1.- Trabajar los flujos de datos.

Comprender como y cuando utilizar un determinado flujo de datos para ello, trabajar haciendo ejercicio con los siguientes métodos y clases

- Buffered Reader
- InputStream
- OutputStream
- ObjectInputStream (WriteObject / ReadObject)
- ObjectOutputStream
- DataInputStream
- DataOutputStream
- BufferedInputStream
- BufferedOutputStream
- FileInputStream
- FileOutputStream

Un 'stream' o **flujo** es una abstracción implementada en Java8 para enviar una secuencia de datos entre una fuente y un destino de información. Tanto la fuente como el destino pueden ser ficheros en disco, sockets de red o procesos; entre otros. Los 'streams' soportan varios tipos de datos, como pueden ser los bytes simples, datos primitivos e incluso objetos de Java.

La acción de leer desde una fuente es conocida como *Input*, mientras que la acción de escribir a un destino se la llama *Output*; ambos procesos quedan definidos dentro del paquete *Java.io*. Los 'streams' son unidireccionales, lo que significa que un flujo se podrá usar como lectura o como escritura, pero no para ambas acciones al mismo tiempo.

Existen tres flujos estándar en Java:

- Flujo de entrada (System.in): Instancia de la superclase *InputStream*.
- Flujo de salida (System.out): Instancia de la superclase *OutputStream*.
- Flujo de error (System.err): Se emplea para lanzar mensajes a un log o consola.

Existen multitud de tipos de 'streams', los cuales pueden ser clasificados en dos grandes grupos dependiendo de la representación de la información que manipulan:

- **Streams' orientados a carácter**. Operan con caracteres de 2 bytes (16 bits) como unidad de trabajo. Suelen ser utilizados para leer y escribir información almacenada en texto, como archivos con extensión TXT, CSV o XML; entre otros. La superclase abstracta utilizada para leer 'streams' orientados a carácter es la clase *Reader*, mientras que la superclase abstracta para escribir 'streams' orientados a carácter es la clase *Writer*.

- **Streams' orientados a byte**. Operan con bytes (8 bits) dispuestos en forma unitaria. Se emplean para leer y escribir información almacenada en forma binaria, como imágenes o audio; entre otros. La superclase abstracta encargada de leer 'streams' orientados a byte es la clase *InputStream*, mientras que la superclase abstracta utilizada para escribir 'streams' orientados a byte es la clase *OutputStream*.

TIPOS DE STREAMS

Clase 'InputStream':

Es la superclase de la que heredan todos los 'streams' de entrada binarios declarando los métodos para leer los datos desde una fuente concreta.

MÉTODOS

Constructor: InputStream() *crea un objeto de la clase InputStream*

Methods

Modifier and Type	Method and Description
`int`	`available()` Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
`void`	`close()` Closes this input stream and releases any system resources associated with the stream.
`void`	`mark(int readlimit)` Marks the current position in this input stream.
`boolean`	`markSupported()` Tests if this input stream supports the `mark` and `reset` methods.
`abstract int`	`read()` Reads the next byte of data from the input stream.
`int`	`read(byte[] b)` Reads some number of bytes from the input stream and stores them into the buffer array `b`.
`int`	`read(byte[] b, int off, int len)` Reads up to `len` bytes of data from the input stream into an array of bytes.
`void`	`reset()` Repositions this stream to the position at the time the `mark` method was last called on this input stream.
`long`	`skip(long n)` Skips over and discards `n` bytes of data from this input stream.

Clase 'OutputStream':

De manera análoga a lo que hace la clase 'InputStream', ésta es la superclase de la que heredan todos los 'streams' de salida binarios que aceptan bytes de salida y los manda a su destino, proporcionando los métodos para manejar el flujo.

Métodos

Constructor: OutputStream() *crea un objeto de la clase OutputStream*

Methods

Modifier and Type	Method and Description
`void`	`close()` Closes this output stream and releases any system resources associated with this stream.
`void`	`flush()` Flushes this output stream and forces any buffered output bytes to be written out.
`void`	`write(byte[] b)` Writes `b.length` bytes from the specified byte array to this output stream.
`void`	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to this output stream.
`abstract void`	`write(int b)` Writes the specified byte to this output stream.

Clase 'BufferedReader':

Clase derivada de 'Reader' que lee el texto desde un flujo de entrada de caracteres, almacenándolos en un buffer para proporcionar una lectura eficiente de caracteres, matrices y líneas.

Constructores:

Constructors

Constructor and Description
`BufferedReader(Reader in)` Creates a buffering character-input stream that uses a default-sized input buffer.
`BufferedReader(Reader in, int sz)` Creates a buffering character-input stream that uses an input buffer of the specified size.

Métodos:

Methods

Modifier and Type	Method and Description
`void`	`close()` Closes the stream and releases any system resources associated with it.
`void`	`mark(int readAheadLimit)` Marks the present position in the stream.
`boolean`	`markSupported()` Tells whether this stream supports the mark() operation, which it does.
`int`	`read()` Reads a single character.
`int`	`read(char[] cbuf, int off, int len)` Reads characters into a portion of an array.
`String`	`readLine()` Reads a line of text.
`boolean`	`ready()` Tells whether this stream is ready to be read.
`void`	`reset()` Resets the stream to the most recent mark.
`long`	`skip(long n)` Skips characters.

En general, cada solicitud de lectura realizada por un lector provoca que se realice una solicitud de lectura correspondiente del carácter subyacente o del flujo de bytes. Por lo tanto, es aconsejable envolver un 'BufferedReader' alrededor de cualquier 'Reader' cuyas operaciones *read()* puedan ser costosas, como las clases 'FileReader' o 'InputStreamReader'. Sin almacenamiento en búfer, cada invocación de los módulos *read()* o *readLine()* podría hacer que se lean bytes del archivo, se conviertan en caracteres y luego se devuelvan, lo que puede ser muy ineficiente.
Los programas que usan 'DataInputStream' para la entrada de texto se pueden localizar reemplazando cada 'DataInputStream' con un 'BufferedReader' apropiado.

Clase 'PrintWriter':

Clase derivada de 'Writer' que imprime representaciones formateadas de objetos en un flujo de salida de texto. Esta clase implementa todos los métodos de impresión que se encuentran en la clase 'PrintStream', por lo que no contiene métodos para escribir bytes sin procesar.
Los programas que usan la clase 'DataInputStream' para la entrada de texto se pueden localizar reemplazando cada 'DataInputStream' con un 'BufferedReader' apropiado. A diferencia de la clase 'PrintStream', si el vaciado automático está habilitado, se realizará solo cuando se invoque uno de los métodos *println()*, *printf()* o *format()*, en lugar de cuando se genere un carácter de nueva línea. Estos

métodos utilizan la propia noción de separador de línea de la plataforma en lugar del carácter de nueva línea.
Los métodos de esta clase nunca arrojan excepciones de E/S, aunque algunos de sus constructores pueden hacerlo. El cliente puede preguntar si se han producido errores invocando *checkError()*.

Constructores:

Constructors

Constructor and Description
`PrintWriter(File file)` Creates a new PrintWriter, without automatic line flushing, with the specified file.
`PrintWriter(File file, String csn)` Creates a new PrintWriter, without automatic line flushing, with the specified file and charset.
`PrintWriter(OutputStream out)` Creates a new PrintWriter, without automatic line flushing, from an existing OutputStream.
`PrintWriter(OutputStream out, boolean autoFlush)` Creates a new PrintWriter from an existing OutputStream.
`PrintWriter(String fileName)` Creates a new PrintWriter, without automatic line flushing, with the specified file name.
`PrintWriter(String fileName, String csn)` Creates a new PrintWriter, without automatic line flushing, with the specified file name and charset.
`PrintWriter(Writer out)` Creates a new PrintWriter, without automatic line flushing.
`PrintWriter(Writer out, boolean autoFlush)` Creates a new PrintWriter.

Métodos:

Methods

Modifier and Type	Method and Description
`PrintWriter`	`append(char c)` Appends the specified character to this writer.
`PrintWriter`	`append(CharSequence csq)` Appends the specified character sequence to this writer.
`PrintWriter`	`append(CharSequence csq, int start, int end)` Appends a subsequence of the specified character sequence to this writer.
`boolean`	`checkError()` Flushes the stream if it's not closed and checks its error state.
`protected void`	`clearError()` Clears the error state of this stream.
`void`	`close()` Closes the stream and releases any system resources associated with it.
`void`	`flush()` Flushes the stream.
`PrintWriter`	`format(Locale l, String format, Object... args)` Writes a formatted string to this writer using the specified format string and arguments.
`PrintWriter`	`format(String format, Object... args)` Writes a formatted string to this writer using the specified format string and arguments.
`void`	`print(boolean b)` Prints a boolean value.
`void`	`print(char c)` Prints a character.
`void`	`print(char[] s)` Prints an array of characters.
`void`	`print(double d)` Prints a double-precision floating-point number.

void	print(float f) Prints a floating-point number.
void	print(int i) Prints an integer.
void	print(long l) Prints a long integer.
void	print(Object obj) Prints an object.
void	print(String s) Prints a string.
PrintWriter	printf(Locale l, String format, Object... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.
PrintWriter	printf(String format, Object... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.
void	println() Terminates the current line by writing the line separator string.
void	println(boolean x) Prints a boolean value and then terminates the line.
void	println(char x) Prints a character and then terminates the line.
void	println(char[] x) Prints an array of characters and then terminates the line.
void	println(double x) Prints a double-precision floating-point number and then terminates the line.
void	println(float x) Prints a floating-point number and then terminates the line.

void	println(int x) Prints an integer and then terminates the line.
void	println(long x) Prints a long integer and then terminates the line.
void	println(Object x) Prints an Object and then terminates the line.
void	println(String x) Prints a String and then terminates the line.
protected void	setError() Indicates that an error has occurred.
void	write(char[] buf) Writes an array of characters.
void	write(char[] buf, int off, int len) Writes A Portion of an array of characters.
void	write(int c) Writes a single character.
void	write(String s) Writes a string.
void	write(String s, int off, int len) Writes a portion of a string.

Clase 'BufferedInputStream':

Clase derivada de 'InputStream' que realiza el mismo proceso que su clase padre, pero mediante el uso de un búfer de memoria, el cual genera un flujo continuo entre la fuente y el destino para que la información se contenga y almacene en dicho búfer, realizando una gestión más eficiente de los recursos del ordenador. Se gana en rapidez de la información y acceso a los datos con respecto a la clase padre al leer mayor cantidad de bytes de golpe, siendo recomendable para flujos con un elevado volumen de trabajo. Esta clase pertenece al grupo de los 'streams' de filtro, ya que se construyen sobre la base de otro flujo que la envuelve.

Constructores:

Constructors

Constructor and Description
BufferedInputStream(InputStream in) Creates a BufferedInputStream and saves its argument, the input stream in, for later use.
BufferedInputStream(InputStream in, int size) Creates a BufferedInputStream with the specified buffer size, and saves its argument, the input stream in, for later use.

Métodos:

Methods

Modifier and Type	Method and Description
int	available() Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
void	close() Closes this input stream and releases any system resources associated with the stream.
void	mark(int readlimit) See the general contract of the mark method of InputStream.
boolean	markSupported() Tests if this input stream supports the mark and reset methods.
int	read() See the general contract of the read method of InputStream.
int	read(byte[] b, int off, int len) Reads bytes from this byte-input stream into the specified byte array, starting at the given offset.
void	reset() See the general contract of the reset method of InputStream.
long	skip(long n) See the general contract of the skip method of InputStream.

Clase 'BufferedOutputStream':

Clase heredada de 'OutputStream' que se comporta de manera análoga a 'BufferedInputStream' aunque dedicada a los flujos de salida. Crea un búfer de memoria local en el que se almacenan los datos recibidos por otro flujo para poder agilizar el tiempo y el rendimiento computacional de la computadora. Al configurar tal flujo de salida, una aplicación puede escribir bytes en el flujo de salida subyacente sin causar necesariamente una llamada al sistema por cada byte escrito.

Constructores:

Constructors

Constructor and Description
BufferedOutputStream(OutputStream out) Creates a new buffered output stream to write data to the specified underlying output stream.
BufferedOutputStream(OutputStream out, int size) Creates a new buffered output stream to write data to the specified underlying output stream with the specified buffer size.

Métodos:

Methods

Modifier and Type	Method and Description
void	flush() Flushes this buffered output stream.
void	write(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to this buffered output stream.
void	write(int b) Writes the specified byte to this buffered output stream.

Clase 'DataInputStream':

Clase derivada de 'InputStream' que se utiliza para la transmisión de datos de tipo primitivo (*int*, *char*, *double*, *boolean*, etcétera) a través de un flujo. De manera análoga a las clases 'BufferedInputStream' y 'BufferedOutputStream', esta clase y la que sigue también son clasificadas como 'streams' de filtro, ya que envuelve a otro flujo inicial, ya sea de entrada o de salida, proporcionando métodos para la lectura de tipos primitivos de un modo independiente de la máquina. El punto fuerte de estos 'strems' radica en la diversidad de métodos *readXXX()* y *writeXXX()* que poseen, uno por cada tipo primitivo permitido. UTF (Unicode Transformation Format) es un convenio empleado para transmitir cadenas de texto de 16 bits.

Constructor:

Constructors

Constructor and Description
`DataInputStream(InputStream in)` Creates a DataInputStream that uses the specified underlying InputStream.

Métodos:

Methods

Modifier and Type	Method and Description
`int`	`read(byte[] b)` Reads some number of bytes from the contained input stream and stores them into the buffer array `b`.
`int`	`read(byte[] b, int off, int len)` Reads up to `len` bytes of data from the contained input stream into an array of bytes.
`boolean`	`readBoolean()` See the general contract of the `readBoolean` method of `DataInput`.
`byte`	`readByte()` See the general contract of the `readByte` method of `DataInput`.
`char`	`readChar()` See the general contract of the `readChar` method of `DataInput`.
`double`	`readDouble()` See the general contract of the `readDouble` method of `DataInput`.
`float`	`readFloat()` See the general contract of the `readFloat` method of `DataInput`.
`void`	`readFully(byte[] b)` See the general contract of the `readFully` method of `DataInput`.
`void`	`readFully(byte[] b, int off, int len)` See the general contract of the `readFully` method of `DataInput`.
`int`	`readInt()` See the general contract of the `readInt` method of `DataInput`.
`long`	`readLong()` See the general contract of the `readLong` method of `DataInput`.
`short`	`readShort()` See the general contract of the `readShort` method of `DataInput`.
`int`	`readUnsignedByte()` See the general contract of the `readUnsignedByte` method of `DataInput`.
`int`	`readUnsignedShort()` See the general contract of the `readUnsignedShort` method of `DataInput`.
`String`	`readUTF()` See the general contract of the `readUTF` method of `DataInput`.
`static String`	`readUTF(DataInput in)` Reads from the stream `in` a representation of a Unicode character string encoded in modified UTF-8 format; this string of characters is then returned as a `String`.
`int`	`skipBytes(int n)` See the general contract of the `skipBytes` method of `DataInput`.

Clase 'DataOutputStream':

Clase derivada de 'OutputStream' que, de idéntica forma que su análoga 'DataInputStream', se utiliza para la transmisión de datos de tipo primitivo a través de un flujo de salida hacia su destino.

Constructor:

Constructors

Constructor and Description
`DataOutputStream(OutputStream out)` Creates a new data output stream to write data to the specified underlying output stream.

Métodos:

Methods

Modifier and Type	Method and Description
void	flush() Flushes this data output stream.
int	size() Returns the current value of the counter written, the number of bytes written to this data output stream so far.
void	write(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to the underlying output stream.
void	write(int b) Writes the specified byte (the low eight bits of the argument b) to the underlying output stream.
void	writeBoolean(boolean v) Writes a boolean to the underlying output stream as a 1-byte value.
void	writeByte(int v) Writes out a byte to the underlying output stream as a 1-byte value.
void	writeBytes(String s) Writes out the string to the underlying output stream as a sequence of bytes.
void	writeChar(int v) Writes a char to the underlying output stream as a 2-byte value, high byte first.
void	writeChars(String s) Writes a string to the underlying output stream as a sequence of characters.
void	writeDouble(double v) Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.
void	writeFloat(float v) Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.
void	writeInt(int v) Writes an int to the underlying output stream as four bytes, high byte first.
void	writeLong(long v) Writes a long to the underlying output stream as eight bytes, high byte first.
void	writeShort(int v) Writes a short to the underlying output stream as two bytes, high byte first.
void	writeUTF(String str) Writes a string to the underlying output stream using modified UTF-8 encoding in a machine-independent manner.

Clase 'ObjectInputStream':

Clase derivada de 'InputStream' que permite el flujo de entrada de datos pertenecientes a objetos Java serializados en una secuencia de bytes (IMPORTANTE: para poder usar este 'stream', la clase que define los objetos a enviar debe implementar la interfaz *Serializable*) que hayan sido escritos previamente usando un flujo 'ObjectOutputStream'. Esta pareja de 'streams' pueden proporcionar una aplicación con almacenamiento persistente para gráficos de objetos empleados con 'FileOutputStream' y 'FileInputStream'. También se pueden emplear para pasar objetos entre 'hosts' usando flujos de 'sockets.

'ObjectInputStream' asegura que los tipos de todos los objetos en el gráfico creado a partir de la secuencia coincidan con las clases presentes en la máquina virtual Java. Las clases se cargan según sea necesario utilizando los mecanismos estándar. El método *readObject()* se usa para leer un objeto de la secuencia, al que se le debe realizar una conversión implícita para obtener el tipo deseado. En Java, las cadenas y las matrices son objetos y se tratan como objetos durante la serialización; por lo que, cuando son leídos, deben convertirse al tipo esperado.

Leer un objeto es análogo a ejecutar los constructores de un nuevo objeto. La memoria se asigna para el objeto y se inicializa a NULL. Se invocan constructores sin argumentos para las clases no serializables y luego los campos de las clases serializables

se restauran desde la secuencia comenzando con la clase serializable más cercana a *java.lang.object* y terminando con la clase más específica del objeto.

Constructores:

Constructors

Modifier	Constructor and Description
protected	ObjectInputStream() Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	ObjectInputStream(InputStream in) Creates an ObjectInputStream that reads from the specified InputStream.

Métodos:

Methods

Modifier and Type	Method and Description
int	available() Returns the number of bytes that can be read without blocking.
void	close() Closes the input stream.
void	defaultReadObject() Read the non-static and non-transient fields of the current class from this stream.
protected boolean	enableResolveObject(boolean enable) Enable the stream to allow objects read from the stream to be replaced.
int	read() Reads a byte of data.
int	read(byte[] buf, int off, int len) Reads into an array of bytes.
boolean	readBoolean() Reads in a boolean.
byte	readByte() Reads an 8 bit byte.
char	readChar() Reads a 16 bit char.
protected ObjectStreamClass	readClassDescriptor() Read a class descriptor from the serialization stream.
double	readDouble() Reads a 64 bit double.
ObjectInputStream.GetField	readFields() Reads the persistent fields from the stream and makes them available by name.
float	readFloat() Reads a 32 bit float.
void	readFully(byte[] buf) Reads bytes, blocking until all bytes are read.
void	readFully(byte[] buf, int off, int len) Reads bytes, blocking until all bytes are read.
int	readInt() Reads a 32 bit int.
long	readLong() Reads a 64 bit long.
Object	readObject() Read an object from the ObjectInputStream.
protected Object	readObjectOverride() This method is called by trusted subclasses of ObjectOutputStream that constructed ObjectOutputStream using the protected no-arg constructor.
short	readShort() Reads a 16 bit short.
protected void	readStreamHeader() The readStreamHeader method is provided to allow subclasses to read and verify their own stream headers.
Object	readUnshared() Reads an "unshared" object from the ObjectInputStream.
int	readUnsignedByte() Reads an unsigned 8 bit byte.
int	readUnsignedShort() Reads an unsigned 16 bit short.
String	readUTF() Reads a String in modified UTF-8 format.
void	registerValidation(ObjectInputValidation obj, int prio) Register an object to be validated before the graph is returned.
protected Class<?>	resolveClass(ObjectStreamClass desc) Load the local class equivalent of the specified stream class description.
protected Object	resolveObject(Object obj) This method will allow trusted subclasses of ObjectInputStream to substitute one object for another during deserialization.
protected Class<?>	resolveProxyClass(String[] interfaces) Returns a proxy class that implements the interfaces named in a proxy class descriptor; subclasses may implement this method to read custom data from the stream along with the descriptors for dynamic proxy classes, allowing them to use an alternate loading mechanism for the interfaces and the proxy class.
int	skipBytes(int len) Skips bytes.

Clase 'ObjectOutputStream':

Clase derivada de 'OutputStream' que, de idéntica forma que su análoga 'ObjectInputStream', permite el flujo de salida de datos pertenecientes a objetos Java serializados en una secuencia de bytes.

El método *writeObject()* se usa para escribir un objeto en la secuencia. Cualquier objeto, incluidas las cadenas y las matrices, se puede escribir con este método. Se pueden escribir varios objetos o primitivas en la secuencia. Los objetos deben leerse desde el flujo 'ObjectInputstream' correspondiente con los mismos tipos y en el mismo orden en que fueron escritos.

Constructores:

Constructors

Modifier	Constructor and Description
protected	ObjectOutputStream() Provide a way for subclasses that are completely reimplementing ObjectOutputStream to not have to allocate private data just used by this implementation of ObjectOutputStream.
	ObjectOutputStream(OutputStream out) Creates an ObjectOutputStream that writes to the specified OutputStream.

Métodos:

Methods

Modifier and Type	Method and Description
protected void	annotateClass(Class<?> cl) Subclasses may implement this method to allow class data to be stored in the stream.
protected void	annotateProxyClass(Class<?> cl) Subclasses may implement this method to store custom data in the stream along with descriptors for dynamic proxy classes.
void	close() Closes the stream.
void	defaultWriteObject() Write the non-static and non-transient fields of the current class to this stream.
protected void	drain() Drain any buffered data in ObjectOutputStream.
protected boolean	enableReplaceObject(boolean enable) Enable the stream to do replacement of objects in the stream.
void	flush() Flushes the stream.
ObjectOutputStream.PutField	putFields() Retrieve the object used to buffer persistent fields to be written to the stream.
protected Object	replaceObject(Object obj) This method will allow trusted subclasses of ObjectOutputStream to substitute one object for another during serialization.
void	reset() Reset will disregard the state of any objects already written to the stream.
void	useProtocolVersion(int version) Specify stream protocol version to use when writing the stream.
void	write(byte[] buf) Writes an array of bytes.
void	write(byte[] buf, int off, int len) Writes a sub array of bytes.
void	write(int val) Writes a byte.
void	writeBoolean(boolean val) Writes a boolean.
void	writeByte(int val) Writes an 8 bit byte.
void	writeBytes(String str) Writes a String as a sequence of bytes.
void	writeChar(int val) Writes a 16 bit char.
void	writeChars(String str) Writes a String as a sequence of chars.
protected void	writeClassDescriptor(ObjectStreamClass desc) Write the specified class descriptor to the ObjectOutputStream.
void	writeDouble(double val) Writes a 64 bit double.
void	writeFields() Write the buffered fields to the stream.
void	writeFloat(float val) Writes a 32 bit float.
void	writeInt(int val) Writes a 32 bit int.
void	writeLong(long val) Writes a 64 bit long.
void	writeObject(Object obj) Write the specified object to the ObjectOutputStream.
protected void	writeObjectOverride(Object obj) Method used by subclasses to override the default writeObject method.
void	writeShort(int val) Writes a 16 bit short.
protected void	writeStreamHeader() The writeStreamHeader method is provided so subclasses can append or prepend their own header to the stream.
void	writeUnshared(Object obj) Writes an "unshared" object to the ObjectOutputStream.
void	writeUTF(String str) Primitive data write of this String in modified UTF-8 format.

Clase 'FileInputStream':

Clase derivada de 'InputStream' que obtiene un flujo de entrada de bytes procedente de un sistema de archivos. Los archivos que se encontrarán disponibles dependerá del entorno del anfitrión.

Tanto esta clase como su contraparte 'FileOutputStream' aportan tres constructores cada una, dependiendo de la información aportada para identificar el fichero a abrir:

- Un constructor que toma como argumento un String que es el nombre del fichero que se va a abrir.
- Un constructor que toma un objeto de tipo File que se refiere al fichero.
- Un constructor que toma un objeto de tipo FileDescriptor, que constituye un valor dependiente del sistema de ficheros y que describe un fichero abierto.

Constructores:

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

Métodos:

Methods

Modifier and Type	Method and Description
`int`	`available()` Returns an estimate of the number of remaining bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
`void`	`close()` Closes this file input stream and releases any system resources associated with the stream.
`protected void`	`finalize()` Ensures that the `close` method of this file input stream is called when there are no more references to it.
`FileChannel`	`getChannel()` Returns the unique `FileChannel` object associated with this file input stream.
`FileDescriptor`	`getFD()` Returns the `FileDescriptor` object that represents the connection to the actual file in the file system being used by this `FileInputStream`.
`int`	`read()` Reads a byte of data from this input stream.
`int`	`read(byte[] b)` Reads up to `b.length` bytes of data from this input stream into an array of bytes.
`int`	`read(byte[] b, int off, int len)` Reads up to `len` bytes of data from this input stream into an array of bytes.
`long`	`skip(long n)` Skips over and discards `n` bytes of data from the input stream.

Clase 'FileOutputStream':

Clase derivada de 'OutputStream' que, de manera análoga a la clase 'FileInputStream', obtiene un flujo de salida de bytes dirigido hacia un sistema de archivos.

Constructores:

Constructors

Constructor and Description
`FileOutputStream(File file)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(File file, boolean append)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(FileDescriptor fdObj)` Creates a file output stream to write to the specified file descriptor, which represents an existing connection to an actual file in the file system.
`FileOutputStream(String name)` Creates a file output stream to write to the file with the specified name.
`FileOutputStream(String name, boolean append)` Creates a file output stream to write to the file with the specified name.

Métodos:

Methods

Modifier and Type	Method and Description
`void`	`close()` Closes this file output stream and releases any system resources associated with this stream.
`protected void`	`finalize()` Cleans up the connection to the file, and ensures that the `close` method of this file output stream is called when there are no more references to this stream.
`FileChannel`	`getChannel()` Returns the unique `FileChannel` object associated with this file output stream.
`FileDescriptor`	`getFD()` Returns the file descriptor associated with this stream.
`void`	`write(byte[] b)` Writes `b.length` bytes from the specified byte array to this file output stream.
`void`	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to this file output stream.
`void`	`write(int b)` Writes the specified byte to this file output stream.

2.- Trabajar las interfaces visuales (Componentes Swing)

3.- Conexión con base de datos (Alta y consulta de una tabla)

Material :

Tutorial de java : https://www.tutorialesprogramacionya.com/javaya/index.php?inicio=40

https://www.w3schools.com/java/default.asp

Libro de problemas y ejercicios de Java

https://drive.google.com/file/d/1rG8ily4qI9Dk8zgzZ1xzGNTRkCYmKh9G/view?usp=sharing