10장. 파일 입출력에 사용되는 자바 클래스들

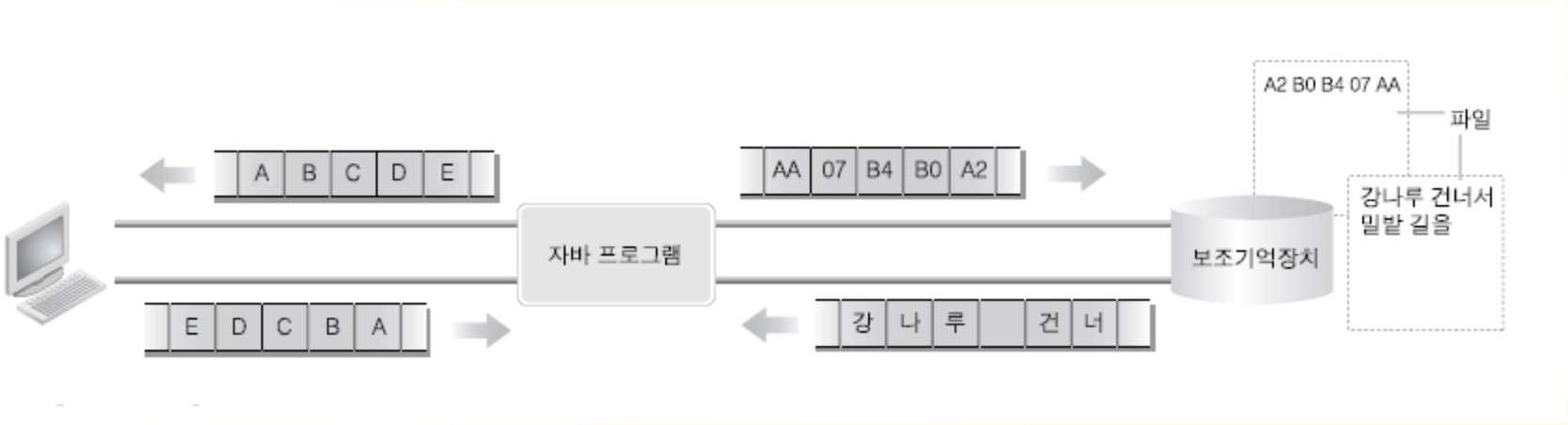
학습 목표

- JDK 라이브러리의 파일을 다루는 클래스들
- 파일의 내용을 읽고 쓰는 클래스들
- 입출력 기능/성능을 향상시키는 클래스들
- 데이터를 포맷해서 출력하는 클래스들
- 파일 관리에 사용되는 클래스

01. 파일을 다루는 자바 클래스들

스트림이란?

- 일차원적인 데이터의 흐름



- 흐름의 방향에 따른 분류
  - 입력 스트림(input stream)
  - 출력 스트림(output stream)
- 데이터의 형태에 따른 분류
  - 문자 스트림(character stream)
  - 바이트 스트림(byte stream)

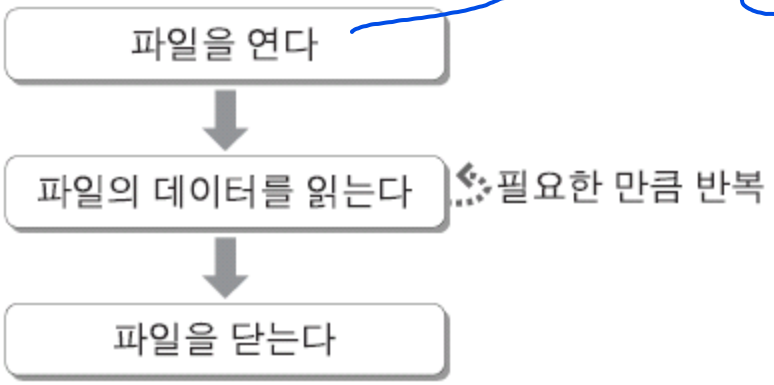
02. 파일의 내용을 읽고 쓰는 클래스들

파일 입출력의 과정

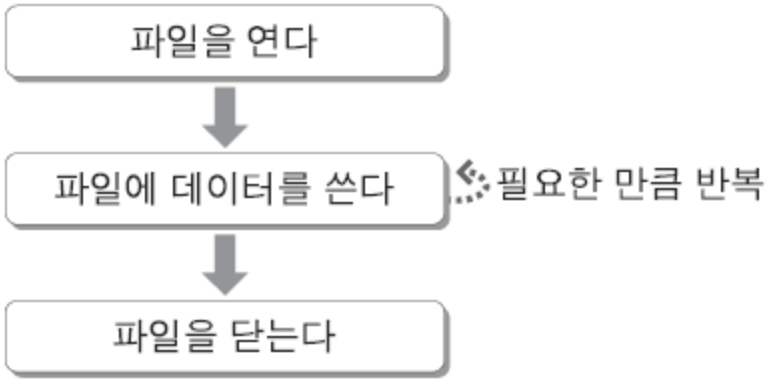
- 3단계로 이루어짐

파일객체 생성

a) 파일로부터 데이터를 읽는 3단계



b) 파일에 데이터를 쓰는 3단계



02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

- FileReader 클래스 : 텍스트 파일을 읽는 클래스
- 사용 방법

1) 1단계: 파일을 엽니다

```
FileReader reader = new FileReader("poem.txt");
```

생성자 안에서 현재 디렉토리의 poem.txt 파일을 엽니다.

02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

- 사용 방법

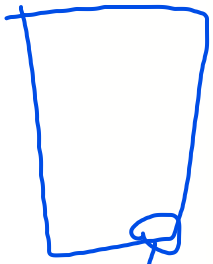
  2) 2단계: 파일을 읽습니다

char : 16 bit

int

```
data = reader.read();
```

이 메서드는 파일에 있는 문자 하나를 읽어서 리턴합니다.



파일 끝 문자는 -1 → EOF (End Of File)

```
while (true) {
    int data = reader.read();          // 데이터를 읽어서
    if (data < 0)                      // 마이너스 값이면 반복을 중단하고,
        break;
    char ch = (char) data;             // 아니면 char 타입으로 캐스트합니다.
    ...                                // 데이터 처리 로직이 들어가는 부분
}
```

02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

- 사용 방법

  3) 3단계: 파일을 닫습니다

```
reader.close();
```

파일을 닫는 메소드

02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

[예제 10-1] 텍스트 파일을 읽는 프로그램 – 미완성 (1)

```
1   import java.io.*;
2   class ReaderExample1 {
3       public static void main(String args[]) {
4           FileReader reader = new FileReader("poem.txt");     ----- 파일을 여는 부분
5           while (true) {
6               int data = reader.read();
7               if (data == -1)
8                   break;                                      ----- 파일을 읽어서 처리하는 부분
9               char ch = (char) data;
10              System.out.print(ch);
11          }
12          reader.close();   ------------------------------------ 파일을 닫는 부분
13      }
14  }
```

02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

[예제 10-2] 텍스트 파일을 읽는 프로그램 – 미완성 (2)

```
1   import java.io.*;
2   class ReaderExample1 {
3       public static void main(String args[]) {
4           try {
5               FileReader reader = new FileReader("poem.txt");
6               while (true) {
7                   int data = reader.read();
8                   if (data == -1)
9                       break;
10                  char ch = (char) data;
11                  System.out.print(ch);
12              }
13              reader.close();
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (IOException ioe) {
19              System.out.println("파일을 읽을 수 없습니다.");
20          }
21      }
22  }
```

익셉션이 발생하는 부분을 try 문으로 묶었습니다.

FileReader의 생성자가 발생하는 익셉션을 처리

FileReader의 read, close 메소드가 발생하는 익셉션을 처리

02. 파일의 내용을 읽고 쓰는 클래스들

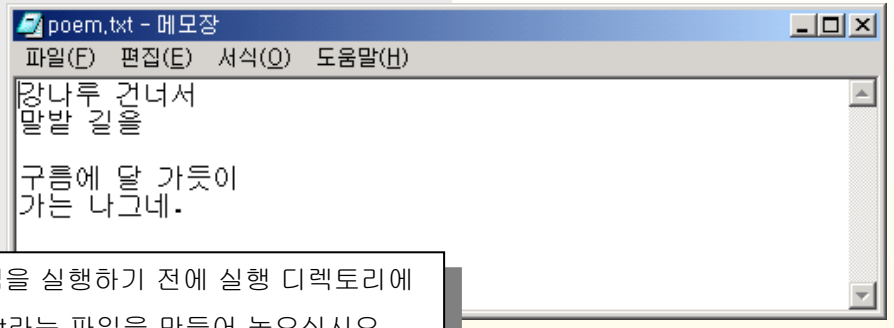
FileReader 클래스

[예제 10-3] 텍스트 파일을 읽는 프로그램 – 완성

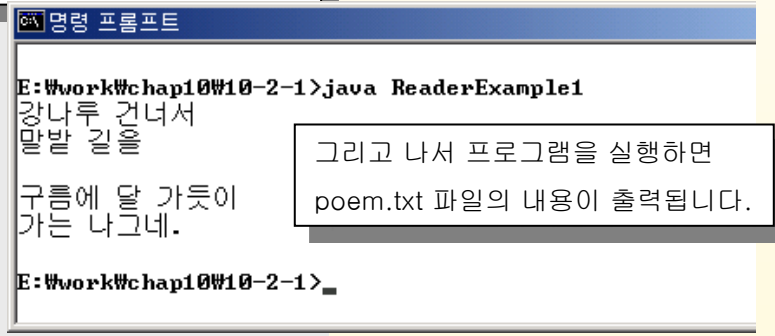
```
1   import java.io.*;
2   class ReaderExample1 {
3       public static void main(String args[]) {
4          FileReader reader = null;
5           try {
6               reader = new FileReader("poem.txt");
7               while (true) {
8                   int data = reader.read();
9                   if (data == -1)
10                      break;
11                  char ch = (char) data;
12                  System.out.print(ch);
13              }
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (IOException ioe) {
19              System.out.println("파일을 읽을 수 없습니다.");
20          }
21          finally {
22             try {
23                 reader.close();
24             }
25             catch (Exception e) {
26             }
27          }
28      }
29  }
```

파일을 닫는 명령문을 finally 블록으로 이동

close 메소드가 발생하는 익셉션을 처리



프로그램을 실행하기 전에 실행 디렉토리에 poem.txt라는 파일을 만들어 놓으십시오.



그리고 나서 프로그램을 실행하면 poem.txt 파일의 내용이 출력됩니다.

02. 파일의 내용을 읽고 쓰는 클래스들

FileReader 클래스

- 한꺼번에 여러 문자를 읽는 read 메소드

```
int num = reader.read(arr);
```

파일로부터 읽은 문자를 담을 char 배열

[올바른 예]

```
char arr[] = new char[100];
int num = reader.read(arr);
```

[잘못된 예] 저장 공간 X

```
char arr[];
int num = reader.read(arr);
```

02. 파일의 내용을 읽고 쓰는 클래스들

FileWriter 클래스

- **FileWriter 클래스** : 텍스트를 파일에 쓰는 클래스
- 사용 방법

1) 1단계: 파일을 엽니다

```
FileWriter writer = new FileWriter("output.txt");
```

현재 디렉토리에 output.txt라는 파일을 새로 만들어서 엽니다.

02. 파일의 내용을 읽고 쓰는 클래스들

FileWriter 클래스

- 사용 방법

2) 2단계: 파일에 문자를 씁니다

```
writer.write(ch);
```

이 문자를 파일에 씁니다.

3) 3단계: 파일을 닫습니다

```
writer.close();
```

파일을 닫는 메소드

02. 파일의 내용을 읽고 쓰는 클래스들

FileWriter 클래스

[예제 10-4] FileWriter 클래스로 문자 데이터를 파일에 쓰는 프로그램

```
1   import java.io.*;
2   class WriterExample1 {
3       public static void main(String args[]) {
4           FileWriter writer = null;
5           try {
6               writer = new FileWriter("output.txt");  ---------- 파일을 엽니다.
7               char arr[] = { '뇌', '를', ' ', '자', '극', '하', '는', ' ', 'J', 'a', 'v', 'a' };
8               for (int cnt = 0; cnt < arr.length; cnt++)
9                   writer.write(arr[cnt]);  ------ 파일에 반복해서 문자들을 씁니다.
10          }
11          catch (IOException ioe) {
12              System.out.println("파일로 출력할 수 없습니다.");
13          }
14          finally {
15              try {
16                  writer.close();  ---------- 파일을 닫습니다.
17              }
18              catch (Exception e) {
19              }
20          }
21      }
22  }
```

02. 파일의 내용을 읽고 쓰는 클래스들

FileWriter 클래스

- 한꺼번에 여러 문자를 출력하는 write 메소드

```
writer.write(arr);
```

arr 배열에 있는 모든 문자들을 파일로 출력

char 배열

02. 파일의 내용을 읽고 쓰는 클래스들

FileOutputStream 클래스 바이트 스트림

- **FileOutputStream 클래스** : 바이트 데이터를 파일에 쓰는 클래스
- 사용 방법

1) 1단계: 파일을 엽니다

* 방법은 FileWriter 클래스와 동일

2) 2단계: 파일에 데이터를 씁니다

```
outputStream.write(71);
```

71의 비트 패턴인 01000111을 갖는 하나의 바이트를 파일로 출력

3) 3단계: 파일을 닫습니다

* 방법은 FileWriter 클래스와 동일

02. 파일의 내용을 읽고 쓰는 클래스들

FileOutputStream 클래스

[예제 10-5] FileOutputStream 클래스로 바이트 데이터를 파일에 쓰는 프로그램

```
1   import java.io.*;
2   class OutputStreamExample1 {
3       public static void main(String args[]) {
4           FileOutputStream out = null;
5           try {
6               out = new FileOutputStream("output.dat");  ---------- 파일을 엽니다.
7               byte arr[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
8                              10, 11, 12, 13, 14, 15, 16, 17, 18, 19 };
9               for (int cnt = 0; cnt < arr.length; cnt++)
10                  out.write(arr[cnt]);                  ------ 파일에 반복해서 byte 타입 데이터를 씁니다.
11          }
12          catch (IOException ioe) {
13              System.out.println("파일로 출력할 수 없습니다.");
14          }
15          finally {
16              try {
17                  out.close(); ------------------------------- 파일을 닫습니다.
18              }
19              catch (Exception e) {
20              }
21          }
22      }
23  }
```

02. 파일의 내용을 읽고 쓰는 클래스들

FileInputStream 클래스

- **FileInputStream 클래스** : 파일로부터 바이트 단위로 데이터를 읽는 클래스
- 사용 방법

1) 1단계: 파일을 엽니다

* 방법은 FileReader 클래스와 동일

2) 2단계: 파일로부터 데이터를 읽습니다

byte 형식으로 읽어서 int 형으로 반환

```
while (true) {
    int data = inputStream.read();   ← 데이터를 읽어서
    if (data < 0)                    ← -1이면 반복을 중단하고
        break;
    byte b = (byte) data;            ← 아니면 byte 타입으로 캐스트
    ...
}
```

3) 3단계: 파일을 닫습니다

* 방법은 FileReader 클래스와 동일

02. 파일의 내용을 읽고 쓰는 클래스들

FileInputStream 클래스

- 한꺼번에 여러 바이트를 읽는 read 메소드

```
byte arr = new byte[16];

int num = inputStream.read(arr);
```

byte 타입의 배열을 생성해서 넘겨줘야 합니다.

읽은 바이트수

02. 파일의 내용을 읽고 쓰는 클래스들

FileInputStream 클래스

[예제 10-6] 파일의 내용을 읽어서 16진수로 출력하는 프로그램

```
1   import java.io.*;
2   class FileDump {
3       public static void main(String args[]) {
4           if (args.length < 1) {
5               System.out.println("Usage: java FileDump <filename>");
6               return;
7           }
8           FileInputStream in = null;
9           try {
10              in = new FileInputStream(args[0]);  ------------ 파일을 엽니다.
11              byte arr[] = new byte[16];
12              while (true) {
13                  int num = in.read(arr);  ----------------- 파일로부터 16바이트씩 읽습니다.
14                  if (num < 0)
15                      break;
16                  for (int cnt = 0; cnt < num; cnt ++)
17                      System.out.printf("%02X ", arr[cnt]);   } 읽어들인 바이트 데이터를 16진수로 출력합니다.
18                  System.out.println();
19              }
20          }
21          catch (FileNotFoundException fnfe) {
22              System.out.println(args[0] + " 파일이 존재하지 않습니다.");
23          }
24          catch (IOException ioe) {
25              System.out.println(args[0] + " 파일을 읽을 수 없습니다.");
26          }
27          finally {
28              try {
29                  in.close();  ------------------------- 파일을 닫습니다.
30              }
31              catch (Exception e) {
32              }
33          }
34      }
35  }
```

03. 입출력 기능/성능 향상 클래스들

입출력 기능/성능을 향상시키는 클래스들

파일객체에 다 줄수 없음

클래스 이름	설명
DataInputStream	프리미티브 타입의 데이터를 입출력하는 클래스
DataOutputStream	
ObjectInputStream	프리미티브 타입과 레퍼런스 타입의 데이터를 입출력하는 클래스
ObjectOutputStream	
BufferedReader	데이터를 한꺼번에 읽어서 버퍼에 저장해두는 클래스
BufferedInputStream	
BufferedWriter	데이터를 버퍼에 저장해두었다가 한꺼번에 출력하는 클래스
BufferedOutputStream	
LineNumberReader	텍스트 파일의 각 행에 번호를 붙여가면서 읽는 클래스

- 이 클래스들은 모두 java.io 패키지에 속함
- 이 클래스들은 단독으로는 사용될 수 없음

행 버퍼

03. 입출력 기능/성능 향상 클래스들

버퍼를 이용해서 입출력 성능을 향상시키는 클래스들

- 스트림의 종류에 따라 4개의 클래스가 있음

클래스 이름	설명
BufferedInputStream	바이트 입력 스트림을 버퍼링하는 클래스
BufferedOutputStream	바이트 출력 스트림을 버퍼링하는 클래스
BufferedReader	문자 입력 스트림을 버퍼링하는 클래스
BufferedWriter	문자 출력 스트림을 버퍼링하는 클래스

03. 입출력 기능/성능 향상 클래스들

BufferedInputStream 클래스

- **BufferedInputStream 클래스** : 바이트 입력 스트림의 성능을 향상시키는 클래스
- 사용 방법

1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileInputStream in1 = new FileInputStream("input.dat");
BufferedInputStream in2 = new BufferedInputStream(in1);
```

FileInputStream 객체를 생성해서 BuffredInputStream 생성자의 파라미터로 사용합니다.

2) read 메소드를 호출하여 데이터를 읽습니다.

* FileInputStream 클래스의 read 메소드 호출 방법과 동일

3) 파일을 닫습니다.

* FileInputStream 클래스의 close 메소드 호출 방법과 동일

03. 입출력 기능/성능 향상 클래스들

BufferedInputStream 클래스

[예제 10-11] BufferedInputStream로 성능을 향상시킨 FileDump 프로그램

```
1   import java.io.*;
2   class FileDump {
3       public static void main(String args[]) {
4           if (args.length < 1) {
5               System.out.println("Usage: java FileDump <filename>");
6               return;
7           }
8           BufferedInputStream in = null;
9           try {
10              in = new BufferedInputStream(new FileInputStream(args[0]));
11              byte arr[] = new byte[16];
12              while (true) {
13                  int num = in.read(arr);
14                  if (num < 0)
15                      break;
16                  for (int cnt = 0; cnt < num; cnt ++)
17                       System.out.printf("%02X ", arr[cnt]);
18                  System.out.println();
19              }
20          }
21          catch (FileNotFoundException fnfe) {
22              System.out.println(args[0] + " 파일이 존재하지 않습니다.");
23          }
24          catch (IOException ioe) {
25              System.out.println(args[0] + " 파일을 읽을 수 없습니다.");
26          }
27          finally {
28              try {
29                   in.close();
30              }
31              catch (Exception e) {
32              }
33          }
34      }
35  }
```

이 두 부분을 제외한 나머지 부분은 [예제 10-6]과 동일합니다.

03. 입출력 기능/성능 향상 클래스들

BufferedInputStream 클래스

- 버퍼의 크기를 지정하는 방법

잘 안씀

```
BufferedInputStream in2 = new BufferedInputStream(in1, 1024);
```

in1: FileInputStream 객체
1024: 버퍼의 크기

03. 입출력 기능/성능 향상 클래스들

BufferedWriter 클래스

- **BufferedWriter 클래스** : 문자 출력 스트림의 성능을 향상시키는 클래스
- 사용 방법

1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileWriter writer1 = new FileWriter("output.txt");
BufferedWriter writer2 = new BufferedWriter(writer1);
```

FileWriter 객체를 생성해서
BuffredWriter 생성자의 파라미터로 사용합니다.

2) write 메소드를 호출하여 데이터를 출력합니다.

* FileWriter 클래스의 write 메소드 호출 방법과 동일

3) 파일을 닫습니다.

* FileWriter 클래스의 close 메소드 호출 방법과 동일

03. 입출력 기능/성능 향상 클래스들

BufferedWriter 클래스

[예제 10-12] BufferedWriter 클래스로 데이터 출력 성능을 향상시킨 프로그램

```
1   import java.io.*;
2   class WriterExample1 {
3       public static void main(String args[]) {
4           BufferedWriter writer = null;
5           try {
6               writer = new BufferedWriter(new FileWriter("output.txt"));
7               char arr[] = { '뇌', '를', ' ', '자', '극', '하', '는', ' ', 'J', 'a', 'v', 'a' };
8               for (int cnt = 0; cnt < arr.length; cnt++)
9                   writer.write(arr[cnt]);
10          }
11          catch (IOException ioe) {
12              System.out.println("파일로 출력할 수 없습니다.");
13          }
14          finally {
15              try {
16                  writer.close();
17              }
18              catch (Exception e) {
19              }
20          }
21      }
22  }
```

이 두 부분을 제외한 나머지 부분은 [예제 10-4]와 동일합니다.

03. 입출력 기능/성능 향상 클래스들

BufferedWriter 클래스

- 버퍼에 있는 데이터를 파일에 즉시 출력하는 flush메소드

강제 비우기

```
writer.flush();
```

↑ 호출되는 즉시 버퍼의 데이터를 모두 출력하는 메소드

05. 파일 관리에 사용되는 File 클래스

File 클래스

- **File 클래스** : (파일의 내용이 아니라) 파일 자체를 관리하는 클래스
- 다음과 같은 메소드 제공
  - 파일 정보를 가져오는 메소드
  - 파일 정보를 수정하는 메소드
  - 파일을 생성/삭제하는 메소드

- 디렉토리 관리에도 사용됨

특수한 형태의 파일이라고 할 수 있음

05. 파일 관리에 사용되는 File 클래스

파일/디렉토리 정보 가져오기

- 사용 방법

  1) 다음과 같은 방법으로 File 객체를 생성합니다.

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

현재 디렉토리에 있는 poem.txt에 대한 File 객체를 생성합니다.

```
File file = new File("C:\\work\\chap10");
```

C 드라이브의 work 디렉토리 아래에 있는 chap10에 대한 File 객체를 생성합니다.

05. 파일 관리에 사용되는 File 클래스

파일/디렉토리 정보 가져오기

- 사용 방법

  2) 파일/디렉토리 정보를 가져오는 메소드를 호출합니다.

```
Boolean isThere = file.exists();
```

파일 또는 디렉토리가 있으면 true,
없으면 false를 리턴

```
Boolean isFile = file.isFile();
```

파일이면 true,
아니면 false를 리턴

```
Boolean isDir = file.isDirectory();
```

디렉토리면 true,
아니면 false를 리턴

05. 파일 관리에 사용되는 File 클래스

파일/디렉토리 정보 가져오기

- 사용 방법

2) 파일/디렉토리 정보를 가져오는 메소드를 호출합니다. (계속)

```
String name = file.getName();              // 이름을 리턴
long size = file.length();                 // 크기를 리턴
long time = file.lastModified();           // 최종 수정일시를 리턴
boolean readMode = file.canRead();         // 읽기 가능 여부를 리턴
boolean writeMode = file.canWrite();       // 쓰기 가능 여부를 리턴
boolean hiddenMode = file.isHidden();      // 숨김 여부를 리턴
String parent = file.getParent();          // 부모 디렉토리 경로명을 리턴
```

```
Files childs[] = file.listFiles();
```

서브디렉토리와 파일들의 목록을 리턴하는 메소드

05. 파일 관리에 사용되는 File 클래스

파일/디렉토리 정보 가져오기

[예제 10-16] 현재 디렉토리의 서브디렉토리와 파일 목록을 출력하는 프로그램

```
1   import java.io.*;
2   import java.util.*;
3   class FileExample1 {
4       public static void main(String args[]) {
5           File file = new File(".");
6           File arr[] = file.listFiles();
7           for (int cnt = 0; cnt < arr.length; cnt++) {
8               String name = arr[cnt].getName();
9               if (arr[cnt].isFile())
10                  System.out.printf("%-25s %7d ", name, arr[cnt].length());
11              else
12                  System.out.printf("%-25s   <DIR> ", name);
13              long time = arr[cnt].lastModified();
14              GregorianCalendar calendar = new GregorianCalendar();
15              calendar.setTimeInMillis(time);
16              System.out.printf("%1$tF %1$tT %n", calendar);
17          }
18      }
19  }
```

현재 디렉토리 경로명을 가지고 File 객체를 생성합니다.

서브디렉토리와 파일 목록을 가져옵니다.

가져온 서브디렉토리와 파일의 이름, 크기, 최종수정 일시를 출력합니다.

05. 파일 관리에 사용되는 File 클래스

파일/디렉토리 생성/삭제하기

- 파일을 생성하고 삭제하는 메소드

```
File file1 = new File("poem.txt");
file1.createNewFile();
```

현재 디렉토리에 poem.txt라는 이름의 파일을 생성합니다.

```
File file2 = new File("C:\\doc\\회의록.hwp");
file2.delete();
```

C:\doc\회의록.hwp 파일을 삭제합니다.

- 디렉토리를 생성하고 삭제하는 메소드

```
File file1 = new File("C:\\올빼미");
file1.mkdir();
```

C 드라이브의 루트 디렉토리에 "올빼미" 라는 디렉토리를 생성합니다.

```
File file2 = new File("두루미");
file2.delete();
```

현재 디렉토리에 있는 "두루미" 라는 디렉토리를 삭제합니다