



입출력 스트림과 파일 입출력

스트림

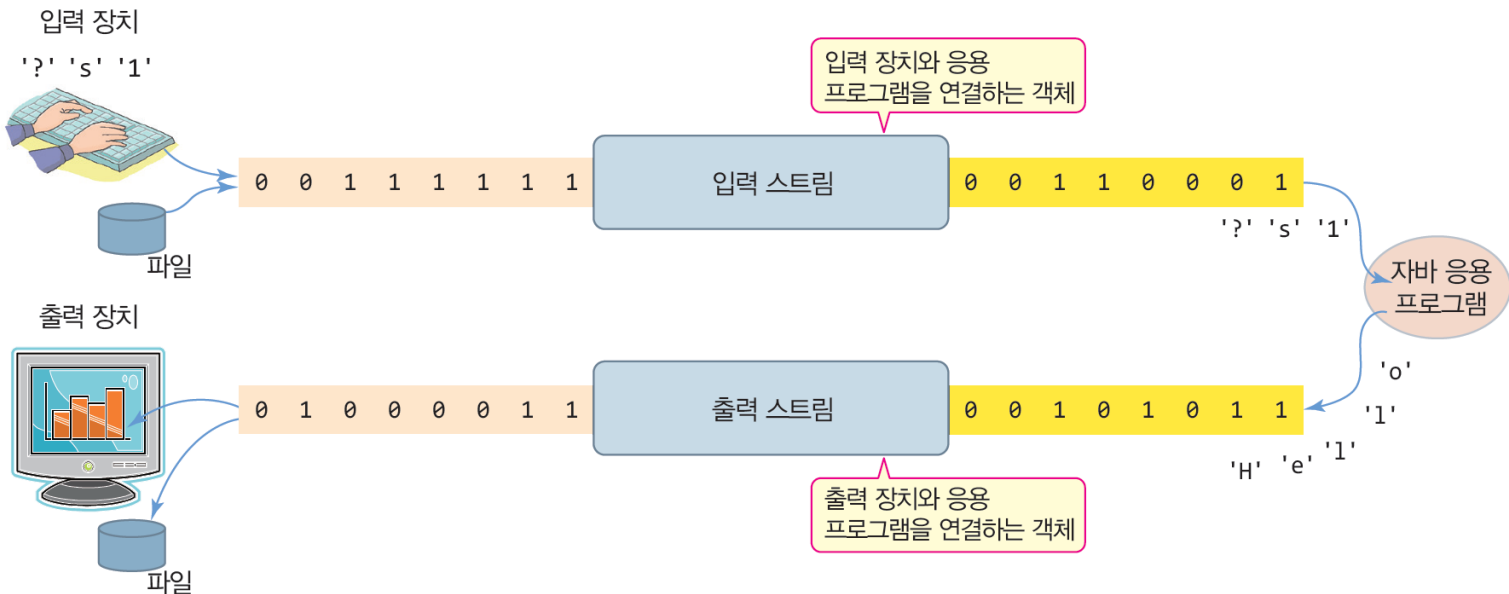
2

□ 스트림 입출력

- ▣ 버퍼를 가지고 순차적으로 이루어지는 입출력

□ 자바의 입출력 스트림

- ▣ 응용프로그램과 입출력 장치를 연결하는 소프트웨어 모듈
 - 입력 스트림 : 입력 장치로부터 자바 프로그램으로 데이터를 전달
 - 출력 스트림 : 출력 장치로 데이터 출력



자바의 입출력 스트림 특징

3

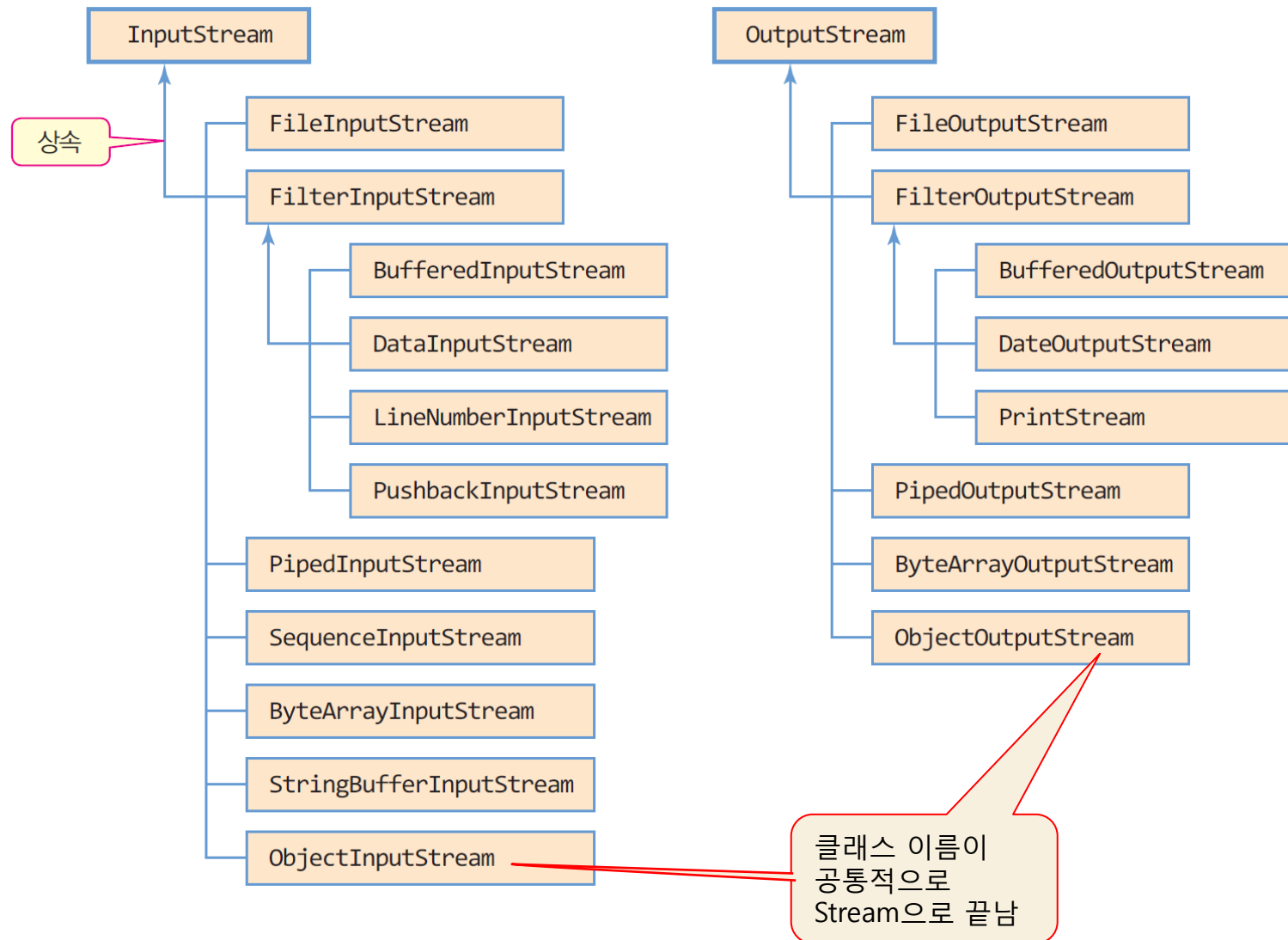
- 스트림의 양끝에 입출력장치와 자바 응용프로그램 연결
- 스트림은 단방향
 - ▣ 입력과 출력을 동시에 하는 스트림 없음
- 입출력 스트림 기본 단위
 - ▣ 바이트 스트림의 경우 : 바이트
 - ▣ 문자 스트림의 경우 : 문자(자바에서는 문자1개 : 2 바이트)
- 선입선출 구조

자바의 입출력 스트림 종류

4

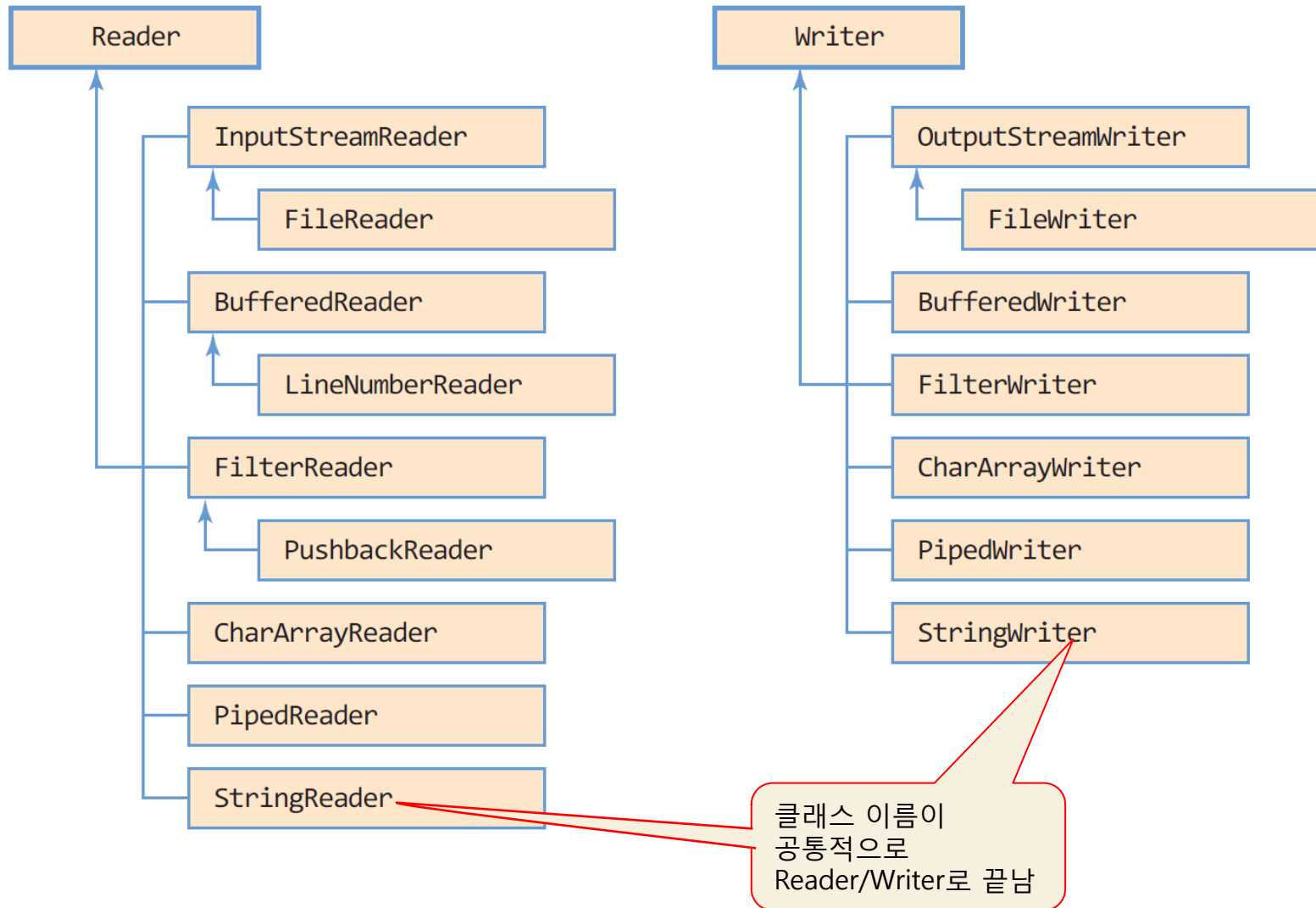
- 바이트 스트림과 문자 스트림
 - ▣ 바이트 스트림
 - 입출력되는 데이터를 단순 바이트로 처리
 - 예) 바이너리 파일을 읽는 입력 스트림
 - ▣ 문자 스트림
 - 문자만 입출력하는 스트림
 - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
 - 예) 텍스트 파일을 읽는 입력 스트림
- JDK는 입출력 스트림을 구현한 다양한 클래스 제공
 - ▣ 다음 슬라이드

5

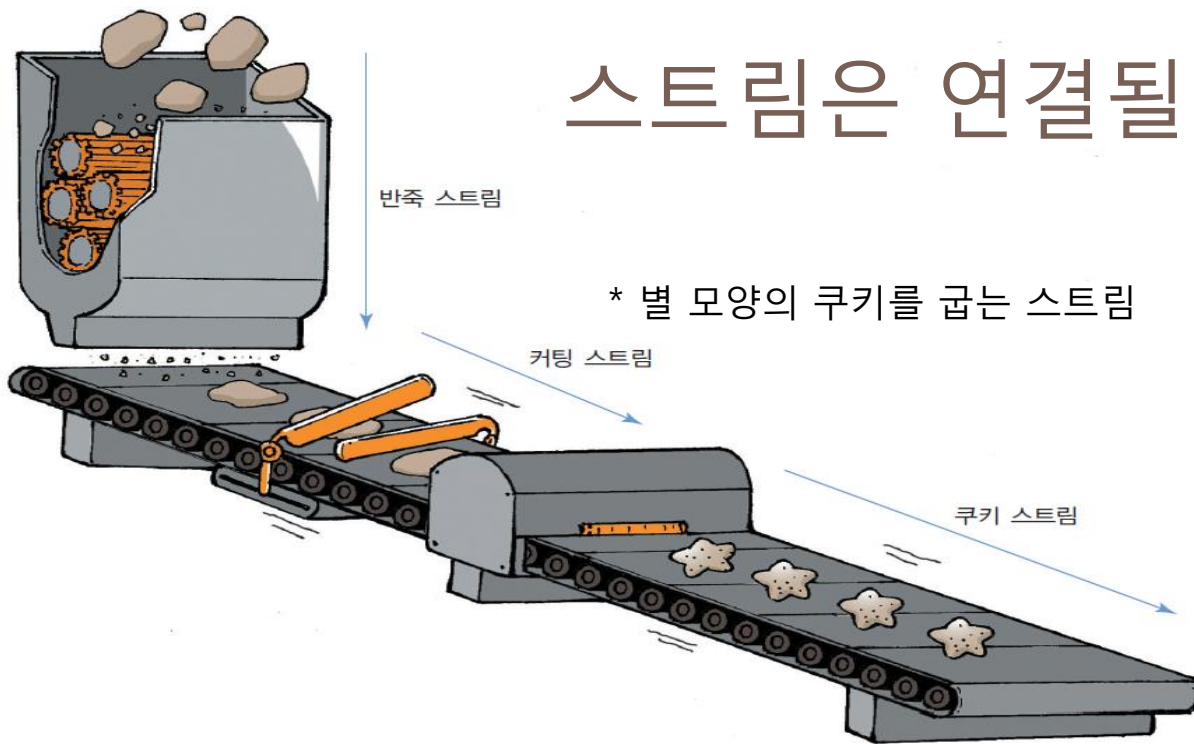


JDK의 문자 스트림 클래스 계층 구조

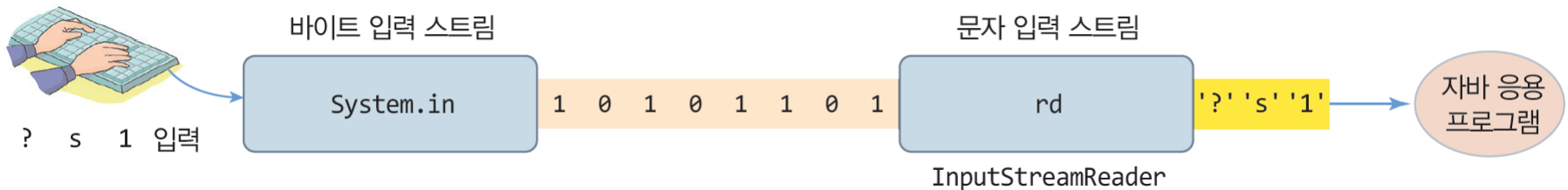
6



스트림은 연결될 수 있다



* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결한 사례



```
InputStreamReader rd = new InputStreamReader(System.in);  
int c = rd.read(); // 키보드에서 문자 읽음
```

문자 스트림

8

- 문자 스트림
 - ▣ 유니 코드(2바이트) 문자를 입출력 하는 스트림
 - 문자로 표현되지 않는 데이터는 다루지 못함
 - 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음
- 문자 스트림을 다루는 클래스
 - ▣ Reader/Writer
 - ▣ InputStreamReader/OutputStreamWriter
 - ▣ FileReader/FileWriter
 - 텍스트 파일에서 문자 데이터 입출력

FileReader을 이용한 파일 읽기

9

□ 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
FileReader fin = new FileReader ("c:\Wtest.txt");
```

C:\Wtest.txt 파일을 열고 파일과 입력 바이트 스트림 객체 fin 연결

```
int c;
```

```
while((c = fin.read()) != -1) {
```

파일 끝까지 바이트씩 c에 읽어 들임.
파일의 끝을 만나면 read()는 -1 리턴

```
    System.out.print((char)c);
```

바이트 c를 문자로 변환하여 화면에 출력

```
}
```

```
fin.close();
```

스트림을 닫음. 파일도 닫힘.
스트림과 파일의 연결을 끊음.
더 이상 스트림으로부터 읽을 수 없음

예제 8-1 : FileReader로 텍스트 파일 읽기

10

FileReader를 이용하여 c:\windows\system.ini 파일을 읽어 화면에 출력하는 프로그램을 작성하라. system.ini는 텍스트 파일이다.

```
import java.io.*;

public class FileReaderEx {
    public static void main(String[] args) {
        FileReader fin = null;
        try {
            fin = new FileReader("c:\\windows\\system.ini");
            int c;
            while ((c = fin.read()) != -1) { // 한 문자씩 파일 끝까지 읽기
                System.out.print((char)c);
            }
            fin.close();
        }
        catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

파일의 끝을 만나면 read()는 -1
리턴

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.driv

[mci]
```

문자 집합과 InputStreamReader를 이용한 텍스트 파일 읽기

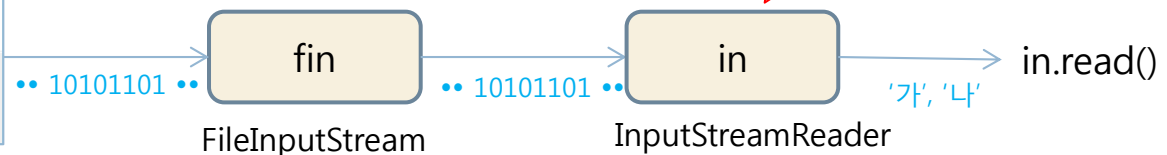
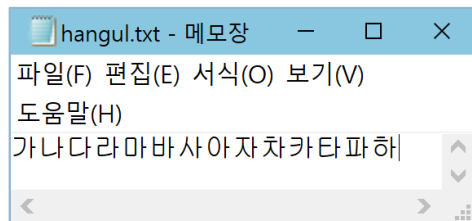
11

```
FileInputStream fin = new FileInputStream("c:\\Temp\\hangul.txt");  
InputStreamReader in = new InputStreamReader(fin, "MS949");
```

```
while ((c = in.read()) != -1) {  
    System.out.print((char)c);  
}
```

한글 완성형 확장형 문자 집합

문자 집합 사용
(윈도우에서 MS949)



예제 8-2 : InputStreamReader로 한글 텍스트 파일 읽기

12

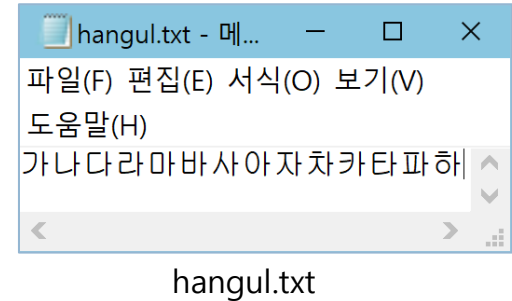
InputStreamReader를 이용하여 MS949 문자 집합으로 한글 텍스트 파일을 읽고 출력하라.

```
import java.io.*;

public class FileReadHangulSuccess {
    public static void main(String[] args) {
        InputStreamReader in = null;
        FileInputStream fin = null;
        try {
            fin = new FileInputStream("c:\\Temp\\hangul.txt");
            in = new InputStreamReader(fin, "MS949");
            int c;

            System.out.println("인코딩 문자 집합은 " + in.getEncoding());
            while ((c = in.read()) != -1) {
                System.out.print((char)c);
            }
            in.close();
            fin.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

MS에서 만든 한글 확장
완성형 문자 집합



인코딩 문자 집합은 MS949
가나다라마바사아자차카타파하

예제 8-3 : 한글 텍스트 파일 읽기(문자 집합 지정이 잘못된 경우)

13

InputStreamReader의 문자 집합을 US-ASCII로 지정하여 한글 파일을 읽고 출력하라.

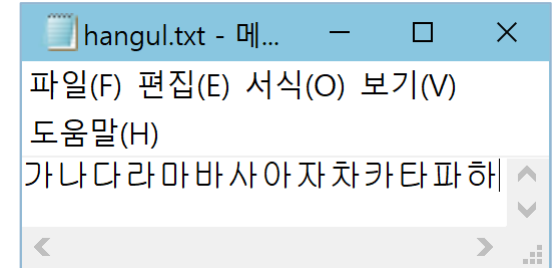
```
import java.io.*;

public class FileReadHangulFail {
    public static void main(String[] args) {
        InputStreamReader in = null;
        FileInputStream fin = null;
        try {
            fin = new FileInputStream("c:\\Temp\\hangul.txt");
            in = new InputStreamReader(fin, "US-ASCII");
            int c;

            System.out.println("인코딩 문자 집합은 " + in.getEncoding());
            while ((c = in.read()) != -1) {
                System.out.print((char)c);
            }
            in.close();
            fin.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

문자 집합 지정이 잘못된 경우의 예를 보이기
위해 일부러 틀린 문자 집합 지정

문자 집합 지정이 잘못되어
읽은 문자가 제대로 인식되지 못함.
출력 결과가 깨짐



hangul.txt

인코딩 문자 집합은 ASCII
????????????????????????????????

FileWriter 사용 예

14

- c:\Temp\test.txt로의 문자 출력 스트림 생성

```
FileWriter fout = new FileWriter("c:\\Temp\\test.txt");
```

- 파일 쓰기

- 문자 단위 쓰기

```
FileWriter fout = new FileWriter("c:\\Temp\\test.txt");  
fout.write('A'); // 문자 'A' 출력  
fout.close();
```

- 블록 단위 쓰기

```
char [] buf = new char [1024];  
  
// buf[] 배열의 처음부터 배열 크기(1024개 문자)만큼 쓰기  
fout.write(buf, 0, buf.length);
```


예제 8-4 : 키보드 입력을 파일로 저장하기

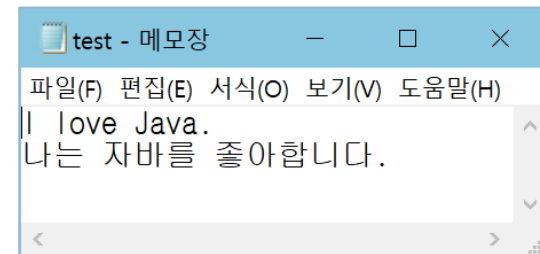
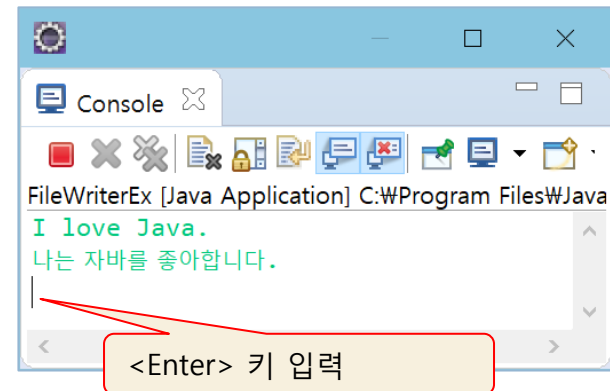
15

Scanner를 이용하여 키보드에서 입력받은 데이터를 c:\Temp\test.txt 파일에 저장하는 프로그램을 작성하라.

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

public class FileWriterEx {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        FileWriter fout = null;
        int c;
        try {
            fout = new FileWriter("c:\Temp\test.txt");
            while(true) {
                String line = scanner.nextLine();
                if(line.length() == 0)
                    break;
                fout.write(line, 0, line.length());
                fout.write("\r\n", 0, 2);
            }
            fout.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
        scanner.close();
    }
}
```

한 줄 띄기 위해
WrWn을 파일에 저장



실행 결과 test.txt 파일 생성

바이트 스트림 클래스

16

- 바이트 스트림
 - ▣ 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
- 바이트 스트림 클래스
 - ▣ InputStream/OutputStream
 - 추상 클래스
 - 바이트 스트림을 다루는 모든 클래스의 슈퍼 클래스
 - ▣ FileInputStream/FileOutputStream
 - 파일로부터 바이트 단위로 읽거나 저장하는 클래스
 - 바이너리 파일의 입출력 용도
 - ▣ DataInputStream/DataOutputStream
 - 자바의 기본 데이터 타입의 값(변수)을 바이너리 값 그대로 입출력
 - 문자열도 바이너리 형태로 입출력

FileOutputStream을 이용한 파일 쓰기

17

□ 바이너리 값을 파일에 저장하는 바이트 스트림 코드

```
FileOutputStream fout = new FileOutputStream("c:\\Temp\\test.out");
```

```
byte b[] = {7,51,3,4,-1,24};  
for(int i=0; i<b.length; i++)  
    fout.write(b[i]);
```

파일에 배열 b[i]의 정수 값(바이너리)을 그대로 기록

```
fout.close();
```

스트림을 닫음. 파일도 닫힘. 더 이상 스트림으로부터 읽을 수 없음

7 51 3 4 -1 24

00000000h:07 33 03 04 FF 18

; 2 3 4 . ↑

파일에 있는 각 바이너리 값들은
문자 정보가 아님. 바이너리 값에
대응하는 그래픽 심볼들

test.out 파일의 내부

예제 8-5 : FileOutputStream으로 바이너리 파일 쓰기

18

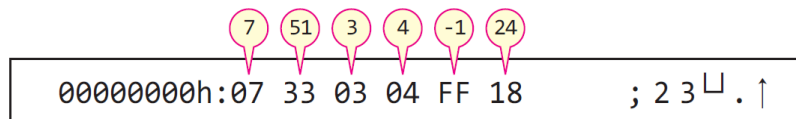
FileOutputStream을 이용하여 byte [] 배열 속에 들어 있는 바이너리 값을 c:\Temp\test.out 파일에 저장하라. 이 파일은 바이너리 파일이 되므로 메모장으로 볼 수 없다. 이 파일은 예제 8-6에서 FileInputStream을 이용하여 읽어 다시 출력한다.

```
import java.io.*;

public class FileOutputStreamEx {
    public static void main(String[] args) {
        byte b[] = {7,51,3,4,-1,24};
        try {
            FileOutputStream fout =
                new FileOutputStream("c:\\Temp\\test.out");
            for(int i=0; i<b.length; i++)
                fout.write(b[i]); // 배열 b의 바이너리를 그대로 기록
            fout.close();
        } catch(IOException e) {
            System.out.println("c:\\Temp\\test.out에 저장할 수
                                없었습니다. 경로명을 확인해 주세요.");
            return;
        }
        System.out.println("c:\\Temp\\test.out을 저장하였습니다.");
    }
}
```

c:\Temp\test.out을 저장하였습니다.

test.out 파일의 내부



예제 8-6 : FileInputStream으로 바이너리 파일 읽기

19

FileInputStream을 이용하여 c:\Temp\test.out 파일(예제 8-5에서 저장한 파일)을 읽어 byte [] 배열 속에 저장하고 화면에 출력하라.

```
import java.io.*;

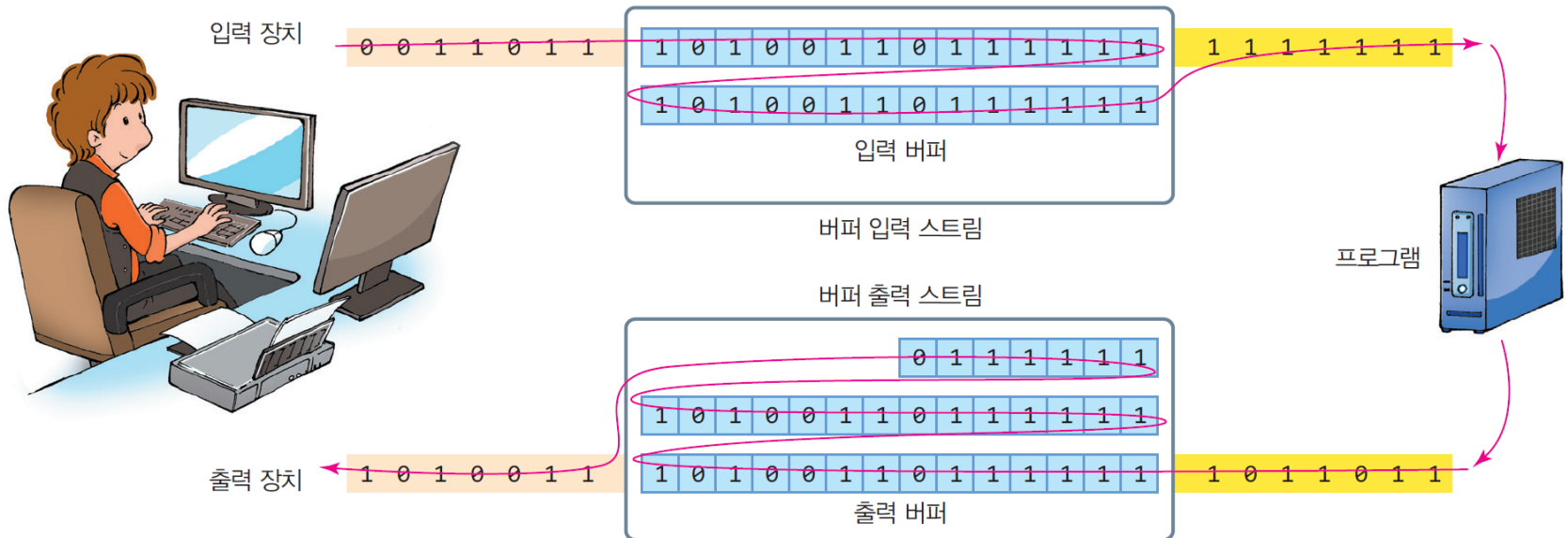
public class FileInputStreamEx {
    public static void main(String[] args) {
        byte b[] = new byte [6]; // 비어 있는 byte 배열
        try {
            FileInputStream fin = new FileInputStream("c:\Temp\test.out");
            int n=0, c;
            while((c = fin.read())!= -1) {
                b[n] = (byte)c;
                n++;
            }
            System.out.println("c:\Temp\test.out에서 읽은 배열을 출력합니다.");
            for(int i=0; i<b.length; i++) System.out.print(b[i] + " ");
            System.out.println();
            fin.close();
        } catch(IOException e) {
            System.out.println( "c:\Temp\test.out에서 읽지 못했습니다. 경로명을 체크해보세요");
        }
    }
}
```

c:\Temp\test.out에서 읽은 배열을 출력합니다.
7 51 3 4 -1 24

버퍼 입출력 스트림과 버퍼 입출력의 특징

20

- 버퍼 스트림
 - ▣ 버퍼를 가진 스트림
 - ▣ 입출력 데이터를 일시적으로 저장하는 버퍼를 이용하여 입출력 효율 개선
- 버퍼 입출력의 목적
 - ▣ 입출력 시 운영체제의 API 호출 횟수를 줄여 입출력 성능 개선
 - 출력시 여러 번 출력되는 데이터를 버퍼에 모아두고 한 번에 장치로 출력



버퍼 스트림의 종류

21

- 바이트 버퍼 스트림
 - ▣ 바이트 단위의 바이너리 데이터를 처리하는 버퍼 스트림
 - ▣ BufferedInputStream와 BufferedOutputStream
- 문자 버퍼 스트림
 - ▣ 유니코드의 문자 데이터만 처리하는 버퍼 스트림
 - ▣ BufferedReader와 BufferedWriter

20바이트 버퍼를 가진 BufferedOutputStream

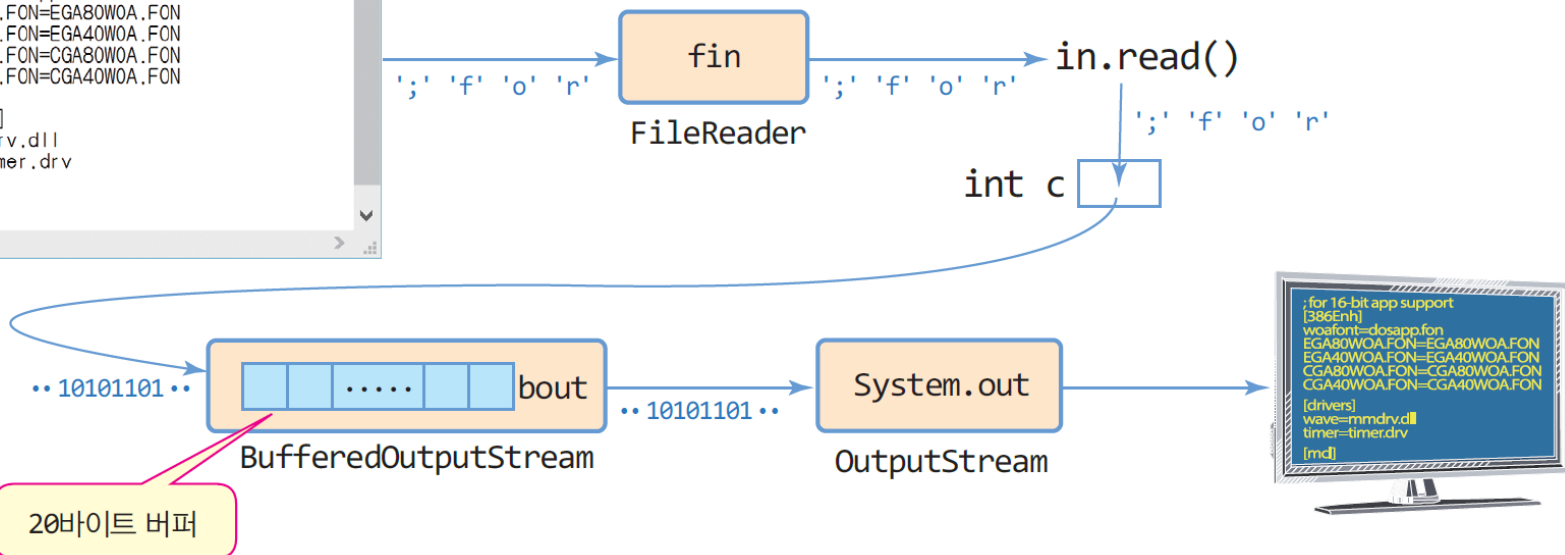
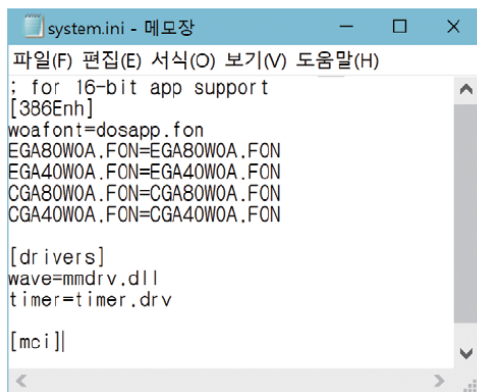
```
BufferedOutputStream bout = new BufferedOutputStream(System.out, 20);  
FileReader fin = new FileReader("c:\\windows\\system.ini");
```

```
int c;  
while ((c = fin.read()) != -1) {  
    bout.write((char)c);  
}  
fin.close();  
bout.close();
```

파일 전체를 읽어 화면에 출력

스트림 닫음

20바이트 크기의 버퍼 설정.
System.out 표준 스트림에 출력



예제 8-7 : 버퍼 스트림을 이용한 출력

23

버퍼 크기를 5로 하고, 표준 출력 스트림(System.out)과 연결한 버퍼 출력 스트림을 생성하라. c:\Temp\test2.txt 파일을 저장된 영문 텍스트를 읽어 버퍼 출력 스트림을 통해 출력하라.

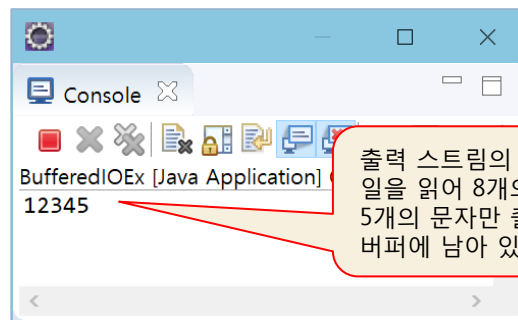
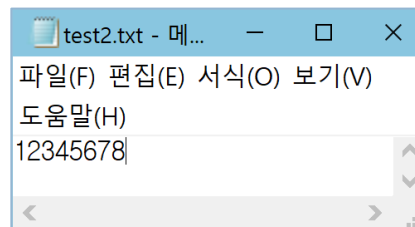
```
import java.io.*;
import java.util.Scanner;

public class BufferedIOEx {
    public static void main(String[] args) {
        FileReader fin = null;
        int c;
        try {
            fin = new FileReader("c:\\Temp\\test2.txt");
            BufferedOutputStream out = new
                BufferedOutputStream(System.out, 5);
            while ((c = fin.read()) != -1) {
                out.write(c);
            }

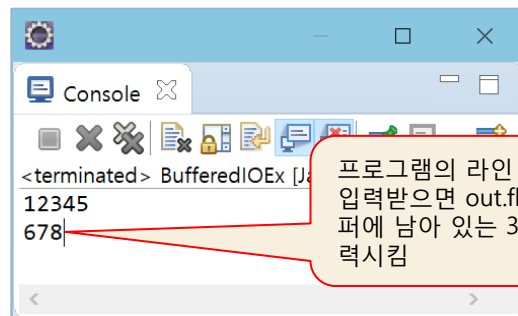
            // 파일 데이터가 모두 출력된 상태 <Enter> 키 기다림
            new Scanner(System.in).nextLine();
            out.flush(); // 버퍼에 남아 있던 문자 모두 출력
            fin.close();
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

버퍼가 꽉 찰 때 문자가 화면에 출력

<Enter> 키 기다림



출력 스트림의 버퍼 크기가 5이므로 파일을 읽어 8개의 문자를 출력하였지만 5개의 문자만 출력되고 3개의 문자는 버퍼에 남아 있어 화면에 보이지 않음



프로그램의 라인 15에서 <Enter>키를 입력받으면 out.flush()를 실행하여 버퍼에 남아 있는 3개의 문자를 강제로 출력시킴

File 클래스

24

□ File 클래스

- ▣ 파일의 경로명을 다루는 클래스
 - java.io.File
 - 파일과 디렉터리 경로명의 추상적 표현
- ▣ 파일 관리 기능
 - 파일 이름 변경, 삭제, 디렉터리 생성, 크기 등 파일 관리
 - File 객체는 파일 읽고 쓰기 기능 없음

File 클래스 사용 예

25

- 파일 객체 생성

```
File f = new File("c:\Windows\system.ini");
```

- 파일의 경로명

```
String filename = f.getName(); // "system.ini"
String path = f.getPath();      // "c:\Windows\system.ini"
String parent = f.getParent();  // "c:\Windows"
```

- 파일인지
디렉터리인지
구분

```
if(f.isFile()) // 파일인 경우
    System.out.println(f.getPath() + "는 파일입니다.");
else if(f.isDirectory()) // 디렉터리인 경우
    System.out.println(f.getPath() + "는 디렉터리입니다.");
```

- 서브 디렉터리
리스트 얻기

```
File f = new File("c:\Temp");
File[] subfiles = f.listFiles(); // c:\Temp 파일 및 서브디렉터리 리스트 얻기

for(int i=0; i<subfiles.length; i++) {
    System.out.print(subfiles[i].getName()); // 파일명 출력
    System.out.println("\t파일 크기: " + subfiles[i].length()); // 크기 출력
}
```

예제 8-8 : File 클래스 활용한 파일 관리

26

File 클래스를 이용하여 파일의 타입을 알아내고, 디렉터리에 있는 파일들을 나열하며, 디렉터리 이름을 변경하는 프로그램을 작성해보자.

```
import java.io.File;

public class FileEx {
    public static void listDirectory(File dir) {
        System.out.println("-----" + dir.getPath() +
            "의 서브 리스트 입니다.-----");
        File[] subFiles = dir.listFiles();
        for(int i=0; i<subFiles.length; i++) {
            File f = subFiles[i];
            long t = f.lastModified();
            System.out.print(f.getName());
            System.out.print("\t파일 크기: " + f.length());
            System.out.printf("\t수정한 시간: %tb %td %ta %tWn",t,
                t, t, t);
        }
    }
}
```

C:\Temp의 파일과
디렉터리 리스트

```
public static void main(String[] args) {
    File f1 = new File("c:\Temp\windows\system.ini");
    System.out.println(f1.getPath() + ", " + f1.getParent() + ", " +
        f1.getName());
    String res="";
    if(f1.isFile()) res = "파일";
    else if(f1.isDirectory()) res = "디렉터리";
    System.out.println(f1.getPath() + "은 " + res + "입니다.");
}
```

Java_sample을 javasample로
변경한 이후

```
File f2 = new File("c:\Temp\java_sample");
if(!f2.exists()) {
    f2.mkdir(); // 존재하지 않으면 디렉토리 생성
}
listDirectory(new File("c:\Temp"));
f2.renameTo(new File("c:\Temp\javasample"));
listDirectory(new File("c:\Temp"));
}
```

c:\windows\system.ini, c:\windows, system.ini

c:\windows\system.ini은 파일입니다.

-----c:\Temp의 서브 리스트 입니다.-----

Calc.class	파일	파일 크기: 754	수정한 시간: 3월 17 금 18:13:39
Calc.java	파일	파일 크기: 282	수정한 시간: 3월 17 금 18:13:25
hangul.txt	파일	파일 크기: 14	수정한 시간: 4월 03 월 20:58:51
java_sample	디렉터리	파일 크기: 0	수정한 시간: 4월 04 화 15:32:22
test.out	파일	파일 크기: 6	수정한 시간: 4월 04 화 11:32:10
test.txt	파일	파일 크기: 13	수정한 시간: 4월 03 월 21:17:51
test2.txt	파일	파일 크기: 8	수정한 시간: 4월 04 화 15:05:08

-----c:\Temp의 서브 리스트 입니다.-----

Calc.class	파일	파일 크기: 754	수정한 시간: 3월 17 금 18:13:39
Calc.java	파일	파일 크기: 282	수정한 시간: 3월 17 금 18:13:25
hangul.txt	파일	파일 크기: 14	수정한 시간: 4월 03 월 20:58:51
javasample	디렉터리	파일 크기: 0	수정한 시간: 4월 04 화 15:32:22
test.out	파일	파일 크기: 6	수정한 시간: 4월 04 화 11:32:10
test.txt	파일	파일 크기: 13	수정한 시간: 4월 03 월 21:17:51
test2.txt	파일	파일 크기: 8	수정한 시간: 4월 04 화 15:05:08

예제 8-9 : 텍스트 파일 복사

27

문자 스트림 `FileReader`와 `FileWriter`를 이용하여 `c:\windows\system.ini`를 `c:\Temp\system.txt` 파일로 복사하는 프로그램을 작성하라.

```
import java.io.*;

public class TextCopyEx {
    public static void main(String[] args){
        File src = new File("c:\\windows\\system.ini"); // 원본 파일 경로명
        File dest = new File("c:\\Temp\\system.txt"); // 복사 파일 경로명
        int c;
        try {
            FileReader fr = new FileReader(src);
            FileWriter fw = new FileWriter(dest);
            while((c = fr.read()) != -1) { // 문자 하나 읽고
                fw.write((char)c); // 문자 하나 쓰고
            }
            fr.close(); fw.close();
            System.out.println(src.getPath() + "를 " + dest.getPath() + "로 복사하였습니다.");
        } catch (IOException e) {
            System.out.println("파일 복사 오류");
        }
    }
}
```

`c:\windows\system.ini`를 `c:\Temp\system.txt`로 복사하였습니다.

예제 8-10 : 바이너리 파일 복사

28

바이트 스트림을 이용하여 바이너리 파일을 복사하는 프로그램을 작성하라

```
import java.io.*;

public class BinaryCopyEx {
    public static void main(String[] args) {
        File src = new File("c:\\Windows\\Web\\Wallpaper\\Theme1\\wimg1.jpg");
        File dest = new File("c:\\Temp\\copyimg.jpg");
        int c;
        try {
            FileInputStream fi = new FileInputStream(src);
            FileOutputStream fo = new FileOutputStream(dest);
            while((c = fi.read()) != -1) {
                fo.write((byte)c);
            }
            fi.close();
            fo.close();
            System.out.println(src.getPath() + "를 " +
                               dest.getPath() + "로 복사하였습니다.");
        } catch (IOException e) {
            System.out.println("파일 복사 오류");
        }
    }
}
```



c:\\Windows\\Web\\Wallpaper\\Theme1\\wimg1.jpg를 c:\\Temp\\copyimg.jpg로 복사하였습니다.

예제 8-11 : 블록 단위로 바이너리 파일 고속 복사

29

예제 8-10을 10KB 단위로 읽고 쓰도록 수정하여 고속으로 파일을 복사하라.

```
import java.io.*;

public class BlockBinaryCopyEx {
    public static void main(String[] args) {
        File src = new File("c:\\Windows\\Web\\Wallpaper\\Theme1\\wimg1.jpg");
        File dest = new File("c:\\Temp\\desert.jpg");
        try {
            FileInputStream fi = new FileInputStream(src);
            FileOutputStream fo = new FileOutputStream(dest);
            byte [] buf = new byte [1024*10]; // 10KB 버퍼
            while(true) {
                int n = fi.read(buf); // 버퍼 크기만큼 읽기. n은 실제 읽은 바이트
                fo.write(buf, 0, n); // buf[0]부터 n 바이트 쓰기
                if(n < buf.length)
                    break;
            }
            fi.close();
            fo.close();
            System.out.println( src.getPath() + "를 " + dest.getPath() +
                               "로 복사하였습니다.");
        } catch (IOException e) { System.out.println("파일 복사 오류"); }
    }
}
```

c:\\Windows\\Web\\Wallpaper\\Theme1\\wimg1.jpg를 c:\\Temp\\copyimg.jpg로 복사하였습니다.