File IO

A stream can be defined as a sequence of data.

I/O Streams

- Byte Streams handle I/O of raw binary data.
- Character Streams handle I/O of character data, automatically handling translation to and from the local character set.
- Buffered Streams optimize input and output by reducing the number of calls to the native API.
- Scanning and Formatting allows a program to read and write formatted text.
- I/O from the Command Line describes the Standard Streams and the Console object.
- Data Streams handle binary I/O of primitive data type and String values.
- Object Streams handle binary I/O of objects.

Byte Streams: Java byte streams are used to perform input and output of 8-bit bytes. eg:FileInputStream and FileOutputStream

Character Streams: Java Character streams are used to perform input and output for 16-bit unicode. eg:FileReader and FileWriter.

Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time

Standard Streams: three standard streams
Standard Input: used to feed data for program eg: keyboard and represented as System.in.

Standard Output: used to output data eg: computer screen and represented as System.out.

Standard Error: used to output the error eg: computer screen and represented as System.err.

Object -- OutputStream -- FileOutputStream
-- ByteArrayOutputStream
-- FilterOutputStream -- BufferedOutputStream
-- DataOutputStream
-- PrintStream
-- InputStream -- FileInputStream
-- ByteArrayInputStream
-- StringBufferInputStream
-- SequenceInputStream
-- FilterInputStream -- BufferedInputStream
-- DataInputStream
--PushbackInputStream

OutputStream class:
- write(byte)
- write(byte[])
- flush()
- close()

InputStream class:
- read()
- available() - returns an estimate of the number of bytes that can be read from the current input stream.
- close()

FileInputStream and FileOutputStream (File Handling):
If you have to write primitive values then use FileOutputStream.Instead, for character-oriented data, prefer FileWriter.

FileInputStream is used for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader.

ByteArrayOutputStream class:
ByteArrayOutputStream class is used to write data into multiple files. In this stream, the data is written into a byte array that can be written to multiple stream.

The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams.

The buffer of ByteArrayOutputStream automatically grows according to data.
- writeTo(OutputStream)
- write(byte)
- write(byte[])
- flush()
- close() - Closing the ByteArrayOutputStream has no effect.

SequenceInputStream class:
SequenceInputStream class is used to read data from multiple streams. It reads data of streams one by one.

SequenceInputStream(InputStream s1, InputStream s2) - creates a new input stream by reading the data of two input stream in order, first s1 and then s2.

SequenceInputStream(Enumeration e) - creates a new input stream by reading the data of an enumeration whose type is InputStream.

BufferedOutputStream class:

Java BufferedOutputStream class uses an internal buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

```
FileOutputStream fout=new FileOutputStream("f1.txt");
BufferedOutputStream bout=new BufferedOutputStream(fout);
```

BufferedInputStream class:

Java BufferedInputStream class is used to read information from stream. It internally uses buffer mechanism to make the performance fast.

```
FileInputStream fin=new FileInputStream("f1.txt");
BufferedInputStream bin=new BufferedInputStream(fin);
```

FileWriter and FileReader (File Handling in java)
These are character-oriented classes, used for file handling in java.

Java has suggested not to use the FileInputStream and FileOutputStream classes if you have to read and write the textual information.

CharArrayWriter class:(similar to ByteArrayOutputStream)

The CharArrayWriter class can be used to write data to multiple files. This class implements the Appendable interface. Its buffer automatically grows when data is written in this stream. Calling the close()

method on this object has no effect.

Reading data from keyboard:
There are many ways to read data from the keyboard. For example:

InputStreamReader
Console
Scanner
DataInputStream

InputStreamReader class:

```
InputStreamReader r=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(r);
```

BufferedReader class can be used to read data line by line by readLine() method.

Console class:
Console class is be used to get input from console. It provides methods to read text and password.

```
Console c=System.console();
String n=c.readLine();
```

Scanner class:
Scanner class breaks the input into tokens using a delimiter that is whitespace bydefault.
Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

PrintStream class:
The PrintStream class provides methods to write data to another stream. The PrintStream class automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.

Compressing and Uncompressing File
The DeflaterOutputStream and InflaterInputStream classes provide mechanism to compress and uncompress the data in the deflate compression format.

PipedInputStream and PipedOutputStream classes
The PipedInputStream and PipedOutputStream classes can be used to read and write data simultaneously. Both streams are connected with each other using the connect() method of the PipedOutputStream class.

What is an I/O filter?
An I/O filter is an Object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

Relative or Absolute:
A path is either relative or absolute. An absolute path always contains the root element and the complete directory list required to locate the file. A relative path needs to be combined with another path in order to access a file.

Basic File Methods:
Creation - createNewFile()
Delete - delete()
File or Directory - isFile(), isDirectory()
Rename or Move - renameTo(newFile)
Size(bytes) - length()
Path - getPath(), getAbsolutePath(), getCanonicalPath()
Create Temp File - createTempFile(String prefix, String suffix, File directory), createTempFile(String prefix, String suffix)