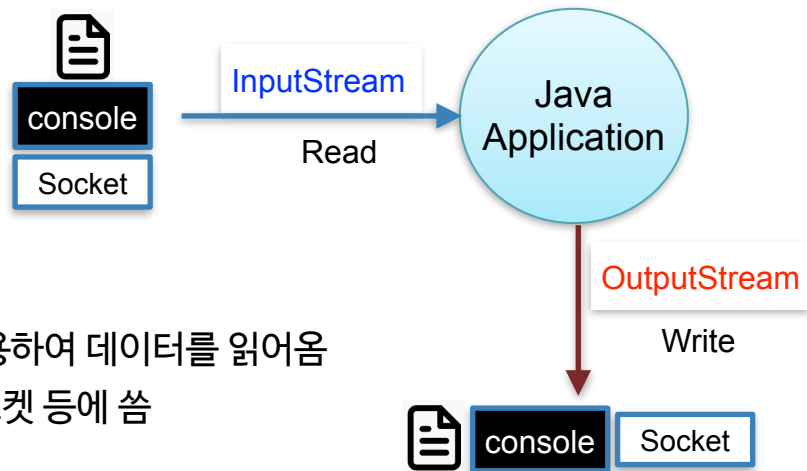


참고

Java File IO

Java input/output

- **Stream**
 - 데이터의 흐름, 바이트로 구성
 - 콘솔에 연결된 3개 스트림 자동 생성됨
 - System.out : 표준 출력 스트림
 - System.in : 표준 입력 스트림
 - System.err : 표준 에러 스트림
- **InputStream / OutputStream : Byte 단위**
 - InputStream : 파일, 주변장치, 소켓 등을 사용하여 데이터를 읽어옴
 - OutputStream : 데이터를 파일, 주변장치, 소켓 등에 씀
- **Reader / Writer : 문자 단위 (2bytes)**
- **BufferedReader / BufferedWriter : 문자열단위**



1) OutputStream class

- **abstract** class for writing
- the superclass of all classes representing an output stream of bytes
- Methods
 - public **abstract** void write(int)
 - public void write(byte[])
 - public void write(byte[], int off, int Len)
 - public void flush()
 - public void close()

Module java.base

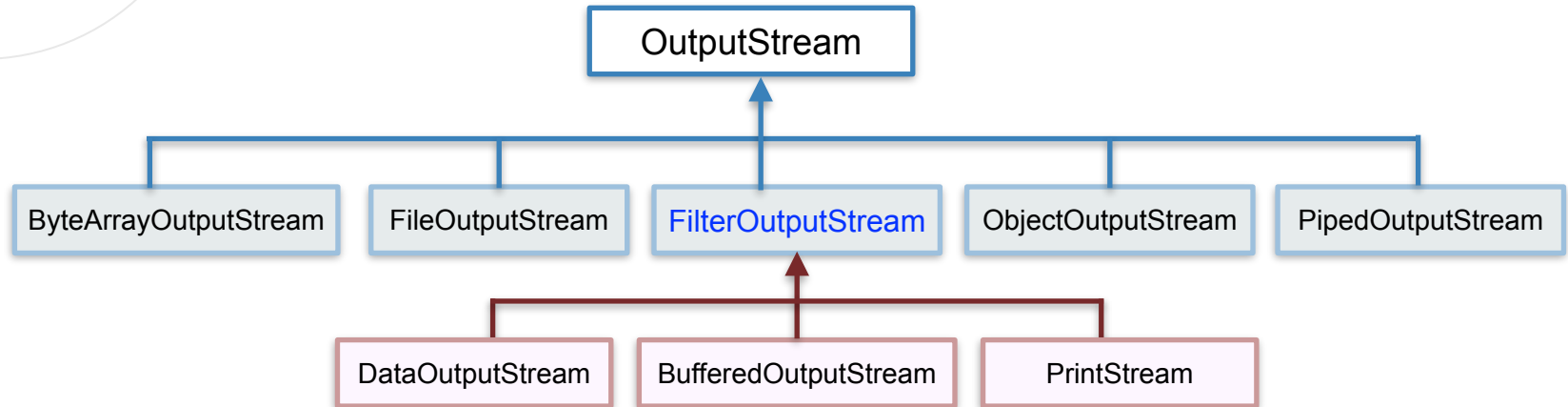
Package java.io

Class OutputStream

```
public abstract class OutputStream
extends Object
implements Closeable, Flushable
```

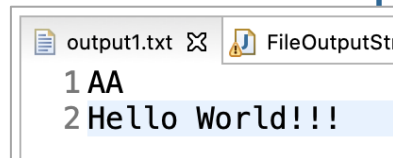
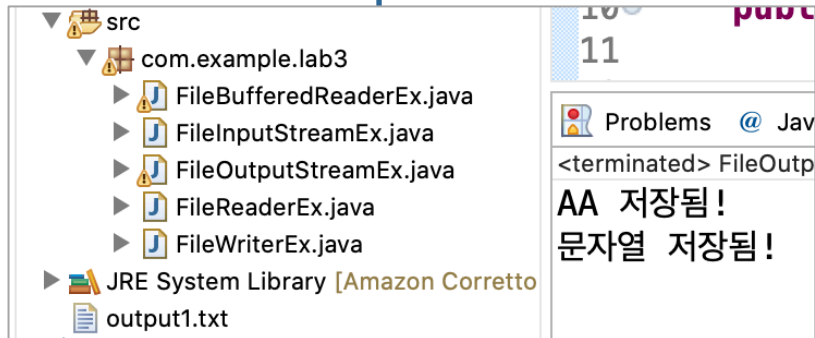
1) OutputStream class

- Direct Known Subclasses
 - ByteArrayOutputStream, FileOutputStream, [FilterOutputStream](#), ObjectOutputStream, PipedOutputStream



예제 1 - FileOutputStreamEx

```
public class FileOutputStreamEx {  
  
    public static void main(String[] args) {  
        try {  
            FileOutputStream fos = new FileOutputStream("output1.txt");  
  
            fos.write('A'); // A의 아스키코드는 65  
            fos.write(65);  
            fos.write('\n');  
            System.out.println("AA 저장됨!");  
  
            String str = "Hello World!!!";  
            fos.write(str.getBytes());  
            System.out.println("문자열 저장됨!");  
  
            fos.close();  
  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



2) InputStream class

- **abstract class** for reading
- the superclass of all classes representing an input stream of bytes
- Methods
 - public **abstract** int read() - 입력스트림에서 데이터를 byte 단위로 읽음, 파일 끝은 -1 리턴
 - public int read(byte[] b)
 - public int read(byte[] b, int off, int Len)
 - public int available() - 현재 입력 스트림에서 읽을 수 있는 byte수
 - public void close()

Module java.base

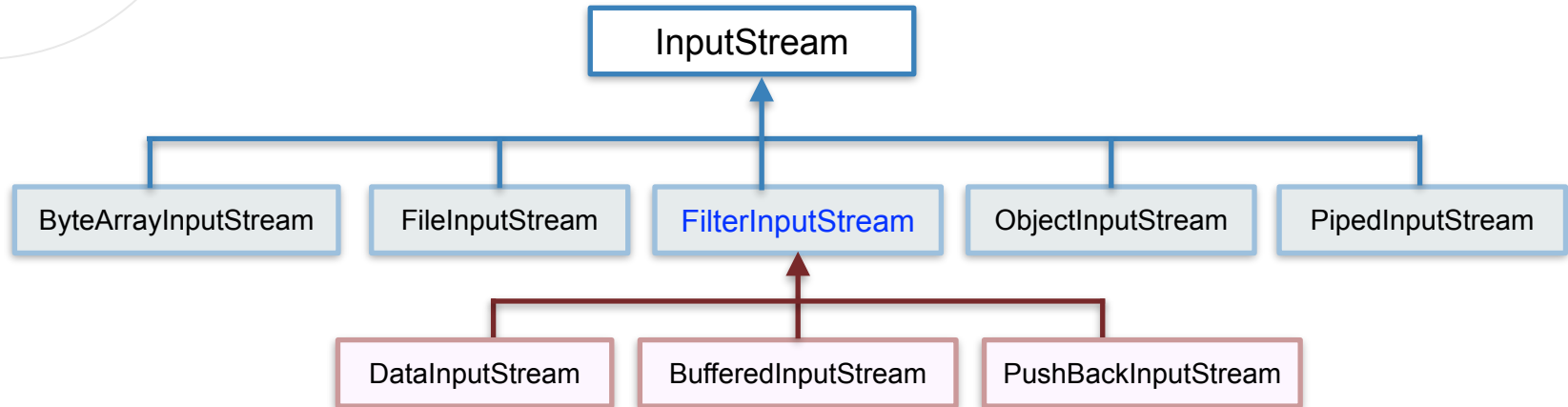
Package java.io

Class InputStream

```
public abstract class InputStream  
extends Object  
implements Closeable
```

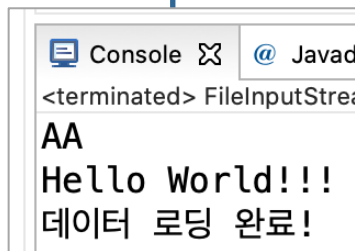
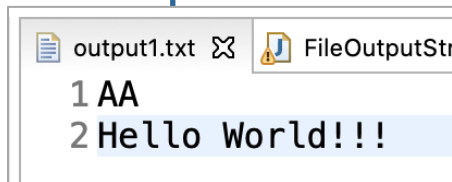
2) InputStream class

- Direct Known Subclasses
 - `AudioInputStream`, `ByteArrayInputStream`, `FileInputStream`, `FilterInputStream`, `ObjectInputStream`, `PipedInputStream`, `SequenceInputStream`, `StringBufferInputStream`



예제 2 - FileInputStreamEx

```
public class FileInputStreamEx {  
  
    public static void main(String[] args) {  
  
        try {  
            FileInputStream fis = new FileInputStream("output1.txt");  
  
            int one;  
            while((one = fis.read()) != -1) {  
                System.out.print((char)one);  
            }  
            System.out.println("\n데이터 로딩 완료!");  
            fis.close();  
  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



3) Writer class

- an **abstract** class for writing to character streams
- Direct Known Subclasses
 - BufferedWriter, CharArrayWriter, FilterWriter, OutputStreamWriter, PipedWriter, PrintWriter, StringWriter
- Methods
 - public Writer append(char c)
 - public Writer append(CharSequence cs)
 - public void write(char[] cbuf)
 - public void write(int c)
 - public void write(String str)
 - **abstract** void write(char[] cbuf, int off, int len)
 - **abstract** void close()

Module java.base

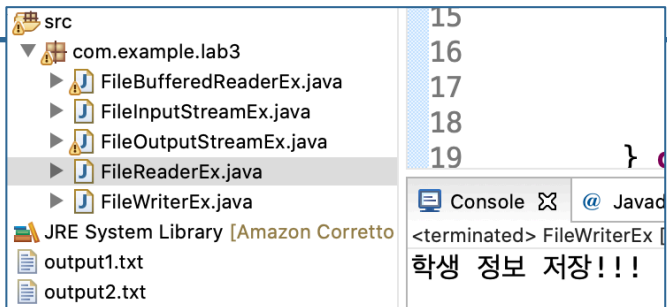
Package java.io

Class Writer

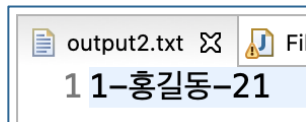
```
public abstract class Writer
extends Object
implements Appendable, Closeable, Flushable
```

예제 3 - FileWriterEx

```
class Student {  
    int no;           // number  
    String name;      // name  
    int age;          // age  
  
    public Student(int no, String name, int age) {  
        super();  
        this.no = no;  
        this.name = name;  
        this.age = age;  
    }  
  
    @Override  
    public String toString() {  
        return no + "-" + name + "-" + age;  
    }  
}
```



```
public class FileWriterEx {  
  
    public static void main(String[] args) {  
        try {  
            Writer w = new FileWriter("output2.txt");  
  
            Student s = new Student(1, "홍길동", 21);  
            w.write(s.toString());  
            w.close();  
  
            System.out.println("학생 정보 저장!!! ");  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



4) Reader class

- an **abstract** class for reading to character streams
- Direct Known Subclasses
 - BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader
- Methods
 - public int read()
 - public int read(char[] cbuf)
 - **abstract** int read(char[] cbuf, int off, int len)
 - **abstract** void close()

Module java.base

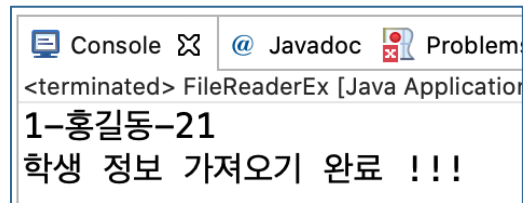
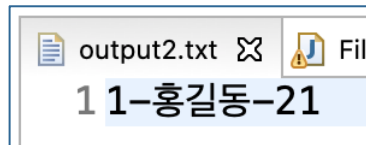
Package java.io

Class Reader

```
public abstract class Reader
extends Object
implements Readable, Closeable
```

예제 4 - FileReaderEx

```
public class FileReaderEx {  
  
    public static void main(String[] args) {  
        try {  
            Reader reader = new FileReader("output2.txt");  
  
            int data ;  
            while ((data = reader.read()) != -1) {  
                System.out.print((char) data);  
            }  
            reader.close();  
  
            System.out.println("\n학생 정보 가져오기 완료 !!! ");  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



5) BufferedReader class

- Scanner class와 유사
- 속도 측면에서는 Scanner 보다 빠름
- 데이터는 문자열로만 읽어올 수 있으므로 데이터 가공(정수, 실수) 필수
- 선언 및 객체화

```
BufferedReader in  
    = new BufferedReader(new FileReader("foo.in"));
```

- 데이터 가공
 - split() 함수사용

```
String[] strarr = s.split(" ");
```

- StringTokenizer class

```
StringTokenizer st = new StringTokenizer(s);  
// StringTokenizer st = new StringTokenizer(s, "-");  
int a = Integer.parseInt(st.nextToken());  
String b = st.nextToken();
```

Module java.base

Package java.io

Class BufferedReader

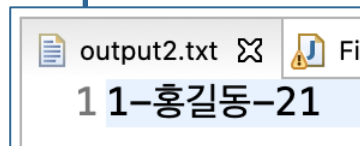
java.lang.Object

java.io.Reader

java.io.BufferedReader

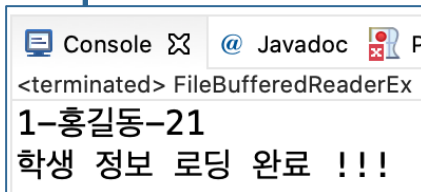
예제 5 - FileBufferedReaderEx

```
public class FileBufferedReaderEx {  
  
    public static void main(String[] args) {  
        try {  
            BufferedReader br = new BufferedReader(new FileReader("output2.txt"));  
  
            String oneline;  
            while((oneline = br.readLine()) != null) {  
                //System.out.println(oneline);  
                StringTokenizer st = new StringTokenizer(oneline, "-");  
                int no = Integer.parseInt(st.nextToken());  
                String name = st.nextToken();  
                int age = Integer.parseInt(st.nextToken());  
  
                Student s = new Student(no, name, age);  
                System.out.println(s.toString());  
            }  
            br.close();  
            System.out.println("학생 정보 로딩 완료 !!! ");  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



output2.txt

1 1-홍길동-21



Console

<terminated> FileBufferedReaderEx [

1-홍길동-21

학생 정보 로딩 완료 !!!