1. **Which class is used to create a directory in Java?**

This is tricky becuase there is no Directory class, File class is used to create both file and directory in Java. You can see the linked answer for more details.

File directory = **new** File(dir);

success = directory.mkdir();

It is a non-stream (not used for file operations) class used to know the properties of a file like when it was created (or modified), has read and write permissions, size etc.

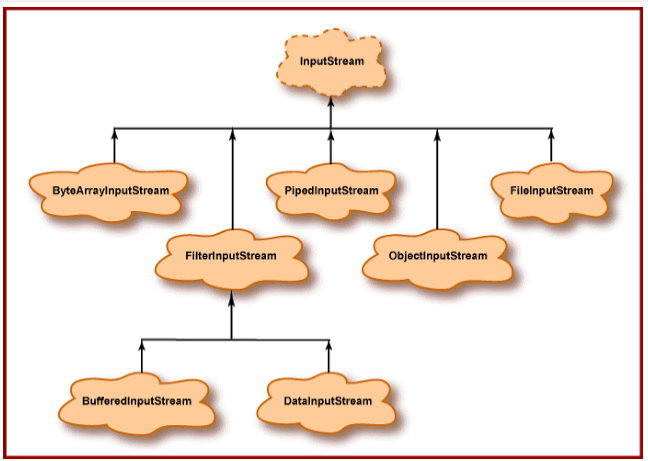
**What are the super most classes of all streams?**

java.io.InputStream

java.io.OutputStream

java.io.Reader

java.io.Writer.



**What are FileInputStream and FileOutputStream?**

These two are general purpose classes used by the programmer very often to copy file to file. These classes work well with files containing less data of a few thousand bytes as by performance these are very poor. For larger data, it is preferred to use BufferedInputStream (or BufferedReader) and BufferedOutputStream (or BufferedWriter).

**Which you feel better to use – byte streams or character streams?**

I feel personally to go with character streams as they are the latest. Many features exist in character streams that do not in byte streams like a) using BufferedReader in place of BufferedInputStreams and DataInputStream (one stream for two) and b) using newLine() method to go for next line and for this effect we must go for extra coding in byte streams etc.

**What System.out.println()?**

"println()" is a method of PrintStream class.

"out" is a static object of PrintStream class defined in "System" class.

System is a class from java.lang package used to interact with the underlying operating system by the programmer.

**What are filter streams?**

Filter streams are a category of IO streams whose responsibility is to add extra functionality (advantage) to the existing streams like giving line numbers in the destination file that do not exist int the source file or increasing performance of copying etc.

FilterInputStream, FilterOutputStream, FilterReader and FilterWriter.

**FilterStream**

LineNumberInputStream

DataInputStream(contains special methods like readInt(), readDouble() and readLine())

BufferedInputStream (gives buffering effect that increases the performance to the peak)

**What is PrintStream and PrintWriter?**

Functionally both are same but belong to two different categories – byte streams and character streams. println() method exists in both classes.

**Which streams are advised to use to have maximum performance in file copying?**

BufferedInputStream and BufferedOutputStream on byte streams side and BufferedReader and BufferedWriter on character streams side.

**What are piped streams?**

There are four piped streams – PipedInputStream, PipedOutputStream, PipedReader and PipedWriter. These streams are very useful to pass data between two running threads (say, processes).

**What is RandomAccessFile?**

It is a special class from java.io package which is neither a input stream nor a output stream (because it can do both). It is directly a subclass of Object class. Generally, a stream does only one purpose of either reading or writing; but RandomAccessFile can do both reading from a file and writing to a file. All the methods of **DataInputStream** and **DataOutStream** exist in RandomAccessFile.

**BufferedReader**

BufferedReader is faster as compared to scanner because scanner does parsing of input data and BufferedReader simply reads sequence of characters.

BufferedReader has significantly larger buffer memory (8kb) than Scanner (1kb)

BufferedReader is synchronous while Scanner is not. BufferedReader should be used if we are working with multiple threads.

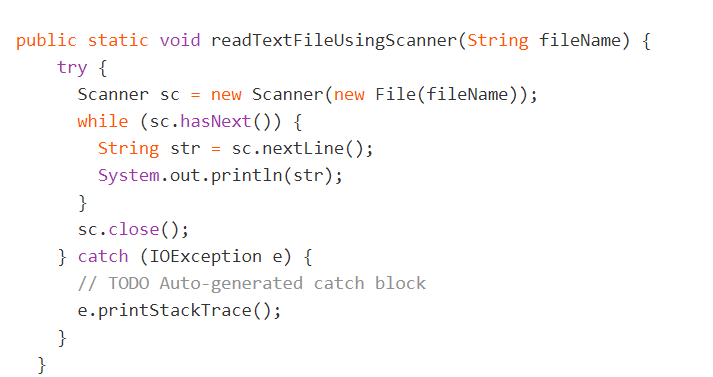
But there some drawbacks as well, like code to write BufferedReader is bit complex than that of Scanner

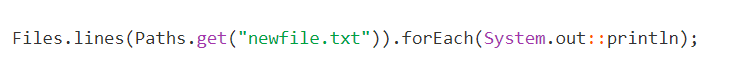
So, if you are concerned more with time and space complexity of your program, Go for BufferedReader. Otherwise Scanner is always there for you

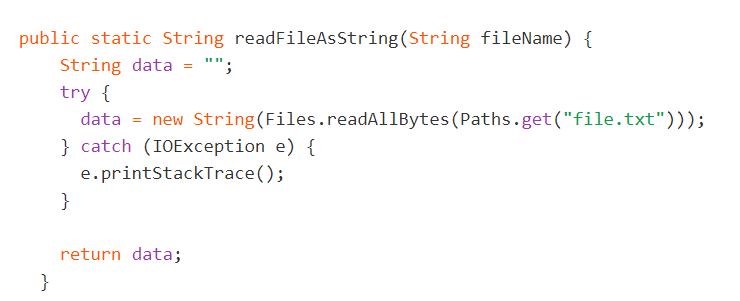
**InputStream.read() vs BufferedReader.read()**

From the perspective of optimisation, it would be better to use the BufferedReader, since it'll read several kilobytes at once, and you can take each character when you want (not necessarily forcing a new IO read).

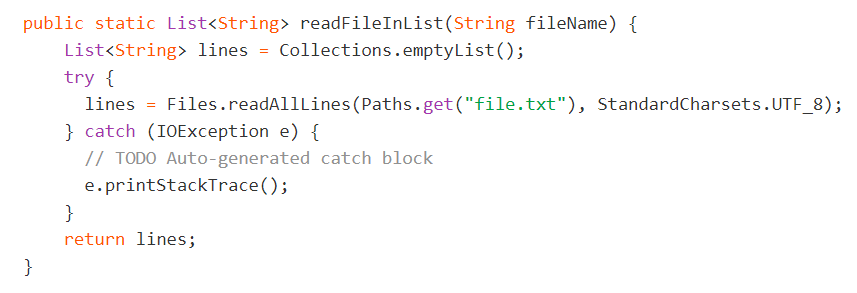
Different ways of Reading a file:







Reading the whole file in a List



How to read a file line by line in Java

