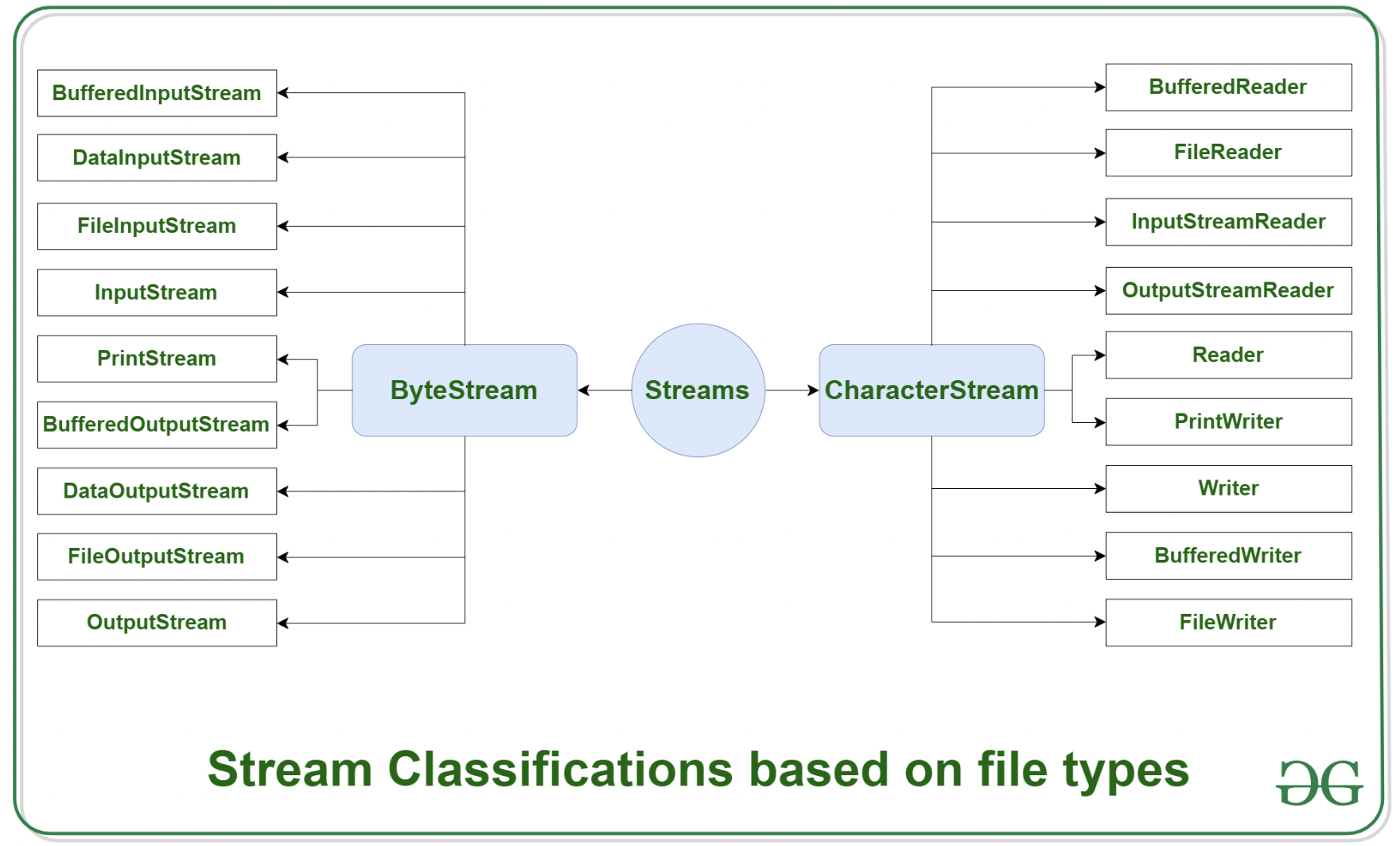
## **What are Streams in Java?**

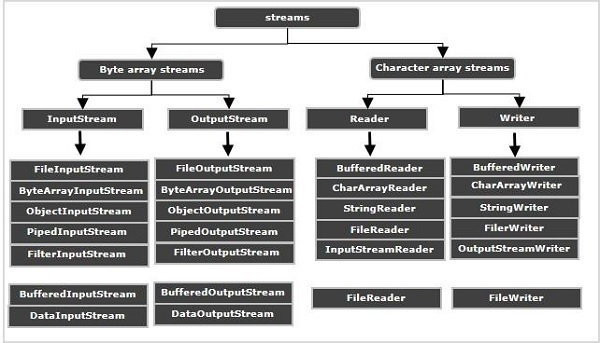
Streams are used in Java to transfer data between programs and I/O devices like a file, network connections, or consoles. A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

**Java I/O** (Input and Output) is used *to process the input* and *produce the output*.

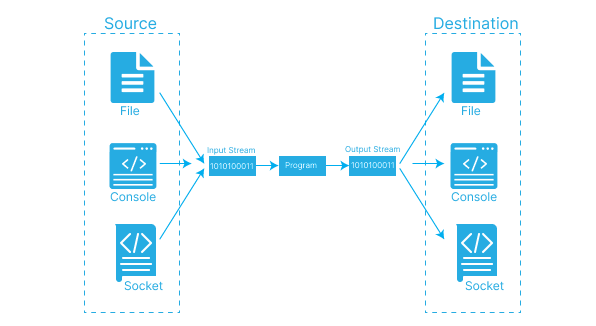
Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform **file handling in Java** by Java I/O API.





Streams are used in Java to transfer data between programs and I/O devices like a file, network connections, or consoles.



## **What are the different Types of Streams in Java?**

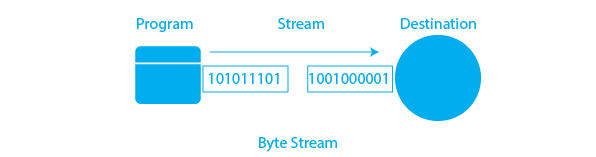
There are two types of java streams:

### **1. Byte Streams**

ByteStream classes are used to read bytes from source and write bytes to destination . ByteStream classes read and write 8-bit data. Using ByteStream classes, we can save video, audio, characters, and so on. The java.io package contains these classes.

Byte Java streams have a three-phase mechanism:

* **Split:** A spliterator divides the input data source into a stream. The Java Spliterator interface is an internal iterator that divides the stream into smaller parts for traversal.
* **Apply:** The elements in the stream are processed.
* **Combine:** After the elements have been processed, they are combined once more to produce a single result.



The ByteStream classes are classified into two types: InputStream and OutputStream. These are the abstract super classes for all Input/Output stream classes.

* **InputStream:**This type of java stream reads data from a source. An InputStream can read data from a file, network connection, or any other I/O device. An InputStream can be used to read data from a file  
  **Syntax of InputStream:**FileInputStream sourceStream = new FileInputStream("path\_to\_file");
* **OutputStream:**The outputstream is a java stream that takes data from a Java program and sends or writes it to the destination (data sink). A data flow out of a program is represented by an output. The output stream connects a Java program to a data sink. A Java program writes or sends data to the data sink.  
  The data flows from a data source to a Java program via an input stream. The data flows from a Java program to a data sink via an output stream.  
  **Syntax of OutputStream:**FileOutputStream targetStream = new FileOutputStream("path\_to\_file");

### **2. Character Stream**

The Character stream is used for 16-bit Unicode input and output operations. Character streams would be advantageous if we wanted to copy a text file containing characters from one source to another using streams because it deals with characters. Java characters are 2 bytes (or 16 bits) in size. CharacterStream classes are provided by the java.io package to overcome the limitations of ByteStream classes, which can only handle 8-bit bytes and are incompatible with working directly with Unicode characters. CharacterStream classes are used to work with Unicode characters that are 16 bits in length. They can work with characters, char arrays, and Strings.

The CharacterStream classes, on the other hand, are primarily used to read characters from the source and write them to the destination. CharacterStream classes are divided into two types for this purpose: Reader classes and Writer classes.

Character streams in Java, like Byte Streams, have a three-phase mechanism, as explained above.

There are numerous character stream classes in Java, but the most common ones are-

* **FileReader**It is used to read two bytes at a time from the source. The function Object() { [native code] } for creating an instance of the FileReader class is as follows.  
  **Syntax of FileReader:**FileFileWriter in = new FileWriter("path\_to\_file");
* **FileWriter**It is used to write two bytes at a time to the destination. The function Object() { [native code] } for creating an instance of the FileWriter class is as follows.  
  **Syntax of FileWriter:**FileWriter in = new FileWriter("path\_to\_file");

### **Difference between Byte Stream and Character Stream:**

Here we have the differences:

| **Parameters** | **Character Streams** | **Byte Streams** |
| --- | --- | --- |
| Data Handling | Handle character-based data | Handle raw binary data |
| Representation | Classes end with "Reader" or "Writer" | Classes end with "InputStream" or "OutputStream" |
| Suitable for | Textual data, strings, human-readable info | Non-textual data, binary files, multimedia |
| Character Encoding | Automatic encoding and decoding | No encoding or decoding |
| Text vs non-Text data | Text-based data, strings | Binary data, images, audio, video |

| **Byte Stream** | **Character Stream** |
| --- | --- |
| It operates on raw binary data and data is processed byte by byte | It operates on text data and is processed character by character |
| Suitable for processing non-textual data such as images, videos, etc. | Suitable for processing textual data such as documents, HTML, etc. |
| Input and Output operations are performed using InputStream and OutputStream classes | Input and Output operations are performed using Reader and Writer classes |
| Suitable for low-level input and output operations | Suitable for high-level input and output operations |
| Examples include FileInputStream and FileOutputStream | Examples include FileReader and FileWriter |

**Conclusion**The java stream method aids in sequentially accessing a file. There are two types of java streams: Byte Stream and Character Stream. Byte streams are used to perform input and output of 8-bit bytes. When we want to read/write binary data, we can use byte streams. The character stream is used to perform 16-bit Unicode input and output operations. Character streams are used to read and write characters. Byte and Character Streams have a three-phase mechanism that includes Split, Apply, and Combine.FileInputStream and FileOutputStream are common classes for reading and writing data using byte streams.FileReader and FileWriter are commonly used to read and write data using character streams.

## 

**Q1: What are InputStream and OutputStream classes in Java?  
A:** InputStream and OutputStream are abstract classes in Java that represent the input and output streams of bytes. They provide methods for reading and writing bytes from and to a source.

**Q2: What are Reader and Writer classes in Java?  
A:** Reader and Writer are abstract classes in Java that represent the input and output streams of characters. They provide methods for reading and writing characters from and to a source.

**Q3: What is the difference between InputStream/OutputStream and Reader/Writer classes?  
A:** InputStream/OutputStream classes operate on raw binary data, while Reader/Writer classes operate on text data. InputStream/OutputStream classes process data byte by byte, while Reader/Writer classes process data character by character. InputStream/OutputStream classes do not perform any character encoding or decoding, while Reader/Writer classes perform character encoding and decoding using specified character sets.

**Q4: What are some examples of Byte Stream classes in Java?  
A:** Some examples of Byte Stream classes in Java include FileInputStream, FileOutputStream, BufferedInputStream, and BufferedOutputStream.

**Q5: What are some examples of Character Stream classes in Java?  
A:** Some examples of Character Stream classes in Java include FileReader, FileWriter, BufferedReader, and BufferedWriter.

**Serialization & Deserialization**

Serialization is the process of converting an object to a byte stream. This byte stream can then be saved to a file, sent over a network, or stored in a database.

Deserialization is the reverse process of serialization. It involves taking a byte stream and converting it back into an object.

