
Class java.io.FilterInputStream

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

---

public class **FilterInputStream**
extends InputStream

Abstract class representing a filtered input stream of bytes. This class is the basis for enhancing input stream functionality. It allows multiple input stream filters to be chained together, each providing additional functionality.

Version:
1.12, 08/13/95
Author:
Jonathan Payne

Variable Index

o **in**
The actual input stream.

Constructor Index

o **FilterInputStream**(InputStream)
Creates an input stream filter.

Method Index

o **available**()
Returns the number of bytes that can be read.
o **close**()
Closes the input stream.
o **mark**(int)

Marks the current position in the input stream.

- **markSupported**()
  Returns true if this stream type supports mark/reset
- **read**()
  Reads a byte.
- **read**(byte[])
  Reads into an array of bytes.
- **read**(byte[], int, int)
  Reads into an array of bytes.
- **reset**()
  Repositions the stream to the last marked position.
- **skip**(long)
  Skips bytes of input.

Variables

- **in**

```
protected InputStream in
```

The actual input stream.

Constructors

- **FilterInputStream**

```
protected FilterInputStream(InputStream in)
```

Creates an input stream filter.
Parameters:
in – the input stream

Methods

- **read**

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

Reads a byte. 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 InputStream

- **read**

```
public int read(byte b[]) throws IOException
```

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

Parameters:

b – the buffer into which the data is 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 InputStream

- **read**

```
public int read(byte b[],
                int off,
                int len) throws IOException
```

Reads into an array of bytes. Blocks until some input is available. This method should be overridden in a subclass for efficiency (the default implementation reads 1 byte at a time).

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 InputStream

- **skip**

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

Skips bytes of input.

Parameters:

n – bytes to be skipped

Returns:

actual number of bytes skipped

Throws: IOException

If an I/O error has occurred.

Overrides:

skip in class InputStream

o **available**

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

Returns the number of bytes that can be read. without blocking.

Returns:

the number of available bytes

Overrides:

available in class InputStream

o **close**

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

Closes the input stream. Must be called to release any resources associated with the stream.

Throws: IOException

If an I/O error has occurred.

Overrides:

close in class InputStream

o **mark**

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

Marks the current position in the input stream. A subsequent call to reset() 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 tobe read before the mark position becomes invalid.

Overrides:

mark in class InputStream

o **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 parse, it just chugs along happily. If the stream is *not* of that type, the parser should toss an exception when it fails, which, if it happens within readlimit bytes, allows the outer code to reset the stream and try another parser.

Overrides:

reset in class InputStream

o **markSupported**

```
public boolean markSupported()
```

Returns true if this stream type supports mark/reset

Overrides:

markSupported in class InputStream
