File I/O

Part 1: File Streams and Binary I/O

Chapter 2, Core Java, Volume II &

Chapter 15, Java How to Program

Contents

Files and Streams

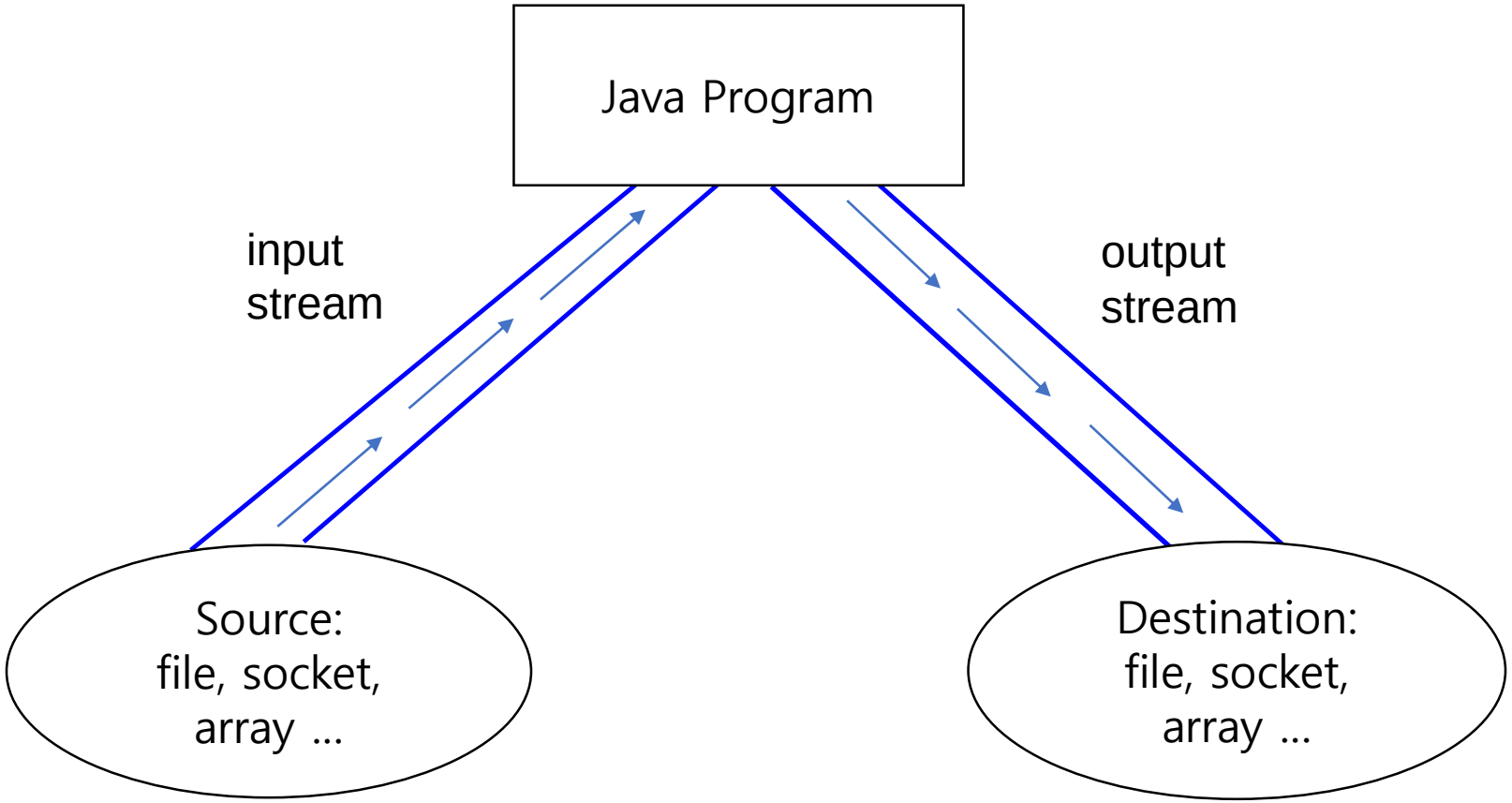
- A java program views each file as a sequential stream of bytes as shown in the following diagram.

- Mechanisms to determine the end of a file is provided by operating system. A Java program simply receives an indication for end of file from the operating system.

- In java, an object from which we read a sequence of bytes is called an input stream.
- In java, an object to which we write a sequence of bytes is called an output stream.
- Files, network connection, and block of memory can be sources and destinations of byte sequences.

Files and Streams

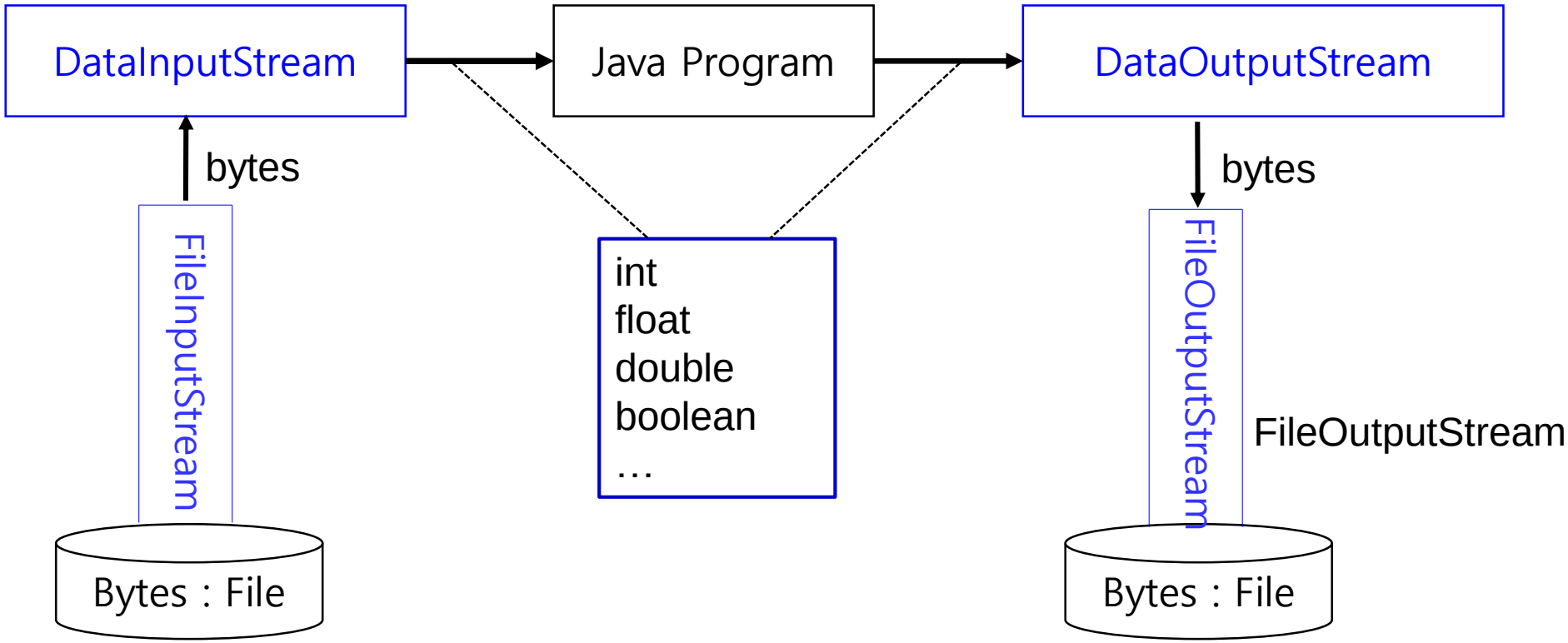
- Conceptual Streams



Data flows in a stream from source to java program and from program to destination.

Files and Streams

- Sample Streams



DataInputsteam(high-level steam) is attached to FileInpusteam(low-level stream), which in turn is attached to the file.

Java API for Input and Output Streams

- Byte-based streams output and input data in its binary format
  - to read and write binary files
- Character-based streams output and input data as a sequence of characters in which every character is two bytes.
  - to read and write text files which can be read in editors

Java API for Input and Output Streams

- Class Hierarchy of Byte-based Input Streams (java.io)

java.lang.Object
- java.io.*InputStream* // abstract base class
  - java.io.ByteArrayInputStream
  - java.io.FileInputStream
  - java.io.ObjectInputStream
  - java.io.StringBufferInputstream
  - java.io.FilterInputStream
    - java.io.BuffedredInputStream
    - java.io.DataInputStream

Java API for Input and Output Streams

- Class Hierarchy of Byte-based Output Streams (java.io)

java.lang.Object
- java.io.*OutputStream* // abstract base class
  - java.io.ByteArrayOutputStream
  - java.io.FileOutputStream
  - java.io.ObjectOutputStream
  - java.io.FilterOutputStream
    - java.io.BuffedredOutputStream
    - java.io.DataOutputStream
    - java.io.PrintStream

Java API for Input and Output Streams

- Class Hierarchy of Character-based Input Streams (java.io)

java.lang.Object
    java.io.*Reader* // abstract base class
        java.io.BufferedReader
        java.io.CharArrayReader
        java.io.StringReader
        java.io.InputStreamReader
            java.io.FileReader

- Note: An InputStreamReader is a bridge from byte streams to character streams.

Java API for Input and Output Streams

- Class Hierarchy of Character-based Output Streams (java.io)

Java.lang.Object
- java.io.*Writer* // abstract base class
  - java.io.BufferedWriter
  - java.io.CharArrayWriter
  - java.io.StringWriter
  - jva.io.PrintWriter
  - java.io.OutputStreamWriter
    - java.io.FileWriter

- Note: An OutputStreamWriter is a bridge from character streams to bytes streams.

Obtaining Streams

- It is easy to use static methods from the java.nio.file.Files class:

```
Path path = Paths.get(filenameString);
InputStream in = Files.newInputStream(path);
OutputStream out = Files.newOutputStream(path);
```

```
≈ Path.of(filenameString);
```

```
≈ InputStream in = new FileInputStream(filenameString);
≈ OutputStream out = new FileOutputStream(filenameString);
```

- Get an input stream from any URL:

```
URL url = new URL("http://horstmann.com/index.html");
InputStream in = url.openStream();
```

- Get an input stream from a byte[] array:

```
byte[] bytes =.....;
InputStream in = new ByteArrayInputStream(bytes);
```

- Conversely, you can write to a ByteArrayOutputStream and then collect the bytes:

```
ByteArrayOutputStream out = new ByteArrayOutputStream();
// write to out here
byte[] bytes = out.toByteArray();
```

Binary Input/Output

- The read method returns a single byte (as an int) or -1 at the end of input:

```
Path path = Paths.get(filenameString);
InputStream in = Files.newInputStream(path);      // you have to declare or catch IOException
int b = in.read();
if (b != -1)
{
    byte value = ( byte) b;
     .....;
 }
in.close();
```

- It is more common to read bytes in bulk:

```
byte[] bytes = ..................;
int len = in.read(bytes);
```

- Reading all bytes from files:

```
byte[] bytes = Files.readAllBytes(path);
```

Binary Input/Output

- You can write one byte or bytes from an array:

```
Path path = Paths.get(filenameString);
OutputStream out = Files.newOutputStream(path);
int b = ..... ;
out.write(b);
byte[] bytes = ..... ;
out.write(bytes);
out.write(bytes, start, length);
```

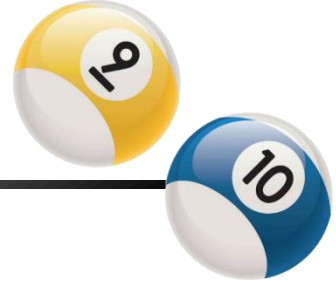
- When writing to a stream, close it when you are done:

```
out.close();
```

- Or better, use a try-with-resources block:

```
try (OutputStream out = ..........................)
{
   out.write(bytes);
}
```

Binary Input/Output



- To read all bytes from an input stream:

```
byte[] bytes = url.openStream().readAllBytes();
```

  - There is also `readNBytes()`.

- To transfer all bytes from an input stream to an output stream:

```
InputStream in = ...
OutputStream out = ...
in.transferTo(out);
```