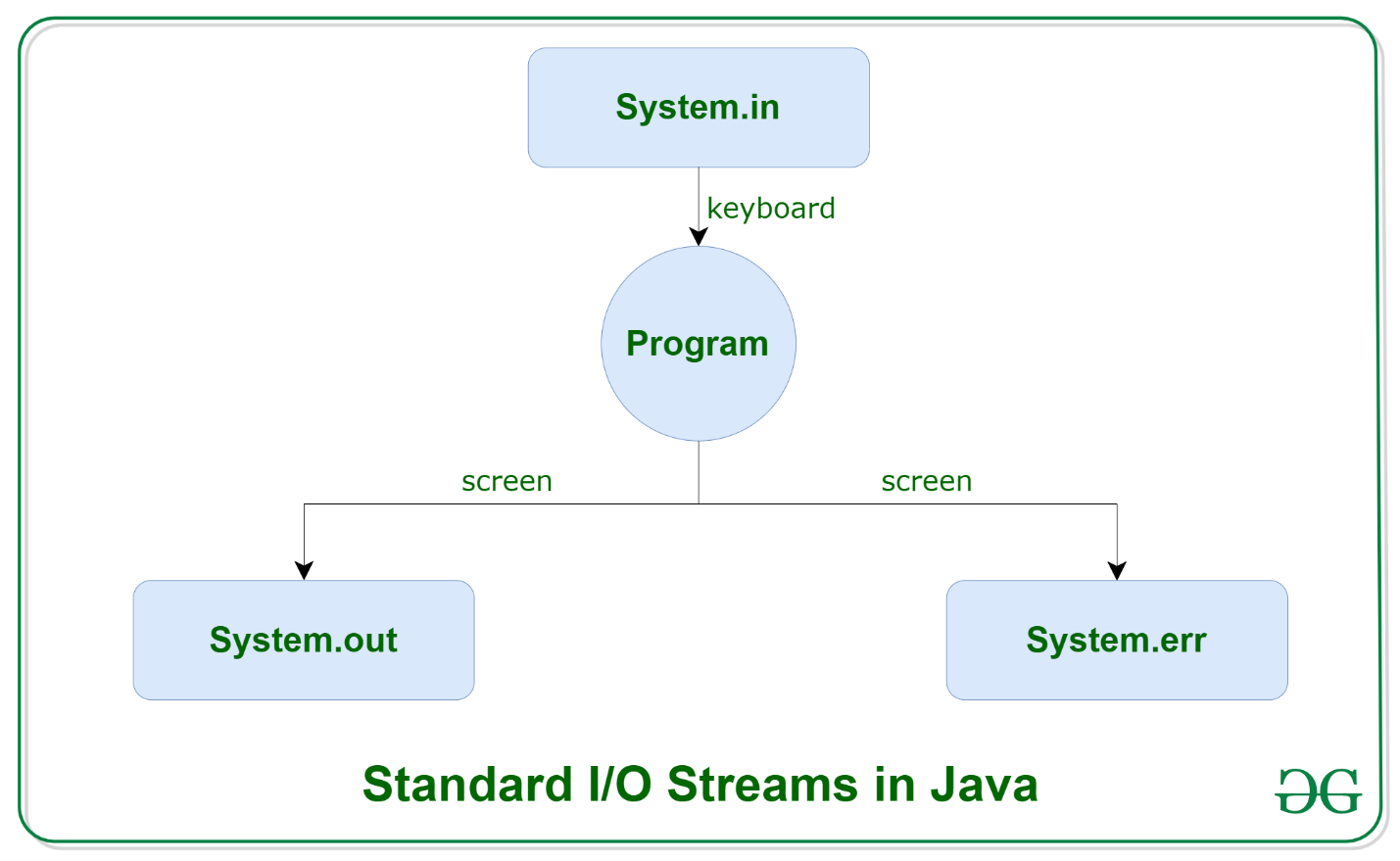
## Java IO : Input-output in Java with Examples

[Java](https://www.geeksforgeeks.org/java/) brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.



Before exploring various input and output streams lets look at **3 standard or default streams** that Java has to provide which are also most common in use:

[](https://media.geeksforgeeks.org/wp-content/uploads/20191127113736/Java-Basic-input-output1.png)

1. [**System.in**](https://www.geeksforgeeks.org/java-lang-system-class-java/)**:** This is the **standard input stream** that is used to read characters from the keyboard or any other standard input device.
2. [**System.out**](https://www.geeksforgeeks.org/java-lang-system-class-java/)**:** This is the **standard output stream** that is used to produce the result of a program on an output device like the computer screen.

Here is a list of the various print functions that we use to output statements:

[**print()**](https://www.geeksforgeeks.org/difference-between-print-and-println-in-java/)**:** This method in Java is used to display a text on the console. This text is passed as the parameter to this method in the form of String. This method prints the text on the console and the cursor remains at the end of the text at the console. The next printing takes place from just here.  
**Syntax:**

System.out.print(*parameter*);

[**println()**](https://www.geeksforgeeks.org/difference-between-print-and-println-in-java/)**:** This method in Java is also used to display a text on the console. It prints the text on the console and the cursor moves to the start of the next line at the console. The next printing takes place from the next line.  
**Syntax:**

System.out.println(*parameter*);

[**printf()**](https://www.geeksforgeeks.org/formatted-output-in-java/)**:** This is the easiest of all methods as this is similar to printf in C. Note that System.out.print() and System.out.println() take a single argument, but printf() may take multiple arguments. This is used to format the output in Java.

 // this will print it upto

        // 2 decimal places

        System.out.printf(

            "Formatted with"

                + " precison: PI = %.2f\n",

            Math.PI);

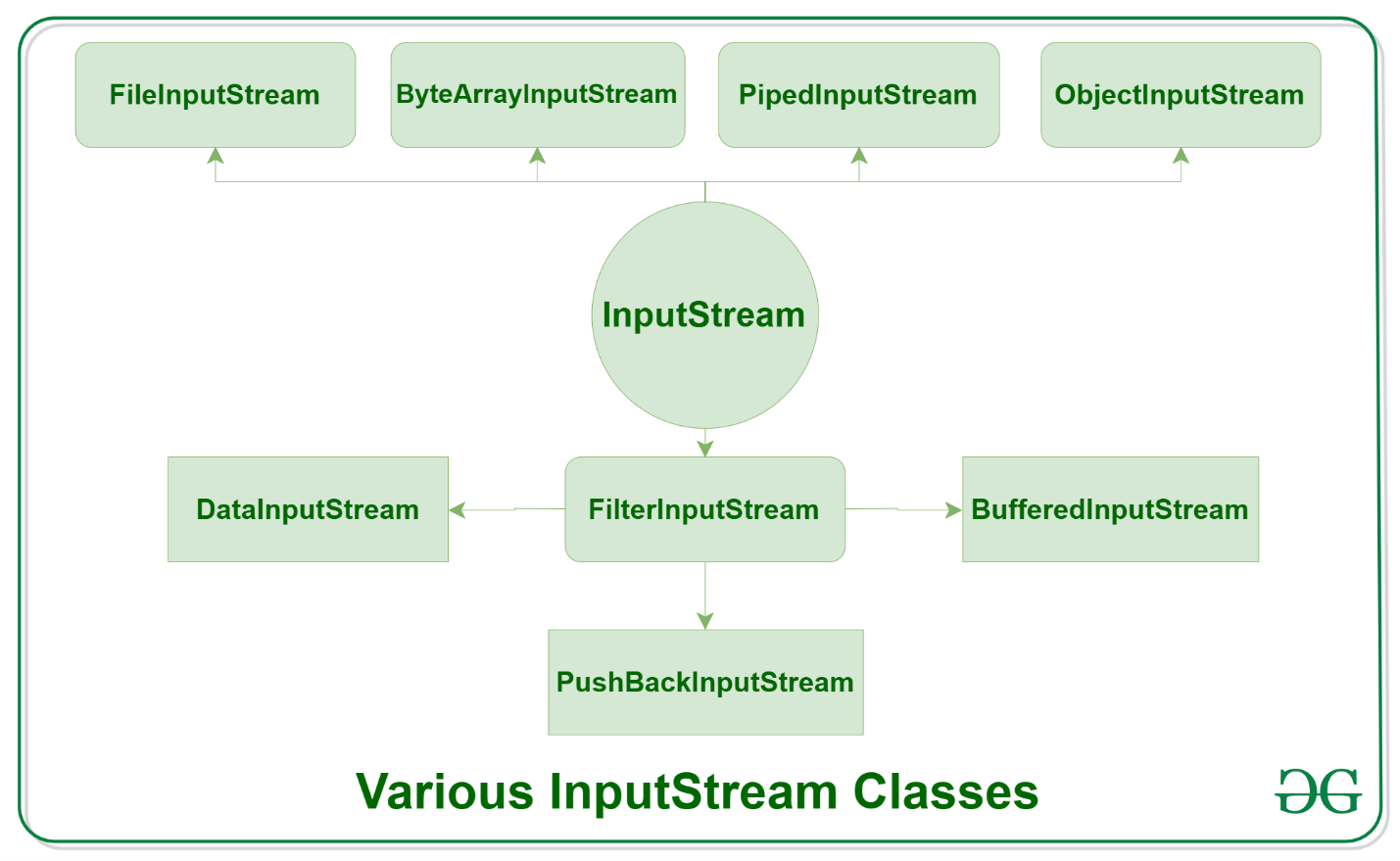
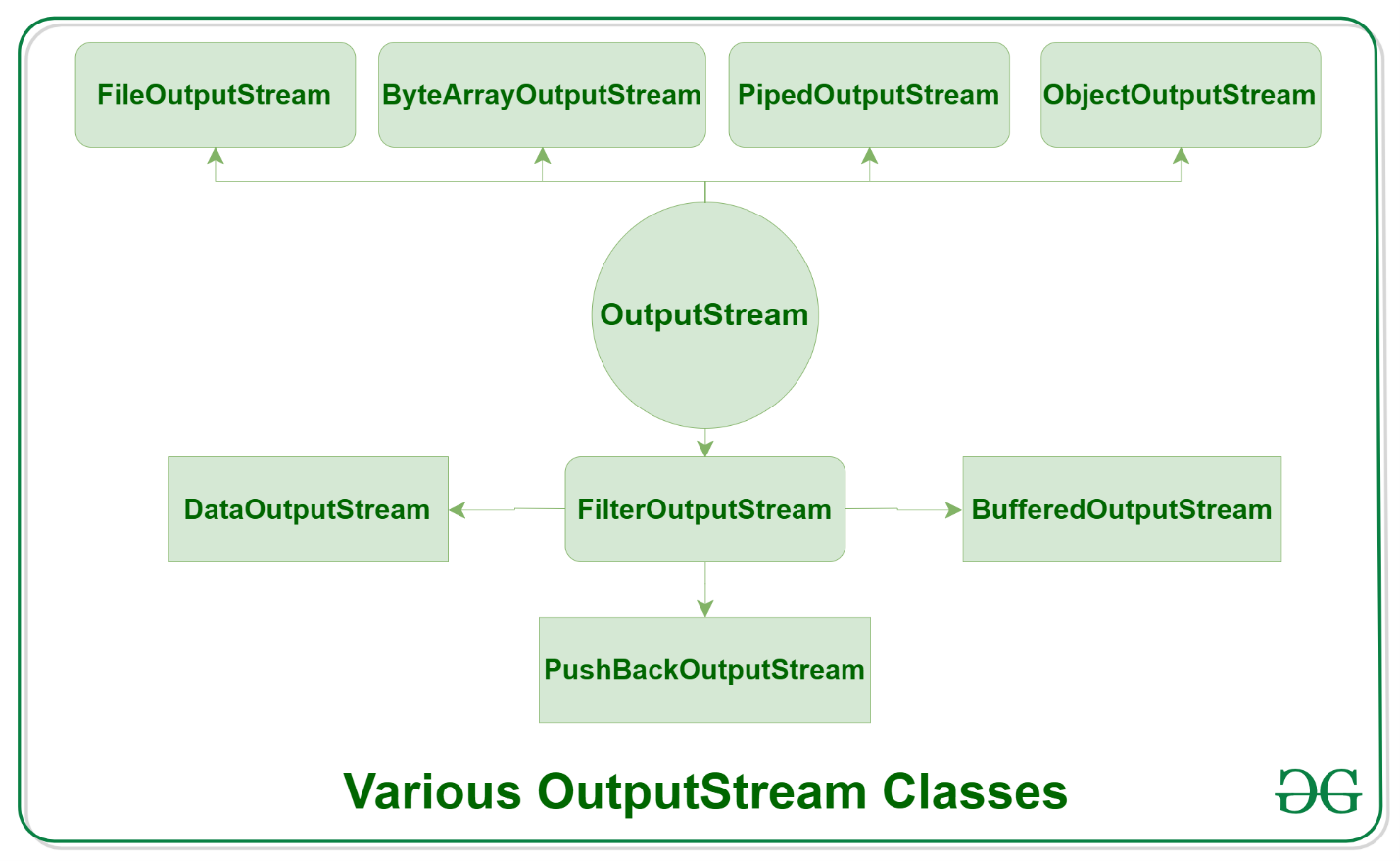
1. [**System.err**](https://www.geeksforgeeks.org/java-lang-system-class-java/)**:** This is the **standard error stream** that is used to output all the error data that a program might throw, on a computer screen or any standard output device.

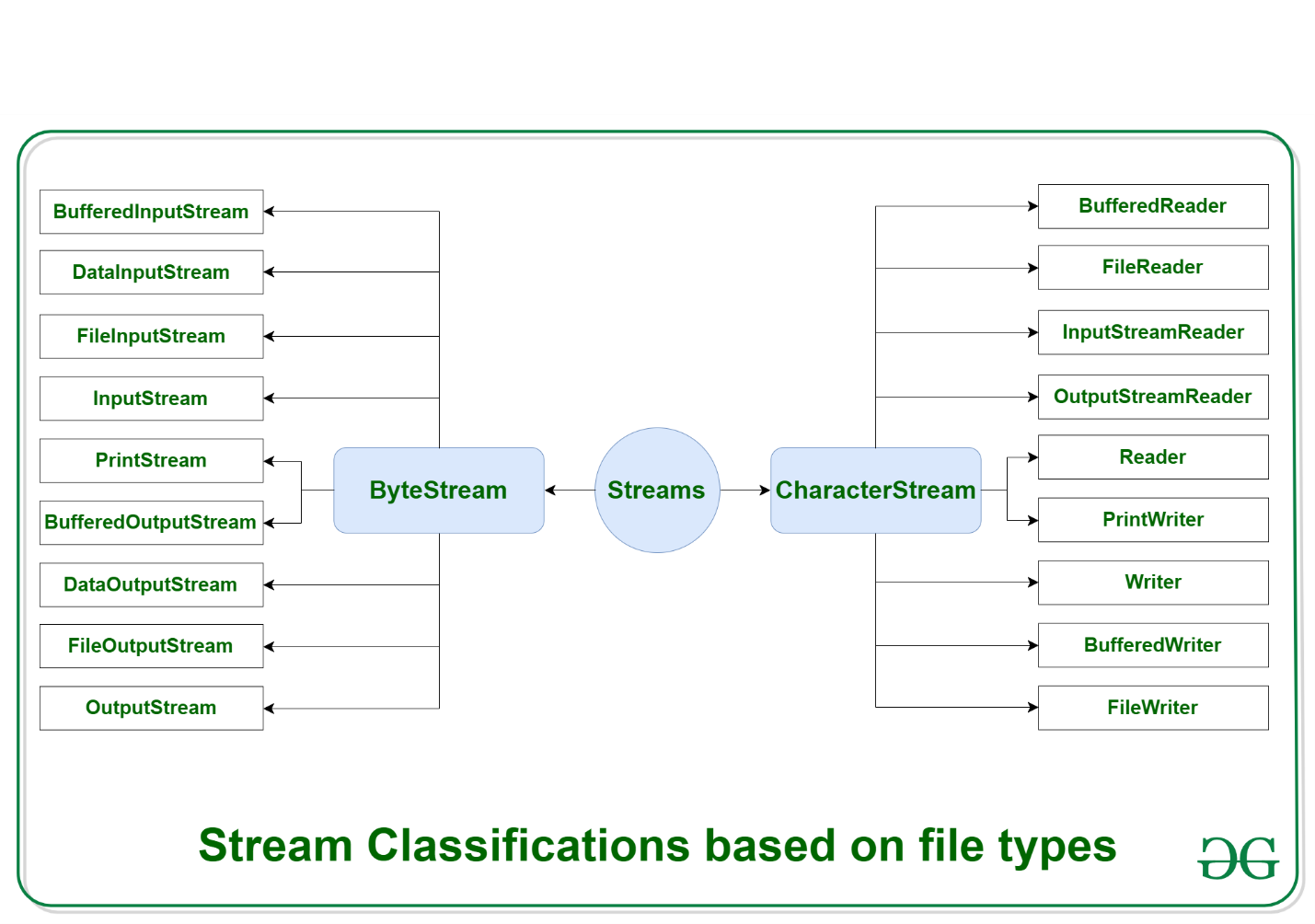
This stream also uses all the 3 above-mentioned functions to output the error data:

* print()
* println()
* printf()

## Types of Streams:

Java IO

* **Depending on the type of operations**, streams can be divided into two primary classes:
  1. [**Input Stream:**](https://www.geeksforgeeks.org/java-io-inputstream-class-in-java/) These streams are used to read data that must be taken as an input from a source array or file or any peripheral device. For eg., FileInputStream, BufferedInputStream, ByteArrayInputStream etc.  
     [](https://media.geeksforgeeks.org/wp-content/uploads/20191126132719/Java-Input-Stream.png)
  2. [**Output Stream:**](https://www.geeksforgeeks.org/java-io-outputstream-class-java/) These streams are used to write data as outputs into an array or file or any output peripheral device. For eg., FileOutputStream, BufferedOutputStream, ByteArrayOutputStream etc.  
     [](https://media.geeksforgeeks.org/wp-content/uploads/20191126133237/Java-Output-Stream.png)
  3. **Depending on the types of file**, Streams can be divided into two primary classes which can be further divided into other classes as can be seen through the diagram below followed by the explanations.

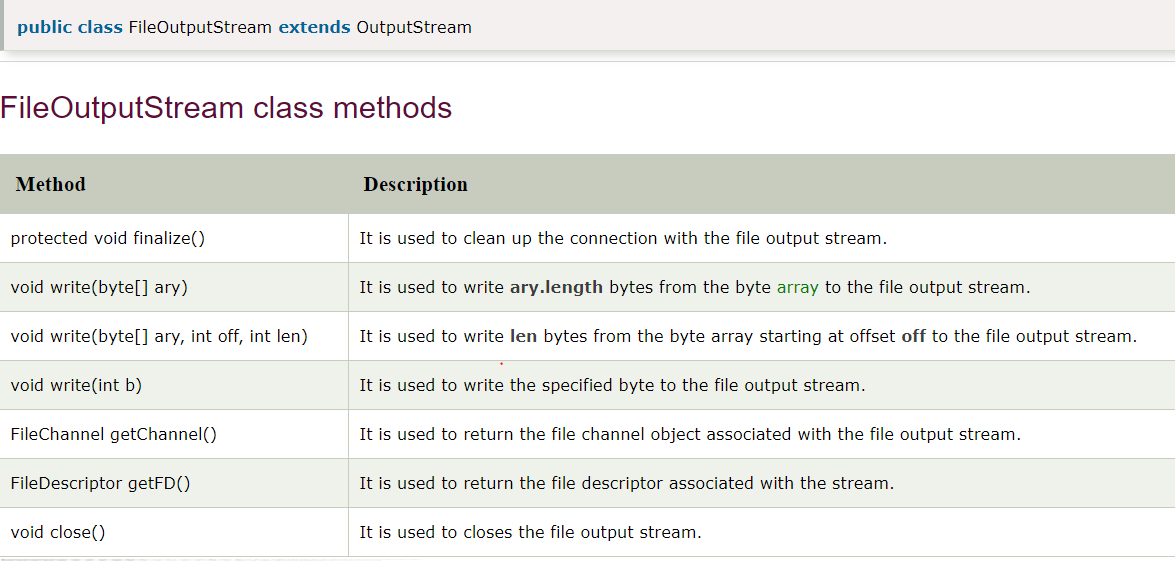
[](https://media.geeksforgeeks.org/wp-content/uploads/20191127121553/Java-stream-classification-filetype2.png)

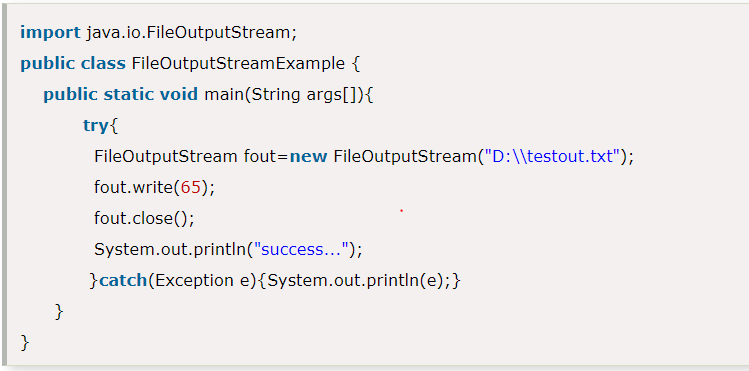
* 1. **ByteStream:** This is used to process data byte by byte (8 bits). Though it has many classes, the FileInputStream and the FileOutputStream are the most popular ones. The FileInputStream is used to read from the source and FileOutputStream is used to write to the destination.
  2. **CharacterStream:** In Java, characters are stored using Unicode conventions (Refer this for details). Character stream automatically allows us to read/write data character by character. Though it has many classes, the FileReader and the FileWriter are the most popular ones. FileReader and FileWriter are character streams used to read from the source and write to the destination respectively.

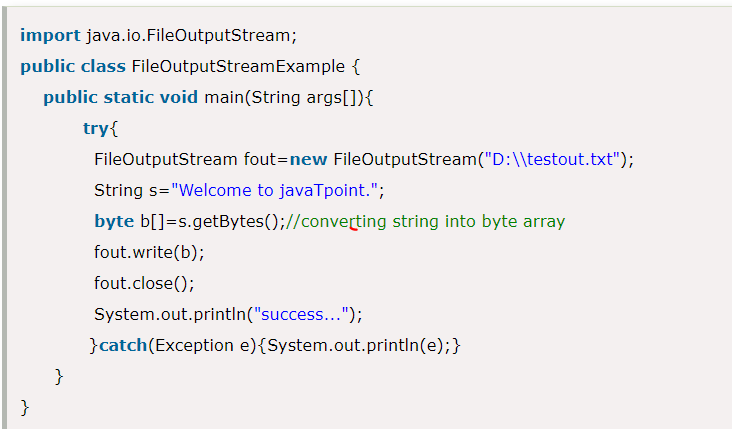
<https://www.javatpoint.com/java-fileinputstream-class> -- we can find all types of steams with example.

## FileOutputStream Class

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use [FileWriter](https://www.javatpoint.com/java-filterwriter-class) than FileOutputStream.

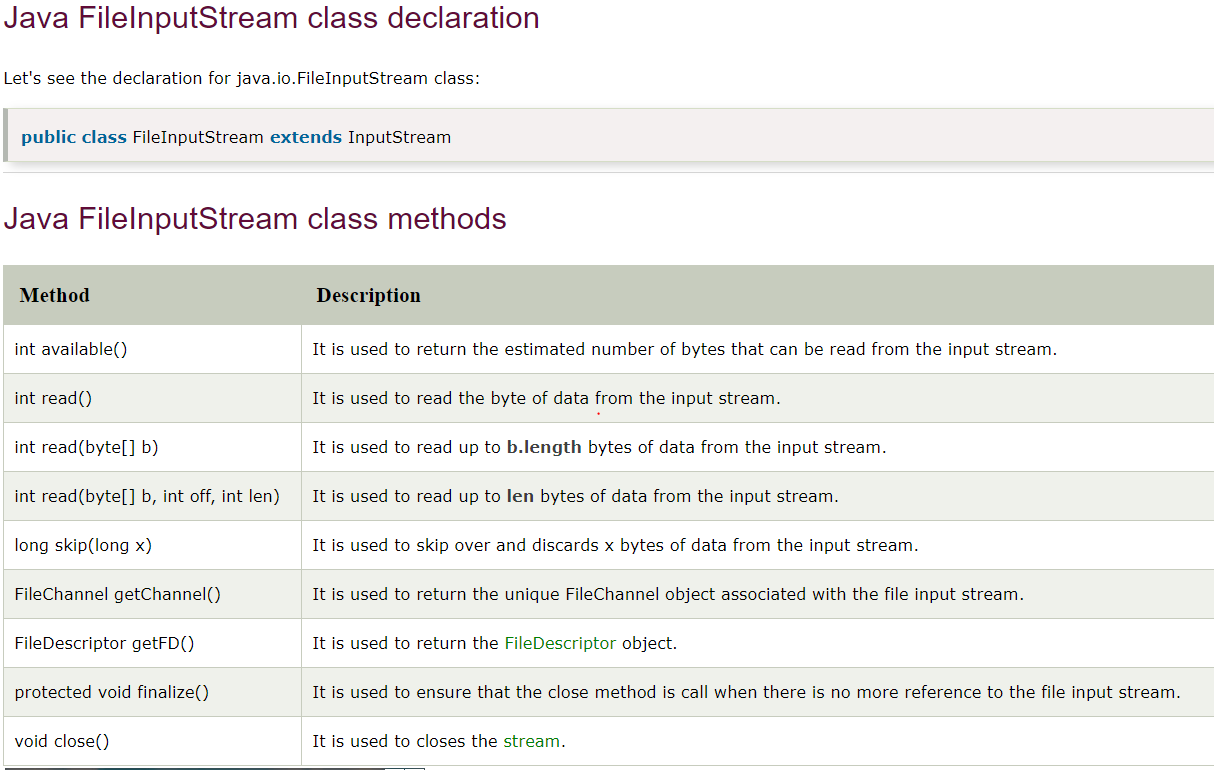


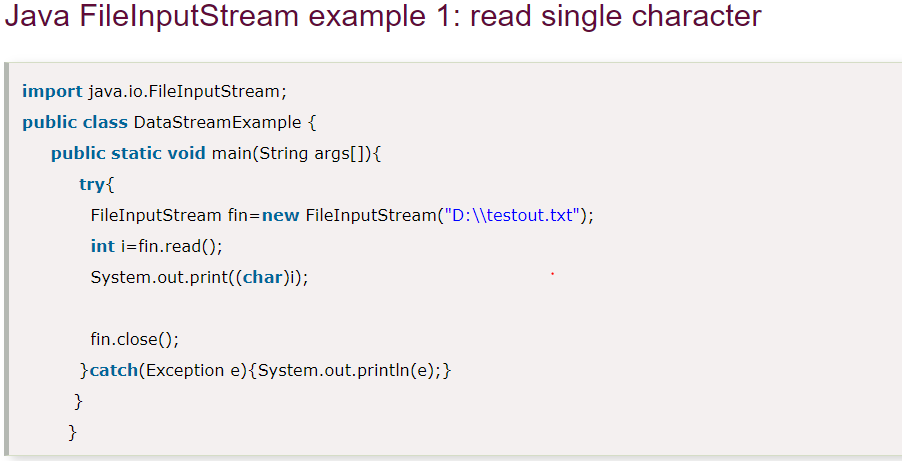


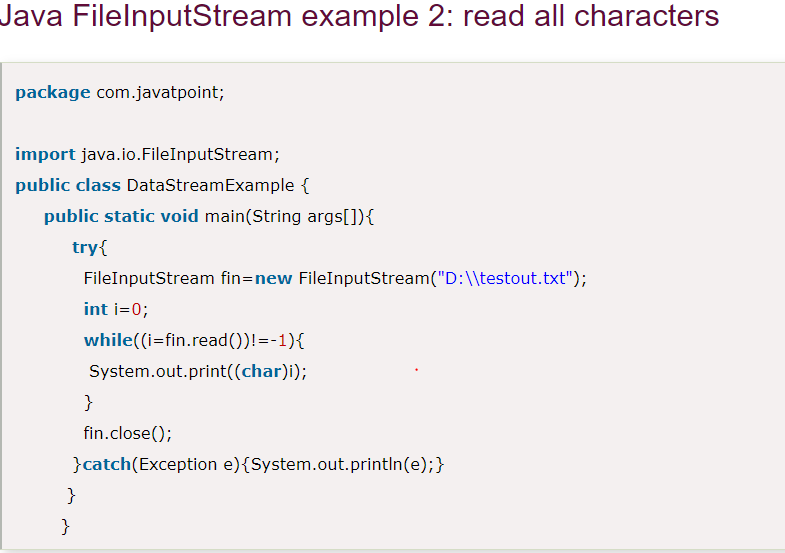


## Java FileInputStream Class

Java FileInputStream class obtains input bytes from a [file](https://www.javatpoint.com/java-file-class). It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use [FileReader](https://www.javatpoint.com/java-filereader-class) class.





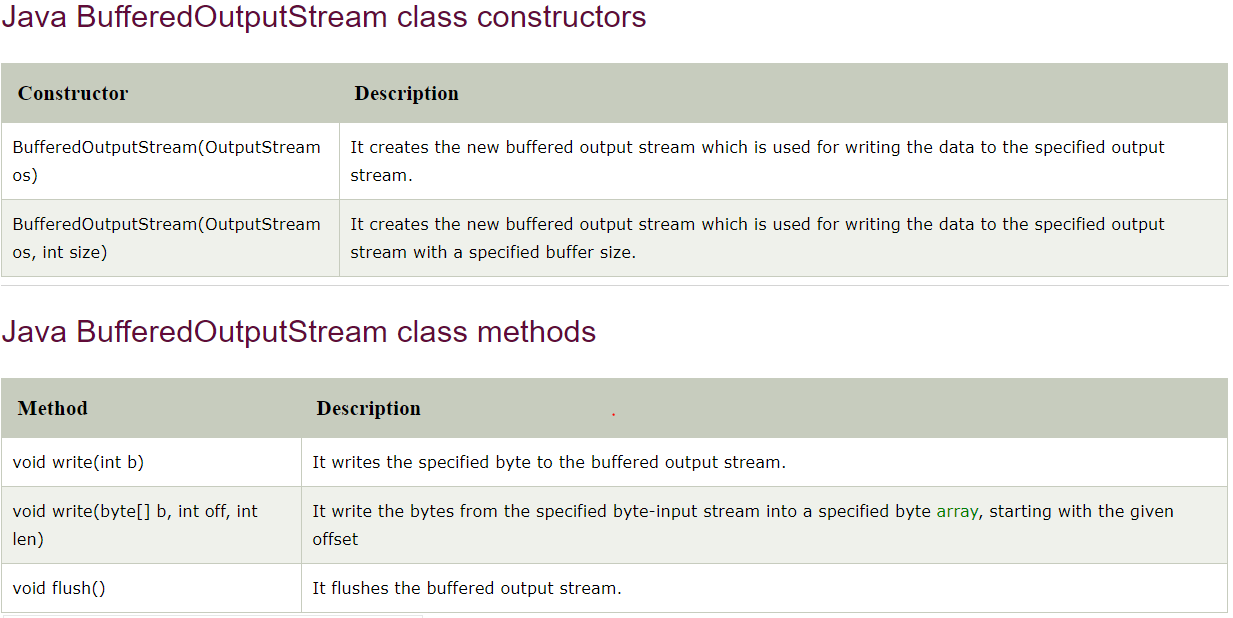


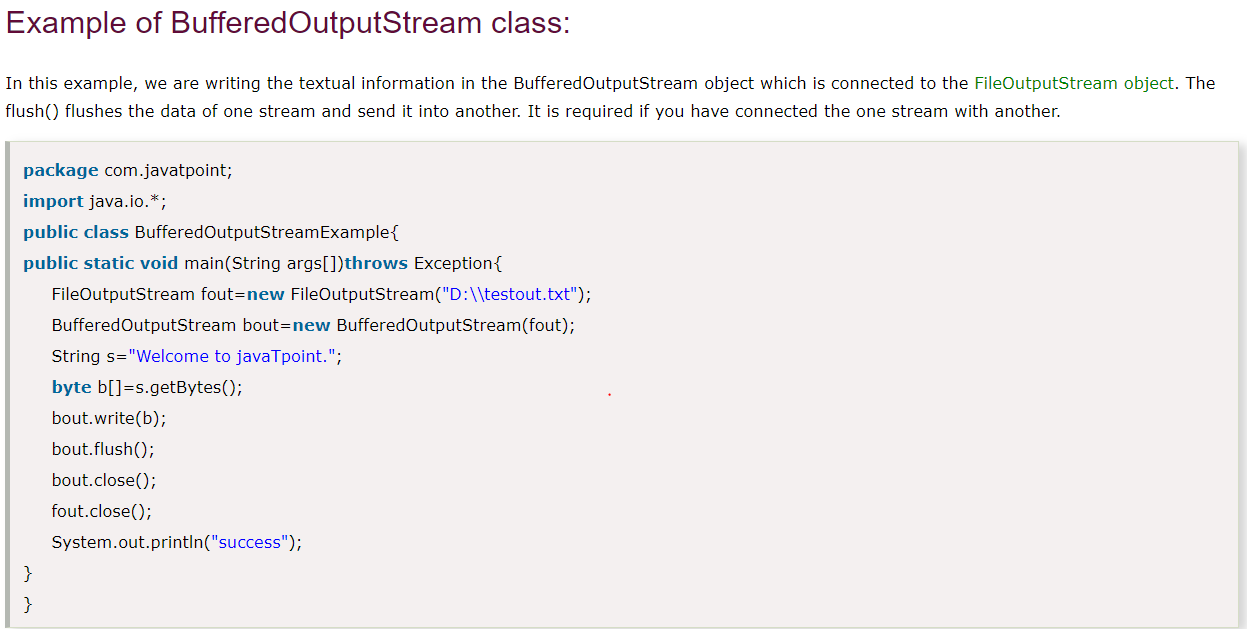
## Java BufferedOutputStream Class

Java BufferedOutputStream [class](https://www.javatpoint.com/object-and-class-in-java) is used for buffering an output stream. It internally uses buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

For adding the buffer in an OutputStream, use the BufferedOutputStream class. Let's see the syntax for adding the buffer in an OutputStream:

OutputStream os= **new** BufferedOutputStream(**new** FileOutputStream("D:\\IO Package\\testout.txt"));



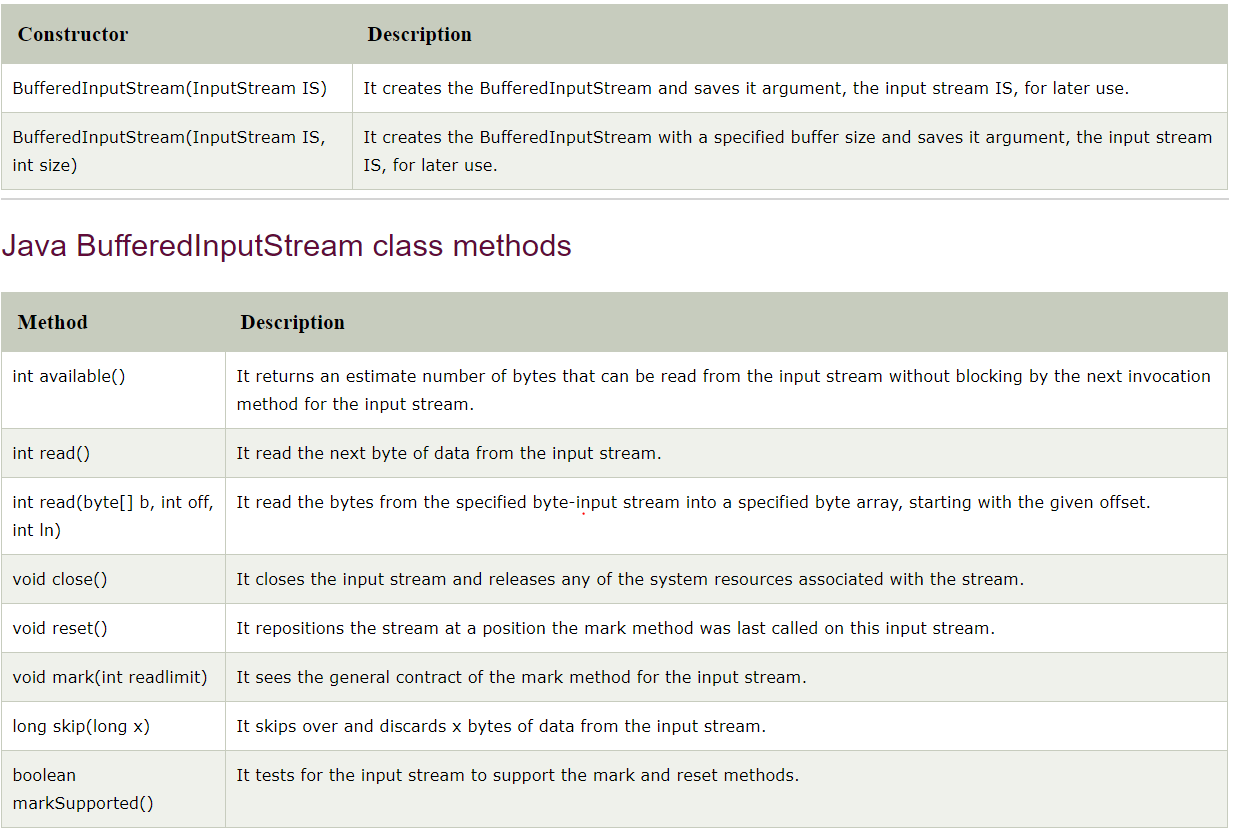


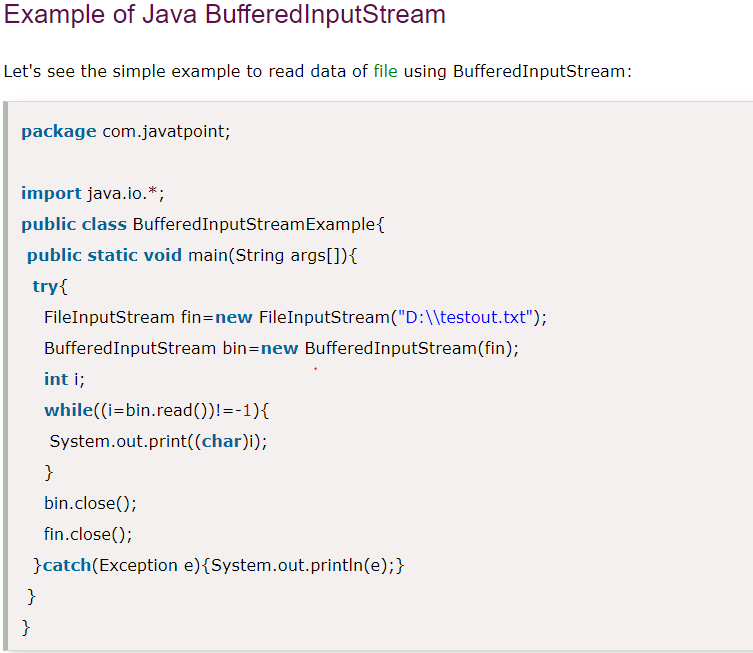
## Java BufferedInputStream Class

Java BufferedInputStream [class](https://www.javatpoint.com/object-and-class-in-java) is used to read information from [stream](https://www.javatpoint.com/java-8-stream). It internally uses buffer mechanism to make the performance fast.

The important points about BufferedInputStream are:

* When the bytes from the stream are skipped or read, the internal buffer automatically refilled from the contained input stream, many bytes at a time.
* When a BufferedInputStream is created, an internal buffer [array](https://www.javatpoint.com/array-in-java) is created.





## Java SequenceInputStream Class

[Java](https://www.javatpoint.com/java-tutorial) SequenceInputStream [class](https://www.javatpoint.com/object-class) is used to read data from multiple [streams](https://www.javatpoint.com/java-8-stream). It reads data sequentially (one by one).

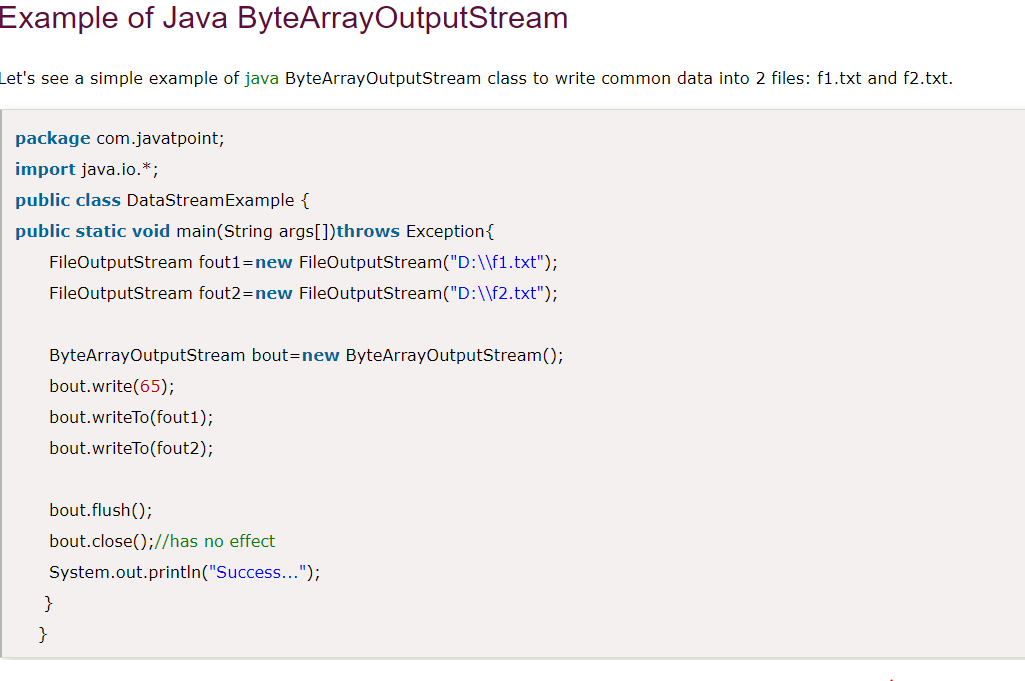


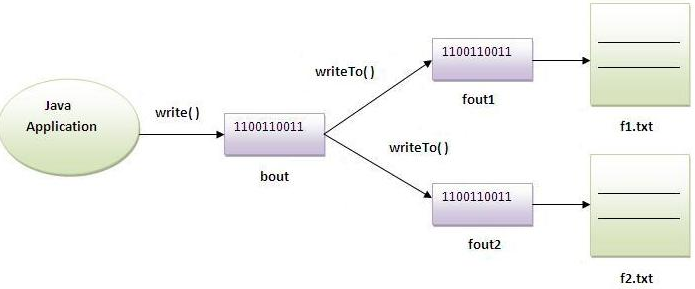
## Java ByteArrayOutputStream Class

Java ByteArrayOutputStream class is used to **write common data** into multiple files. In this stream, the data is written into a byte [array](https://www.javatpoint.com/array-in-java) which can be written to multiple streams later.

The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams.

The buffer of ByteArrayOutputStream automatically grows according to data.



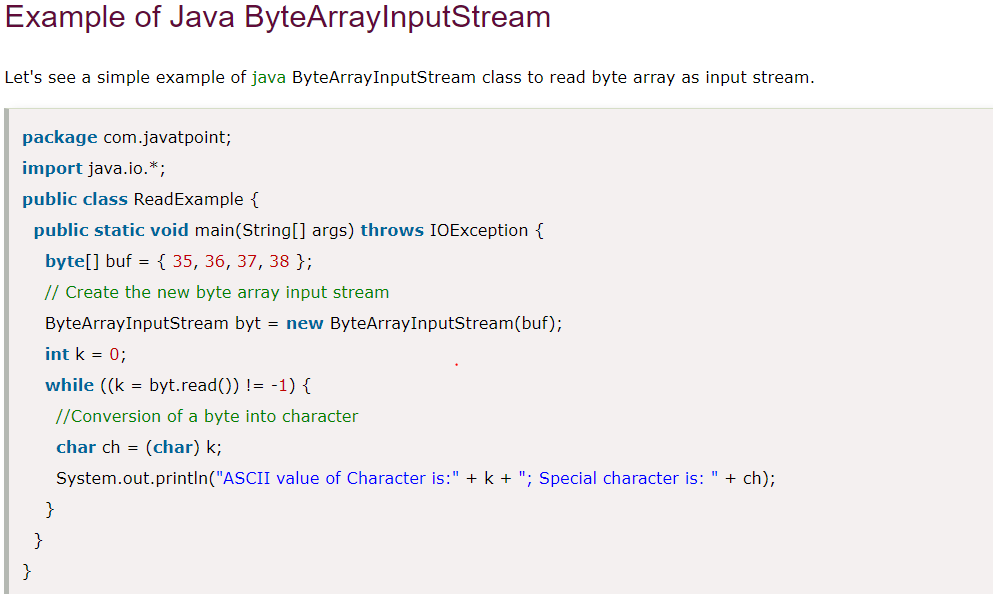


## Java ByteArrayInputStream Class

The ByteArrayInputStream is composed of two words: ByteArray and InputStream. As the name suggests, it can be used to read byte [array](https://www.javatpoint.com/array-in-java) as input stream.

Java ByteArrayInputStream [class](https://www.javatpoint.com/object-and-class-in-java) contains an internal buffer which is used to **read byte array** as stream. In this stream, the data is read from a byte array.

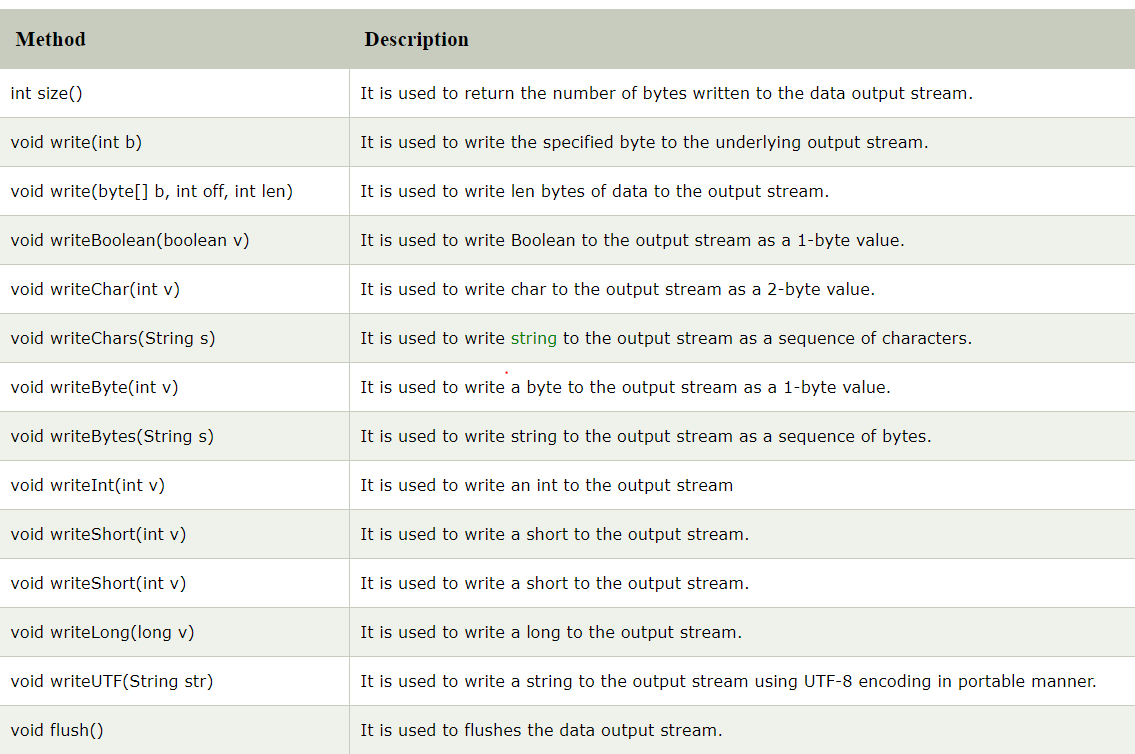
The buffer of ByteArrayInputStream automatically grows according to data.

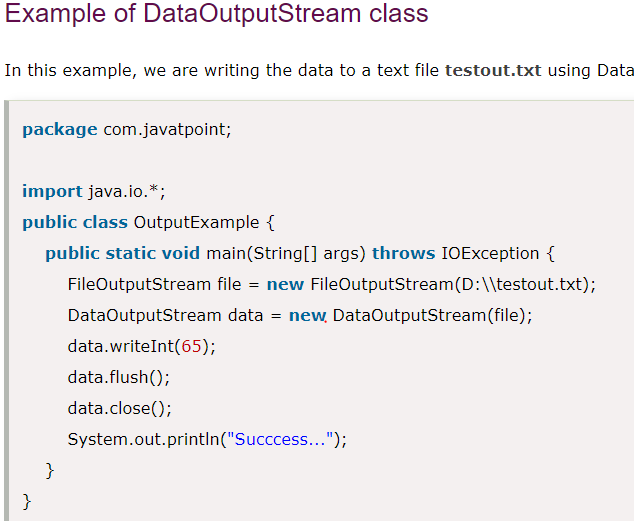


## Java DataOutputStream Class

Java DataOutputStream [class](https://www.javatpoint.com/object-and-class-in-java) allows an application to write primitive [Java](https://www.javatpoint.com/java-tutorial) data types to the output stream in a machine-independent way.

Java application generally uses the data output stream to write data that can later be read by a data input stream.

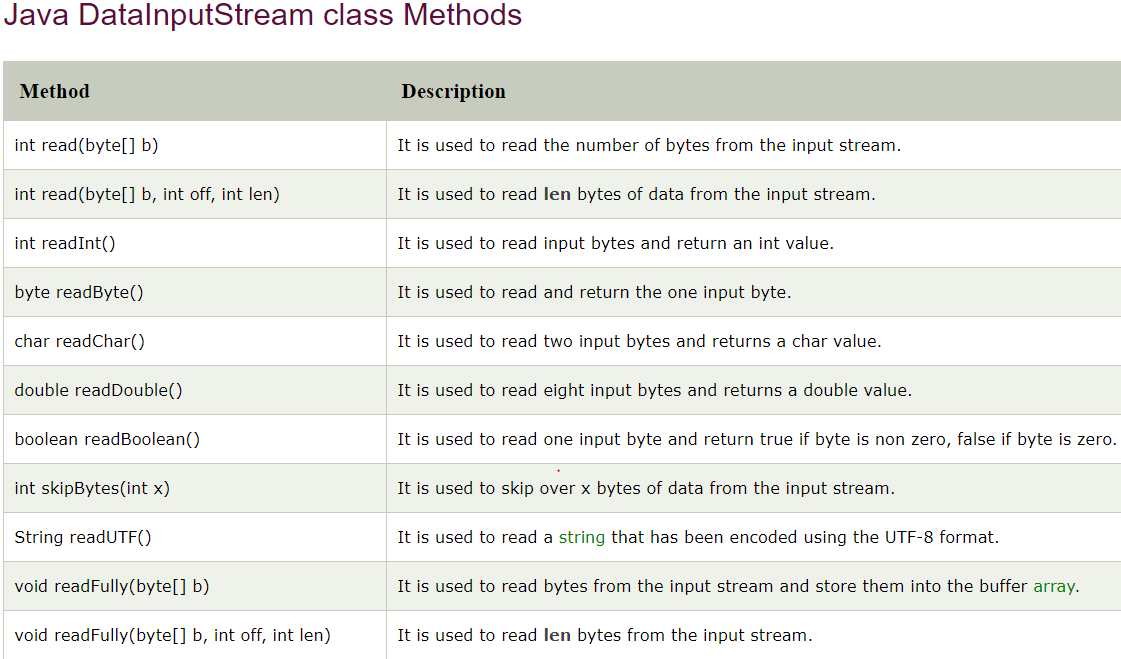




## Java DataInputStream Class

Java DataInputStream [class](https://www.javatpoint.com/object-and-class-in-java) allows an application to read primitive data from the input stream in a machine-independent way.

Java application generally uses the data output stream to write data that can later be read by a data input stream.



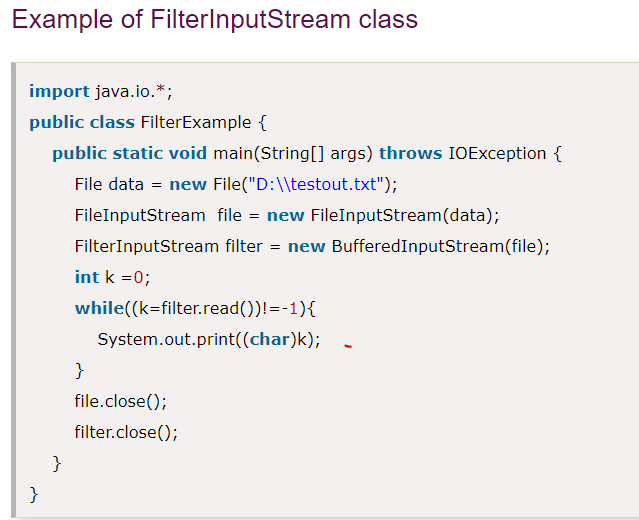


## Java FilterOutputStream Class

Java FilterOutputStream class implements the OutputStream [class](https://www.javatpoint.com/object-and-class-in-java). It provides different sub classes such as [BufferedOutputStream](https://www.javatpoint.com/java-bufferedoutputstream-class) and [DataOutputStream](https://www.javatpoint.com/java-dataoutputstream-class) to provide additional functionality. So it is less used individually.

## Java FilterInputStream Class

Java FilterInputStream class implements the InputStream. It contains different sub classes as [BufferedInputStream](https://www.javatpoint.com/java-bufferedinputstream-class), [DataInputStream](https://www.javatpoint.com/java-datainputstream-class) for providing additional functionality. So it is less used individually.



## Java Writer

It is an [abstract](https://www.javatpoint.com/abstract-class-in-java) class for writing to character streams. The methods that a subclass must implement are write(char[], int, int), flush(), and close(). Most subclasses will override some of the methods defined here to provide higher efficiency, functionality or both.

## Java Reader

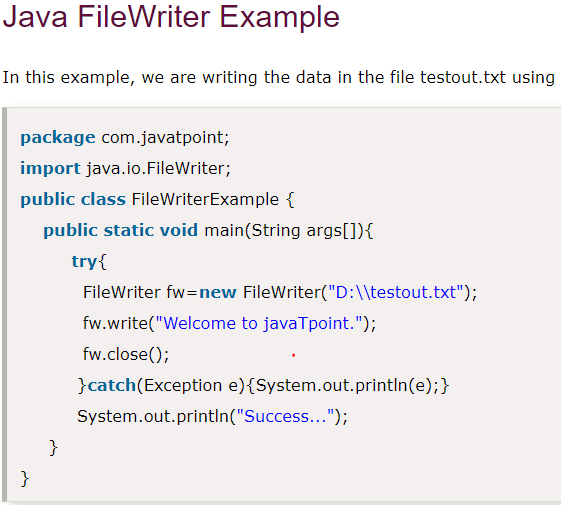
[Java](https://www.javatpoint.com/java-tutorial) Reader is an [abstract class](https://www.javatpoint.com/abstract-class-in-java) for reading character [streams](https://www.javatpoint.com/java-8-stream). The only methods that a subclass must implement are read(char[], int, int) and close(). Most subclasses, however, will [override](https://www.javatpoint.com/method-overriding-in-java) some of the methods to provide higher efficiency, additional functionality, or both.

## Java FileWriter Class

Java FileWriter class is used to write character-oriented data to a [file](https://www.javatpoint.com/java-file-class). It is character-oriented class which is used for file handling in [java](https://www.javatpoint.com/java-tutorial).

Unlike FileOutputStream class, you don't need to convert string into byte [array](https://www.javatpoint.com/array-in-java) because it provides method to write string directly.





## Java FileReader Class

Java FileReader class is used to read data from the file. It returns data in byte format like [FileInputStream](https://www.javatpoint.com/java-fileinputstream-class) class.

It is character-oriented class which is used for [file](https://www.javatpoint.com/java-file-class) handling in [java](https://www.javatpoint.com/java-tutorial).

