File Handling

File Handing in java comes under IO operations. Java IO package java.io classes are specially provided for file handling in java.

File Handling in Java

To use the File class, create an object of the class, and specify the filename or directory name.

```
import java.io.File;  // Import the File class
File myObj = new File("filename.txt"); // Specify the filename
```

The File class has many useful methods for creating and getting information about files. For example:

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

Get File Information

Now that we have created a file, we can use other File methods to get information about that file:

```
import java.io.File;
public class GetFileInfo {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.exists()) {
      System.out.println("File name: " + myObj.getName());
```

```
      System.out.println("Absolute path: " + myObj.getAbsolutePath());
      System.out.println("Writeable: " + myObj.canWrite());
      System.out.println("Readable " + myObj.canRead());
      System.out.println("File size in bytes " + myObj.length());
    } else {
      System.out.println("The file does not exist.");
    }
  }
}
```

Some of the common file handling operations are;

1. Create file
2. Delete file
3. Read file
4. Write file
5. Change file permissions

1. **Create file –** FileHandling1.java

We can use File class **createNewFile()** method to create new file. This method returns true if file is successfully created, otherwise it returns false.

Note that the method is enclosed in a try...catch block. This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason).

In subsequent execution the file will be present so the createNewFile will return false.

2. **Delete file -** FileHandling2.java

   File class delete method is used to delete a file or an empty directory. File delete method returns true if file is deleted successfully or else it returns false.

3. **Read File –** FileHandling3.java

   There are many ways to read a file in java. We can use BufferedReader, FileReader or Files class.

4. **Write File-** FileHandling4.java

   We can use FileWriter, BufferedWriter, Files or FileOutputStream to write file in java.

5. **Change File Permissions-**FileHandling5.java

   File class provide methods to get file permission details as well as change them.

Note: There are many available classes in the Java API that can be used to read and write files in Java: FileReader, BufferedReader, Files, Scanner, FileInputStream, FileWriter, BufferedWriter, FileOutputStream, etc. Which one to use depends on the Java version you're working with and whether you need to read bytes or characters, and the size of the file/lines etc.

There are some rules associated with absolute path and relative path, read about them at java create new file. --- https://www.journaldev.com/825/java-create-new-file

https://www.journaldev.com/855/java-set-file-permissions-posixfilepermission

What is File I/O?

Java I/O stream is the flow of data that you can either read from, or you can write to.

It is used to perform read and write operations in file permanently.

Java uses streams to perform these tasks. Java I/O stream is also called **File Handling, or File I/O**. It is available in java.io package.

Some input-output stream will be initialized automatically by the JVM and these streams are available in System class as in, out, and err variable.

In reference refers the default input device, i.e. keyboard.

Out and err refers to the default output device, i.e. console.

Streams:

Streams are the sequence of bits(data).

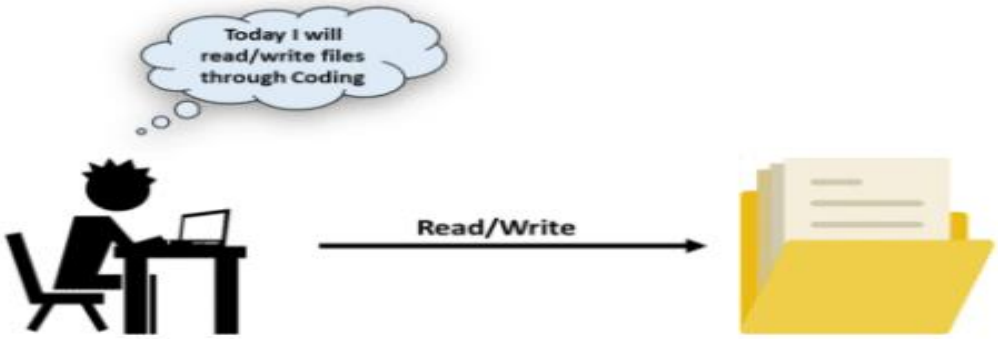
There are two types of streams:

Input Streams

Output Streams

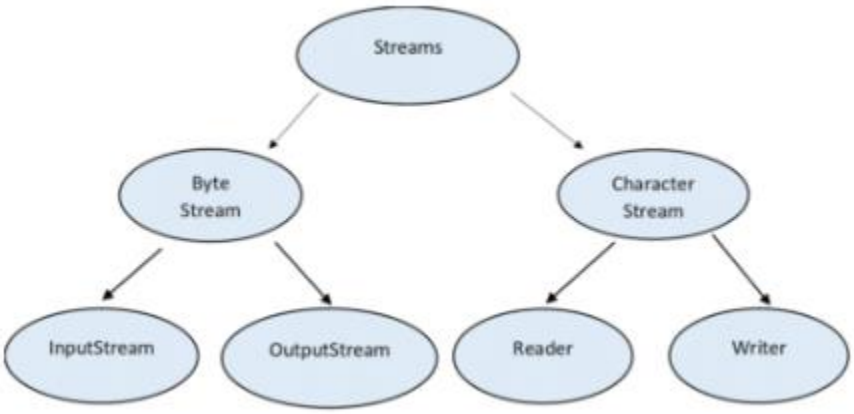
Input Streams: Input streams are used to read the data from various input devices like keyboard, file, network, etc.

Output Streams: Output streams are used to write the data to various output devices like monitor, file, network, etc.



There are **two types of streams based on data**:

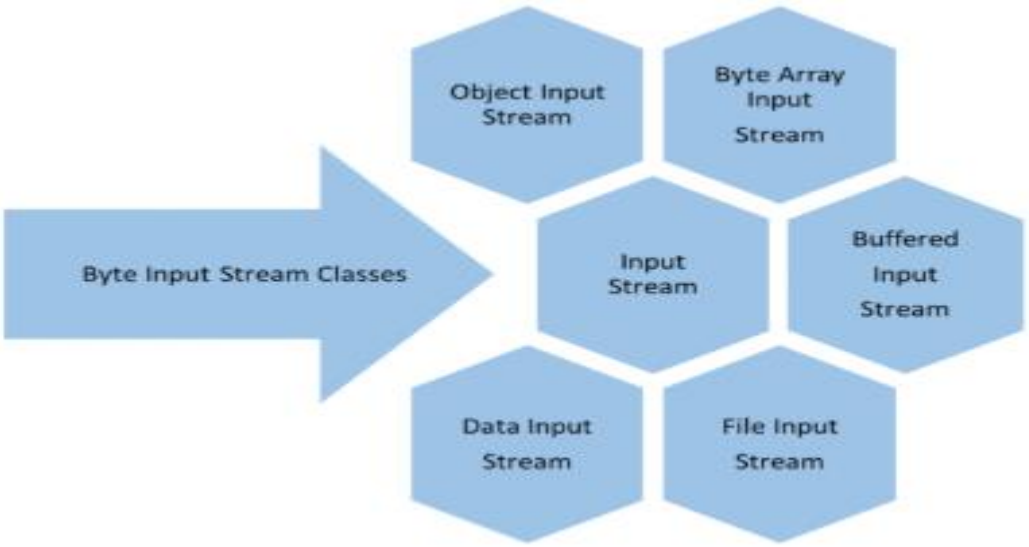
- **Byte Stream:** used to read or write byte data.
- **Character Stream:** used to read or write character data.



Byte Input Stream:

These are used to read byte data from various input devices.
InputStream is an abstract class and it is the super class of all the input byte streams.

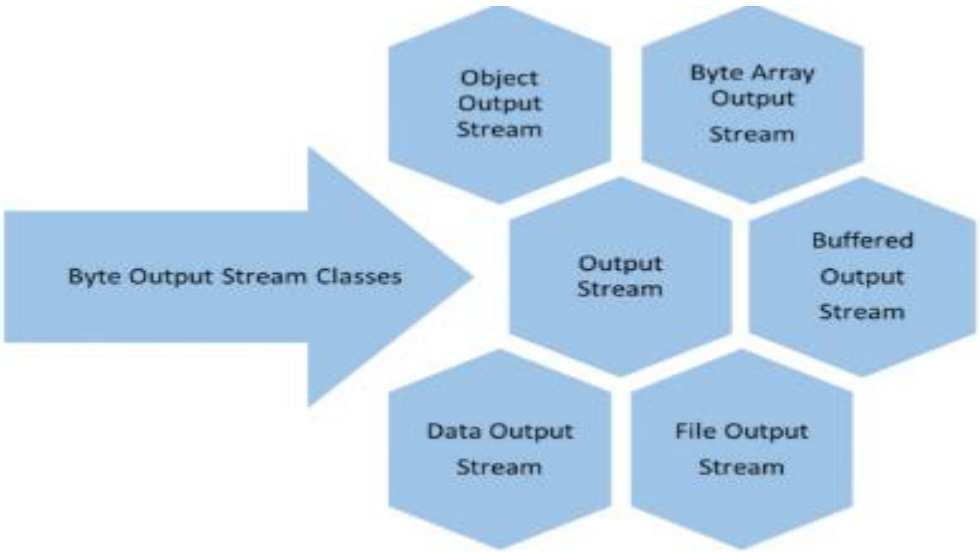
List of Byte Input Streams:



Byte Output Stream:

- These are used to write byte data to various output devices.
- **Output Stream** is an abstract class and it is the superclass for all the output byte streams.

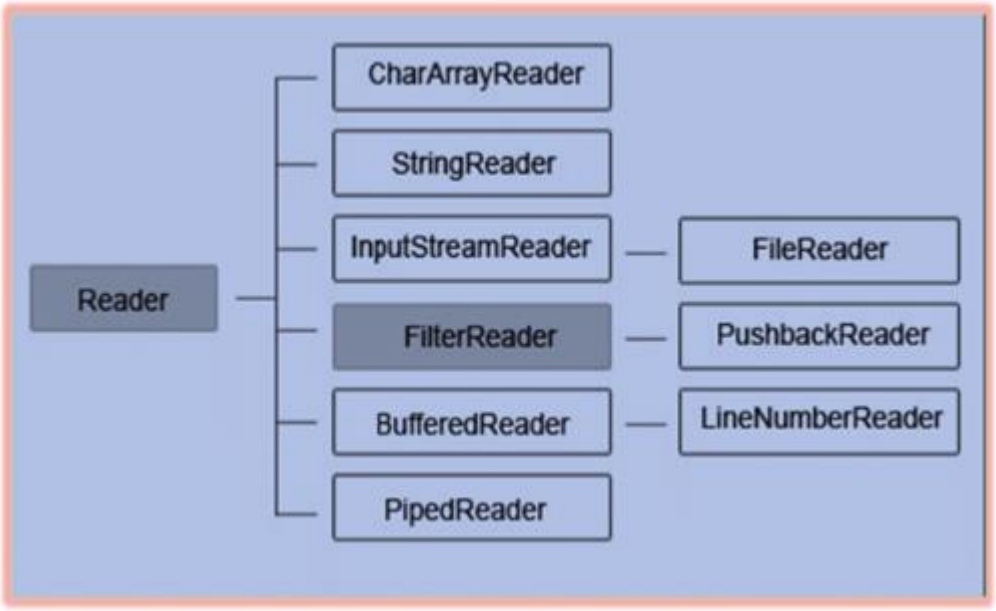
List of Byte Output Streams:



Character Input Stream:

- These are used to read char data from various input devices.
- **Reader** is an abstract class and is the super class for all the character input streams.

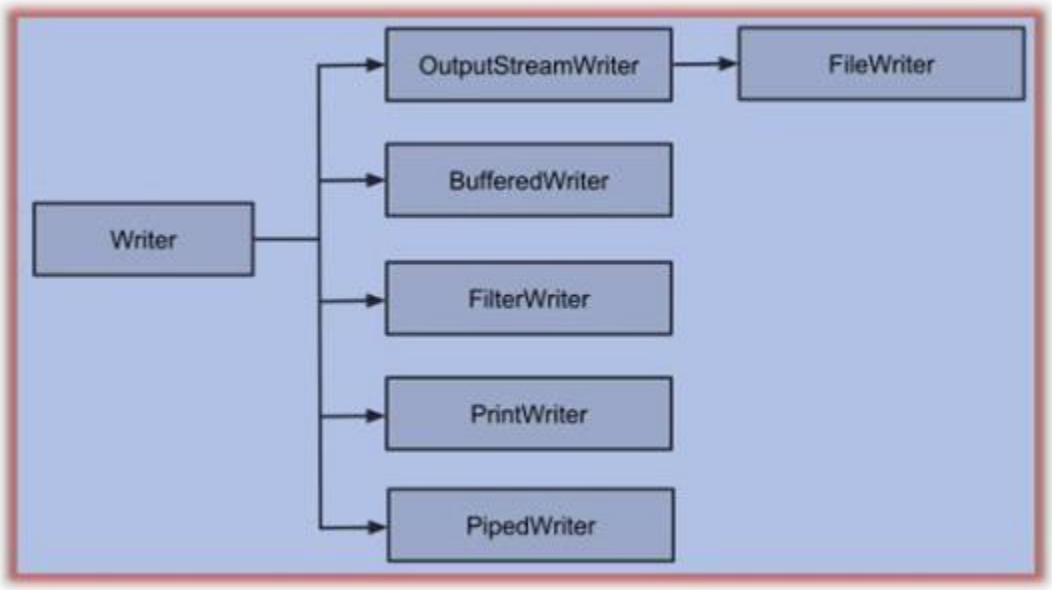
List of Character Input Streams:



Character Output Stream:

- These are used to write char data to various output devices.
- **Writer** is an abstract class and is the super class of all the character output streams.

List of Character Output Stream:



Example to read contents of file: Example1.java, Example2.java

Example to write in a file: Example3.java, Example4.java

When to use Character Stream over Byte Stream?

In Java, characters are stored using Unicode conventions. Character stream is useful when we want to process text files. These text files can be processed character by character. A character size is typically 16 bits.

When to use Byte Stream over Character Stream?

Byte oriented reads byte by byte. A byte stream is suitable for processing raw data like binary files.

Similarly, different subclasses can also be used to read and write.

Part-5 - Practical 6 : https://www.codesdope.com/java-file-io/