Stream, Buffer, File I/O

514760-1
2017년 가을학기
11/13/2017
박경신

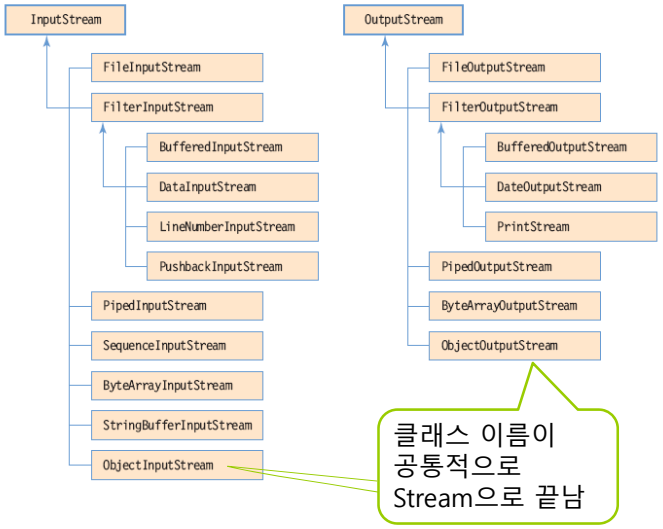
스트림

- 자바의 스트림
  - 자바 스트림은 입출력 장치와 자바 응용 프로그램 연결
  - 입출력 장치와 프로그램 사이의 데이터 흐름을 처리하는 소프트웨어 모듈
  - 입력 스트림
    - 입력 장치로부터 자바 프로그램으로 데이터를 전달하는 소프트웨어 모듈
  - 출력 스트림
    - 자바 프로그램에서 출력 장치로 데이터를 보내는 소프트웨어 모듈
- 입출력 스트림 기본 단위 : 바이트
- 자바 입출력 스트림 특징
  - 단방향 스트림, 선입선출 구조

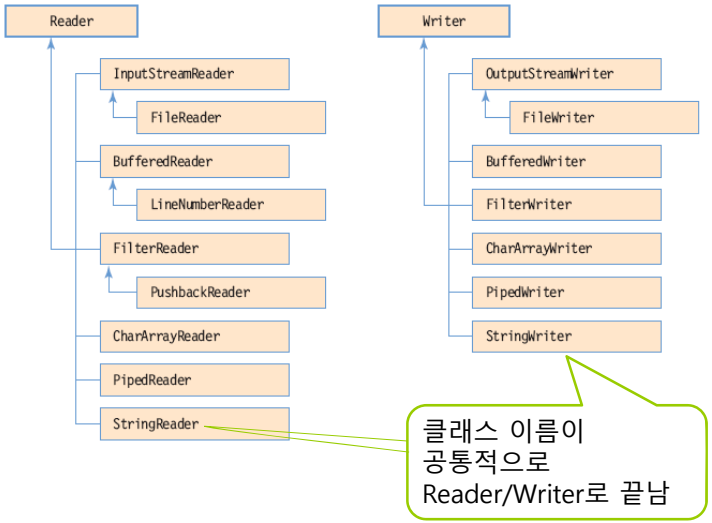
자바의 입출력 스트림 종류

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

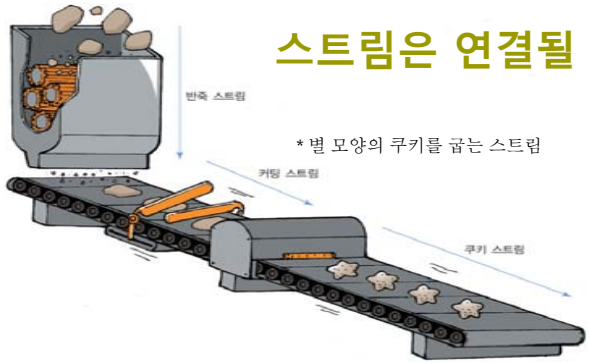
JDK의 바이트 스트림 클래스 계층 구조



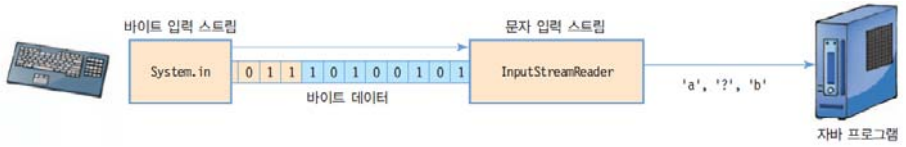
JDK의 문자 스트림 클래스 계층 구조



스트림은 연결될 수 있다



* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결하는 사례




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

바이트 스트림 클래스

- 바이트 스트림
  - 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
- 바이트 스트림 클래스
  - java.io 패키지에 포함
  - 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

FileInputStream을 이용한 파일 읽기

- 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
FileInputStream fin = new FileInputStream("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 : 윈도우에 있는 system.ini 파일을 읽어 화면에 출력하기

FileInputStream을 이용하여 사용자 컴퓨터의 windows 디렉터리에 있는 system.ini 파일을 읽고 화면에 출력하라. system.ini 파일은 텍스트 파일이다.

```
import java.io.*;
public class FileInputStreamEx {
  public static void main(String[] args) {
    FileInputStream in = null;
    try {
      in = new FileInputStream("c:₩₩windows₩₩system.ini");
      int c;
      while ((c = in.read()) != -1) {
        System.out.print((char)c);
      }
      in.close();
    } catch (IOException e) {
        System.out.println("입출력 오류");
    }
  }
}
```

파일 끝을 만나면 -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]
```

FileOutputStream을 이용한 파일 쓰기

- 바이너리 값을 파일에 저장하는 바이트 스트림 코드

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

int num[]={1,4,-1,88,50};
byte b[]={7,51,3,4,1,24};

for(int i=0; i<num.length; i++)
  fout.write(num[i]);

fout.write(b);

fout.close();
```

c:₩test.out 파일을 열고, 출력 바이트 스트림인 객체와 연결

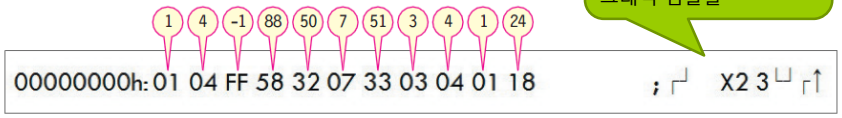
파일에 배열 num[i]의 정수 값(바이너리)을 그대로 기록

파일에 바이트 배열(바이너리) 값을 그대로 기록

스트림을 닫음. 파일도 닫힘. 더 이상 스트림으로부터 읽을 수 없음

FileOutputStream을 이용한 파일 쓰기

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



test.out 파일의 내부

예제 : FileOutputStream을 이용한 파일 쓰기

정수 타입의 결과 값을 FileOutputStream을 이용하여 파일에 저장한다. 다시 이 파일에서 정수형 변수로 읽고 이전에 계산된 결과 값과 같은지 확인하라.

```
import java.io.IOException;
public class FileOutputStreamEx {
  public static void main(String[] args) {
    try {
      FileOutputStream fout = new FileOutputStream("c:₩₩test.out");
      FileInputStream fin = null;

      for (int i=0; i<10; i++) {
        int n = 10-i; // 계산의 결과를 저장
        fout.write(n); // 파일에 결과값을 바이너리로 저장
      }
      fout.close(); //스트림을 닫는다.
```

예제 : FileOutputStream을 이용한 파일 쓰기

```
    fin = new FileInputStream("c:₩₩test.out");
    int c=0;
    while ((c = fin.read()) != -1) {
      System.out.print(c + " ");
    }
    fin.close();
  } catch (IOException e) {
    System.out.println("입출력 오류");
  }
 }
}
```

10 9 8 7 6 5 4 3 2 1

문자 스트림

- 문자 스트림
  - 유니 코드로 된 문자를 입출력 하는 스트림
    - 문자로 표현되지 않는 데이터는 다루지 않음
    - 문자 스트림은 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음 - 문자 스트림은 문자 데이터만 입출력 가능
- 문자 스트림을 다루는 클래스
  - Reader/Writer
    - java.io 패키지에 포함
    - 추상 클래스. 문자 스트림을 다루는 모든 클래스의 슈퍼 클래스
  - InputStreamReader/OutputStreamWriter
    - 바이트 스트림과 문자 스트림을 연결시켜주는 다리 역할
    - 지정된 문자집합 이용
    - InputStreamReader : 바이트를 읽어 문자로 인코딩
    - OutputStreamWriter : 문자를 바이트로 디코딩하여 출력
  - FileReader/FileWriter
    - 텍스트 파일에서 문자 데이터 입출력

예제 : FileReader를 이용한 텍스트 파일 읽기 - system.ini 파일 읽기

FileReader를 이용하여 사용자 컴퓨터의 windows 디렉터리에 있는 system.ini 파일을 읽고 화면에 출력하라. system.ini 파일은 텍스트 파일이다.

```
import java.io.*;
public class FileReaderEx {
  public static void main(String[] args) {
    FileReader in = null;
    try {
      // 파일로부터 문자 입력 스트림 생성
      in = new FileReader("c:₩₩windows₩₩system.ini");
      int c;
      while ((c = in.read()) != -1) { // 한 문자씩 읽는다.
            System.out.print((char)c);
      }
      in.close();
    }
    catch (IOException e) {
      System.out.println("입출력 오류");
    }    }  }
```

파일의 끝을 만나면 read()는 -1 리턴

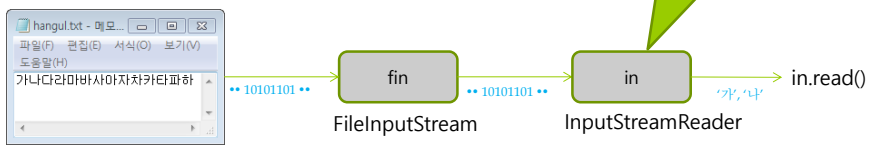
문자 집합과 InputStreamReader로 텍스트 파일 읽기

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

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

한글 확장 완성형 문자 집합

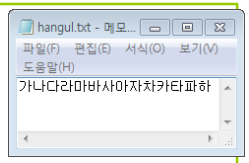
문자 집합 사용 (윈도우에서 MS949)



예제: 한글 텍스트 파일 읽기

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:₩₩tmp₩₩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"); } } }
```



hangul.txt

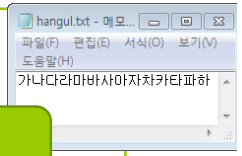
MS에서 만든 한글 확장 완성형 문자 집합

인코딩 문자 집합은 MS949
가나다라마바사아자차카타파하

예제: 문자 집합 지정이 잘못된 한글 텍스트 파일 읽기

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:₩₩tmp₩₩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"); } }}
```



angul.txt

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

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

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

FileWriter 사용 예

- c:₩test.txt 파일에 문자 출력 스트림을 생성하는 코드

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

- 파일에 문자 출력

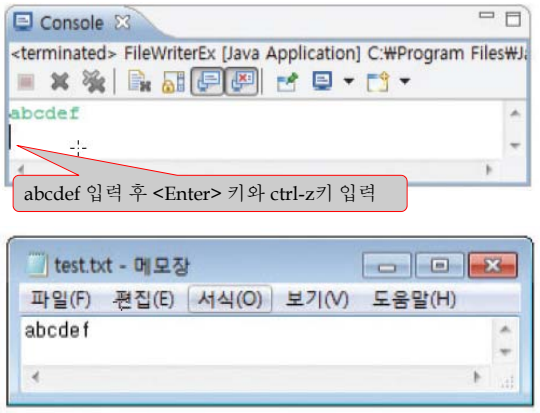
```
FileWriter fout = new
FileWriter("c:₩₩tmp₩₩test.txt");
fout.write('A'); // 문자 'A' 출력
fout.close();
```

예제: 키보드 입력을 파일로 저장하기

키보드 입력 데이터를 c:₩tmp₩test.txt 파일에 저장하는 코드를 작성하라.

```
import java.io.*;
public class FileWriterEx {
  public static void main(String[] args) {
    InputStreamReader in = new InputStreamReader(System.in);

    FileWriter fout = null;
    int c;
    try {
      fout = new FileWriter("c:₩₩tmp₩₩test.txt");
      while ((c = in.read()) != -1) {
        fout.write(c);
      }
      in.close();
      fout.close();
    } catch (IOException e) { System.out.println("입출력  오류");  }
  }
}
```

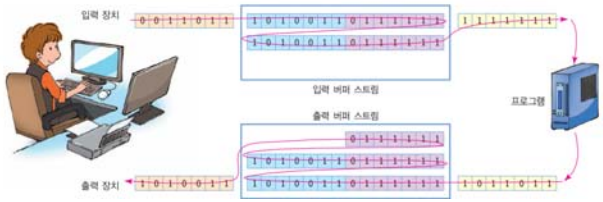
예제: 키보드 입력을 파일로 저장하기



실행 결과 test.txt 파일 생성

버퍼 입출력 스트림과 버퍼 입출력의 특징

- 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
    - 출력 시 여러 번 출력되는 데이터를 버퍼에 모아두고 한 번에 장치로 출력
    - 입력 시 입력 데이터를 버퍼에 모아두고 한번에 프로그램에게 전달



버퍼 스트림의 종류

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

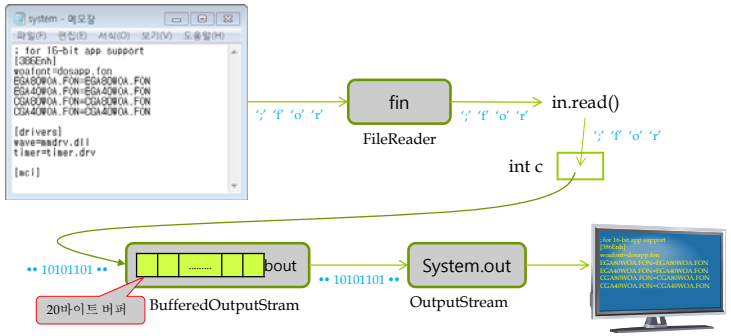
20바이트 버퍼를 가진 BufferedOutputStream

```
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: 버퍼 스트림을 이용하는 출력 예제

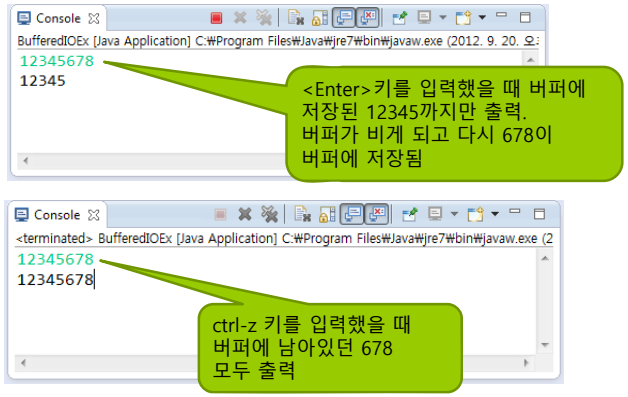
버퍼 크기를 5로 하고, 표준 출력 스트림과 연결된 버퍼 출력 스트림을 생성하라. 키보드에서 입력 받은 문자를 출력 스트림에 출력하고, 입력의 끝을 알리면(ctrl-z) 버퍼에 남아 있는 모든 문자를 출력하는 프로그램을 작성하라.

```
import java.io.*;
public class BufferedIOEx {
  public static void main(String[] args) {
    InputStreamReader in =  new InputStreamReader(System.in);
    BufferedOutputStream out = new BufferedOutputStream(System.out, 5);
    try {
      int c;
      while ((c = in.read()) != -1)
        out.write(c);
      out.flush(); // 버퍼에 남아 있던 문자 출력
      if (in != null) {
        in.close();
        out.close();
      }
    }
    catch (IOException e) {
      System.out.println("입출력 오류");
    }
```

ctrl-z가 입력될 때까지 반복

버퍼가 다 찰 때 문자가 화면에 출력

예제: 버퍼 스트림을 이용하는 출력 예제



File 클래스

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

File 클래스 생성자와 주요 메소드

메소드	설명
File(File parent, String child)	parent 디렉터리에 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(String pathname)	pathname이 나타내는 File 객체 생성
File(String parent, String child)	parent 디렉터리에 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(URI uri)	file:URI를 추상 경로명으로 변환하여 File 객체 생성

메소드	설명
boolean mkdir()	새로운 디렉터리 생성
String[] list()	디렉터리 내의 파일과 디렉터리 이름의 문자열 배열 리턴
File[] listFiles()	디렉터리 내의 파일 이름의 File 배열 리턴
boolean renameTo(File dest)	dest가 지정하는 파일 이름 변경
boolean delete()	파일 또는 디렉터리 삭제
long length()	파일의 크기 리턴. 디렉터리나 장치 파일인 경우 0 리턴
String getPath()	파일 경로명 전체를 문자열로 변환하여 리턴
String getName()	파일 또는 디렉터리 이름을 문자열로 리턴
boolean isFile()	일반 파일이면 true 리턴
boolean isDirectory()	디렉터리이면 true 리턴
long lastModified()	파일이 마지막으로 변경된 시간 리턴
boolean exists()	파일 또는 디렉터리가 존재하면 true 리턴

File 클래스 사용 예

파일 객체 생성

```
File f = new File("c:₩₩test.txt");
```

파일인지 디렉터리인지 구분

```
File f = new File("c:₩₩windows₩₩system.ini");
String res;
if(f.isFile()) // 파일 타입이면
  res = "파일";
else // 디렉터리 타입이면
  res = "디렉터리";
System.out.println(f.getPath() + "은 " + res + "입니다.");
```

```
c:₩windows₩system.ini은 파일입니다.
```

서브 디렉터리 리스트 얻기

```
File f = new File("c:₩₩tmp₩₩java_sample");
String[] filenames = f.list(); // 파일명 리스트 얻기
for (int i=0; i<filenames.length; i++) {
  File sf = new File(f, filenames[i]);
  System.out.print(filenames[i]);
  System.out.print("₩t파일 크기: " + sf.length());
}
```

예제: File 클래스 활용한 파일 관리

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

```
import java.io.File;
public class FileClassExample {
  // 디렉터리에 포함된 파일과 디렉터리의 이름,
  // 크기, 수정 시간을 출력하는 메소드
  public static void dir(File fd) {
    // 디렉터리에 포함된 파일 리스트 얻기
    String[] filenames = fd.list();
    for (String s : filenames) {
      File f = new File(fd, s);
      long t = f.lastModified(); // 마지막으로 수정된 시간
      System.out.print(s);
      System.out.print("₩t파일 크기: " + f.length()); // 파일 크기
      System.out.printf("₩t수정한 시간: %tb %td %ta %tT₩n",t, t, t, t);
    }
  }
```

예제: File 클래스 활용한 파일 관리

```
public static void main(String[] args) {
  File f1 = new File("c:₩₩windows₩₩system.ini");
  File f2 = new File("c:₩₩tmp₩₩java_sample");
  File f3 = new File("c:₩₩tmp");
  String res;
  if(f1.isFile()) // 파일 타입이면
    res = "파일";
  else // 디렉터리 타입이면
    res = "디렉터리";
  System.out.println(f1.getPath() + "은 " + res + "입니다.");
  if (!f2.exists()) { //f2가 나타내는 파일이 존재하는지 검사
    if (!f2.mkdir()) // 존재하지 않으면 디렉터리 생성
      System.out.println("디렉터리 생성 실패");
  }
```

예제: File 클래스 활용한 파일 관리

```
    if(f2.isFile()) // 파일 타입이면
      res = "파일";
    else // 디렉터리 타입이면
      res = "디렉터리";
    System.out.println(f2.getPath() + "은 " + res + "입니다.");
    dir(f3); // c:₩tmp에 있는 파일과 디렉터리 화면에 출력

    // 파일 이름 변경
    f2.renameTo(new File("c:₩₩tmp₩₩javasample"));
    dir(f3);
  }
}
```

C:₩tmp의 파일과 디렉터리 리스트

```
c:₩windows₩system.ini은 파일입니다.
c:₩tmp₩java_sample은 디렉터리입니다.
hangul.txt       파일 크기: 28  수정한 시간: 11월 29 일 21:04:46
Hello.java       파일 크기: 469 수정한 시간: 10월 06 수 13:23:59
Hello2010.java 파일 크기: 126 수정한 시간: 10월 06 수 10:01:56
HelloDoc.java 파일 크기: 669 수정한 시간: 10월 06 수 14:23:32
java_sample   파일 크기: 0   수정한 시간: 11월 14 일 16:46:27
hangul.txt       파일 크기: 28  수정한 시간: 11월 29 일 21:04:46
Hello.java       파일 크기: 469 수정한 시간: 10월 06 수 13:23:59
Hello2010.java 파일 크기: 126 수정한 시간: 10월 06 수 10:01:56
HelloDoc.java 파일 크기: 669 수정한 시간: 10월 06 수 14:23:32
javasample    파일 크기: 0   수정한 시간: 11월 14 일 16:46:27
```

예제: 텍스트 파일 복사

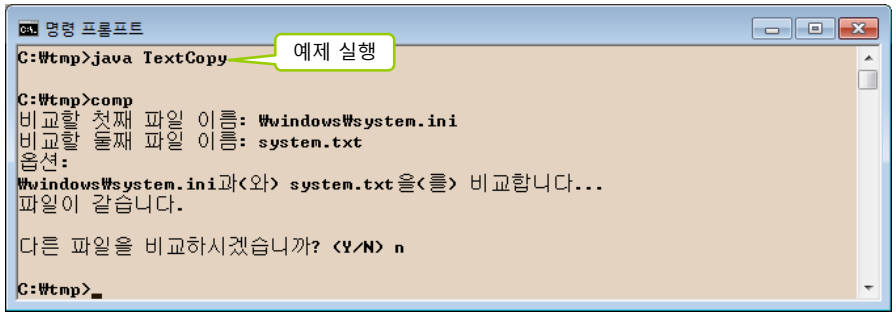
문자 스트림을 이용하여 텍스트 파일을 복사하는 프로그램을 작성하라

```
import java.io.*;
public class TextCopy {
  public static void main(String[] args){
    File src = new File("c:₩₩windows₩₩system.ini"); // 소스 파일
    File dst = new File("c:₩₩tmp₩₩system.txt"); // 목적 파일
    FileReader fr = null;
    FileWriter fw = null;
    BufferedReader in = null;
    BufferedWriter out = null;
    int c;
```

예제: 텍스트 파일 복사

```
    try {
      fr = new FileReader(src);
      fw = new FileWriter(dst);
      in = new BufferedReader(fr);
      out = new BufferedWriter(fw);
      while ((c = in.read()) != -1) {
        out.write((char)c);
      }
      in.close();
      out.close();
      fr.close();
      fw.close();
    } catch (IOException e) {
      System.out.println("파일 복사 오류");
    }
  }
}
```

예제: 텍스트 파일 복사



예제: 바이너리 파일 복사

바이트 스트림을 이용하여 바이너리 파일을 복사하는 프로그램을 작성하라

```
import java.io.*;
public class BinaryCopy {
  public static void main(String[] args) {
    File src = new File("c:₩₩windows₩₩explorer.exe"); // 소스 파일
    File dst = new File("c:₩₩tmp₩₩explorer.bin"); // 목적 파일
    FileInputStream fi = null;
    FileOutputStream fo = null;
    BufferedInputStream in = null;
    BufferedOutputStream out = null;
    int c;
```

예제: 바이너리 파일 복사

```
    try {
      fi = new FileInputStream(src);
      fo = new FileOutputStream(dst);
      in = new BufferedInputStream(fi);
      out = new BufferedOutputStream(fo);
      while ((c = in.read()) != -1) {
        out.write((char)c);
      }
      in.close();
      out.close();
      fi.close();
      fo.close();
    } catch (IOException e) {
      System.out.println("파일 복사 오류");
    }
  }
}
```

예제: 바이너리 파일 복사