| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PushbackReader.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/io/PushbackInputStream.html)   [**NEXT CLASS**](http://docs.google.com/java/io/RandomAccessFile.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/PushbackReader.html)    [**NO FRAMES**](http://docs.google.com/PushbackReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **java.io**

Class PushbackReader

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.io.Reader](http://docs.google.com/java/io/Reader.html)  
 [java.io.FilterReader](http://docs.google.com/java/io/FilterReader.html)  
 **java.io.PushbackReader**

**All Implemented Interfaces:** [Closeable](http://docs.google.com/java/io/Closeable.html), [Readable](http://docs.google.com/java/lang/Readable.html)

public class **PushbackReader**extends [FilterReader](http://docs.google.com/java/io/FilterReader.html)

A character-stream reader that allows characters to be pushed back into the stream.

**Since:** JDK1.1

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.io.**[**FilterReader**](http://docs.google.com/java/io/FilterReader.html) |
| --- |
| [in](http://docs.google.com/java/io/FilterReader.html#in) |

| **Fields inherited from class java.io.**[**Reader**](http://docs.google.com/java/io/Reader.html) |
| --- |
| [lock](http://docs.google.com/java/io/Reader.html#lock) |

| **Constructor Summary** | |
| --- | --- |
| [**PushbackReader**](http://docs.google.com/java/io/PushbackReader.html#PushbackReader(java.io.Reader))([Reader](http://docs.google.com/java/io/Reader.html) in)            Creates a new pushback reader with a one-character pushback buffer. |
| [**PushbackReader**](http://docs.google.com/java/io/PushbackReader.html#PushbackReader(java.io.Reader,%20int))([Reader](http://docs.google.com/java/io/Reader.html) in, int size)            Creates a new pushback reader with a pushback buffer of the given size. |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/java/io/PushbackReader.html#close())()            Closes the stream and releases any system resources associated with it. |
| void | [**mark**](http://docs.google.com/java/io/PushbackReader.html#mark(int))(int readAheadLimit)            Marks the present position in the stream. |
| boolean | [**markSupported**](http://docs.google.com/java/io/PushbackReader.html#markSupported())()            Tells whether this stream supports the mark() operation, which it does not. |
| int | [**read**](http://docs.google.com/java/io/PushbackReader.html#read())()            Reads a single character. |
| int | [**read**](http://docs.google.com/java/io/PushbackReader.html#read(char%5B%5D,%20int,%20int))(char[] cbuf, int off, int len)            Reads characters into a portion of an array. |
| boolean | [**ready**](http://docs.google.com/java/io/PushbackReader.html#ready())()            Tells whether this stream is ready to be read. |
| void | [**reset**](http://docs.google.com/java/io/PushbackReader.html#reset())()            Resets the stream. |
| long | [**skip**](http://docs.google.com/java/io/PushbackReader.html#skip(long))(long n)            Skips characters. |
| void | [**unread**](http://docs.google.com/java/io/PushbackReader.html#unread(char%5B%5D))(char[] cbuf)            Pushes back an array of characters by copying it to the front of the pushback buffer. |
| void | [**unread**](http://docs.google.com/java/io/PushbackReader.html#unread(char%5B%5D,%20int,%20int))(char[] cbuf, int off, int len)            Pushes back a portion of an array of characters by copying it to the front of the pushback buffer. |
| void | [**unread**](http://docs.google.com/java/io/PushbackReader.html#unread(int))(int c)            Pushes back a single character by copying it to the front of the pushback buffer. |

| **Methods inherited from class java.io.**[**Reader**](http://docs.google.com/java/io/Reader.html) |
| --- |
| [read](http://docs.google.com/java/io/Reader.html#read(char%5B%5D)), [read](http://docs.google.com/java/io/Reader.html#read(java.nio.CharBuffer)) |

| **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)) |

| **Constructor Detail** |
| --- |

### PushbackReader

public **PushbackReader**([Reader](http://docs.google.com/java/io/Reader.html) in,  
 int size)

Creates a new pushback reader with a pushback buffer of the given size.

**Parameters:**in - The reader from which characters will be readsize - The size of the pushback buffer **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is <= 0

### PushbackReader

public **PushbackReader**([Reader](http://docs.google.com/java/io/Reader.html) in)

Creates a new pushback reader with a one-character pushback buffer.

**Parameters:**in - The reader from which characters will be read

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

### read

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

Reads a single character.

**Overrides:**[read](http://docs.google.com/java/io/FilterReader.html#read()) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Returns:**The character read, or -1 if the end of the stream has been reached **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

### read

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

Reads characters into a portion of an array.

**Overrides:**[read](http://docs.google.com/java/io/FilterReader.html#read(char%5B%5D,%20int,%20int)) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Parameters:**cbuf - Destination bufferoff - Offset at which to start writing characterslen - 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](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

### unread

public void **unread**(int c)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Pushes back a single character by copying it to the front of the pushback buffer. After this method returns, the next character to be read will have the value (char)c.

**Parameters:**c - The int value representing a character to be pushed back **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If the pushback buffer is full, or if some other I/O error occurs

### unread

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

Pushes back a portion of an array of characters by copying it to the front of the pushback buffer. After this method returns, the next character to be read will have the value cbuf[off], the character after that will have the value cbuf[off+1], and so forth.

**Parameters:**cbuf - Character arrayoff - Offset of first character to push backlen - Number of characters to push back **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If there is insufficient room in the pushback buffer, or if some other I/O error occurs

### unread

public void **unread**(char[] cbuf)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Pushes back an array of characters by copying it to the front of the pushback buffer. After this method returns, the next character to be read will have the value cbuf[0], the character after that will have the value cbuf[1], and so forth.

**Parameters:**cbuf - Character array to push back **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If there is insufficient room in the pushback buffer, or if some other I/O error occurs

### ready

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

Tells whether this stream is ready to be read.

**Overrides:**[ready](http://docs.google.com/java/io/FilterReader.html#ready()) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **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](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

### mark

public void **mark**(int readAheadLimit)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Marks the present position in the stream. The mark for class PushbackReader always throws an exception.

**Overrides:**[mark](http://docs.google.com/java/io/FilterReader.html#mark(int)) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Parameters:**readAheadLimit - Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Always, since mark is not supported

### reset

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

Resets the stream. The reset method of PushbackReader always throws an exception.

**Overrides:**[reset](http://docs.google.com/java/io/FilterReader.html#reset()) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Always, since reset is not supported

### markSupported

public boolean **markSupported**()

Tells whether this stream supports the mark() operation, which it does not.

**Overrides:**[markSupported](http://docs.google.com/java/io/FilterReader.html#markSupported()) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Returns:**true if and only if this stream supports the mark operation.

### close

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

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

**Specified by:**[close](http://docs.google.com/java/io/Closeable.html#close()) in interface [Closeable](http://docs.google.com/java/io/Closeable.html)**Overrides:**[close](http://docs.google.com/java/io/FilterReader.html#close()) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

### skip

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

Skips characters. This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

**Overrides:**[skip](http://docs.google.com/java/io/FilterReader.html#skip(long)) in class [FilterReader](http://docs.google.com/java/io/FilterReader.html) **Parameters:**n - The number of characters to skip **Returns:**The number of characters actually skipped **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If n is negative. [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PushbackReader.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/io/PushbackInputStream.html)   [**NEXT CLASS**](http://docs.google.com/java/io/RandomAccessFile.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/PushbackReader.html)    [**NO FRAMES**](http://docs.google.com/PushbackReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[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).