파일과 I/O 스트림

컴퓨터공학전공
박요한

수업내용

- I/O 소개
- 스트림의 이해
  - ✓문자 스트림
  - ✓바이트 스트림
- 파일 입출력
- 보조 스트림

I/O의 범위

일반적인 입출력의 대상

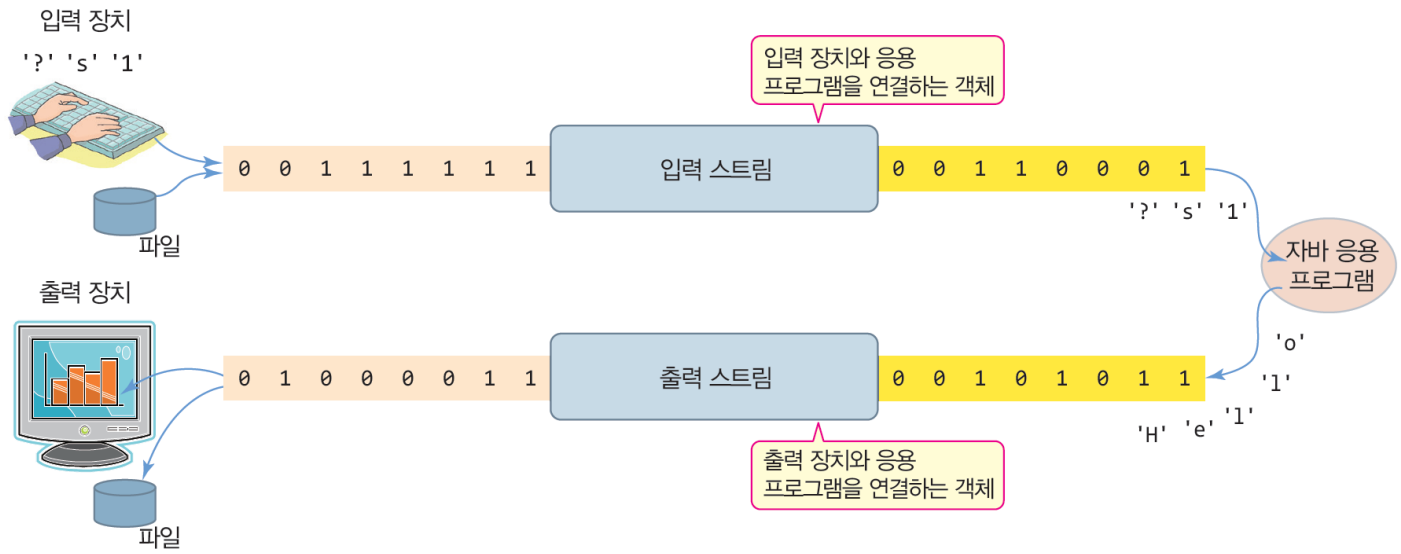
- 키보드와 모니터
- 하드디스크에 저장되어 있는 파일
- USB와 같인 외부 메모리 장치
- 네트워크로 연결되어 있는 컴퓨터
- 사운드카드, 오디오카드와 같은 멀티미디어 장치
- 프린터, 팩시밀리와 같은 출력장치

자바 스트림의 큰 분류

- 입력 스트림(Input Stream) 프로그램으로 데이터를 읽어 들이는 스트림
- 출력 스트림(Output Stream) 프로그램으로부터 데이터를 내보내는 스트림

스트림

- 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



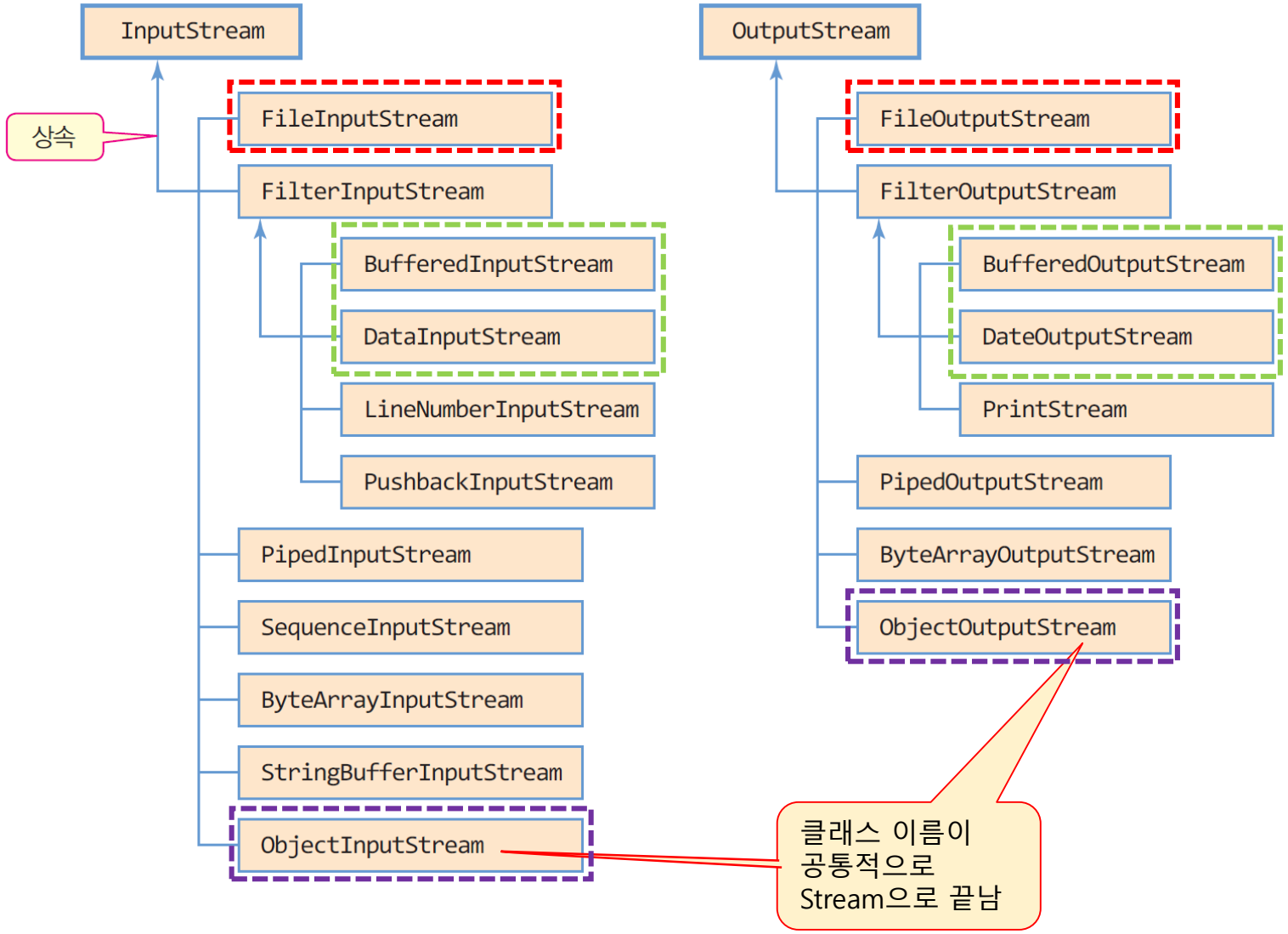
자바의 입출력 스트림 특징

- 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(FIFO) 구조

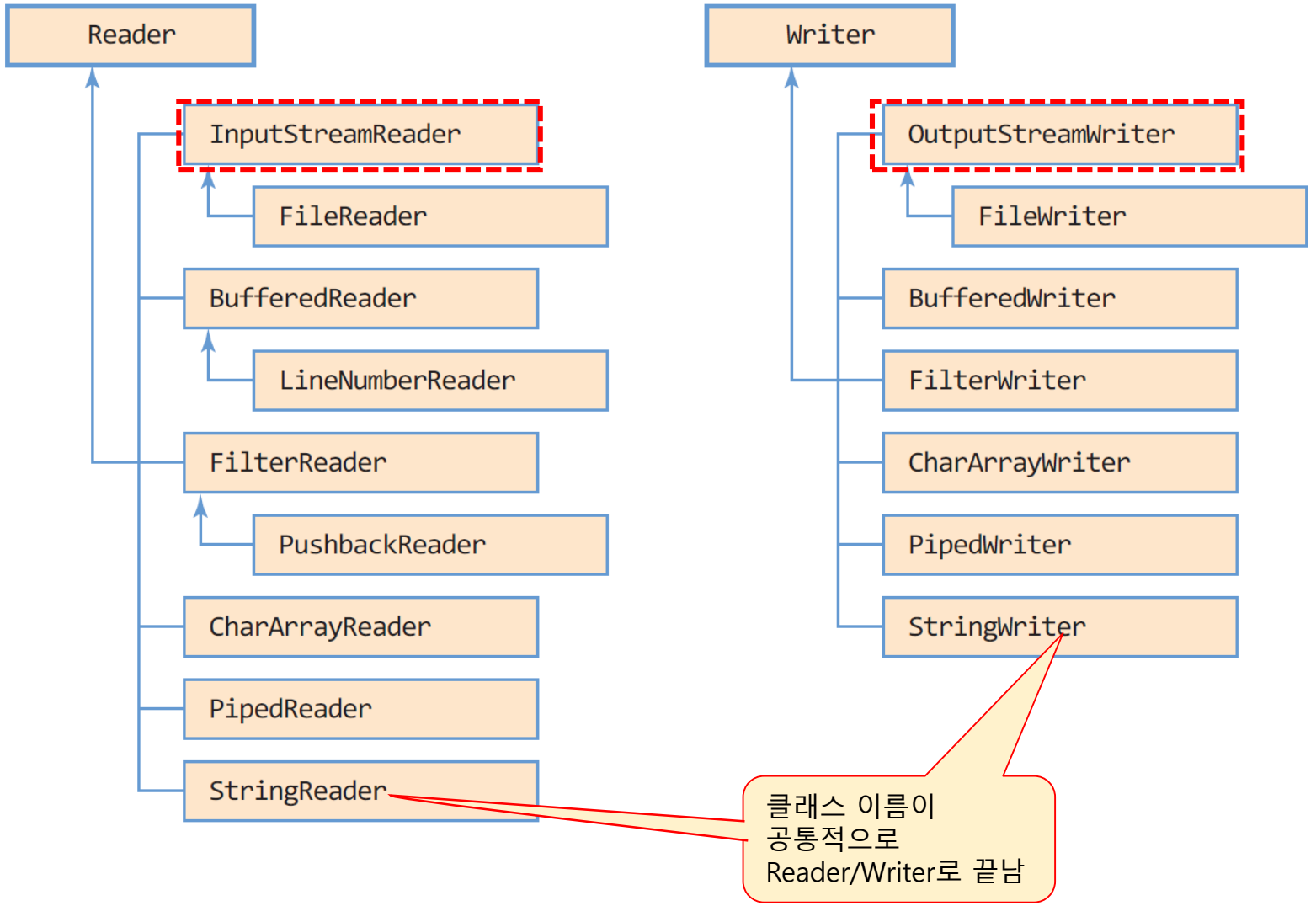
자바의 입출력 스트림 종류

- 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

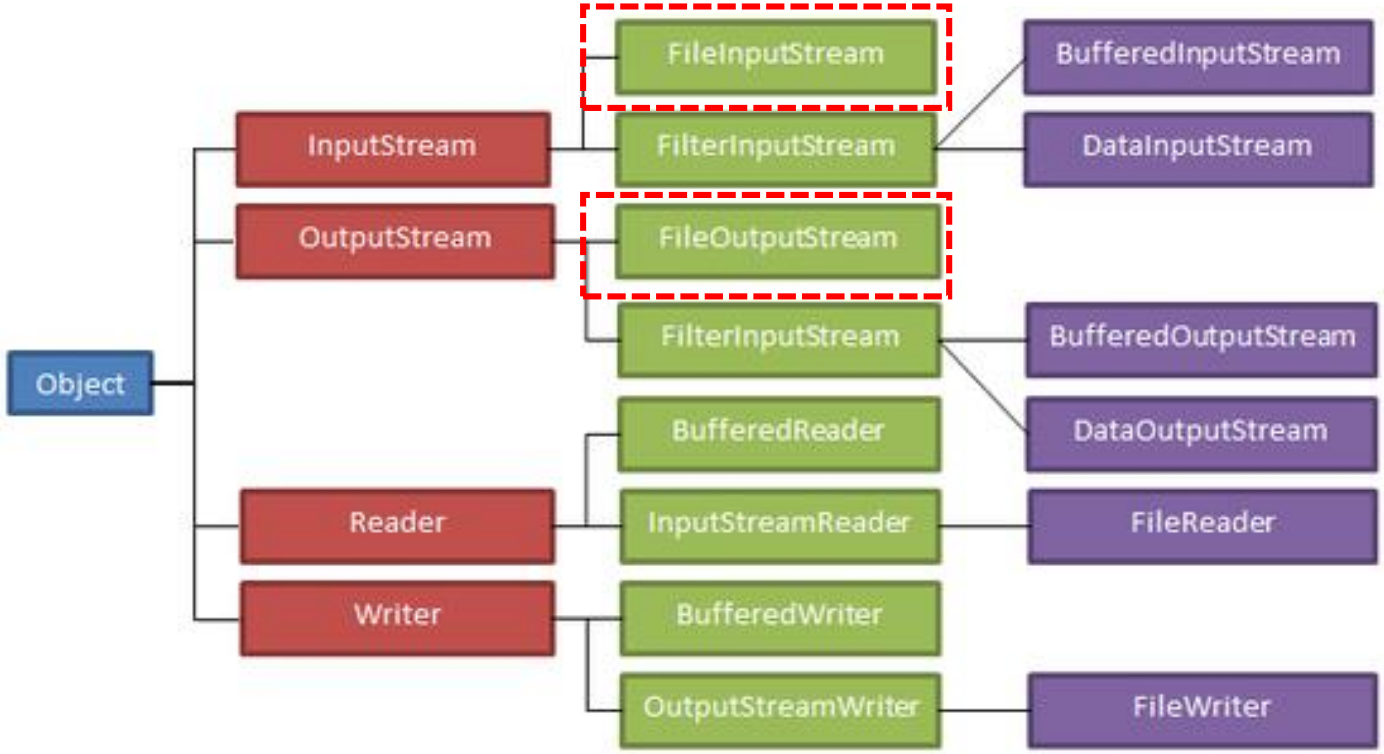
JDK의 바이트 스트림 클래스 계층 구조



JDK의 문자 스트림 클래스 계층 구조

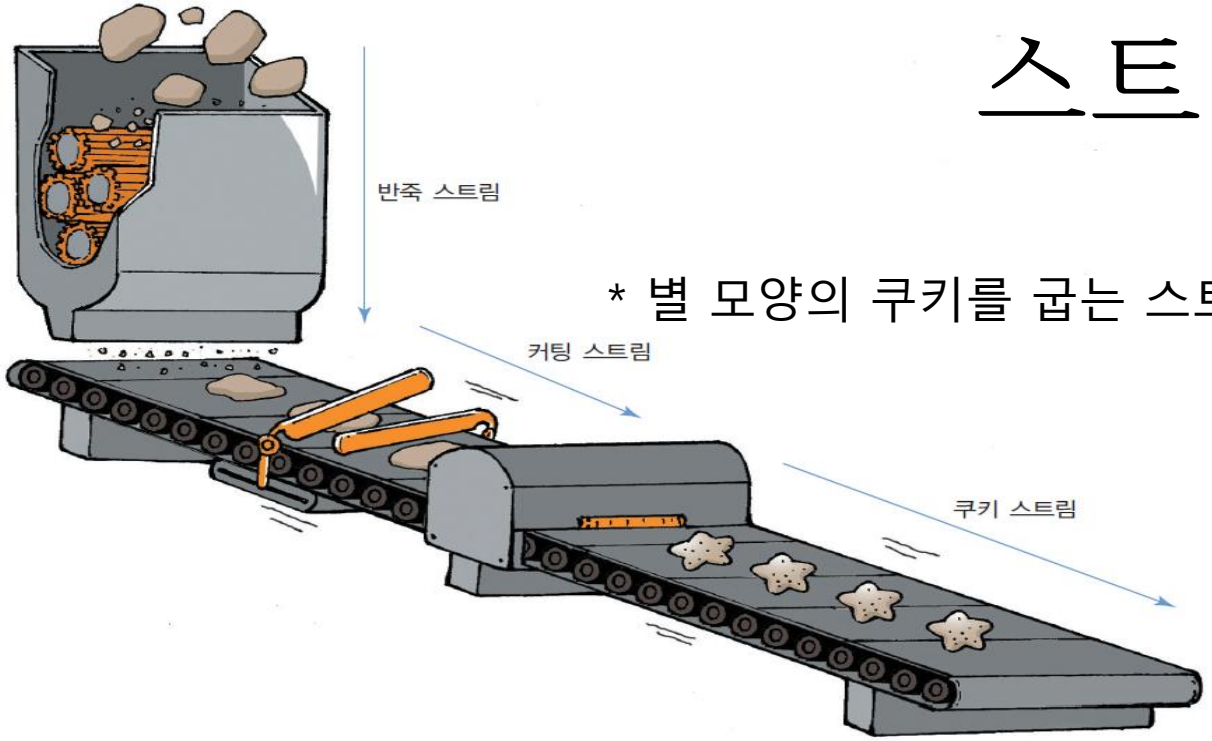


Input, Output 정리



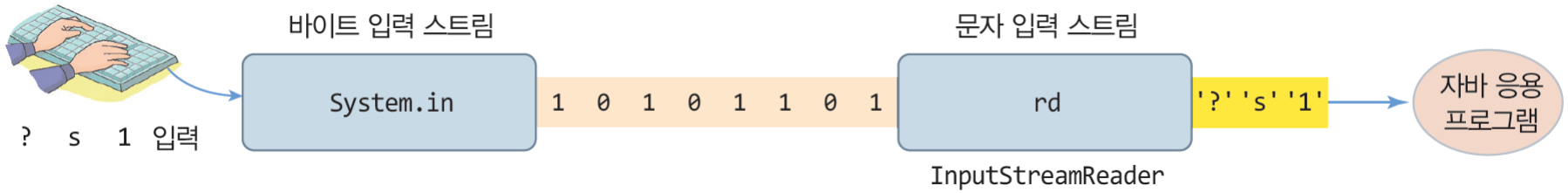
- InputStream/OutputStream – 바이트 단위 입출력의 최상위 클래스
- Reader/Writer – 문자 단위 입출력을 위한 최상위 스트림 클래스

스트림은 연결될 수 있다



* 별 모양의 쿠키를 굽는 스트림

* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결한 사례




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

문자 스트림

FileReader을 이용한 파일 읽기

- 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
FileReader fin = new FileReader ("c:₩₩test.txt");

int c;

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

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

}

fin.close();
```

C:₩test.txt 파일을 열고 파일과 입력 바이트 스트림 객체 fin 연결

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

바이트 c를 문자로 변환하여 화면에 출력

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

예제 8-1 : FileReader로 텍스트 파일 읽기

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

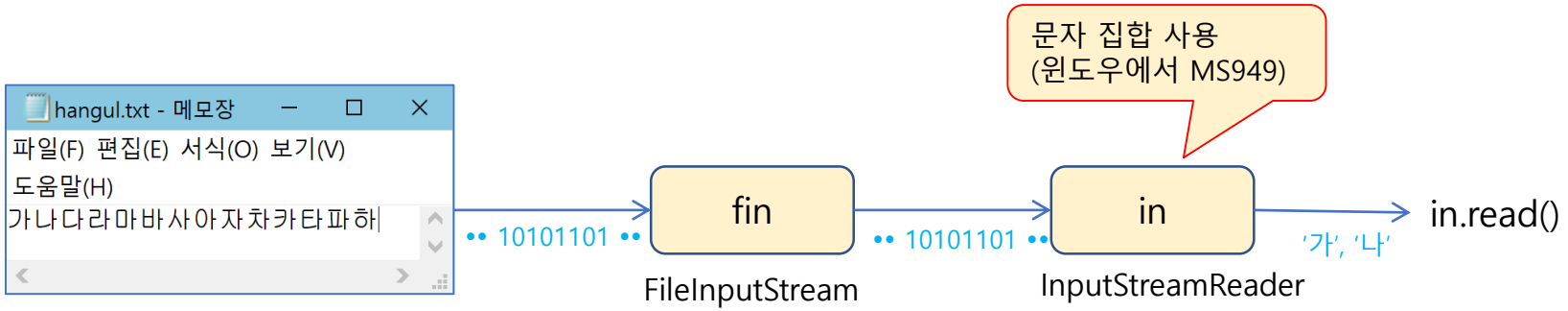
```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

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

[mci]
```

문자 집합과 InputStreamReader를 이용한 텍스트 파일 읽기

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

한글 완성형 확장형 문자 집합

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

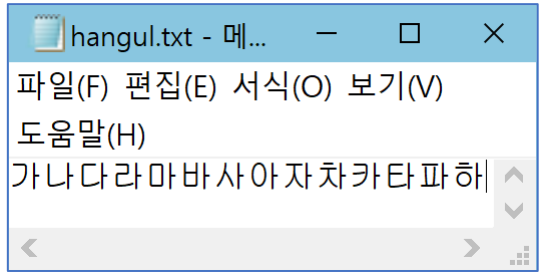


예제 8-3 : 한글 텍스트 파일 읽기(문자 집합 지정이 잘못된 경우)

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



hangul.txt

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

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

FileWriter 사용 예

- c:\Temp\test.txt로의 문자 출력 스트림 생성

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

- 파일 쓰기
  - ✓문자 단위 쓰기

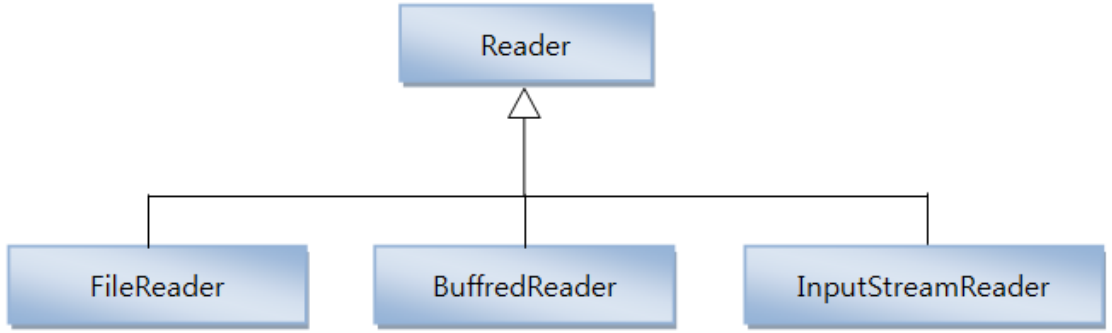
```
FileWriter fout = new FileWriter("c:₩₩Temp₩₩test.txt");
fout.write('A'); // 문자 'A' 출력
fout.close();
```

  - ✓블록 단위 쓰기

```
char [] buf = new char [1024];

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

입력 스트림과 출력 스트림

- Reader
  - ✓문자 기반 입력 스트림의 최상위 클래스로 추상 클래스

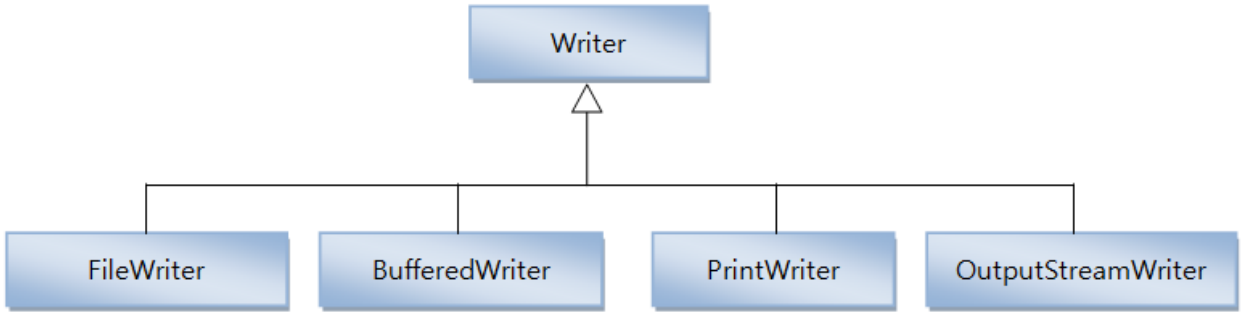


  - ✓Reader의 주요 메소드

메소드		설명
int	read()	입력 스트림으로부터 한개의 문자를 읽고 리턴한다.
int	read(char[] cbuf)	입력 스트림으로부터 읽은 문자들을 매개값으로 주어진 문자 배열 cbuf 에 저장하고 실제로 읽은 문자 수를 리턴한다.
int	read(char[] cbuf, int off, int len)	입력 스트림으로부터 len 개의 문자를 읽고 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 문자 수인 len 개를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

입력 스트림과 출력 스트림

- Writer
  - ✓문자 기반 출력 스트림의 최상위 클래스로 추상 클래스



  - ✓Writer의 주요 메소드

리턴타입	메소드	설명
void	wrtie(int c)	출력 스트림으로 매개값으로 주어진 한 문자를 보낸다.
void	write(char[] cbuf)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf 의 모든 문자를 보낸다.
void	write(char[] cbuf, int off, int len)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지의 문자를 보낸다.
void	write(String str)	출력 스트림에 매개값으로 주어진 문자열을 전부 보낸다.
void	write(String str, int off, int len)	출력 스트림에 매개값으로 주어진 문자열 off 순번부터 len 개까지의 문자를 보낸다.
void	flush()	버퍼에 잔류하는 모든 문자열을 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

바이트 스트림

바이트 스트림 클래스

- 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

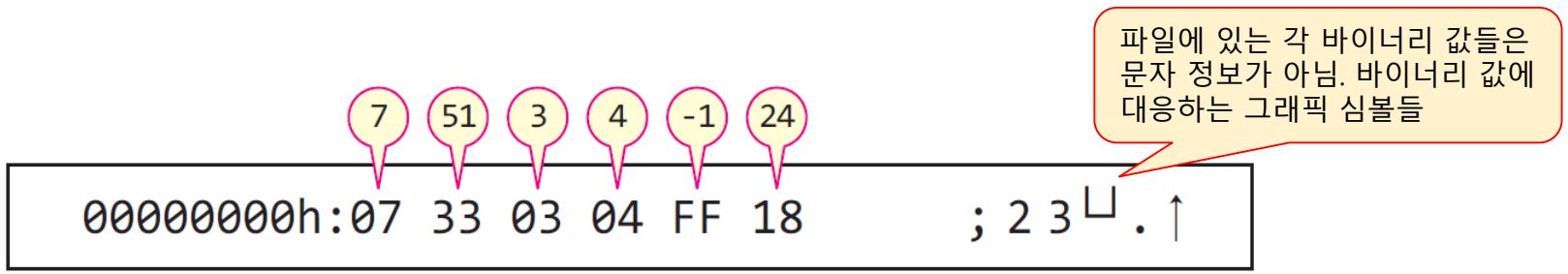
FileOutputStream을 이용한 파일 쓰기

- 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]);

fout.close();
```

파일에 배열 b[i]의 정수 값(바이너리)을 그대로 기록

스트림을 닫음. 파일도 닫힘. 더 이상 스트림으로부터 읽을 수 없음



test.out 파일의 내부