    }  
} catch(IOException e) {}
```



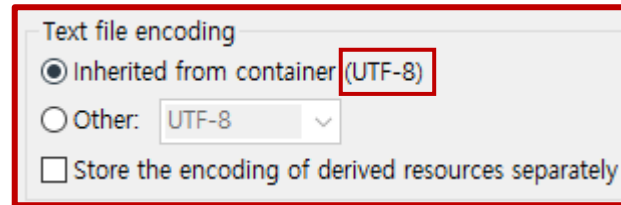
2



안녕하세요
Hello
습니다

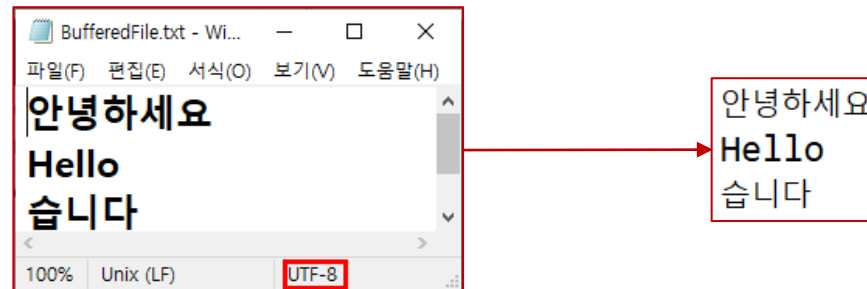
4

① Charset.defaultCharset() → UTF-8



② UTF-8 모드 텍스트 파일 → FileWriter/FileReader → BufferedWriter/BufferedReader → char 단위 텍스트 읽기

③



char 단위의 입출력 (Reader/Writer)

☞ BufferedReader/BufferedWriter

← 입출력과정에서 메모리 버퍼를
사용함으로써 **속도 향상**

- BufferedWriter/BufferedReader를 활용한 쓰기 읽기

예시

//#.파일 객체 선언

```
File buffredFile = new File("src/pack02_javaio/sec05_files/BuffredFile.txt");
```

1

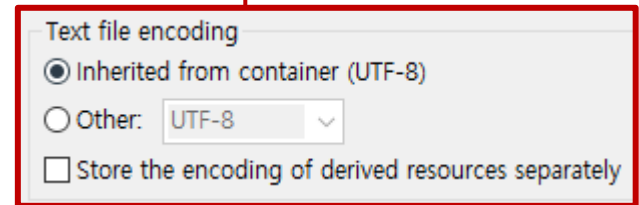
//#1. FileWriter/BufferedWriter를 이용한 파일 쓰기 (**UTF-8** 모드)

```
try(Writer writer = new FileWriter(buffredFile);  
    BufferedWriter bw = new BufferedWriter(writer);{  
    bw.write("안녕하세요\n".toCharArray());  
    bw.write("Hello");  
    bw.write('\n');  
    bw.write("반갑습니다", 2, 3);  
    bw.flush();  
} catch(IOException e) {}
```

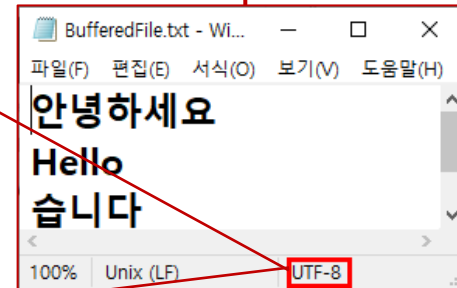
3

//#2. FileReader/BufferedReader를 이용한 파일 읽기 (**UTF-8** 모드)

```
try(Reader reader = new FileReader(buffredFile);  
    BufferedReader br = new BufferedReader(reader);{  
    String data;  
    while((data=br.readLine())!=null) {  
        System.out.println(data);  
    }  
} catch(IOException e) {}
```



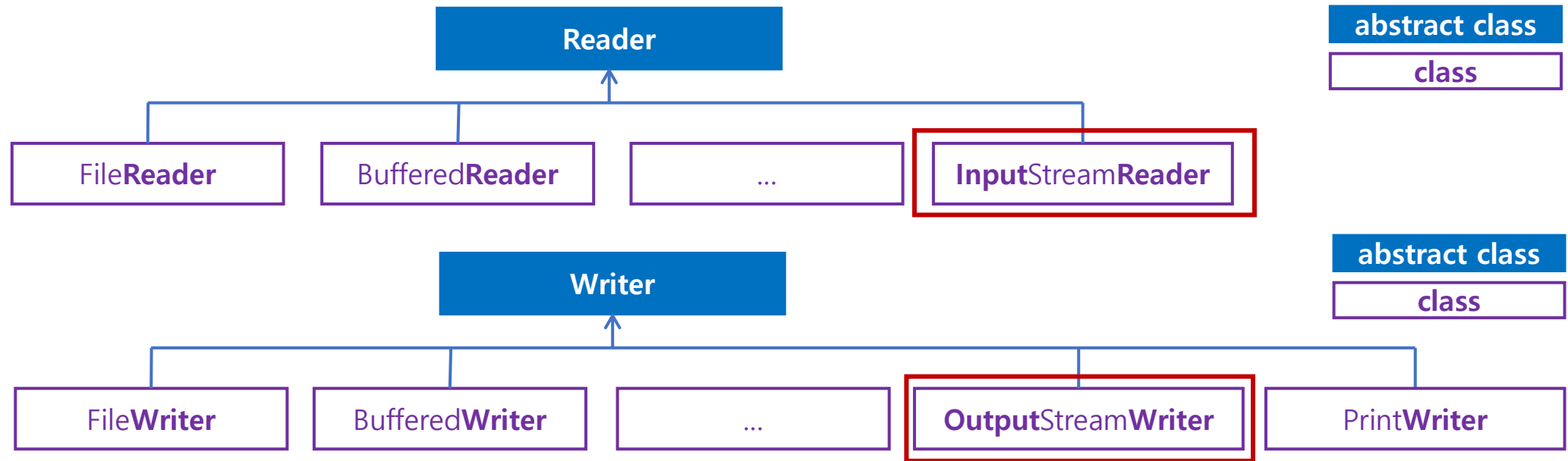
2



안녕하세요
Hello
습니다

4

The End



#3. **InputStreamReader/OutputStreamWriter**로 Reader/Writer 객체 생성

char 단위의 입출력 (Reader/Writer)

1

👉 InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

- InputStreamReader 생성자

Charset에 따라 문자를 읽어드려 char형태로 저장

2

```
InputStreamReader(InputStream in)
InputStreamReader(InputStream in, Charset cs)
InputStreamReader(InputStream in, String charsetName)
```

예시

```
///  
File inputStreamReaderTest = new File("test.txt");  
InputStream is = new FileInputStream(inputStreamReaderTest);  
Reader isr = new InputStreamReader(is, "UTF-8");
```

3

InputStreamReader

InputStream Byte 단위

Reader Char 단위

- OutputStreamWriter 생성자

Charset에 따라 문자를 읽어드려 char형태로 출력

4

```
OutputStreamWriter(OutputStream out)
OutputStreamWriter(OutputStream out, Charset cs)
OutputStreamWriter(OutputStream out, String charsetName)
```

예시

```
///  
File outputStreamWriterTest = new File("test.txt");  
OutputStream os = new FileOutputStream(outputStreamWriterTest);  
Writer isr = new OutputStreamWriter(os, "UTF-8");
```

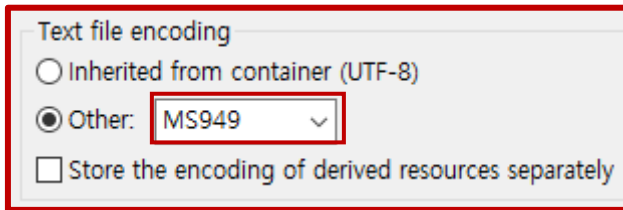
5

OutputStreamWriter

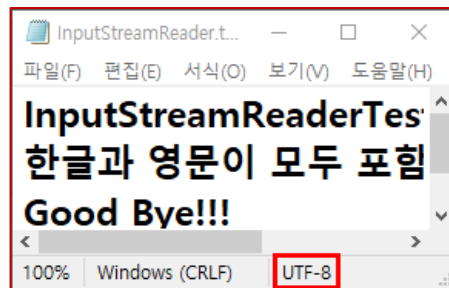
OutputStream Byte 단위

Writer Char 단위

1 Charset.defaultCharset() → MS949

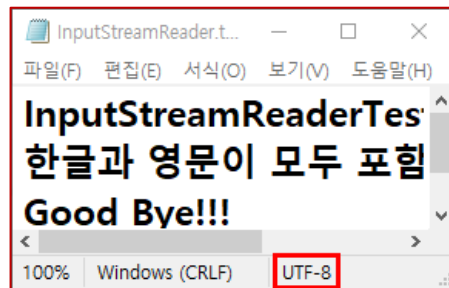


2 UTF-8 모드 텍스트 파일 → FileReader → char 단위 텍스트 읽기



```
InputStreamReaderTest ?뽳?젠  
?뵚?媛? ?뽕與갸읻 紐㉿몫 ?릹?봣?랴?뽕 ?연?똥?땡?땡.  
Good Bye!!!
```

3 UTF-8 모드 텍스트 파일 → FileInputStream + InputStreamReader → char 단위 텍스트 읽기



InputStreamReaderTest 예제
▶ 한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!

char 단위의 입출력 (Reader/Writer)

👉 InputStreamReader/OutputStreamWriter

← Byte 단위 입출력
→ Char 단위 입출력으로 변환

1 - Charset.defaultCharset() = MS949인 경우 → (UTF-8) 파일 읽어오기

예시

//#.파일 객체 선언

```
File inputStreamReader = new File("src/pack02_javaio/sec05_files/InputStreamReader.txt");
```

//#1. FileReader만을 이용하여 읽어오기 (UTF-8 모드 파일)

```
try(Reader reader = new FileReader(inputStreamReader)){
    int data;
    while((data=reader.read())!=-1) {System.out.print((char)data); }
} catch(IOException e) {}
System.out.println(); System.out.println();
```

//#2. FileInputStream+InputStreamReader를 사용하여 읽어오기(UTF-8 모드 파일)

```
try(InputStream is = new FileInputStream(inputStreamReader);
    InputStreamReader isr = new InputStreamReader(is, "UTF-8")){
    int data;
    while((data=isr.read())!=-1) { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding()); //UTF8
} catch(IOException e) {}
```

인코딩 방식 가져오기

2

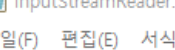
Text file encoding

- ☐ Inherited from container (UTF-8)

☒ Other: MS949

☐ Store the encoding of derived resources separately

3



InputStreamReader.t... — □ ×

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

InputStreamReaderTest
한글과 영문이 모두 포함
Good Bye!!!

100% Windows (CRLF) **UTF-8**

5

InputStreamReaderTest ?뽕?젠
?뵐?媛? ?뽕?뽕?뽕?뽕?뽕?뽕?뽕?뽕?뽕?뽕?
Good Bye!!!

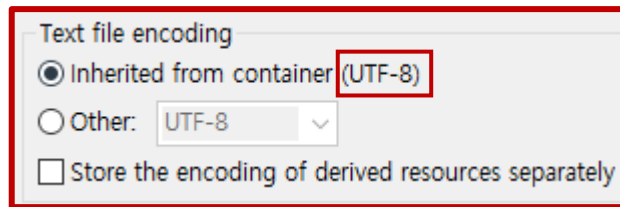
7

InputStreamReaderTest 예제
한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!

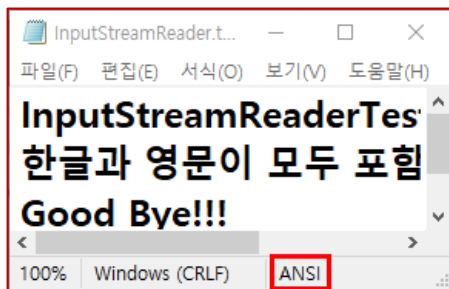
8

UTF8

1 Charset.defaultCharset() → UTF-8

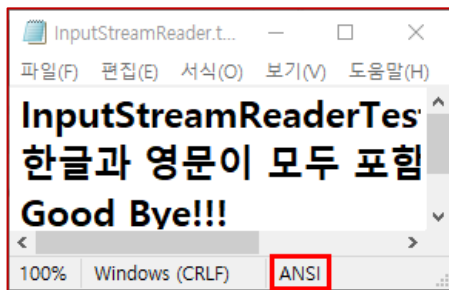


2 MS949 모드 텍스트 파일 → FileReader → char 단위 텍스트 읽기



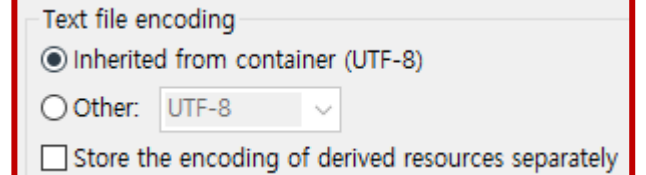
InputStreamReaderTest ㅅㅅㅅㅅ
ψ•♦ ♦♦♦♦♦♦♦♦ ♦♦♦ ♦♦♦♦♦♦ 0♦ ♦♦♦♦♦♦.
Good Bye!!!

3 MS949 모드 텍스트 파일 → FileInputStream + InputStreamReader → char 단위 텍스트 읽기



InputStreamReaderTest 예제
한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!

char 단위의 입출력 (Reader/Writer)



☞ InputStreamReader/OutputStreamWriter

← Byte 단위 입출력
→ Char 단위 입출력으로 변환

1 - **Charset.defaultCharset() = UTF-8**인 경우 → (**MS949**) 파일 읽어오기

예시

//#.파일 객체 선언

File inputStreamReader = new File("src/pack02_javaio/sec05_files/InputStreamReader.txt");

//#1. FileReader만을 이용하여 읽어오기 (**MS949** 모드 파일)

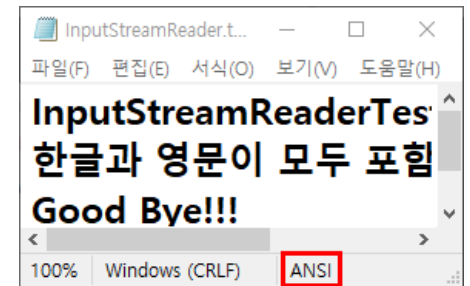
```
try(Reader reader = new FileReader(inputStreamReader)){
    int data;
    while((data=reader.read())!=-1) {System.out.print((char)data); }
} catch(IOException e) {}
System.out.println(); System.out.println();
```

//#2. FileInputStream+InputStreamReader를 사용하여 읽어오기(**MS949** 모드 파일)

```
try(InputStream is = new FileInputStream(inputStreamReader);
    InputStreamReader isr = new InputStreamReader(is, "MS949")){
    int data;
    while((data=isr.read())!=-1) { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding());//MS949
} catch(IOException e) {}
```

인코딩 방식 가져오기

3



5

InputStreamReaderTest ㅁㅁㅁㅁ
한글과 영문이 모두 포함 Good Bye!!!

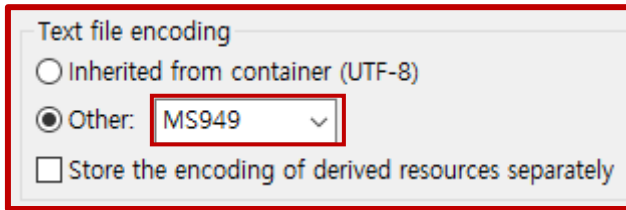
7

InputStreamReaderTest 예제
한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!

8

MS949

1 Charset.defaultCharset() → MS949



2 한글 콘솔 입력 → InputStreamReader : MS949/UTF-8 → char 단위 출력

MS949 읽기



안녕하세요
안녕하세요
MS949

UTF-8 읽기



안녕하세요
????????
UTF8

char 단위의 입출력 (Reader/Writer)

☞ InputStreamReader/OutputStreamWriter

← Byte 단위 입출력 → Char 단위 입출력으로 변환

① - InputSteramReader : 콘솔 입력을 문자단위로 읽어오기 (**Charset.defaultCharset() = MS949** 기준)

예시

②

//#1.콘솔 입력: InputStreamReader를 사용하여 콘솔 입력 읽어오기 (MS949 타입으로 읽기)

```
try{
    InputStreamReader isr = new InputStreamReader(System.in , "MS949");
    int data;
    while((data=isr.read())!='\r') { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding()); //MS949
} catch(IOException e) {}
```

System.in은 자원
반납 하지 않음

④

//#2.콘솔 입력: InputStreamReader를 사용하여 콘솔 입력 읽어오기 (UTF-8 타입으로 읽기) : 한글깨짐

```
try{
    InputStreamReader isr = new InputStreamReader(System.in, "UTF-8");
    int data;
    while((data=isr.read())!='\r') { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding()); //UTF8
} catch(IOException e) {}
```

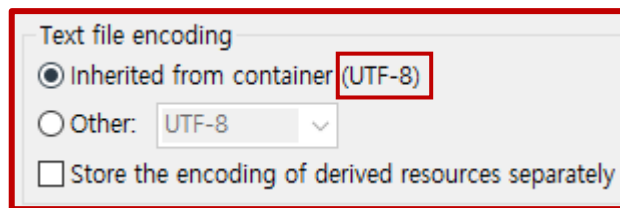
안녕하세요
안녕하세요
MS949

③

안녕하세요
????????
UTF8

⑤

1 Charset.defaultCharset() → UTF-8



2 한글 콘솔 입력 → InputStreamReader : UTF-8/MS949 → char 단위 출력

UTF-8 읽기



안녕하세요
안녕하세요
UTF8

MS949 읽기



안녕하세요
뵘뵡뵡뵡뵡뵡뵡
MS949

char 단위의 입출력 (Reader/Writer)

☞ InputStreamReader/OutputStreamWriter

← Byte 단위 입출력 → Char 단위 입출력으로 변환

① - InputSteramReader : 콘솔 입력을 문자단위로 읽어오기 (**Charset.defaultCharset() = UTF-8** 기준)

예시

②

##1.콘솔 입력: InputStreamReader를 사용하여 콘솔 입력 읽어오기 (UTF-8타입으로 읽기)

```
try{
    InputStreamReader isr = new InputStreamReader(System.in , "UTF-8");
    int data;
    while((data=isr.read())!='\r') { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding()); //UTF-8
} catch(IOException e) {}
```

System.in은 자원
반납 하지 않음

④

##2.콘솔 입력: InputStreamReader를 사용하여 콘솔 입력 읽어오기 (MS949타입으로 읽기) : 한글깨짐

```
try{
    InputStreamReader isr = new InputStreamReader(System.in, "MS949");
    int data;
    while((data=isr.read())!='\r') { System.out.print((char)data); }
    System.out.println("\n"+isr.getEncoding()); //MS949
} catch(IOException e) {}
```

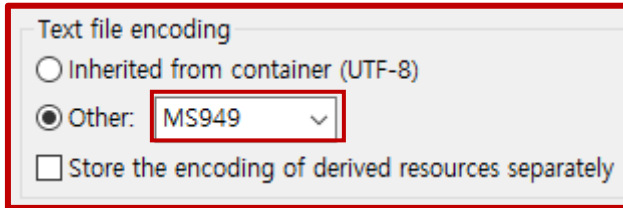
안녕하세요
안녕하세요
UTF8

③

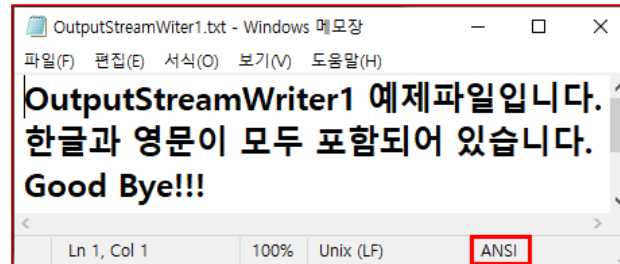
안녕하세요
뵘뵡뵡뵡뵡뵡뵡
MS949

⑤

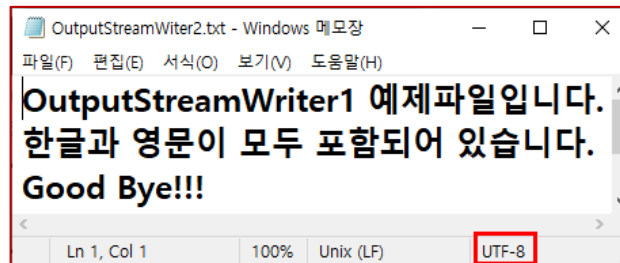
1 Charset.defaultCharset() → MS949



2 FileWriter → MS949 모드 텍스트 파일 쓰기



3 FileOutputStream + OutputStreamWriter → UTF-8 모드 텍스트 파일 쓰기



char 단위의 입출력 (Reader/Writer)

👉 InputStreamReader/OutputStreamWriter

← Byte 단위 입출력 → Char 단위 입출력으로 변환

- 1 - OutputStreamWriter : **Charset.defaultCharset() = MS949**인 경우
디폴트 문자셋(**MS949(ANSI)**)과 **UTF-8** 문자셋으로 각각 **파일** 쓰기

예시

///**파일 객체 선언**

```
File outputStreamWriter1 = new File("src/pack02_javaio/sec05_files/OutputStreamWriter1.txt");
```

3

///**#1. FileWriter만을 이용하여 파일쓰기 (디폴트(MS949))**

```
try(Writer writer = new FileWriter(outputStreamWriter1)){
```

```
    writer.write("OutputStreamWriter1 예제파일입니다.\n".toCharArray());
```

```
    writer.write("한글과 영문이 모두 포함되어 있습니다.");
```

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

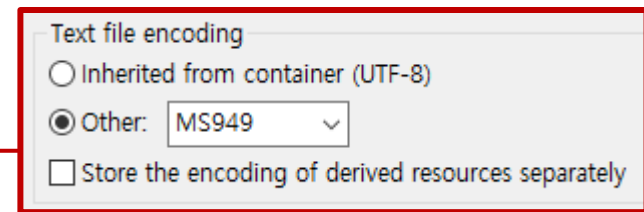
```
    writer.write("Good Bye!!!\n\n");
```

```
    writer.flush();
```

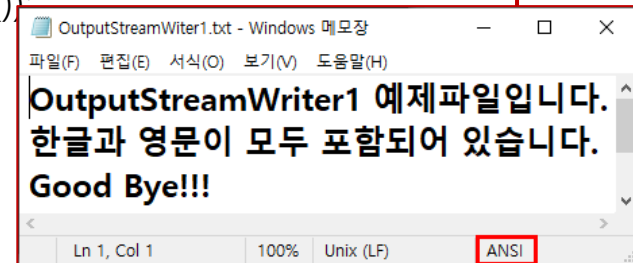
```
}
```

```
catch(IOException e) {}
```

2



4



char 단위의 입출력 (Reader/Writer)

👉 InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

- 1 - OutputStreamWriter : **Charset.defaultCharset() = MS949**인 경우
디폴트 문자셋(**MS949(ANSI)**)과 **UTF-8** 문자셋으로 각각 파일 쓰기

예시

3 **//#.파일 객체 선언**

```
File outputStreamWriter2 = new File("src/pack02_javaio/sec05_files/OutputStreamWriter2.txt");
```

//#2. FileOutputStream+OutputStreamWriter를 사용하여 파일쓰기(디폴트(MS949)→UTF-8 생성)

```
try(OutputStream os = new FileOutputStream(outputStreamWriter2, true);  
    OutputStreamWriter osw = new OutputStreamWriter(os, "UTF-8")){
```

```
    osw.write("OutputStreamWriter2 예제파일입니다.\n".toCharArray());
```

```
    osw.write("한글과 영문이 모두 포함되어 있습니다.");
```

```
    osw.write('\n');
```

```
    osw.write("Good Bye!!!\n");
```

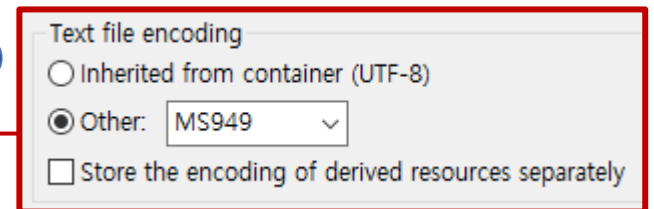
```
    osw.flush();
```

```
    System.out.println(osw.getEncoding()); //UTF8
```

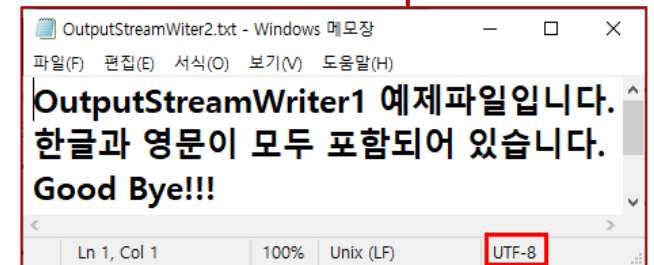
```
}
```

```
catch(IOException e) {}
```

2



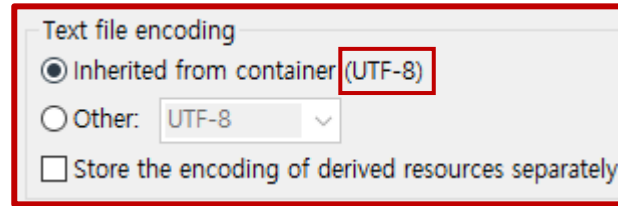
4



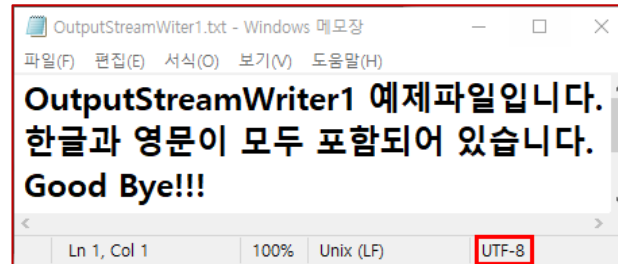
5

UTF8

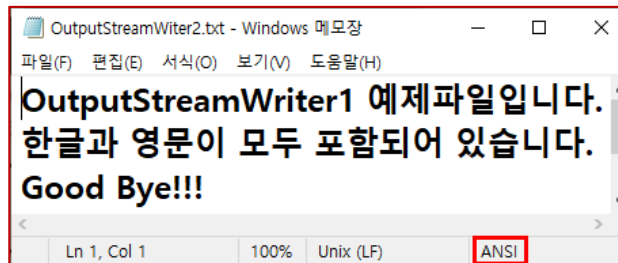
1 Charset.defaultCharset() → UTF-8



2 FileWriter → UTF-8 모드 텍스트 파일 쓰기



3 FileOutputStream + OutputStreamWriter → MS949 모드 텍스트 파일 쓰기



char 단위의 입출력 (Reader/Writer)

👉 InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

- 1 - OutputStreamWriter : **Charset.defaultCharset() = UTF-8**인 경우
디폴트 문자셋(**UTF-8**)과 **MS949** 문자셋으로 각각 **파일** 쓰기

예시

//#.파일 객체 선언

```
File outputStreamWriter1 = new File("src/pack02_javaio/sec05_files/OutputStreamWriter1.txt");
```

3

//#1. FileWriter만을 이용하여 파일쓰기 (디폴트(UTF-8))

```
try(Writer writer = new FileWriter(outputStreamWriter1)){
```

```
    writer.write("OutputStreamWriter1 예제파일입니다.\n".toCharArray());
```

```
    writer.write("한글과 영문이 모두 포함되어 있습니다.");
```

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

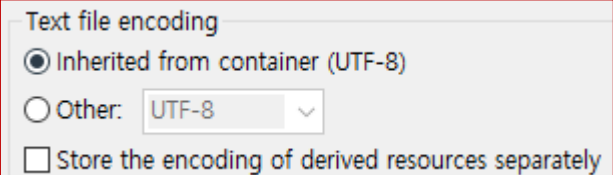
```
    writer.write("Good Bye!!!\n\n");
```

```
    writer.flush();
```

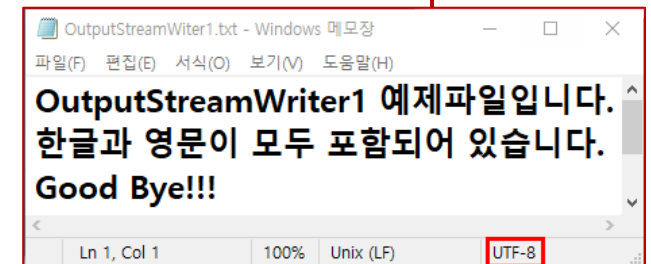
```
}
```

```
catch(IOException e) {}
```

2



4



char 단위의 입출력 (Reader/Writer)

👉 InputStreamReader/OutputStreamWriter

← Byte 단위 입출력 → Char 단위 입출력으로 변환

- 1 - OutputStreamWriter : **Charset.defaultCharset() = UTF-8**인 경우
디폴트 문자셋(**UTF-8**)과 **MS949** 문자셋으로 각각 **파일** 쓰기

예시

//#.파일 객체 선언

File outputStreamWriter2 = new File("src/pack02_javaio/sec05_files/OutputStreamWriter2.txt");

//#2. FileOutputStream+OutputStreamWriter를 사용하여 파일쓰기 (디폴트(UTF-8)→MS949 생성)

try(OutputStream os = new FileOutputStream(outputStreamWriter2, true);
OutputStreamWriter isr = new OutputStreamWriter(os, "MS949")){

isr.write("OutputStreamWriter2 예제파일입니다.\n".toCharArray());

isr.write("한글과 영문이 모두 포함되어 있습니다.");

isr.write('\n');

isr.write("Good Bye!!!\n");

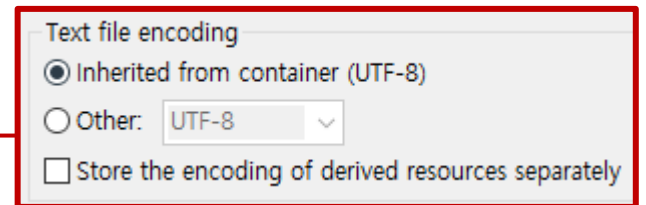
isr.flush();

System.out.println(isr.getEncoding());//MS949

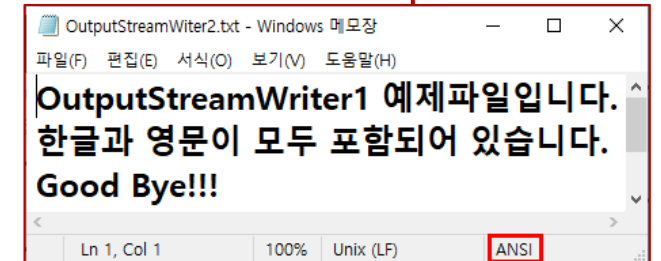
}

catch(IOException e) {}

2



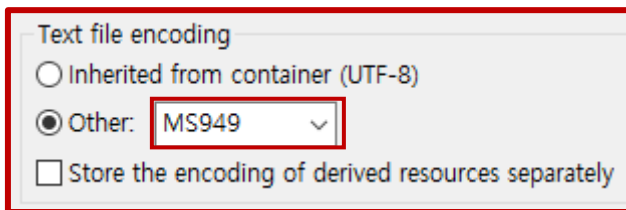
4



5

MS949

1 Charset.defaultCharset() → MS949



2 한글 콘솔 입력 → InputStreamReader → char 단위 읽어와 출력

MS949로 읽어와 출력

```
OutputStreamWriter를 이용한  
콘솔출력 예제입니다.  
→ 한글과 영문이 모두 포함되어 있습니다.  
Good Bye!!!  
MS949|
```

UTF-8로 읽어와 출력

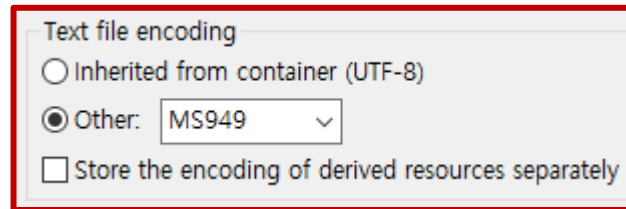
```
OutputStreamWriter瑜? ?씸?숏?뵆  
肄생낙異쑹젼 ?뵆?젠?엣?땡?땡.  
→ ?뵆潑?怨? ?뵆與몫씸 紐⑤몫 ?릍?뵆?릍?뵆 ?연?뵆?땡?땡.  
Good Bye!!!  
UTF8
```

char 단위의 입출력 (Reader/Writer)

☞ InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

1 - OutputStreamWriter : **Charset.defaultCharset() = MS949**인 경우 문자단위 콘솔 출력하기

2



예시

//#1.콘솔 출력: OutputStreamWriter를 사용하여 문자단위 콘솔 출력하기 (MS949 타입 출력)

3

```
try{ //MS949
    OutputStreamWriter osw = new OutputStreamWriter(System.out , "MS949");
    osw.write("OutputStreamWriter를 이용한\n".toCharArray());
    osw.write("콘솔출력 예제입니다.\n한글과 영문이 모두 포함되어 있습니다.");
    osw.write('\n');
    osw.write("Good Bye!!!\n");
    osw.flush();
    System.out.println(osw.getEncoding());// MS949
} catch(IOException e) {}
```

System.out 자원은
반납 하지 않음

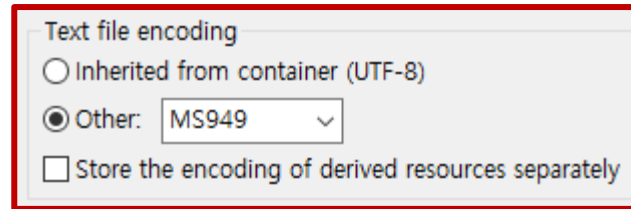
4

OutputStreamWriter를 이용한
콘솔출력 예제입니다.
한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!
MS949

char 단위의 입출력 (Reader/Writer)

☞ InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

- **OutputSteramWriter** : **Charset.defaultCharset() = MS949**인 경우 문자단위 **콘솔 출력**하기



예시

//#2. 콘솔 출력: OutputStreamWriter를 사용하여 문자단위 콘솔 출력하기 (UTF-8 타입 출력) 한글깨짐

1

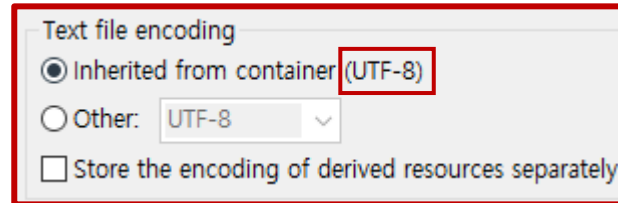
```
try{ //UTF-8
    OutputStreamWriter osw = new OutputStreamWriter(System.out, "UTF-8");
    osw.write(" OutputStreamWriter 를 이용한\n".toCharArray());
    osw.write("콘솔출력 예제입니다.\n한글과 영문이 모두 포함되어 있습니다. ");
    osw.write('\n');
    osw.write("Good Bye!!!\n");
    osw.flush();
    System.out.println(osw.getEncoding());// UTF-8
} catch(IOException e) {}
```

System.out 자원은
반납 하지 않음

2

```
OutputStreamWriter瑜? ?싯?쑈?뵆
肄생낙異쑈젯 ?뵆?젠?엣?땡?땡.
?뵆?뵆?怨? ?뵆?뵆?뵆 紐?뵆 ?뵆?뵆?뵆?뵆 ?연?뵆?땡?땡.
Good Bye!!!
UTF8
```

1 Charset.defaultCharset() → UTF-8



2 한글 콘솔 입력 → InputStreamReader → char 단위 읽어와 출력

**UTF-8로
읽어와 출력**



```
OutputStreamWriter를 이용한  
콘솔출력 예제입니다.  
한글과 영문이 모두 포함되어 있습니다.  
Good Bye!!!  
UTF8
```

**MS949로
읽어와 출력**



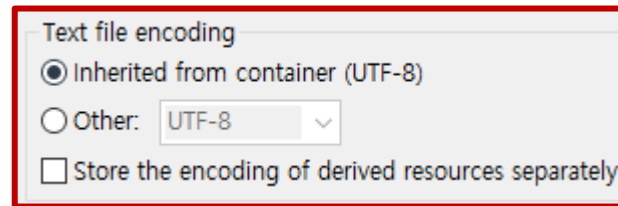
```
OutputStreamWriter를 이용한  
콘솔출력 예제입니다.  
한글과 영문이 모두 포함되어 있습니다.  
Good Bye!!!  
MS949
```

char 단위의 입출력 (Reader/Writer)

☞ InputStreamReader/OutputStreamWriter ← Byte 단위 입출력 → Char 단위 입출력으로 변환

1 - OutputStreamWriter : **Charset.defaultCharset() = UTF-8**인 경우 문자단위 콘솔 출력하기

2



예시

//#1.콘솔 출력: OutputStreamWriter를 사용하여 문자단위 콘솔 출력하기 (UTF-8 타입 출력)

3

```
try{ //UTF-8
    OutputStreamWriter osw = new OutputStreamWriter(System.out , "UTF-8");
    osw.write("OutputStreamWriter를 이용한\n".toCharArray());
    osw.write("콘솔출력 예제입니다.\n한글과 영문이 모두 포함되어 있습니다.");
    osw.write('\n');
    osw.write("Good Bye!!!\n");
    osw.flush();
    System.out.println(osw.getEncoding());// UTF-8
} catch(IOException e) {}
```

System.out 자원은
반납 하지 않음

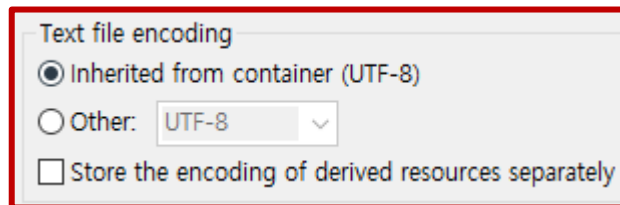
4

OutputStreamWriter를 이용한
콘솔출력 예제입니다.
한글과 영문이 모두 포함되어 있습니다.
Good Bye!!!
UTF8

char 단위의 입출력 (Reader/Writer)

👉 `InputStreamReader/OutputStreamWriter` ←————→ **Byte 단위 입출력** → **Char 단위 입출력으로 변환**

- **OutputStreamWriter** : **Charset.defaultCharset()** = UTF-8인 경우 문자단위 콘솔 출력하기



예시 // #2. 콘솔 출력: OutputStreamWriter를 사용하여 문자단위 콘솔 출력하기 (MS949 타입 출력) 한글깨짐

1

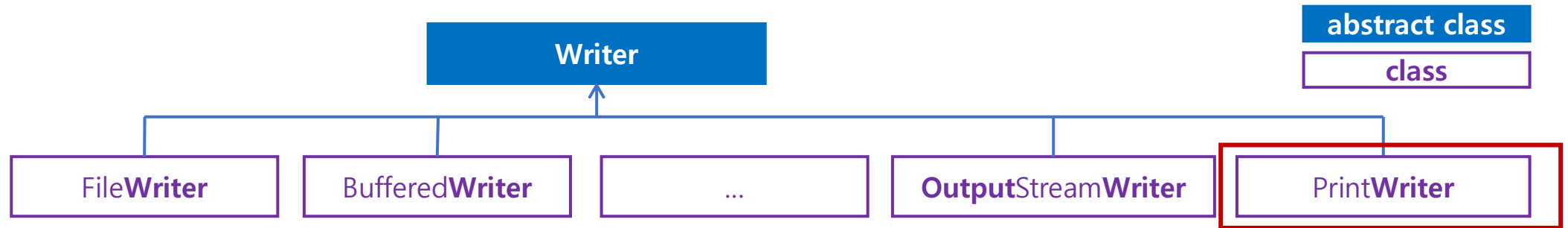
```
try{ // MS949
    OutputStreamWriter osw = new OutputStreamWriter(System.out, "MS949");
    osw.write(" OutputStreamWriter 를 이용한\n".toCharArray());
    osw.write("콘솔출력 예제입니다.\n한글과 영문이 모두 포함되어 있습니다. ");
    osw.write('\n');
    osw.write("Good Bye!!!\n");
    osw.flush();
    System.out.println(osw.getEncoding()); // MS949
} catch(IOException e) {}
```

System.out 자원은 반납 하지 않음

2

```
OutputStreamWriter?? ????
????? ???????.
?ψ·? ?????? ???? ???? ?????.
Good Bye!!!
MS949
```

The End



#4. `PrintWriter`로 `Writer` 객체 생성

char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

1

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

2 - PrintWriter의 생성자 (매개변수:파일,OutputStream,Writer)

```
PrintWriter(File file)
PrintWriter(String fileName)
PrintWriter(OutputStream out)
PrintWriter(OutputStream out, boolean autoFlush)
PrintWriter(Writer out)
PrintWriter(Writer out, boolean autoFlush)
```

3

Byte단위 IO인 PrintStream과의 차이점

예시

4

```
///  
File → PrintWriter  
PrintWriter pw1 = new PrintWriter(new File(...));  
///  
OutputStream → PrintWriter  
PrintWriter pw2 = new PrintWriter(new FileOutputStream (...));  
///  
Console(System.out) → PrintWriter  
PrintWriter pw3 = new PrintWriter(System.out);  
///  
Writer → PrintWriter  
PrintWriter pw4 = new PrintWriter(new FileWriter(...));
```

char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

- PrintWriter의 대표적 메서드

1

```
void    print(boolean b)
void    print(char c)
void    print(int i)
void    print(long l)
void    print(float f)
void    print(double d)
void    print(String s)

void    print(Object obj)
```

2

```
void    println()
```

3

```
void    println(boolean b)
void    println(char c)
void    println(int i)
void    println(long l)
void    println(float f)
void    println(double d)
void    println(String s)

void    println(Object obj)
```

4

```
PrintWriter    printf(String format, Object... args)
```

연속호출 가능

char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

- PrintWriter의 생성자 매개변수별 객체 생성

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

예시

1 //#.파일 객체 선언

```
File printWriter1 = new File("src/pack02_javaio/sec05_files/PrintWriter1.txt");  
File printWriter2 = new File("src/pack02_javaio/sec05_files/PrintWriter2.txt");  
File printWriter3 = new File("src/pack02_javaio/sec05_files/PrintWriter3.txt");
```

2 //#1. PrintWriter(File file) 생성자

```
try(PrintWriter pw = new PrintWriter(printWriter1)){  
    pw.println("PrintWriter 예제#1");  
    pw.println(13);  
    pw.println(5.8);  
    pw.print("안녕하세요! ");  
    pw.println("반갑습니다.");  
    pw.printf("%d",7).printf("%S %f", "감사", 3.7);  
}  
catch(IOException e) {}
```

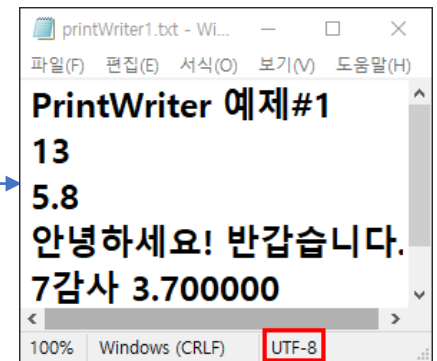
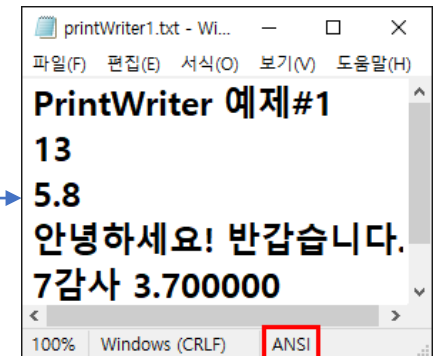
Default
Charset

MS949

2

Default
Charset

UTF-8



char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

- PrintWriter의 생성자 매개변수별 객체 생성

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

예시

1

```
//#2. PrintWriter(OutputStream os) 생성자  
try(PrintWriter pw = new PrintWriter(new  
FileOutputStream(printWriter2)));{  
    pw.println("PrintWriter 예제#2");  
    pw.println(13);  
    pw.println(5.8);  
    pw.print("안녕하세요! ");  
    pw.println("반갑습니다.");  
    pw.printf("%d",7).printf("%S %f", "감사", 3.7);  
}  
catch(IOException e) {}
```

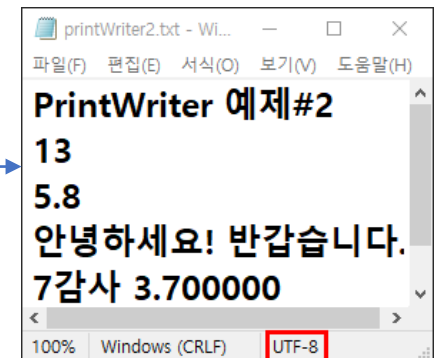
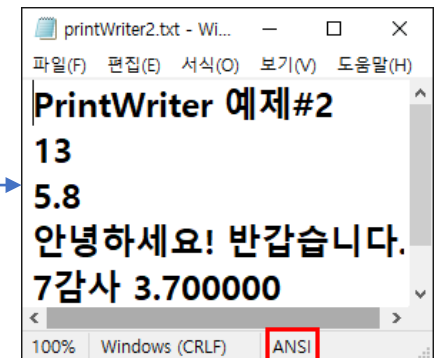
Default
Charset

MS949

2

Default
Charset

UTF-8



char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

- PrintWriter의 생성자 매개변수별 객체 생성

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

예시

1

```
//#3. PrintWriter(Writer w) 생성자  
try(PrintWriter pw = new PrintWriter(new FileWriter(printWriter3))){  
    pw.println("PrintWriter 예제#3");  
    pw.println(13);  
    pw.println(5.8);  
    pw.print("안녕하세요! ");  
    pw.println("반갑습니다.");  
    pw.printf("%d",7).printf("%S %f", "감사", 3.7);  
}  
catch(IOException e) {}
```

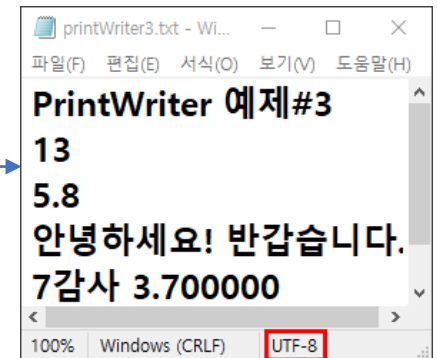
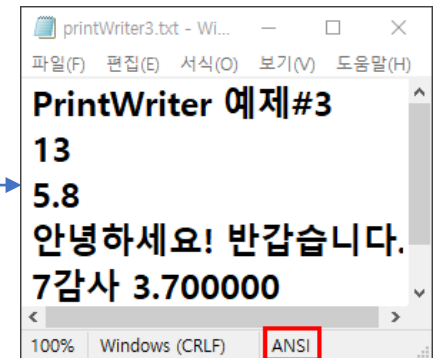
Default
Charset

MS949

2

Default
Charset

UTF-8



char 단위의 입출력 (Reader/Writer)

👉 PrintWriter

- PrintWriter의 생성자 매개변수별 객체 생성

PrintStream과 같이 다양한 타입의
출력에 특화된 클래스
자동 flush()기능 제공 (autoflush=true)
또는 자원 반납시 autoflush 됨

예시

1

```
//#4. PrintWriter(System.out) 생성자 : 콘솔출력
PrintWriter pw = new PrintWriter(System.out, true);
pw.println("PrintWriter 예제#4");
pw.println(13);
pw.println(5.8);
pw.print("안녕하세요! ");
pw.println("반갑습니다.");
pw.printf("%d", 7).printf("%S %f", "감사", 3.7);
```

Autoflush=true로 설정해 놓으면
자원을 반납하지 않아도 바로 출력
(자원 반납시 autoflush 됨)

2

```
PrintWriter 예제#4
13
5.8
안녕하세요! 반갑습니다.
7감사 3.700000
```

The End

JavalO Homework

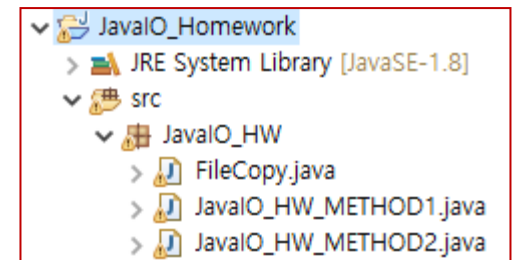
☞ 콘솔(WrWn 처리필요) 로 원본파일과 복사파일의 절대경로를 입력받아 파일 복사 수행

방법1#. InputStream 이용 : String 생성자를 이용하여 byte[]--> 특정 문자셋 String으로 변경

과제1

프로젝트명 : JavalO_Homework
패키지명 : JavalO_HW
클래스명 : JavalO_HW_Method1.class

방법#2. InputStreamReader를 사용하여 InputStream(System.in)을 Reader객체로 변환



과제2

프로젝트명 : JavalO_Homework
패키지명 : JavalO_HW
클래스명 : JavalO_HW_Method2.class

조건

조건1 : 경로는 절대경로로 입력
조건2 : 경로명에는 반드시 한글 포함
조건3 : .avi, .jpg, .txt 등 문서타입에 상관없이 복사 가능해야 함

소스파일의 위치를 절대경로로 입력하세요 (경로에 반드시 한글포함)
c:/한글파일.txt
c:/한글파일.txt
복사파일의 위치를 절대경로로 입력하세요 (경로에 반드시 한글포함)
d:/복사파일.txt
d:/복사파일.txt
File 복사가 완료되었습니다.

추가옵션

방법#3. Scanner 클래스 활용