| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageOutputStream.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/stream/ImageOutputStream.html)    [**NO FRAMES**](http://docs.google.com/ImageOutputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **javax.imageio.stream**

Interface ImageOutputStream

**All Superinterfaces:** [DataInput](http://docs.google.com/java/io/DataInput.html), [DataOutput](http://docs.google.com/java/io/DataOutput.html), [ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) **All Known Implementing Classes:** [FileCacheImageOutputStream](http://docs.google.com/javax/imageio/stream/FileCacheImageOutputStream.html), [FileImageOutputStream](http://docs.google.com/javax/imageio/stream/FileImageOutputStream.html), [ImageOutputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html), [MemoryCacheImageOutputStream](http://docs.google.com/javax/imageio/stream/MemoryCacheImageOutputStream.html)

public interface **ImageOutputStream**extends [ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html), [DataOutput](http://docs.google.com/java/io/DataOutput.html)

A seekable output stream interface for use by ImageWriters. Various output destinations, such as OutputStreams and Files, as well as future fast I/O destinations may be "wrapped" by a suitable implementation of this interface for use by the Image I/O API.

Unlike a standard OutputStream, ImageOutputStream extends its counterpart, ImageInputStream. Thus it is possible to read from the stream as it is being written. The same seek and flush positions apply to both reading and writing, although the semantics for dealing with a non-zero bit offset before a byte-aligned write are necessarily different from the semantics for dealing with a non-zero bit offset before a byte-aligned read. When reading bytes, any bit offset is set to 0 before the read; when writing bytes, a non-zero bit offset causes the remaining bits in the byte to be written as 0s. The byte-aligned write then starts at the next byte position.

**See Also:**[ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html)

| **Method Summary** | |
| --- | --- |
| void | [**flushBefore**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#flushBefore(long))(long pos)            Flushes all data prior to the given position to the underlying destination, such as an OutputStream or File. |
| void | [**write**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#write(byte%5B%5D))(byte[] b)            Writes a sequence of bytes to the stream at the current position. |
| void | [**write**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#write(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)            Writes a sequence of bytes to the stream at the current position. |
| void | [**write**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#write(int))(int b)            Writes a single byte to the stream at the current position. |
| void | [**writeBit**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeBit(int))(int bit)            Writes a single bit, given by the least significant bit of the argument, to the stream at the current bit offset within the current byte position. |
| void | [**writeBits**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeBits(long,%20int))(long bits, int numBits)            Writes a sequence of bits, given by the numBits least significant bits of the bits argument in left-to-right order, to the stream at the current bit offset within the current byte position. |
| void | [**writeBoolean**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeBoolean(boolean))(boolean v)            Writes a boolean value to the stream. |
| void | [**writeByte**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeByte(int))(int v)            Writes the 8 low-order bits of v to the stream. |
| void | [**writeBytes**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeBytes(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Writes a string to the output stream. |
| void | [**writeChar**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeChar(int))(int v)            This method is a synonym for [writeShort](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeShort(int)). |
| void | [**writeChars**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeChars(char%5B%5D,%20int,%20int))(char[] c, int off, int len)            Writes a sequence of chars to the stream at the current position. |
| void | [**writeChars**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeChars(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Writes a string to the output stream. |
| void | [**writeDouble**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeDouble(double))(double v)            Writes a double value, which is comprised of four bytes, to the output stream. |
| void | [**writeDoubles**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeDoubles(double%5B%5D,%20int,%20int))(double[] d, int off, int len)            Writes a sequence of doubles to the stream at the current position. |
| void | [**writeFloat**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeFloat(float))(float v)            Writes a float value, which is comprised of four bytes, to the output stream. |
| void | [**writeFloats**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeFloats(float%5B%5D,%20int,%20int))(float[] f, int off, int len)            Writes a sequence of floats to the stream at the current position. |
| void | [**writeInt**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeInt(int))(int v)            Writes the 32 bits of v to the stream. |
| void | [**writeInts**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeInts(int%5B%5D,%20int,%20int))(int[] i, int off, int len)            Writes a sequence of ints to the stream at the current position. |
| void | [**writeLong**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeLong(long))(long v)            Writes the 64 bits of v to the stream. |
| void | [**writeLongs**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeLongs(long%5B%5D,%20int,%20int))(long[] l, int off, int len)            Writes a sequence of longs to the stream at the current position. |
| void | [**writeShort**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeShort(int))(int v)            Writes the 16 low-order bits of v to the stream. |
| void | [**writeShorts**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeShorts(short%5B%5D,%20int,%20int))(short[] s, int off, int len)            Writes a sequence of shorts to the stream at the current position. |
| void | [**writeUTF**](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeUTF(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Writes two bytes of length information to the output stream in network byte order, followed by the [modified UTF-8](http://docs.google.com/java/io/DataInput.html#modified-utf-8) representation of every character in the string s. |

| **Methods inherited from interface javax.imageio.stream.**[**ImageInputStream**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) |
| --- |
| [close](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#close()), [flush](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#flush()), [getBitOffset](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getBitOffset()), [getByteOrder](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder()), [getFlushedPosition](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getFlushedPosition()), [getStreamPosition](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getStreamPosition()), [isCached](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCached()), [isCachedFile](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedFile()), [isCachedMemory](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedMemory()), [length](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#length()), [mark](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#mark()), [read](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read()), [read](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read(byte%5B%5D)), [read](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read(byte%5B%5D,%20int,%20int)), [readBit](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBit()), [readBits](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBits(int)), [readBoolean](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBoolean()), [readByte](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readByte()), [readBytes](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBytes(javax.imageio.stream.IIOByteBuffer,%20int)), [readChar](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readChar()), [readDouble](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readDouble()), [readFloat](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFloat()), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(byte%5B%5D)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(byte%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(char%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(double%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(float%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(int%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(long%5B%5D,%20int,%20int)), [readFully](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(short%5B%5D,%20int,%20int)), [readInt](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readInt()), [readLine](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readLine()), [readLong](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readLong()), [readShort](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readShort()), [readUnsignedByte](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedByte()), [readUnsignedInt](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedInt()), [readUnsignedShort](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedShort()), [readUTF](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUTF()), [reset](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#reset()), [seek](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#seek(long)), [setBitOffset](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setBitOffset(int)), [setByteOrder](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setByteOrder(java.nio.ByteOrder)), [skipBytes](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#skipBytes(int)), [skipBytes](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#skipBytes(long)) |

| **Method Detail** |
| --- |

### write

void **write**(int b)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a single byte to the stream at the current position. The 24 high-order bits of b are ignored.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write. Implementers can use the [flushBits](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html#flushBits()) method of [ImageOutputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html) to guarantee this.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**b - an int whose lower 8 bits are to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### write

void **write**(byte[] b)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of bytes to the stream at the current position. If b.length is 0, nothing is written. The byte b[0] is written first, then the byte b[1], and so on.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(byte%5B%5D)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**b - an array of bytes to be written. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if b is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### write

void **write**(byte[] b,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of bytes to the stream at the current position. If len is 0, nothing is written. The byte b[off] is written first, then the byte b[off + 1], and so on.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write. Implementers can use the [flushBits](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html#flushBits()) method of [ImageOutputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html) to guarantee this.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(byte%5B%5D,%20int,%20int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**b - an array of bytes to be written.off - the start offset in the data.len - the number of bytes to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than b.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if b is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeBoolean

void **writeBoolean**(boolean v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a boolean value to the stream. If v is true, the value (byte)1 is written; if v is false, the value (byte)0 is written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeBoolean](http://docs.google.com/java/io/DataOutput.html#writeBoolean(boolean)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - the boolean to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeByte

void **writeByte**(int v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes the 8 low-order bits of v to the stream. The 24 high-order bits of v are ignored. (This means that writeByte does exactly the same thing as write for an integer argument.)

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeByte](http://docs.google.com/java/io/DataOutput.html#writeByte(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - an int containing the byte value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeShort

void **writeShort**(int v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes the 16 low-order bits of v to the stream. The 16 high-order bits of v are ignored. If the stream uses network byte order, the bytes written, in order, will be:

(byte)((v >> 8) & 0xff)  
 (byte)(v & 0xff)

Otherwise, the bytes written will be:

(byte)(v & 0xff)  
 (byte)((v >> 8) & 0xff)

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeShort](http://docs.google.com/java/io/DataOutput.html#writeShort(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - an int containing the short value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeChar

void **writeChar**(int v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

This method is a synonym for [writeShort](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeShort(int)).

**Specified by:**[writeChar](http://docs.google.com/java/io/DataOutput.html#writeChar(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - an int containing the char (unsigned short) value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[writeShort(int)](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html#writeShort(int))

### writeInt

void **writeInt**(int v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes the 32 bits of v to the stream. If the stream uses network byte order, the bytes written, in order, will be:

(byte)((v >> 24) & 0xff)  
 (byte)((v >> 16) & 0xff)  
 (byte)((v >> 8) & 0xff)  
 (byte)(v & 0xff)

Otheriwse, the bytes written will be:

(byte)(v & 0xff)  
 (byte)((v >> 8) & 0xff)  
 (byte)((v >> 16) & 0xff)  
 (byte)((v >> 24) & 0xff)

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeInt](http://docs.google.com/java/io/DataOutput.html#writeInt(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - an int containing the value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeLong

void **writeLong**(long v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes the 64 bits of v to the stream. If the stream uses network byte order, the bytes written, in order, will be:

(byte)((v >> 56) & 0xff)  
 (byte)((v >> 48) & 0xff)  
 (byte)((v >> 40) & 0xff)  
 (byte)((v >> 32) & 0xff)  
 (byte)((v >> 24) & 0xff)  
 (byte)((v >> 16) & 0xff)  
 (byte)((v >> 8) & 0xff)  
 (byte)(v & 0xff)

Otherwise, the bytes written will be:

(byte)(v & 0xff)  
 (byte)((v >> 8) & 0xff)  
 (byte)((v >> 16) & 0xff)  
 (byte)((v >> 24) & 0xff)  
 (byte)((v >> 32) & 0xff)  
 (byte)((v >> 40) & 0xff)  
 (byte)((v >> 48) & 0xff)  
 (byte)((v >> 56) & 0xff)

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeLong](http://docs.google.com/java/io/DataOutput.html#writeLong(long)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - a long containing the value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeFloat

void **writeFloat**(float v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a float value, which is comprised of four bytes, to the output stream. It does this as if it first converts this float value to an int in exactly the manner of the Float.floatToIntBits method and then writes the int value in exactly the manner of the writeInt method.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeFloat](http://docs.google.com/java/io/DataOutput.html#writeFloat(float)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - a float containing the value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeDouble

void **writeDouble**(double v)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a double value, which is comprised of four bytes, to the output stream. It does this as if it first converts this double value to an long in exactly the manner of the Double.doubleToLongBits method and then writes the long value in exactly the manner of the writeLong method.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeDouble](http://docs.google.com/java/io/DataOutput.html#writeDouble(double)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**v - a double containing the value to be written. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeBytes

void **writeBytes**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a string to the output stream. For every character in the string s, taken in order, one byte is written to the output stream. If s is null, a NullPointerException is thrown.

If s.length is zero, then no bytes are written. Otherwise, the character s[0] is written first, then s[1], and so on; the last character written is s[s.length-1]. For each character, one byte is written, the low-order byte, in exactly the manner of the writeByte method. The high-order eight bits of each character in the string are ignored.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeBytes](http://docs.google.com/java/io/DataOutput.html#writeBytes(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**s - a String containing the value to be written. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if s is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeChars

void **writeChars**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a string to the output stream. For every character in the string s, taken in order, two bytes are written to the output stream, ordered according to the current byte order setting. If network byte order is being used, the high-order byte is written first; the order is reversed otherwise. If s is null, a NullPointerException is thrown.

If s.length is zero, then no bytes are written. Otherwise, the character s[0] is written first, then s[1], and so on; the last character written is s[s.length-1].

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Specified by:**[writeChars](http://docs.google.com/java/io/DataOutput.html#writeChars(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**s - a String containing the value to be written. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if s is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeUTF

void **writeUTF**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes two bytes of length information to the output stream in network byte order, followed by the [modified UTF-8](http://docs.google.com/java/io/DataInput.html#modified-utf-8) representation of every character in the string s. If s is null, a NullPointerException is thrown. Each character in the string s is converted to a group of one, two, or three bytes, depending on the value of the character.

If a character c is in the range \u0001 through \u007f, it is represented by one byte:

(byte)c

If a character c is \u0000 or is in the range \u0080 through \u07ff, then it is represented by two bytes, to be written in the order shown:

(byte)(0xc0 | (0x1f & (c >> 6)))  
 (byte)(0x80 | (0x3f & c))

If a character c is in the range \u0800 through uffff, then it is represented by three bytes, to be written in the order shown:

(byte)(0xe0 | (0x0f & (c >> 12)))  
 (byte)(0x80 | (0x3f & (c >> 6)))  
 (byte)(0x80 | (0x3f & c))

First, the total number of bytes needed to represent all the characters of s is calculated. If this number is larger than 65535, then a UTFDataFormatException is thrown. Otherwise, this length is written to the output stream in exactly the manner of the writeShort method; after this, the one-, two-, or three-byte representation of each character in the string s is written.

The current byte order setting is ignored.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Note:** This method should not be used in the implementation of image formats that use standard UTF-8, because the modified UTF-8 used here is incompatible with standard UTF-8.

**Specified by:**[writeUTF](http://docs.google.com/java/io/DataOutput.html#writeUTF(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**s - a String containing the value to be written. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if s is null. [UTFDataFormatException](http://docs.google.com/java/io/UTFDataFormatException.html) - if the modified UTF-8 representation of s requires more than 65536 bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeShorts

void **writeShorts**(short[] s,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of shorts to the stream at the current position. If len is 0, nothing is written. The short s[off] is written first, then the short s[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**s - an array of shorts to be written.off - the start offset in the data.len - the number of shorts to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than s.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if s is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeChars

void **writeChars**(char[] c,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of chars to the stream at the current position. If len is 0, nothing is written. The char c[off] is written first, then the char c[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**c - an array of chars to be written.off - the start offset in the data.len - the number of chars to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than c.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeInts

void **writeInts**(int[] i,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of ints to the stream at the current position. If len is 0, nothing is written. The int i[off] is written first, then the int i[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**i - an array of ints to be written.off - the start offset in the data.len - the number of ints to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than i.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if i is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeLongs

void **writeLongs**(long[] l,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of longs to the stream at the current position. If len is 0, nothing is written. The long l[off] is written first, then the long l[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**l - an array of longs to be written.off - the start offset in the data.len - the number of longs to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than l.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if l is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeFloats

void **writeFloats**(float[] f,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of floats to the stream at the current position. If len is 0, nothing is written. The float f[off] is written first, then the float f[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**f - an array of floats to be written.off - the start offset in the data.len - the number of floats to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than f.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if f is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeDoubles

void **writeDoubles**(double[] d,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of doubles to the stream at the current position. If len is 0, nothing is written. The double d[off] is written first, then the double d[off + 1], and so on. The byte order of the stream is used to determine the order in which the individual bytes are written.

If the bit offset within the stream is non-zero, the remainder of the current byte is padded with 0s and written out first. The bit offset will be 0 after the write.

**Parameters:**d - an array of doubless to be written.off - the start offset in the data.len - the number of doubles to write. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than d.length. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if d is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeBit

void **writeBit**(int bit)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a single bit, given by the least significant bit of the argument, to the stream at the current bit offset within the current byte position. The upper 31 bits of the argument are ignored. The given bit replaces the previous bit at that position. The bit offset is advanced by one and reduced modulo 8.

If any bits of a particular byte have never been set at the time the byte is flushed to the destination, those bits will be set to 0 automatically.

**Parameters:**bit - an int whose least significant bit is to be written to the stream. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### writeBits

void **writeBits**(long bits,  
 int numBits)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sequence of bits, given by the numBits least significant bits of the bits argument in left-to-right order, to the stream at the current bit offset within the current byte position. The upper 64 - numBits bits of the argument are ignored. The bit offset is advanced by numBits and reduced modulo 8. Note that a bit offset of 0 always indicates the most-significant bit of the byte, and bytes of bits are written out in sequence as they are encountered. Thus bit writes are always effectively in network byte order. The actual stream byte order setting is ignored.

Bit data may be accumulated in memory indefinitely, until flushBefore is called. At that time, all bit data prior to the flushed position will be written.

If any bits of a particular byte have never been set at the time the byte is flushed to the destination, those bits will be set to 0 automatically.

**Parameters:**bits - a long containing the bits to be written, starting with the bit in position numBits - 1 down to the least significant bit.numBits - an int between 0 and 64, inclusive. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if numBits is not between 0 and 64, inclusive. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### flushBefore

void **flushBefore**(long pos)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Flushes all data prior to the given position to the underlying destination, such as an OutputStream or File. Attempting to seek to the flushed portion of the stream will result in an IndexOutOfBoundsException.

**Specified by:**[flushBefore](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#flushBefore(long)) in interface [ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) **Parameters:**pos - a long containing the length of the stream prefix that may be flushed to the destination. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if pos lies in the flushed portion of the stream or past the current stream position. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageOutputStream.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/stream/ImageOutputStream.html)    [**NO FRAMES**](http://docs.google.com/ImageOutputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).