File과 Files 클래스

▣ File과 Files 클래스

- java.io 패키지 와 java.nio.file 패키지는 파일과 디렉토리 정보를 가지고 있는 File 과 Files 클래스를 제공한다.

- Files는 File을 개선한 클래스로, 좀 더 많은 기능을 가지고 있다.

> 자바에서는 File 클래스를 통해서 파일과 디렉터리를 다룰 수 있도록 하고 있다.
> 그래서 File 인스턴스는 파일 일 수도 있고, 디렉터리 일 수도 있다.

◎ File 클래스

- 파일의 경로명을 다루는 클래스이며, 파일 이름 변경/삭제/디렉터리 새성/크기 등 파일 관리기능을 가지고 있다.

- 경로 구분자는 운영체제마다 조금씩 다르다.
  (윈도우는 ₩₩ 또는 / 둘 다 사용할 수 있고, 맥OS, 리눅스는 / 를 사용한다.)

- File 객체를 생성했다고 해서 파일이나 디렉토리가 생성되는 것은 아니다.
- 그리고 경로에 실제 파일이나 디렉토리가 없더라도 예외가 발생하지 않는다.
- File 객체는 파일 읽고/쓰기 기능이 없다.

▷ 파일이나 디렉토리가 실제 있는지 확인하고 싶다면, File 객체를 생성하고 나서 exists() 메소드를 호출해보면 된다.

```
File file = new File("C:/Temp/file.txt");
File file = new File("C:\\Temp\\file.txt");
```

```
boolean isExist = file.exists();   //파일이나 폴더가 존재한다면 true를 리턴
```

▷ exists() 메소드가 false를 리턴할 경우, 다음 메소드로 파일 또는 폴더를 생성할 수 있다.

리턴 타입	메소드	설명
boolean	createNewFile( )	새로운 파일을 생성
booelan	mkdir( )	새로운 디렉토리를 생성
boolean	mkdirs( )	경로상에 없는 모든 디렉토리를 생성

▷ exists() 메소드의 리턴값이 true라면 다음 메소드를 사용할 수 있다.

리턴 타입	메소드	설명
boolean	delete( )	파일 또는 디렉토리 삭제
boolean	canExecute( )	실행할 수 있는 파일인지 여부
boolean	canRead( )	읽을 수 있는 파일인지 여부
boolean	canWrite( )	수정 및 저장할 수 있는 파일인지 여부
String	getName( )	파일의 이름을 리턴
String	getParent( )	부모 디렉토리를 리턴
File	getParentFile( )	부모 디렉토리를 File 객체로 생성 후 리턴
String	getPath( )	전체 경로를 리턴
boolean	isDirectory( )	디렉토리인지 여부
boolean	isFile( )	파일인지 여부
boolean	isHidden( )	숨김 파일인지 여부
long	lastModified( )	마지막 수정 날짜 및 시간을 리턴
long	length( )	파일의 크기 리턴
String[]	list( )	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 String 배열로 리턴
String[]	list(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중에 FilenameFilter에 맞는 것만 String 배열로 리턴
File[]	listFiles( )	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 File 배열로 리턴
File[]	listFiles(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브 디렉토리 목록 중에 FilenameFilter에 맞는 것만 File 배열로 리턴

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
}
```

파일 또는 폴더의 정보를 얻기 위해 File 객체를 단독으로 사용할 수 있지만, 파일 입출력 스트림을 생성할 때 경로 정보를 제공할 목적으로 사용되기도 한다.

```
//첫 번째 방법
FileInputStream fis = new FileInputStream("C:/Temp/image.gif");

//두 번째 방법
File file = new File("C:/Temp/image.gif");
FileInputStream fis = new FileInputStream(file);
```

◎ Files 클래스

- Files 클래스는 정적 메소드로 구성되어 있기 때문에 File 클래스처럼 객체로 만들 필요가 없다.
- Files의 정적 메소드는 운영체제의 파일 시스템에게 파일 작업을 수행하도록 위임한다.

- Files 클래스에는 너무 많은 static 메소드가 있고, 사용법도 매우 복잡한 편입니다.
- 일단 import 하는 순간 모든 static 메소드가 다 메모리에 올라오기 때문에
  . 간단한 파일 시스템 작업은 그냥 java.io.File 클래스를 사용해서 하고,
  . 복잡한 파일 시스템 작업이 많이 필요할 때 java.nio.file.Files 클래스를 사용하는 것이 좋을 듯합니다.

기능	관련 메소드
복사	copy
생성	createDirectories, createDirectory, createFile, createLink, createSymbolicLink, createTempDirectory, createTempFile
이동	move
삭제	delete, deleteIfExists
존재, 검색, 비교	exists, notExists, find, mismatch
속성	getLastModifiedTime, getOwner, getPosixFilePermissions, isDirectory, isExecutable, isHidden, isReadable, isSymbolicLink, isWritable, size, setAttribute, setLastModifiedTime, setOwner, setPosixFilePermissions, probeContentType
디렉토리 탐색	list, newDirectoryStream, walk
데이터 입출력	newInputStream, newOutputStream, newBufferedReader, newBufferedWriter, readAllBytes, lines, readAllLines, readString, readSymbolicLink, write, writeString

boolean	isDirectory(Path p)	- 폴더인지 아닌지 검사
boolean	exists(Path p)	- 파일이 실제 존재하는지 검사
Path	createDirectory(Path p)	- 디렉토리 생성
Path	createFile(Path p)	- 파일 생성 (이미 해당 파일 있으면 예외 발생)

- 이 메소드들은 매개값으로 Path 객체를 받는다. Path 객체는 파일이나 디렉토리를 찾기 위한 경로 정보를 가지고 있는데, 정적 메소드인 get() 메소드로 다음과 같이 얻을 수 있다.

- get() 메소드의 매개값은 파일 경로인데, 전체 경로를 한꺼번에 지정해도 좋고, 상위 디렉토리와 하위 디렉토리를 나열해서 지정해도 좋다. 파일의 경로는 절대 경로 와 상대 경로를 모두 사용할 수 있다.

```
Path path = Paths.get(String first, String... more)
```

<예> "C:₩Temp₩dir₩file.txt" 경로를 이용해서 Path 객체를 얻는 방법을 보여준다.

```
Path path = Paths.get("C:/Temp/dir/file.txt");
Path path = Paths.get("C:/Temp/dir", "file.txt");
Path path = Paths.get("C:", "Temp", "dir", "file.txt");
```

<예> 만약 현재 디렉토리 위치가 "C:₩Temp"일 경우 "C:₩Temp₩dir₩file.txt"는 다음과 같이 지정이 가능하다.

```
Path path = Paths.get("dir/file.txt");
Path path = Paths.get("./dir/file.txt");
```

<예> 현재 위치가 "C:₩Temp₩dir1"이라면 "C:₩Temp₩dir2₩file.txt"는 다음과 같이 지정이 가능하다.

```
Path path = Paths.get("../dir2/file.txt");
```