Understanding Character Streams:

- In character Streams data is transferred in the form of characters.
- In character Streams the length of each data packet is 2 bytes.
- All character stream classes are sub classes for Reader & Writer classes which are abstract classes. (present in 'java.io.Reader' & 'java.io.Writer')

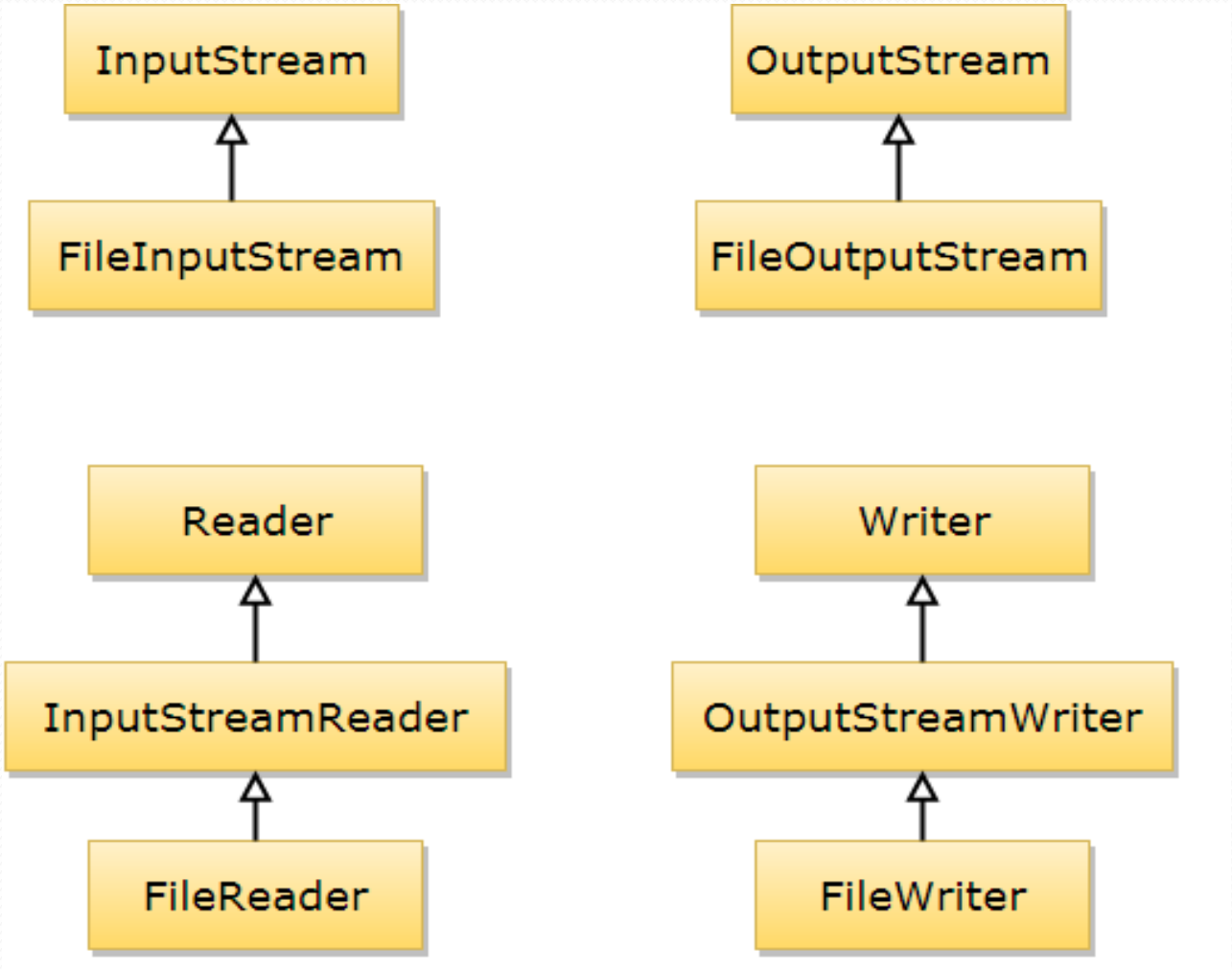
FileReader Class:

- It is the child class for InputStreamReader.
- We can use this class to read the data character by character from the stream.
- Data like images, audio, video etc we can't read by using FileReader class. (We are supossed to use Byte Streams)
- FileReader fr=new FileReader ("abc.txt");

FileWriter Class:

- It is the child class for OutputStreamWriter.
- We can use this class to write the data character by character in the from of stream in to destination file.
- Same as FileOutput Stream class in FileWriter Class also We can use append mode.(By setting the second parameter as **‘true’**).
- FileWriter fw=new FileWriter ("abc.txt");

Byte Streams Vs Character Streams



Understanding Buffered Streams:

- A Buffer is a portion in the memory that is used to store a stream of data.
- In I/O operations each read or write request is handled directly by the underlying OS.
- This can make a program much less efficient, since each such request often triggers disk access, network activity, or some other operation that is relatively expensive.
- To reduce this kind of overhead, the Java platform implemented buffered I/O streams.
- Buffered input streams read data from a memory area known as a buffer.
- Buffered output streams write data to a buffer.
- Buffered streams are same like Byte & Character Streams but with more efficiency.

- There are four buffered stream classes used to wrap unbuffered streams
- BufferedInputStream and BufferedOutputStream create buffered byte streams
- BufferedReader and BufferedWriter create buffered character streams.

Syntax:

BufferedInputStream:

BufferedInputStream br=new BufferedInputStream(**new FileInputStream(“FilePath"));**

BufferedOutputStream:

BufferedOutputStream br=new BufferedOutputStream(**new FileOutputStream(“FilePath"));**

- **flush() :** When you write data to a stream, it is not written immediately, and it is buffered. So use flush() when you need to be sure that all your data from buffer is written.

BufferedReader:

BufferedReader br=new BufferedReader(new FileReader(" FilePath "));

BufferedWriter :

BufferedWriter bw= new BufferedWriter(new FileWriter(" FilePath "));