| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ByteBuffer.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/java/nio/BufferUnderflowException.html)   [**NEXT CLASS**](http://docs.google.com/java/nio/ByteOrder.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/ByteBuffer.html)    [**NO FRAMES**](http://docs.google.com/ByteBuffer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3dy6vkm) | DETAIL: FIELD | CONSTR | [METHOD](#2s8eyo1) |

## **java.nio**

Class ByteBuffer

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

**All Implemented Interfaces:** [Comparable](http://docs.google.com/java/lang/Comparable.html)<[ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)> **Direct Known Subclasses:** [MappedByteBuffer](http://docs.google.com/java/nio/MappedByteBuffer.html)

public abstract class **ByteBuffer**extends [Buffer](http://docs.google.com/java/nio/Buffer.html)implements [Comparable](http://docs.google.com/java/lang/Comparable.html)<[ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)>

A byte buffer.

This class defines six categories of operations upon byte buffers:

* Absolute and relative [*get*](http://docs.google.com/java/nio/ByteBuffer.html#get()) and [*put*](http://docs.google.com/java/nio/ByteBuffer.html#put(byte)) methods that read and write single bytes;
* Relative [*bulk get*](http://docs.google.com/java/nio/ByteBuffer.html#get(byte%5B%5D)) methods that transfer contiguous sequences of bytes from this buffer into an array;
* Relative [*bulk put*](http://docs.google.com/java/nio/ByteBuffer.html#put(byte%5B%5D)) methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer;
* Absolute and relative [*get*](http://docs.google.com/java/nio/ByteBuffer.html#getChar()) and [*put*](http://docs.google.com/java/nio/ByteBuffer.html#putChar(char)) methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order;
* Methods for creating [*view buffers*](#tyjcwt), which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and
* Methods for [compacting](http://docs.google.com/java/nio/ByteBuffer.html#compact()), [duplicating](http://docs.google.com/java/nio/ByteBuffer.html#duplicate()), and [slicing](http://docs.google.com/java/nio/ByteBuffer.html#slice()) a byte buffer.

Byte buffers can be created either by [*allocation*](http://docs.google.com/java/nio/ByteBuffer.html#allocate(int)), which allocates space for the buffer's content, or by [*wrapping*](http://docs.google.com/java/nio/ByteBuffer.html#wrap(byte%5B%5D)) an existing byte array into a buffer.

#### Direct ***vs.*** non-direct buffers

A byte buffer is either *direct* or *non-direct*. Given a direct byte buffer, the Java virtual machine will make a best effort to perform native I/O operations directly upon it. That is, it will attempt to avoid copying the buffer's content to (or from) an intermediate buffer before (or after) each invocation of one of the underlying operating system's native I/O operations.

A direct byte buffer may be created by invoking the [allocateDirect](http://docs.google.com/java/nio/ByteBuffer.html#allocateDirect(int)) factory method of this class. The buffers returned by this method typically have somewhat higher allocation and deallocation costs than non-direct buffers. The contents of direct buffers may reside outside of the normal garbage-collected heap, and so their impact upon the memory footprint of an application might not be obvious. It is therefore recommended that direct buffers be allocated primarily for large, long-lived buffers that are subject to the underlying system's native I/O operations. In general it is best to allocate direct buffers only when they yield a measureable gain in program performance.

A direct byte buffer may also be created by [mapping](http://docs.google.com/java/nio/channels/FileChannel.html#map(java.nio.channels.FileChannel.MapMode,%20long,%20long)) a region of a file directly into memory. An implementation of the Java platform may optionally support the creation of direct byte buffers from native code via JNI. If an instance of one of these kinds of buffers refers to an inaccessible region of memory then an attempt to access that region will not change the buffer's content and will cause an unspecified exception to be thrown either at the time of the access or at some later time.

Whether a byte buffer is direct or non-direct may be determined by invoking its [isDirect](http://docs.google.com/java/nio/ByteBuffer.html#isDirect()) method. This method is provided so that explicit buffer management can be done in performance-critical code.

#### Access to binary data

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order, which may be retrieved and modified via the [order](http://docs.google.com/java/nio/ByteBuffer.html#order()) methods. Specific byte orders are represented by instances of the [ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) class. The initial order of a byte buffer is always [BIG\_ENDIAN](http://docs.google.com/java/nio/ByteOrder.html#BIG_ENDIAN).

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative *get* and *put* methods for each type. For 32-bit floating-point values, for example, this class defines:

float [getFloat()](http://docs.google.com/java/nio/ByteBuffer.html#getFloat())  
 float [getFloat(int index)](http://docs.google.com/java/nio/ByteBuffer.html#getFloat(int))  
 void [putFloat(float f)](http://docs.google.com/java/nio/ByteBuffer.html#putFloat(float))  
 void [putFloat(int index, float f)](http://docs.google.com/java/nio/ByteBuffer.html#putFloat(int,%20float))

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute *get* and *put* methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create *views* of a given byte buffer. A *view buffer* is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The [asFloatBuffer](http://docs.google.com/java/nio/ByteBuffer.html#asFloatBuffer()) method, for example, creates an instance of the [FloatBuffer](http://docs.google.com/java/nio/FloatBuffer.html) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have three important advantages over the families of type-specific *get* and *put* methods described above:

* A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;
* A view buffer provides relative bulk *get* and *put* methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and
* A view buffer is potentially much more efficient because it will be direct if, and only if, its backing byte buffer is direct.

The byte order of a view buffer is fixed to be that of its byte buffer at the time that the view is created.

#### 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. The sequence of statements

bb.putInt(0xCAFEBABE);  
 bb.putShort(3);  
 bb.putShort(45);

can, for example, be replaced by the single statement

bb.putInt(0xCAFEBABE).putShort(3).putShort(45);

**Since:** 1.4

| **Method Summary** | |
| --- | --- |
| static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**allocate**](http://docs.google.com/java/nio/ByteBuffer.html#allocate(int))(int capacity)            Allocates a new byte buffer. |
| static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**allocateDirect**](http://docs.google.com/java/nio/ByteBuffer.html#allocateDirect(int))(int capacity)            Allocates a new direct byte buffer. |
| byte[] | [**array**](http://docs.google.com/java/nio/ByteBuffer.html#array())()            Returns the byte array that backs this buffer  *(optional operation)*. |
| int | [**arrayOffset**](http://docs.google.com/java/nio/ByteBuffer.html#arrayOffset())()            Returns the offset within this buffer's backing array of the first element of the buffer  *(optional operation)*. |
| abstract  [CharBuffer](http://docs.google.com/java/nio/CharBuffer.html) | [**asCharBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asCharBuffer())()            Creates a view of this byte buffer as a char buffer. |
| abstract  [DoubleBuffer](http://docs.google.com/java/nio/DoubleBuffer.html) | [**asDoubleBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asDoubleBuffer())()            Creates a view of this byte buffer as a double buffer. |
| abstract  [FloatBuffer](http://docs.google.com/java/nio/FloatBuffer.html) | [**asFloatBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asFloatBuffer())()            Creates a view of this byte buffer as a float buffer. |
| abstract  [IntBuffer](http://docs.google.com/java/nio/IntBuffer.html) | [**asIntBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asIntBuffer())()            Creates a view of this byte buffer as an int buffer. |
| abstract  [LongBuffer](http://docs.google.com/java/nio/LongBuffer.html) | [**asLongBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asLongBuffer())()            Creates a view of this byte buffer as a long buffer. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**asReadOnlyBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asReadOnlyBuffer())()            Creates a new, read-only byte buffer that shares this buffer's content. |
| abstract  [ShortBuffer](http://docs.google.com/java/nio/ShortBuffer.html) | [**asShortBuffer**](http://docs.google.com/java/nio/ByteBuffer.html#asShortBuffer())()            Creates a view of this byte buffer as a short buffer. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**compact**](http://docs.google.com/java/nio/ByteBuffer.html#compact())()            Compacts this buffer  *(optional operation)*. |
| int | [**compareTo**](http://docs.google.com/java/nio/ByteBuffer.html#compareTo(java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) that)            Compares this buffer to another. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**duplicate**](http://docs.google.com/java/nio/ByteBuffer.html#duplicate())()            Creates a new byte buffer that shares this buffer's content. |
| boolean | [**equals**](http://docs.google.com/java/nio/ByteBuffer.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) ob)            Tells whether or not this buffer is equal to another object. |
| abstract  byte | [**get**](http://docs.google.com/java/nio/ByteBuffer.html#get())()            Relative *get* method. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**get**](http://docs.google.com/java/nio/ByteBuffer.html#get(byte%5B%5D))(byte[] dst)            Relative bulk *get* method. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**get**](http://docs.google.com/java/nio/ByteBuffer.html#get(byte%5B%5D,%20int,%20int))(byte[] dst, int offset, int length)            Relative bulk *get* method. |
| abstract  byte | [**get**](http://docs.google.com/java/nio/ByteBuffer.html#get(int))(int index)            Absolute *get* method. |
| abstract  char | [**getChar**](http://docs.google.com/java/nio/ByteBuffer.html#getChar())()            Relative *get* method for reading a char value. |
| abstract  char | [**getChar**](http://docs.google.com/java/nio/ByteBuffer.html#getChar(int))(int index)            Absolute *get* method for reading a char value. |
| abstract  double | [**getDouble**](http://docs.google.com/java/nio/ByteBuffer.html#getDouble())()            Relative *get* method for reading a double value. |
| abstract  double | [**getDouble**](http://docs.google.com/java/nio/ByteBuffer.html#getDouble(int))(int index)            Absolute *get* method for reading a double value. |
| abstract  float | [**getFloat**](http://docs.google.com/java/nio/ByteBuffer.html#getFloat())()            Relative *get* method for reading a float value. |
| abstract  float | [**getFloat**](http://docs.google.com/java/nio/ByteBuffer.html#getFloat(int))(int index)            Absolute *get* method for reading a float value. |
| abstract  int | [**getInt**](http://docs.google.com/java/nio/ByteBuffer.html#getInt())()            Relative *get* method for reading an int value. |
| abstract  int | [**getInt**](http://docs.google.com/java/nio/ByteBuffer.html#getInt(int))(int index)            Absolute *get* method for reading an int value. |
| abstract  long | [**getLong**](http://docs.google.com/java/nio/ByteBuffer.html#getLong())()            Relative *get* method for reading a long value. |
| abstract  long | [**getLong**](http://docs.google.com/java/nio/ByteBuffer.html#getLong(int))(int index)            Absolute *get* method for reading a long value. |
| abstract  short | [**getShort**](http://docs.google.com/java/nio/ByteBuffer.html#getShort())()            Relative *get* method for reading a short value. |
| abstract  short | [**getShort**](http://docs.google.com/java/nio/ByteBuffer.html#getShort(int))(int index)            Absolute *get* method for reading a short value. |
| boolean | [**hasArray**](http://docs.google.com/java/nio/ByteBuffer.html#hasArray())()            Tells whether or not this buffer is backed by an accessible byte array. |
| int | [**hashCode**](http://docs.google.com/java/nio/ByteBuffer.html#hashCode())()            Returns the current hash code of this buffer. |
| abstract  boolean | [**isDirect**](http://docs.google.com/java/nio/ByteBuffer.html#isDirect())()            Tells whether or not this byte buffer is direct. |
| [ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) | [**order**](http://docs.google.com/java/nio/ByteBuffer.html#order())()            Retrieves this buffer's byte order. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**order**](http://docs.google.com/java/nio/ByteBuffer.html#order(java.nio.ByteOrder))([ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) bo)            Modifies this buffer's byte order. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**put**](http://docs.google.com/java/nio/ByteBuffer.html#put(byte))(byte b)            Relative *put* method  *(optional operation)*. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**put**](http://docs.google.com/java/nio/ByteBuffer.html#put(byte%5B%5D))(byte[] src)            Relative bulk *put* method  *(optional operation)*. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**put**](http://docs.google.com/java/nio/ByteBuffer.html#put(byte%5B%5D,%20int,%20int))(byte[] src, int offset, int length)            Relative bulk *put* method  *(optional operation)*. |
| [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**put**](http://docs.google.com/java/nio/ByteBuffer.html#put(java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src)            Relative bulk *put* method  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**put**](http://docs.google.com/java/nio/ByteBuffer.html#put(int,%20byte))(int index, byte b)            Absolute *put* method  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putChar**](http://docs.google.com/java/nio/ByteBuffer.html#putChar(char))(char value)            Relative *put* method for writing a char value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putChar**](http://docs.google.com/java/nio/ByteBuffer.html#putChar(int,%20char))(int index, char value)            Absolute *put* method for writing a char value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putDouble**](http://docs.google.com/java/nio/ByteBuffer.html#putDouble(double))(double value)            Relative *put* method for writing a double value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putDouble**](http://docs.google.com/java/nio/ByteBuffer.html#putDouble(int,%20double))(int index, double value)            Absolute *put* method for writing a double value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putFloat**](http://docs.google.com/java/nio/ByteBuffer.html#putFloat(float))(float value)            Relative *put* method for writing a float value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putFloat**](http://docs.google.com/java/nio/ByteBuffer.html#putFloat(int,%20float))(int index, float value)            Absolute *put* method for writing a float value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putInt**](http://docs.google.com/java/nio/ByteBuffer.html#putInt(int))(int value)            Relative *put* method for writing an int value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putInt**](http://docs.google.com/java/nio/ByteBuffer.html#putInt(int,%20int))(int index, int value)            Absolute *put* method for writing an int value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putLong**](http://docs.google.com/java/nio/ByteBuffer.html#putLong(int,%20long))(int index, long value)            Absolute *put* method for writing a long value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putLong**](http://docs.google.com/java/nio/ByteBuffer.html#putLong(long))(long value)            Relative *put* method for writing a long value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putShort**](http://docs.google.com/java/nio/ByteBuffer.html#putShort(int,%20short))(int index, short value)            Absolute *put* method for writing a short value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**putShort**](http://docs.google.com/java/nio/ByteBuffer.html#putShort(short))(short value)            Relative *put* method for writing a short value  *(optional operation)*. |
| abstract  [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**slice**](http://docs.google.com/java/nio/ByteBuffer.html#slice())()            Creates a new byte buffer whose content is a shared subsequence of this buffer's content. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/nio/ByteBuffer.html#toString())()            Returns a string summarizing the state of this buffer. |
| static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**wrap**](http://docs.google.com/java/nio/ByteBuffer.html#wrap(byte%5B%5D))(byte[] array)            Wraps a byte array into a buffer. |
| static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) | [**wrap**](http://docs.google.com/java/nio/ByteBuffer.html#wrap(byte%5B%5D,%20int,%20int))(byte[] array, int offset, int length)            Wraps a byte array into a buffer. |

| **Methods inherited from class java.nio.**[**Buffer**](http://docs.google.com/java/nio/Buffer.html) |
| --- |
| [capacity](http://docs.google.com/java/nio/Buffer.html#capacity()), [clear](http://docs.google.com/java/nio/Buffer.html#clear()), [flip](http://docs.google.com/java/nio/Buffer.html#flip()), [hasRemaining](http://docs.google.com/java/nio/Buffer.html#hasRemaining()), [isReadOnly](http://docs.google.com/java/nio/Buffer.html#isReadOnly()), [limit](http://docs.google.com/java/nio/Buffer.html#limit()), [limit](http://docs.google.com/java/nio/Buffer.html#limit(int)), [mark](http://docs.google.com/java/nio/Buffer.html#mark()), [position](http://docs.google.com/java/nio/Buffer.html#position()), [position](http://docs.google.com/java/nio/Buffer.html#position(int)), [remaining](http://docs.google.com/java/nio/Buffer.html#remaining()), [reset](http://docs.google.com/java/nio/Buffer.html#reset()), [rewind](http://docs.google.com/java/nio/Buffer.html#rewind()) |

| **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()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [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** |
| --- |

### allocateDirect

public static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **allocateDirect**(int capacity)

Allocates a new direct byte buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. Whether or not it has a [backing array](http://docs.google.com/java/nio/ByteBuffer.html#hasArray()) is unspecified.

**Parameters:**capacity - The new buffer's capacity, in bytes **Returns:**The new byte buffer **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the capacity is a negative integer

### allocate

public static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **allocate**(int capacity)

Allocates a new byte buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a [backing array](http://docs.google.com/java/nio/ByteBuffer.html#array()), and its [array offset](http://docs.google.com/java/nio/ByteBuffer.html#arrayOffset()) will be zero.

**Parameters:**capacity - The new buffer's capacity, in bytes **Returns:**The new byte buffer **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the capacity is a negative integer

### wrap

public static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **wrap**(byte[] array,  
 int offset,  
 int length)

Wraps a byte array into a buffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its [backing array](http://docs.google.com/java/nio/ByteBuffer.html#array()) will be the given array, and its [array offset](http://docs.google.com/java/nio/ByteBuffer.html#arrayOffset()) will be zero.

**Parameters:**array - The array that will back the new bufferoffset - The offset of the subarray to be used; must be non-negative and no larger than array.length. The new buffer's position will be set to this value.length - The length of the subarray to be used; must be non-negative and no larger than array.length - offset. The new buffer's limit will be set to offset + length. **Returns:**The new byte buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If the preconditions on the offset and length parameters do not hold

### wrap

public static [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **wrap**(byte[] array)

Wraps a byte array into a buffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be array.length, its position will be zero, and its mark will be undefined. Its [backing array](http://docs.google.com/java/nio/ByteBuffer.html#array()) will be the given array, and its [array offset](http://docs.google.com/java/nio/ByteBuffer.html#arrayOffset()) will be zero.

**Parameters:**array - The array that will back this buffer **Returns:**The new byte buffer

### slice

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **slice**()

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**The new byte buffer

### duplicate

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **duplicate**()

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**The new byte buffer

### asReadOnlyBuffer

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **asReadOnlyBuffer**()

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

If this buffer is itself read-only then this method behaves in exactly the same way as the [duplicate](http://docs.google.com/java/nio/ByteBuffer.html#duplicate()) method.

**Returns:**The new, read-only byte buffer

### get

public abstract byte **get**()

Relative *get* method. Reads the byte at this buffer's current position, and then increments the position.

**Returns:**The byte at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If the buffer's current position is not smaller than its limit

### put

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **put**(byte b)

Relative *put* method  *(optional operation)*.

Writes the given byte into this buffer at the current position, and then increments the position.

**Parameters:**b - The byte to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If this buffer's current position is not smaller than its limit [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### get

public abstract byte **get**(int index)

Absolute *get* method. Reads the byte at the given index.

**Parameters:**index - The index from which the byte will be read **Returns:**The byte at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit

### put

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **put**(int index,  
 byte b)

Absolute *put* method  *(optional operation)*.

Writes the given byte into this buffer at the given index.

**Parameters:**index - The index at which the byte will be writtenb - The byte value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### get

public [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **get**(byte[] dst,  
 int offset,  
 int length)

Relative bulk *get* method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if length > remaining(), then no bytes are transferred and a [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) is thrown.

Otherwise, this method copies length bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

In other words, an invocation of this method of the form src.get(dst, off, len) has exactly the same effect as the loop

for (int i = off; i < off + len; i++)  
 dst[i] = src.get();

except that it first checks that there are sufficient bytes in this buffer and it is potentially much more efficient.

**Parameters:**dst - The array into which bytes are to be writtenoffset - The offset within the array of the first byte to be written; must be non-negative and no larger than dst.lengthlength - The maximum number of bytes to be written to the given array; must be non-negative and no larger than dst.length - offset **Returns:**This buffer **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than length bytes remaining in this buffer [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If the preconditions on the offset and length parameters do not hold

### get

public [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **get**(byte[] dst)

Relative bulk *get* method.

This method transfers bytes from this buffer into the given destination array. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation

src.get(a, 0, a.length)

**Returns:**This buffer **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than length bytes remaining in this buffer

### put

public [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **put**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src)

Relative bulk *put* method  *(optional operation)*.

This method transfers the bytes remaining in the given source buffer into this buffer. If there are more bytes remaining in the source buffer than in this buffer, that is, if src.remaining() > remaining(), then no bytes are transferred and a [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) is thrown.

Otherwise, this method copies *n* = src.remaining() bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by *n*.

In other words, an invocation of this method of the form dst.put(src) has exactly the same effect as the loop

while (src.hasRemaining())  
 dst.put(src.get());

except that it first checks that there is sufficient space in this buffer and it is potentially much more efficient.

**Parameters:**src - The source buffer from which bytes are to be read; must not be this buffer **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there is insufficient space in this buffer for the remaining bytes in the source buffer [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the source buffer is this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### put

public [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **put**(byte[] src,  
 int offset,  
 int length)

Relative bulk *put* method  *(optional operation)*.

This method transfers bytes into this buffer from the given source array. If there are more bytes to be copied from the array than remain in this buffer, that is, if length > remaining(), then no bytes are transferred and a [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

In other words, an invocation of this method of the form dst.put(src, off, len) has exactly the same effect as the loop

for (int i = off; i < off + len; i++)  
 dst.put(a[i]);

except that it first checks that there is sufficient space in this buffer and it is potentially much more efficient.

**Parameters:**src - The array from which bytes are to be readoffset - The offset within the array of the first byte to be read; must be non-negative and no larger than array.lengthlength - The number of bytes to be read from the given array; must be non-negative and no larger than array.length - offset **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there is insufficient space in this buffer [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If the preconditions on the offset and length parameters do not hold [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### put

public final [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **put**(byte[] src)

Relative bulk *put* method  *(optional operation)*.

This method transfers the entire content of the given source byte array into this buffer. An invocation of this method of the form dst.put(a) behaves in exactly the same way as the invocation

dst.put(a, 0, a.length)

**Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there is insufficient space in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### hasArray

public final boolean **hasArray**()

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

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

**Specified by:**[hasArray](http://docs.google.com/java/nio/Buffer.html#hasArray()) in class [Buffer](http://docs.google.com/java/nio/Buffer.html) **Returns:**true if, and only if, this buffer is backed by an array and is not read-only

### array

public final byte[] **array**()

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

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/ByteBuffer.html#hasArray()) method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Specified by:**[array](http://docs.google.com/java/nio/Buffer.html#array()) in class [Buffer](http://docs.google.com/java/nio/Buffer.html) **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

### arrayOffset

public final 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/ByteBuffer.html#hasArray()) method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Specified by:**[arrayOffset](http://docs.google.com/java/nio/Buffer.html#arrayOffset()) in class [Buffer](http://docs.google.com/java/nio/Buffer.html) **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

### compact

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **compact**()

Compacts this buffer  *(optional operation)*.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index *p* = position() is copied to index zero, the byte at index *p* + 1 is copied to index one, and so forth until the byte at index limit() - 1 is copied to index *n* = limit() - 1 - *p*. The buffer's position is then set to *n+1* and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative *put* method.

Invoke this method after writing data from a buffer in case the write was incomplete. The following loop, for example, copies bytes from one channel to another via the buffer buf:

buf.clear(); // Prepare buffer for use  
 while (in.read(buf) >= 0 || buf.position != 0) {  
 buf.flip();  
 out.write(buf);  
 buf.compact(); // In case of partial write  
 }

**Returns:**This buffer **Throws:** [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### isDirect

public abstract boolean **isDirect**()

Tells whether or not this byte buffer is direct.

**Specified by:**[isDirect](http://docs.google.com/java/nio/Buffer.html#isDirect()) in class [Buffer](http://docs.google.com/java/nio/Buffer.html) **Returns:**true if, and only if, this buffer is direct

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns a string summarizing the state of this buffer.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**A summary string

### hashCode

public int **hashCode**()

Returns the current hash code of this buffer.

The hash code of a byte buffer depends only upon its remaining elements; that is, upon the elements from position() up to, and including, the element at limit() - 1.

Because buffer hash codes are content-dependent, it is inadvisable to use buffers as keys in hash maps or similar data structures unless it is known that their contents will not change.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**The current hash code of this buffer**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) ob)

Tells whether or not this buffer is equal to another object.

Two byte buffers are equal if, and only if,

1. They have the same element type,
2. They have the same number of remaining elements, and
3. The two sequences of remaining elements, considered independently of their starting positions, are pointwise equal.

A byte buffer is not equal to any other type of object.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**ob - The object to which this buffer is to be compared **Returns:**true if, and only if, this buffer is equal to the given object**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### compareTo

public int **compareTo**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) that)

Compares this buffer to another.

Two byte buffers are compared by comparing their sequences of remaining elements lexicographically, without regard to the starting position of each sequence within its corresponding buffer.

A byte buffer is not comparable to any other type of object.

**Specified by:**[compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) in interface [Comparable](http://docs.google.com/java/lang/Comparable.html)<[ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)> **Parameters:**that - the object to be compared. **Returns:**A negative integer, zero, or a positive integer as this buffer is less than, equal to, or greater than the given buffer

### order

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

Retrieves this buffer's byte order.

The byte order is used when reading or writing multibyte values, and when creating buffers that are views of this byte buffer. The order of a newly-created byte buffer is always [BIG\_ENDIAN](http://docs.google.com/java/nio/ByteOrder.html#BIG_ENDIAN).

**Returns:**This buffer's byte order

### order

public final [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **order**([ByteOrder](http://docs.google.com/java/nio/ByteOrder.html) bo)

Modifies this buffer's byte order.

**Parameters:**bo - The new byte order, either [BIG\_ENDIAN](http://docs.google.com/java/nio/ByteOrder.html#BIG_ENDIAN) or [LITTLE\_ENDIAN](http://docs.google.com/java/nio/ByteOrder.html#LITTLE_ENDIAN) **Returns:**This buffer

### getChar

public abstract char **getChar**()

Relative *get* method for reading a char value.

Reads the next two bytes at this buffer's current position, composing them into a char value according to the current byte order, and then increments the position by two.

**Returns:**The char value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than two bytes remaining in this buffer

### putChar

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putChar**(char value)

Relative *put* method for writing a char value  *(optional operation)*.

Writes two bytes containing the given char value, in the current byte order, into this buffer at the current position, and then increments the position by two.

**Parameters:**value - The char value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than two bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getChar

public abstract char **getChar**(int index)

Absolute *get* method for reading a char value.

Reads two bytes at the given index, composing them into a char value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The char value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus one

### putChar

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putChar**(int index,  
 char value)

Absolute *put* method for writing a char value  *(optional operation)*.

Writes two bytes containing the given char value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The char value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus one [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asCharBuffer

public abstract [CharBuffer](http://docs.google.com/java/nio/CharBuffer.html) **asCharBuffer**()

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new char buffer

### getShort

public abstract short **getShort**()

Relative *get* method for reading a short value.

Reads the next two bytes at this buffer's current position, composing them into a short value according to the current byte order, and then increments the position by two.

**Returns:**The short value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than two bytes remaining in this buffer

### putShort

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putShort**(short value)

Relative *put* method for writing a short value  *(optional operation)*.

Writes two bytes containing the given short value, in the current byte order, into this buffer at the current position, and then increments the position by two.

**Parameters:**value - The short value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than two bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getShort

public abstract short **getShort**(int index)

Absolute *get* method for reading a short value.

Reads two bytes at the given index, composing them into a short value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The short value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus one

### putShort

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putShort**(int index,  
 short value)

Absolute *put* method for writing a short value  *(optional operation)*.

Writes two bytes containing the given short value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The short value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus one [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asShortBuffer

public abstract [ShortBuffer](http://docs.google.com/java/nio/ShortBuffer.html) **asShortBuffer**()

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new short buffer

### getInt

public abstract int **getInt**()

Relative *get* method for reading an int value.

Reads the next four bytes at this buffer's current position, composing them into an int value according to the current byte order, and then increments the position by four.

**Returns:**The int value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than four bytes remaining in this buffer

### putInt

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putInt**(int value)

Relative *put* method for writing an int value  *(optional operation)*.

Writes four bytes containing the given int value, in the current byte order, into this buffer at the current position, and then increments the position by four.

**Parameters:**value - The int value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than four bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getInt

public abstract int **getInt**(int index)

Absolute *get* method for reading an int value.

Reads four bytes at the given index, composing them into a int value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The int value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus three

### putInt

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putInt**(int index,  
 int value)

Absolute *put* method for writing an int value  *(optional operation)*.

Writes four bytes containing the given int value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The int value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus three [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asIntBuffer

public abstract [IntBuffer](http://docs.google.com/java/nio/IntBuffer.html) **asIntBuffer**()

Creates a view of this byte buffer as an int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new int buffer

### getLong

public abstract long **getLong**()

Relative *get* method for reading a long value.

Reads the next eight bytes at this buffer's current position, composing them into a long value according to the current byte order, and then increments the position by eight.

**Returns:**The long value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than eight bytes remaining in this buffer

### putLong

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putLong**(long value)

Relative *put* method for writing a long value  *(optional operation)*.

Writes eight bytes containing the given long value, in the current byte order, into this buffer at the current position, and then increments the position by eight.

**Parameters:**value - The long value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than eight bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getLong

public abstract long **getLong**(int index)

Absolute *get* method for reading a long value.

Reads eight bytes at the given index, composing them into a long value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The long value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus seven

### putLong

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putLong**(int index,  
 long value)

Absolute *put* method for writing a long value  *(optional operation)*.

Writes eight bytes containing the given long value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The long value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus seven [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asLongBuffer

public abstract [LongBuffer](http://docs.google.com/java/nio/LongBuffer.html) **asLongBuffer**()

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new long buffer

### getFloat

public abstract float **getFloat**()

Relative *get* method for reading a float value.

Reads the next four bytes at this buffer's current position, composing them into a float value according to the current byte order, and then increments the position by four.

**Returns:**The float value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than four bytes remaining in this buffer

### putFloat

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putFloat**(float value)

Relative *put* method for writing a float value  *(optional operation)*.

Writes four bytes containing the given float value, in the current byte order, into this buffer at the current position, and then increments the position by four.

**Parameters:**value - The float value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than four bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getFloat

public abstract float **getFloat**(int index)

Absolute *get* method for reading a float value.

Reads four bytes at the given index, composing them into a float value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The float value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus three

### putFloat

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putFloat**(int index,  
 float value)

Absolute *put* method for writing a float value  *(optional operation)*.

Writes four bytes containing the given float value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The float value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus three [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asFloatBuffer

public abstract [FloatBuffer](http://docs.google.com/java/nio/FloatBuffer.html) **asFloatBuffer**()

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new float buffer

### getDouble

public abstract double **getDouble**()

Relative *get* method for reading a double value.

Reads the next eight bytes at this buffer's current position, composing them into a double value according to the current byte order, and then increments the position by eight.

**Returns:**The double value at the buffer's current position **Throws:** [BufferUnderflowException](http://docs.google.com/java/nio/BufferUnderflowException.html) - If there are fewer than eight bytes remaining in this buffer

### putDouble

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putDouble**(double value)

Relative *put* method for writing a double value  *(optional operation)*.

Writes eight bytes containing the given double value, in the current byte order, into this buffer at the current position, and then increments the position by eight.

**Parameters:**value - The double value to be written **Returns:**This buffer **Throws:** [BufferOverflowException](http://docs.google.com/java/nio/BufferOverflowException.html) - If there are fewer than eight bytes remaining in this buffer [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### getDouble

public abstract double **getDouble**(int index)

Absolute *get* method for reading a double value.

Reads eight bytes at the given index, composing them into a double value according to the current byte order.

**Parameters:**index - The index from which the bytes will be read **Returns:**The double value at the given index **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus seven

### putDouble

public abstract [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) **putDouble**(int index,  
 double value)

Absolute *put* method for writing a double value  *(optional operation)*.

Writes eight bytes containing the given double value, in the current byte order, into this buffer at the given index.

**Parameters:**index - The index at which the bytes will be writtenvalue - The double value to be written **Returns:**This buffer **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If index is negative or not smaller than the buffer's limit, minus seven [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - If this buffer is read-only

### asDoubleBuffer

public abstract [DoubleBuffer](http://docs.google.com/java/nio/DoubleBuffer.html) **asDoubleBuffer**()

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be direct if, and only if, this buffer is direct, and it will be read-only if, and only if, this buffer is read-only.

**Returns:**A new double buffer

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ByteBuffer.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/java/nio/BufferUnderflowException.html)   [**NEXT CLASS**](http://docs.google.com/java/nio/ByteOrder.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/ByteBuffer.html)    [**NO FRAMES**](http://docs.google.com/ByteBuffer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3dy6vkm) | DETAIL: FIELD | CONSTR | [METHOD](#2s8eyo1) |

[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).