
Class java.io.InputStream

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

---

public class **InputStream**
extends Object

An abstract class representing an input stream of bytes. All InputStreams are based on this class.

See Also:
OutputStream, FilterInputStream, BufferedInputStream, DataInputStream, ByteArrayInputStream, PushbackInputStream

Version:
1.15, 08/11/95

Author:
Arthur van Hoff

Constructor Index

Method Index

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 n bytes of input.

Constructors

- **InputStream**

```
public InputStream()
```

Methods

- **read**

```
public abstract 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.

- **read**

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

Reads into an array of bytes. This method will block 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.

- **read**

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

Reads into an array of bytes. This method will block 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.

- **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.

- **available**

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

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

Returns:

the number of available bytes.

- **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.

- **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 to be read before the mark position becomes invalid.

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 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.

Throws: IOException

If the stream has not been marked or if the mark has been invalidated.

o **markSupported**

```
public boolean markSupported()
```

Returns a boolean indicating whether or not this stream type supports mark/reset.

Returns:

true if this stream type supports mark/reset; false otherwise.
