

### Java Io

- 1 [Java IO Tutorial](#)
- 2 **Java IO Overview**
- 3 [Java IO: Files](#)
- 4 [Java IO: Pipes](#)
- 5 [Java IO: Networking](#)
- 6 [Java IO: Byte & Char Arrays](#)
- 7 [Java IO: System.in, System.out, and System.error](#)
- 8 [Java IO: Streams](#)
- 9 [Java IO: Input Parsing](#)
- 10 [Java IO: Readers and Writers](#)
- 11 [Java IO: Concurrent IO](#)
- 12 [Java IO: Exception Handling](#)
- 13 [Java IO: InputStream](#)
- 14 [Java IO: OutputStream](#)
- 15 [Java IO: FileInputStream](#)
- 16 [Java IO: FileOutputStream](#)
- 17 [Java IO: RandomAccessFile](#)
- 18 [Java IO: File](#)
- 19 [Java IO: PipedInputStream](#)
- 20 [Java IO: PipedOutputStream](#)
- 21 [Java IO: ByteArrayInputStream](#)
- 22 [Java IO: ByteArrayOutputStream](#)
- 23 [Java IO: FilterInputStream](#)
- 24 [Java IO: FilterOutputStream](#)
- 25 [Java IO: BufferedInputStream](#)
- 26 [Java IO: BufferedOutputStream](#)
- 27 [Java IO: PushbackInputStream](#)
- 28 [Java IO: SequenceInputStream](#)
- 29 [Java IO: DataInputStream](#)
- 30 [Java IO: DataOutputStream](#)
- 31 [Java IO: PrintStream](#)
- 32 [Java IO: ObjectInputStream](#)
- 33 [Java IO: ObjectOutputStream](#)
- 34 [Java IO: Serializable](#)
- 35 [Java IO: Reader](#)
- 36 [Java IO: Writer](#)
- 37 [Java IO: InputStreamReader](#)

# Java IO Overview



By Jakob Jenkov

Connect with me:



Rate article:



Share article:



### Table of Contents

- [Input and Output - Source and Destination](#)
- [Java IO Purposes and Features](#)
- [Java IO Class Overview Table](#)

In this text I will try to give you an overview of the classes in the Java IO (`java.io`) package. More specifically, I will try to group the classes after their purpose. This grouping should make it easier for you in the future, to determine the purpose of a class, or find the class you need for a specific purpose.

## Input and Output - Source and Destination

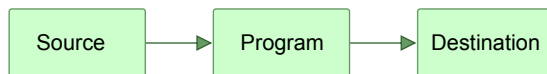
The terms "input" and "output" can sometimes be a bit confusing. The input of one part of an application is often the output of another. Is an `OutputStream` a stream where output is written to, or output comes out from (for you to read)? After all, an `InputStream` outputs its data to the reading program, doesn't it? Personally, I found this a bit confusing back in the day when I first started out learning about Java IO.

In an attempt to clear out this possible confusion, I have tried to put some different names on input and output to try to link them conceptually to where the input comes from, and where the output goes.

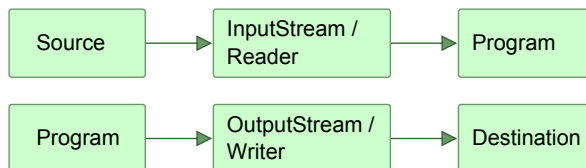
Java's IO package mostly concerns itself with the reading of raw data from a source and writing of raw data to a destination. The most typical sources and destinations of data are these:

- Files
- Pipes
- Network Connections
- In-memory Buffers (e.g. arrays)
- `System.in`, `System.out`, `System.error`

The diagram below illustrates the principle of a program reading data from a source and writing it to some destination:



A program that needs to read data from some source needs an input stream or Reader. A program that needs to write data to some destination needs an output stream or writer. This is also illustrated in the diagram below:



An `InputStream` or `Reader` is linked to a **source** of data. An `OutputStream` or `Writer` is linked to a **destination** of data.

## Java IO Purposes and Features

The Java IO classes, which mostly consists of streams and readers / writers, are addressing various purposes.

- 37 [Java IO: InputStreamReader](#)
- 38 [Java IO: OutputStreamWriter](#)
- 39 [Java IO: FileReader](#)
- 40 [Java IO: FileWriter](#)
- 41 [Java IO: PipedReader](#)
- 42 [Java IO: PipedWriter](#)
- 43 [Java IO: CharArrayReader](#)
- 44 [Java IO: CharArrayWriter](#)
- 45 [Java IO: BufferedReader](#)
- 46 [Java IO: BufferedWriter](#)
- 47 [Java IO: FilterReader](#)
- 48 [Java IO: FilterWriter](#)
- 49 [Java IO: PushbackReader](#)
- 50 [Java IO: LineNumberReader](#)
- 51 [Java IO: StreamTokenizer](#)
- 52 [Java IO: PrintWriter](#)
- 53 [Java IO: StringReader](#)
- 54 [Java IO: StringWriter](#)

The data is abstract, which mostly consists of streams and readers / writers, are addressing various purposes. That is why there are so many different classes. The purposes addressed are summarized below:

- File Access
- Network Access
- Internal Memory Buffer Access
- Inter-Thread Communication (Pipes)
- Buffering
- Filtering
- Parsing
- Reading and Writing Text (Readers / Writers)
- Reading and Writing Primitive Data (long, int etc.)
- Reading and Writing Objects

These purposes are nice to know about when reading through the Java IO classes. They make it somewhat easier to understand what the classes are targeting.

## Java IO Class Overview Table

Having discussed sources, destinations, input, output and the various IO purposes targeted by the Java IO classes, this text will finish off with a table of most (if not all) Java IO classes divided by input, output, being byte based or character based, and any more specific purpose they may be addressing, like buffering, parsing etc.

	Byte Based		Character Based	
	Input	Output	Input	Output
Basic	InputStream	OutputStream	Reader InputStreamReader	Writer OutputStreamWriter
Arrays	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
Files	FileInputStream RandomAccessFile	FileOutputStream RandomAccessFile	FileReader	FileWriter
Pipes	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
Buffering	BufferedReader	BufferedOutputStream	BufferedReader	BufferedWriter
Filtering	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
Parsing	PushbackInputStream StreamTokenizer		PushbackReader LineNumberReader	
Strings			StringReader	StringWriter
Data	DataInputStream	DataOutputStream		
Data - Formatted		PrintStream		PrintWriter
Objects	ObjectInputStream	ObjectOutputStream		
Utilities	SequenceInputStream			


**Next:** [Java IO: Files](#)

**Get all my free tips & tutorials!**

Connect with me, or sign up for my news letter or [RSS feed](#), and get all my tips that help you become a more skilled and efficient developer.



**Newsletter**

**Connect with me:** 

**Newsletter - Get all my free tips!**