**Resumen Jerarquía de Flujos**

**Flujos de bytes**

|  |  |
| --- | --- |
| Clase | Descripción |
| InputStream | Madre de un conjunto de clases para leer ficheros byte a byte. |
| FileInputStream  Y FileOutputStream | convierten flujos  binarios en flujos de textos y viceversa.  Permiten recodificar el texto. |
| BufferedInputStream  Y  BufferedOutputStream | BufferedInputStream no es abstracta, es una clase concreta, por tanto puedes crear instancias de esta clase.  se utiliza para capturar bytes escritos en BufferedOutputStream en un búfer y escribir todo el búfer en un lote en un **Java OutputStream** subyacente para un mayor rendimiento. El almacenamiento en búfer puede acelerar bastante la E / S, especialmente cuando se escriben datos en el acceso al disco o en la red. |
| DataInputStream  Y  DataOutputStream | La clase DataInputStream es útil para leer datos del tipo primitivo de una forma portable. Esta clase tiene un sólo constructor que toma un objeto de la clase InputStream o sus derivadas como parámetro.  La clase DataOutputStream es útil para escribir datos del tipo primitivo de una forma portable. Esta clase tiene un sólo constructor que toma un objeto de la clase OutputStream o sus derivadas como parámetro. |
| PipedInputStream  Y  PipedOutputStream | La clase PipedInputStream y la clase PipedOutputStream se usan para crear una comunicación de tubería en la aplicación.  Estas dos clases se utilizan principalmente para completar la comunicación entre hilos. El objeto PipedInputStream de un hilo puede leer datos del objeto PipedOutputStream de otro hilo. |
| ByteArrayInputStream  Y  ByteArrayOutputStream | Esta clase implementa una secuencia de salida en la que los datos se escriben en una matriz de bytes. El búfer cambiará con los datos.**Escribe continuamente y crece automáticamente**  se utiliza para almacenar en caché los datos (el objetivo de la escritura de datos (el significado original de la secuencia de salida)), escribir datos en su búfer interno, el búfer crece automáticamente y los datos se pueden extraer de él cuando se completa la escritura |
| StringBuffer | Los métodos de StringBuffer están sincronizados   ayuda a crear un flujo de entrada donde se pueden leer bytes de la cadena. Si usamos esta clase, solo podemos leer los 8 bits inferiores de cada carácter presente en la cadena. |

**Flujos de caracteres**

|  |  |
| --- | --- |
| Clase | Descripción |
| InputStreamReader  Y  OutputStreamWriter | InputStreamReader es un Reader. Se comporta igual que in Reader y se puede poner en cualquier sitio que admita un Reader. Es decir, podemos leer de él caracteres. Cuando a **InputStreamReader** le pedimos caracteres, él le pide al **InputStream** que tiene guardado dentro los bytes, los convierte a caracteres y nos los devuelve.  InputStreamReader y OutputStreamWriter pueden usar cualquier instancia de la subclase InputStream y OutputStream como variables al construir el objeto. InputStreamReader y OutputStreamWriter utilizan la codificación de caracteres predeterminada del sistema para la conversión de caracteres al acceder, o puede especificar la codificación de caracteres usted mismo. |
| BufferedReader  Y  BufferedWriter | BufferedReader es una clase de Java para leer el texto de una secuencia de entrada (como un archivo) almacenando en el búfer caracteres que leen a la perfección caracteres, matrices o líneas.  En general, cada solicitud de lectura hecha de un Reader hace que se realice una solicitud de lectura correspondiente del carácter subyacente o secuencia de bytes.  Clase que nos permite escribir texto en un Outputstream, utilizando un buffer para proporcionar una escritura eficiente de caracteres, arrays y strings. |
| FilterReader  Y  FilterWriter | FileReader y FileWriter son clases que se suelen usar para leer y escribir archivos de texto al usar 16 bits para el flujo de caracteres BufferedReader puede mejorar el rendimiento utilizando el bufer para la lectura de esos caracteres |
| CharArrayReader  Y  CharArrayWriter | Debido a que los caracteres en Java son caracteres Unicode, a través de esta función, puede leer el contenido del archivo de texto en la matriz de caracteres, acceder aleatoriamente a la matriz de caracteres y luego volver a escribirlo en el archivo. De esta manera, el archivo de texto también puede lograr una función similar al acceso aleatorio. |
| PipedReader  Y  PipedWriter | Pipedwriter es un flujo de salida de tubería de caracteres que se hereda en el escritor. PipedReader es un flujo de entrada de tubería de caracteres que hereda en el escritor. El papel de Pipedwriter y PipedReader es comunicarse entre hilos a través de la tubería. Cuando se utiliza la comunicación de tuberías, debe usarse Pipedwriter y PipeDReader. |
| StringReader  Y  StringWriter | StringReader es para leer, leer de una cadena, por lo que se necesita una cadena, que se pasa a través del método de construcción 2. StringWriter escribe y escribe en un String, por lo que proporciona un StringBuffer internamente para almacenar datos |
|  |  |

**BufferedReader**

|  |  |
| --- | --- |
| Método | Descripción |
| read | Lee caracteres. |
| readLine | Lee una línea de texto |
| ready | Indica si esta transmisión está lista para leerse. Un flujo de caracteres almacenado en búfer está listo si el búfer no está vacío o si el flujo de caracteres subyacente está listo |
| reset | Restablece la transmisión a la marca más reciente. |
| close | Cierra la transmisión y libera los recursos del sistema asociados a ella. Una vez que se ha cerrado la transmisión, las invocaciones adicionales read (), ready (), mark (), reset () o skip () arrojarán una IOException. |
| Mark | Marca la posición actual en la secuencia. Las siguientes llamadas a reset () intentarán reposicionar la transmisión hasta este punto. |
|  |  |

**BufferedWriter**

|  |  |
| --- | --- |
| Método | Descripción |
| write | Escribe una cadena de caracteres |
| newLine | Escribe una línea de separación |
| flush | “Enjuaga” el stream |
| close | Cierra el write y lo “enjuaga” antes |
|  |  |
|  |  |
|  |  |

**FileReader**

|  |  |
| --- | --- |
| Método | Descripción |
| FileReader.abort() | Interrumpe la operación de lectura. A su regreso readyState será DONE |
| FileReader.readAsArrayBuffer() | Comienza la lectura del contenido del objeto Blob especificado, una vez terminada, el atributo result contiene un  ArrayBuffer representando los datos del fichero. |
| FileReader.readAsBinaryString() | Comienza la lectura del contenido del objeto Blob, una vez terminada, el atributo result contiene los datos binarios en bruto del archivo como una cadena. |
| FileReader.readAsDataURL() | Comienza la lectura del contenido del objeto Blob, una vez terminada, el atributo result contiene un data: URL que representa los datos del fichero. |
| FileReader.readAsText() | Comienza la lectura del contenido del objeto Blob, una vez terminada, el atributo result contiene el contenido del fichero como una cadena de texto. |
|  |  |

**FileWriter**

|  |  |
| --- | --- |
| Método | Descripción |
| ****write ()**** | Escribe posiciones de los caracteres |
| ****getEncoding()**** | Este método se usa para obtener el tipo de codificación que se usa para escribir los datos. |
| ****close()**** | Después de terminar de escribir caracteres en un FileWriter, deberíamos cerrarlo. |

**InputStreamReader**

|  |  |
| --- | --- |
| Método | Descripción |
| getEncoding() | Devuelve el nombre de la codificación de caracteres que utiliza esta secuencia. Devuelve el nombre histórico o null si está vacio.  Lanza ioexception |
| read () | Lee caracteres. Lanza ioexception |
| ready () | Indica si esta transmisión está lista para leerse. Un InputStreamReader está listo si su búfer de entrada no está vacío o si hay bytes disponibles para leer desde el flujo de bytes subyacente. Devuelve verdadero si se garantiza que el siguiente read () no se bloqueará para la entrada, falso en caso contrario |
| close () | Cierra la transmisión y libera los recursos del sistema asociados a ella. Una vez que se ha cerrado la transmisión, las invocaciones adicionales read (), ready (), mark (), reset () o skip () arrojarán una IOException. |

**OutputStreamWriter**

|  |  |
| --- | --- |
| Método | Descripción |
| getEncoding() | Devuelve el nombre de la codificación de caracteres que utiliza esta secuencia. Devuelve el nombre histórico o null si está vacio.  Lanza ioexception |
| Write() | Escribe una porción o una cantidad de caracteres |
| Flush() | Limpia la interfaz |
| Close() | Cierra el output o el fichero |
|  |  |

**FileInputStream**

|  |  |
| --- | --- |
| Método | Descripción |
| **available**() | Devuelve una estimación del número de bytes restantes que se pueden leer (u omitir) de este flujo de entrada sin que se bloquee mediante la siguiente invocación de un método para este flujo de entrada. |
| **close**() | Cierra este flujo de entrada de archivo y libera los recursos del sistema asociados con el flujo. |
| **finalize**() | Garantiza que close llame al método de este flujo de entrada de archivo cuando no haya más referencias a él. |
| **getChannel**() | Devuelve el FileChannel objeto único asociado con este flujo de entrada de archivo. |
| ****skip****(long n) | Omite y descarta bytes de datos del flujo de entrada. |
| Read() | Lee longitud de flujos de datos de bytes |
|  |  |

**FileOuputStream**

|  |  |
| --- | --- |
| Método | Descripción |
| **close**() | Cierra este flujo de salida de archivo y libera los recursos del sistema asociados con este flujo. |
| ****finalize****() | Limpia la conexión al archivo y garantiza que closese llame al método de este flujo de salida de archivo cuando no haya más referencias a este flujo. |
| **getChannel**() | Devuelve el FileChannel objeto único asociado con este flujo de salida de archivo. |
| ****getFD****() | Devuelve el descriptor de archivo asociado con esta secuencia. |
| ****write**** | Escribe una longitud de bytes en donde se le puede especificar en que parte queremos empezar a escribir |
|  |  |
|  |  |