
File
+ separatorChar: char
+ separator: String
+ pathSeparatorChar: char
+ pathSeparator: String
- getPrefixLength(): int
+ File(in pathname: String)
+ File(in parent: String, in child: String)
+ File(in parent: File, in child: String)
+ File(in uri: URI)
+ getName(): String
+ getParent(): String
+ getParentFile(): File
+ getPath(): String
+ isAbsolute(): boolean
+ getAbsolutePath(): String
+ getAbsoluteFile(): File
+ getCanonicalPath(): String
+ getCanonicalFile(): File
+ toURL(): URL
+ toURI(): URI
+ canRead(): boolean
+ canWrite(): boolean
+ exists(): boolean
+ isDirectory(): boolean
+ isFile(): boolean
+ isHidden(): boolean
+ lastModified(): long
+ length(): long
+ createNewFile(): boolean
+ delete(): boolean
+ deleteOnExit()
+ list(): String[]
+ list(in filter: FilenameFilter): String[]
+ listFiles(): File[]
+ listFiles(in filter: FilenameFilter): File[]
+ listFiles(in filter: FileFilter): File[]
+ mkdir(): boolean
+ mkdirs(): boolean
+ renameTo(in dest: File): boolean
+ setLastModified(in time: long): boolean
+ setReadOnly(): boolean
+ listRoots(): File[]
+ createTempFile(in prefix: String, in suffix: String, in directory: File): File
+ createTempFile(in prefix: String, in suffix: String): File
+ compareTo(in pathname: File): int
+ compareTo(in o: Object): int
+ equals(in obj: Object): boolean
+ hashCode(): int
+ toString(): String

Field Summary

static String	**pathSeparator** The system-dependent path-separator character, represented as a string for convenience.
static char	**pathSeparatorChar** The system-dependent path-separator character.
static String	**separator** The system-dependent default name-separator character, represented as a string for convenience.
static char	**separatorChar** The system-dependent default name-separator character.

Constructor Summary

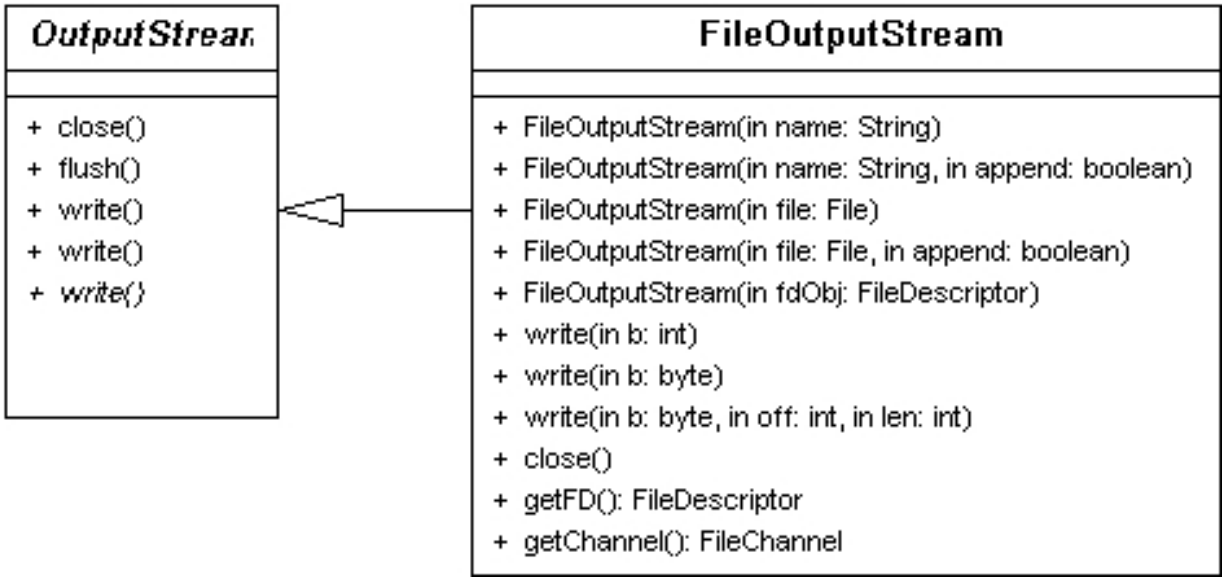
File(File parent, String child) Creates a new File instance from a parent abstract pathname and a child pathname string.
File(String pathname) Creates a new File instance by converting the given pathname string into an abstract pathname.
File(String parent, String child) Creates a new File instance from a parent pathname string and a child pathname string.
File(URI uri) Creates a new File instance by converting the given file: URI into an abstract pathname.

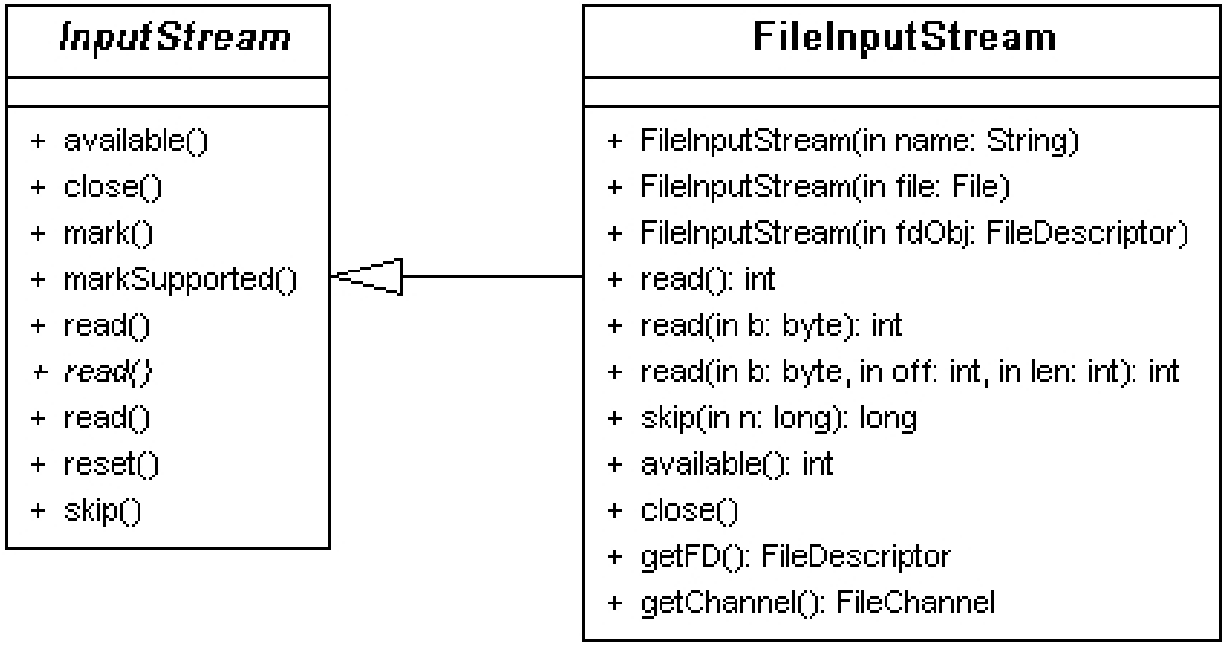
Method Summary

boolean	**canRead**() Tests whether the application can read the file denoted by this abstract pathname.
boolean	**canWrite**() Tests whether the application can modify to the file denoted by this abstract pathname.
int	**compareTo**(File pathname) Compares two abstract pathnames lexicographically.
int	**compareTo**(Object o) Compares this abstract pathname to another object.
boolean	**createNewFile**() Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.

static File	**createTempFile**(String prefix, String suffix, File directory) Creates a new empty file in the specified directory, using the given prefix and suffix strings to generate its name.
boolean	**delete**() Deletes the file or directory denoted by this abstract pathname.
void	**deleteOnExit**() Requests that the file or directory denoted by this abstract pathname be deleted when the virtual machine terminates.
boolean	**equals**(Object obj) Tests this abstract pathname for equality with the given object.
boolean	**exists**() Tests whether the file or directory denoted by this abstract pathname exists.
File	**getAbsoluteFile**() Returns the absolute form of this abstract pathname.
String	**getAbsolutePath**() Returns the absolute pathname string of this abstract pathname.
File	**getCanonicalFile**() Returns the canonical form of this abstract pathname.
String	**getCanonicalPath**() Returns the canonical pathname string of this abstract pathname.
String	**getName**() Returns the name of the file or directory denoted by this abstract pathname.
String	**getParent**() Returns the pathname string of this abstract pathname's parent, or `null` if this pathname does not name a parent directory.
File	**getParentFile**() Returns the abstract pathname of this abstract pathname's parent, or `null` if this pathname does not name a parent directory.
String	**getPath**() Converts this abstract pathname into a pathname string.
int	**hashCode**() Computes a hash code for this abstract pathname.
boolean	**isAbsolute**() Tests whether this abstract pathname is absolute.
boolean	**isDirectory**() Tests whether the file denoted by this abstract pathname is a directory.

boolean	**isFile**() Tests whether the file denoted by this abstract pathname is a normal file.
boolean	**isHidden**() Tests whether the file named by this abstract pathname is a hidden file.
long	**lastModified**() Returns the time that the file denoted by this abstract pathname was last modified.
long	**length**() Returns the length of the file denoted by this abstract pathname.
String[]	**list**() Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.
String[]	**list**(FilenameFilter filter) Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
File[]	**listFiles**() Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.
File[]	**listFiles**(FileFilter filter) Returns an array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
File[]	**listFiles**(FilenameFilter filter) Returns an array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
static File[]	**listRoots**() List the available filesystem roots.
boolean	**mkdir**() Creates the directory named by this abstract pathname.
boolean	**mkdirs**() Creates the directory named by this abstract pathname, including any necessary but nonexistent parent directories.
boolean	**renameTo**(File dest) Renames the file denoted by this abstract pathname.
boolean	**setLastModified**(long time) Sets the last-modified time of the file or directory named by this abstract pathname.
boolean	**setReadOnly**() Marks the file or directory named by this abstract pathname so that only read operations are allowed.
String	**toString**() Returns the pathname string of this abstract pathname.
URI	**toURI**() Constructs a `file:` URI that represents this abstract pathname.
URL	**toURL**() Converts this abstract pathname into a `file:` URL.


OutputStream
+ close()
+ flush()
+ write()
+ write()
+ write()
FileOutputStream
+ FileOutputStream(in name: String)
+ FileOutputStream(in name: String, in append: boolean)
+ FileOutputStream(in file: File)
+ FileOutputStream(in file: File, in append: boolean)
+ FileOutputStream(in fdObj: FileDescriptor)
+ write(in b: int)
+ write(in b: byte)
+ write(in b: byte, in off: int, in len: int)
+ close()
+ getFD(): FileDescriptor
+ getChannel(): FileChannel


InputStream
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ read()
+ read()
+ reset()
+ skip()
FileInputStream
+ FileInputStream(in name: String)
+ FileInputStream(in file: File)
+ FileInputStream(in fdObj: FileDescriptor)
+ read(): int
+ read(in b: byte): int
+ read(in b: byte, in off: int, in len: int): int
+ skip(in n: long): long
+ available(): int
+ close()
+ getFD(): FileDescriptor
+ getChannel(): FileChannel

Constructor Summary

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 an output file 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 an output file stream to write to the file with the specified name.
FileOutputStream(String name, boolean append) Creates an output file stream to write to the file with the specified name.

Method Summary

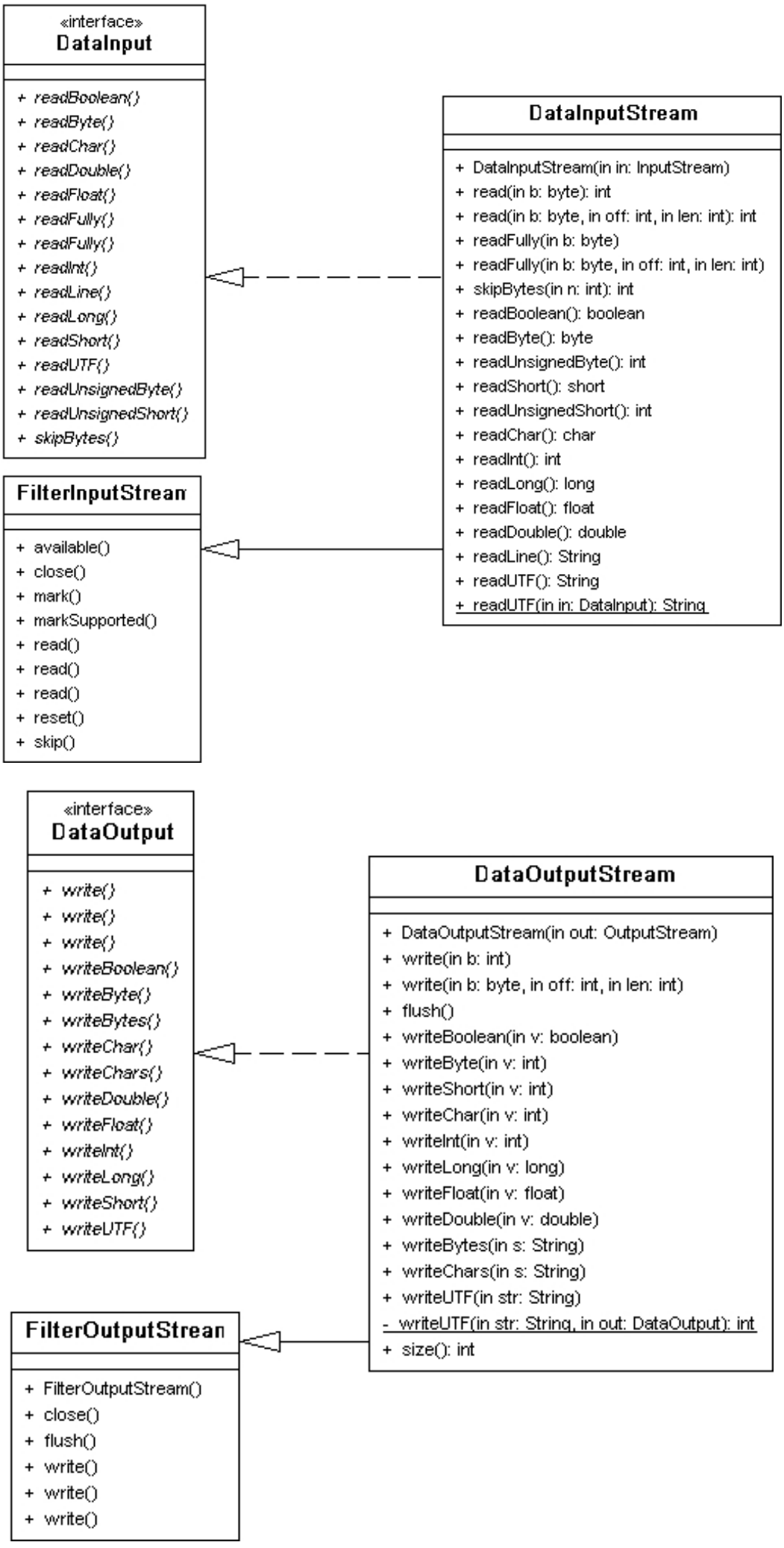
void	**close**() Closes this file output stream and releases any system resources associated with this stream.
protected void	**finalize**() Cleans up the connection to the file, and ensures that the close method of this file output stream is called when there are no more references to this stream.
FileChannel	**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.

Constructor Summary

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.

Method Summary

int	**available**() Returns the number of bytes that can be read from this file input stream without blocking.
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.


«interface»
DataInput
+ readBoolean()
+ readByte()
+ readChar()
+ readDouble()
+ readFloat()
+ readFully()
+ readFully()
+ readInt()
+ readLine()
+ readLong()
+ readShort()
+ readUTF()
+ readUnsignedByte()
+ readUnsignedShort()
+ skipBytes()
DataInputStream
+ DataInputStream(in in: InputStream)
+ read(in b: byte): int
+ read(in b: byte, in off: int, in len: int): int
+ readFully(in b: byte)
+ readFully(in b: byte, in off: int, in len: int)
+ skipBytes(in n: int): int
+ readBoolean(): boolean
+ readByte(): byte
+ readUnsignedByte(): int
+ readShort(): short
+ readUnsignedShort(): int
+ readChar(): char
+ readInt(): int
+ readLong(): long
+ readFloat(): float
+ readDouble(): double
+ readLine(): String
+ readUTF(): String
+ readUTF(in in: DataInput): String
FilterInputStrean
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ read()
+ read()
+ reset()
+ skip()
«interface»
DataOutput
+ write()
+ write()
+ write()
+ writeBoolean()
+ writeByte()
+ writeBytes()
+ writeChar()
+ writeChars()
+ writeDouble()
+ writeFloat()
+ writeInt()
+ writeLong()
+ writeShort()
+ writeUTF()
DataOutputStream
+ DataOutputStream(in out: OutputStream)
+ write(in b: int)
+ write(in b: byte, in off: int, in len: int)
+ flush()
+ writeBoolean(in v: boolean)
+ writeByte(in v: int)
+ writeShort(in v: int)
+ writeChar(in v: int)
+ writeInt(in v: int)
+ writeLong(in v: long)
+ writeFloat(in v: float)
+ writeDouble(in v: double)
+ writeBytes(in s: String)
+ writeChars(in s: String)
+ writeUTF(in str: String)
- writeUTF(in str: String, in out: DataOutput): int
+ size(): int
FilterOutputStrean
+ FilterOutputStream()
+ close()
+ flush()
+ write()
+ write()
+ write()

Constructor Summary

DataInputStream(InputStream in) Creates a DataInputStream that uses the specified underlying InputStream.	

Method Summary

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.

String	**readLine**()
	Deprecated. *This method does not properly convert bytes to characters. As of JDK 1.1, the preferred way to read lines of text is via the* `BufferedReader.readLine()` *method. Programs that use the* `DataInputStream` *class to read lines can be converted to use the* `BufferedReader` *class by replacing code of the form:*
	`DataInputStream d = new DataInputStream(in);`
	with:
	`BufferedReader d = new BufferedReader(new InputStreamReader(in));`

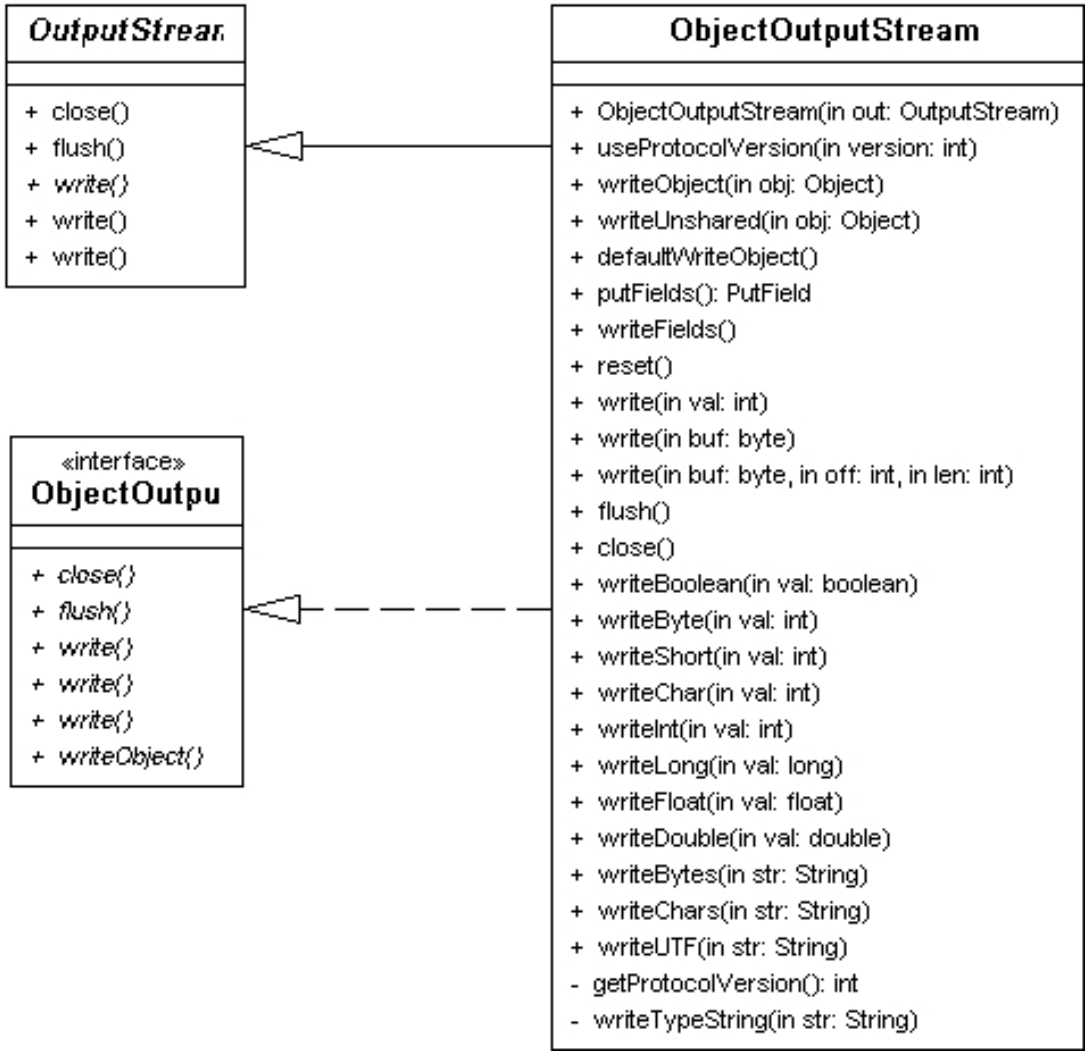
long	**readLong**() See the general contract of the readLong method of DataInput.
short	**readShort**() See the general contract of the readShort method of DataInput.
int	**readUnsignedByte**() See the general contract of the readUnsignedByte method of DataInput.
int	**readUnsignedShort**() See the general 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 Java 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.

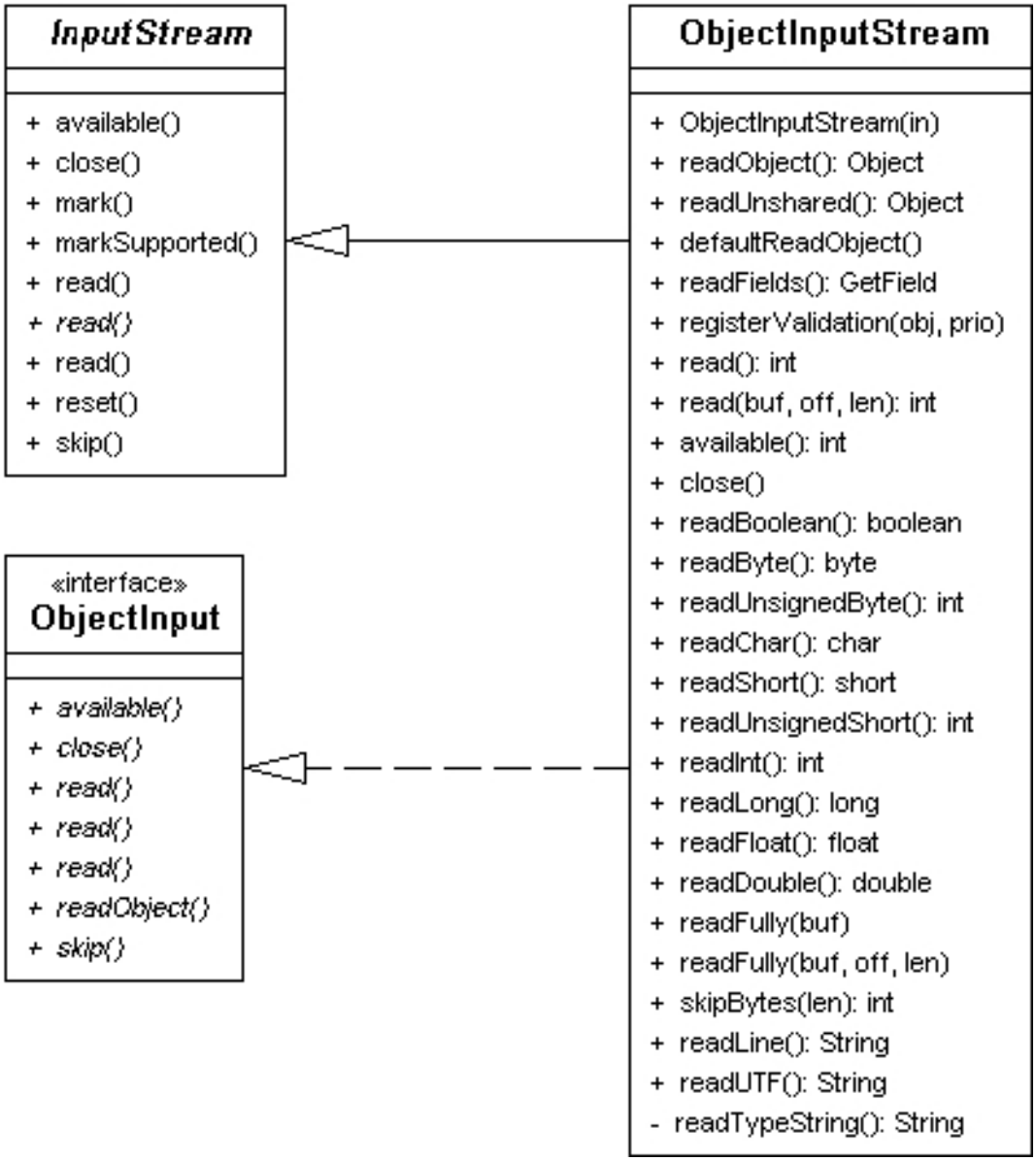
Constructor Summary

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

Method Summary

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 Java modified UTF-8 encoding in a machine-independent manner.


OutputStrear
+ close()
+ flush()
+ write()
+ write()
+ write()
ObjectOutputStream
+ ObjectOutputStream(in out: OutputStream)
+ useProtocolVersion(in version: int)
+ writeObject(in obj: Object)
+ writeUnshared(in obj: Object)
+ defaultWriteObject()
+ putFields(): PutField
+ writeFields()
+ reset()
+ write(in val: int)
+ write(in buf: byte)
+ write(in buf: byte, in off: int, in len: int)
+ flush()
+ close()
+ writeBoolean(in val: boolean)
+ writeByte(in val: int)
+ writeShort(in val: int)
+ writeChar(in val: int)
+ writeInt(in val: int)
+ writeLong(in val: long)
+ writeFloat(in val: float)
+ writeDouble(in val: double)
+ writeBytes(in str: String)
+ writeChars(in str: String)
+ writeUTF(in str: String)
- getProtocolVersion(): int
- writeTypeString(in str: String)
«interface»
ObjectOutpu
+ close()
+ flush()
+ write()
+ write()
+ write()
+ writeObject()


InputStream
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ read()
+ read()
+ reset()
+ skip()
ObjectInputStream
+ ObjectInputStream(in)
+ readObject(): Object
+ readUnshared(): Object
+ defaultReadObject()
+ readFields(): GetField
+ registerValidation(obj, prio)
+ read(): int
+ read(buf, off, len): int
+ available(): int
+ close()
+ readBoolean(): boolean
+ readByte(): byte
+ readUnsignedByte(): int
+ readChar(): char
+ readShort(): short
+ readUnsignedShort(): int
+ readInt(): int
+ readLong(): long
+ readFloat(): float
+ readDouble(): double
+ readFully(buf)
+ readFully(buf, off, len)
+ skipBytes(len): int
+ readLine(): String
+ readUTF(): String
- readTypeString(): String
«interface»
ObjectInput
+ available()
+ close()
+ read()
+ read()
+ read()
+ readObject()
+ skip()

Constructor Summary

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.

Method Summary

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 UTF format.

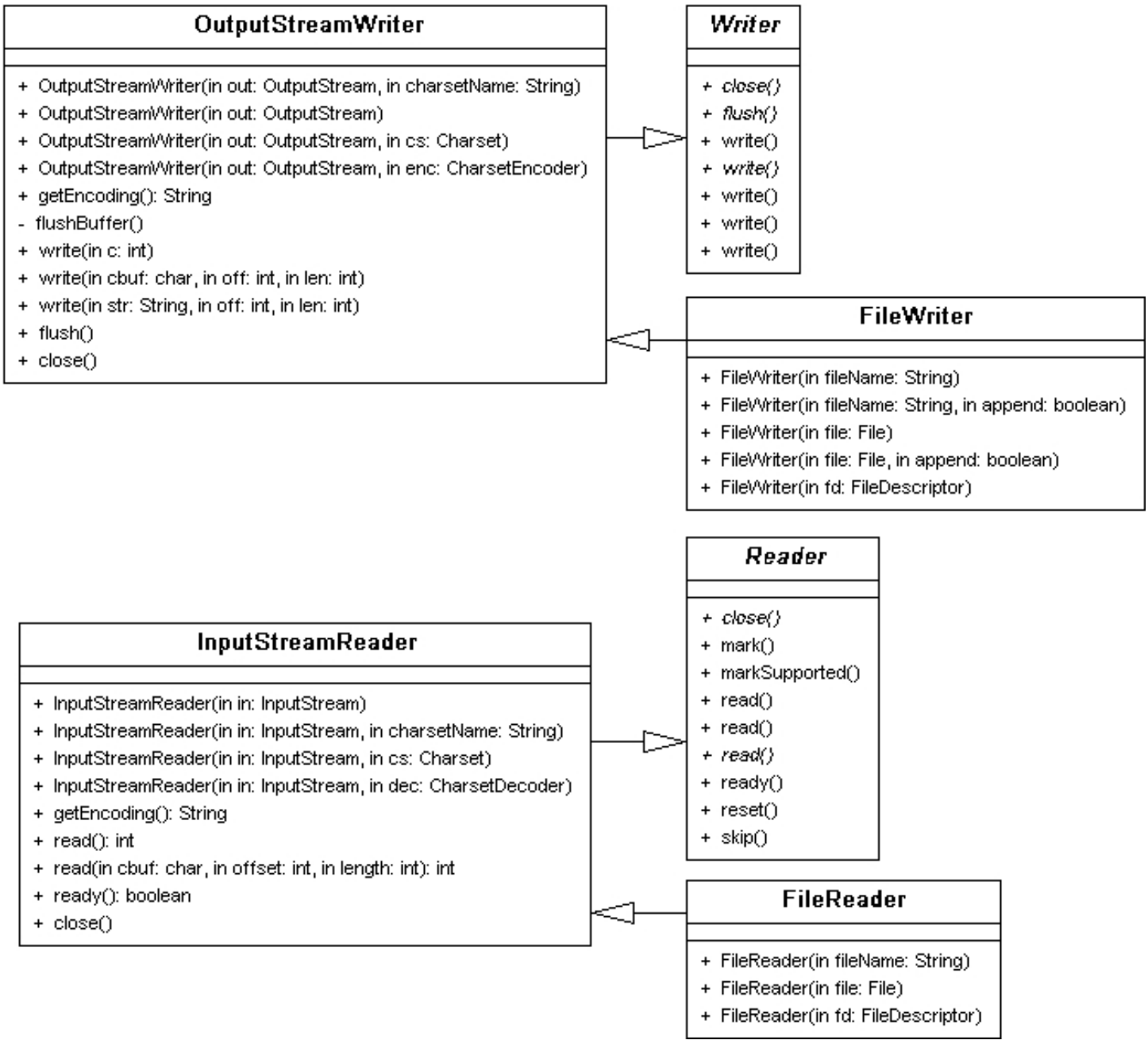
Constructor Summary

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.

Method Summary

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.
String	**readLine**() **Deprecated**. *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.*
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 UTF format String.
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, block until all bytes are skipped.


OutputStreamWriter
+ OutputStreamWriter(in out: OutputStream, in charsetName: String)
+ OutputStreamWriter(in out: OutputStream)
+ OutputStreamWriter(in out: OutputStream, in cs: Charset)
+ OutputStreamWriter(in out: OutputStream, in enc: CharsetEncoder)
+ getEncoding(): String
- flushBuffer()
+ write(in c: int)
+ write(in cbuf: char, in off: int, in len: int)
+ write(in str: String, in off: int, in len: int)
+ flush()
+ close()
Writer
+ close()
+ flush()
+ write()
+ write()
+ write()
+ write()
FileWriter
+ FileWriter(in fileName: String)
+ FileWriter(in fileName: String, in append: boolean)
+ FileWriter(in file: File)
+ FileWriter(in file: File, in append: boolean)
+ FileWriter(in fd: FileDescriptor)
InputStreamReader
+ InputStreamReader(in in: InputStream)
+ InputStreamReader(in in: InputStream, in charsetName: String)
+ InputStreamReader(in in: InputStream, in cs: Charset)
+ InputStreamReader(in in: InputStream, in dec: CharsetDecoder)
+ getEncoding(): String
+ read(): int
+ read(in cbuf: char, in offset: int, in length: int): int
+ ready(): boolean
+ close()
Reader
+ close()
+ mark()
+ markSupported()
+ read()
+ read()
+ read()
+ ready()
+ reset()
+ skip()
FileReader
+ FileReader(in fileName: String)
+ FileReader(in file: File)
+ FileReader(in fd: FileDescriptor)

Constructor Summary

`InputStreamReader(InputStream in)` Create an InputStreamReader that uses the default charset.
`InputStreamReader(InputStream in, Charset cs)` Create an InputStreamReader that uses the given charset.
`InputStreamReader(InputStream in, CharsetDecoder dec)` Create an InputStreamReader that uses the given charset decoder.
`InputStreamReader(InputStream in, String charsetName)` Create an InputStreamReader that uses the named charset.

Method Summary

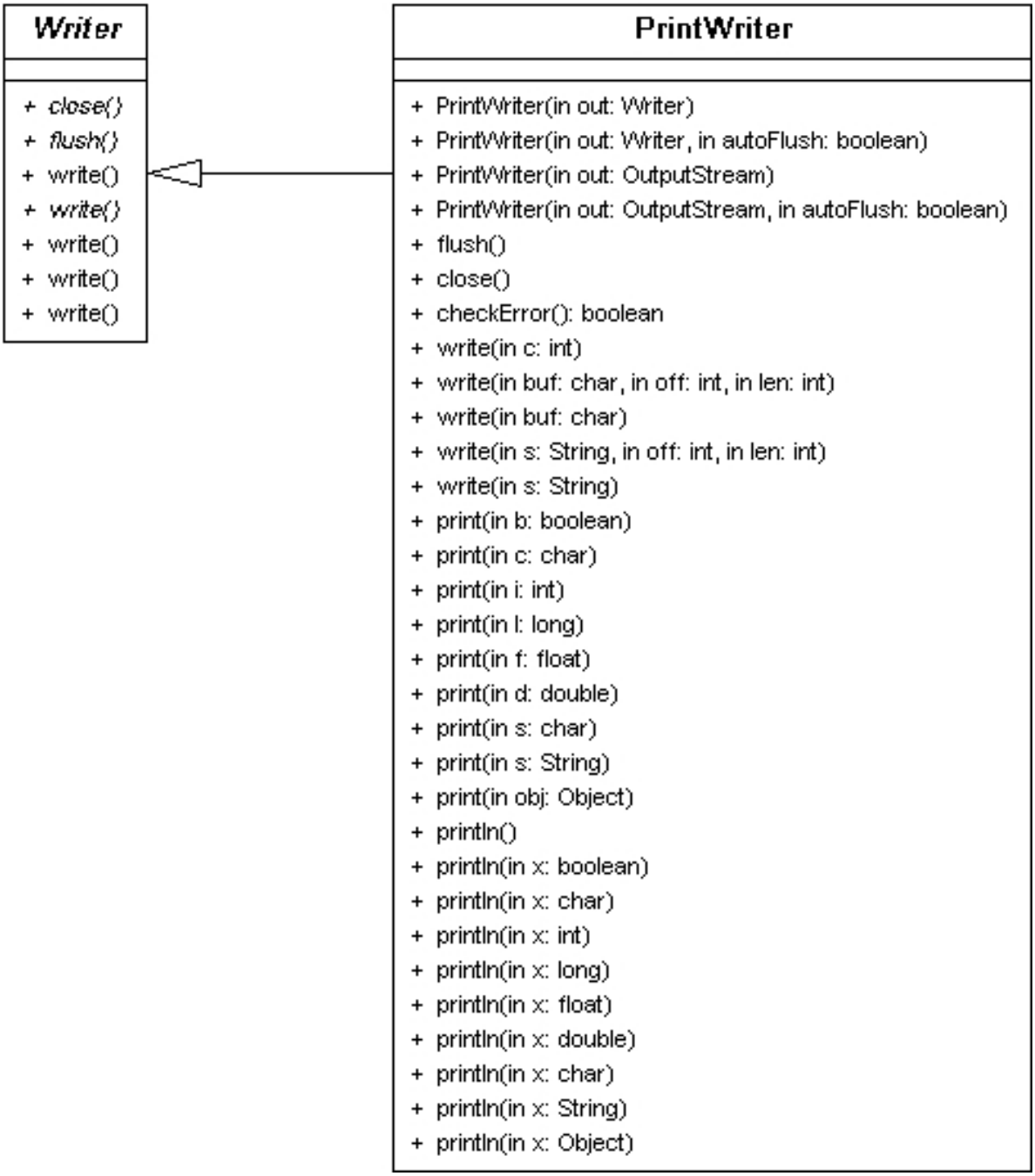
void	`close()` Close the stream.
String	`getEncoding()` Return the name of the character encoding being used by this stream.
int	`read()` Read a single character.
int	`read(char[] cbuf, int offset, int length)` Read characters into a portion of an array.
boolean	`ready()` Tell whether this stream is ready to be read.

Constructor Summary

`OutputStreamWriter(OutputStream out)` Create an OutputStreamWriter that uses the default character encoding.
`OutputStreamWriter(OutputStream out, Charset cs)` Create an OutputStreamWriter that uses the given charset.
`OutputStreamWriter(OutputStream out, CharsetEncoder enc)` Create an OutputStreamWriter that uses the given charset encoder.
`OutputStreamWriter(OutputStream out, String charsetName)` Create an OutputStreamWriter that uses the named charset.

Method Summary

void	`close()` Close the stream.
void	`flush()` Flush the stream.
String	`getEncoding()` Return the name of the character encoding being used by this stream.
void	`write(char[] cbuf, int off, int len)` Write a portion of an array of characters.
void	`write(int c)` Write a single character.
void	`write(String str, int off, int len)` Write a portion of a string.


Writer
+ close()
+ flush()
+ write()
+ write()
+ write()
+ write()
+ write()
PrintWriter
+ PrintWriter(in out: Writer)
+ PrintWriter(in out: Writer, in autoFlush: boolean)
+ PrintWriter(in out: OutputStream)
+ PrintWriter(in out: OutputStream, in autoFlush: boolean)
+ flush()
+ close()
+ checkError(): boolean
+ write(in c: int)
+ write(in buf: char, in off: int, in len: int)
+ write(in buf: char)
+ write(in s: String, in off: int, in len: int)
+ write(in s: String)
+ print(in b: boolean)
+ print(in c: char)
+ print(in i: int)
+ print(in l: long)
+ print(in f: float)
+ print(in d: double)
+ print(in s: char)
+ print(in s: String)
+ print(in obj: Object)
+ println()
+ println(in x: boolean)
+ println(in x: char)
+ println(in x: int)
+ println(in x: long)
+ println(in x: float)
+ println(in x: double)
+ println(in x: char)
+ println(in x: String)
+ println(in x: Object)

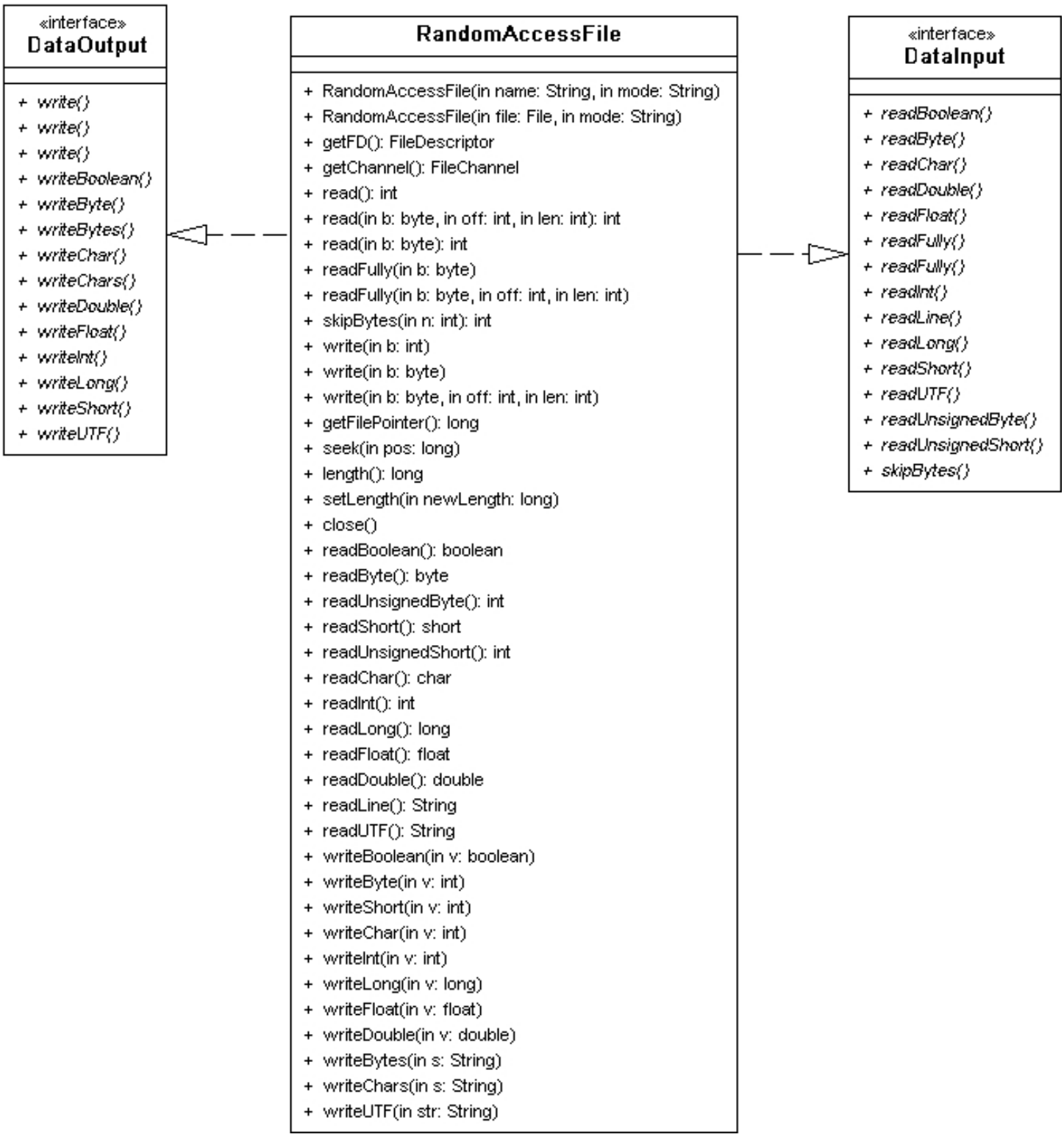
Constructor Summary

`PrintWriter(OutputStream out)` Create a new PrintWriter, without automatic line flushing, from an existing OutputStream.
`PrintWriter(OutputStream out, boolean autoFlush)` Create a new PrintWriter from an existing OutputStream.
`PrintWriter(Writer out)` Create a new PrintWriter, without automatic line flushing.
`PrintWriter(Writer out, boolean autoFlush)` Create a new PrintWriter.

Method Summary

boolean	`checkError()` Flush the stream if it's not closed and check its error state.
void	`close()` Close the stream.
void	`flush()` Flush the stream.
void	`print(boolean b)` Print a boolean value.
void	`print(char c)` Print a character.
void	`print(char[] s)` Print an array of characters.
void	`print(double d)` Print a double-precision floating-point number.
void	`print(float f)` Print a floating-point number.
void	`print(int i)` Print an integer.
void	`print(long l)` Print a long integer.
void	`print(Object obj)` Print an object.

void	print(String s) Print a string.
void	println() Terminate the current line by writing the line separator string.
void	println(boolean x) Print a boolean value and then terminate the line.
void	println(char x) Print a character and then terminate the line.
void	println(char[] x) Print an array of characters and then terminate the line.
void	println(double x) Print a double-precision floating-point number and then terminate the line.
void	println(float x) Print a floating-point number and then terminate the line.
void	println(int x) Print an integer and then terminate the line.
void	println(long x) Print a long integer and then terminate the line.
void	println(Object x) Print an Object and then terminate the line.
void	println(String x) Print a String and then terminate the line.
protected void	setError() Indicate that an error has occurred.
void	write(char[] buf) Write an array of characters.
void	write(char[] buf, int off, int len) Write a portion of an array of characters.
void	write(int c) Write a single character.
void	write(String s) Write a string.
void	write(String s, int off, int len) Write a portion of a string.


«interface»
DataOutput
+ write()
+ write()
+ write()
+ writeBoolean()
+ writeByte()
+ writeBytes()
+ writeChar()
+ writeChars()
+ writeDouble()
+ writeFloat()
+ writeInt()
+ writeLong()
+ writeShort()
+ writeUTF()
RandomAccessFile
+ RandomAccessFile(in name: String, in mode: String)
+ RandomAccessFile(in file: File, in mode: String)
+ getFD(): FileDescriptor
+ getChannel(): FileChannel
+ read(): int
+ read(in b: byte, in off: int, in len: int): int
+ read(in b: byte): int
+ readFully(in b: byte)
+ readFully(in b: byte, in off: int, in len: int)
+ skipBytes(in n: int): int
+ write(in b: int)
+ write(in b: byte)
+ write(in b: byte, in off: int, in len: int)
+ getFilePointer(): long
+ seek(in pos: long)
+ length(): long
+ setLength(in newLength: long)
+ close()
+ readBoolean(): boolean
+ readByte(): byte
+ readUnsignedByte(): int
+ readShort(): short
+ readUnsignedShort(): int
+ readChar(): char
+ readInt(): int
+ readLong(): long
+ readFloat(): float
+ readDouble(): double
+ readLine(): String
+ readUTF(): String
+ writeBoolean(in v: boolean)
+ writeByte(in v: int)
+ writeShort(in v: int)
+ writeChar(in v: int)
+ writeInt(in v: int)
+ writeLong(in v: long)
+ writeFloat(in v: float)
+ writeDouble(in v: double)
+ writeBytes(in s: String)
+ writeChars(in s: String)
+ writeUTF(in str: String)
«interface»
DataInput
+ readBoolean()
+ readByte()
+ readChar()
+ readDouble()
+ readFloat()
+ readFully()
+ readFully()
+ readInt()
+ readLine()
+ readLong()
+ readShort()
+ readUTF()
+ readUnsignedByte()
+ readUnsignedShort()
+ skipBytes()

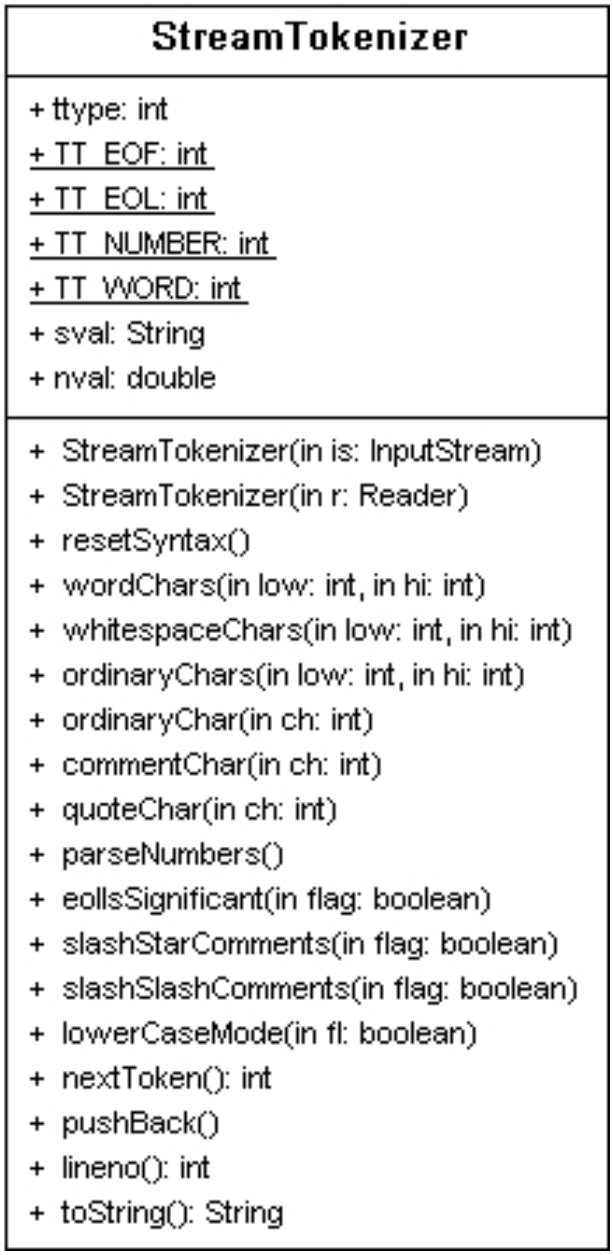
Constructor Summary

`RandomAccessFile(File file, String mode)`	Creates a random access file stream to read from, and optionally to write to, the file specified by the `File` argument.
`RandomAccessFile(String name, String mode)`	Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Method Summary

void	`close()` Closes this random access file stream and releases any system resources associated with the stream.
FileChannel	`getChannel()` Returns the unique `FileChannel` object associated with this file.
FileDescriptor	`getFD()` Returns the opaque file descriptor object associated with this stream.
long	`getFilePointer()` Returns the current offset in this file.
long	`length()` Returns the length of this file.
int	`read()` Reads a byte of data from this file.
int	`read(byte[] b)` Reads up to `b.length` bytes of data from this file into an array of bytes.
int	`read(byte[] b, int off, int len)` Reads up to `len` bytes of data from this file into an array of bytes.
boolean	`readBoolean()` Reads a `boolean` from this file.
byte	`readByte()` Reads a signed eight-bit value from this file.
char	`readChar()` Reads a Unicode character from this file.
double	`readDouble()` Reads a `double` from this file.
float	`readFloat()` Reads a `float` from this file.
void	`readFully(byte[] b, int off, int len)` Reads exactly `len` bytes from this file into the byte array, starting at the current file pointer.
int	`readInt()` Reads a signed 32-bit integer from this file.
String	`readLine()` Reads the next line of text from this file.
long	`readLong()` Reads a signed 64-bit integer from this file.
short	`readShort()` Reads a signed 16-bit number from this file.

int	readUnsignedShort() Reads an unsigned 16-bit number from this file.
String	readUTF() Reads in a string from this file.
void	seek(long pos) Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
void	setLength(long newLength) Sets the length of this file.
int	skipBytes(int n) Attempts to skip over n bytes of input discarding the skipped bytes.
void	write(byte[] b) Writes b.length bytes from the specified byte array to this file, starting at the current file pointer.
void	write(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to this file.
void	write(int b) Writes the specified byte to this file.
void	writeBoolean(boolean v) Writes a boolean to the file as a one-byte value.
void	writeByte(int v) Writes a byte to the file as a one-byte value.
void	writeBytes(String s) Writes the string to the file as a sequence of bytes.
void	writeChar(int v) Writes a char to the file as a two-byte value, high byte first.
void	writeChars(String s) Writes a string to the file 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 file as an eight-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 file as a four-byte quantity, high byte first.
void	writeInt(int v) Writes an int to the file as four bytes, high byte first.
void	writeLong(long v) Writes a long to the file as eight bytes, high byte first.
void	writeShort(int v) Writes a short to the file as two bytes, high byte first.
void	writeUTF(String str) Writes a string to the file using UTF-8 encoding in a machine-independent manner.


StreamTokenizer
+ ttype: int
+ TT_EOF: int
+ TT_EOL: int
+ TT_NUMBER: int
+ TT_WORD: int
+ sval: String
+ nval: double
+ StreamTokenizer(in is: InputStream)
+ StreamTokenizer(in r: Reader)
+ resetSyntax()
+ wordChars(in low: int, in hi: int)
+ whitespaceChars(in low: int, in hi: int)
+ ordinaryChars(in low: int, in hi: int)
+ ordinaryChar(in ch: int)
+ commentChar(in ch: int)
+ quoteChar(in ch: int)
+ parseNumbers()
+ eolIsSignificant(in flag: boolean)
+ slashStarComments(in flag: boolean)
+ slashSlashComments(in flag: boolean)
+ lowerCaseMode(in fl: boolean)
+ nextToken(): int
+ pushBack()
+ lineno(): int
+ toString(): String

Field Summary

double	**nval** If the current token is a number, this field contains the value of that number.
String	**sval** If the current token is a word token, this field contains a string giving the characters of the word token.
static int	**TT_EOF** A constant indicating that the end of the stream has been read.
static int	**TT_EOL** A constant indicating that the end of the line has been read.
static int	**TT_NUMBER** A constant indicating that a number token has been read.
static int	**TT_WORD** A constant indicating that a word token has been read.
int	**ttype** After a call to the `nextToken` method, this field contains the type of the token just read.

Constructor Summary

StreamTokenizer(InputStream is)

Deprecated. *As of JDK version 1.1, the preferred way to tokenize an input stream is to convert it into a character stream, for example:*

```
Reader r = new BufferedReader(new InputStreamReader(is));
StreamTokenizer st = new StreamTokenizer(r);
```

StreamTokenizer(Reader r)

Create a tokenizer that parses the given character stream.

Method Summary

void	`commentChar(int ch)` Specified that the character argument starts a single-line comment.
void	`eolIsSignificant(boolean flag)` Determines whether or not ends of line are treated as tokens.
int	`lineno()` Return the current line number.
void	`lowerCaseMode(boolean fl)` Determines whether or not word token are automatically lowercased.
int	`nextToken()` Parses the next token from the input stream of this tokenizer.
void	`ordinaryChar(int ch)` Specifies that the character argument is "ordinary" in this tokenizer.
void	`ordinaryChars(int low, int hi)` Specifies that all characters *c* in the range `low <= c <= high` are "ordinary" in this tokenizer.
void	`parseNumbers()` Specifies that numbers should be parsed by this tokenizer.
void	`pushBack()` Causes the next call to the `nextToken` method of this tokenizer to return the current value in the `ttype` field, and not to modify the value in the `nval` or `sval` field.
void	`quoteChar(int ch)` Specifies that matching pairs of this character delimit string constants in this tokenizer.
void	`resetSyntax()` Resets this tokenizer's syntax table so that all characters are "ordinary." See the `ordinaryChar` method for more information on a character being ordinary.
void	`slashSlashComments(boolean flag)` Determines whether or not the tokenizer recognizes C++-style comments.
void	`slashStarComments(boolean flag)` Determines whether or not the tokenizer recognizes C-style comments.
String	`toString()` Returns the string representation of the current stream token and the line number it occurs on.
void	`whitespaceChars(int low, int hi)` Specifies that all characters *c* in the range `low <= c <= high` are white space characters.
void	`wordChars(int low, int hi)` Specifies that all characters *c* in the range `low <= c <= high` are word constituents.