파일(FILE)

경복대학교
소프트웨어융합과
배희호 교수
010-2369-4112
031-570-9600
hhbae@kbu.ac.kr

File

- Computer에서 File은 매우 중요한 입출력 대상
- Computer의 RAM(Random Access Memory)은 휘발성으로 Computer의 전원이 꺼지면 Memory 상의 Data는 모두 삭제됨
- Computer를 이용하다 보면 게임의 세이브 Data, 보고서 작성 Data, 채팅이나 문자 Message 등 Computer를 종료하더라도 삭제되면 안 되는 Data들이 있음
- 이처럼 'File'은 Computer의 Memory 한계를 벗어나 Data를 저장하고 공유하는 중요한 수단
- File은 Computer Disk에 Text나 Binary 형태로 저장하기 위해 고안됨
  - Text File은 Program Source나 메모장에서 작성한 단순 정보를 기록하는 것이 목적
  - Binary File은 이진 형태로 저장되며 Program 실행 File이나 Program에서 저장하는 Data File

File

- Directory
  - 대용량 File을 관리하기 위해 고안됨
  - ‘Folder'라고도 지칭
  - Disk System의 최 상위를 루트(ROOT)라 하며, File은 Root를 비롯한 하위 Directory에 들어 있음

- Path
  - 경로는 Disk System에서 File의 위치를 관리하는 체계
  - File을 처리 하기 위해서는 File의 위치 정보가 필요하고, File과 Directort는 위치를 표현하기 위해 사용됨
  - UNIX 계열은 '/‘, Windows 계열은 '₩'를 구분자로 사용

File 클래스

- File을 제어하기 위한 클래스
- File 클래스 자체에서는 입출력 기능을 제공하지 않음
  - File 입출력은 Stream 클래스를 사용해야 함
- File 클래스로부터 생성된 객체는 변경할 수 없음
  (즉, 경로를 변경하는 것이 불가능)

- 생성자

생성자	설명
File(File parent, String child)	parent 폴더에 child라는 파일에 대한 File 객체 생성
File(String parent, String child)	parent 폴더에 child라는 파일에 대한 File 객체 생성
File(String parent)	parent에 해당하는 파일 File 객체를 생성

File 클래스

- 메소드

메소드	설명
boolean delete()	파일이나 폴더를 삭제 (폴더가 비어있어야만 삭제가 가능)
boolean exists()	파일의 존재 유무를 반환
String getAbsolutePath()	파일의 절대경로를 반환
String getName()	파일이나 폴더의 이름을 반환
boolean isDirectory()	폴더인지 여부를 반환

HDD 정보 출력(예제 1)

```
public static void main(String[] args) {
    FileSystemView system = FileSystemView.getFileSystemView();
    File[] drives = File.listRoots();
    if (drives != null && drives.length > 0) {
        for (int i = 0; i < drives.length; i++) {
            System.out.println("Drive Letter : " + drives[i]);
            System.out.println("₩tType : " +
                                    system.getSystemTypeDescription(drives[i]));
            System.out.printf("₩tTotal space : %,d Bytes₩n",
                                                    drives[i].getTotalSpace());
            System.out.printf("₩tFree space : %,d Bytes₩n“, drives[i].getFreeSpace());
            System.out.println();
        }
    }
}
```

HDD 정보 출력(예제 1)

- Drive 유형(Local Disk 또는 CD Drive 또는 Floppy Disk)을 찾기 위해 FileSystemView의 getFileSystemView() 메소드를 사용
- HDD(하드 디스크)의 Root Driver를 배열로 반환

```
File[] roots = File.listRoots()
```

- getUsableSpace()와 getFreeSpace()는 결과가 같음
- 사용한 디스크 용량 = 총 용량 – 사용 가능한 용량

HDD 정보 출력(예제 2)

```
public static void main(String[] args) {
    String dir = "c:/";
    File file = new File(dir);
    System.out.println(" Directory Name : " + dir);
    System.out.println(" last Modified : " + file.lastModified());
    System.out.println(" Can Read ? : " + file.canRead());
    System.out.println(" Can Write ? : " + file.canWrite());
    System.out.println(" is File ? : " + file.isFile());
    System.out.println(" is Directory ? : " + file.isDirectory());
    System.out.println("- Now we check c:/'s sub directory ");
    String[] list = file.list();
    for (int i = 0; i < list.length; i++) {
        File newfile = new File(dir, list[i]);
        if (newfile.isDirectory()) {
            System.out.println(list[i] + " is a Dirctory ");
        } else {
            System.out.print(list[i] + " is a File ");
            System.out.println("& size is " + newfile.length());
        }
    }
}
```

File인지 Directory인지(예제 3)

```
public static void main(String[] args) {
        String path = "c:₩₩windows₩₩system.ini";
    //   String path = "c:₩₩windows";
        File file = new File(path);
        String result;
        if (file.isFile())      // 파일 타입이면
            result = "파일";
        else                    // 디렉터리 타입이면
            result = "디렉터리";
        System.out.println(file.getPath() + "은 " + result + "입니다.");
}
```

File인지 Directory인지(예제 3-1)

```
public static void main(String[] args) {
    String path = "c:₩₩windows₩₩system.ini";
   // String path = "c:₩₩windows";
    File file = new File(path);
    String result = " ";
    if (file.isDirectory())      // 디렉토리 타입이면
        result = "디렉토리";
    else if (file.exists())     // 파일 타입이면
        result = "파일";

    System.out.println(file.getPath() + "은 " + result + "입니다.");
}
```

File인지 Directory인지(예제 3)

- File이 존재하는지 확인하는 방법
  - File.exists()
    - File.exists()는 File 또는 Folder가 존재하는지 반환
    - 만약 Folder가 아닌, File이 존재하는지 확인하려면 File.isDirectory()도 함께 체크해야 함
  - File.isFile()
    - File.isFile()는 File이 존재하는 경우 true를 반환
    - File의 주소가 Folder인 경우는 false를 반환
    - 이 API를 사용하면 File 존재 유무를 체크할 때 File.isDirectory()를 호출하지 않아도 됨
  - Path.toFile()

File인지 Directory인지(예제 3)

- File이 존재하는지 확인하는 방법
  - Files.exists(path)
    - Parameter로 전달된 Path 객체가 가리키는 Directory 또는 File이 존재하는지 확인
  - Files.isDirectory(path)
  - Files.isRegularFile(path)
    - Path 객체가 가리키는 대상이 Directory인지 File인지 체크

Directory 정보(예제 4)

```
public class Main {
    static int totalFiles = 0;
    static int totalDirs = 0;

    public static void main(String[] args) {
        File dir = new File("./");

        if (!dir.exists() || !dir.isDirectory()) {
            System.out.println("유효하지 않은 디렉토리입니다.");
            System.exit(0);
        } else {
            printFileList(dir);
            System.out.println();
            System.out.println("총 " + totalFiles + "개의 파일");
            System.out.println("총 " + totalDirs + "개의 디렉토리");
        }
    }
```

Directory 정보(예제 4)

```
public static void printFileList(File dir) {
    System.out.println(dir.getAbsolutePath() + " 디렉토리");
    File[] files = dir.listFiles();
    List<String> subDir = new ArrayList<>();
    for (int i = 0; i < files.length; i++) {
        String filename = files[i].getName();
        if (files[i].isDirectory()) {
            filename = "[" + filename + "]";
            subDir.add(i + "");
        }
        System.out.println(filename);
    }
```

Directory 정보(예제 4)

```
        int dirNum = subDir.size();
        int fileNum = files.length - dirNum;
        totalFiles += fileNum;
        totalDirs += dirNum;
        System.out.println(fileNum + "개의 파일, " + dirNum + "개의 디렉토리");
        System.out.println();
        for (int i = 0; i < subDir.size(); i++) {
            int index = Integer.parseInt(subDir.get(i));
            printFileList(files[index]);
        }
    }
}
```

Directory 정보(예제 5)

- 현재 디렉토리의 서브 디렉토리와 파일 목록을 출력하는 프로그램을 작성하여라

```
public static void main(String[] args) {
    File file = new File(".");
    File[] list = file.listFiles();
    for (int index = 0; index < list.length; index++) {
        String name = list[index].getName();
        if (list[index].isFile())
            System.out.printf("%-25s %7d ", name, list[index].length());
        else
            System.out.printf("%-25s   <DIR> ", name);
        long time = list[index].lastModified();
        GregorianCalendar calendar = new GregorianCalendar();
        calendar.setTimeInMillis(time);
        System.out.printf("%1$tF %1$tT %n", calendar);
    }
}
```

File 정보 출력(예제 6)

```
public static void main(String[] args) {
    String path = "c:₩₩windows₩₩system.ini";
    String str = "";

    File file = new File(path);
    if (file.exists()) {
        str += "파일명: " + file.getName() + "₩n" + "파일의 크기 : "
                + file.length() + "₩n" + "마지막 수정일 : " + file.lastModified()
                + "₩n" + "부모 디렉토리 : " + file.getParent();
    } else {
        str = "해당파일이 존재하지 않습니다.";
    }
    System.out.println(str);
    Date date = new Date(file.lastModified());
    System.out.println("마지막 수정일:" + date);
}
```

File 정보 출력(예제 6-1)

```
public static void main(String[] args) {
    File dir = new File("./");
    String[] strs = dir.list();
    for (int i = 0; i < strs.length; i++) {
        System.out.println(strs[i]);
    }
}
```

특정 폴더의 File 목록 출력(예제 7)

```
public static void main(String[] args) {
        File file = new File("..");
        File[] list = file.listFiles();
        for (int index = 0; index < list.length; index++) {
            String name = list[index].getName();
            if (list[index].isFile())
                System.out.printf("%-25s %7d ", name, list[index].length());
            else
                System.out.printf("%-25s   <DIR> ", name);
            long time = list[index].lastModified();
            GregorianCalendar calendar = new GregorianCalendar();
            calendar.setTimeInMillis(time);
            System.out.printf("%1$tF %1$tT %n", calendar);
        }
}
```

특정 폴더의 File 목록 출력(예제 7)

- new File(".")는 현재 Project Folder를 나타냄
- new File("..")는 현재 Folder의 상위 Folder를 나타냄
- file.list()로 해당 Folder의 File들을 문자열 배열로 반환
  (자식 Folder 안의 File은 가져오지 않음)

Directory 만들기(예제 8)

- C://test/test2로 test2 디렉토리를 만들어보자

```
public static void main(String[] args) {
    File file = new File("C://test//test2");
    boolean result = file.mkdir();
    if(result)
        System.out.println("디렉토리 만들기 성공");
    else
        System.out.println("디렉토리 만들기 실패");
}
```

  - mkdir() 메소드는 Directory를 만들고, 결과로 성공여부를 boolean을 반환
  - C://test 폴더가 존재하지 않으면 실패하게 됨
  - 이럴 땐 mkdirs() 메소드를 사용

Directory 만들기(예제 8-1)

- C://test/test2로 test2 디렉토리를 만들어보자

```
public static void main(String[] args) {
    File file = new File("C://test/test2");
    boolean result = file.mkdirs();
    if(result)
        System.out.println("디렉토리 만들기 성공");
    else
        System.out.println("디렉토리 만들기 실패");
}
```

  - mkdirs() 메소드는 상위 폴더들이 없으면 상위폴더들까지 만들고, 결과로 성공여부를 boolean을 반환

File 존재 확인(예제 9)

- exists() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test//a.txt");
    if(file.exists()) {
        System.out.println("파일 존재");
    } else {
        System.out.println("파일 없음");
    }
}
```

  - 파일이 존재하는지 여부를 알 수 있음
  - 반환 결과가 boolean으로 파일이 존재하면 참, 없으면 거짓을 반환

File 존재 확인(예제 9-1)

- isFile() 메소드

```
public static void main(String[] args) {
    File file = new File(".//data//a.txt");
    if(file.isFile()) {
        String name = file.getName();
        System.out.println("File Name : " + name);
} else {
        System.out.println("파일이 아님");
    }
}
```

  - 파일인지를 검사하는 함수
  - 파일이 존재하지 않거나 디렉토리이면 false를 반환, 파일이면 true 반환

File 이름 확인(예제 9 -1)

- getName() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test//a.txt");
    if(file.isFile()) {
        String name = file.getName();
        System.out.println("File Name : " + name);
    } else {
        System.out.println("파일이 아님");
    }
}
```

  - 파일의 이름을 반환하는 함수
  - 앞에 파일 경로를 제외하고 파일 이름만 String 타입으로 반환

File Size 알아보기(예제 10)

- length() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test/a.txt");
    if(file.exists()) {
        System.out.println("file Size : " +
                                        String.format("%,d Bytes", file.length()));
    } else {
        System.out.println("파일 없음");
    }
}
```

  - file의 Size를 측정하는 함수
  - Byte 사이즈 크기를 변수 타입 long으로 반환

숨김 File 확인 방법(예제 11)

- isHidden() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test/a.txt");
    if(file.isHidden()) {
        System.out.println("숨겨진 파일");
    } else {
        System.out.println("숨겨진 파일이 아님");
    }
}
```

  - 숨김 파일인지를 검사하는 함수
  - 파일이나 디렉토리가 숨김으로 되어있으면 true를 반환하고 아니면 false를 반환

File 경로와 이름 확인(예제 12)

- getPath() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test/a.txt");
    if (file.exists()) {
        String path = file.getPath();
        System.out.println("File Path : " + path);
    } else {
        System.out.println("파일 없음");
    }
}
```

  - 파일의 전체 경로와 이름을 반환하는 함수

Directory 확인 방법(예제 13)

- isDirectory() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test//");
    if(file.isDirectory()) {
        System.out.println("디랙토리");
    } else {
        System.out.println("디랙토리 아님");
    }
}
```

  - 디렉토리인지를 검사하는 함수
  - 디렉토리이면 true를 반환하고 파일이거나 존재하지 않으면 false를 반환

File 삭제 방법(예제 14)

- delete() 메소드

```
public static void main(String[] args) {
    File file = new File("C://test/a.txt");
    if(file.exists()){
        if(file.delete()){
            System.out.println("파일삭제 성공");
        }else{
            System.out.println("파일삭제 실패");
        }
    }else{
        System.out.println("파일이 존재하지 않습니다.");
    }
}
```

  - 파일을 삭제하는 함수
  - 파일이 사용중일경우 파일 삭제가 정상적으로 이루어지지 않을 수 있음

```
public static void main(String[] args) {
    File file = new File("C://dir");
    if (file.exists()){ //파일존재여부확인
        if(file.isDirectory()){ //파일이 디렉토리인지 확인
            File[] files = file.listFiles();
            for( int i = 0; i < files.length; i++){
                if(files[i].delete()){
                    System.out.println(files[i].getName()+" 삭제 성공");
                }else{
                    System.out.println(files[i].getName()+" 삭제 실패");
                }
            }
        } else {
            if (file.delete()) {
                System.out.println("파일삭제 성공");
            } else {
                System.out.println("파일삭제 실패");
            }
        }
    }else{
        System.out.println("파일이 존재하지 않습니다.");
    }
}
```

File 이름 변경(예제 15)

- renameTo() 메소드
  - 변경 전 파일(oldfile)을 새로운 경로 및 이름을 가진 파일(newfile)로 변경하는 함수
  - renameTo() 함수는 boolean값을 번환 해주므로 성공여부를 확인할 수 있음

```
public static void main(String[] args) {
    File oldfile = new File("c://test//test.txt");
    File newfile = new File("c://test//test.bak");
    if (newfile.exists()) {
        newfile.delete();
    }
    if(oldfile.renameTo(newfile)){
        System.out.println("파일 이름 변경 성공");
    }else{
        System.out.println("파일 이름 변경 실패");
    }
}
```

특정 File 생성(예제 16)

- JAVA에서 File을 생성하는 3가지 방법
  - File.createNewFile() 메소드
    - 이 메소드는 File을 성공적으로 생성하면 true를, 기존에 이미 동일한 이름의 File이 있을 경우에는 false
  - Files.createFile() 메소드
  - FileOutputStream 클래스

특정 File 생성(예제 16)

```
public static void main(String[] args) throws IOException {
        String filename = "..₩₩data₩₩filesample.txt";

        File file = new File(filename);
        if (file.exists()) {
            System.out.printf("%s 파일은 이미 존재합니다.₩n", filename);
        } else if (file.createNewFile()) {
            System.out.println("파일 생성 성공₩n");
        } else {
            System.out.println("파일 생성 오류₩n");
        }
}
```

Directory 내용 가져오기 예제

```
public static void main(String[] args) {
    File file = new File("c:₩₩windows₩₩system.ini");
    File dir1 = new File("c:₩₩test₩₩java_sample");
    File dir2 = new File("c:₩₩test");
    String res;
    if(file.isFile())
        res = "파일";
    else
        res = "디렉터리";
    System.out.println(file.getPath() + "은 " + res + "입니다.");
    if (!dir1.exists()) {
        if (!dir1.mkdir())
            System.out.println("디렉터리 생성 실패");
    }
    if(dir1.isFile())
        res = "파일";
    else
        res = "디렉터리";
    System.out.println(dir1.getPath() + "은 " + res + "입니다.");
    directory(dir2);
    dir1.renameTo(new File("c:₩₩test₩₩javasample"));
    directory(dir2);
}
```

Directory 내용 가져오기 예제

```
    private static void directory(File file) {
        String[] filenames = file.list();
        for (String s : filenames) {
            File list = new File(file, s);
            long time = list.lastModified();
            System.out.printf("%20s", s);
            System.out.printf(" 파일 크기: %10d", list.length()); // 파일 크기
            System.out.printf(" 수정한 시간: %tb %td %ta %tT₩n", time, time, time, time);
        }
    }
}
```

Directory 내용 가져오기 예제

- C drive에 test 폴더를 생성
- filesample.txt 파일을 만들어 넣음

- 폴더 생성
  - 폴더 생성 전 exists() 메소드를 통해 현재 생성하려는 폴더가 있는지 확인
  - 해당 폴더가 없는 경우에 [File].mkdir() 메소드를 사용해서 폴더를 생성
  - mkdir() 메소드의 반환 값은 boolean 타입
    - 폴더가 성공적으로 생성되었을 경우 true를, 생성되지 않은 경우 false를 반환

- 파일 생성
  - createNewFile() 메소드를 사용해 새로운 파일을 생성

File과 관련된 정보 출력

```
public class Main {
    public static void main(String[] args) {
        String path = "c://test//filesample.txt";
        File f1 = new File(path);
        p("파일 이름 : " + f1.getName());
        p("파일 경로 : " + f1.getPath());
        p("절대 경로 : " + f1.getAbsolutePath());
        p(f1.exists() ? "파일 존재" : "파일 없음");
        p(f1.canWrite() ? "수정가능" : "수정 불가능");
        p(f1.canRead() ? "읽기가능" : "읽기 불가능");
        p(f1.isDirectory() ? "디렉토리" : "디렉토리아님");
        p(f1.isFile() ? "파일" : "파일 아님");
        p(f1.isAbsolute() ? "절대 경로" : "상대 경로");
        p("1970년 1월 1일부터 파일이 마지막 수정된 날짜까지의 밀리초 : "
                                                    + f1.lastModified());
        p("파일의 크기 : " + f1.length() + " Bytes");
    }

    static void p(String s) {
        System.out.println(s);
    }
}
```

File과 관련된 정보 출력

- test.txt 파일을 프로젝트 폴더 아래에 data 폴더를 생성하여 그 아래 넣어두자
- new File("."); 경로는 해당 프로젝트의 폴더 경로로 상대경로로 지정할 수 있음
- test.txt의 실제 경로는 프로젝트 폴더 아래에 data₩test.txt 이지만, ₩는 이스케이프 문자로 인식되므로, ₩₩로 해줘야 실제 ₩로 인식됨
- new File("C:₩₩ … "); 같이 절대 경로도 표현 가능

File 이동 예제

```
public class Main {

    public static void main(String[] args) {
        File myDir = new File("data");
        System.out.println("절대 경로 : " + myDir.getAbsolutePath());
        boolean dir = myDir.mkdir();
        if (dir) {
            System.out.println("디렉토리 생성");
        }
        File moveFile = new File("c:₩₩test₩₩filesample.txt");
        boolean move = moveFile.renameTo(new File(myDir.getAbsolutePath()
                + File.separator + moveFile.getName()));
        if (move) {
            System.out.println("이동완료");
        }
    }
}
```

File 이동 예제

- JAVA에서 File 이동하는 방법
  - File.renameTo() 메소드
    - 제일 많이 사용하는 것
    - renameTo를 사용하려면 dst 쪽의 Directory가 존재해야 함, Directory가 없다면 Error가 발생
  - Files.move() 메소드
    - option을 사용하여 File이 존재할 경우, 덮어씌우도록 할 수 있음
    - 그러나 Directory가 없다면, Error가 발생
  - FileUtils.moveFile() 메소드
    - 만약 복사할 곳의 Folder가 존재하지 않는다면, Folder를 따로 만들 필요 없이 생성까지 해줌

임시 File

- JAVA에서 File을 생성하고 관리할 때, 임시 파일을 생성하고 해당 Process가 종료되면 삭제되도록 하는 방법이 있음

- 메소드

메소드	설명
createTempFile()	임시 파일을 생성
deleteOnExit()	JVM이 종료 될 때 자동으로 지정된 파일을 삭제 하게 됨

임시 File

- createTempFile(String prefix, String suffix, File directory)
  - prefix
    - 접두사 문자열은 File 이름을 정의
    - 문자열은 최소 3개 이상의 문자이어야 함
  - suffix
    - 접미사 문자열은 File의 확장자를 정의함
    - 지정하지 않으면 “*.tmp"가 사용됨
  - directory
    - File을 만들 Directory 지정
    - 지정하지 않으면 기본 임시 File Directory C:₩WINDOWS₩TEMP₩ 폴더에 생성
  - 똑같은 File명으로 만들어도 알아서 숫자가 붙는 것이 특징

임시 File

- deleteOnExit()
  - JVM System이 종료될 때 추상 경로 이름으로 정의된 File 또는 Directory를 삭제
  - File 또는 Directory는 등록 될 때 역순으로 삭제 됨
  - 설정을 꼭 해주는 게 좋음
  - 안 그러면 실행할 때마다 무한정 File이 계속해서 생겨나는 문제점이 있음

임시 File을 생성 예제

```
public static void main(String[] args) {
    FileWriter writer = null;
    try {
        File file = File.createTempFile("temp", ".txt", new File("C:₩₩test"));
        writer = new FileWriter(file);
        writer.write('자');
        writer.write('바');
        file.deleteOnExit();
    } catch (IOException ioe) {
        System.out.println("임시 파일에 쓸 수 없습니다.");
    } finally {
        try {
            writer.close();
        }
        catch (Exception e) {
        }
    }
}
```

임시 File을 생성 예제

- File Ojbect 내의 createTempFile() 메소드를 이용하여 임시 File을 생성 할 수 있음
- 말 그대로 임시 File이므로 사용이 끝나면 삭제 처리 또한 잊지 말고 해주어야 함
- 삭제 처리를 하지 않는다면 수많은 임시 File만 남게 될 것
- File 삭제를 위해서 deleteOnExit()를 사용
- 이 메소드는 특징은 삭제 처리한다고 바로 File을 지우는 것이 아닌 JVM이 종료 될 때 자동으로 지정된 File을 삭제 하게 됨

임시 File 생성 예제

```
public static void main(String[] args) {
    try{
        // test_xxxxxxxxx.tmp 라는 임시파일을 생성한다
        File tempfile = File.createTempFile("test_", ".tmp", new File(".₩₩data"));
        // 생성된 파일의 경로를 system.out 에 출력한다.
        System.out.println(tempfile.getAbsolutePath());
        // 생성된 파일은 해당 프로세스가 종료될 때 지운다.
        tempfile.deleteOnExit();
            // 5초 동안 멈춘다.
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```