| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Buffer.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   [**NEXT CLASS**](http://docs.google.com/java/nio/BufferOverflowException.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/Buffer.html)    [**NO FRAMES**](http://docs.google.com/Buffer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.nio**

Class Buffer

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.nio.Buffer**

**Direct Known Subclasses:** [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html), [CharBuffer](http://docs.google.com/java/nio/CharBuffer.html), [DoubleBuffer](http://docs.google.com/java/nio/DoubleBuffer.html), [FloatBuffer](http://docs.google.com/java/nio/FloatBuffer.html), [IntBuffer](http://docs.google.com/java/nio/IntBuffer.html), [LongBuffer](http://docs.google.com/java/nio/LongBuffer.html), [ShortBuffer](http://docs.google.com/java/nio/ShortBuffer.html)

public abstract class **Buffer**extends [Object](http://docs.google.com/java/lang/Object.html)

A container for data of a specific primitive type.

A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's *capacity* is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's *limit* is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's *position* is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

#### Transferring data

Each subclass of this class defines two categories of *get* and *put* operations:

*Relative* operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative *get* operation throws a [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) and a relative *put* operation throws a [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html); in either case, no data is transferred.

*Absolute* operations take an explicit element index and do not affect the position. Absolute *get* and *put* operations throw an [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

#### Marking and resetting

A buffer's *mark* is the index to which its position will be reset when the [reset](http://docs.google.com/java/nio/Buffer.html#reset()) method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the [reset](http://docs.google.com/java/nio/Buffer.html#reset()) method causes an [InvalidMarkException](http://docs.google.com/java/nio/InvalidMarkException.html) to be thrown.

#### Invariants

The following invariant holds for the mark, position, limit, and capacity values:

0 <= *mark* <= *position* <= *limit* <= *capacity*

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

#### Clearing, flipping, and rewinding

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

* [clear()](http://docs.google.com/java/nio/Buffer.html#clear()) makes a buffer ready for a new sequence of channel-read or relative *put* operations: It sets the limit to the capacity and the position to zero.
* [flip()](http://docs.google.com/java/nio/Buffer.html#flip()) makes a buffer ready for a new sequence of channel-write or relative *get* operations: It sets the limit to the current position and then sets the position to zero.
* [rewind()](http://docs.google.com/java/nio/Buffer.html#rewind()) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

#### Read-only buffers

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as *optional operations* that will throw a [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its [isReadOnly](http://docs.google.com/java/nio/Buffer.html#isReadOnly()) method.

#### Thread safety

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

#### Invocation chaining

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

b.flip();  
 b.position(23);  
 b.limit(42);

can be replaced by the single, more compact statement

b.flip().position(23).limit(42);

**Since:** 1.4

| **Method Summary** | |
| --- | --- |
| abstract  [Object](http://docs.google.com/java/lang/Object.html) | [**array**](http://docs.google.com/java/nio/Buffer.html#array())()            Returns the array that backs this buffer  *(optional operation)*. |
| abstract  int | [**arrayOffset**](http://docs.google.com/java/nio/Buffer.html#arrayOffset())()            Returns the offset within this buffer's backing array of the first element of the buffer  *(optional operation)*. |
| int | [**capacity**](http://docs.google.com/java/nio/Buffer.html#capacity())()            Returns this buffer's capacity. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**clear**](http://docs.google.com/java/nio/Buffer.html#clear())()            Clears this buffer. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**flip**](http://docs.google.com/java/nio/Buffer.html#flip())()            Flips this buffer. |
| abstract  boolean | [**hasArray**](http://docs.google.com/java/nio/Buffer.html#hasArray())()            Tells whether or not this buffer is backed by an accessible array. |
| boolean | [**hasRemaining**](http://docs.google.com/java/nio/Buffer.html#hasRemaining())()            Tells whether there are any elements between the current position and the limit. |
| abstract  boolean | [**isDirect**](http://docs.google.com/java/nio/Buffer.html#isDirect())()            Tells whether or not this buffer is [*direct*](http://docs.google.com/ByteBuffer.html#direct). |
| abstract  boolean | [**isReadOnly**](http://docs.google.com/java/nio/Buffer.html#isReadOnly())()            Tells whether or not this buffer is read-only. |
| int | [**limit**](http://docs.google.com/java/nio/Buffer.html#limit())()            Returns this buffer's limit. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**limit**](http://docs.google.com/java/nio/Buffer.html#limit(int))(int newLimit)            Sets this buffer's limit. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**mark**](http://docs.google.com/java/nio/Buffer.html#mark())()            Sets this buffer's mark at its position. |
| int | [**position**](http://docs.google.com/java/nio/Buffer.html#position())()            Returns this buffer's position. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**position**](http://docs.google.com/java/nio/Buffer.html#position(int))(int newPosition)            Sets this buffer's position. |
| int | [**remaining**](http://docs.google.com/java/nio/Buffer.html#remaining())()            Returns the number of elements between the current position and the limit. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**reset**](http://docs.google.com/java/nio/Buffer.html#reset())()            Resets this buffer's position to the previously-marked position. |
| [Buffer](http://docs.google.com/java/nio/Buffer.html) | [**rewind**](http://docs.google.com/java/nio/Buffer.html#rewind())()            Rewinds this buffer. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### capacity

public final int **capacity**()

Returns this buffer's capacity.

**Returns:**The capacity of this buffer

### position

public final int **position**()

Returns this buffer's position.

**Returns:**The position of this buffer

### position

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **position**(int newPosition)

Sets this buffer's position. If the mark is defined and larger than the new position then it is discarded.

**Parameters:**newPosition - The new position value; must be non-negative and no larger than the current limit **Returns:**This buffer **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the preconditions on newPosition do not hold

### limit

public final int **limit**()

Returns this buffer's limit.

**Returns:**The limit of this buffer

### limit

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **limit**(int newLimit)

Sets this buffer's limit. If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

**Parameters:**newLimit - The new limit value; must be non-negative and no larger than this buffer's capacity **Returns:**This buffer **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the preconditions on newLimit do not hold

### mark

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **mark**()

Sets this buffer's mark at its position.

**Returns:**This buffer

### reset

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **reset**()

Resets this buffer's position to the previously-marked position.

Invoking this method neither changes nor discards the mark's value.

**Returns:**This buffer **Throws:** [InvalidMarkException](http://docs.google.com/java/nio/InvalidMarkException.html) - If the mark has not been set

### clear

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **clear**()

Clears this buffer. The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or *put* operations to fill this buffer. For example:

buf.clear(); // Prepare buffer for reading  
 in.read(buf); // Read data

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

**Returns:**This buffer

### flip

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **flip**()

Flips this buffer. The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or *put* operations, invoke this method to prepare for a sequence of channel-write or relative *get* operations. For example:

buf.put(magic); // Prepend header  
 in.read(buf); // Read data into rest of buffer  
 buf.flip(); // Flip buffer  
 out.write(buf); // Write header + data to channel

This method is often used in conjunction with the [compact](http://docs.google.com/java/nio/ByteBuffer.html#compact()) method when transferring data from one place to another.

**Returns:**This buffer

### rewind

public final [Buffer](http://docs.google.com/java/nio/Buffer.html) **rewind**()

Rewinds this buffer. The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or *get* operations, assuming that the limit has already been set appropriately. For example:

out.write(buf); // Write remaining data  
 buf.rewind(); // Rewind buffer  
 buf.get(array); // Copy data into array

**Returns:**This buffer

### remaining

public final int **remaining**()

Returns the number of elements between the current position and the limit.

**Returns:**The number of elements remaining in this buffer

### hasRemaining

public final boolean **hasRemaining**()

Tells whether there are any elements between the current position and the limit.

**Returns:**true if, and only if, there is at least one element remaining in this buffer

### isReadOnly

public abstract boolean **isReadOnly**()

Tells whether or not this buffer is read-only.

**Returns:**true if, and only if, this buffer is read-only

### hasArray

public abstract boolean **hasArray**()

Tells whether or not this buffer is backed by an accessible array.

If this method returns true then the [array](http://docs.google.com/java/nio/Buffer.html#array()) and [arrayOffset](http://docs.google.com/java/nio/Buffer.html#arrayOffset()) methods may safely be invoked.

**Returns:**true if, and only if, this buffer is backed by an array and is not read-only**Since:** 1.6

### array

public abstract [Object](http://docs.google.com/java/lang/Object.html) **array**()

Returns the array that backs this buffer  *(optional operation)*.

This method is intended to allow array-backed buffers to be passed to native code more efficiently. Concrete subclasses provide more strongly-typed return values for this method.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the [hasArray](http://docs.google.com/java/nio/Buffer.html#hasArray()) method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**The array that backs this buffer **Throws:** [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is backed by an array but is read-only [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - If this buffer is not backed by an accessible array**Since:** 1.6

### arrayOffset

public abstract int **arrayOffset**()

Returns the offset within this buffer's backing array of the first element of the buffer  *(optional operation)*.

If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + arrayOffset().

Invoke the [hasArray](http://docs.google.com/java/nio/Buffer.html#hasArray()) method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns:**The offset within this buffer's array of the first element of the buffer **Throws:** [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is backed by an array but is read-only [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - If this buffer is not backed by an accessible array**Since:** 1.6

### isDirect

public abstract boolean **isDirect**()

Tells whether or not this buffer is [*direct*](http://docs.google.com/ByteBuffer.html#direct).

**Returns:**true if, and only if, this buffer is direct**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Buffer.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   [**NEXT CLASS**](http://docs.google.com/java/nio/BufferOverflowException.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/Buffer.html)    [**NO FRAMES**](http://docs.google.com/Buffer.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).