CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

Stream Basics

Overview

1. Introduction
2. Types of Streams in Java
3. Input Stream
4. Output Stream
5. Streams in Java
6. System Streams
7. Read data from Source into Java Program
8. Write data to Destination from Java Program

Introduction

Stream in Java represents sequential flow (or unbroken flow) of data from one place to another place.

In other words, a stream is a path along which data flows (like a pipe along which water flows).

It is required to accept data as input from the keyboard. The data in the form of stream may be bytes, characters, objects, etc.

Introduction

Let’s understand it with the help of a real-time example.

A river is a stream of water that flows from one place (i.e. source) to another place (i.e. destination) in a continuous sequence. Source and destination are connected through the continuous flow of water.

Similarly, a stream carries data from one place (source) to another place (destination).

Introduction

In Java I/O, data flows from a source known as data source to a destination known as data sink as shown in the below figure. Data source and data sink are connected through a Java program.

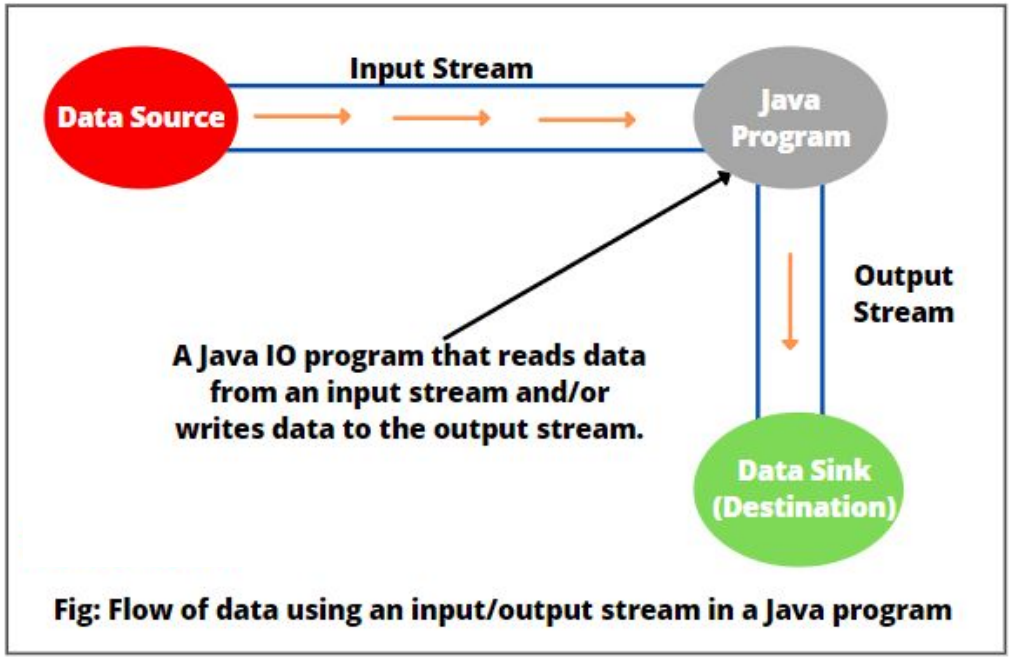

Fig: Flow of data using an input/output stream in a Java program

A stream is always required if we want to move data from data source to data sink. For example, a stream can carry data from:

- keyboard to memory
- memory to printer
- memory to a text file

Types of Stream in Java

Basically, there are two types of streams in Java. They are as follows:

- Input stream
- Output stream

Input Stream

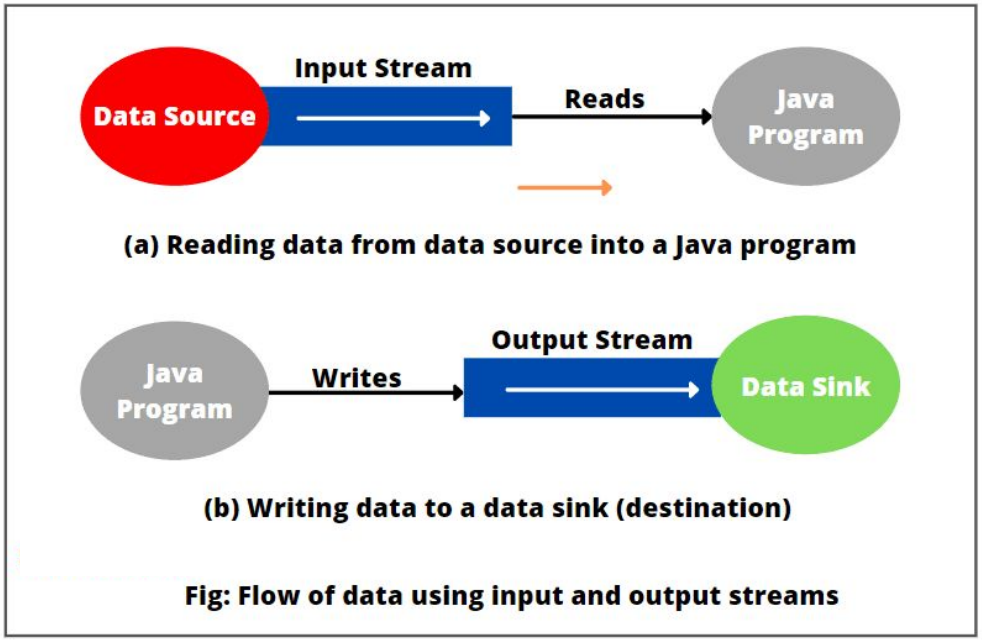

Fig: Flow of data using input and output streams

A stream that receives or reads data from a data source and sends it to a Java program is called **input stream**. Input represents the flow of data into a program.

The input stream connects the data source and a Java program, as shown in the figure.

As you can see in the figure, the data is read from a data source and flow sequentially to a Java program.

Output Stream

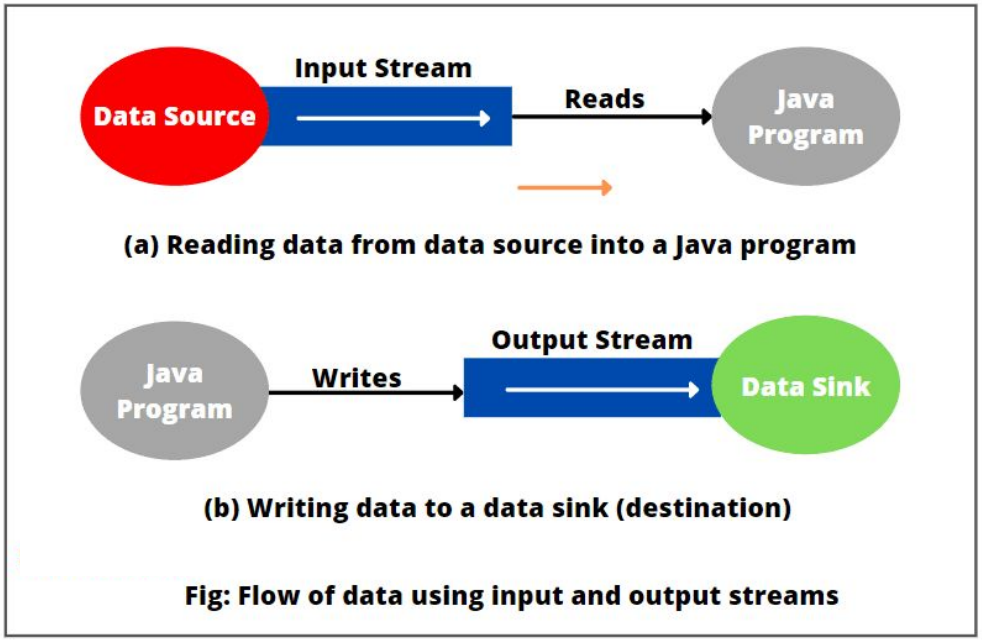

Fig: Flow of data using input and output streams

A stream that takes data from a Java program and sends or writes data to the destination (data sink) is called **output stream**. A output represents the flow of data out of a program.

Output stream connects java program and a data sink. A Java program writes or sends data to the data sink.

In the figure, the data flow from a Java program to a data sink via an output stream.

Streams in Java

Note: Java represents all streams by classes defined in `java.io` (input and output) package. This package contains a lot of classes, all of which are divided into two basic categories: input streams and output streams.

`System` Streams

Java represents a keyboard by a field, called "in" in `System` class. `System.in` represents a standard input device, i.e. keyboard, by default. The `System` class is found in `java.lang` (language) package.

`System` contains three fields that represent some type of stream:

a) `System.in`:
This field represents an `InputStream` object, which is a standard input device, known as a keyboard by default.

b) `System.out`:
This field represents a `PrintStream` object, which represents a standard output device, known as a monitor by default. Normally, we use it to display normal messages.

c) `System.err`:
It also represents `PrintStream` object, which by default represents a monitor. Normally, we use it to display error messages.

We use both `System.out` and `System.err` to represent the monitor. Hence, we can use either to send data to a monitor.

Read data from Source into Java Program

To read data from data source into a Java program, we need to perform the following steps:

1. Identify the data source. It may be a keyboard, file, array, string, network connection, etc.

2. Create an input stream by using the data source identified.

3. Read data from the input stream. Basically, we read data in a loop as long as we have received all data from input stream.

4. Finally, close the input stream when reading data has finished. From Java 7 onwards, we can use the `try-with-resources` block that closes the input stream automatically.

Write Data to Destination from Java Program

To write data to a destination (data sink) from the Java program, perform the following steps:

1. Identify the destination where data will be written. A destination may be a monitor, a file, array, string, network connection, etc.

2. Create an output stream using data sink that we have identified.

3. Now, write data to the output stream.

4. Close the output stream once writing data is completed. From Java 7 onwards, we can use a `try-with-resources` block to close the output stream automatically.

END