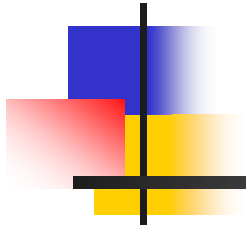


## *Lecture 9*

# 입출력 프로그래밍




2018년도 2학기

컴퓨터프로그래밍2

김 영 국

충남대학교 컴퓨터공학과

# 이번 주에 학습할 내용

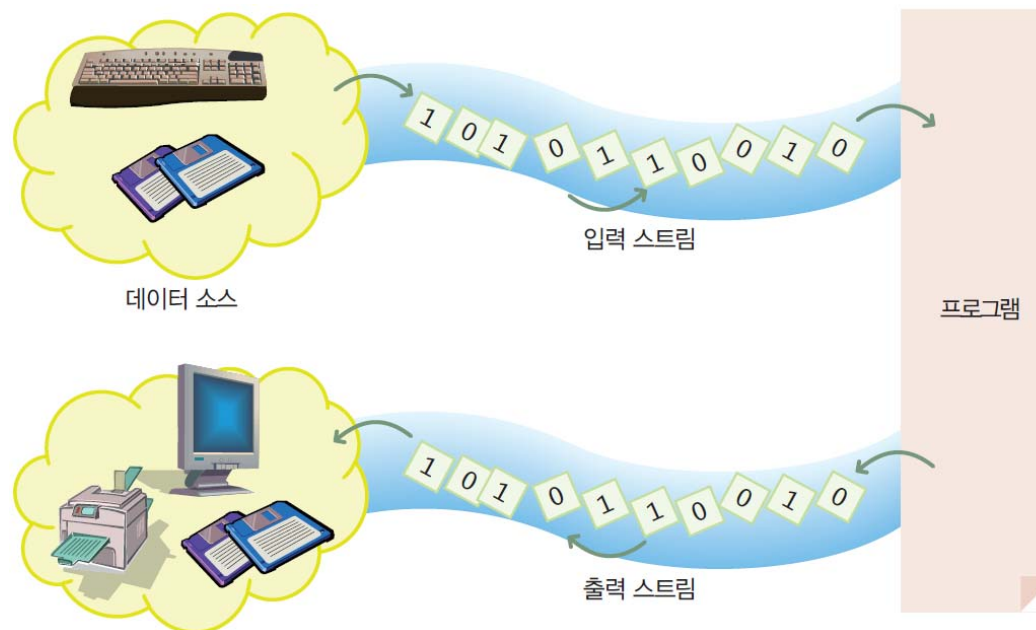
- 
- 스트림이란?
  - 스트림의 분류
  - 바이트 스트림
  - 문자 스트림
  - 형식 입출력
  - 명령어행에서 입출력
  - 파일 입출력

스트림을  
이용한  
입출력에  
대하여  
살펴봅시다.



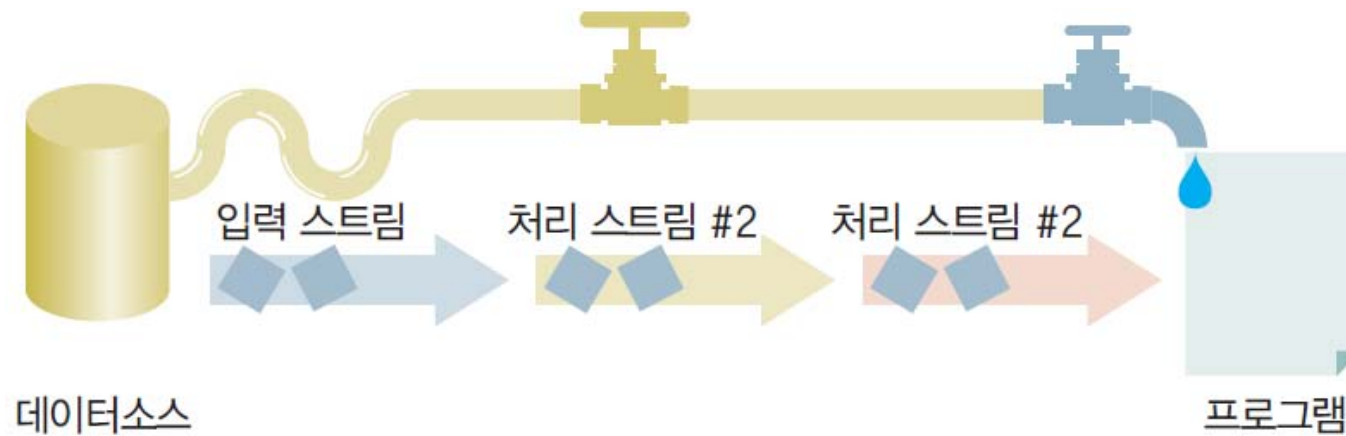
# 스트림(Stream)

- 스트림(stream)은 "순서가 있는 데이터의 연속적인 흐름"이다.
- 스트림은 입출력을 물의 흐름처럼 간주하는 것이다.



< 스트림의 개념 >

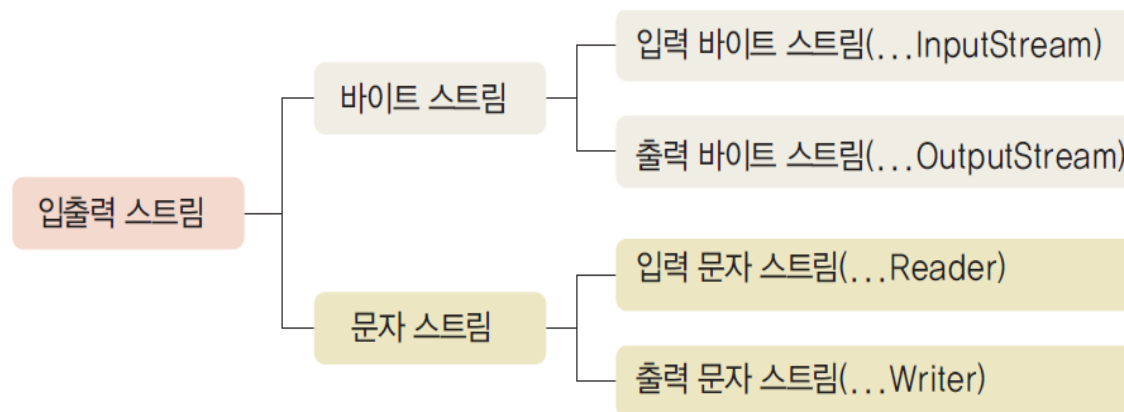
# 스트림들은 연결될 수 있다.



< 스트림은 연결될 수 있다. >

# 스트림의 종류

- 입출력의 단위에 따라서 분류



< 스트림의 분류 >



# 바이트 스트림과 문자 스트림

---

- **바이트 스트림(byte stream):** 8비트의 바이트 단위로 입출력하는 클래스
  - 이진 데이터를 읽고 쓰기 위하여 사용
  - 클래스 이름에 InputStream(입력)과 OutputStream(출력)이 붙는다.
- **문자 스트림(character stream):** 문자 단위로 입출력하는 클래스
  - 문자 스트림은 유니코드 단위로 입출력한다.
  - 클래스 이름에 Reader(입력)와 Writer(출력)가 붙는다.

# 중간 점검 문제

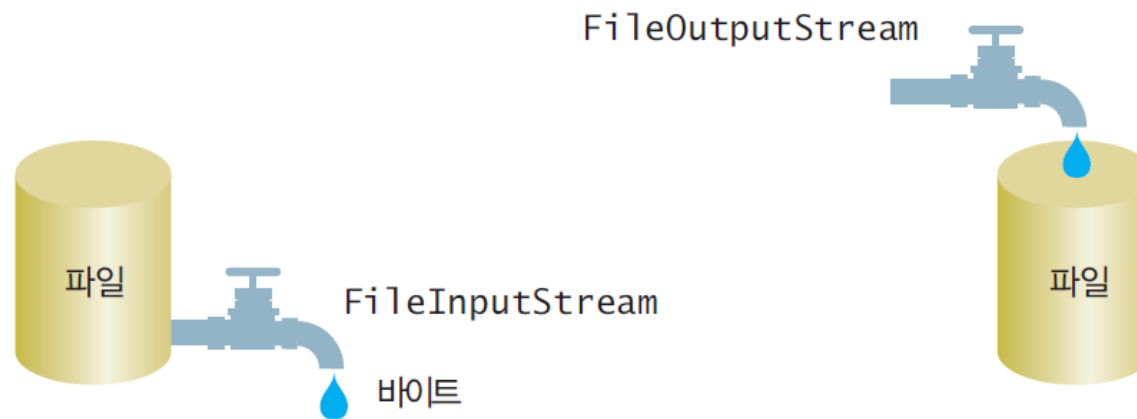


## 중간점검

1. 자바에서는 입출력을 무엇이라고 추상화하는가?
2. 스트림은 \_\_\_\_\_가 있는 데이터의 \_\_\_\_\_인 흐름이다.
3. 문자 스트림과 바이트 스트림의 차이점은 무엇인가?

# 바이트 스트림

- 8비트의 바이트 단위로 입출력을 수행하는 스트림
- 모든 바이트 스트림은 InputStream과 OutputStream에서 파생된다.



< 파일 입출력 바이트 스트림 >



# 파일 입출력 바이트 스트림

CopyFile1.java

```
01  ...
02  public class CopyFile1 {
03      public static void main(String[] args) throws IOException {
04
05          FileInputStream in = null;
06          FileOutputStream out = null;
07
08          try {
09              in = new FileInputStream("input.txt");
10              out = new FileOutputStream("output.txt");
11              int c;
12
13              while ((c = in.read()) != -1) {
14                  out.write(c);
15              }
16          } finally {
17              if (in != null) {
18                  in.close();
19              }
20              if (out != null) {
21                  out.close();
22              }
23          }
24      }
25  }
```

input.txt 파일에 연결된 파일  
입력 스트림을 생성한다.

output.txt 파일에 연결된 파일  
출력 스트림을 생성한다.

하나의 바이트를 읽을 때는  
read()를 사용하고 하나의 바이트  
를 쓸 때는 write()를 사용한다.



# 예제

## 실행결과

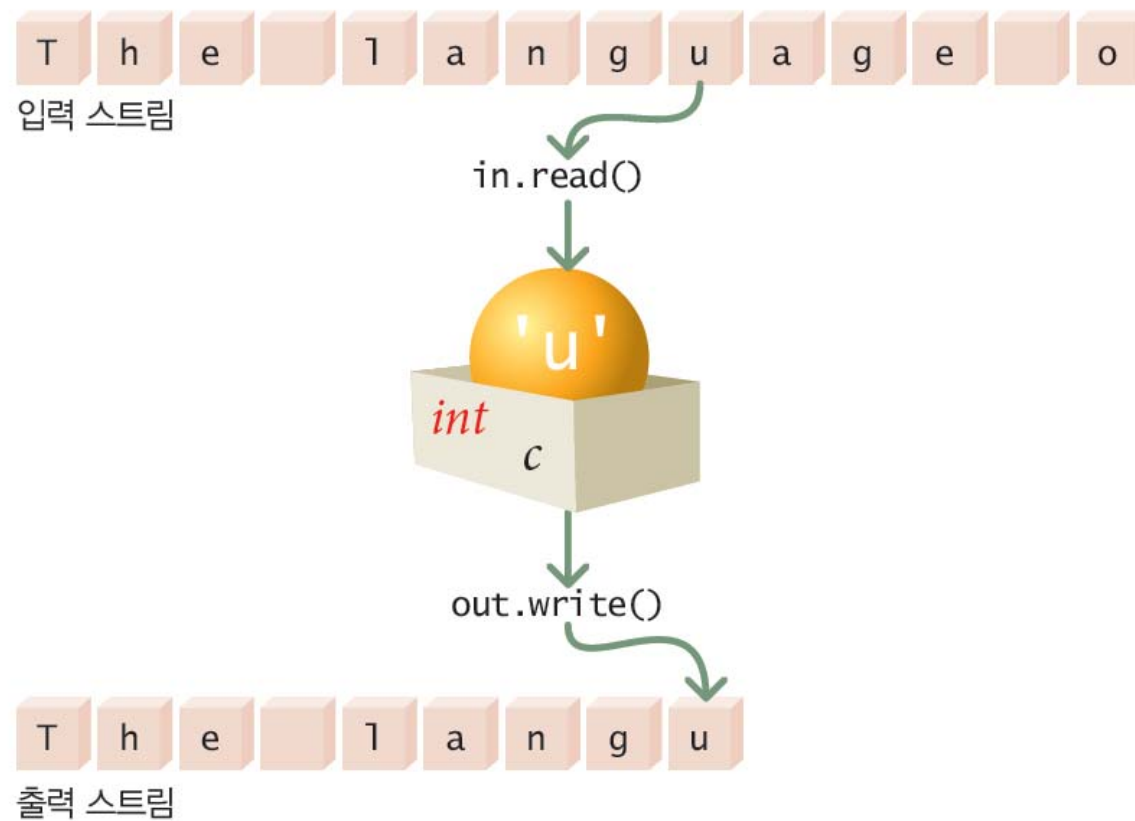
*input.txt*

The language of truth is simple.  
Easier said than done.  
First think and speak.  
Translators, traitors.  
No smoke with~~out~~ fire.

*output.txt*

The language of truth is simple.  
Easier said than done.  
First think and speak.  
Translators, traitors.  
No smoke with~~out~~ fire.

# 예제



# 문자 스트림

- 문자 스트림은 입출력 단위가 문자
- 자바 플랫폼은 유니코드를 사용해서 문자를 저장한다.
- 문자 스트림은 자동적으로 이 유니코드 문자를 지역 문자 집합으로 변환한다.
- 파일 문자 스트림을 살펴보자.



< 문자 스트림의 개념 >

# 문자 스트림

FileCopy2.java

```
01 ...
02 public class FileCopy2 {
03     public static void main(String[] args) throws IOException {
04
05         FileReader inputStream = null;
06         FileWriter outputStream = null;
07
08         try {
09             inputStream = new FileReader("input.txt");
10             outputStream = new FileWriter("output.txt");
11
12             int c;
13             while ((c = inputStream.read()) != -1) {
14                 outputStream.write(c);
15             }
16         } finally {
17             if (inputStream != null) {
18                 inputStream.close();
19             }
20             if (outputStream != null) {
21                 outputStream.close();
22             }
23         }
24     }
25 }
```

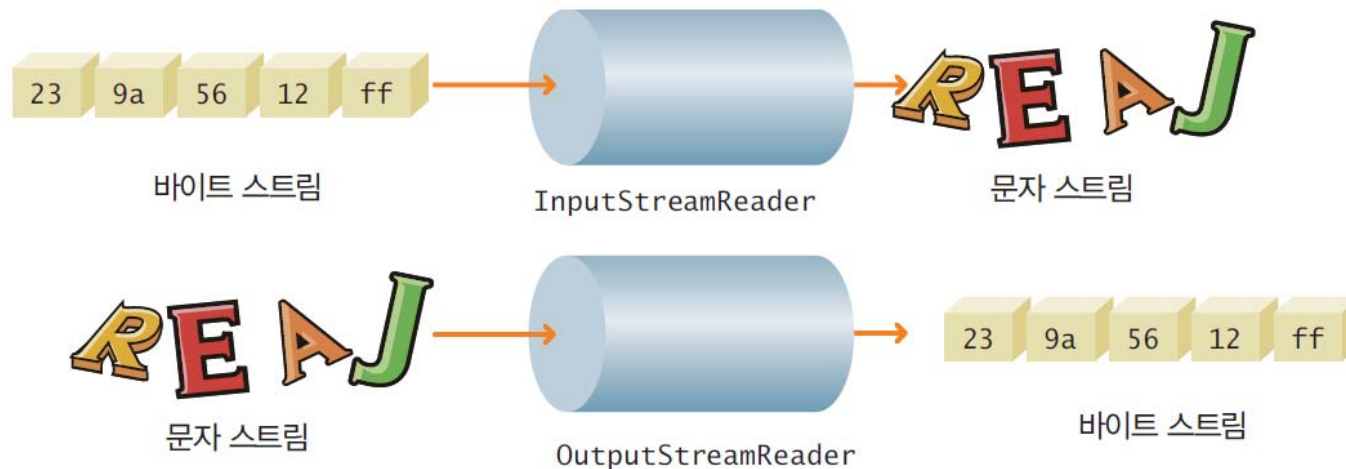
input.txt 파일에 연결된 파일  
입력 스트림을 생성한다.

output.txt 파일에 연결된  
파일 출력 스트림을 생성

하나의 문자를 읽을 때는 read()를 사용하고  
하나의 문자를 쓸 때는 write()를 사용한다.

# InputStreamReader 클래스와 OutputStreamWriter 클래스

- 바이트 스트림과 문자 스트림을 연결하는 클래스



< 브릿지 스트림 >

# 줄 단위의 입출력

CopyLines.java

```
01 ...
02
03 public class CopyLines {
04     public static void main(String[] args) throws IOException {
05
06         BufferedReader inputStream = null;
07         PrintWriter outputStream = null;
08
09         try {
10             inputStream = new BufferedReader(new FileReader("input.txt"));
11             outputStream = new PrintWriter(new FileWriter("output.txt"));
12
13             String l;
14             while ((l = inputStream.readLine()) != null) {
15                 outputStream.println(l);
16             }
17         } finally {
```

FileReader에  
BufferedReader  
를 연결한다.

FileWriter의 출력이  
PrintWriter의 입력이 된다.

한줄 단위로 입출력할 수 있다.

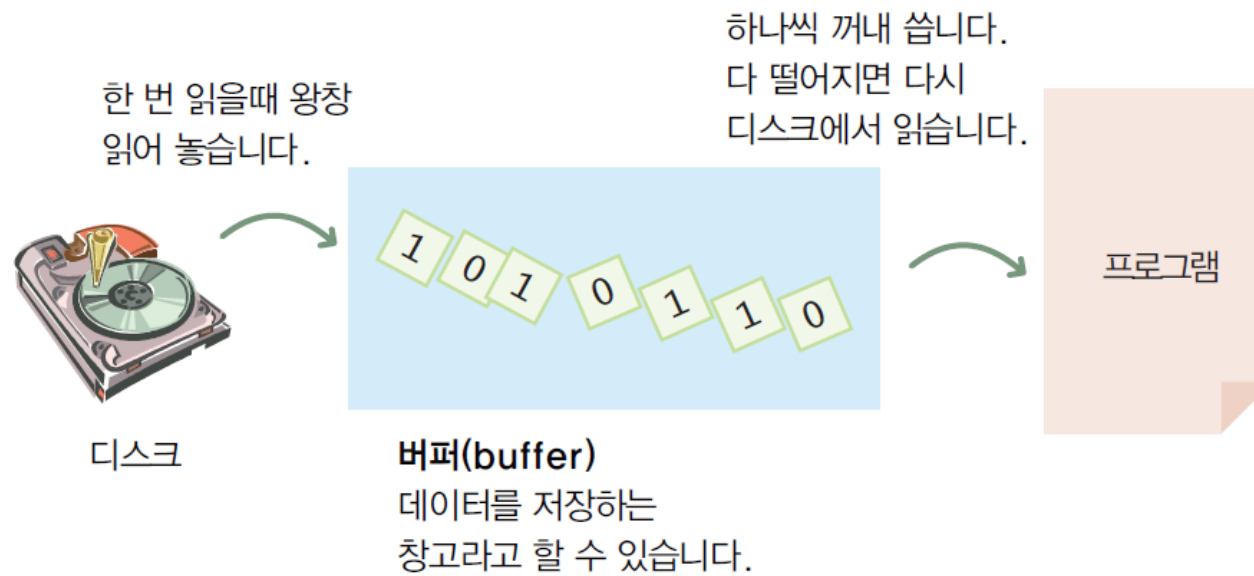


## 줄 단위의 입출력

```
18         if (inputStream != null) {
19             inputStream.close();
20         }
21         if (outputStream != null) {
22             outputStream.close();
23         }
24     }
25 }
26 }
```



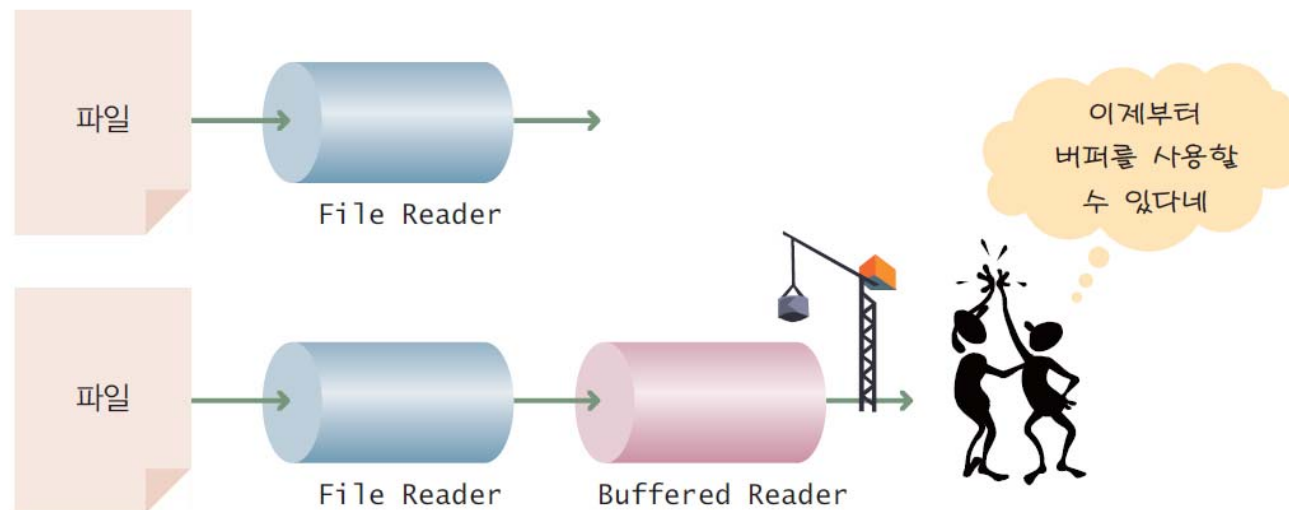
# 버퍼 스트림



< 버퍼 스트림의 개념 >

# 버퍼 스트림의 연결

```
InputStream = new BufferedReader(new FileReader("input.txt"));  
OutputStream = new BufferedWriter(new FileWriter("output.txt"));
```



# Scanner 클래스로 입력을 토큰으로 분리하기

ScanTest.java

```
01 ...
02 public class ScanTest {
03     public static void main(String[] args) throws IOException {
04
05         Scanner s = null;
06
07         try {
08             s = new Scanner(new BufferedReader(new FileReader("input.txt")));
09
10             while (s.hasNext()) {
11                 System.out.println(s.next());
12             }
13         } finally {
14             if (s != null) {
15                 s.close();
16             }
17         }
18     }
19 }
```

문자열을 토큰으로 분리한다.



# 예제

---

## 실행결과

```
The  
language  
of  
...
```

# Scanner 클래스로 기초 자료형 읽기

ScanTest2.java

```
01 import java.io.*;
02 import java.util.*;
03
04 public class ScanTest2 {
05     public static void main(String[] args) throws IOException {
06         Scanner s = null;
07         double sum = 0;
08         PrintWriter out = null;
09
10         out = new PrintWriter(new FileWriter("output.txt"));
11         out.println("9.5");
12         out.println("567,000");
13         out.flush();
14         s = new Scanner(new BufferedReader(new FileReader("output.txt")));
15         while (s.hasNext()) {
16             if (s.hasNextDouble()) {
17                 sum += s.nextDouble();
18             } else {
19                 s.next();
20             }
21         }
22         if (out != null)
23             out.close();
24         if (s != null)
25             s.close();
26         System.out.println(sum);
27     }
28 }
```

← 토큰을 기초 자료형으로  
변환하여 읽는다.



## 예제

실행결과

567009.5

# PrintStream과 PrintWriter 클래스



< PrintStream의 개념 >



# print()와 println()메소드

Log.java

```
01 public class Log {  
02     public static void main(String[] args) {  
03         int i = 8;  
04         double r = Math.log(i);  
05  
06         System.out.println("log(" + i + ")은 " + r + "입니다.");  
07     }  
08 }
```

실행결과

log(8)은 2.0794415416798357입니다.





# format() 메소드

Root.java

```
01 public class Root {  
02     public static void main(String[] args) {  
03         int i = 8;  
04         double r = Math.log(i);  
05  
06         System.out.format("log(" + "%d" + ")은 " + "%f" + "입니다.%n", i, r);  
07     }  
08 }
```

실행결과

log(8)은 2.079442입니다.

# format() 메소드

Log2.java

```
01 public class Log2 {  
02     public static void main(String[] args) {  
03         int i = 8;  
04         double r = Math.log(i);  
05  
06         System.out.format("log(" + %d + ")은 " + %+020.10f + "입니다.%n", i, r);  
07     }  
08 }
```

실행결과

log(8)은 2.079442입니다.



< 형식 지정자 >



# 예제

## CopyFile2.java

```
01 import java.io.*;
02
03 public class CopyFile2 {
04     public static void main(String[] args) throws IOException {
05         BufferedReader in = null;
06         PrintWriter out = null;
07
08         out = new PrintWriter(new FileWriter("output.txt"));
09         out.println("변화를 원한다면,");
10         out.println("제일 먼저 자신이 변화할 수 있다는 것과");
11         out.println("변화하기까지 포기하지 않고");
12         out.println("계속해서 노력할 수 있다는 것을 믿어야 한다.");
13         out.flush();
14         in = new BufferedReader(new FileReader("output.txt"));
15         String line;
16         while ((line = in.readLine()) != null) {
17             System.out.println(line);
18         }
19         if (in != null)
20             in.close();
21         if (out != null)
```



# 예제

```
22         out.close();  
23     }  
24 }
```

## 실행결과

변화를 원한다면,  
제일 먼저 자신이 변화할 수 있다는 것과  
변화하기까지 포기하지 않고  
계속해서 노력할 수 있다는 것을 믿어야 한다.

# 중간 점검 문제



## 중간점검

1. 텍스트를 파일로 쓰는 `print()`와 `println()`을 제공하는 클래스는?
2. `format()` 메소드를 이용하여서 정수를 10칸의 필드에 부호를 붙여서 출력하려면 어떤 형식 지정자를 사용하여야 하는가?

# 명령어 행에서 입출력

- 표준 스트림은 많은 운영체제의 특징이다.
- 기본적으로 키보드에서 읽으며 모니터로 출력한다.



```
InputStreamReader cin = new InputStreamReader(System.in);
```



# DataInputStream과 DataOutputStream

- DataInputStream과 DataOutputStream 클래스는 기초 자료형 단위로 데이터를 읽고 쓸 수 있다.

생성자 또는 메소드	설명
<code>DataStream(InputStream in)</code>	주어진 <code>InputStream</code> 과 연결된 객체를 생성한다.
<code>boolean readBoolean(boolean b)</code>	스트림으로 <code>boolean</code> 타입을 읽어서 반환한다.
<code>byte readByte()</code>	해당되는 자료형을 읽어서 반환한다.
<code>char readChar()</code>	
<code>double readDouble()</code>	
<code>float readFloat()</code>	
<code>int readInt()</code>	
<code>long readLong()</code>	
<code>short readShort()</code>	
<code>int readUnsignedByte()</code>	
<code>int readUnsignedShort()</code>	



# DataInputStream과 DataOutputStream

<code>String readUTF()</code>	UTF-8 형식으로 코딩된 문자열을 읽는다.
<code>void readFully(byte[] b, int off, int len)</code>	입력 스트림에서 len 바이트를 읽어서 b[]의 off 위치에 저장한다.
<code>void readFully(byte[] b)</code>	입력 스트림에서 바이트를 읽어서 b[]에 저장한다.
<code>int skipBytes(int n)</code>	입력 스트림에서 n 바이트를 건너뛴다.





# DataInputStream과 DataOutputStream

생성자 또는 메소드	설명
<code>DataOutputStream(OutputStream out)</code>	주어진 OutputStream과 연결된 객체를 생성한다.
<code>void writeBoolean(boolean b)</code>	스트림으로 boolean 타입을 출력한다.
<code>void writeByte(int v)</code>	해당되는 자료형을 출력한다.
<code>void writeChar(int v)</code>	
<code>void writeDouble(double v)</code>	
<code>void writeFloat(float v)</code>	
<code>void writeInt(int v)</code>	
<code>void writeLong(long v)</code>	
<code>void writeShort(int v)</code>	UTF-8 형식으로 코딩된 문자열을 출력한다.
<code>void writeUTF(String str)</code>	

# 예제

DataStreamTest.java

```
01 import java.io.*;
02
03 public class DataStreamTest {
04     public static void main(String[] args) throws IOException {
05         DataInputStream in = null;
06         DataOutputStream out = null;
07         try {
08             int c;
09
10             out = new DataOutputStream(new BufferedOutputStream(
11                 new FileOutputStream("data.bin")));
12             out.writeDouble(3.14);
13             out.writeInt(100);
14             out.writeUTF("자신의 생각을 바꾸지 못하는 사람은 결코 현실을 바꿀 수 없다.");
15
16             out.flush();
17             in = new DataInputStream(new BufferedInputStream(
18                 new FileInputStream("data.bin")));
19
20             System.out.println(in.readDouble());
21             System.out.println(in.readInt());
22             System.out.println(in.readUTF());
23         }
```

파일 스트림 → 버퍼 스트림 → 데이터 스트림



## 예제

```
24         } finally {  
25             if (in != null) {  
26                 in.close();  
27             }  
28             if (out != null) {  
29                 out.close();  
30             }  
31         }  
32     }  
33 }
```

### 실행결과

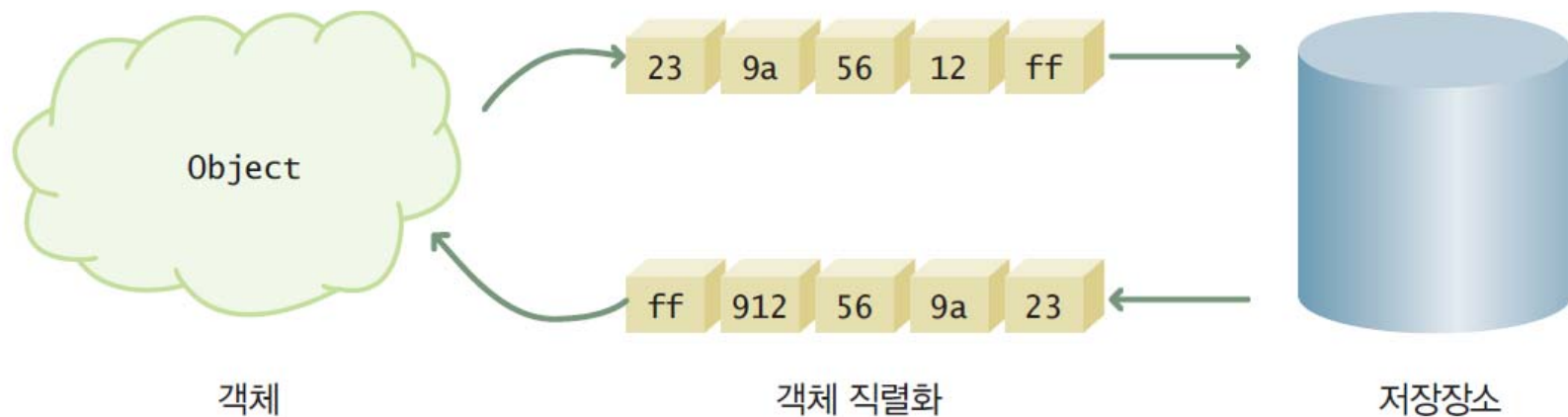
3.14

100

자신의 생각을 바꾸지 못하는 사람은 결코 현실을 바꿀 수 없다.

# ObjectInputStream과 ObjectOutputStream

- 직렬화(serialization):
  - 객체가 가진 데이터들을 순차적인 데이터로 변환하는 것



< 객체 직렬화의 개념 >

# 예제

## ObjectStreamTest.java

```
01 import java.io.*;
02 import java.util.Date;
03
04 public class ObjectStreamTest {
05     public static void main(String[] args) throws IOException {
06         ObjectInputStream in = null;
07         ObjectOutputStream out = null;
08         try {
09             int c;
10
11             out = new ObjectOutputStream(new FileOutputStream("object.dat"));
12             out.writeObject(new Date());
13
14             out.flush();
15             in = new ObjectInputStream(new FileInputStream("object.dat"));
16             Date d = (Date) in.readObject();
17             System.out.println(d);
18
19         } catch (ClassNotFoundException e) {
20         } finally {
21             if (in != null) {
22                 in.close();
23             }
24         }
25     }
26 }
```

객체 기록

객체 출력 스트림 생성

객체를 읽는다.

객체 입력 스트림 생성



## 예제

```
23         }  
24         if (out != null) {  
25             out.close();  
26         }  
27     }  
28 }  
29 }
```

### 실행결과

Fri May 01 15:46:56 KST 2009

# 중간 점검 문제



## 중간점검

1. 파일 data.bin에서 바이트 형태로 버퍼를 사용하여 데이터를 읽는 스트림을 생성하여 보라.
2. 객체를 네트워크를 통하여 보냈다가 다시 받으려면 어떤 클래스들을 이용하여야 하는가?
3. double형의 데이터를 저장하였다가 다시 읽으려면 어떤 스트림 클래스가 적합한가?



# File 객체

- File 클래스는 파일을 나타낸다.

```
File file = new File("data.txt");
```

- 파일에 대한 여러 가지 메소드를 제공

반환형	메소드	설명
boolean	canExecute()	파일을 실행할 수 있는지의 여부
boolean	canRead()	파일을 읽을 수 있는지의 여부
boolean	canWrite()	파일을 변경할 수 있는지의 여부
static File	createTempFile(String prefix, String suffix)	임시 파일을 생성한다.





# File 객체

boolean	delete()	파일을 삭제한다.
void	deleteOnExit()	가상 기계가 종료되면 파일을 삭제한다.
boolean	exists()	파일의 존재 여부
String	getAbsolutePath()	절대 경로를 반환
String	getCanonicalPath()	정규 경로를 반환
String	getName()	파일의 이름을 반환
String	getParent()	부모 경로 이름을 반환
File	getParentFile()	부모 파일을 반환
boolean	isDirectory()	디렉토리이면 참
boolean	isFile()	파일이면 참
long	lastModified()	파일이 변경되었는지 여부
long	length()	파일 길이 반환
String[]	list()	디렉토리 안에 포함된 파일과 디렉토리를 반환
boolean	mkdir()	디렉토리를 생성한다.
boolean	renameTo(File dest)	파일 이름을 변경한다.



# File 객체

boolean	<code>renameTo(File dest)</code>	파일 이름을 변경한다.
boolean	<code>setExecutable(boolean executable)</code>	파일을 실행 가능하게 설정
boolean	<code>setLastModified(long time)</code>	파일을 변경된 것으로 설정

- 다음 문장은 파일을 현재 시간으로 변경한다.

```
new File("data.txt").setLastModified(new Date().getTime());
```

# 예제

FileTest.java

```
01  import java.io.File;
02  import java.io.IOException;
03
04  public class FileTest {
05      public static void main(String[] args) throws IOException {
06          String name = "c:/eclipse";
07          File dir = new File(name);
08          String[] fileNames = dir.list();          // 현재 디렉토리의 전체 파일 리스트
09          for (String s : fileNames) {
10              File f = new File(name + "/" + s);    // 절대 경로로 이름을 주어야 함
11              System.out.println("=====");
12              System.out.println("이름: " + f.getName());
13              System.out.println("경로: " + f.getPath());
14              System.out.println("부모: " + f.getParent());
15              System.out.println("절대경로: " + f.getAbsolutePath());
16              System.out.println("정규경로: " + f.getCanonicalPath());
17              System.out.println("디렉토리 여부:" + f.isDirectory());
18              System.out.println("파일 여부:" + f.isFile());
19              System.out.println("=====");
20          }
21      }
22  }
```



# 예제

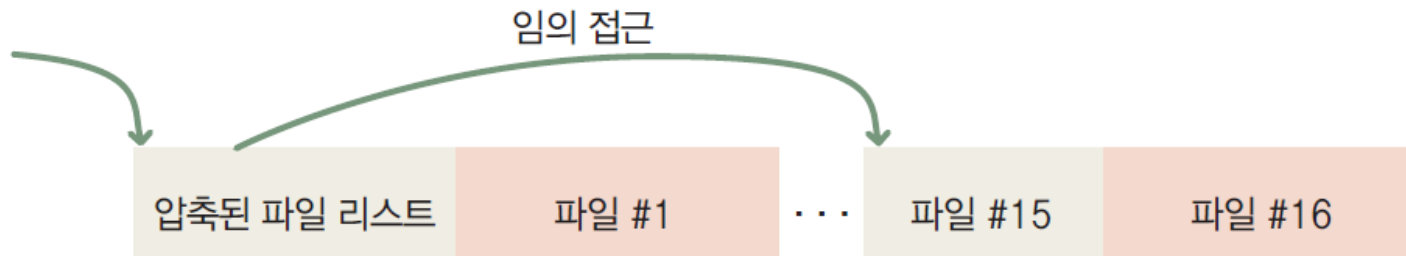
## 실행결과

```
=====
이름: .eclipseproduct
경로: c:\eclipse\.eclipseproduct
부모: c:\eclipse
절대경로: c:\eclipse\.eclipseproduct
정규경로: C:\eclipse\.eclipseproduct
디렉토리 여부: false
파일 여부: true
=====
```

# 임의 접근 파일

- 임의 접근 파일은 파일에 비순차적인 접근을 가능하게 한다.

```
new RandomAccessFile("all.zip", "r");
```



메소드	설명
int skipBytes(int)	지정된 바이트만큼 파일 포인터를 앞쪽으로 이동한다.
void seek(long)	지정된 바이트 위치로 파일 포인터를 설정한다.
long getFilePointer()	파일 포인터의 현재 위치를 반환한다.

# 중간 점검 문제

## 중간점검



1. 텍스트 파일과 이진 파일의 차이점은 무엇인가?
2. File 객체를 생성하여서 파일에서 텍스트 데이터를 읽으려면 어떤 절차를 밟아야 하는가?

# Q & A

