16. 파일 입/출력

강사 강태광

I] 입출력(I/O)과 스트림(stream)

▶ 입출력(I/O)이란
- 입력(Input)과 출력(Output)을 줄여 부르는 말
- 두 대상 간의 데이터를 주고 받는 것

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

2] 스트림의 이해과 File 객체

3] 바이트기반 스트림 – InputStream, OutputStream

- 데이터를 바이트(byte)단위로 주고 받는다

InputStream	OutputStream
abstract **int** read()	abstract void write(**int** b)
int read(**byte**[] b)	void write(**byte**[] b)
int read(**byte**[] b, int off, int len)	void write(**byte**[] **b**, int off, int len)

입력스트림	출력스트림	대상
FileInputStream	**File**OutputStream	파일
ByteArrayInputStream	**ByteArray**OutputStream	메모리
PipedInputStream	**Piped**OutputStream	프로세스
AudioInputStream	**Audio**OutputStream	오디오장치

4] 보조스트림

- 스트림의 기능을 향상시키거나 새로운 기능을 추가하기 위해 사용
- 독립적으로 입출력을 수행할 수 없다

```
// 먼저 기반스트림을 생성한다.
FileInputStream fis = new FileInputStream("test.txt");
// 기반스트림을 이용해서 보조스트림을 생성한다.
BufferedInputStream bis = new BufferedInputStream(fis);

bis.read();   // 보조스트림인 BufferedInputStream으로부터 데이터를 읽는다.
```

입력	출력	설명
FilterInputStream	FilterOutputStream	필터를 이용한 입출력 처리
BufferedInputStream	BufferedOutputStream	버퍼를 이용한 입출력 성능향상
DataInputStream	DataOutputStream	int, float와 같은 기본형 단위(primitive type)로 데이터를 처리하는 기능
SequenceInputStream	SequenceOutputStream	두 개의 스트림을 하나로 연결
LineNumberInputStream	없음	읽어 온 데이터의 라인 번호를 카운트 (JDK1.1부터 LineNumberReader로 대체)
ObjectInputStream	ObjectOutputStream	데이터를 객체단위로 읽고 쓰는데 사용. 주로 파일을 이용하며 객체 직렬화와 관련있음
없음	PrintStream	버퍼를 이용하며, 추가적인 print관련 기능(print, printf, println메서드)
PushbackInputStream	없음	버퍼를 이용해서 읽어 온 데이터를 다시 되돌리는 기능 (unread, push back to buffer)

4] 문자기반 스트림 – Reader, Writer

- 입출력 단위가 문자(char, 2 byte)인 스트림. 문자기반 스트림의 최고조상

바이트기반 스트림	문자기반 스트림	대상
FileInputStream **File**OutputStream	**File**Reader **File**Writer	파일
ByteArrayInputStream **ByteArray**OutputStream	**CharArray**Reader **CharArray**Writer	메모리
PipedInputStream **Piped**OutputStream	**Piped**Reader **Piped**Writer	프로세스
***StringBuffer**InputStream* ***StringBuffer**OutputStream*	**String**Reader **String**Writer	메모리

바이트기반 보조스트림	문자기반 보조스트림
BufferedInputStream **Buffered**OutputStream	**Buffered**Reader **Buffered**Writer
FilterInputStream **Filter**OutputStream	**Filter**Reader **Filter**Writer
***LineNumber**InputStream*	**LineNumber**Reader
PrintStream	**Print**Writer
PushbackInputStream	**Pushback**Reader

4] 바이트기반 스트림

▶ InputStream(바이트기반 입력스트림의 최고 조상)의 메서드

메서드명	설 명
int available()	스트림으로부터 읽어 올 수 있는 데이터의 크기를 반환한다.
void close()	스트림을 닫음으로써 사용하고 있던 자원을 반환한다.
void mark(int readlimit)	현재위치를 표시해 놓는다. 후에 reset()에 의해서 표시해 놓은 위치로 다시 돌아갈 수 있다. readlimit은 되돌아갈 수 있는 byte의 수이다.
boolean markSupported()	mark()와 reset()을 지원하는지를 알려 준다. mark()와 reset()기능을 지원하는 것은 선택적이므로, mark()와 reset()을 사용하기 전에 markSupported()를 호출해서 지원여부를 확인해야한다.
abstract int read()	1 byte를 읽어 온다(0~255사이의 값). 더 이상 읽어 올 데이터가 없으면 -1을 반환한다. abstract메서드라서 InputStream의 자손들은 자신의 상황에 알맞게 구현해야한다.
int read(byte[] b)	배열 b의 크기만큼 읽어서 배열을 채우고 읽어 온 데이터의 수를 반환한다. 반환하는 값은 항상 배열의 크기보다 작거나 같다.
int read(byte[] b, int off, int len)	최대 len개의 byte를 읽어서, 배열 b의 지정된 위치(off)부터 저장한다. 실제로 읽어 올 수 있는 데이터가 len개보다 적을 수 있다.
void reset()	스트림에서의 위치를 마지막으로 mark()이 호출되었던 위치로 되돌린다.
long skip(long n)	스트림에서 주어진 길이(n)만큼을 건너뛴다.

4] 바이트기반 스트림

▶ OutputStream(바이트기반 출력스트림의 최고 조상)의 메서드

메서드명	설 명
void close()	입력소스를 닫음으로써 사용하고 있던 자원을 반환한다.
void flush()	스트림의 버퍼에 있는 모든 내용을 출력소스에 쓴다.
abstract void write(int b)	주어진 값을 출력소스에 쓴다.
void write(byte[] b)	주어진 배열 b에 저장된 모든 내용을 출력소스에 쓴다.
void write(byte[] b, int off, int len)	주어진 배열 b에 저장된 내용 중에서 off번째부터 len개 만큼만 읽어서 출력소스에 쓴다.

▶ ByteArrayInputStream과 ByteArrayOutputStream
- 바이트배열(byte[])에 데이터를 입출력하는 바이트기반 스트림

abstract int read()	1 byte를 읽어 온다(0~255사이의 값). 더 이상 읽어 올 데이터가 없으면 -1을 반환한다.

5] File 클래스

| 표 11-5 | File 클래스의 유용한 생성자

public File(String pathname)
pathname을 경로로 하는 파일 객체를 만든다.
public File(String parent, String child)
'parent + child'를 경로로 하는 파일 객체를 만든다.
public File(File parent, String child)
'parent(부모 파일) + child'를 경로로 하는 파일 객체를 만든다.

```
File f=new File("sample.txt");
File f=new File("c:\\java_work\\chapter11\\sample.txt");
```

5] File 클래스 - 유용 메소드

| 표 11-6 | File 클래스의 유용한 메소드

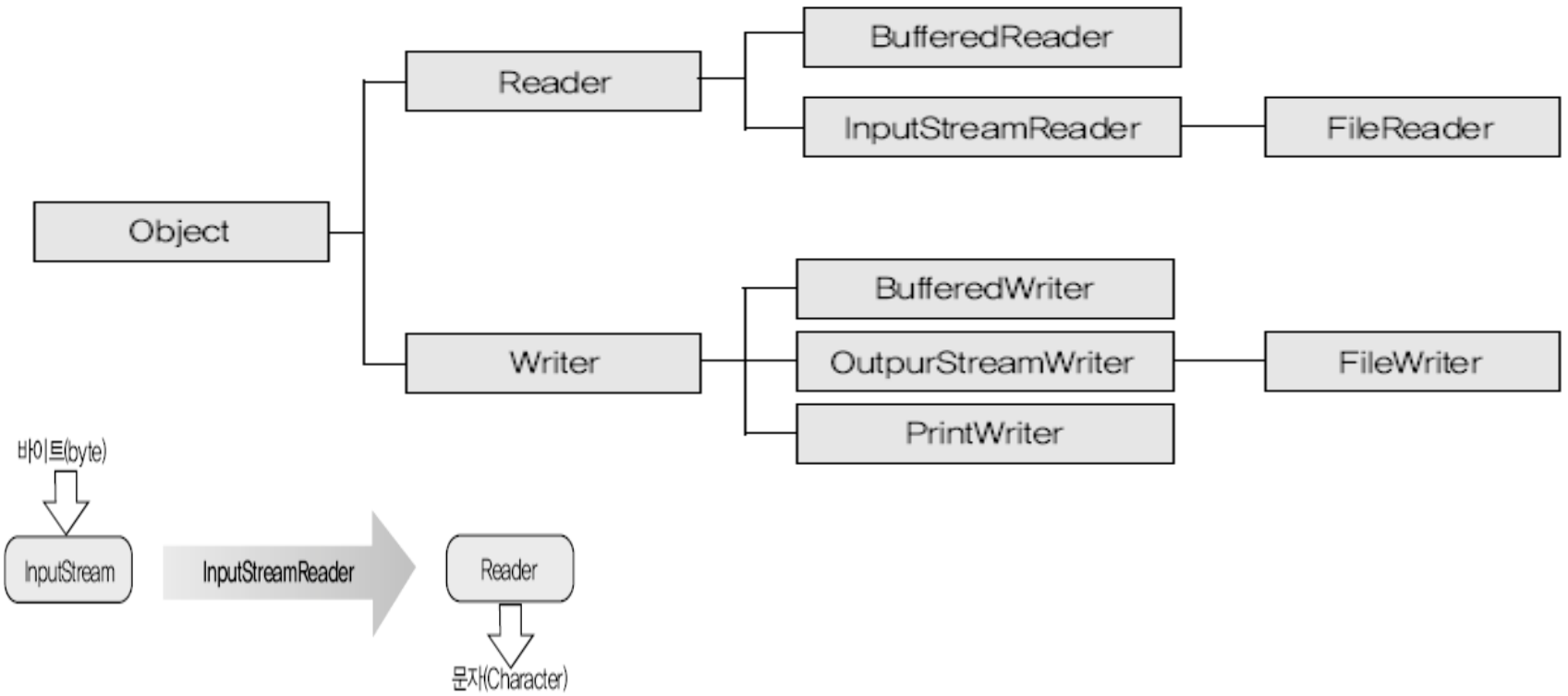
public boolean canRead( ), public boolean canWrite( )
읽을 수 있거나(can Read) 쓸 수 있는(can Write) 파일이면 true를, 아니면 false를 반환한다.
public boolean createNewFile( ) throws IOException
this가 존재하지 않는 파일이면 새 파일을 만들고 true를 반환하다. this가 존재하는 파일이면 false를 반환한다.
public static File createTempFile(String prefix, String suffix) throws IOException
파일 이름이 prefix이고 확장자는 suffix인 임시 파일을 임시(Temp) 디렉터리에 만든다.
public boolean isDirectory( )
디렉터리이면 true를, 아니면 false를 반환한다.
public boolean isFile( )
파일이면 true를, 아니면 false를 반환한다.
public boolean isHidden( )
숨겨진 파일이면 true를, 아니면 false를 반환한다.
public long lastModified( )
마지막으로 수정된 날짜를 long으로 반환한다.
public long length( )
파일의 크기(bytes)를 반환한다.
public String getName( ), public String getParent( ), public String getPath( )
각각 파일의 이름, 파일이 있는 디렉터리의 경로, 파일의 경로를 반환한다.

5] File 클래스 - 예시

```
package ch16;
import java.io.*;

public class Dir {
	public static void main(String[] args) {
		//  현재 위치
		//File file = new File(".");
		//File file = new File("src/ch16/");
		//File file = new File("../ch15/src/ch15/");
		File file = new File("../ch15/");
		String[] list = file.list();
		for(String str:list) {
		System.out.println(str);
		}
	}
}
```

5] 문자 기반 스트림

자바의 Writer 클래스와 Reader 클래스는 2byte 유니코드로 입출력할 수 있는 문자 기반 스트림을 제공한다



| 표 11-10 | InputStreamReader 클래스의 유용한 생성자

생성자	설 명
InputStreamReader(InputStream instreams)	instreams : 입력될 바이트 타입의 스트림(InputStream 클래스)
InputStreamReader(InputStream instreams, String encoding)	encoding : 변환 방법

5]문자 기반 스트림2

BufferedReader, BufferedWriter
버퍼를 사용하면 보다 효율적으로 데이터를 처리할 수 있는데,
이를 위해 BufferedReader, BufferedWriter 클래스가 제공된다

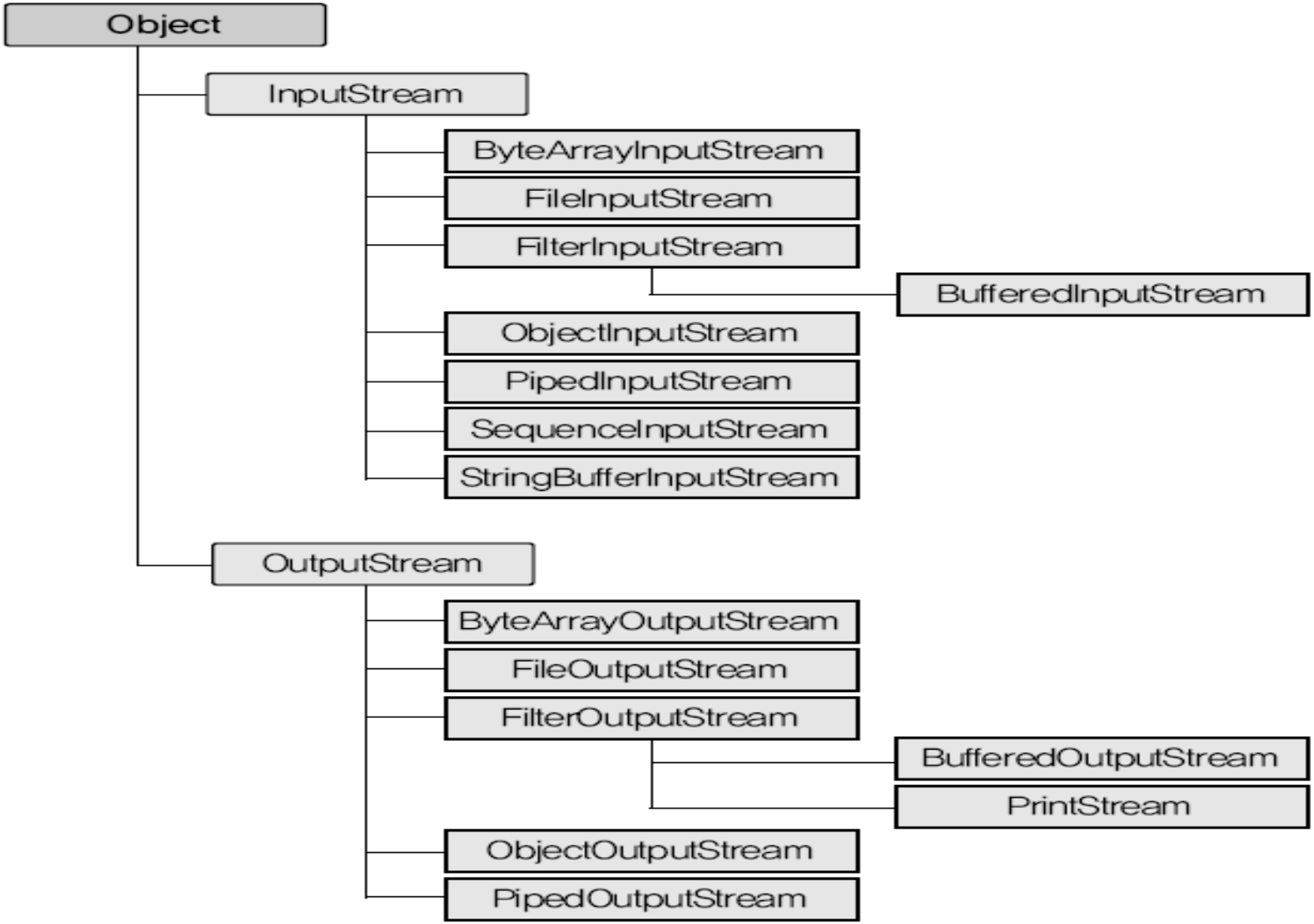
```
package ch16; // BufferedReader 예시문

import java.io.*;
class ReadLine {
	public static void main(String [] args)throws IOException{
		String name=null, addr=null ;
		BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
		System.out.print("이름을 입력하세요 => " );
		name = reader.readLine();
		System.out.print("주소를 입력하세요 => ");
		addr = reader.readLine();
		System.out.println(name+"님 반가워요! 당신은 "+ addr + "에 살고 계시군요..^^");
	}
}
```

5] FileReader, FileWriter

1. FileReader 클래스와 FileWriter 클래스는 파일에서 데이터를 읽거나 저장하는 각종 메소드를 제공한다.
2. FileReader 클래스는 파일 정보를 쉽게 읽을 수 있도록 InputStreamReader 클래스를 상속 받아 만든 클래스
3. FileWriter 클래스는 출력할 유니코드 문자를 바이트로 변환하여 파일에 저장

```
package ch16;
import java.io.*;
import java.util.Date;
import java.util.Scanner;
public class FileWriter1 {
		public static void main(String[] args) throws IOException {
				Scanner sc = new Scanner(System.in);
				System.out.println("출력할 파일명");
				String fileName  = sc.nextLine();
				System.out.println("저장할 글을 입력하시요");
				String msg  = sc.nextLine();
				FileWriter fw = new FileWriter(fileName);
				Date date  = new Date();
				fw.write(date+" : " + msg);
				fw.close();  // 반드시 해야 함
				System.out.println("출력끝");
				sc.close();
		}
}
```

5] InputStream, OutputStream 계층도



5] InputStream, Class 주요 메서드

표 11-1 InputStream 클래스의 주요 메소드

public abstract int read( ) throws IOException;	
스트림으로부터 다음의1 byte를 읽어와서 int로 반환한다. 파일의 모든 데이터를 읽어왔거나 더 이상 읽어 올 데이터가 없으면 -1을 반환한다.	
public int read(byte b[ ]) throws IOException	
스트림으로부터 다수의 byte를 읽어와서 b에 대입한다. 실제로 읽어온 byte 수를 반환하거나 읽어 올 것이 없으면 -1을 반환한다.	
public int read(byte b[ ], int off, int len) throws IOException	
스트림으로부터 len 만큼의 byte를 읽어와서 b[off]부터 차례대로 대입한다. 실제로 읽어온 byte 수를 반환하거나 읽어 올 것이 없으면 -1을 반환한다.	
public void close( ) throws IOException	
스트림을 닫는다. 스트림과 관련된 시스템 자원(메모리)을 반납한다.	

5] OutputStream, Class 주요 메서드

|표 11-2| OutputStream 클래스의 주요 메소드

public void write(byte b[ ], int off, int len) throws IOException
b[off]부터 b[len+off-1]까지의 데이터를 스트림에 출력한다.
public void write(byte b[ ]) throws IOException
바이트 배열 b를 스트림에 출력한다.
public abstract void write(int b) throws IOException;
b를 byte로 형변환하여 스트림에 출력한다. 1byte를 기록한다.
public void close( ) throws IOException
스트림을 닫는다. 스트림과 관련된 시스템 자원(메모리)을 반납한다.

6] FileInputStream과 FileOutputStream

▶ 파일(file)에 데이터를 입출력하는 바이트기반 스트림

생성자	설 명
FileInputStream(String name)	지정된 파일이름(name)을 가진 실제 파일과 연결된 FileInput Stream을 생성한다.
FileInputStream(File file)	파일의 이름이 String이 아닌 File인스턴스로 지정해주어야 하는 점을 제외하고 FileInputStream(String name)와 같다.
FileOutputStream(String name)	지정된 파일이름(name)을 가진 실제 파일과의 연결된 File OutputStream을 생성한다.
FileOutputStream(String name, boolean append)	지정된 파일이름(name)을 가진 실제 파일과 연결된 File OutputStream을 생성한다. 두번째 인자인 append를 true로 하면, 출력 시 기존의 파일내용의 마지막에 덧붙인다. false면, 기존의 파일내용을 덮어쓰게 된다.
FileOutputStream(File file)	파일의 이름을 String이 아닌 File인스턴스로 지정해주어야 하는 점을 제외하고 FileOutputStream(String name)과 같다.

6] FileInputStream과 FileOutputStream 예시1

▶ 파일(file)에 데이터를 입출력하는 바이트기반 스트림

```
package ch16;

import java.io.*;
import java.util.*;

class FileOutputStreamExample {
	public static void main(String[] args) throws Exception {
		String file, str;
		Date date = new Date();
		str = "파일 생성시간₩n" + date + "₩n";
		BufferedReader read = new BufferedReader(new InputStreamReader(System.in));
		System.out.print("파일 이름을 입력하세요 >> ");
		file = read.readLine();
		System.out.println("저장할 문자열을 입력하세요 ==> ");
		str += read.readLine();
		byte byte_str[] = str.getBytes();
		FileOutputStream fos = new FileOutputStream(file);
		fos.write(byte_str);
		fos.close();
		System.out.println(file + " 파일을 성공적으로 저장했습니다.");
	}
}
```

6] FileInputStream과 FileOutputStream 예시2

▶ 파일(file)에 데이터를 입출력하는 바이트기반 스트림

```
package ch16;

import java.io.*;
class FileInputStreamExample{
	public static void main(String[] args) throws Exception {
		int i = 0;
		String file;
		BufferedReader read = new BufferedReader( new
		InputStreamReader(System.in));
		System.out.print("읽어올 파일 이름을 입력하세요 >> ");
		file = read.readLine();
		System.out.println(file + ": 문서 내용 ********** ");
		FileInputStream fis = new FileInputStream(file);
		while((i = fis.read()) != -1){
			System.out.print((char)i);
		}
		fis.close();
	}
}
```

7] RandomAccessFile

| 표 11-7 | RandomAccessFile의 유용한 메소드

public native void seek(long pos) throws IOException;	파일 포인터를 pos로 이동시킨다.
public native long length( ) throws IOException;	파일의 길이를 반환한다.
public native long getFilePointer( ) throws IOException	파일 포인터의 위치를 반환한다.

| 표 11-8 | DataOutput으로부터 상속받은 메소드

write(int b), writeBoolean(boolean v), writeByte(int v), writeChar(int v), writeShort(int v), writeInt(int v), writeLong(long v), writeFloat(float v), writeDouble(double v), writeUTF(String str)
해당 자료를 파일에 출력한다.

| 표 11-9 | DataInput으로부터 상속받은 메소드

readBoolean( ), readByte( ), readChar( ), readShort( ), readInt( ), readLong( ), readFloat( ), readDouble( ), readUTF( )
해당 자료를 읽어온다

7] RandomAccessFile 예시

```
package ch16;
import java.io.*;
public class Random1 {
	public static void main(String[] args) throws IOException {
		FileOutputStream fos = new FileOutputStream(new File("ran"));
		for (int i = 0; i < 100; i ++ ) {
			fos.write(i);
		}
		fos.close();
		RandomAccessFile raf = new RandomAccessFile("ran", "rw");
		for(int i = 0; i < 100; i+=10) {
			raf.seek(i);
			System.out.println(raf.readByte());
		}
		raf.close();
	}

}
```

8] DataInputStream과 DataOutputStream1

- 기본형 단위로 읽고 쓰는 보조스트림
- 각 자료형의 크기가 다르므로 출력할 때와 입력할 때 순서에 주의

메서드 / 생성자	설 명
DataInputStream(InputStream in)	주어진 InputStream인스턴스를 기반스트림으로 하는 DataInputStream인스턴스를 생성한다.
boolean readBoolean() byte readByte() char readChar() short readShort() int readInt() long readLong() float readFloat() double readDouble()	각 자료형에 알맞은 값들을 읽어 온다. 더 이상 읽을 값이 없으면 EOFException을 발생시킨다.
String readUTF()	UTF형식으로 쓰여진 문자를 읽는다. 더 이상 읽을 값이 없으면 EOFException을 발생시킨다.
int skipBytes(int n)	현재 읽고 있는 위치에서 지정된 숫자(n) 만큼을 건너뛴다.

8] DataInputStream과 DataOutputStream2

메서드 / 생성자	설 명
DataOutputStream(OutputStream out)	주어진 OutputStream인스턴스를 기반스트림으로 하는 DataOutputStream인스턴스를 생성한다.
void writeBoolean(boolean b) void writeByte(int b) void writeChar(int c) void writeShort(int s) void writeInt(int l) void writeLong(long l) void writeFloat(float f) void writeDouble(double d)	각 자료형에 알맞은 값들을 출력한다.
void writeUTF(String s)	UTF형식으로 문자를 출력한다.
void writeChars(String s)	주어진 문자열을 출력한다. writeChar(char c)메서드를 여러 번 호출한 결과와 같다.
int size()	지금까지 DataOutputStream에 쓰여진 byte의 수를 알려준다.

8] DataInputStream과 DataOutputStream2 – 적용 예시

```
package ch16;
import java.io.*;
import java.util.Arrays;
class DataOutputStreamEx2 {
	public static void main(String args[]) {
		ByteArrayOutputStream bos = null;
		DataOutputStream dos = null;
		byte[] result = null;
		try {
			bos = new ByteArrayOutputStream();
			dos = new DataOutputStream(bos);
			dos.writeInt(10);
			dos.writeFloat(20.0f);
			dos.writeBoolean(true);
			result = bos.toByteArray();
			String[] hex = new String[result.length];
			for(int i=0;i<result.length; i++) {
				if(result[i] < 0) {
				    hex[i] = Integer.toHexString(result[i]+256);
				} else {
				  hex[i] = Integer.toHexString(result[i]);}
				  // hex[i] = "0"+hex[i];
				  // hex[i] = hex[i].substring(hex[i].length()-2);
			}
			System.out.println("10진수:" + Arrays.toString(result));
			System.out.println("16진수:" + Arrays.toString(hex));
			dos.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	} // main
}
```

9] PrintStream

- 데이터를 다양한 형식의 문자로 출력하는 기능을 제공하는 보조스트림
- System.out과 System.err이 PrintStream 이다.
- PrintStream보다 PrintWriter를 사용할 것을 권장한다

생성자 / 메서드		설 명
PrintStream(File file) PrintStream(File file, String csn) PrintStream(OutputStream out) PrintStream(OutputStream out,boolean autoFlush) PrintStream(OutputStream out,boolean autoFlush, String encoding) PrintStream(String fileName) PrintStream(String fileName, String csn)		지정된 출력스트림을 기반으로 하는 PrintStream인스턴스를 생성한다. autoFlush의 값을 true로 하면 println메서드가 호출되거나 개행문자가 출력될 때 자동으로 flush된다. 기본값은 false이다.
boolean checkError()		스트림을 flush하고 에러가 발생했는지를 알려 준다.
void print(boolean b) void print(char c) void print(char[] c) void print(double d) void print(float f) void print(int i) void print(long l) void print(Object o) void print(String s)	void println(boolean b) void println(char c) void println(char[] c) void println(double d) void println(float f) void println(int l) void println(long l) void println(Object o) void println(String s)	인자로 주어진 값을 출력소스에 문자로 출력한다. println메서드는 출력 후 줄바꿈을 하고, print메서드는 줄을 바꾸지 않는다.
void println()		줄바꿈 문자(line separator)를 출력함으로써 줄을 바꾼다.
PrintStream printf(String format, Object... args)		정형화된(formatted) 출력을 가능하게 한다.
protected void setError()		작업 중에 오류가 발생했음을 알린다.(setError()를 호출한 후에, checkError()를 호출하면 true를 반환한다.)

9] PrintStream-format1

format	설 명	결 과(int i=65)
%d	10진수(decimal integer)	65
%o	8진수(octal integer)	101
%x	16진수(hexadecimal integer)	41
%c	문자	A
%s	문자열	65
%5d	5자리 숫자. 빈자리는 공백으로 채운다.	65
%-5d	5자리 숫자. 빈자리는 공백으로 채운다.(왼쪽 정렬)	65
%05d	5자리 숫자. 빈자리는 0으로 채운다.	00065

format	설 명	결 과
%s	문자열(string)	ABC
%5s	5자리 문자열. 빈자리는 공백으로 채운다.	ABC
%-5s	5자리 문자열. 빈자리는 공백으로 채운다.(왼쪽 정렬)	ABC

9] PrintStream-format2

format	설 명	결과
%e	지수형태표현(exponent)	1.234568e+03
%f	10진수(decimal float)	1234.56789
%3.1f	출력될 자리수를 최소 3자리(소수점포함), 소수점 이하 1자리(2번째 자리에서 반올림)	1234.6
%8.1f	소수점이상 최소 6자리, 소수점 이하 1자리. 출력될 자리수를 최소 8자리(소수점포함)를 확보한다. 빈자리는 공백으로 채워진다.(오른쪽 정렬)	1234.6
%08.1f	소수점이상 최소 6자리, 소수점 이하 1자리. 출력될 자리수를 최소 8자리(소수점포함)를 확보한다. 빈자리는 0으로 채워진다.	001234.6
%-8.1f	소수점이상 최소 6자리, 소수점 이하 1자리. 출력될 자리수를 최소 8자리(소수점포함)를 확보한다. 빈자리는 공백으로 채워진다.(왼쪽 정렬) 1234.6	1234.6

format	설 명	결 과
%tR %tH:%tM	시분(24시간)	21:05 21:05
%tT %tH:%tM:%tS	시분초(24시간)	21:05:33 21:05:33
%tD %tm/%td/%ty	연월일	02/16/07 02/16/07
%tF %tY-%tm-%td	연월일	2007-02-16 2007-02-16

format	설 명
\t	탭(tab)
\n	줄바꿈 문자(new line)
%%	%

9] PrintStream-예시

```
package ch16;

import java.util.Date;

class PrintStreamEx1 {
	public static void main(String[] args) {
		int i = 65;
		float f = 1234.56789f;
		Date d = new Date();
		System.out.printf("문자 %c의 코드는 %d₩n", i, i);
		System.out.printf("%d는 8진수로 %o, 16진수로 %x₩n", i, i, i);
		System.out.printf("%3d%3d%3d₩n", 100, 90, 80);
		System.out.println();
		System.out.printf("12345678901234567890123456789O₩n");
		System.out.printf("%s%-5s%5s₩n", "123", "123", "123");
		System.out.println();
		System.out.printf("%-8.1f%8.1f %e₩n", f, f, f);
		System.out.println();
		System.out.printf("오늘은 %tY년 %tm월 %td일 입니다.₩n", d, d, d, d);
		System.out.printf("지금은 %tH시 %tM분 %tS초 입니다.₩n", d, d, d, d);
		System.out.printf("지금은 %1$tH시 %1$tM분 %1$tS초 입니다.₩n", d);
	}
}
```

10] StringReader와 StringWriter

- CharArrayReader, CharArrayWriter처럼 메모리의 입출력에 사용한다.
- StringWriter에 출력되는 데이터는 내부의 StringBuffer에 저장된다.

```
package ch16;

import java.io.*;
class StringReaderWriterEx {
	public static void main(String[] args) {
		String inputData = "ABCD";
		StringReader input = new StringReader(inputData);
		StringWriter output = new StringWriter();
		int data = 0;
		try {
			while((data = input.read())!=-1){
			output.write(data); // void write(int b)
		}
		}
		catch(IOException e) {
		}
		System.out.println("Input Data :" + inputData);
		System.out.println("Output Data :" + output.toString());
		// System.out.println("Output Data :" +output.getBuffer().toString());
	}
}
```

11] 콘솔 입출력-Scanner 클래스

Console 클래스의 단점
- 문자열은 읽을 수 있지만 기본 타입(정수, 실수) 값을 바로 읽을 수 없음
  java.util.Scanner
- 콘솔로부터 기본 타입의 값을 바로 읽을 수 있음

- 제공하는 메소드

리턴타입	메소드	설명
boolean	nextBoolean()	boolean(true/false) 값을 읽는다.
byte	nextByte()	byte 값을 읽는다.
short	nextShort()	short 값을 읽는다.
int	nextInt()	int 값을 읽는다.
long	nextLong()	long 값을 읽는다.
float	nextFloat()	float 값을 읽는다.
double	nextDouble()	double 값을 읽는다.
String	nextLine()	String 값을 읽는다.

12] 직렬화(serialization)/ 역직렬화(deserialization)

- **직렬화** : 객체를 '연속적인 데이터'로 변환하는 것. 반대과정은 '역직렬화'라고 한다.
- 객체의 인스턴스변수들의 값을 일렬로 나열하는 것

- 객체를 저장하기 위해서는 객체를 직렬화해야 한다.
- 객체를 저장한다는 것은 객체의 모든 인스턴스변수의 값을 저장하는 것

역직렬화(deserialization) : 직렬화된 객체를 원래 상태로 돌려놓는 것

1.FileInputStream

```
FileInputStream fileStream = new FileInputStream("MyGame.ser");
```

2.ObjectInputStream 만들기

```
ObjectInputStream os = new ObjectInputStream(fileStream);
```

3.객체 읽기

```
Object one = os.readObject();
Object two = os.readObject();
Object three = os.readObject();
```

4.객체 캐스팅

```
GameCharacter elf = (GameCharacter) one;
GameCharacter troll = (GameCharacter) two;
GameCharacter magician = (GameCharacter) three;
```

5.ObjectInputStream 닫기

```
os.close();
```

12] ObjectInputStream, ObjectOutputStream

- 객체를 직렬화하여 입출력할 수 있게 해주는 보조스트림

```
ObjectInputStream(InputStream in)
ObjectOutputStream(OutputStream out)
```

-객체를 파일에 저장하는 방법

```
FileOutputStream fos = new FileOutputStream("objectfile.ser");
ObjectOutputStream out = new ObjectOutputStream(fos);

out.writeObject(new UserInfo());
```

- 파일에 저장된 객체를 다시 읽어오는 방법

```
FileInputStream fis = new FileInputStream("objectfile.ser");
ObjectInputStream in= new ObjectInputStream(fis);

UserInfo info = (UserInfo)in.readObject();
```

12] ObjectInputStream, ObjectOutputStream

ObjectOutputStream
void defaultWriteObject()
void write(byte[] buf)
void write(byte[] buf, int off, int len)
void write(int val)
void writeBoolean(boolean val)
void writeByte(int val)
void writeBytes(String str)
void writeChar(int val)
void writeChars(String str)
void writeDouble(double val)
void writeFloat(float val)
void writeInt(int val)
void writeLong(long val)
void writeObject(Object obj)
void writeShort(int val)
void writeUTF(String str)

ObjectInputStream
void defaultReadObject()
int read()
int read(byte[] buf, int off, int len)
boolean readBoolean()
byte readByte()
char readChar()
double readDouble()
float readFloat()
int readInt()
long readLong()
short readShort()
Object readObject()
String readUTF

12] ObjectInputStream, ObjectOutputStream

ObjectOutputStream
void defaultWriteObject() void write(byte[] buf) void write(byte[] buf, int off, int len) void write(int val) void writeBoolean(boolean val) void writeByte(int val) void writeBytes(String str) void writeChar(int val) void writeChars(String str) void writeDouble(double val) void writeFloat(float val) void writeInt(int val) void writeLong(long val) void writeObject(Object obj) void writeShort(int val) void writeUTF(String str)

ObjectInputStream
void defaultReadObject() int read() int read(byte[] buf, int off, int len) boolean readBoolean() byte readByte() char readChar() double readDouble() float readFloat() int readInt() long readLong() short readShort() Object readObject() String readUTF

12] 객체를 직열화하는 예시 프로그램 1

```
package ch16;       // 객체를 역직렬화하는 프로그램
import  java.io.*;
import java.util.GregorianCalendar;
public class ObjectRe1 {
      public static void main(String[] args) throws FileNotFoundException, IOException,
                                                      ClassNotFoundException {
                        ObjectInputStream ois = null;
                        ois   = new ObjectInputStream(new FileInputStream("bb"));
                        try {
                              while(true) {
                                    GregorianCalendar gc = (GregorianCalendar)ois.readObject();
                                    int year   = gc.get(GregorianCalendar.YEAR);
                                    int month = gc.get(GregorianCalendar.MONTH);
                                    int date   = gc.get(GregorianCalendar.DAY_OF_MONTH);
                                    System.out.printf("%d/%d/%d₩n",year,month,date);
                              }
                        } catch (EOFException e) {
                                    System.out.println("작업끝");
                        } finally {
                                    ois.close();
                        }
            }
```

12] 객체를 직열화하는 예시 프로그램 2

```
package ch16;   // 객체를 직열화하는 예시 프로그램
import  java.io.*;
import java.util.GregorianCalendar;
public class ObjectWr1 {
    public static void main(String[] args) throws FileNotFoundException,
IOException {
        ObjectOutputStream  oos = null;
        oos = new ObjectOutputStream(new  FileOutputStream("bb"));
        // 월은 0부터 시작이므로 9->10월
        oos.writeObject(new  GregorianCalendar(2015,9,4));
        oos.writeObject(new  GregorianCalendar(2015,9,5));
        oos.writeObject(new  GregorianCalendar(2015,10,12));
        oos.close();
        System.out.println("출력끝");
        }
}
```