File Class

File Class - 파일의 컨트롤

- 디렉터리의 생성 및 소멸
- 파일의 소멸 (생성은 I/O 스트림으로도 가능하므로)
- 디렉토리 내의 존재하는 파일이름 출력

- 디렉토리 생성

```
public static void main(String[] args){
     File myDir = new File("C:\\YourJava\\JavaDir");
     // 인스턴스를 생성한다고 해서 파일, 디렉토리가 생성되지 않는다.
     // 생성자에 경로정보만 전달할 뿐, 인스턴스 생성은 파일의 존재유무와 상관없음

     // 파일경로 정보를 이용해 파일 컨트롤
     myDir.mkDir(); // 디렉토리 생성
     . . . . .
}
```

- 파일 이동

```
public static void main(String[] args){
     File myFile = new File("C:\\MyJava\\my.bin");
     File reFile = new File("C:\\YourJava\\my.bin");

     // exists() : 파일이 존재하면 true, 존재하지 않으면 false 반환
     if(myFile.exists()==false){
          System.out.println("해당 파일이 준비되어 있지 않습니다.");
          return;
     }

     // 파일경로 정보를 이용해 파일 컨트롤
     myFile.renameTo(reFile); // 파일 이동
     . . . . .
}
```

Write Once, Run Anywhere! - 디렉터리, 파일 경로 구분자의 separator

- 이스케이프 시퀀스 (Escape Sequence)
  Java에서는 문자열 안에서 \를 만나면 \다음의 문자를 특별한 의미로 해석되는 문자로 취급한다.

\n	개행
\t	탭
\"	큰따옴표
\\	역슬래쉬

- Windows에서는 \ 로 경로를 구분한다.
  C:\MyJava\Bin

그런데 Java Code상에서 문자열에 역슬래쉬를 표현하려면 이스케이프 시퀀스를 사용해야 한다.
C:\\MyJava\\Bin

- Linux에서는 / 로 경로를 구분한다.
  C:/MyJava/Bin

그런데 문제는 OS마다 다른 구분자를 사용하고 있다는 것이다.
그래서 운영체제에 상관없이 구분자를 사용하려면 어떻게 할까?

File Class의 separator 상수

- public static final String separator
- public static final char separatorChar

Separator 상수는 운영체제 별 구분자가 문자열, 문자형태로 저장되어 있다.
구동하는 OS 환경에 맞게 구분자 정보를 담는다.
따라서, separator 상수를 사용하면 OS에 상관없이 구분자 사용가능!

```
public static void main(String[] args){
     File myFile =
          new File("C:"+File.separator+"MyJava"+File.separator+"my.bin");

     if(myFile.exists()==false){
          System.out.println("해당 파일이 준비되어 있지 않습니다.");
          return;
     }
     . . . . .
}
```

File Class 기반의 IO Stream 생성

- File 인스턴스를 생성하고 이를 이용해 I/O Stream을 생성하면 보다 다양한 메소드의 호출이 가능해진다.
- File 기반 I/O Stream Class들은 다음과 같이 생성자가 정의되어 있다.

```
• public FileInputStream(File file)        // FileInputStream의 생성자
            throws FileNotFoundException
• public FileOutputStream(File file)       // FileOutputStream의 생성자
            throws FileNotFoundException
• public FileReader(File file)         // FileReader의 생성자
            throws FileNotFoundException
• public FileWriter(File file)         // FileWriter의 생성자
            throws IOException
```

```
File inFile = new File("data.bin");

if(inFile.exists()==false){
     // 데이터를 읽어들일 대상 파일이 존재하지 않을 때의 적절한 처리
}

InputStream in = new FileInputStream(inFile);
```

상대경로 기반의 File 인스턴스 생성

실제 프로그램 개발은 상대경로를 이용하는 것이 일반적이다.
그래야 실행환경 및 실행위치 변경에 따른 문제점을 최소화할 수 있다.

- **상대경로**
  현재 디렉토리 위치를 기준으로 한 경로정보

  AAA\BBB
  현재 위치의 디렉토리의 하위 디렉토리 AAA의 하위 디렉토리 BBB 위치

- **절대경로**
  드라이브 명을 기준으로 한 경로정보

  C:\AAA\BBB
  C 드라이브 하위의 AAA 디렉토리 하위의 BBB 디렉토리

- **Java 프로그램에서의 상대경로의 현재 디렉토리 위치는**
  실행되고 있는 Java Program이 위치한 디렉토리가 된다.

- **getAbsolutePath()**
  File 인스턴스에 전달된 위치경로에 대한 절대경로 반환

```
import java.io.File;

public class RelativePath {
     public static void main(String[] args) {
          File curDir = new File("AAA");
          System.out.println(curDir.getAbsolutePath());

          File upperDir = new File("AAA"+File.separator+"BBB");
          System.out.println(upperDir.getAbsolutePath());
     }
}
```
