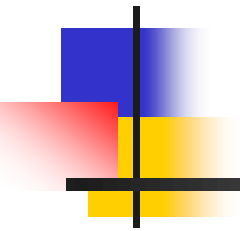

File 입출력

경복대학교
소프트웨어융합T과
배 희호

File and File System

- File
  - File은 논리적인 저장 단위로, 관련된 정보 자료들의 집합에 이름을 붙인 것
  - Computer System의 편리한 사용을 위해 정보 저장의 일괄된 논리적 관점을 제공
  - 일반적으로 레코드(Record) 혹은 블록(Block) 단위로 비휘발성 보조 기억 장치에 저장
  - 특히, Keyboard(입력 자원, in)도 File이고, Monitor(출력 대상, out)도 File이고, 각 Printer(출력 대상) 역시 File 임

File and File System

- File 속성(File attribute) 또는 File의 메타 데이터(metadata)
  - File을 관리하기 위한 각종 정보들
  - File 자체의 내용은 아님
  - File 이름, 유형, 저장된 위치, File 크기, 접근 권한, 소유자, 시간(생성/변경/사용) 등 File에 대한 전반적인 정보를 말함

File and File System

- File의 종류
  - 일반(Ordinary) File
    - Data를 가지고 있으면서 Disk에 저장됨 (혹은 Tape에 저장하는 경우도 있음)
  - Directory
    - Disk에 저장되어 있으면서 다른 File을 조직하고 Access하는데 필요한 정보를 가지고 있음
  - 특수 File(special) 또는 Device File
    - 물리적인 장치에 대한 내부적인 표현 (예: /dev/rsd0)

File and File System

- File의 구분
  - 문서(Text) File
    - 단지 ASCII 문자(Keyboard에서 입력할 때 생성되는 Data)만을 가지고 있는 일반 File
    - 문서 File은 문서, Program, Shell Script 등을 작성하는데 사용됨
  - 이진(Binary) File
    - 문자가 아닌 Data가 들어 있는 일반 File
    - Binary File은 일반적으로 Program에 의해서 처리될 때만 의미가 있는 Data를 포함 (실행 File 자체도 Binary File에 해당함)

File and File System

- 문서 File과 Binary File의 기술적 차이
  - Computer에서 Data는 Bit의 연속으로 저장
    - Bit는 하나의 2진수(0 or 1)로 구성
    - Byte : 8개 Bit의 문자열(8-Bit String)
  - 문서 File
    - Byte에 저장된 내용이 문자(ASCII Code, 한글 코드 등)로 해석됨
  - Binary File
    - Byte에 저장된 내용이 특수한 목적에 따라 다르게 해석됨
    - 예) 실행 File의 경우 운영체제에 의해 의미 있게 해석됨

File and File System

- File System
  - File System은 운영 체제와 모든 Data, Program의 저장과 접근을 위한 기법을 제공
  - System 내의 모든 File에 관한 정보를 제공하는 계층적 Directory 구조이고, File 및 File의 MetaData, Directory 정보 등을 관리
- File System 종류
  - FAT(File Allocation Table)
    - 단일 File로 4 GB까지만 저장됨
    - USB에 File Format이 이거면 4 GB 이상의 File을 넣을 수 없음
  - NTFS(New Technology File System)
    - FAT File System을 개선하여 만든 새로운 File System
  - exFAT(Extended File Allocation Table)

File and File System

- Partition
  - Partition은 연속된 저장 공간을 하나 이상의 연속되고 독립적인 영역으로 나누어 사용할 수 있도록 정의한 규약
  - 하나의 물리적 Disk 안에 여러 Partition을 두는 게 일반적이지만, 여러 물리적 Disk를 하나의 Partition으로 구성하기도 함

File and File System

- Directory
  - Directory는 File의 MetaData 중 일부를 보관하고 있는 일종의 특별한 File
  - 해당 Directory에 속한 File 이름과 속성들을 포함
  - File을 계층적으로 조직화하기 위해 Directory를 사용 (우리가 항시 작업하는 Folder 개념과 동일함)
  - Directory 그 자체도 File이므로 한 Directory는 다른 Directory를 포함함으로써 계층을 이룸
  - 부모 Directory는 다른 Directory를 가지고 있는 Directory
  - 다른 Directory 안에 있는 Directory를 Sub Directory라고 함

File and File System

- Directory 구분 기호
  - Windows: ₩
  - Windows 이외: /
  - 직접 지정하면 운영체제 별로 따로 설정 – File.seperator를 이용하면 현재 운영체제의 구분 기호를 사용할 수 있음
- Directory
  - 다음과 같은 기능들을 제공
    - 파일 찾기(Search)
    - 파일 생성(Create)
    - 파일 삭제(Delete)
    - 디렉터리 나열(List)
    - 파일 재명명(Rename)
    - 파일 시스템 순회(Traverse)

Path

- File System에 저장되어 있는 File은 최상위 Directory인 root(/)'로 부터 찾아 내려갈 수 있는 경로(Path)를 갖음
- File의 Path는 절대 경로 일 수 있고, 상대 경로 일 수도 있음
- File의 Path는 문자열로 저장되는데, JAVA NIO에서는 Paths 클래스와 Path 인터페이스 제공

- Path 인터페이스를 얻기 위해서는 Paths 클래스의 정적 메소드인 get()을 사용
  - 이 때 File의 절대 경로를 이용할 수도 있고, 상대 경로로 이용할 수도 있음
  - 상대 경로로 이용할 때에는 Paths.get(basePath, relativePath) 형태로 호출

File and File System

- Path
  - 절대 경로 : Root로부터의 위치
    - Windows -> Root Driver:₩Directory 경로₩File 이름
    - Web -> Protocol://자원 경로
    - 그 이외의 경우 -> /루트 디렉토리/디렉토리 경로/파일 이름
    - Windows 경우는 Directory 기호가 ₩, 그 이외의 경우는 /
  - 상대 경로 : 현재 위치로부터의 경로
    - ./: 현재 Directory
    - ../: 상위 Directory

IO(Input/Output)

- I/O의 대상
  - Keyboard와 Monitor
  - HDD에 저장되어 있는 File 입출력
  - USB와 같은 External Memory 장치의 File 입출력
  - Network로 연결되어 있는 Computer
  - Sound Card, Audio Card와 같은 Multimedia 장치
  - Printer, Facsimile 같은 출력장치
  - Mouse와 같은 입력장치

- JAVA는 입출력 대상에 상관없이 입출력 방식이 동일하도록 별도의 I/O 모델을 정의

IO(Input/Output)

- I/O는 Source로부터 Data를 읽어서 목적지에 Data를 쓰는 것
- Data는 입력 Source로부터 읽고 출력 목적지로 쓰여 짐
  - 예) 당신이 Keyboard가 표준 입력으로 동작하면 Data를 읽어 당신의 Program에 쓰여지는 것
  - 예) 표준출력으로 Text를 출력하기 위해 System.out.println()를 사용함. 우리는 아무 지식없이 I/O를 수행한 것
- JDK에 java.io 및 java.nio Package에는 I/O를 처리하는 JAVA 클래스들이 포함됨
  - java.io Package는 I/O를 수행하기 위한 굉장히 많은 클래스들을 포함
- JAVA I/O를 배우는 것은 약간 복잡(클래스가 너무 많아)

File 클래스

- JAVA File 입출력의 기본이 되는 File과 Directory를 다루기 위해서 JAVA는 File 클래스를 제공함
- File의 경로명을 다루는 클래스
  - java.io.File
  - File과 Directory 경로명의 추상적 표현
- File 이름 변경, 삭제, Directory 생성, 크기 등 File 관리
  - File 객체는 File 읽고 쓰기 기능은 없음
- File 클래스 객체는 File이나 Directory를 생성, 삭제, 변경하는 것 외에 입출력 클래스와 같이 사용하여 File에 Data를 입출력 할 수 있음

File 클래스

생성자

생성자	설명
File(File parent, String Child)	parent 객체 폴더의 child라는 File에 대한 File 객체를 생성
File(String pathname)	pathname에 해당되는 File의 File 객체를 생성
File(String parent, String, child)	parent 폴더 경로의 child라는 File에 대한 File 객체를 생성
File(URI uri)	URI uri 경로에 대한 File의 File 객체를 생성

File 클래스

- File 객체 생성
  - File f1 = new File("/");
    파일의 경로만 가진 객체
  - File f2 = new File("/", "autoexec.bat");
    경로명과 파일 이름을 가진 객체
  - File f3 = new File(f1, "autoexec.bat");

경로명을 가진 File 객체와 파일 이름을 가진 객체

File 클래스

- File 객체 생성

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

- 예) “C:₩Java₩Stream₩Test.java”를 나타내는 File객체를

```
File f1 = new File(“C:₩₩Java₩Stream₩₩Test.java”);
File f2 = new File(“C:₩₩Java₩₩Stream”, “Test.java”);
File dir = new File(“C:₩₩Java₩₩Stream”); // 디렉토리
File f3 = new File(dir, “Test.java”); // 생성된 디렉토리 이용
```

  - 생성된 File객체 f1, f2, f3는 모두 동일한 File을 나타냄

File 클래스

- 클래스 상수
  - 경로(Path)를 구분하기 위해 운영 체제마다 다른 경로 구분자 사용
  - File 클래스 인스턴스 생성시에 지정

상수	설명
public static final char separatorChar	윈도우의 경우에는 '₩n'이고, 유닉스의 경우에는 '/n'
public static final String separator	편의를 위해서 문자열 형태로 상수 제공

File 클래스

- 제공 메소드 종류
  - 파일 정보를 가져오는 메소드
  - 파일 정보를 수정하는 메소드
  - 파일을 생성/삭제하는 메소드
  - 디렉토리 관리에도 사용됨 (디렉토리는 특수한 형태의 파일이라고 할 수 있음)

File 클래스

- File 정보를 가져오는 메소드

메소드	설명
File getAbsoluteFile()	파일의 절대 경로를 반환
String getAbsolutePath()	파일의 절대 경로를 문자열로 반환
File getCanonicalFile()	파일의 Canonical 경로를 반환
String getCanonicalPath()	파일의 Canonical 경로를 문자열로 반환
String getName()	파일이나 폴더의 이름을 반환
String getParent()	부모 경로에 대한 경로명을 문자열로 반환
File getParentFile()	부모 폴더를 File의 형태로 반환
String getPath()	파일의 경로를 문자열의 형태로 반환
long getTotalSpace()	하드디스크의 총 용량을 반환

File 클래스

- File 정보를 가져오는 메소드

메소드	설명
long getUsableSpace()	하드디스크의 사용 가능한 용량을 반환
long getFreeSpace()	하드디스크의 남은 공간을 반환
int hashCode()	hash code를 반환
long lastModified()	해당 경로 파일의 최종 수정 일자를 반환
long length()	해당 경로 파일의 길이를 반환
Path toPath()	java.nio.file.Path 객체로 반환
URI toURI()	URI 형태로 파일 경로를 반환
File[] listRoots()	하드디스크의 루트 경로를 반환
String[] list()	경로의 파일들과 폴더를 문자열 배열로 반환

File 클래스

- File 정보를 가져오는 메소드

메소드	설명
String[] list(FilenameFilter filter)	filter에 만족되는 파일들과 폴더 이름을 문자열 배열로 반환
File[] listFiles()	해당 경로의 파일들과 폴더의 파일을 배열로 반환
File[] listFiles(FileFilter filter)	filter에 만족되는 파일들과 폴더를 File 배열로 반환
File[] listFiles (FilenameFilter filter)	filter에 만족되는 파일들과 폴더를 File 배열로 반환

File 클래스

- File 생성 수정 삭제 메소드

메소드	설명
boolean createNewFile()	주어진 이름의 파일이 없으면 새로 생성
static File createTempFile (String prefix, String suffix)	default temporary-file 디렉토리에 파일 이름에 prefix와 suffix를 붙여 임시파일을 생성
static File createTempFile (String prefix, String suffix, File directory)	새로운 임시파일을 파일 이름에 prefix와 suffix를 붙여 directory 폴더에 생성
boolean delete()	파일이나 폴더를 삭제 단, 폴더가 비어 있지 않으면 삭제할 수 없음
void deleteOnExit()	자바 가상머신이 끝날 때 파일을 삭제
boolean mkdir()	해당 경로에 폴더를 만듦

File 클래스

- File 생성 수정 삭제 메소드

메소드	설명
boolean mkdirs()	존재하지 않는 부모 폴더까지 포함하여 해당 경로에 폴더를 만듦
boolean renameTo (File dest)	dest로 File 이름을 변경

File 클래스

- File 체크 메소드

메소드	설명
boolean exists()	파일의 존재 여부를 반환
boolean isAbsolute()	해당 경로가 절대경로인지 여부를 반환
boolean isDirectory()	해당 경로가 폴더인지 여부를 반환
boolean isFile()	해당 경로가 일반 file 인지 여부를 반환
boolean isHidden()	해당 경로가 숨김 file 인지 여부를 반환

File 클래스

- File 권한 메소드

메소드	설명
boolean canExecute()	파일을 실행할 수 있는지 여부를 반환
boolean canRead()	파일을 읽을 수 있는지 여부를 반환
boolean canWrite()	파일을 쓸 수 있는지 여부를 반환
boolean setExecutable (boolean executable)	파일 소유자의 실행 권한을 설정
boolean setExecutable (boolean executable, boolean ownerOnly)	파일의 실행 권한을 소유자 또는 모두에 대해 설정
boolean setReadable (boolean readable)	파일의 소유자의 읽기 권한을 설정
boolean setReadable (boolean readable, boolean ownerOnly)	파일의 읽기 권한을 소유자 또는 모두에 대해 설정

File 클래스

- File 권한 메소드

메소드	설명
boolean setReadOnly()	파일을 읽기 전용으로 변경
boolean setWritable (boolean writable)	파일의 소유자의 쓰기 권한을 설정
boolean setWritable (boolean writable boolean ownerOnly)	파일의 쓰기 권한을 소유자 또는 모두에 대해 설정

File 클래스

- 경로와 관련된 File의 멤버변수

멤버변수	설명
static String pathSeparator	OS에서 사용하는 경로 구분자 윈도우";", 유닉스":"
static char pathSeparatorChar	OS에서 사용하는 경로 구분자 윈도우';', 유닉스':'
static String separator	OS에서 사용하는 이름 구분자 윈도우"₩", 유닉스"/"
static char separatorChar	OS에서 사용하는 이름 구분자 윈도우'₩', 유닉스'/

File 클래스

- File 클래스의 주된 기능
  - 지정된 File 또는 Directory에 관한 정보를 알려주는 기능
  - File 클래스를 통해 File의 경로(path)에 관한 정보를 다양한 형식으로 알아낼 수 있음

- File의 경로 정보 메소드
  - getPath() 메소드는 Instance를 만들 때 생성자로 입력된 경로명을 반환 (abstract pathname이라고 함)
  - getAbsolutePath() 메소드는 생성자로 입력된 경로명을 절대 경로로 바꾸어서 반환
  - 생성자로 입력된 경로명이 상대 경로일 경우에는 Program이 실행된 Directory를 기준으로 하여 절대 경로로 바꾸는 작업

File 클래스

- File의 경로 정보 메소드
  - 얻어진 절대 경로를 System에서 사용하는 형식의 getCanonicalPath() 메소드는 좀더 정규화된 형태의 절대 경로를 반환하는 메소드
  - getCanonicalPath() 절대 경로로 바꾸어 줌
    - 예) UNIX System의 경우 심볼릭 링크로 연결된 File을 실제의 경로로 바꾸는 작업
  - Windows System의 경우Dirve 문자를 대문자로 바꾸어 주고 실제 존재하는 File의 경우 대소문자까지 원래의 File명과 같이 맞추어 줌(Windows System은 대소문자를 구별하기는 하지만 대소문자를 틀리게 입력해도 같은 File/폴더로 간주)
  - getAbsolutePath() 보다 정확한 경로명을 반환하는 getCanonicalPath()를 사용하기를 추천

File 클래스

- File의 경로 정보 메소드
  - 다만, getCanonicalPath() 메소드를 사용할 때에는 한가지 주의할 점
  - 경로를 알아내는 다른 명령어와는 달리 이 메소드는 System에 File System 정보를 요청하기 때문에 IOException을 발생시킬 수 있도록 설계되었다는 점
  - 따라서 getCanonicalPath()를 사용할 때에는 반드시 IOException을 처리(try-catch)하도록 해야 함

File 클래스

- File이나 Directory 생성
  - 우선 생성할 File이나 Directory의 경로명으로 File 클래스 Instance를 생성

```
File file = new File(생성할 파일 또는 디렉토리의 경로);
```

  - File을 만들기 위해서는 createNewFile(), Directory를 만들기 위해서는 mkdir()이나 mkdirs() 메소드를 사용
  - 이들 메소드는 이미 File 객체가 경로명을 가지고 있기 때문에 아무런 매개 변수를 전달하지 않아도 됨

File 클래스

- mkdir() 메소드와 mkdirs() 메소드의 차이
  - 해당 Directory의 상위 Directory 구조가 없을 경우, 이를 만들어 주는지의 차이임
  - mkdirs() 메소드는 주어진 경로명에 포함된 모든 Directory 구조를 만들어 줌

```
File dir = new File("language/java/scjp");
dir.mkdir();
dir.mkdirs();
```

  - 위의 예에서 아직 language나 java Directory가 아직 없는 경우 mkdir() 메소드는 Directory를 만들지 못했음을 뜻하는 false를 반환
  - mkdirs() 메소드는 language나 java Directory리까지 전부 만들어 줌

File 클래스

- createTempFile()이라는 메소드
  - createNewFile() 메소드와 성격은 약간 다르지만, File을 만드는 기능은 동일한 메소드
  - static 메소드 임 (따라서, 아래 예제와 같이 사용 가능)
  - Program에서 사용하기 위한 임시 File을 생성

```
File file = File.createTempFile(파일명, 확장자);
File file = File.createTempFile(파일명, 확장자, 디렉토리명);
```

File 클래스

- File 객체(인스턴스)의 생성

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

현재의 디렉토리에 있는 test.txt 파일에 대한 "File 객체"를 생성

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

C 드라이브의 work 디렉토리 아래에 있는 test.txt 파일에 대한 File 객체를 생성

File 클래스

- File/Directory 정보 가져오기
  - File/Directory 정보를 가져오는 메소드를 호출

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

파일 또는 디렉토리가 있으면 true,
없으면 false를 반환

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

파일이면 true, 아니면 false를 반환

File 클래스

- File/Directory 정보 가져오기
  - File/Directory 정보를 가져오는 메소드를 호출

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

디렉토리면 true, 아니면 false를 반환

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

서브 디렉토리와 파일들의 목록을 반환

File 클래스

- File/Directory 정보 가져오기
  - File/Directory 정보를 가져오는 메소드를 호출

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

File 클래스

- File의 존재 유무 확인

```
File file = new File(“test.txt");
if(!file.exists()) {
      System.out.println(file.getName() +
                              " 파일이 존재하지 않습니다!");
      System.exit(0);
}
```

File 클래스

- File / Directory 판정

```
File file = new File("c:₩windows");
if(file.isDirectory())
    System.out.println(file.getName() + " is a directory.");
if(file.isFile())
    System.out.println(file.getName() + " is a file.");
```

File 클래스

- File/Directory 생성/삭제하기
  - File을 생성하고 삭제하는 메소드

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

현재 디렉토리에 poem.txt라는 이름의 파일을 생성

```
File file2 = new File("c:₩₩doc₩₩화의록.hwp");
file2.delete();
```

C:₩doc₩회의록.hwp 파일을 삭제

File 클래스

- File/Directory 생성/삭제하기
  - Directory를 생성하고 삭제하는 메소드

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

C 드라이브의 루트 디렉토리에 "올빼미"라는 디렉토리를 생성

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

현재 디렉토리에 있는 "두루미"라는 디렉토리를 삭제

File 클래스

- 임시 File 생성하기
  - 임시 File을 생성하는 메소드

```
File tempFile = file.createTempFile("tmp", ".txt", tmpDir);
```

createTempFile() 메소드를 이용하여 임시파일을 생성 하면, 말 그대로 임시파일 이므로 사용이 끝나면 삭제처리 또한 잊지말고 해주어야 함 삭제 처리를 하지 않는다면 수많은 임시파일만 남게 될 것
파일 삭제를 위해서 **deleteOnExit()**를 사용하며, 이 메소드의 특징은 삭제 처리한다고 바로 파일을 지우는 것이 아닌 **JVM**이 종료 될 때 자동으로 지정된 파일을 삭제함

tmp로 시작하고

.txt로 끝나는

임시파일을 이 디렉토리에 생성

File 클래스

- 임시 File 생성하기
  - FileWriter 클래스에는 File 타입의 파라미터를 받는 생성자 사용

```
FileWriter writer = new FileWriter(tmpFile);
```

생성자를 이용하여 파일을 열 수 있음