

프로그래밍기초

Chapter 12. 파일 입출력

Section 01

파일 처리

파일 처리

■ 파일 처리의 개념

- 파일 처리란 읽기, 쓰기, 편집 등 파일에 대한 다양한 기능을 수행하는 것을 말함
- 필요한 모든 메서드는 java.io 패키지에 있으므로, 파일 처리를 위해 프로그램을 시작 하기 전에 다음 패키지를 포함해야 함

```
import java.io.*;
```

- ✓ java.io 패키지에는 자바에서 입출력(I/O)을 수행하는 데 필요한 거의 모든 클래스가 포함되어 있음

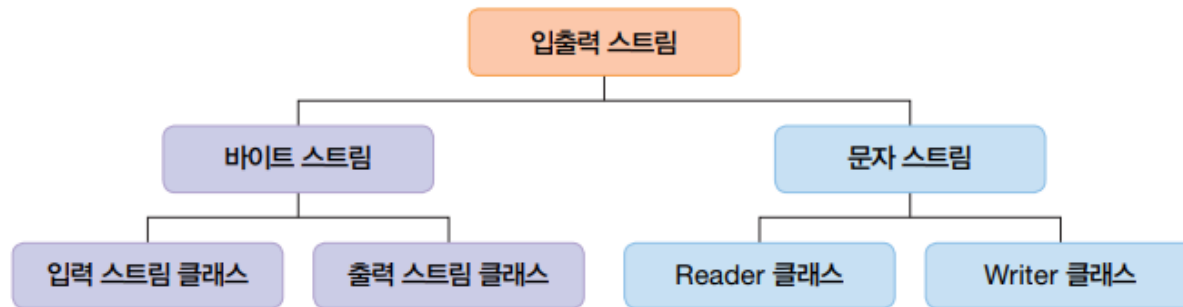


파일 처리

■ 입출력 스트림

• 입출력 스트림의 유형

→ 스트림은 데이터를 전달하는 방식에 따라 바이트 스트림(byte stream)과 문자 스트림(character stream)으로 구분할 수 있음



→ 바이트 스트림: 영상, 음성, 영문자 등의 바이너리 데이터를 처리, 1바이트 단위의 입출력을 처리

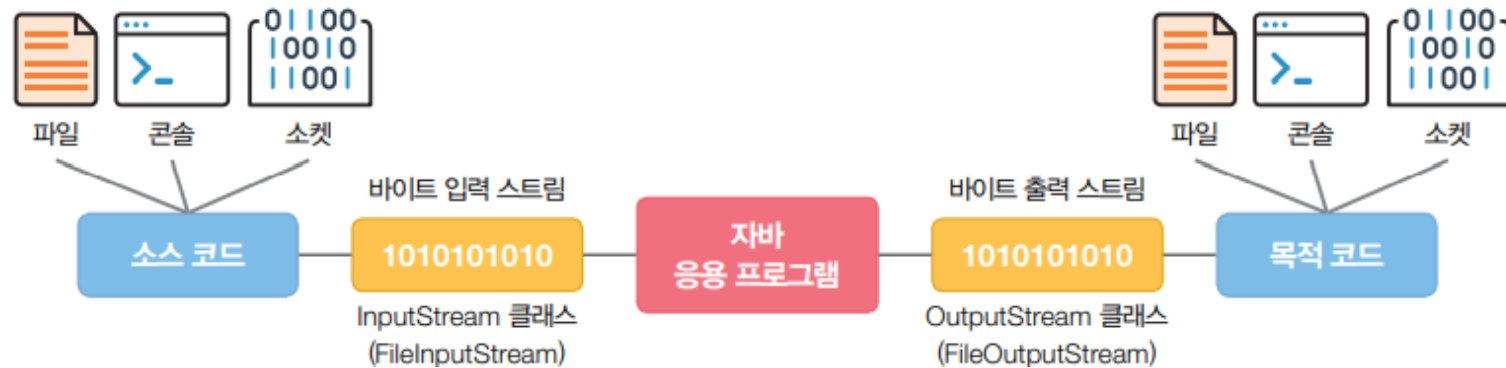
→ 문자 스트림: 2바이트 단위 의 문자 입출력을 처리, 다국어를 표현하는 유니코드를 처리

파일 처리

■ 입출력 스트림

- 바이트 스트림

→ 한 번에 8비트 데이터를 처리하고 8비트당 입출력 연산을 수행하므로 주로 바이트 데이터의 입출력을 처리



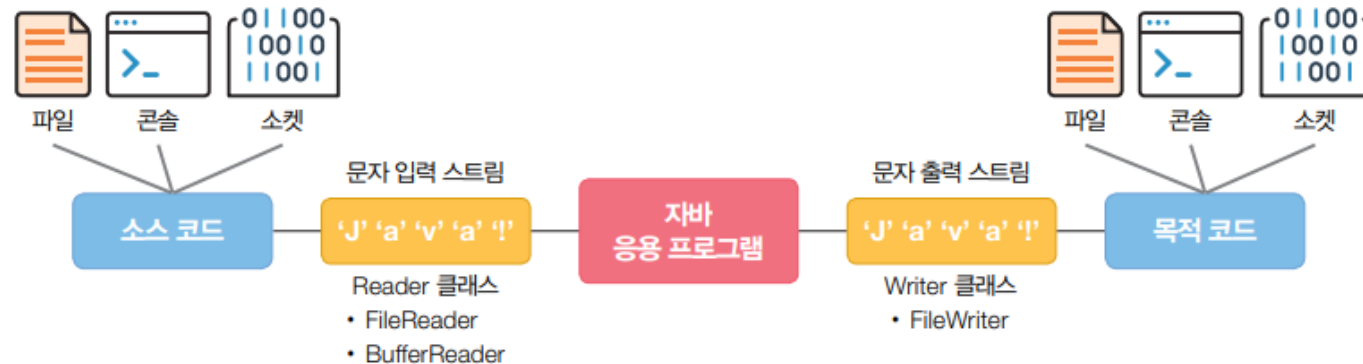
→ 바이트 스트림에서 가장 일반적으로 사용되는 입력 스트림 클래스는 FileInputStream, 출력 스트림 클래스는 FileOutputStream

파일 처리

■ 입출력 스트림

• 문자 스트림

- 한 번에 16비트 유니코드 데이터의 입출력을 처리하므로 바이트 스트림보다 2배의 입출력 연산을 수행할 수 있어 더 빠름
- 일반적으로 문자 스트림은 바이트 스트림 클래스를 사용하여 구현함



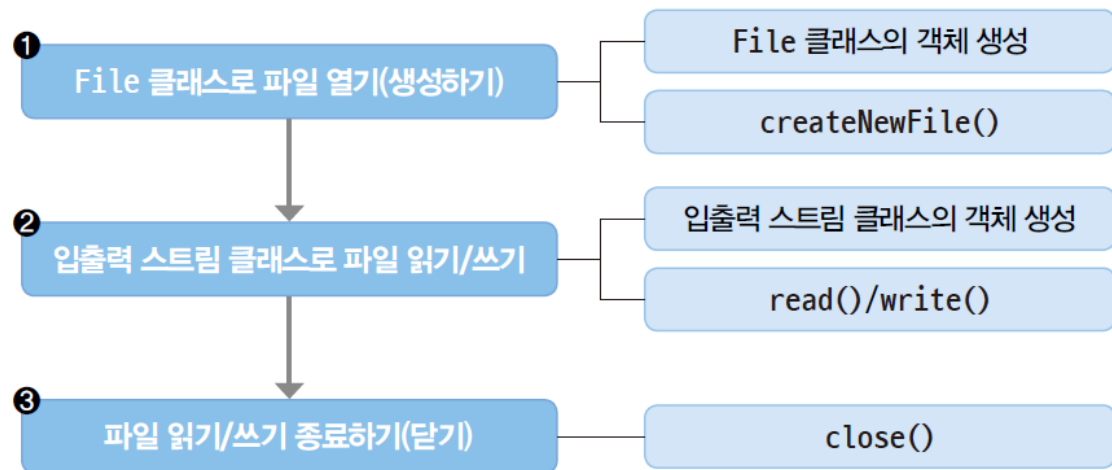
- 문자 스트림에서 가장 일반적으로 사용되는 읽기 스트림 클래스는 FileReader, 쓰기 스트림 클래스는 FileWriter

파일 처리

■ 파일 처리 과정

- **① File 클래스로 파일 열기(생성하기)** File 클래스를 이용하여 특정 경로에 저장된 파일 열기
→ File 클래스의 생성자에 특정 경로의 파일명을 지정하고 File 클래스의 객체를 생성하고, 특정 경로의 파일명이 생성된 것인지 `createNewFile()` 메서드로 확인함
- **② 입출력 스트림 클래스로 파일 읽기/쓰기** 입출력 스트림 클래스를 이용하여 파일 내용 읽기 또는 쓰기 작업 수행
→ 입출력 스트림 클래스의 생성자로 객체를 생성하며, 생성된 객체를 통해 읽기 작업은 `read()` 메서드로, 쓰기 작업은 `write()` 메서드로 수행함
- **③ 파일 읽기/쓰기 종료하기(닫기)** 파일 읽기 또는 쓰기 작업을 완료한 뒤 이를 종료
→ 입출력 스트림 클래스로 생성된 객체를 통해 `close()` 메서드로 파일 읽기 또는 쓰기 작업을 닫음

■ 파일 읽기/쓰기 처리 과정



■ 입출력 스트림 클래스를 이용한 파일 읽기/쓰기 처리 과정



Section 02

파일 생성

파일 생성

■ 파일 생성

• File 클래스

→ 파일 또는 디렉터리 생성, 삭제 및 이름 변경, 디렉터리 내용 나열 등 파일 또는 디렉터리에 대해 다양한 작업을 수행하는 메서드가 포함되어 있음

```
File 파일객체명 = new File("[디렉터리명]");  
File file = new File("test.txt");
```

| 생성자 | 설명 |
|-----------------------------------|--|
| File(File parent, String child) | 부모 추상 경로명과 자식 문자열 경로명에서 파일 인스턴스를 만든다. |
| File(String pathname) | 주어진 경로명을 추상 경로명으로 변환하고 새 파일 인스턴스를 만든다. |
| File(String parent, String child) | 부모 경로명과 자식 경로명을 가져와서 파일 인스턴스를 만든다. |
| File(URI uri) | 주어진 URI를 추상 경로명으로 변환하여 파일 인스턴스를 만든다. |

파일 생성

■ 파일 생성

- File 클래스에 포함된 다양한 메서드 중에서 createNewFile() 메서드를 사용
→ 만약 지정된 위치에 파일이 존재하지 않으면 빈 파일을 생성하여 true를 반환하고, 파일이 존재하면 false를 반환

```
File file = new File("text.txt");  
  
try {  
    boolean b = file.createNewFile();  
    ...  
} catch(IOException e) {  
    ...  
}
```

try 블록 내부에서 생성해도 상관 없음

파일 생성

■ 파일 생성

- File 클래스에서 사용되는 주요 메서드

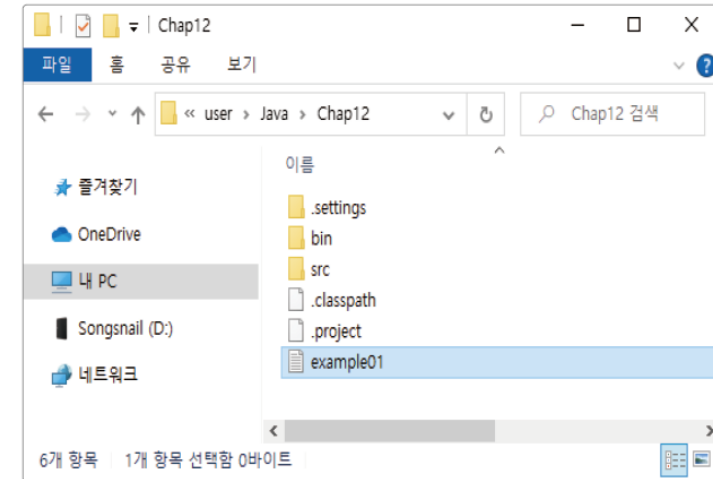
| 메서드 | 설명 |
|---------------------------------------|--|
| <code>Boolean canRead()</code> | 파일을 읽을 수 있는지 확인한다. |
| <code>Boolean createNewFile()</code> | 원하는 경로에 새 파일을 생성한다. 보통은 생성된 파일이 비어 있다. |
| <code>Boolean canWrite()</code> | 파일에 쓸 수 있는지, 즉 읽기 전용 파일이 아닌지 확인한다. |
| <code>Boolean exists()</code> | 요청한 파일이 디렉터리에 있는지 확인한다. |
| <code>Boolean delete()</code> | 디렉터리에서 파일을 삭제한다. |
| <code>String getName()</code> | 디렉터리에서 특정 파일명을 찾는다. |
| <code>String getAbsolutePath()</code> | 주어진 파일의 절대 경로를 반환한다. |
| <code>Long length()</code> | 파일 크기를 바이트 단위로 반환한다. |
| <code>String[] list()</code> | 현재 작업 디렉터리(폴더)에 있는 모든 파일을 반환한다. |
| <code>Boolean mkdir()</code> | 새 디렉터리(파일 아님)를 생성한다. |

파일 생성

■ file 클래스를 이용한 파일 생성 예시

```
import java.io.File;
import java.io.IOException;
public class Example01 {
    public static void main(String[] args) {
        File fileObj = new File("example01.txt");
        try {
            boolean success = fileObj.createNewFile();
            if (success) {
                System.out.println("파일 생성 성공");
            } else {
                System.out.println("파일 생성 실패");
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

실행 결과
파일 생성 성공



파일 생성

■ 예제 12-1. Example01.java 파일 정보 얻어오기

```
01 import java.io.File;
02
03 public class FileHandling01 {
04     public static void main(String[] args) {
05
06         File finfo = new File(" src\\Example01.java ");
07
08         if (finfo.exists()) {
09             System.out.println("파일의 이름: " + finfo. getName() );
10             System.out.println("파일의 경로: " + finfo. getAbsolutePath() );
11             System.out.println("파일 쓰기가 가능한가?: " + finfo. canWrite() );
12             System.out.println("파일 읽기가 가능한가?: " + finfo. canRead() );
13             System.out.println("파일의 크기: " + finfo. length() );
14         } else {
15             System.out.println("존재하는 파일이 아닙니다.");
16         }
17     }
18 }
```

실행 결과

파일의 이름: Example01.java

파일의 경로: C:\Users\user\Java\Chap12\src\Example01.java

파일 쓰기가 가능한가?: true

파일 읽기가 가능한가?: true

파일의 크기: 545

Section 03

파일 쓰기

파일 쓰기

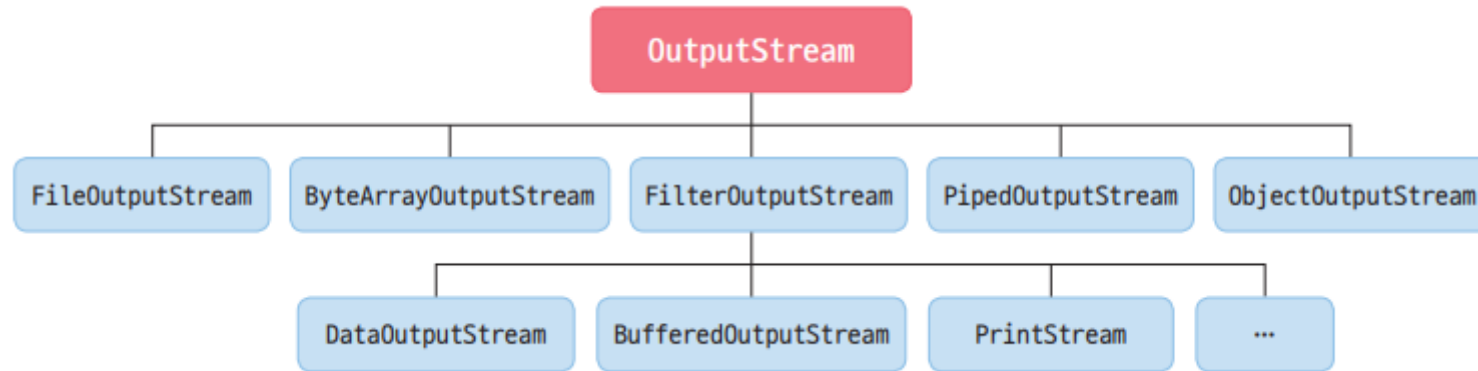
■ 파일 쓰기

- 파일 쓰기에는 바이트 출력 스트림 유형과 문자 출력 스트림 유형이 있음
- 바이트 출력 스트림 유형
 - 응용 프로그램에서 출력 장치로 바이트 데이터를 전송하는 방법
- 문자 출력 스트림 유형
 - 응용 프로그램에서 출력 장치로 문자 데이터를 전송하는 방법

파일 쓰기

■ 바이트 출력 스트림 유형

- OutputStream 클래스
 - 출력(데이터 쓰기)을 하는 추상 클래스
 - 바이트 출력 스트림을 나타내는 모든 클래스의 슈퍼 클래스



파일 쓰기

■ 바이트 출력 스트림 유형

- OutputStream 클래스의 주요 자식 클래스

| 클래스 | 설명 |
|-----------------------|---|
| FileOutputStream | 파일에 쓰는 출력 스트림이다. |
| ByteArrayOutputStream | 바이트 배열에 기록되는 출력 스트림이다. |
| FilterOutputStream | 출력 스트림을 필터링하는 모든 클래스의 상위 클래스이다. |
| DataOutputStream | 표준 데이터 유형을 작성하는 방법을 포함하는 출력 스트림이다. |
| BufferedOutputStream | 버퍼링된 출력 스트림에 사용된다. |
| PrintStream | print(), println() 메서드를 포함하는 출력 스트림이다. |
| PipedOutputStream | 주로 멀티스레드 처리에 사용하며, PipedInputStream에 연결하여 통신 파이프를 생성한다. |
| ObjectOutputStream | 객체를 출력할 수 있는 보조 스트림이다. |

- OutputStream 클래스의 주요 메서드

| 메서드 | 설명 |
|----------------------|---------------------------------|
| void write(int) | 출력 스트림에 바이트를 쓴다. |
| void write(byte[] b) | 출력 스트림에 바이트 배열을 쓴다. |
| void flush() | 출력 스트림에 저장된 버퍼의 내용을 쓰고 버퍼를 비운다. |
| void close() | 출력 스트림을 닫는다. |

파일 쓰기

■ 바이트 출력 스트림 유형

- FileOutputStream 클래스

- 파일에 데이터를 쓰는 데 사용되는 출력 스트림으로, 파일에 기본값(primitive value)을 쓰는 경우 FileOutputStream 클래스를 사용

- FileOutputStream 클래스를 통해 바이트 지향 데이터와 문자 지향 데이터를 모두 작성할 수 있음

| 메서드 | 설명 |
|---|---|
| <code>void finalize()</code> | 파일 출력 스트림과의 연결을 정리한다. |
| <code>void write(byte[] b)</code> | 바이트 배열의 <code>b.length</code> 바이트를 파일 출력 스트림에 쓴다. |
| <code>void write(byte[] b, int off, int len)</code> | 오프셋에서 시작하는 바이트 배열의 <code>len</code> 바이트를 파일 출력 스트림에 쓴다. |
| <code>void write(int b)</code> | 지정된 바이트를 파일 출력 스트림에 쓴다. |
| <code>FileChannel getChannel()</code> | 파일 출력 스트림과 연결된 파일 채널 개체를 반환한다. |
| <code>FileDescriptor getFD()</code> | 스트림과 관련된 파일 설명자를 반환한다. |
| <code>void close()</code> | 파일 출력 스트림을 닫는다. |

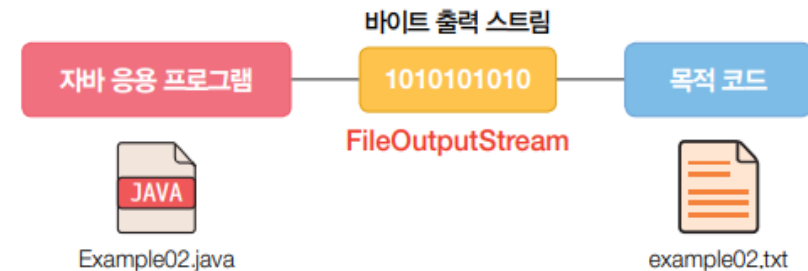
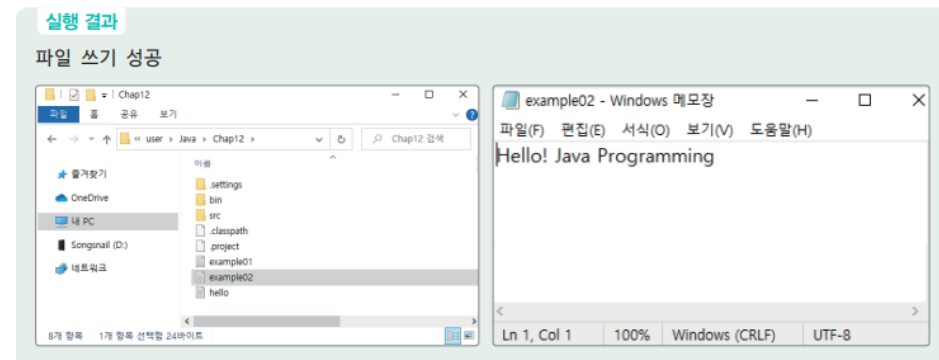
파일 쓰기

■ FileOutputStream 클래스를 이용한 파일 쓰기 예시

```
import java.io.File;
import java.io.FileOutputStream;
public class Example02 {
    public static void main(String[] args) {
        String str = "Hello! Java Programming ";
        try {
            File file = new File("example02.txt");
            if (!file.exists())
                file.createNewFile();

            FileOutputStream fos = new FileOutputStream(file);

            byte[] b = str.getBytes();
            fos.write(b);
            fos.close();
            System.out.println("파일 쓰기 성공");
        } catch (Exception e) {
            e.getMessage();
        }
    }
}
```



파일 쓰기

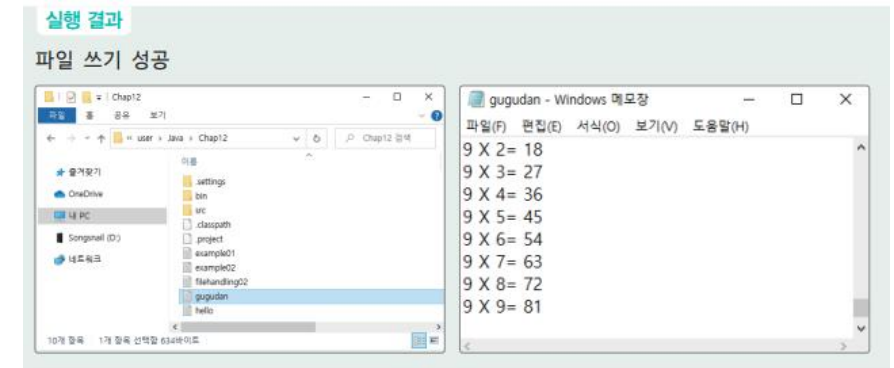
■ 예제 12-2. 파일에 구구단 저장하기

```
01 import java.io.File;
02 import java.io.FileOutputStream;
03
04 public class FileHandling02 {
05     public static void main(String[] args) {
06         File file = new File("gugudan.txt");
07         try {
08             if (!file.exists())
09                 file.createNewFile();
10
11             FileOutputStream fos = new FileOutputStream(file);
12
```

파일 쓰기

■ 예제 12-2. 파일에 구구단 저장하기

```
13     for (int x = 2; x <= 9; x++) {
14         for (int y = 1; y <= 9; y++) {
15             String str = x + " X " + y + "= " + (x * y) + "\n";
16             byte[] b = str.getBytes();
17             fos.write(b);
18         }
19     }
20     fos.close();
21     System.out.println("파일 쓰기 성공");
22 } catch (Exception e) {
23     e.getMessage();
24 }
25 }
26 }
```



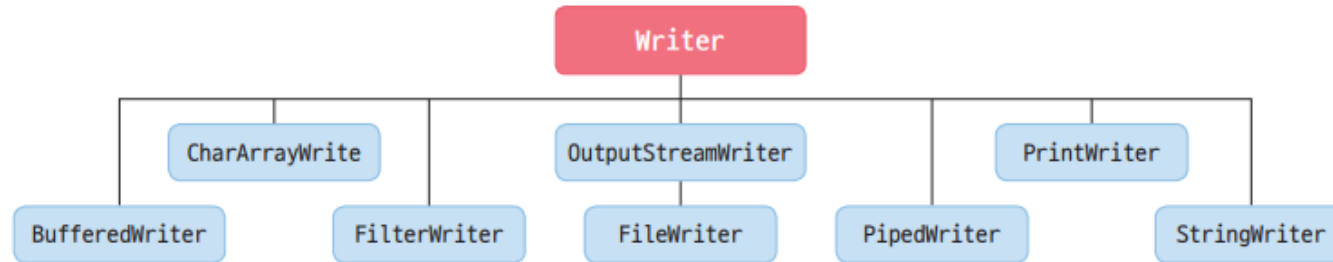
파일 쓰기

■ 문자 출력 스트림 유형

- Writer 클래스

- 문자 스트림에 쓰기 위한 추상 클래스

- 서브클래스가 구현해야 하는 추상 메서드: write(), flush(), close()



| 클래스 | 설명 |
|--------------------|--|
| BufferedWriter | 버퍼링된 출력 스트림을 처리한다. |
| CharArrayWriter | Writer로 사용할 수 있는 문자 버퍼를 구현한다. |
| FilterWriter | 필터링된 문자 스트림을 작성하기 위한 추상 클래스이다. |
| OutputStreamWriter | 문자를 바이트로 변환하는 출력 스트림이다. |
| FileWriter | 파일에 쓰는 출력 스트림이다. |
| PipedWriter | 파이프된 문자 출력 스트림을 처리한다. |
| StringWriter | 문자열을 구성하는 데 사용할 수 있는 문자열 버퍼에서 출력을 수집하는 문자 스트림이다. |
| PrintWriter | 식이 지정된 표현을 텍스트 출력 스트림에 출력한다. |

파일 쓰기

■ 문자 출력 스트림 유형

• FileWriter 클래스

→ 문자 데이터를 파일에 쓰는 데 사용되며, `FileOutputStream` 클래스와 달리 문자열을 직접 쓰는 방법을 제공하므로 문자열을 바이트 배열로 변환할 필요가 없음

| 생성자 | 설명 |
|---------------------------------------|---------------------------------|
| <code>FilterWriter(Writer out)</code> | 새 파일을 생성한다. 문자열에서 파일명을 얻는다. |
| <code>FileWriter(File file)</code> | 새 파일을 생성한다. File 객체에서 파일명을 얻는다. |

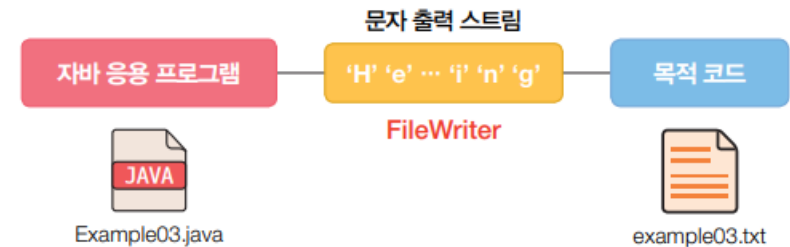
| 메서드 | 설명 |
|--------------------------------------|--------------------------|
| <code>void write(String text)</code> | FileWriter에 문자열을 쓴다. |
| <code>void write(char c)</code> | FileWriter에 문자를 쓴다. |
| <code>void write(char[] c)</code> | FileWriter에 char 배열을 쓴다. |
| <code>void flush()</code> | FileWriter의 데이터를 플러시한다. |
| <code>void close()</code> | FileWriter를 닫는다. |

파일 쓰기

■ FileWriter 클래스를 이용한 파일 쓰기 예시

```
import java.io.File;
import java.io.FileWriter;
public class Example03 {
    public static void main(String[] args) {
        try {
            File file = new File("example03.txt");
            if (!file.exists())
                file.createNewFile();
            FileWriter myWriter = new FileWriter(file);

            myWriter.write("Hello!!\n");
            myWriter.write("Java Programming");
            myWriter.close();
            System.out.println("파일 쓰기 성공");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



파일 쓰기

■ 예제 12-3. 키보드로 입력된 값을 파일에 저장하기

```
01 import java.io.File;
02 import java.io.FileWriter;
03 import java.util.Scanner;
04
05 public class FileHandling03 {
06     public static void main(String[] args) {
07         File file = new File("member.txt");
08
09         try {
10             if (!file.exists())
11                 file.createNewFile();
12
13             FileWriter fw = new FileWriter(file);
14             Scanner input = new Scanner(System.in);
15
16             boolean quit = false;
17             while (!quit) {
18                 System.out.print("아이디 : ");
19                 String userID = input.next();
20                 fw.write("아이디 : " + userID + " ");
```

25

파일 쓰기

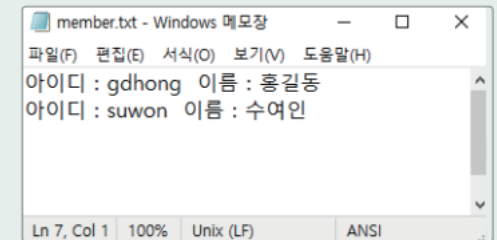
■ 예제 12-3. 키보드로 입력된 값을 파일에 저장하기

```
22      System.out.print("이름 : ");
23      String userName = input.next();
24      fw.write("이름 : " + userName + "\n");
25
26      System.out.println("계속 진행? Y | N ");
27      input = new Scanner(System.in);
28      String str = input.nextLine();
29
30      if (str.toUpperCase().equals("N"))
31          quit = true;
32  }
33  fw.close();
34  System.out.println("파일 쓰기 성공");
35
36  } catch (Exception e) {
37      e.getMessage();
38  }
39  }
40 }
```

실행 결과

```
아이디 : gdhong
이름 : 홍길동
계속 진행? Y | N
y
아이디 : suwon
이름 : 수여인
계속 진행? Y | N
N
```

파일 쓰기 성공



member.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

아이디 : gdhong 이름 : 홍길동
아이디 : suwon 이름 : 수여인

Ln 7, Col 1 100% Unix (LF) ANSI

Section 04

파일 읽기

파일 읽기

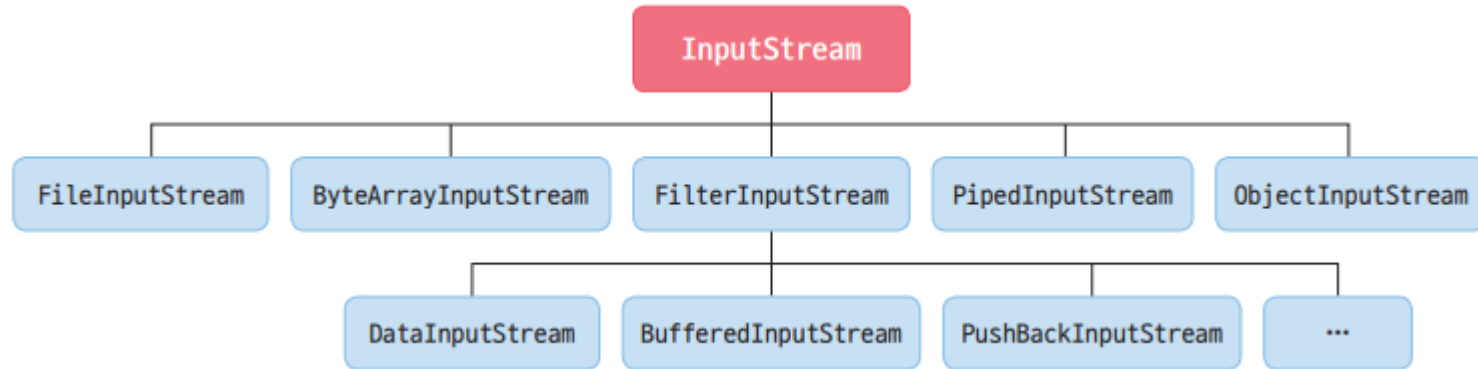
■ 파일 읽기

- 바이트 입력 스트림 유형과 문자 입력 스트림 유형
- 바이트 입력 스트림 유형
 - 입력 장치에서 응용 프로그램으로 바이트 데이터를 전송하는 방법
- 문자 입력 스트림 유형
 - 입력 장치에서 응용 프로그램으로 문자 데이터를 전송하는 방법

파일 읽기

■ 바이트 입력 스트림 유형

- InputStream 클래스는 입력(데이터 읽기)을 가져오는 데 사용하는 추상 클래스
- 바이트 입력 스트림을 나타내는 모든 클래스의 슈퍼 클래스임



파일 읽기

■ 바이트 입력 스트림 유형

- InputStream 클래스의 주요 자식 클래스

| 클래스 | 설명 |
|----------------------|---|
| FileInputStream | 파일에서 읽는 입력 스트림이다. |
| ByteArrayInputStream | 바이트 배열을 차례대로 읽어들이는 입력 스트림이다. |
| FilterInputStream | 필터를 이용한 입력 스트림이다. |
| DataInputStream | 표준 데이터 유형을 읽는 방법을 포함하는 입력 스트림이다. |
| BufferedInputStream | 버퍼링된 입력 스트림을 처리한다. |
| PushBackInputStream | print(), println() 메서드를 포함하는 입력 스트림이다. |
| PipedInputStream | 주로 멀티스레드 처리에 사용하며 PipedOutputStream에 연결해야 한다. PipedOutputStream에 기록되는 데이터 바이트를 제공한다. |
| ObjectInputStream | 객체를 입력할 수 있는 보조 스트림이다. |

- InputStream 클래스의 주요 메서드

| 메서드 | 설명 |
|--------------------|---|
| int read() | 입력 스트림에서 데이터의 다음 바이트를 읽는다. 파일 끝에서 -1을 반환한다. |
| int read(byte[] b) | 입력 스트림에서 몇 바이트를 읽어 버퍼 배열 b에 저장한다. |
| int available() | 입력 스트림에서 읽을 수 있는 바이트 수의 추정치를 반환한다. |
| void close() | 입력 스트림을 닫는다. |

파일 읽기

■ 바이트 입력 스트림 유형

- FileInputStream 클래스

→ 파일에서 이미지 데이터, 오디오, 동영상 등의 바이트 지향 데이터(원시 바이트 스트림)를 읽어오는 데 사용되는 입력 스트림

| 메서드 | 설명 |
|---|---|
| <code>int available()</code> | 입력 스트림에서 읽을 수 있는 예상 바이트 수를 반환한다. |
| <code>int read()</code> | 입력 스트림에서 데이터 바이트를 읽는다. |
| <code>int read(byte[] b)</code> | 입력 스트림에서 최대 <code>b.length</code> 바이트의 데이터를 읽는다. |
| <code>int read(byte[] b, int off, int len)</code> | 입력 스트림에서 최대 <code>len</code> 바이트의 데이터를 읽는다. |
| <code>long skip(long x)</code> | 입력 스트림에서 <code>x</code> 바이트의 데이터를 건너뛰고 삭제한다. |
| <code>FileChannel getChannel()</code> | 파일 입력 스트림과 연결된 고유한 <code>FileChannel</code> 객체를 반환한다. |
| <code>FileDescriptor getFD()</code> | <code>FileDescriptor</code> 객체를 반환한다. |
| <code>protected void finalize()</code> | 파일 입력 스트림에 대한 참조가 더 이상 없을 때 <code>close()</code> 메서드를 호출한다. |
| <code>void close()</code> | 스트림을 닫는다. |

파일 읽기

■ 바이트 입력 스트림 유형

- FileInputStream 클래스를 이용한 파일 읽기 예시

```
import java.io.File;
import java.io.FileInputStream;
public class Example04 {
    public static void main(String[] args) {

        try {
            File file = new File("example03.txt");
            if (!file.exists())
                file.createNewFile();

            FileInputStream fis = new FileInputStream(file);
            int i = 0;

            while ((i=fis.read())!= -1) {
                System.out.print((char)i);
            }
            fis.close();
            System.out.println("\n파일 읽기 성공");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



파일 읽기

■ 예제 12-4. 파일에서 구구단 읽어오기

```
01 import java.io.File;
02 import java.io.FileInputStream;
03
04 public class FileHandling04 {
05     public static void main(String[] args) {
06         File file = new File("gugudan.txt");
07         try {
08             if (!file.exists())
09                 file.createNewFile();
10
11             FileInputStream fis = new FileInputStream(file);
12             int i = 0;
13
14             while ((i = fis.read()) != -1) {
15                 System.out.print((char) i);
16             }
```

파일 읽기

■ 예제 12-4. 파일에서 구구단 읽어오기

```
17
18     fis.close();
19     System.out.println("파일 읽기 성공");
20 } catch (Exception e) {
21     System.out.println(e);
22 }
23 }
24 }
```

실행 결과

...

9 X 2= 18

9 X 3= 27

9 X 4= 36

9 X 5= 45

9 X 6= 54

9 X 7= 63

9 X 8= 72

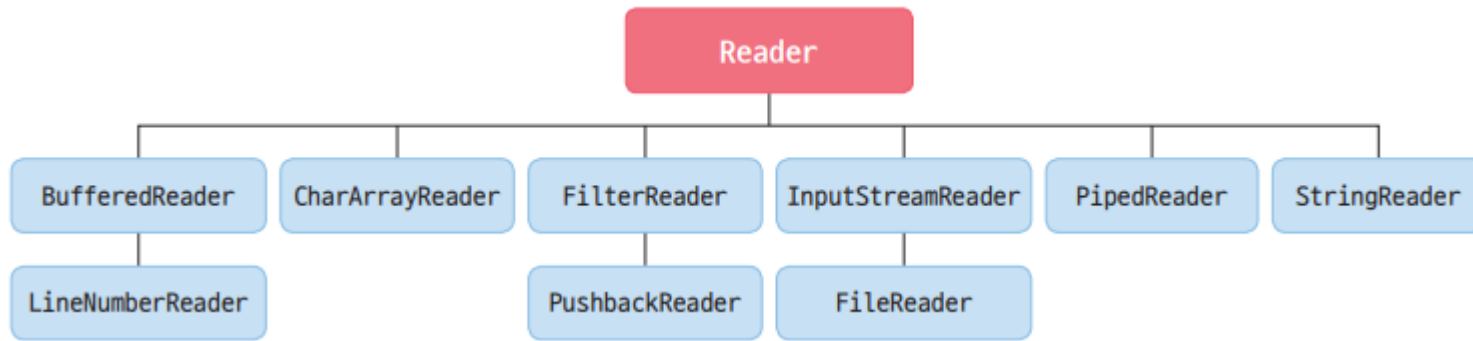
9 X 9= 81

파일 읽기 성공

파일 읽기

■ 문자 입력 스트림 유형

- Reader 클래스
 - 문자 스트림을 읽기 위한 추상 클래스
 - 서브클래스가 구현해야 하는 추상 메서드: `read()`, `close()`



파일 읽기

■ 문자 입력 스트림 유형

| 클래스 | 설명 |
|-------------------|-----------------------------------|
| BufferedReader | 버퍼링된 문자 입력 스트림을 처리한다. |
| CharArrayReader | 문자 입력 스트림으로 사용할 수 있는 문자 버퍼를 구현한다. |
| FilterReader | 필터링된 문자 스트림을 읽기 위한 추상 클래스이다. |
| InputStreamReader | 바이트를 문자로 변환하는 입력 스트림이다. |
| FileReader | 파일에서 읽는 문자 입력 스트림이다. |
| PipedReader | 파이프된 문자 입력 스트림이다. |
| StringReader | 소스가 문자열인 문자 스트림이다. |

파일 읽기

■ 문자 입력 스트림 유형

- FileReader 클래스
- 파일에서 문자 지향 데이터를 읽는 데 사용되며, FileInputStream 클래스와 같은 바이트 형식으로 데이터를 반환
- FileReader 클래스 생성자

| 생성자 | 설명 |
|--|---|
| <code>FileReader(String fileName)</code> | 파일명 문자열을 가져오며, 주어진 파일을 읽기 모드로 연다. 파일이 존재하지 않으면 <code>FileNotFoundException</code> 이 발생한다. |
| <code>FileReader(File file)</code> | 파일 인스턴스에서 파일명을 가져오며, 주어진 파일을 읽기 모드로 연다. 파일이 존재하지 않으면 <code>FileNotFoundException</code> 이 발생한다. |

- FileReader 상위 클래스(`InputStreamReader`)의 주요 메서드

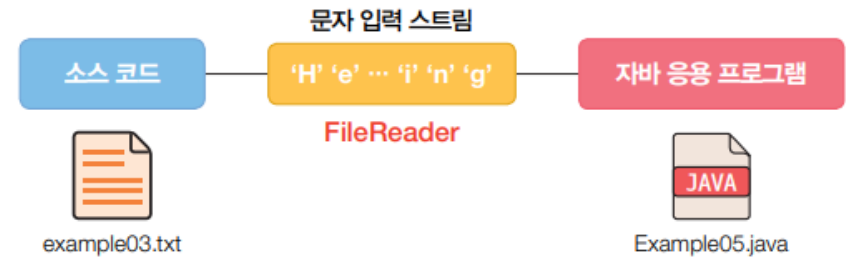
| 메서드 | 설명 |
|---------------------------|--------------------------------------|
| <code>int read()</code> | ASCII 형식의 문자를 반환한다. 파일 끝에서 -1을 반환한다. |
| <code>void close()</code> | FileReader 클래스를 닫는다. |

파일 읽기

■ 문자 입력 스트림 유형

- FileReader 클래스를 이용한 파일 읽기 예시

```
import java.io.File;
import java.io.FileReader;
public class Example05 {
    public static void main(String[] args) {
        try {
            File file = new File("example03.txt");
            if (!file.exists())
                file.createNewFile();
            FileReader fis = new FileReader(file);
            int i = 0;
            while ((i = fis.read()) != -1) {
                System.out.print((char) i);
            }
            fis.close();
            System.out.println("\n파일 읽기 성공");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



실행 결과

```
Hello!!
Java Programming
파일 읽기 성공
```

파일 읽기

■ 예제 12-5. 키보드로 입력된 저장 값 읽어오기

```
01 import java.io.File;
02 import java.io.FileReader;
03
04 public class FileHandling05 {
05     public static void main(String[] args) {
06         File file = new File("member.txt");
07         try {
08             if (!file.exists())
09                 file.createNewFile();
10
11             FileReader fis = new FileReader(file);
12             int i = 0;
13
14             while ((i = fis.read()) != -1) {
15                 System.out.print((char) i);
16             }
17         }
18     }
19 }
```

파일 읽기

■ 예제 12-5. 키보드로 입력된 저장 값 읽어오기

```
17
18     fis.close();
19     System.out.println("파일 읽기 성공");
20 } catch (Exception e) {
21     System.out.println(e);
22 }
23 }
24 }
```

실행 결과

아이디 : gdhong 이름 : 홍길동
아이디 : suwon 이름 : 수여인
파일 읽기 성공

파일 읽기

■ 문자 입력 스트림 유형

- BufferedReader 클래스

- 문자 기반 입력 스트림에서 텍스트를 읽는 데 사용됨

- readLine() 메서드로 데이터를 한 줄씩 읽을 때 사용할 수 있어 성능을 빠르게 함

- BufferedReader 클래스의 생성자

| 생성자 | 설명 |
|-----------------------------------|---|
| BufferedReader(Reader rd) | 입력 버퍼의 기본 크기를 사용하는 버퍼링된 문자 입력 스트림을 만든다. |
| BufferedReader(Reader rd, int sz) | 입력 버퍼에 대해 지정된 크기를 사용하는 버퍼링된 문자 입력 스트림을 만든다. |

- BufferedReader 클래스의 주요 메서드

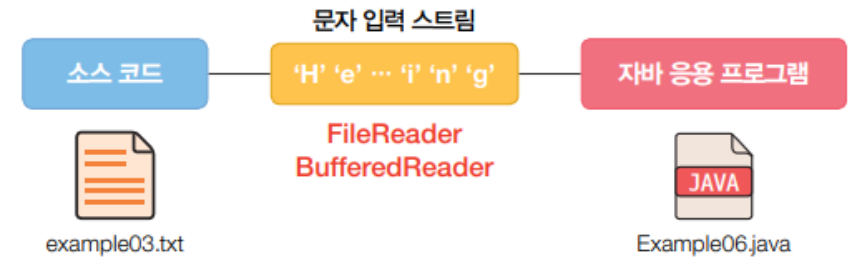
| 메서드 | 설명 |
|---|--|
| int read() | 단일 문자를 읽는다. |
| int read(char[] cbuf, int off, int len) | 배열의 일부로 문자를 읽는다. |
| boolean markSupported() | 표시 및 재설정 방법에 대한 입력 스트림 지원을 테스트한다. |
| String readLine() | 텍스트 한 줄을 읽는다. |
| boolean ready() | 입력 스트림을 읽을 준비가 되었는지 테스트한다. |
| long skip(long n) | 문자를 건너뛴다. |
| void reset() | 입력 스트림에서 mark() 메서드가 마지막으로 호출된 위치에 스트림을 재배치한다. |
| void mark(int readAheadLimit) | 스트림에서의 현재 위치를 표시한다. |
| void close() | 입력 스트림을 닫고 스트림과 연결된 시스템 리소스를 해제한다. |

파일 읽기

■ 문자 입력 스트림 유형

- BufferedReader 클래스를 이용한 파일 읽기 예시

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
public class Example06 {
    public static void main(String[] args) {
        try {
            File file = new File("example03.txt");
            if (!file.exists())
                file.createNewFile();
            FileReader fis = new FileReader(file);
            BufferedReader br = new BufferedReader(fis);
            String str;
            while ((str = br.readLine()) != null) {
                System.out.println(str);
            }
            fis.close();
            System.out.println("파일 읽기 성공");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



실행 결과

```
Hello!!
Java Programming
파일 읽기 성공
```

파일 읽기

■ 예제 12-6. 키보드로 입력된 저장 값 읽어오기

```
01 import java.io.BufferedReader;
02 import java.io.File;
03 import java.io.FileReader;
04
05 public class FileHandling06 {
06     public static void main(String[] args) {
07
08         File file = new File("member.txt");
09         try {
10             if (!file.exists())
11                 file.createNewFile();
12
13             FileReader fis = new FileReader(file);
14             BufferedReader br = new BufferedReader(fis);
15
16             String str;
```

파일 읽기

■ 예제 12-6. 키보드로 입력된 저장 값 읽어오기

```
17
18     while ((str = br.readLine()) != null) {
19         System.out.println(str);
20     }
21
22     fis.close();
23     System.out.println("파일 읽기 성공");
24 } catch (Exception e) {
25     System.out.println(e);
26 }
27 }
28 }
```

실행 결과

아이디 : gdhong 이름 : 홍길동
아이디 : suwon 이름 : 수여인
파일 읽기 성공

프로그래밍기초

Copyright © Lee Seungwon Professor
All rights reserved.

<Q&A : lsw@kopo.ac.kr>