자바 입출력 IO

자바에서의 입출력

자바에서의 모든 데이터의 입출력은 스트림이라는 개념에 의해 이루어집니다. 스트림이란 단어의 뜻이 '흐르는 물'을 의미하듯이 연속된 일련의 데이터를 일컫는 말입니다. 데이터 입출력할 때 모든 데이터들은 형태와는 관계없이 일련의 흐름으로 전송합니다. 이번 장에서는 자바에서 입출력하는 기본적인 원리를 파악해 봅시다.

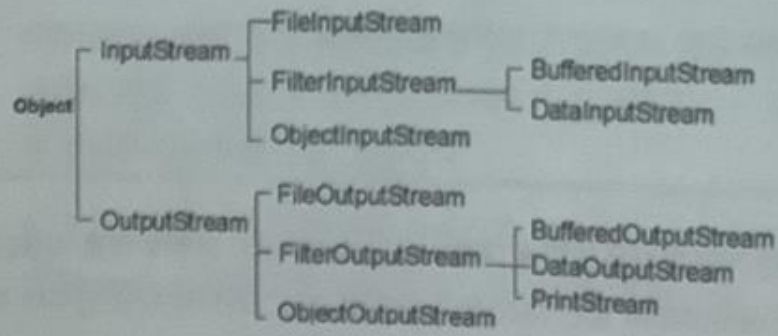
스트림

자바에서는 다양한 입출력 스트림 클래스가 존재합니다. 자바의 입출력 스트림은 크게 바이트 스트림, 텍스트 스트림 두 가지 종류로 나뉩니다.

:: 바이트(숫자) 스트림: 바이트, 바이트 배열, 정수 데이터 등의 흐름(InputStream 클래스와 OutputStream 클래스)

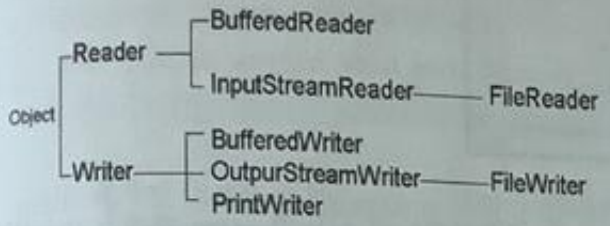
:: 텍스트(문자) 스트림: 문자, 문자 배열, 문자열의 흐름(Reader 클래스와 Writer 클래스)

자바에서는 입출력을 위한 패키지로 java.io를 제공합니다. 이 패키지에 포함된 클래스들의 계층 구조는 다음과 같습니다.



▶ 바이트 스트림과 연관된 클래스

바이트 입력 관련 스트림은 InputStream의 하위 클래스로 구현되어 있으며 InputStream이란 단어가 모두 포함되어 있습니다. 바이트 출력 관련 스트림은 OutputStream의 하위 클래스로 구현되어 있으며 OutputStream이란 단어가 모두 포함되어 있습니다.



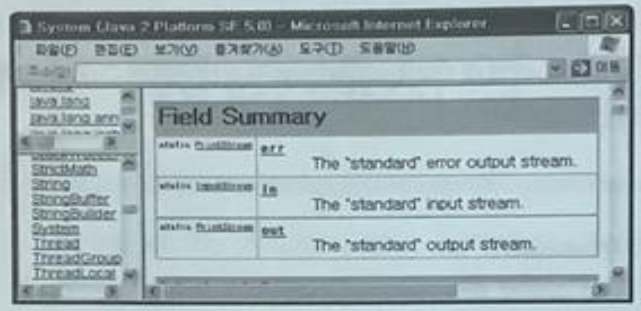
▶ 텍스트 스트림과 연관된 클래스

텍스트 입력 관련 스트림은 Reader의 하위 클래스로 구현되어 있으며 Reader란 단어가 모두 포함되어 있습니다. 텍스트 출력 관련 스트림은 Writer의 하위 클래스로 구현되어 있으며 Writer란 단어가 모두 포함되어 있습니다.

바이트 스트림 클래스	특징	문자 스트림 클래스
InputStream	기본 입력 스트림 클래스	Reader
FileInputStream	파일 입력 스트림 클래스	FileReader
FilterInputStream	다른 필터 클래스의 최상위 클래스	FilteredReader
BufferedInputStream	버퍼 기능이 있어 편리	BufferedReader
DataInputStream	자바 기본형 데이터를 읽는데 편리	없음
OutputStream	기본 출력 스트림 클래스	Writer
FileOutputStream	파일 출력 스트림 클래스	FileWriter
BufferedOutputStream	버퍼 기능이 있어서 편리	BufferedWriter
DataOutputStream	자바 기본형 데이터를 출력할 때 유용	없음
PrintStream	표준 출력 시스템으로 나감	PrintWriter

InputStream 클래스

자바에서는 표준 입출력 장치에 대해서 이미 이들과 연결된 세 개의 객체를 제공해 줍니다. System 클래스에 표준 입력 장치(키보드)로는 in이란 객체가 존재하고 표준 출력 장치(모니터)로는 out이란 객체가 존재합니다.



우리는 지금까지 출력을 위해서 System.out을 사용하였는데 이는 System 클래스 PrintStream 클래스로 선언된 out이란 정적 멤버 변수를 가져다 사용한 것입니다.

만일 키보드에서 입력을 받기 위해서는 System 클래스 내부에 선언된 InputStream 클래스로 선언된 in이란 정적 멤버 변수를 사용합니다.

InputStream 클래스는 추상 클래스로서, 바이트 입력 스트림 클래스의 최상위 클래스입니다. InputStream 클래스는 다음과 같은 메서드들을 제공합니다.

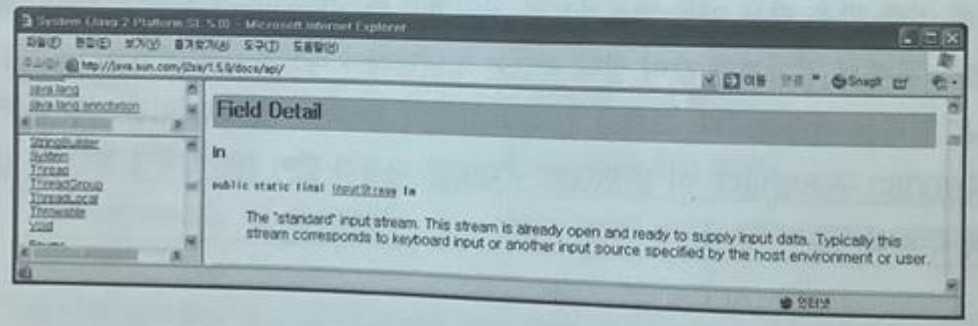
메서드	
int read()	한 바이트를 읽어 들인다.
int read(byte b[])	바이트 배열을 읽어 들인다.
int read(byte b[], int off, int len)	바이트 배열의 주어진 위치에 주어진 길이만큼 읽어 들인다.

위 입력 메서드는 모두 더 이상 읽을 값이 없을 경우 -1을 리턴합니다. 키보드에서 한 바이트를 입력받는 read() 메서드로 글자를 입력받아 입력받은 글자를 출력하는 예제를 작성해 봅시다.

예제. IOTest00.java

추상클래스 InputStream

이번에는 추상클래스인 InputStream 객체 변수 myIn를 선언하고 System.in 객체를 InputStream 객체 변수 myIn에 대입하였습니다. 이렇게 System.in 객체를 InputStream 객체 변수에 대입할 수 있는 이유는 in 객체가 System 클래스의 InputStream 클래스로 선언되어 있기 때문입니다.



위 예제(IOTest00.java)를 InputStream 객체 변수 myIn로 접근해야 입력받는 문장으로 수정해 봅시다.(결과는 위 예제와 동일하므로 생략)

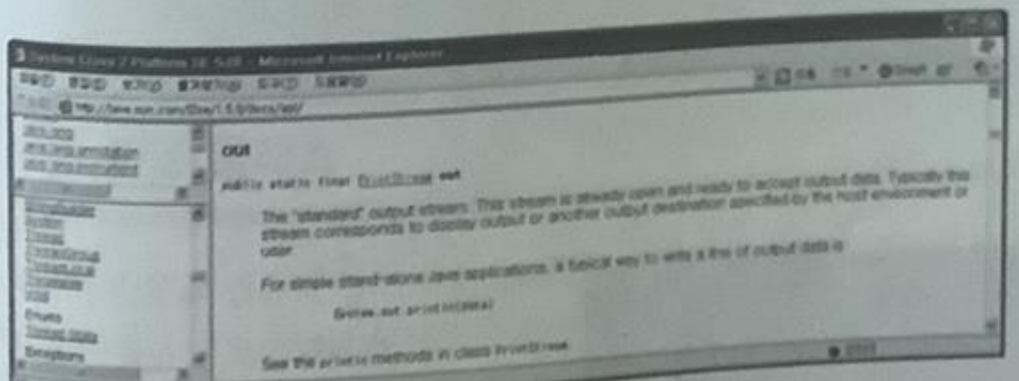
예제. IOTest00.java 복사 후 예제. IOTest01.java 작성

OutputStream 클래스

OutputStream 클래스는 추상 클래스로서 출력 스트림 클래스의 최상위 클래스입니다. OutputStream은 출력 장치에 바이트를 출력하기 위해 다음과 같은 메서드를 제공합니다.

메서드	
void close()	스트림을 닫는다.
void flush()	출력 스트림이 갖고 있는 버퍼의 내용을 모두 출력 스트림으로 내보낸다.
int write(byte[] b, int off, int len)	b 배열의 off 위치부터 len 만큼 출력한다.
void write(byte[] b)	바이트 배열로 출력한다.
void write(int b)	한 바이트로 출력한다.

출력을 위해서 지금까지 System.out 객체를 사용하였습니다. System.out 객체를 API Document를 살펴보면 System 클래스 내부에 PrintStream 클래스로 선언되어 있는 것을 확인할 수 있습니다.



예제. IOTest0A.java

예제. IOTest02.java

File 클래스

지금까지 표준 입출력 장치에서 입출력하는 방법을 살펴보았습니다. 애플리케이션 내부에서 처리된 데이터가 영구히 보관되려면 반드시 파일에 저장되어야 합니다. 파일에서 입출력하는 방법을 살펴보기에 앞서서 파일 클래스부터 살피고 넘어가겠습니다.

File 클래스는 파일 및 디렉토리를 관리할 수 있도록 기능을 제공해 주는 클래스입니다. File 클래스는

파일의 복사 또는 이름 변경 등의 조작을 할 경우에 사용될 뿐, 파일의 내용을 입출력하기 위한 메서드를 제공해 주지는 않습니다. 자바에서는 모든 데이터의 입출력을 스트림에 기반으로 하여 수행하므로, File 클래스 내부적으로 이러한 메서드를 구현할 필요가 없기 때문입니다.

:: File(String pathname) : 주어진 경로명을 추상 경로명으로 변환하여 새로운 File 객체를 생성

다음은 File 클래스의 주요 메서드입니다.

생성자	
File(String directory)	directory : 파일 경로
File(String directory, String file)	file : 파일 이름
File(File obj, String file)	obj : File 객체 멤버
메서드	
boolean canRead()	파일이 읽기 가능하면 true, 아니면 false
boolean canWrite()	파일이 쓰기 가능하면 true, 아니면 false
boolean delete()	파일을 삭제하고 true 반환, 파일을 삭제할 수 없으면 false 반환
boolean exists()	파일이 존재하면 true, 아니면 false 반환
String getAbsolutePath()	파일 절대 경로 반환
String getCanonicalPath()	파일 정규 경로 반환
String getParent()	부모 디렉토리 이름 반환
String getPath()	파일 경로 반환
String getName()	파일 이름 반환
boolean isDirectory()	디렉토리이면 true, 아니면 false 반환
boolean isFile()	파일이면 true, 아니면 false 반환
boolean isHidden()	파일 숨김 속성이면 true, 아니면 false 반환
boolean mkdir()	경로로 지정된 모든 부모 디렉토리가 존재해야 함.
boolean mkdirs()	지정된 디렉토리가 존재하지 않으면 생성한 다음 지정한 디렉토리를 생성
boolean renameTo(File newname)	newname으로 파일 이름을 변경. 성공하면 true, 실패하면 false
long lastModified()	1970년 1월 1일 0시부터 파일이 마지막으로 수정된 날짜까지의 시간을 밀리초 단위로 반환
long length()	파일 크기를 바이트로 반환
String [] list()	디렉토리에 있는 파일 목록 반환

예제. FileTest01.java

예제. FileTest02.java

FileInputStream 클래스

지금까지 표준 입출력 장치에서 입출력하는 방법을 살펴보았습니다. 애플리케이션 내부에서 처리된 데이터가 영구히 보관되려면 반드시 파일에 저장되어야 합니다. 자바에서는 파일에서 바이트 단위로 입력할 수 있도록 하기 위해서 FileInputStream 클래스를 제공해 주고 있습니다.

이 클래스를 다른 입력 관련 클래스와 연결해서 사용하여 파일에서 데이터를 읽거나 쓸 수 있습니다. FileInputStream 객체를 생성할 때 데이터를 읽어올 파일을 지정합니다. File 객체를 지정하는 방법과 문자열 형태로 파일의 경로명을 지정하는 방법이 있습니다.

생성자	
FileInputStream(File file)	주어진 File 객체가 가리키는 파일을 바이트 스트림으로 읽기 위한 FileInputStream 객체를 생성
FileInputStream(String name)	주어진 이름이 가리키는 파일을 바이트 스트림으로 읽기 위한 FileInputStream 객체를 생성

만일 FileInputStream 객체의 생성자에 지정한 파일이 존재하지 않는 경우에는 FileNotFoundException을 발생시킵니다. FileInputStream 클래스는 InputStream 클래스의 하위클래스이므로 InputStream 클래스에서 사용하던 메서드를 그대로 사용할 수 있습니다.

파일 생성하는 순서

```
FileInputStream fis=new FileInputStream("파일이름");
data=fis.read( );
fis.close( );
```

혹은

파일 생성하는 순서

```
File f=new File("파일이름");
FileInputStream fis=new FileInputStream(f);
data=fis.read( );
fis.close( );
```

예제. IOTest03.java

예제. FileType01.java

FileOutputStream 클래스

FileOutputStream 클래스는 데이터를 파일에 바이트 스트림으로 저장하기 위해 사용됩니다. FileOutputStream 클래스는 OutputStream 클래스의 하위클래스입니다. FileOutputStream 클래스의 생성자는 다음과 같습니다.

생성자	
FileOutputStream(File file)	주어진 File 객체가 가리키는 파일을 쓰기 위한 객체를 생성 기존의 파일이 존재할 때는 그 내용을 지우고 새로운 파일을 생성
FileOutputStream(String name)	주어진 이름의 파일을 쓰기 위한 객체를 생성
FileOutputStream(String name, boolean append)	주어진 append 값에 따라 새로운 파일을 생성하거나 또는 기존의 내용에 추가

파일명이나 File 클래스의 객체를 인수로 넘겨줌으로써 시스템에 파일을 직접 생성할 수 있습니다. 다음은 파일을 새로 생성하는 예제입니다.

파일 생성하는 순서

```
FileOutputStream fos = new FileOutputStream("파일이름");
fos.write( 내용 );
fos.close();
```

혹은

파일 생성하는 순서

```
File f=new File("파일이름");
FileOutputStream fos = new FileOutputStream(f);
fos.write( 내용 );
fos.close();
```

예제. IOTest06.java

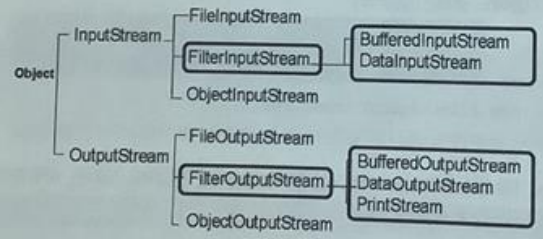
예제. FileCopy.java

36 필터 입출력 스트림

지금까지 살펴 본 입출력 스트림을 위한 클래스들은 단순히 스트림을 순서대로 읽거나 스트림에 순서대로 쓰는 기능만을 제공해 주었습니다. 이번 장에서는 기존의 스트림을 변형하여 새로운 스트림으로 바꾸어 주는 필터 입출력 스트림에 대해서 학습하겠습니다.

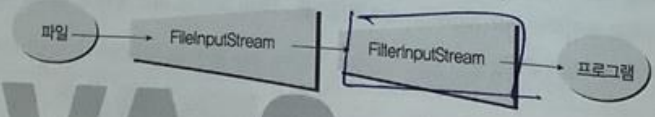
필터 입출력 스트림

실제로 프로그램을 작성하다 보면 앞 장에서 설명한 입출력 스트림들만으로는 기능이 부족한 경우가 있습니다. 이런 경우에는 자신이 원하는 입출력 기능을 사용하기 위하여 기존의 스트림에 필터를 적용하게 됩니다. 필터 기능은 출력할 경우에는 기존의 스트림을 원하는 기능을 제공하는 새로운 스트림으로 바꾸어 주는 역할을 의미하고 반대로, 입력 스트림에서의 필터링은 우리가 원하지 않은 부분을 걸러내는 역할을 하기도 합니다. 다음은 바이트 스트림의 필터 기능을 제공하는 입출력 스트림의 계층도입니다.



FilterInputStream

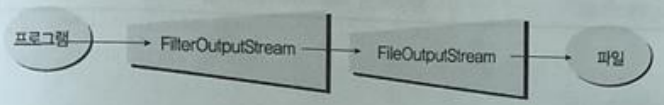
FilterInputStream 클래스는 필터 바이트 입력 스트림이므로 실제 데이터 원본(data source)과 연결된 바이트 입력 스트림을 내부적으로 가지고 있습니다. 그리고 이 바이트 입력 스트림으로부터 데이터를 읽고, 원하는 형태로 변환하기 위한 기능(filter)을 추가적으로 제공해 줍니다.



FilterInputStream 클래스는 기본적으로 InputStream 클래스가 제공해 주는 메서드들을 그대로 재정의하고 있고, 이 FilterInputStream 클래스를 상속하는 바이트 입력 스트림 클래스들이 실제 필터 기능을 제공해 주고 있습니다.

FilterOutputStream

FilterOutputStream 클래스는 필터 바이트 출력 스트림이므로 실제 데이터 대상(data destination)과 연결된 바이트 출력 스트림을 내부적으로 가지고 있습니다. 그리고 이 바이트 출력 스트림에 데이터를 쓰기 전에, 데이터를 원하는 형태로 변환하기 위한 기능(filter)을 추가적으로 제공해 줍니다.



FilterOutputStream 클래스는 기본적으로 OutputStream 클래스가 제공해 주는 메서드들을 그대로 재정의하고 있고, 이 FilterOutputStream 클래스를 상속하는 바이트 출력 스트림 클래스들이 실제 필터 기능을 제공해 주고 있습니다. 다음은 FilterOutputStream 클래스가 제공해 주고 있는 기본적인 객체 생성자입니다.

버퍼 입출력 스트림

바이트 입력 스트림을 사용하여 읽기를 할 경우 한 바이트씩 읽으므로 성능이 떨어질 수 있지만, 버퍼 바이트 입력 스트림은 바이트 입력 스트림으로부터 미리 버퍼에 데이터를 갖다놓게 되므로 데이터를 읽기 위한 동작이 보다 효율적입니다.

이러한 BufferedInputStream 클래스가 제공해주는 객체 생성자는 다음과 같습니다.

생성자	설명
BufferedInputStream(InputStream in):	주어진 바이트 입력 스트림에 대한 BufferedInputStream 객체를 생성하고, 내부 버퍼의 크기인 512 바이트로 설정합니다.
BufferedInputStream(InputStream in, int size):	주어진 바이트 입력 스트림에 대한 BufferedInputStream 객체를 생성하고, 내부 버퍼의 크기를 주어진 크기로 설정합니다.

BufferedOutputStream 클래스는 출력시에 버퍼링을 지원합니다. 물론, BufferedInputStream 클래스와 마찬가지로 실제 데이터 대상과 연결된 OutputStream 객체와 버퍼를 내부적으로 가지고 있습니다. 그리고, 데이터를 출력할 때, 먼저 내부 버퍼에 출력이 되고, 버퍼가 꽉 차거나 flush 메서드 또는 close 메서드가 호출될 때 내부 버퍼의 내용이 실제 OutputStream에 출력되게 됩니다.

따라서 OutputStream 객체가 출력할 때마다 실제 데이터 대상에 출력을 행할 때 발생하는 오버헤드(overhead)를 줄일 수 있습니다. 이러한 BufferedOutputStream 클래스가 제공해주는 객체 생성자는 다음과 같습니다.

생성자	설명
BufferedOutputStream(OutputStream out):	주어진 바이트 출력 스트림에 대한 BufferedOutputStream 객체를 생성하고, 내부 버퍼의 크기인 512 바이트로 설정합니다.
BufferedOutputStream (OutputStream out, int size):	주어진 바이트 출력 스트림에 대한 BufferedOutputStream 객체를 생성하고, 내부 버퍼의 크기를 주어진 크기로 설정합니다.

입력을 위한 파일 생성하는 순서

```
FileInputStream  fis = new FileInputStream("파일이름");
BufferedInputStream bis = new BufferedInputStream(fis);
bis.read( buffer );
bis.close();
fis.close();
```

출력을 위한 파일 생성하는 순서

```
FileOutputStream fos = new FileOutputStream("파일이름");
BufferedOutputStream bos = new BufferedOutputStream(fos);
bos.write( buffer );
bos.close();
fos.close();
```

예제. FileCopy01.java

DataOutputStream 클래스

지금까지 살펴본 바이트 입력 스트림은 바이트 단위의 출력 기능만을 제공해 주었지만, DataOutputStream 클래스는 자바에서 제공해 주고 있는 boolean ,byte, char, short, int, long, float, double 등과 같은 기본형을 직접 쓸 수 있도록 해 주고 있습니다.

생성자	
DataOutputStream(OutputStream output)	OutputStream을 인자로 받아서 DataOutputStream 객체를 생성
메서드	
void write(byte buffer[])	buffer를 스트림으로 출력
void write(int i)	i를 스트림으로 출력
void write(byte buffer[], int off, int len)	buffer 의 off 위치에서 len 만큼의 바이트를 스트림으로 출력
void writeBoolean(boolean b)	b를 스트림으로 출력
void writeByte(int i)	i의 하위 8비트를 스트림으로 출력
void writeBytes(String str)	문자열 str을 스트림으로 출력
void writeChar(int i)	i의 하위 16비트를 스트림으로 출력
void writeChars(String str)	문자열 str을 스트림으로 출력
void writeInt(int i)	i를 스트림으로 출력
void writeShort(short s)	s를 스트림으로 출력
void writeLong(long l)	l을 스트림으로 출력
void writeFloat(float f)	f를 스트림으로 출력
void writeDouble(double d)	d를 스트림으로 출력
void writeUTF(String str)	str을 유니코드 UTF-8 인코딩을 이용해서 변환하여 스트림으로 출력

예제. DataStreamTest1.java

DataInputStream 클래스

지금까지 살펴본 바이트 입력 스트림은 바이트 단위의 입력 기능만을 제공해 주었지만, DataInputStream 클래스는 자바에서 제공해 주고 있는 boolean ,byte, char, short, int, long, float, double 등과 같은 기본형을 직접 읽을 수 있도록 해 주고 있고, 이러한 읽기 동작은 플랫폼과는 무관하게 동작하도록 하고 있습니다.

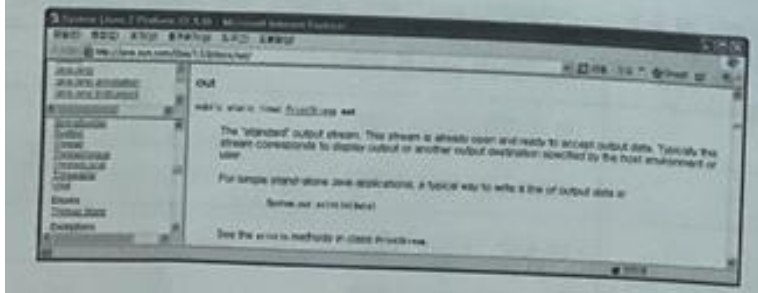
생성자	
DataInputStream(InputStream input)	InputStream을 인자로 받아서 DataInputStream 객체를 생성
메서드	
int read(byte buffer[], int off, int len)	len 개의 바이트를 읽어서 바이트 배열 buffer의 off위치에 저장
int read(byte buffer[])	바이트 배열 buffer 크기만큼 바이트를 반환
void readFully(byte buffer[])	스트림에서 buffer 크기만큼의 바이트를 읽어 buffer 배열에 저장
void readFully(byte buffer[], int off, int len)	스트림에서 len 만큼의 바이트를 읽어 buffer의 off 위치에 저장
byte readByte()	스트림에서 byte를 반환
char readChar()	한 문자를 읽어서 반환
short readShort()	short 값을 읽어서 반환
int readInt()	int 값을 읽어서 반환
long readLong()	long 값을 읽어서 반환
double readDouble()	double 값을 읽어서 반환
float readFloat()	float 값을 읽어서 반환
String readUTF()	UTF 인코딩 값을 읽어서 문자열로 반환

예제. DataStreamTest2.java

PrintStream 클래스

다른 스트림 클래스들이 문자 단위의 출력에 초점이 맞추어져 있는데 반해 PrintStream은 텍스트 출력 기능을 제공하는 스트림 클래스입니다.

표준 출력 장치인 모니터에 출력하기 위해서 지금 까지 우리가 사용해온 System.out.println에서 out 이 바로 PrintStream 클래스로 선언된 객체입니다.



PrintStream은 여러 종류의 데이터 형을 처리할 수 있는 print() 메서드와 println() 메서드를 갖고 있습니다. print() 메서드와 달리 println() 메서드는 데이터가 다 출력이 되면 New Line 문자를 추가로 출력해서 줄을 바꾸어 줍니다.

예제. PrintStreamTest1.java

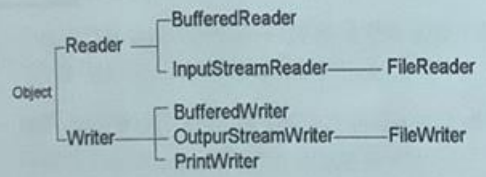
37 문자 스트림

지금까지는 바이트 단위로 입출력 해보았지만, 자바에서는 문자단위로도 파일 입출력을 할 수 있습니다. 자바의 Writer 클래스와 Reader 클래스는 2바이트 유니코드로 입출력할 수 있는 문자 기반 스트림을 제공합니다. 이번 장에서는 문자 단위로 입출력하는 방법을 살펴보겠습니다.

Reader 클래스와 Writer 클래스

Reader 클래스는 추상 문자 입력 스트림 클래스로서, 입력 장소를 기억하고 이 장치로부터 유니코드 문자 자료를 읽어 들이고 제어하는 메서드를 제공하는 문자 입력 스트림 클래스의 최상위 클래스입니다. Reader 클래스는 InputStream과 사용법이 거의 유사합니다.

Writer 클래스는 추상 문자 출력 스트림 클래스로서, 출력 목적지를 기억하고, 문자 출력 스트림 클래스의 최상위 클래스입니다. 그 역할은 OutputStream 클래스와 상당히 유사합니다.



InputStreamReader 클래스와 OutputStreamWriter 클래스

InputStreamReader 클래스는 Reader 클래스로부터 상속된 클래스로서, 바이트 스트림을 문자 스트림으로 변환하는 기능을 제공합니다. 생성자는 다음과 같습니다.

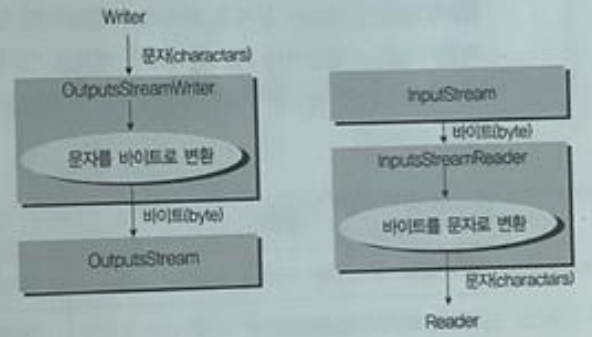
```
InputStreamReader(InputStream instreams)
InputStreamReader(InputStream instreams, String encoding)
```

:: instreams : 입력될 바이트 타입의 스트림(InputStream 클래스)
:: encoding : 변환 방법

OutputStreamWriter 클래스는 Writer 클래스로부터 상속된 클래스입니다. 문자 스트림을 바이트 스트림으로 변환하는 기능을 제공합니다. 생성자는 다음과 같습니다.

```
OutputStreamWriter(OutputStream outstreams)
OutputStreamWriter(Output5Stream outstreams, String encoding)
```

:: outstreams : 바이트 스트림으로 변환되어 출력될 출력스트림
:: encoding : 변환 방법



예제. ReaderWriterTest00.java

FileReader 클래스와 FileWriter 클래스

FileReader 클래스는 파일에 저장된 문자열을 읽어 들이는데 사용됩니다. FileReader 클래스 역시 데이터를 읽어 들일 소스인 파일과 직접 연결하여 사용합니다. 이 때, 지정한 파일이 존재하지 않는 경우에는 FileNotFoundException을 발생시킵니다. FileReader 클래스의 생성자는 다음과 같습니다.

생성자	설명
FileReader(File file)	주어진 File 객체를 이용하여 FileReader 객체 생성
FileReader(String fileName)	주어진 파일을 열어 FileReader 객체 생성

FileWriter 클래스는 문자를 파일에 저장하고자 할 경우에 사용합니다. FileOutputStream 클래스와 마찬가지로 파일명이나 File 클래스의 객체를 인수로 넘겨줌으로 시스템에 파일을 직접 생성하게 됩니다. 기본적으로, 파일이 이미 존재한다면 그 파일을 덮어쓰게 됩니다. FileWriter 클래스의 생성자는 다음과 같습니다.

생성자	설명
FileWriter (File file)	주어진 File 객체를 이용하여 FileWriter 객체 생성
FileWriter (String fileName)	주어진 파일을 열어 FileWriter 객체 생성
FileWriter(String fileName, boolean append)	주어진 파일을 append 값에 따라 읽기/추가 모드로 열어 FileWriter 객체 생성

예제. FileCopy02.java

BufferedReader 클래스

텍스트 입력을 위해 버퍼링을 하지 않을 경우, 파일로부터 한 바이트씩 읽어 문자로 변환하므로 매우 비효율적이 됩니다. BufferedReader 클래스는 문자 입력 스트림으로부터 문자들을 읽어 들이는데, 버퍼링을 함으로써 문자, 문자 배열, 문자열 라인 등을 보다 효율적으로 읽어 들일 수 있도록 해 줍니다. 다음은 BufferedReader 클래스가 제공해 주는 객체 생성자입니다.

생성자	설명
BufferedReader(Reader in)	입력 스트림에 대한 디폴트 크기의 내부 버퍼를 갖는 객체를 생성
BufferedReader(Reader in, int sz)	주어진 크기의 내부 버퍼를 갖는 객체를 생성

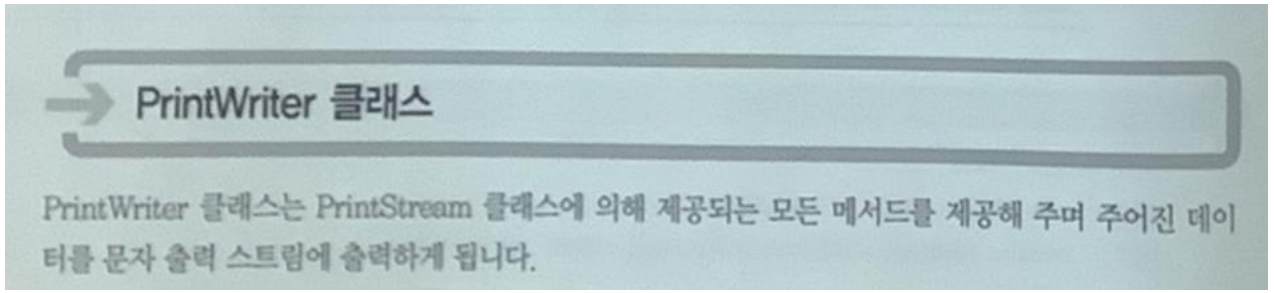
예제. ReaderWriterTest01.java

BufferedWriter 클래스

BufferedWriter 클래스는 문자 출력 스트림에 쓸 때, 버퍼링을 함으로써 문자, 문자 배열, 문자열 들을 보다 효율적으로 출력할 수 있도록 해 줍니다. 버퍼링을 하지 않을 경우, print() 메서드를 수행할 때마다 문자를 바이트로 변환하여 바로 파일에 쓰기 때문에 버퍼링을 할 때보다 훨씬 더 비효율적입니다. 다음은 BufferedWriter 클래스가 제공해 주는 객체생성자입니다.

생성자	
BufferedWriter(Writer out)	주어진 문자 출력 스트림에 대한 디폴트 크기의 내부 버퍼를 갖는 객체 생성
BufferedWriter(Writer out, int sz)	주어진 문자 출력 스트림에 대한 주어진 크기의 내부 버퍼를 갖는 객체 생성

예제. ReaderWriterTest03.java



예제. FileCopy03.java

38 다양한 입출력 클래스

이번 장에서는 다양한 입출력 방식을 배우겠습니다. 객체 직렬화를 통한 객체 데이터를 파일에 입출력하는 방법을 살펴보고 문자 입력 스트림을 이용해서 토큰 단위로 처리할 수 있는 StreamTockenizer 클래스를 살펴보겠습니다. 또한 지금까지는 데이터를 연속적으로 처리하였는데 임의의 위치에서 랜덤하게 입출력하는 방법을 살펴보겠습니다.

객체 직렬화

객체의 직렬화는 데이터들이 한 줄로 나열해서 스트림을 통해서 전송된다는 말입니다. 지금까지 살펴본 입출력 방식과 같이 데이터들이 개별적으로 전송되는 것이 아니고 클래스 내부에 설계된 멤버들이 객체 단위로 파일에 기록하거나 쓴다는 의미입니다. 지금부터 개별적으로 데이터를 전송할 경우 어떤 면에서 불편한지를 살펴보고 객체 직렬화를 이용하면 어떻게 이런 불편함을 없앨 수 있는지 살펴보도록 하겠습니다.

객체 직렬화가 왜 필요할까요?

객체 직렬화의 필요성을 살펴보기 위해서 이미 학습한 DataOutputStream 클래스를 이용한 출력을 살펴봅시다.

예제. DataStreamTest03.java

예제. DataStreamTest04.java

ObjectOutputStream과 ObjectInputStream

지금까지 살펴본 DataOutputStream 객체와 DataInputStream 객체는 데이터들을 개별적으로 전송했습니다. 그렇기 때문에 데이터를 읽어 들일 때에는 데이터를 저장한 순서대로 일렬로 읽어야 한다는 부담감이 있었습니다. 객체의 직렬화는 데이터들이 한 줄로 나열되어 스트림을 통해서 전송된다는 말입니다. 클래스 내부에 설계된 멤버들이 객체 단위로 파일에 기록하거나 읽는다는 의미입니다.

객체를 통째로 저장할 수 있도록 자바에서는 ObjectOutputStream 클래스를 제공합니다. ObjectOutputStream 클래스에서는 writeObject() 메서드를 사용하여 객체단위로 저장할 수 있습니다. 또한 ObjectInputStream 클래스는 readObject() 메서드를 사용하여 객체단위로 읽어올 수 있습니다.

클래스 내부에 설계된 멤버들이 객체 단위로 파일에 입출력하는 방법은 차후에 살펴보기로 하고 이번 예제에서는 ObjectOutputStream 클래스의 writeObject() 메서드와 ObjectInputStream 클래스의 readObject() 메서드를 사용하는 방법을 살펴보겠습니다.

예제. ObjectOutPutStreamTest01.java

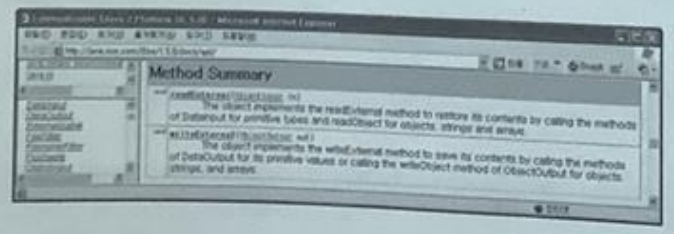
예제. ObjectOutPutStreamTest02.java

예제. ObjectOutPutStreamTest03.java

예제. ObjectOutPutStreamTest04.java

객체 직렬화를 위한 클래스 설계하기

객체 직렬화를 위한 클래스를 설계하려면 Serializable이나 Externalizable 인터페이스를 사용해야 합니다. 이 인터페이스는 두 개의 추상 메서드를 갖고 있습니다.



클래스 내부에 설계된 멤버들이 객체 단위로 파일에 입출력하는 방법을 살펴보기 위해서 우선 3개의 클래스를 사용해야 합니다.

객체를 통째로 저장하거나 읽어오기 위해서는 ObjectOutputStream 클래스와 ObjectInputStream 클래스를 사용해야 하고, 데이터를 직렬화해서 파일에 읽거나 쓰기 위해서 Externalizable 인터페이스를 구현한 클래스를 설계해서 그 내부의 직렬화할 데이터를 멤버로 선언하고 readExternal 메서드와 writeExternal 메서드를 오버라이딩해서 클래스의 멤버변수들을 직렬화시켜야 합니다.

writeExternal 메서드의 전달인자가 ObjectOutput 인터페이스로 선언되어 있습니다. ObjectOutput 인터페이스는 writeObject() 메서드를 제공하기 때문에 writeObject() 메서드를 사용하여 멤버변수들을 직렬화해서 저장하고 이렇게 저장된 데이터들 순서대로 readExternal 메서드에서 읽어오면 됩니다. readExternal 메서드의 전달인자 역시 ObjectInput 인터페이스로 선언되어 있는데 이 역시 readObject() 메서드를 제공되므로 readObject() 메서드를 사용하여 writeExternal 메서드에서 저장한 순서대로 읽어오면 됩니다.

예제. Customer.java

예제. ObjectStreamTest05.java

예제. ObjectStreamTest06.java

StreamTokenizer 클래스 사용하기

StreamTokenizer 클래스는 문자 스트림(Reader)으로부터 토큰을 생성하는 기능을 제공합니다. 문자와 숫자를 구분하여 처리할 때 유용하게 사용됩니다.

생성자	
StreamTokenizer(Reader read)	StreamTokenizer 객체를 생성

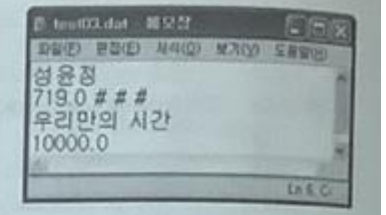
속성	
double nval	만약, 현재 토큰이 숫자이면 숫자값을 반환
String sval	만약, 현재 토큰이 문자이면 문자값을 반환
int type	nextTocken( ) 메서드를 호출한 후 이 필드는 토큰이 읽은 타입(아래 상수값)을 반환

상수			
TT_EOF	파일의 끝	TT_EOL	한 줄의 끝
TT_NUMBER	읽은 숫자	TT_WORD	읽은 단어

메서드	
int nextToken()	다음 토큰이 단어면 TT_WORD 반환, 숫자면 TT_NUMBER 가 반환
void eolIsSignificant(boolean flag)	flag가 true이면 end-of-line을 토큰으로 취급하고, false면 공백으로 처리
int lineno()	현재 줄의 번호를 반환
void whitespaceChars(int a, int b)	a-b 사이의 모든 문자를 공백으로 처리
void wordChars(int a, int b)	a-b 사이의 모든 문자를 단어로 취급
void ordinaryChar(int ch)	ch를 정규 문자로 지정
void quoteChar(int ch)	ch를 quote 문자로 설정

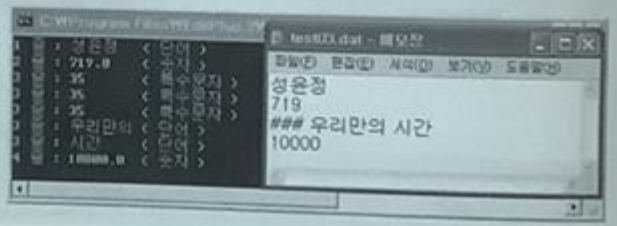
예제. StreamTokenizerTest01.java

아래 예제를 실습하기 전에 다음과 같은 텍스트 파일을 만들어 두어야 합니다.



예제 StreamTokenizer 클래스 사용 예제-[파일이름 : StreamTokenizerTest01.java]

```
001:import java.io.*;
002:class StreamTokenizerTest01 {
003:  public static void main(String [] args)throws IOException{
004:    BufferedReader bf = new BufferedReader(new FileReader("test03.dat"));
005:    StreamTokenizer st = new StreamTokenizer(bf);
006:    //토큰 타입을 알아내서 스트림 끝이 아니면 계속 반복
007:    while(st.nextToken() != StreamTokenizer.TT_EOF){
008:      switch( st.ttype ){
009:        case StreamTokenizer.TT_NUMBER :
010:            System.out.println(st.lineno( ) + "줄 : "+ st.nval+ "\t( 숫자 )");
011:            break;
012:        case StreamTokenizer.TT_WORD:
013:            System.out.println(st.lineno( ) + "줄 : " + st.sval + "\t( 단어 )");
014:            break;
015:       default:
016:            System.out.println( st.lineno( ) + "줄 : "+ st.ttype + "\t( 특수문자 )");
017:    }
018:  }
019:  }
020:}
```



파일에 임의 접근하기

RandomAccessFile 클래스는 기존의 클래스들이 순차적인 입출력만이 가능했던 것에 비하여 파일의 입출력 위치를 임의로 변경하며 입출력을 수행할 수 있습니다. RandomAccessFile 클래스는 일반적인 스트림 관련 클래스들과는 달리 InputStream과 OutputStream으로부터 상속을 받지 않았습니다. 그렇기 때문에 FilterStream 객체에 연결시켜 사용할 수 없습니다. 그 대신 RandomAccessFile에서는 DataInput과 DataOutput 인터페이스를 구현하여 자바 기본 데이터형을 읽고 쓸 수 있는 메서드들을 제공하고 있습니다. RandomAccessFile 클래스 객체는 생성될 때 FileInputStream이나 FileOutputStream에서처럼 파일명이나 File 객체를 넘겨주어야 합니다. 그리고 추가적으로 해 주어야 할 것은 파일 모드를 반드시 설정해 주어야 합니다. 파일 모드는 읽기전용일 때 "r"을, 읽고 쓰는 것이 가능하게 할 때는 "rw"를 문자열 값으로 넘겨주면 됩니다.

:: RandomAccessFile(String name, String mode) – 주어진 이름의 파일을 주어진 모드로 오픈

:: RandomAccessFile(File file, String mode) – 주어진 File 객체가 가리키는 파일을 주어진 모드로 오픈

RandomAccessFile 클래스는 파일에 대해서 임의 접근을 위해서 파일 포인터의 개념을 제공하고 있습니다. 파일 포인터는 파일 안에서의 현재 위치를 나타내며 파일 생성 시 초기 위치 값은 0입니다. 파일을 읽거나 쓰게 되면 파일 포인터는 위치를 변경하게 되며, 이 파일포인터를 명시적으로 지정된 위치로 이동시키거나 위치를 찾는 등의 메서드를 제공하고 있습니다. 그리고 DataInput 인터페이스와 DataOutput 인터페이스에서 정의하고 있는 모든 메서드를 구현하여 제공해 주고 있습니다.

:: void seek(long pos): – 지정된 바이트 수 바로 위치에 파일 포인터를 위치시킨다.

:: long getFilePointer(): – 현재 파일 포인터의 위치를 얻는다.

:: long length(): – 파일의 전체 크기를 얻는다.

예제. RandomAccessFileTest.java

The End