Java provides many stream classes.

Some basic level:

1. Byte by byte (ASCII) read data
2. Char by char (Unicode)

All stream classes are available in java.io

* All these classes are child classes of input and output stream.

1. **File Input Stream (FIS):** Reads bytes from a file.
2. **File Output Stream (FOS**): Writes bytes to a file.

* These both are low level Streams and read/ write data from file.

1. **Buffer Input Stream (BIS**): Adds buffering functionality to an input stream, improving efficiency.
2. **Buffer Output Stream (BOS):** Adds buffering functionality to an output stream.

* Both these are high-level Streams.

1. **Data Input Stream (DIS):** Reads primitive data types from an input stream (e.g., int, double).
2. **Data Output Stream (DOS):** Writes primitive data types to an output stream.

* These DIS and DOS read/ write primitive data type (like int etc). Both these are high-level Streams.

1. **Object Input Stream (OIS):** Reads objects from an input stream.
2. **Object Output Stream (OOS):** Writes objects to an output stream.

* Read/ write object on stream directly. Both these are high-level Streams.

1. **Cypher Input Stream (CIS):** Reads encrypted data from an input stream.
2. **Cypher Output Stream (COS):** Writes encrypted data to an output stream.

* Data encryption, decryption. Both these are high-level Streams.

1. **Print Stream (PS).**

* Provides convenient methods for printing different data types to a stream.

All stream classes of java that end with stream belong to **byte category**. (Naming Convention).

**FileInputStream** and **FileOutputStream** are subclasses of InputStream and OutputStream, respectively.

InputStream (IS) and OutputStream (0S) -> these are abstract classes (to apply rules, constraints). These abstract classes impose child classes to provide functionality of these read/ write (maybe whole file, maybe line by line).

These all classes based on IS, OS provide basis of read() [all classes of IS], write() [all classes of OS]….byte by byte.

* **Reader Writer**

File Reader (FR) File Writer (FW)

Buffer Reader (BR) Buffer Writer (BW)

Print Writer (PW)

All streams of Java that end with writer or reader, belong to **char stream category.**

1. **FileReader (FR):** Reads characters from a file. It's a low-level character stream.
2. **FileWriter (FW):** Writes characters to a file. Also a low-level character stream.
3. **BufferedReader (BR):** Adds buffering functionality to a character input stream, improving efficiency.
4. **BufferedWriter (BW):** Adds buffering functionality to a character output stream.
5. **PrintWriter (PW):** Provides convenient methods for printing different data types to a text file. It's often used for writing formatted text.

* When u want to do reading/ writing with any file on computer without any program, u perform **these 3 steps:**

1. Open file.
2. Type, read, write data
3. Save/ close file.

In java code, we will also do these 3 things.

**Example:**

**FileInputStream f = new FileInputStream(“test.txt”);**

FileInputStream -> This is a connection creation with this file -> test.txt

This line is creating connection with file.

We have given the relative path of file here. You can give any path (relative [ onward from me ] or absolute [ from root ]).

Now you will make call to function:

**int byte\_\_\_ = f.read();**

This read function will return next available byte. It will return ASCII (int) value of that character. And we will save that value in any integer.

When we will call this function for first time, it will return first byte. At second call, it will return second byte and so on.

I don’t know how many bytes and characters are there in the file. So pass this function in a loop.

EOF (End of file): When there is no further byte available in memory, this function will return -1.

**while(f.read() != -1) //EOF**

read -> part of hard disk. When there will be no byte on segment, it will return -1.

Q: What if -1 is present in the file?

A: Then remember it will take the ASCII of minus and one. This will not conflict with the -1 which we are returning. -1 indicates that there is no further data available for reading. Do not put conflict with content.

**f.close();**

You must need to close the connection because system has limited resources. It is necessary to close so that resources can be available for other.

Whenever you are dealing with external resources, close your connection at end.

All classes are present in **java.io**, so first import this package.

* **Checked Exceptions:**

If we won’t write these checked exceptions, our program **wont compile**. Checked exceptions occur whenever you are trying to communicate with external resource (file etc). You must need to handle these exceptions otherwise you won’t be able to compile.

What does compile means here? If u are not able to compile, it means syntax error. Language has made a rule.

**IO Exception ->** **Handle this exception**. If you won’t handle it, you can’t compile (syntax error, violating **language rule).**

* **Scenarios when IO Exception can occur:**

1. File doesn’t exist.
2. You don’t have permission to read the file.

* Use throws or try catch to handle this exception.
* You are trying to write a file and file doesn’t exist, Java will create a file for you.
* In case file already exists and you write, it will erase previous content and write new content.
* How we can open file in append mode? **(HW)**
* In Java, you can open a file in append mode by using the **FileWriter class** with the constructor that takes **a boolean parameter** indicating whether the file should be opened in append mode.
* **FileWriter fileWriter = new FileWriter(fileName, true);**
* If file doesn’t exist in case of reading, you will get an exception.
* **If extension of file is .pdf, .txt, .xyz, then what difference it will make? Can I read .pdf file in any programming language?**
* File is part/ representation/abstraction of hardware. Chunk/ segment of hardware stores this file. Extensions have nothing to do with programming languages. They are just block/ chunks for any programming language. Programming languages have no concern with their extension and can read/ write any type of file. Extensions are for application softwares, for format.
* The file that you have created anywhere else like on word and you read/ write in java or Csharp, then you will also get some additional/ extra content. That extra content will have been written by application software itself like author etc.
* If you have write a file yourself through any program, then you will only see content of file (not any extra info/ content).