compact1, compact2, compact3

java.io

Class BufferedReader

java.lang.Object
    java.io.Reader
        java.io.BufferedReader

All Implemented Interfaces:

`Closeable, AutoCloseable, Readable`

Direct Known Subclasses:

`LineNumberReader`

```
public class BufferedReader
extends Reader
```

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

The buffer size may be specified, or the default size may be used. The default is large enough for most purposes.

In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream. It is therefore advisable to wrap a BufferedReader around any Reader whose read() operations may be costly, such as FileReaders and InputStreamReaders. For example,

```
 BufferedReader in
   = new BufferedReader(new FileReader("foo.in"));
```

will buffer the input from the specified file. Without buffering, each invocation of read() or readLine() could cause bytes to be read from the file, converted into characters, and then returned, which can be very inefficient.

Programs that use DataInputStreams for textual input can be localized by replacing each DataInputStream with an appropriate BufferedReader.

Since:

`JDK1.1`

See Also:

`FileReader, InputStreamReader, Files.newBufferedReader(java.nio.file.Path, java.nio.charset.Charset)`

Field Summary

Fields inherited from class java.io.Reader

lock

Constructor Summary

Constructors

Constructor and Description
`BufferedReader(Reader in)` Creates a buffering character-input stream that uses a default-sized input buffer.
`BufferedReader(Reader in, int sz)` Creates a buffering character-input stream that uses an input buffer of the specified size.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`close()` Closes the stream and releases any system resources associated with it.
`Stream<String>`	`lines()` Returns a `Stream`, the elements of which are lines read from this `BufferedReader`.
`void`	`mark(int readAheadLimit)` Marks the present position in the stream.
`boolean`	`markSupported()` Tells whether this stream supports the mark() operation, which it does.
`int`	`read()` Reads a single character.
`int`	`read(char[] cbuf, int off, int len)` Reads characters into a portion of an array.
`String`	`readLine()` Reads a line of text.
`boolean`	`ready()` Tells whether this stream is ready to be read.
`void`	`reset()` Resets the stream to the most recent mark.
`long`	`skip(long n)`

Skips characters.

Methods inherited from class java.io.Reader

`read, read`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

BufferedReader

```
public BufferedReader(Reader in,
                      int sz)
```

Creates a buffering character-input stream that uses an input buffer of the specified size.

Parameters:

`in - A Reader`

`sz - Input-buffer size`

Throws:

`IllegalArgumentException - If sz <= 0`

BufferedReader

```
public BufferedReader(Reader in)
```

Creates a buffering character-input stream that uses a default-sized input buffer.

Parameters:

`in - A Reader`

Method Detail

read

```
public int read()
         throws IOException
```

Reads a single character.

Overrides:

`read in class Reader`

Returns:

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached

Throws:

IOException - If an I/O error occurs

read

```
public int read(char[] cbuf,
                int off,
                int len)
         throws IOException
```

Reads characters into a portion of an array.

This method implements the general contract of the corresponding `read` method of the `Reader` class. As an additional convenience, it attempts to read as many characters as possible by repeatedly invoking the `read` method of the underlying stream. This iterated `read` continues until one of the following conditions becomes true:

- The specified number of characters have been read,
- The `read` method of the underlying stream returns `-1`, indicating end-of-file, or
- The `ready` method of the underlying stream returns `false`, indicating that further input requests would block.

If the first `read` on the underlying stream returns `-1` to indicate end-of-file then this method returns `-1`. Otherwise this method returns the number of characters actually read.

Subclasses of this class are encouraged, but not required, to attempt to read as many characters as possible in the same fashion.

Ordinarily this method takes characters from this stream's character buffer, filling it from the underlying stream as necessary. If, however, the buffer is empty, the mark is not valid, and the requested length is at least as large as the buffer, then this method will read characters directly from the underlying stream into the given array. Thus redundant `BufferedReader`s will not copy data unnecessarily.

Specified by:

read in class Reader

Parameters:

cbuf - Destination buffer

off - Offset at which to start storing characters

len - Maximum number of characters to read

Returns:

The number of characters read, or -1 if the end of the stream has been reached

Throws:

IOException - If an I/O error occurs

readLine

```
public String readLine()
                throws IOException
```

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a linefeed.

Returns:

A String containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached

Throws:

IOException - If an I/O error occurs

See Also:

Files.readAllLines(java.nio.file.Path, java.nio.charset.Charset)

skip

```
public long skip(long n)
          throws IOException
```

Skips characters.

Overrides:

skip in class Reader

Parameters:

n - The number of characters to skip

Returns:

The number of characters actually skipped

Throws:

IllegalArgumentException - If n is negative.

IOException - If an I/O error occurs

ready

```
public boolean ready()
              throws IOException
```

Tells whether this stream is ready to be read. A buffered character stream is ready if the buffer is not empty, or if the underlying character stream is ready.

Overrides:

ready in class Reader

Returns:

True if the next read() is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Throws:

IOException - If an I/O error occurs

markSupported

```
public boolean markSupported()
```

Tells whether this stream supports the mark() operation, which it does.

Overrides:

`markSupported` in class `Reader`

Returns:

`true if and only if this stream supports the mark operation.`

mark

```
public void mark(int readAheadLimit)
          throws IOException
```

Marks the present position in the stream. Subsequent calls to reset() will attempt to reposition the stream to this point.

Overrides:

`mark` in class `Reader`

Parameters:

`readAheadLimit` - Limit on the number of characters that may be read while still preserving the mark. An attempt to reset the stream after reading characters up to this limit or beyond may fail. A limit value larger than the size of the input buffer will cause a new buffer to be allocated whose size is no smaller than limit. Therefore large values should be used with care.

Throws:

`IllegalArgumentException` - If readAheadLimit < 0

`IOException` - If an I/O error occurs

reset

```
public void reset()
           throws IOException
```

Resets the stream to the most recent mark.

Overrides:

`reset` in class `Reader`

Throws:

`IOException` - If the stream has never been marked, or if the mark has been invalidated

close

```
public void close()
           throws IOException
```

Description copied from class: Reader

Closes the stream and releases any system resources associated with it. Once the stream has been closed, further read(), ready(), mark(), reset(), or skip() invocations will throw an IOException. Closing a previously closed stream has no effect.

Specified by:

`close` in interface `Closeable`

Specified by:

`close` in interface `AutoCloseable`

Specified by:

`close` in class `Reader`

Throws:

`IOException` - If an I/O error occurs

lines

```
public Stream<String> lines()
```

Returns a `Stream`, the elements of which are lines read from this `BufferedReader`. The `Stream` is lazily populated, i.e., read only occurs during the terminal stream operation.

The reader must not be operated on during the execution of the terminal stream operation. Otherwise, the result of the terminal stream operation is undefined.

After execution of the terminal stream operation there are no guarantees that the reader will be at a specific position from which to read the next character or line.

If an `IOException` is thrown when accessing the underlying `BufferedReader`, it is wrapped in an `UncheckedIOException` which will be thrown from the `Stream` method that caused the read to take place. This method will return a Stream if invoked on a BufferedReader that is closed. Any operation on that stream that requires reading from the BufferedReader after it is closed, will cause an UncheckedIOException to be thrown.

Returns:

a Stream<String> providing the lines of text described by this BufferedReader

Since:

1.8

Submit a bug or feature
For further API reference and developer documentation, see Java SE Documentation. That documentation

contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.