
Class java.io.BufferedInputStream

```
java.lang.Object
   |
   +----java.io.InputStream
           |
           +----java.io.FilterInputStream
                   |
                   +----java.io.BufferedInputStream
```

---

public class **BufferedInputStream**
extends FilterInputStream

A buffered input stream. This stream lets you read in characters from a stream without causing a read every time. The data is read into a buffer, subsequent reads result in a fast buffer access.

Version:
1.20, 08/18/95
Author:
Arthur van Hoff

Variable Index

o **buf**
The buffer where data is stored.
o **count**
The number of bytes in the buffer.
o **marklimit**
The maximum readahead allowed after a mark() before subsequent calls to reset() fail.
o **markpos**
The position in the buffer of the current mark.
o **pos**
The current position in the buffer.

Constructor Index

Method Index

Variables

o **buf**

```
protected byte buf[]
```

The buffer where data is stored.

o **count**

```
protected int count
```

The number of bytes in the buffer.

o **pos**

```
protected int pos
```

The current position in the buffer.

o **markpos**

```
protected int markpos
```

The position in the buffer of the current mark. This mark is set to –1 if there is no current mark.

- **marklimit**

```
protected int marklimit
```

The maximum readahead allowed after a mark() before subsequent calls to reset() fail.

Constructors

- **BufferedInputStream**

```
public BufferedInputStream(InputStream in)
```

Creates a new buffered stream with a default buffer size.
Parameters:
in – the input stream

- **BufferedInputStream**

```
public BufferedInputStream(InputStream in,
                           int size)
```

Creates a new buffered stream with the specified buffer size.
Parameters:
in – the input stream
size – the buffer size

Methods

- **read**

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

Reads a byte of data. This method will block if no input is available.
Returns:
the byte read, or –1 if the end of the stream is reached.
Throws: IOException
If an I/O error has occurred.
Overrides:
read in class FilterInputStream

- **read**

```
public synchronized int read(byte b[],
                             int off,
```

```
int len) throws IOException
```

Reads into an array of bytes. Blocks until some input is available.

Parameters:

b – the buffer into which the data is read
off – the start offset of the data
len – the maximum number of bytes read

Returns:

the actual number of bytes read, –1 is returned when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class FilterInputStream

- **skip**

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

Skips n bytes of input.

Parameters:

n – the number of bytes to be skipped

Returns:

the actual number of bytes skipped.

Throws: IOException

If an I/O error has occurred.

Overrides:

skip in class FilterInputStream

- **available**

```
public synchronized int available() throws IOException
```

Returns the number of bytes that can be read without blocking. This total is the number of bytes in the buffer and the number of bytes available from the input stream.

Returns:

the number of available bytes.

Overrides:

available in class FilterInputStream

- **mark**

```
public synchronized void mark(int readlimit)
```

Marks the current position in the input stream. A subsequent call to the reset() method will reposition the stream at the last marked position so that subsequent reads will re–read the same bytes. The stream promises to allow readlimit bytes to be read before the mark position gets invalidated.

Parameters:
readlimit – the maximum limit of bytes allowed to be read before the mark position becomes invalid.
Overrides:
mark in class FilterInputStream

- **reset**

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

Repositions the stream to the last marked position. If the stream has not been marked, or if the mark has been invalidated, an IOException is thrown. Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parser, it just chugs along happily. If the stream is *not* of that type, the parser should toss an exception when it fails. If an exception gets tossed within readlimit bytes, the parser will allow the outer code to reset the stream and to try another parser.
Throws: IOException
If the stream has not been marked or if the mark has been invalidated.
Overrides:
reset in class FilterInputStream

- **markSupported**

```
public boolean markSupported()
```

Returns a boolean indicating if this stream type supports mark/reset.
Overrides:
markSupported in class FilterInputStream
