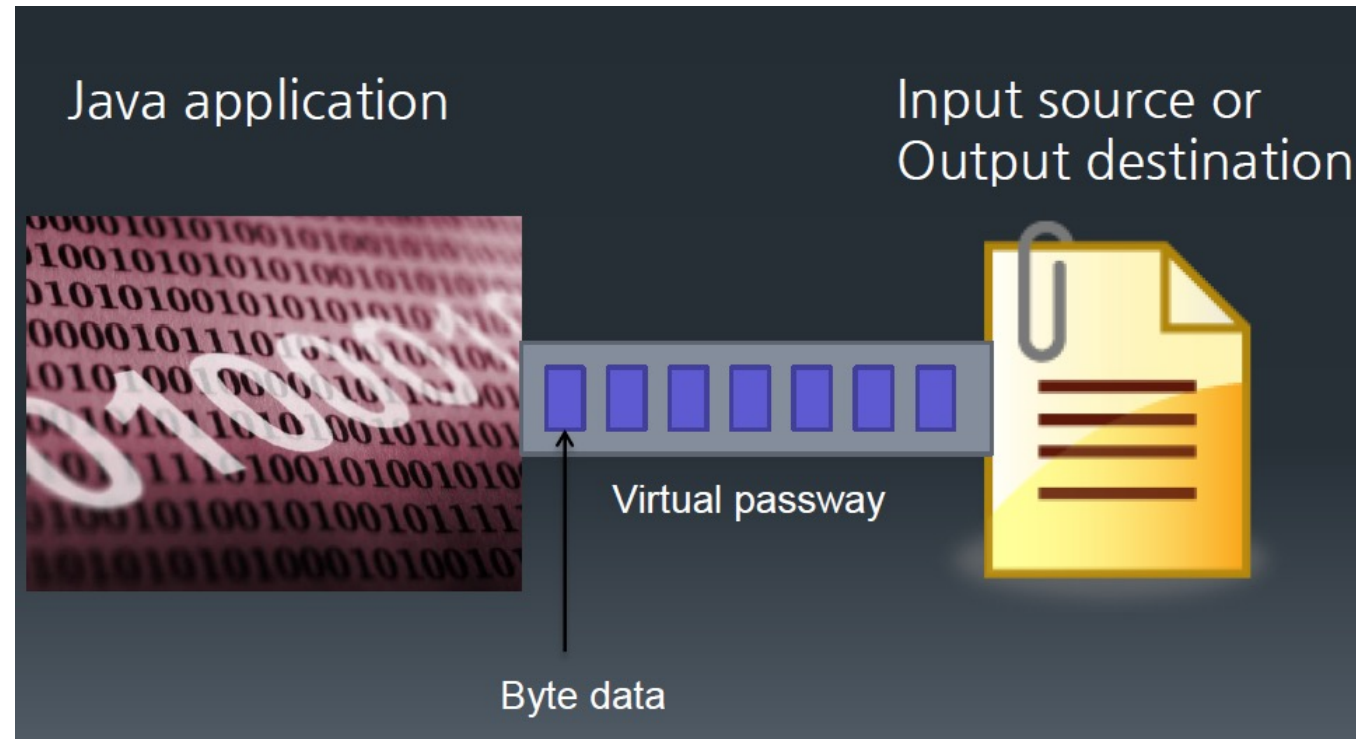


# 15. 자 바 입 출 력

# 스트림이란?

- 네트워크에서 자료의 흐름이 물과 같다는 의미에서 유래
- 다양한 입출력 장치에 독립적으로 일관성 있는 입출력을 제공하는 방식



- 입출력이 구현되는 곳 : 파일 디스크, 키보드, 마우스, 메모리 네트워크 등

# 스트림의 구분

- 대상 기준

입력 스트림 / 출력 스트림

- 자료의 종류

바이트 스트림/ 문자 스트림

- 기능

기반 스트림/ 보조 스트림

# 입력 스트림과 출력 스트림

- 입력 스트림 : 대상으로 부터 자료를 읽어 들이는 스트림
- 출력 스트림 : 대상으로 자료를 출력하는 스트림

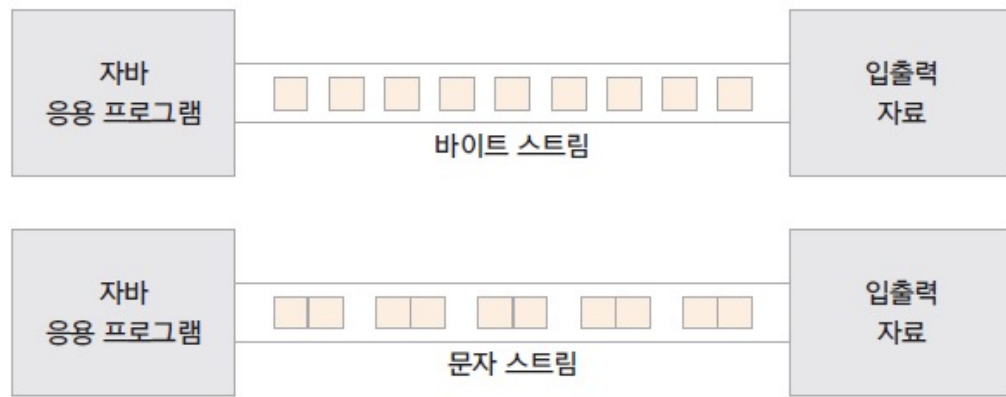


- 스트림의 종류

종류	예시
입력 스트림	FileInputStream, FileReader, BufferedInputStream, BufferedReader 등
출력 스트림	FileOutputStream, FileWriter, BufferedOutputStream, BufferedWriter 등

# 바이트 단위 스트림과 문자단위 스트림

- 바이트 단위 스트림 : 동영상, 음악 파일 등을 읽고 쓸 때 사용
- 문자 단위 스트림 : 바이트 단위로 자료를 처리하면 문자는 깨짐  
2 바이트 단위로 처리하도록 구현된 스트림

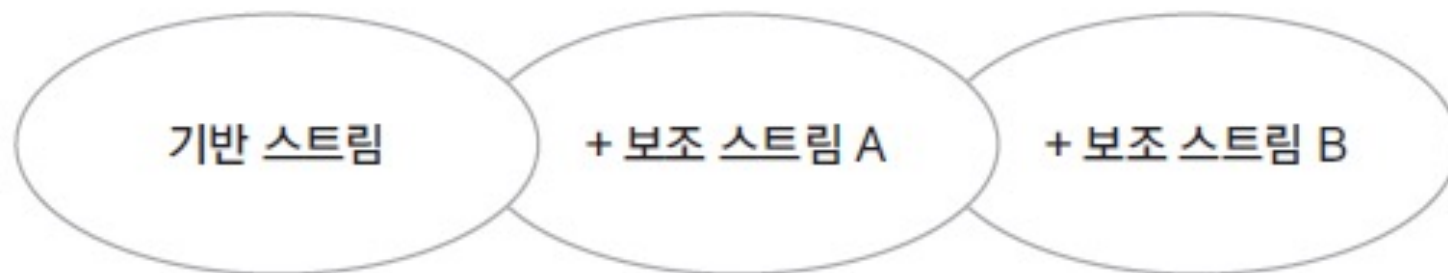


- 스트림의 종류

종류	예시
바이트 스트림	FileInputStream, FileOutputStream, BufferedInputStream, BufferedOutputStream 등
문자 스트림	FileReader, FileWriter, BufferedReader, BufferedWriter 등

# 기반 스트림과 보조 스트림

- 기반 스트림 : 대상에 직접 자료를 읽고 쓰는 기능의 스트림
- 보조 스트림 : 직접 읽고 쓰는 기능은 없이 추가적인 기능을 더해주는 스트림
- 보조 스트림은 직접 읽고 쓰는 기능은 없으므로 항상 기반 스트림이나 또다른 보조 스트림을 생성자 매개변수로 포함 함



- 스트림 종류

종류	예시
기반 스트림	FileInputStream, FileOutputStream, FileReader, FileWriter 등
보조 스트림	InputStreamReader, OutputStreamWriter, BufferedInputStream, BufferedOutputStream 등

# 표준 입출력

- **System** 클래스의 표준 입출력 멤버

```
public class System{  
    public static PrintStream out;  
    public static InputStream in;  
    public static PrintStream err;  
}
```

- System.out

표준 출력(모니터) 스트림

```
System.out.println("에러메시지");
```

- System.in

표준 입력(키보드) 스트림

```
int d = System.in.read(); //한 바이트 읽어내기
```

- System.err

표준 에러 출력(모니터) 스트림

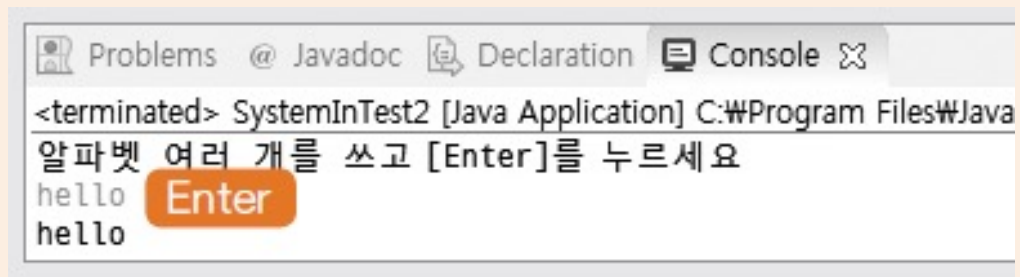
```
System.err.println("데이터");
```

# System.in 사용하기 예제

- 여러 개의 문자를 엔터를 누를때 까지 입력 받기

```
public class SystemInTest2 {  
    public static void main(String[ ] args) {  
        System.out.println("알파벳 여러 개를 쓰고 [Enter]를 누르세요");  
  
        int i;  
        try {  
            while((i = System.in.read( )) != -1) {  
                System.out.print((char)i);  
            }  
        } catch (IOException e) {  
            e.printStackTrace( );  
        }  
    }  
}
```

while문에서 read( ) 메서드로  
한 바이트를 반복해 읽음





# Scanner 클래스

- java.util 패키지에 있는 입력 클래스
- 문자뿐 아니라 정수, 실수 등 다른 자료형도 읽을 수 있음
- 여러 대상에서 자료를 읽을 수 있음 (콘솔, 파일 등)
- 생성자

생성자	설명
Scanner(File source)	파일을 매개변수로 받아 Scanner를 생성합니다.
Scanner(InputStream source)	바이트 스트림을 매개변수로 받아 Scanner를 생성합니다.
Scanner(String source)	String을 매개변수로 받아 Scanner를 생성합니다.

# Scanner 클래스

- 메서드

메서드	설명
<code>boolean nextBoolean( )</code>	boolean 자료를 읽습니다.
<code>byte nextByte( )</code>	한 바이트 자료를 읽습니다.
<code>short nextShort( )</code>	short 자료형을 읽습니다.
<code>int nextInt( )</code>	int 자료형을 읽습니다.
<code>long nextLong( )</code>	long 자료형을 읽습니다.
<code>float nextFloat( )</code>	float 자료형을 읽습니다.
<code>double nextDouble( )</code>	double 자료형을 읽습니다.
<code>String nextLine( )</code>	문자열 String을 읽습니다.

# Scanner 클래스 예제

```
public class ScannerTest {  
    public static void main(String[ ] args) {  
        Scanner scanner = new Scanner(System.in);
```

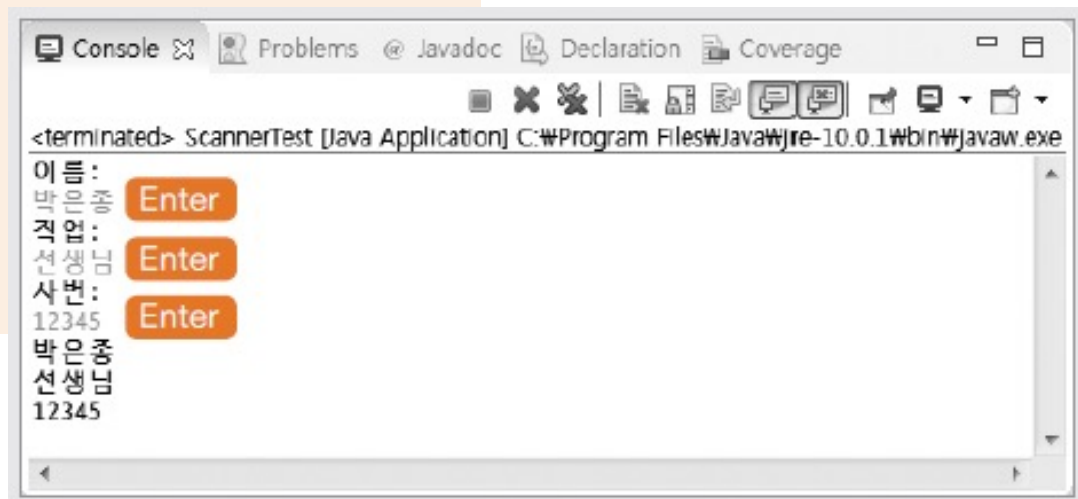
```
        System.out.println("이름:");  
        String name = scanner.nextLine( );  
        System.out.println("직업:");  
        String job = scanner.nextLine( );
```

문자열을 읽는 nextLine( ) 메서드로  
이름과 직업 입력받음

```
        System.out.println("사번:");  
        int num = scanner.nextInt( );
```

int형을 읽는 nextInt( ) 메서드로 사번을  
입력받음

```
        System.out.println(name);  
        System.out.println(job);  
        System.out.println(num);  
    }  
}
```



# Console 클래스

- System.in 을 사용하지 않고 콘솔에서 표준 입력을 할 수 있음
- 이클립스와는 연동되지 않음
- command창에서 입력 함
- 메서드

메서드	설명
String readLine( )	문자열을 읽습니다.
char[ ] readPassword( )	사용자에게 문자열을 보여 주지 않고 읽습니다.
Reader reader( )	Reader 클래스를 반환합니다.
PrintWriter writer( )	PrintWriter 클래스를 반환합니다.

# Console 클래스 예제

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

```
    Console console = System.console( );
```

콘솔 클래스 반환

```
    System.out.println("이름:");
```

```
    String name = console.readLine( );
```

```
    System.out.println("직업:");
```

```
    String job = console.readLine( );
```

```
    System.out.println("비밀번호:");
```

```
    char[ ] pass = console.readPassword( );
```

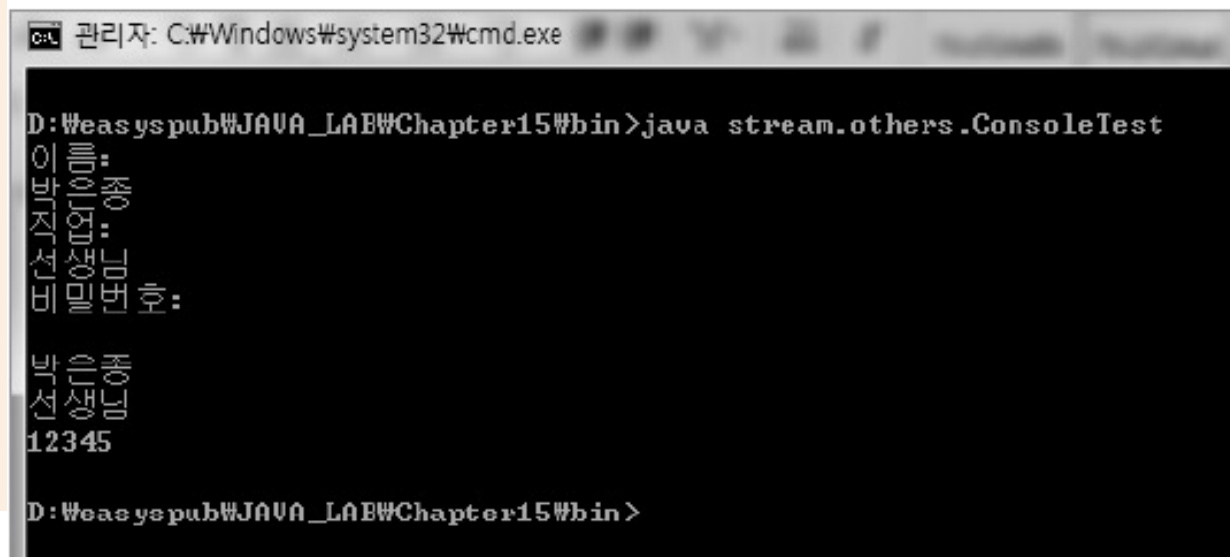
```
    String strPass = new String(pass);
```

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

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

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

```
}
```



```
관리자: C:\Windows\system32\cmd.exe
D:\Weasyspub\JAVA_LAB\Chapter15\bin>java stream.others.ConsoleTest
이름: 박선생
직업: 교수
비밀번호: *
이름: 박선생
직업: 교수
비밀번호: 12345
D:\Weasyspub\JAVA_LAB\Chapter15\bin>
```



# 바이트 단위 스트림-InputStream

- 바이트 단위 입력용 최상위 스트림
- 추상 메서드를 포함한 추상 클래스로 하위 클래스가 구현하여 사용
- 주요 하위 클래스

스트림 클래스	설명
FileInputStream	파일에서 바이트 단위로 자료를 읽습니다.
ByteArrayInputStream	Byte 배열 메모리에서 바이트 단위로 자료를 읽습니다.
FilterInputStream	기반 스트림에서 자료를 읽을 때 추가 기능을 제공하는 보조 스트림의 상위 클래스입니다(보조 스트림은 '15-5 보조 스트림'에서 자세히 설명합니다).

- 메서드

메서드	설명
int read( )	입력 스트림으로부터 한 바이트의 자료를 읽습니다. 읽은 자료의 바이트 수를 반환합니다.
int read(byte b[ ])	입력 스트림으로부터 b[ ] 크기의 자료를 b[ ]에 읽습니다. 읽은 자료의 바이트 수를 반환합니다.
int read(byte b[ ], int off, int len)	입력 스트림으로부터 b[ ] 크기의 자료를 b[ ]의 off 변수 위치부터 저장하며 len만큼 읽습니다. 읽은 자료의 바이트 수를 반환합니다.
void close( )	입력 스트림과 연결된 대상 리소스를 닫습니다. (예 : FileInputStream인 경우 파일 닫음)

# FileInputStream 클래스

- 파일로 부터 자료를 읽어 들이는 스트림
- 읽어들이는 파일이 경로에 없는 경우 예외가 발생

```
public class FileInputStreamTest1 {  
    public static void main(String[ ] args) {  
        FileInputStream fis = null;
```

```
        try {
```

```
            fis = new FileInputStream("input.txt");
```

input.txt 파일 입력 스트림 생성

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

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

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

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

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

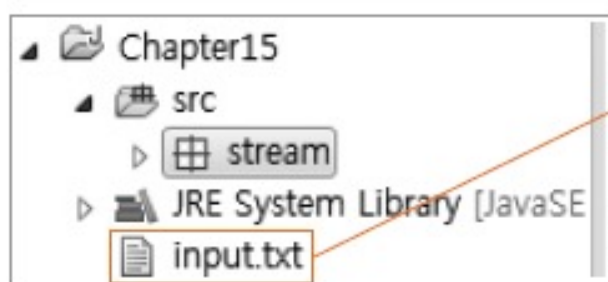
```
        } finally {
```

# FileInputStream 클래스

```
try {  
    fis.close( );  
} catch (IOException e) {  
    System.out.println(e);  
} catch (NullPointerException e){  
    System.out.println(e);  
}  
}  
System.out.println("end");  
}
```

열린 스트림은 finally 블록에서 닫음

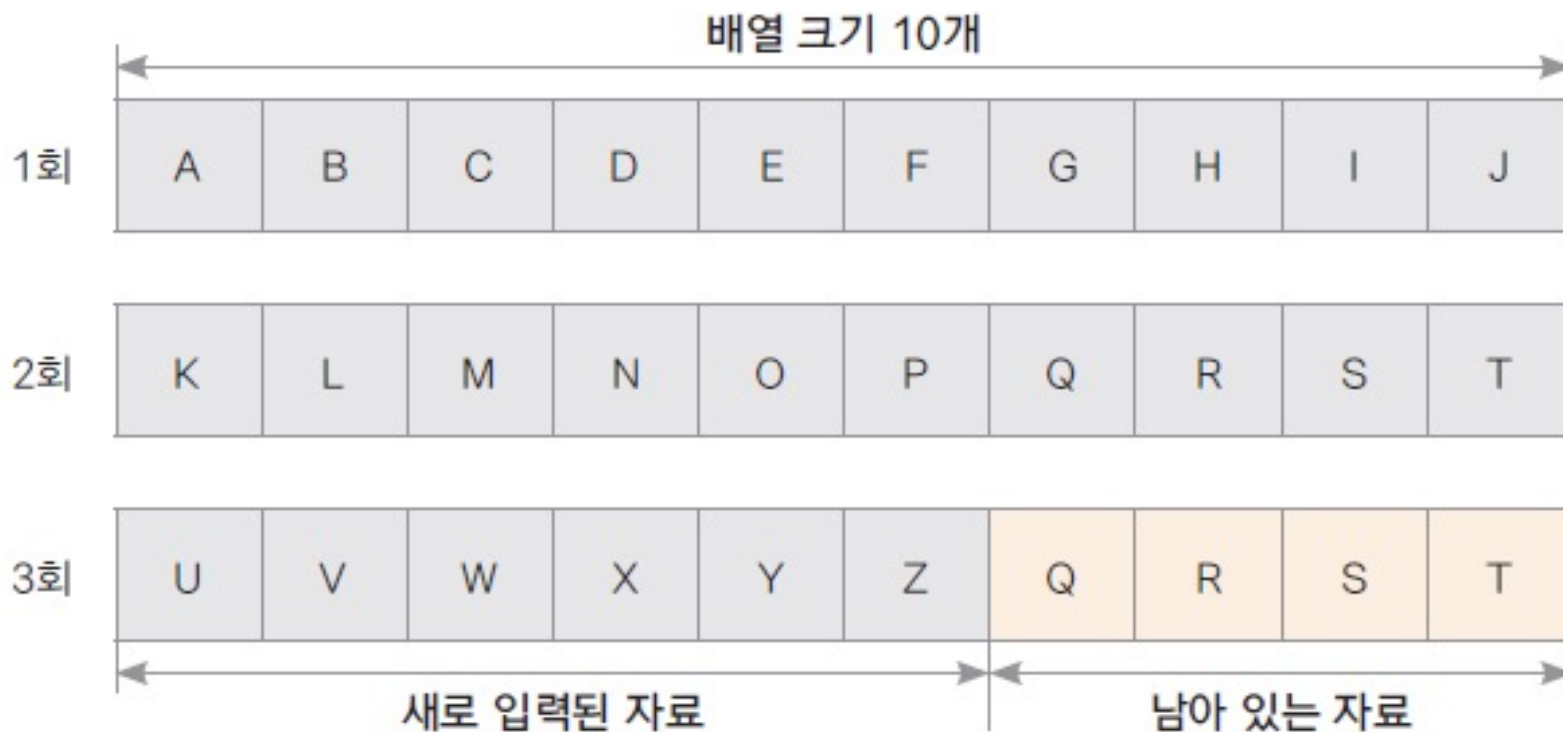
스트림이 null인 경우





# byte 배열로 읽기

- A-Z까지의 알파벳을 10 크기의 배열로 읽을 때 맨 마지막에는 자료가 남게 된다.



# byte 배열로 읽기

```
public class FileInputStreamTest3 {  
    public static void main(String[] args) {  
        try(FileInputStream fis = new FileInputStream("input2.txt")) {  
            byte[] bs = new byte[10] ;  
            int i;  
            while((i = fis.read(bs)) != -1) {  
                for(byte b : bs) {  
                    System.out.print((char)b);  
                }  
                System.out.println(": " + i + "바이트 읽음");  
            }  
        } catch (IOException e) {  
            e.printStackTrace( );  
        }  
        System.out.println("end");  
    }  
}
```



```
for(int k = 0; k < i; k++) {  
    System.out.print((char)bs[k]);  
}
```

읽어 들인 개수만큼만 출력함

```
Console Problems @ Javadoc  
  
<terminated> FileInputStreamTest3 [Java Applica  
ABCDEFGHIJ: 10바이트 읽음  
KLMNOPQRST: 10바이트 읽음  
UVWXYZ: 6바이트 읽음  
end
```

# 바이트 단위 스트림-InputStream

- 바이트 단위 출력용 최상위 스트림
- 추상 메서드를 포함한 추상 클래스로 하위 클래스가 구현하여 사용
- 주요 하위 클래스

스트림 클래스	설명
FileOutputStream	바이트 단위로 파일에 자료를 씁니다.
ByteArrayOutputStream	Byte 배열에 바이트 단위로 자료를 씁니다.
FilterOutputStream	기반 스트림에서 자료를 쓸 때 추가 기능을 제공하는 보조 스트림의 상위 클래스입니다.

- 메서드

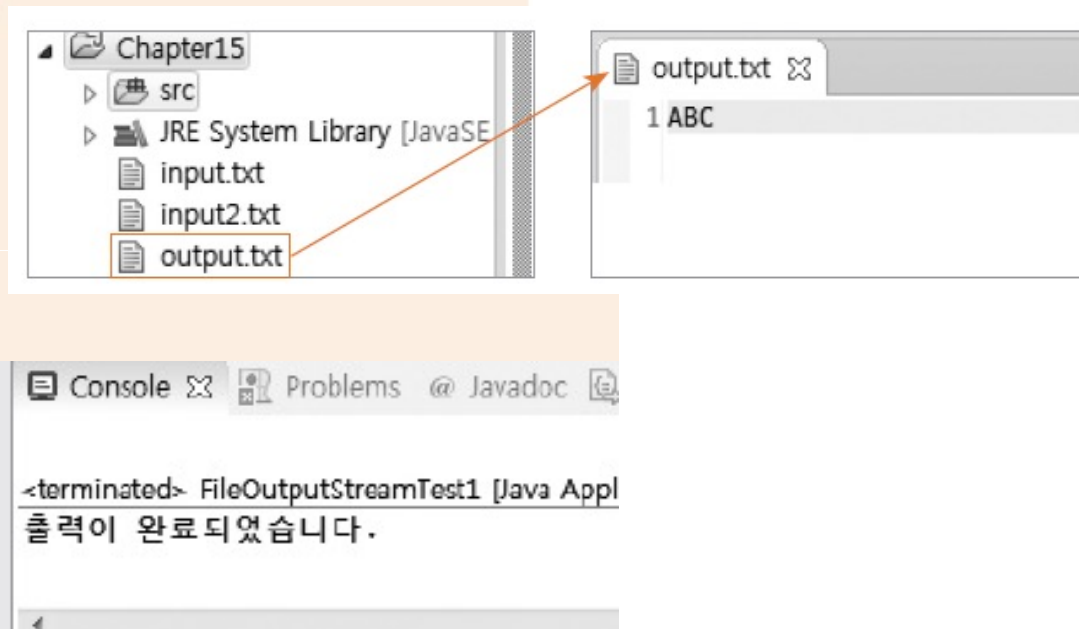
메서드	설명
void write(int b)	한 바이트를 출력합니다.
void write(byte[ ] b)	b[ ] 배열에 있는 자료를 출력합니다.
void write(byte b[ ] , int off, int len)	b[ ] 배열에 있는 자료의 off 위치부터 len 개수만큼 자료를 출력합니다.
void flush( )	출력을 위해 잠시 자료가 머무르는 출력 버퍼를 강제로 비워 자료를 출력합니다.
void close( )	출력 스트림과 연결된 대상 리소스를 닫습니다. 출력 버퍼가 비워집니다. (예 : FileOutputStream인 경우 파일 닫음)

# FileOutputStream 클래스

- 파일에 자료를 출력하는 스트림
- 출력하려는 파일이 경로에 없는 경우 생성 함

```
public class FileOutputStreamTest1 {  
    public static void main(String[ ] args) {  
        try(FileOutputStream fos = new FileOutputStream("output.txt")) {  
            fos.write(65);  
            fos.write(66);  
            fos.write(67);  
        } catch(IOException e) {  
            e.printStackTrace( );  
        }  
  
        System.out.println("출력이 완료되었습니다.");  
    }  
}
```

FileOutputStream은 파일에 숫자를 쓰면  
해당하는 아스키 코드 값으로 변환됨



# 바이트 배열로 출력하기

```
public class FileOutputStreamTest2 {  
    public static void main(String[] args) throws IOException {  
        FileOutputStream fos = new FileOutputStream("output2.txt", true);  
        try(fos) {  
            byte[] bs = new byte[26];  
            byte data = 65; // 'A'의 아스키 값  
            for(int i = 0; i < bs.length; i++) {  
                bs[i] = data;  
                data++;  
            }  
            fos.write(bs); // 배열을 한꺼번에 출력  
        } catch(IOException e) {  
            e.printStackTrace();  
        }  
        System.out.println("출력이 완료되었습니다.");  
    }  
}
```

자바 9부터 제공하는 향상된 try-with-resources문

A부터 Z까지 배열에 넣기

output2.txt

1 ABCDEFGHIJKLMNOPQRSTUVWXYZ

# flush() 와 close() 메서드

- 출력 버퍼를 비울때 flush() 메서드를 사용
- close() 메서드 내부에서 flush() 가 호출되므로 close() 메서드가 호출되면 출력버퍼가 비워 짐

# 문자 단위 스트림 - Reader

- 문자 단위로 읽는 최상위 스트림
- 하위 클래스에서 상속 받아 구현 함

스트림 클래스	설명
FileReader	파일에서 문자 단위로 읽는 스트림 클래스입니다.
InputStreamReader	바이트 단위로 읽은 자료를 문자로 변환해 주는 보조 스트림 클래스입니다.
BufferedReader	문자로 읽을 때 배열을 제공하여 한꺼번에 읽을 수 있는 기능을 제공해 주는 보조 스트림입니다.

- 주요 메서드

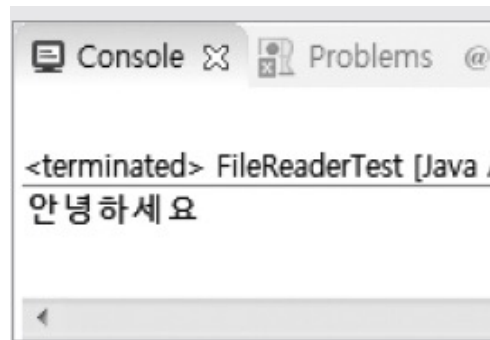
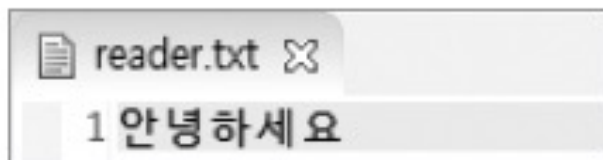
메서드	설명
int read( )	파일로부터 한 문자를 읽습니다. 읽은 값을 반환합니다.
int read(char[ ] buf)	파일로부터 buf 배열에 문자를 읽습니다.
int read(char[ ] buf, int off, int len)	파일로부터 buf 배열의 off 위치에서부터 len 개수만큼 문자를 읽습니다.
void close( )	스트림과 연결된 파일 리소스를 닫습니다.



# FileReader 클래스

- 한글 파일 읽기

```
public class FileReaderTest {  
    public static void main(String[ ] args) {  
        try(FileReader fr = new FileReader("reader.txt")) {  
            int i;  
            while((i = fr.read( )) != -1) {  
                System.out.print((char)i);  
            }  
        } catch (IOException e) {  
            e.printStackTrace( );  
        }  
    }  
}
```





# 문자 단위 스트림 - Writer

- 문자 단위로 쓰는 최상위 스트림
- 하위 클래스에서 상속 받아 구현 함

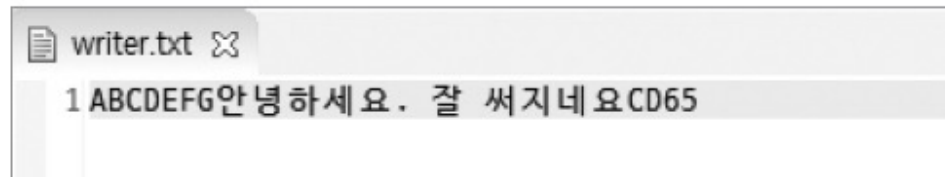
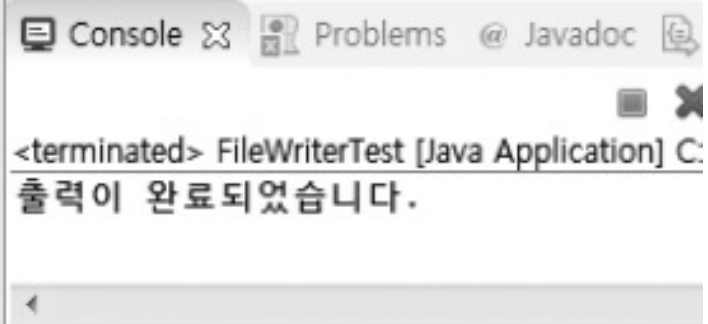
스트림 클래스	설명
FileWriter	파일에 문자 단위로 출력하는 스트림 클래스입니다.
OutputStreamWriter	파일에 바이트 단위로 출력한 자료를 문자로 변환해 주는 보조 스트림입니다.
BufferedWriter	문자로 쓸 때 배열을 제공하여 한꺼번에 쓸 수 있는 기능을 제공해 주는 보조 스트림입니다.

- 주요 메서드

메서드	설명
void write(int c)	한 문자를 파일에 출력합니다.
void write(char[ ] buf)	문자 배열 buf의 내용을 파일에 출력합니다.
void write(char[ ] buf, int off, int len)	문자 배열 buf의 off 위치에서부터 len 개수의 문자를 파일에 출력합니다.
void write(String str)	문자열 str를 파일에 출력합니다.
void write(String str, int off, int len)	문자열 str의 off번째 문자부터 len 개수만큼 파일에 출력합니다.
void flush( )	파일에 출력하기 전에 자료가 있는 공간(출력 버퍼)을 비워 출력합니다.
void close( )	파일과 연결된 스트림을 닫습니다. 출력 버퍼도 비워집니다.

# FileWriter 클래스

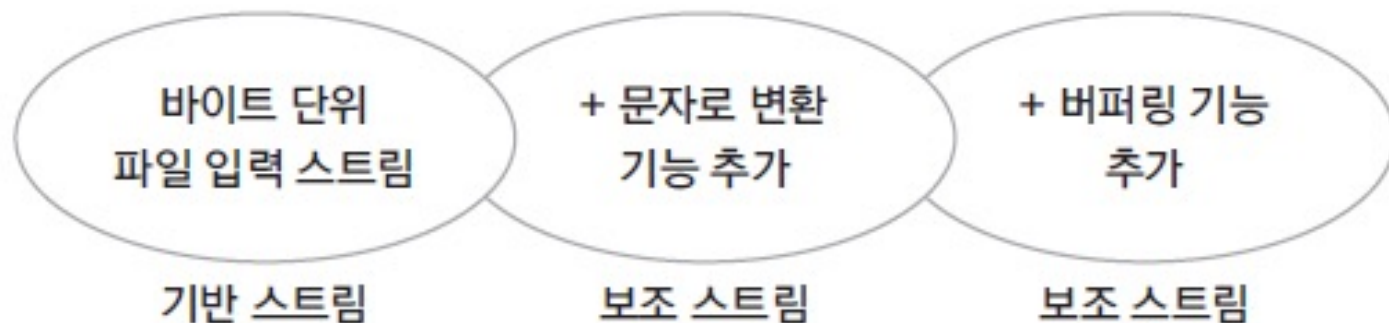
```
public class FileWriterTest {  
    public static void main(String[] args) {  
        try(FileWriter fw = new FileWriter("writer.txt")) {  
            fw.write('A');                // 문자 하나 출력  
            char buf[] = {'B', 'C', 'D', 'E', 'F', 'G'};  
  
            fw.write(buf);                // 문자 배열 출력  
            fw.write("안녕하세요. 잘 쓰지네요"); // 문자열 출력  
            fw.write(buf, 1, 2);          // 문자 배열의 일부 출력  
            fw.write("65");               // 숫자를 그대로 출력  
        } catch(IOException e) {  
            e.printStackTrace();  
        }  
        System.out.println("출력이 완료되었습니다.");  
    }  
}
```



# 보조 스트림

- 실제 읽고 쓰는 스트림이 아닌 보조적인 기능을 추가하는 스트림
- 데코레이터 패턴
- `FilterInputStream`과 `FilterOutputStream`이 보조스트림의 상위 클래스
- 생성자의 매개 변수로 또 다른 스트림을 가짐

생성자	설명
<code>protected FilterInputStream(InputStream in)</code>	생성자의 매개변수로 <code>InputStream</code> 을 받습니다.
<code>public FilterOutputStream(OutputStream out)</code>	생성자의 매개변수로 <code>OutputStream</code> 을 받습니다.



# InputStreamReader와 OutputStreamWriter

- 바이트 단위로 읽거나 쓰는 자료를 문자로 변환해주는 보조 스트림
- FileInputStream(바이트 스트림)으로 읽은 자료를 문자와 변환하는 예

```
public class InputStreamReaderTest {  
    public static void main(String[ ] args) {  
        try(InputStreamReader isr = new InputStreamReader(new FileInputStream("reader.  
txt"))) {  
  
            int i;  
            while((i = isr.read( )) != -1) {  
                System.out.print((char)i);  
            }  
        } catch (IOException e) {  
            e.printStackTrace( );  
        }  
    }  
}
```

보조 스트림인 InputStreamReader의 매개변수로  
기본 스트림인 FileInputStream을 받아 생성함

파일의 끝인 -1이 반환될 때까지  
보조 스트림으로 자료를 읽음

Console Problems @ Javac  
<terminated> InputStreamReaderTest [Java  
안녕하세요

# Buffered 스트림

- 내부적으로 8192 바이트 배열을 가지고 읽거나 쓰기 기능을 제공하여 속도가 빨라짐

스트림 클래스	설명
BufferedInputStream	바이트 단위로 읽는 스트림에 버퍼링 기능을 제공합니다.
BufferedOutputStream	바이트 단위로 출력하는 스트림에 버퍼링 기능을 제공합니다.
BufferedReader	문자 단위로 읽는 스트림에 버퍼링 기능을 제공합니다.
BufferedWriter	문자 단위로 출력하는 스트림에 버퍼링 기능을 제공합니다.



# 버퍼링 기능으로 파일 복사하기

```
public class BufferedStreamTest {  
    public static void main(String[ ] args) {  
        long millisecond = 0;  
        try(FileInputStream fis = new FileInputStream("a.zip");  
            FileOutputStream fos = new FileOutputStream("copy.zip");  
            BufferedInputStream bis = new BufferedInputStream(fis);  
            BufferedOutputStream bos = new BufferedOutputStream(fos)) {  
            millisecond = System.currentTimeMillis( );  
            int i;  
            while(( i = bis.read( )) != -1){  
                bos.write(i);  
            }  
            millisecond = System.currentTimeMillis( ) - millisecond;  
        } catch(IOException e) {  
            e.printStackTrace( );  
        }  
        System.out.println("파일 복사하는 데" + millisecond + " milliseconds 소요되었습니다.");  
    }  
}
```

Console Problems @ Javadoc Declaration Coverage

<terminated> BufferedStreamTest [Java Application] C:\Program Files\Java\W  
파일 복사하는 데 79 milliseconds 소요되었습니다.

# DataInputStream과 DataOutputStream

- 자료가 메모리에 저장된 0, 1 상태 그대로 읽거나 쓰는 스트림
- 읽는 메서드

메서드	설명
byte readByte( )	1바이트를 읽어 반환합니다.
boolean readBoolean( )	읽은 자료가 0이 아니면 true를, 0이면 false를 반환합니다.
char readChar( )	한 문자를 읽어 반환합니다.
short readShort( )	2바이트를 읽어 정수 값을 반환합니다.
int readInt( )	4바이트를 읽어 정수 값을 반환합니다.
long readLong( )	8바이트를 읽어 정수 값을 반환합니다.
float readFloat( )	4바이트를 읽어 실수 값을 반환합니다.
double readDouble( )	8바이트를 읽어 실수 값을 반환합니다.
String readUTF( )	수정된 UTF-8 인코딩 기반으로 문자열을 읽어 반환합니다.

# DataInputStream과 DataOutputStream

- 쓰는 메서드

메서드	설명
<code>void writeByte(int v)</code>	1바이트의 자료를 출력합니다.
<code>void writeBoolean(boolean v)</code>	1바이트 값을 출력합니다.
<code>void writeChar(int v)</code>	2바이트를 출력합니다.
<code>void writeShort(int v)</code>	2바이트를 출력합니다
<code>void writeInt(int v)</code>	4바이트를 출력합니다.
<code>void writeLong(long v)</code>	8바이트를 출력합니다.
<code>void writeFloat(float v)</code>	4바이트를 출력합니다.
<code>void writeDouble(double v)</code>	8바이트를 출력합니다.
<code>void writeUTF(String str)</code>	수정된 UTF-8 인코딩 기반으로 문자열을 출력합니다.



# 직렬화(serialization)

- 인스턴스의 상태를 그대로 저장 하거나(serialization) 다시 복원하는 (deserialization) 방식
- 파일에 쓰거나 네트워크로 전송
- `ObjectInputStream`과 `ObjectOutputStream` 사용

생성자	설명
<code>ObjectInputStream(InputStream in)</code>	<code>InputStream</code> 을 생성자의 매개변수로 받아 <code>ObjectInputStream</code> 을 생성합니다.
<code>ObjectOutputStream(OutputStream out)</code>	<code>OutputStream</code> 을 생성자의 매개변수로 받아 <code>ObjectOutputStream</code> 을 생성합니다

# Serializable 인터페이스

- 직렬화는 인스턴스 내용이 외부로 유출되는 것이므로 프로그래머가 직렬화 의도를 표시해야함
- 구현코드가 없는 maker interface

```
class Person implements Serializable {
```

```
...
```

```
String name;
```

```
String job;
```

```
...
```

```
}
```

직렬화하겠다는 의도를 표시

# externalizable 인터페이스

- 프로그래머가 자료를 읽고 쓰는 방식을 직접 구현 함

```
public Dog( ) { }
```

Externalizable 인터페이스의 메서드 구현

```
@Override
```

```
public void writeExternal(ObjectOutput out) throws IOException {  
    out.writeUTF(name);  
}
```

```
@Override
```

```
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {  
    name = in.readUTF( );  
}
```

# 그 외 입출력 클래스

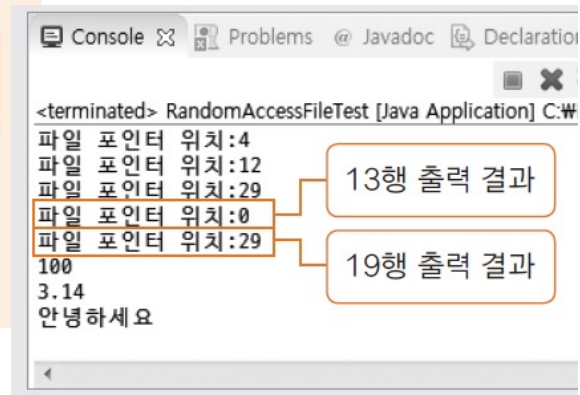
- File 클래스  
파일개념을 추상화한 클래스  
입출력 기능은 없고 파일의 속성, 경로, 이름 등을 알 수 있음
- RandomAccessFile 클래스  
입출력 클래스 중 유일하게 파일 입출력을 동시에 할 수 있는 클래스  
파일 포인터가 있어서 읽고 쓰는 위치의 이동이 가능함  
다양한 자요형에 대한 메서드가 제공 됨

# RandomAccessFile 예제

```
public class RandomAccessFileTest {  
    public static void main(String[] args) throws IOException {  
        ...  
        rf.writeUTF("안녕하세요");  
        System.out.println("파일 포인터 위치:" + rf.getFilePointer( ));  
  
        rf.seek(0);  
        System.out.println("파일 포인터 위치:" + rf.getFilePointer( ));  
  
        int i = rf.readInt( );  
        double d = rf.readDouble( );  
        String str = rf.readUTF( );  
  
        System.out.println("파일 포인터 위치:" + rf.getFilePointer( ));  
  
        System.out.println(i);  
        System.out.println(d);  
        System.out.println(str);  
    }  
}
```

파일 포인터 위치를 맨 처음으로 옮기고 위치를 출력함

읽기가 끝난 후 파일 포인터 위치를 출력함



```
<terminated> RandomAccessFileTest [Java Application] C:\W  
파일 포인터 위치:4  
파일 포인터 위치:12  
파일 포인터 위치:29  
파일 포인터 위치:0  
파일 포인터 위치:29  
100  
3.14  
안녕하세요
```

13행 출력 결과

19행 출력 결과

감사합니다.

끝