Class java.io.LineNumberInputStream

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

---

public class **LineNumberInputStream**
extends FilterInputStream

An input stream that keeps track of line numbers.

Version:
1.7, 08/10/95
Author:
Arthur van Hoff

Constructor Index

o **LineNumberInputStream**(InputStream)
Constructs a new LineNumberInputStream initialized with the specified input stream

Method Index

o **available**()
Returns the number of bytes that can be read.
o **getLineNumber**()
Returns the current line number.
o **mark**(int)
Marks the current position in the input stream.
o **read**()
Reads a byte of data.
o **read**(byte[], int, int)
Reads into an array of bytes.
o **reset**()

Repositions the stream to the last marked position.

- **setLineNumber**(int)
  Sets the current line number.
- **skip**(long)
  Skips n bytes of input.

Constructors

- **LineNumberInputStream**

```
public LineNumberInputStream(InputStream in)
```

Constructs a new LineNumberInputStream initialized with the specified input stream

Parameters:

in – the input stream

Methods

- **read**

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

Reads a byte of data. The 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 int read(byte b[],
                int off,
                int len) throws IOException
```

Reads into an array of bytes. This method will 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

o **setLineNumber**

```
public void setLineNumber(int lineNumber)
```

Sets the current line number.
Parameters:
lineNumber – the line number to be set

o **getLineNumber**

```
public int getLineNumber()
```

Returns the current line number.

o **skip**

```
public 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

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 FilterInputStream

o **mark**

```
public 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 to be read before the mark position becomes invalid.
Overrides:

mark in class FilterInputStream

o **reset**

```
public 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, which, if it happens within readlimit bytes, allows the outer code to reset the stream and try another parser.

Overrides:

reset in class FilterInputStream