| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageInputStream.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/IIOByteBuffer.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/stream/ImageInputStream.html)    [**NO FRAMES**](http://docs.google.com/ImageInputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.imageio.stream**

Interface ImageInputStream

**All Superinterfaces:** [DataInput](http://docs.google.com/java/io/DataInput.html) **All Known Subinterfaces:** [ImageOutputStream](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html) **All Known Implementing Classes:** [FileCacheImageInputStream](http://docs.google.com/javax/imageio/stream/FileCacheImageInputStream.html), [FileCacheImageOutputStream](http://docs.google.com/javax/imageio/stream/FileCacheImageOutputStream.html), [FileImageInputStream](http://docs.google.com/javax/imageio/stream/FileImageInputStream.html), [FileImageOutputStream](http://docs.google.com/javax/imageio/stream/FileImageOutputStream.html), [ImageInputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html), [ImageOutputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageOutputStreamImpl.html), [MemoryCacheImageInputStream](http://docs.google.com/javax/imageio/stream/MemoryCacheImageInputStream.html), [MemoryCacheImageOutputStream](http://docs.google.com/javax/imageio/stream/MemoryCacheImageOutputStream.html)

public interface **ImageInputStream**extends [DataInput](http://docs.google.com/java/io/DataInput.html)

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

**See Also:**[ImageInputStreamImpl](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html), [FileImageInputStream](http://docs.google.com/javax/imageio/stream/FileImageInputStream.html), [FileCacheImageInputStream](http://docs.google.com/javax/imageio/stream/FileCacheImageInputStream.html), [MemoryCacheImageInputStream](http://docs.google.com/javax/imageio/stream/MemoryCacheImageInputStream.html)

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#close())()            Closes the stream. |
| void | [**flush**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#flush())()            Discards the initial position of the stream prior to the current stream position. |
| void | [**flushBefore**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#flushBefore(long))(long pos)            Discards the initial portion of the stream prior to the indicated postion. |
| int | [**getBitOffset**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getBitOffset())()            Returns the current bit offset, as an integer between 0 and 7, inclusive. |
| [ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) | [**getByteOrder**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())()            Returns the byte order with which data values will be read from this stream as an instance of the java.nio.ByteOrder enumeration. |
| long | [**getFlushedPosition**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getFlushedPosition())()            Returns the earliest position in the stream to which seeking may be performed. |
| long | [**getStreamPosition**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getStreamPosition())()            Returns the current byte position of the stream. |
| boolean | [**isCached**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCached())()            Returns true if this ImageInputStream caches data itself in order to allow seeking backwards. |
| boolean | [**isCachedFile**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedFile())()            Returns true if this ImageInputStream caches data itself in order to allow seeking backwards, and the cache is kept in a temporary file. |
| boolean | [**isCachedMemory**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedMemory())()            Returns true if this ImageInputStream caches data itself in order to allow seeking backwards, and the cache is kept in main memory. |
| long | [**length**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#length())()            Returns the total length of the stream, if known. |
| void | [**mark**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#mark())()            Marks a position in the stream to be returned to by a subsequent call to reset. |
| int | [**read**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read())()            Reads a single byte from the stream and returns it as an integer between 0 and 255. |
| int | [**read**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read(byte%5B%5D))(byte[] b)            Reads up to b.length bytes from the stream, and stores them into b starting at index 0. |
| int | [**read**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#read(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)            Reads up to len bytes from the stream, and stores them into b starting at index off. |
| int | [**readBit**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBit())()            Reads a single bit from the stream and returns it as an int with the value 0 or 1. |
| long | [**readBits**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBits(int))(int numBits)            Reads a bitstring from the stream and returns it as a long, with the first bit read becoming the most significant bit of the output. |
| boolean | [**readBoolean**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBoolean())()            Reads a byte from the stream and returns a boolean value of true if it is nonzero, false if it is zero. |
| byte | [**readByte**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readByte())()            Reads a byte from the stream and returns it as a byte value. |
| void | [**readBytes**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBytes(javax.imageio.stream.IIOByteBuffer,%20int))([IIOByteBuffer](http://docs.google.com/javax/imageio/stream/IIOByteBuffer.html) buf, int len)            Reads up to len bytes from the stream, and modifies the supplied IIOByteBuffer to indicate the byte array, offset, and length where the data may be found. |
| char | [**readChar**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readChar())()            Equivalent to readUnsignedShort, except that the result is returned using the char datatype. |
| double | [**readDouble**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readDouble())()            Reads 8 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a double. |
| float | [**readFloat**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFloat())()            Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a float. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(byte%5B%5D))(byte[] b)            Reads b.length bytes from the stream, and stores them into b starting at index 0. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)            Reads len bytes from the stream, and stores them into b starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(char%5B%5D,%20int,%20int))(char[] c, int off, int len)            Reads len chars (unsigned 16-bit integers) from the stream according to the current byte order, and stores them into c starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(double%5B%5D,%20int,%20int))(double[] d, int off, int len)            Reads len doubles (64-bit IEEE double-precision floats) from the stream according to the current byte order, and stores them into d starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(float%5B%5D,%20int,%20int))(float[] f, int off, int len)            Reads len floats (32-bit IEEE single-precision floats) from the stream according to the current byte order, and stores them into f starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(int%5B%5D,%20int,%20int))(int[] i, int off, int len)            Reads len ints (signed 32-bit integers) from the stream according to the current byte order, and stores them into i starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(long%5B%5D,%20int,%20int))(long[] l, int off, int len)            Reads len longs (signed 64-bit integers) from the stream according to the current byte order, and stores them into l starting at index off. |
| void | [**readFully**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readFully(short%5B%5D,%20int,%20int))(short[] s, int off, int len)            Reads len shorts (signed 16-bit integers) from the stream according to the current byte order, and stores them into s starting at index off. |
| int | [**readInt**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readInt())()            Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as an int. |
| [String](http://docs.google.com/java/lang/String.html) | [**readLine**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readLine())()            Reads the next line of text from the input stream. |
| long | [**readLong**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readLong())()            Reads 8 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a long. |
| short | [**readShort**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readShort())()            Reads two bytes from the stream, and (conceptually) concatenates them according to the current byte order, and returns the result as a short value. |
| int | [**readUnsignedByte**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedByte())()            Reads a byte from the stream, and (conceptually) converts it to an int, masks it with 0xff in order to strip off any sign-extension bits, and returns it as a byte value. |
| long | [**readUnsignedInt**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedInt())()            Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order, converts the result to a long, masks it with 0xffffffffL in order to strip off any sign-extension bits, and returns the result as an unsigned long value. |
| int | [**readUnsignedShort**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedShort())()            Reads two bytes from the stream, and (conceptually) concatenates them according to the current byte order, converts the resulting value to an int, masks it with 0xffff in order to strip off any sign-extension buts, and returns the result as an unsigned int value. |
| [String](http://docs.google.com/java/lang/String.html) | [**readUTF**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUTF())()            Reads in a string that has been encoded using a [modified UTF-8](http://docs.google.com/java/io/DataInput.html#modified-utf-8) format. |
| void | [**reset**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#reset())()            Returns the stream pointer to its previous position, including the bit offset, at the time of the most recent unmatched call to mark. |
| void | [**seek**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#seek(long))(long pos)            Sets the current stream position to the desired location. |
| void | [**setBitOffset**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setBitOffset(int))(int bitOffset)            Sets the bit offset to an integer between 0 and 7, inclusive. |
| void | [**setByteOrder**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setByteOrder(java.nio.ByteOrder))([ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) byteOrder)            Sets the desired byte order for future reads of data values from this stream. |
| int | [**skipBytes**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#skipBytes(int))(int n)            Moves the stream position forward by a given number of bytes. |
| long | [**skipBytes**](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#skipBytes(long))(long n)            Moves the stream position forward by a given number of bytes. |

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

### setByteOrder

void **setByteOrder**([ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) byteOrder)

Sets the desired byte order for future reads of data values from this stream. For example, the sequence of bytes '0x01 0x02 0x03 0x04' if read as a 4-byte integer would have the value '0x01020304' using network byte order and the value '0x04030201' under the reverse byte order.

The enumeration class java.nio.ByteOrder is used to specify the byte order. A value of ByteOrder.BIG\_ENDIAN specifies so-called big-endian or network byte order, in which the high-order byte comes first. Motorola and Sparc processors store data in this format, while Intel processors store data in the reverse ByteOrder.LITTLE\_ENDIAN order.

The byte order has no effect on the results returned from the readBits method (or the value written by ImageOutputStream.writeBits).

**Parameters:**byteOrder - one of ByteOrder.BIG\_ENDIAN or java.nio.ByteOrder.LITTLE\_ENDIAN, indicating whether network byte order or its reverse will be used for future reads.**See Also:**[ByteOrder](http://docs.google.com/java/nio/ByteOrder.html), [getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder()), [readBits(int)](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readBits(int))

### getByteOrder

[ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) **getByteOrder**()

Returns the byte order with which data values will be read from this stream as an instance of the java.nio.ByteOrder enumeration.

**Returns:**one of ByteOrder.BIG\_ENDIAN or ByteOrder.LITTLE\_ENDIAN, indicating which byte order is being used.**See Also:**[ByteOrder](http://docs.google.com/java/nio/ByteOrder.html), [setByteOrder(java.nio.ByteOrder)](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setByteOrder(java.nio.ByteOrder))

### read

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

Reads a single byte from the stream and returns it as an integer between 0 and 255. If the end of the stream is reached, -1 is returned.

The bit offset within the stream is reset to zero before the read occurs.

**Returns:**a byte value from the stream, as an int, or -1 to indicate EOF. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### read

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

Reads up to b.length bytes from the stream, and stores them into b starting at index 0. The number of bytes read is returned. If no bytes can be read because the end of the stream has been reached, -1 is returned.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**b - an array of bytes to be written to. **Returns:**the number of bytes actually read, or -1 to indicate EOF. **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.

### read

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

Reads up to len bytes from the stream, and stores them into b starting at index off. The number of bytes read is returned. If no bytes can be read because the end of the stream has been reached, -1 is returned.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**b - an array of bytes to be written to.off - the starting position within b to write to.len - the maximum number of bytes to read. **Returns:**the number of bytes actually read, or -1 to indicate EOF. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if b is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if off is negative, len is negative, or off + len is greater than b.length. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readBytes

void **readBytes**([IIOByteBuffer](http://docs.google.com/javax/imageio/stream/IIOByteBuffer.html) buf,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads up to len bytes from the stream, and modifies the supplied IIOByteBuffer to indicate the byte array, offset, and length where the data may be found. The caller should not attempt to modify the data found in the IIOByteBuffer.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**buf - an IIOByteBuffer object to be modified.len - the maximum number of bytes to read. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if len is negative. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if buf is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readBoolean

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

Reads a byte from the stream and returns a boolean value of true if it is nonzero, false if it is zero.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readBoolean](http://docs.google.com/java/io/DataInput.html#readBoolean()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a boolean value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the end of the stream is reached. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readByte

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

Reads a byte from the stream and returns it as a byte value. Byte values between 0x00 and 0x7f represent integer values between 0 and 127. Values between 0x80 and 0xff represent negative values from -128 to /1.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readByte](http://docs.google.com/java/io/DataInput.html#readByte()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a signed byte value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the end of the stream is reached. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readUnsignedByte

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

Reads a byte from the stream, and (conceptually) converts it to an int, masks it with 0xff in order to strip off any sign-extension bits, and returns it as a byte value.

Thus, byte values between 0x00 and 0x7f are simply returned as integer values between 0 and 127. Values between 0x80 and 0xff, which normally represent negative bytevalues, will be mapped into positive integers between 128 and 255.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readUnsignedByte](http://docs.google.com/java/io/DataInput.html#readUnsignedByte()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**an unsigned byte value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the end of the stream is reached. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readShort

short **readShort**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads two bytes from the stream, and (conceptually) concatenates them according to the current byte order, and returns the result as a short value.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readShort](http://docs.google.com/java/io/DataInput.html#readShort()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a signed short value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readUnsignedShort

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

Reads two bytes from the stream, and (conceptually) concatenates them according to the current byte order, converts the resulting value to an int, masks it with 0xffff in order to strip off any sign-extension buts, and returns the result as an unsigned int value.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readUnsignedShort](http://docs.google.com/java/io/DataInput.html#readUnsignedShort()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**an unsigned short value from the stream, as an int. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readChar

char **readChar**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Equivalent to readUnsignedShort, except that the result is returned using the char datatype.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readChar](http://docs.google.com/java/io/DataInput.html#readChar()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**an unsigned char value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[readUnsignedShort()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#readUnsignedShort())

### readInt

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

Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as an int.

The bit offset within the stream is ignored and treated as though it were zero.

**Specified by:**[readInt](http://docs.google.com/java/io/DataInput.html#readInt()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a signed int value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readUnsignedInt

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

Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order, converts the result to a long, masks it with 0xffffffffL in order to strip off any sign-extension bits, and returns the result as an unsigned long value.

The bit offset within the stream is reset to zero before the read occurs.

**Returns:**an unsigned int value from the stream, as a long. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readLong

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

Reads 8 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a long.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readLong](http://docs.google.com/java/io/DataInput.html#readLong()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a signed long value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readFloat

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

Reads 4 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a float.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readFloat](http://docs.google.com/java/io/DataInput.html#readFloat()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a float value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readDouble

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

Reads 8 bytes from the stream, and (conceptually) concatenates them according to the current byte order and returns the result as a double.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readDouble](http://docs.google.com/java/io/DataInput.html#readDouble()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a double value from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getByteOrder()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getByteOrder())

### readLine

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

Reads the next line of text from the input stream. It reads successive bytes, converting each byte separately into a character, until it encounters a line terminator or end of file; the characters read are then returned as a String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then null is returned. Otherwise, each byte that is read is converted to type char by zero-extension. If the character '\n' is encountered, it is discarded and reading ceases. If the character '\r' is encountered, it is discarded and, if the following byte converts to the character '\n', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '\n' and '\r' is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order. Note that every character in this string will have a value less than \u0100, that is, (char)256.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readLine](http://docs.google.com/java/io/DataInput.html#readLine()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a String containing a line of text from the stream. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readUTF

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

Reads in a string that has been encoded using a [modified UTF-8](http://docs.google.com/java/io/DataInput.html#modified-utf-8) format. The general contract of readUTF is that it reads a representation of a Unicode character string encoded in modified UTF-8 format; this string of characters is then returned as a String.

First, two bytes are read and used to construct an unsigned 16-bit integer in the manner of the readUnsignedShort method, using network byte order (regardless of the current byte order setting). This integer value is called the *UTF length* and specifies the number of additional bytes to be read. These bytes are then converted to characters by considering them in groups. The length of each group is computed from the value of the first byte of the group. The byte following a group, if any, is the first byte of the next group.

If the first byte of a group matches the bit pattern 0xxxxxxx (where x means "may be 0 or 1"), then the group consists of just that byte. The byte is zero-extended to form a character.

If the first byte of a group matches the bit pattern 110xxxxx, then the group consists of that byte a and a second byte b. If there is no byte b (because byte a was the last of the bytes to be read), or if byte b does not match the bit pattern 10xxxxxx, then a UTFDataFormatException is thrown. Otherwise, the group is converted to the character:

(char)(((a& 0x1F) << 6) | (b & 0x3F))

If the first byte of a group matches the bit pattern 1110xxxx, then the group consists of that byte a and two more bytes b and c. If there is no byte c (because byte a was one of the last two of the bytes to be read), or either byte b or byte c does not match the bit pattern 10xxxxxx, then a UTFDataFormatException is thrown. Otherwise, the group is converted to the character:

(char)(((a & 0x0F) << 12) | ((b & 0x3F) << 6) | (c & 0x3F))

If the first byte of a group matches the pattern 1111xxxx or the pattern 10xxxxxx, then a UTFDataFormatException is thrown.

If end of file is encountered at any time during this entire process, then an EOFException is thrown.

After every group has been converted to a character by this process, the characters are gathered, in the same order in which their corresponding groups were read from the input stream, to form a String, which is returned.

The current byte order setting is ignored.

The bit offset within the stream is reset to zero before the read occurs.

**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:**[readUTF](http://docs.google.com/java/io/DataInput.html#readUTF()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a String read from the stream. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if this stream reaches the end before reading all the bytes. [UTFDataFormatException](http://docs.google.com/java/io/UTFDataFormatException.html) - if the bytes do not represent a valid modified UTF-8 encoding of a string. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len bytes from the stream, and stores them into b starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readFully](http://docs.google.com/java/io/DataInput.html#readFully(byte%5B%5D,%20int,%20int)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**b - an array of bytes to be written to.off - the starting position within b to write to.len - the maximum number of bytes to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads b.length bytes from the stream, and stores them into b starting at index 0. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Specified by:**[readFully](http://docs.google.com/java/io/DataInput.html#readFully(byte%5B%5D)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**b - an array of bytes. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if b is null. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len shorts (signed 16-bit integers) from the stream according to the current byte order, and stores them into s starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**s - an array of shorts to be written to.off - the starting position withinb to write to.len - the maximum number of shorts to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len chars (unsigned 16-bit integers) from the stream according to the current byte order, and stores them into c starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**c - an array of chars to be written to.off - the starting position withinb to write to.len - the maximum number of chars to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len ints (signed 32-bit integers) from the stream according to the current byte order, and stores them into i starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**i - an array of ints to be written to.off - the starting position withinb to write to.len - the maximum number of ints to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len longs (signed 64-bit integers) from the stream according to the current byte order, and stores them into l starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**l - an array of longs to be written to.off - the starting position withinb to write to.len - the maximum number of longs to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len floats (32-bit IEEE single-precision floats) from the stream according to the current byte order, and stores them into f starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**f - an array of floats to be written to.off - the starting position withinb to write to.len - the maximum number of floats to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readFully

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

Reads len doubles (64-bit IEEE double-precision floats) from the stream according to the current byte order, and stores them into d starting at index off. If the end of the stream is reached, an EOFException will be thrown.

The bit offset within the stream is reset to zero before the read occurs.

**Parameters:**d - an array of doubles to be written to.off - the starting position withinb to write to.len - the maximum number of doubles to read. **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. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bytes. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### getStreamPosition

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

Returns the current byte position of the stream. The next read will take place starting at this offset.

**Returns:**a long containing the position of the stream. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### getBitOffset

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

Returns the current bit offset, as an integer between 0 and 7, inclusive. The bit offset is updated implicitly by calls to the readBits method. A value of 0 indicates the most-significant bit, and a value of 7 indicates the least significant bit, of the byte being read.

The bit offset is set to 0 when a stream is first opened, and is reset to 0 by calls to seek, skipBytes, or any read or readFully method.

**Returns:**an int containing the bit offset between 0 and 7, inclusive. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[setBitOffset(int)](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#setBitOffset(int))

### setBitOffset

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

Sets the bit offset to an integer between 0 and 7, inclusive. The byte offset within the stream, as returned by getStreamPosition, is left unchanged. A value of 0 indicates the most-significant bit, and a value of 7 indicates the least significant bit, of the byte being read.

**Parameters:**bitOffset - the desired offset, as an int between 0 and 7, inclusive. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bitOffset is not between 0 and 7, inclusive. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.**See Also:**[getBitOffset()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#getBitOffset())

### readBit

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

Reads a single bit from the stream and returns it as an int with the value 0 or 1. The bit offset is advanced by one and reduced modulo 8.

**Returns:**an int containing the value 0 or 1. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bits. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### readBits

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

Reads a bitstring from the stream and returns it as a long, with the first bit read becoming the most significant bit of the output. The read starts within the byte indicated by getStreamPosition, at the bit given by getBitOffset. The bit offset is advanced by numBits and reduced modulo 8.

The byte order of the stream has no effect on this method. The return value of this method is constructed as though the bits were read one at a time, and shifted into the right side of the return value, as shown by the following pseudo-code:

long accum = 0L;  
 for (int i = 0; i < numBits; i++) {  
 accum <<= 1; // Shift left one bit to make room  
 accum |= readBit();  
 }

Note that the result of readBits(32) may thus not be equal to that of readInt() if a reverse network byte order is being used (i.e., getByteOrder() == false).

If the end of the stream is encountered before all the bits have been read, an EOFException is thrown.

**Parameters:**numBits - the number of bits to read, as an int between 0 and 64, inclusive. **Returns:**the bitstring, as a long with the last bit read stored in the least significant bit. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if numBits is not between 0 and 64, inclusive. [EOFException](http://docs.google.com/java/io/EOFException.html) - if the stream reaches the end before reading all the bits. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### length

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

Returns the total length of the stream, if known. Otherwise, -1 is returned.

**Returns:**a long containing the length of the stream, if known, or else -1. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### skipBytes

int **skipBytes**(int n)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Moves the stream position forward by a given number of bytes. It is possible that this method will only be able to skip forward by a smaller number of bytes than requested, for example if the end of the stream is reached. In all cases, the actual number of bytes skipped is returned. The bit offset is set to zero prior to advancing the position.

**Specified by:**[skipBytes](http://docs.google.com/java/io/DataInput.html#skipBytes(int)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**n - an int containing the number of bytes to be skipped. **Returns:**an int representing the number of bytes skipped. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### skipBytes

long **skipBytes**(long n)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Moves the stream position forward by a given number of bytes. This method is identical to skipBytes(int) except that it allows for a larger skip distance.

**Parameters:**n - a long containing the number of bytes to be skipped. **Returns:**a long representing the number of bytes skipped. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### seek

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

Sets the current stream position to the desired location. The next read will occur at this location. The bit offset is set to 0.

An IndexOutOfBoundsException will be thrown if pos is smaller than the flushed position (as returned by getflushedPosition).

It is legal to seek past the end of the file; an EOFException will be thrown only if a read is performed.

**Parameters:**pos - a long containing the desired file pointer position. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if pos is smaller than the flushed position. [IOException](http://docs.google.com/java/io/IOException.html) - if any other I/O error occurs.

### mark

void **mark**()

Marks a position in the stream to be returned to by a subsequent call to reset. Unlike a standard InputStream, all ImageInputStreams support marking. Additionally, calls to mark and reset may be nested arbitrarily.

Unlike the mark methods declared by the Reader and InputStream interfaces, no readLimit parameter is used. An arbitrary amount of data may be read following the call to mark.

The bit position used by the readBits method is saved and restored by each pair of calls to mark and reset.

Note that it is valid for an ImageReader to call flushBefore as part of a read operation. Therefore, if an application calls mark prior to passing that stream to an ImageReader, the application should not assume that the marked position will remain valid after the read operation has completed.

### reset

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

Returns the stream pointer to its previous position, including the bit offset, at the time of the most recent unmatched call to mark.

Calls to reset without a corresponding call to mark have no effect.

An IOException will be thrown if the previous marked position lies in the discarded portion of the stream.

**Throws:** [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)

Discards the initial portion of the stream prior to the indicated postion. Attempting to seek to an offset within the flushed portion of the stream will result in an IndexOutOfBoundsException.

Calling flushBefore may allow classes implementing this interface to free up resources such as memory or disk space that are being used to store data from the stream.

**Parameters:**pos - a long containing the length of the stream prefix that may be flushed. **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.

### flush

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

Discards the initial position of the stream prior to the current stream position. Equivalent to flushBefore(getStreamPosition()).

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs.

### getFlushedPosition

long **getFlushedPosition**()

Returns the earliest position in the stream to which seeking may be performed. The returned value will be the maximum of all values passed into previous calls to flushBefore.

**Returns:**the earliest legal position for seeking, as a long.

### isCached

boolean **isCached**()

Returns true if this ImageInputStream caches data itself in order to allow seeking backwards. Applications may consult this in order to decide how frequently, or whether, to flush in order to conserve cache resources.

**Returns:**true if this ImageInputStream caches data.**See Also:**[isCachedMemory()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedMemory()), [isCachedFile()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedFile())

### isCachedMemory

boolean **isCachedMemory**()

Returns true if this ImageInputStream caches data itself in order to allow seeking backwards, and the cache is kept in main memory. Applications may consult this in order to decide how frequently, or whether, to flush in order to conserve cache resources.

**Returns:**true if this ImageInputStream caches data in main memory.**See Also:**[isCached()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCached()), [isCachedFile()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedFile())

### isCachedFile

boolean **isCachedFile**()

Returns true if this ImageInputStream caches data itself in order to allow seeking backwards, and the cache is kept in a temporary file. Applications may consult this in order to decide how frequently, or whether, to flush in order to conserve cache resources.

**Returns:**true if this ImageInputStream caches data in a temporary file.**See Also:**[isCached()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCached()), [isCachedMemory()](http://docs.google.com/javax/imageio/stream/ImageInputStream.html#isCachedMemory())

### close

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

Closes the stream. Attempts to access a stream that has been closed may result in IOExceptions or incorrect behavior. Calling this method may allow classes implementing this interface to release resources associated with the stream such as memory, disk space, or file descriptors.

**Throws:** [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/ImageInputStream.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/IIOByteBuffer.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/stream/ImageInputStreamImpl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/stream/ImageInputStream.html)    [**NO FRAMES**](http://docs.google.com/ImageInputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[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).