package java.io

Interface Index

- <u>DataInput</u>
- <u>DataOutput</u>
- <u>FilenameFilter</u>

Class Index

- <u>BufferedInputStream</u>
- <u>BufferedOutputStream</u>
- <u>ByteArrayInputStream</u>
- <u>ByteArrayOutputStream</u>
- <u>DataInputStream</u>
- <u>DataOutputStream</u>
- <u>File</u>
- <u>FileDescriptor</u>
- <u>FileInputStream</u>
- <u>FileOutputStream</u>
- <u>FilterInputStream</u>
- <u>FilterOutputStream</u>
- InputStream
- <u>LineNumberInputStream</u>
- <u>OutputStream</u>
- <u>PipedInputStream</u>
- <u>PipedOutputStream</u>
- <u>PrintStream</u>
- <u>PushbackInputStream</u>
- <u>RandomAccessFile</u>
- <u>SequenceInputStream</u>
- <u>StreamTokenizer</u>
- <u>StringBufferInputStream</u>

Exception Index

- <u>EOFException</u>
- <u>FileNotFoundException</u>

- <u>IOException</u>
 <u>InterruptedIOException</u>
 <u>UTFDataFormatException</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.BufferedInputStream

java.lang.Object

----<u>java.io.InputStream</u> | +----<u>java.io.FilterInputStream</u> | +----java.io.BufferedInputStream

 $public \ class \ \textbf{BufferedInputStream} \\ extends \ \underline{FilterInputStream}$

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.

Variable Index

<u>• buf</u>

The buffer where data is stored.

<u>• count</u>

The number of bytes in the buffer.

<u>• marklimit</u>

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

markpos

The position in the buffer of the current mark.

<u>• pos</u>

The current position in the buffer.

Constructor Index

<u>BufferedInputStream</u>(InputStream)

Creates a new buffered stream with a default buffer size.

<u>BufferedInputStream</u>(InputStream, int)

Creates a new buffered stream with the specified buffer size.

Method Index

available()

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

• **mark**(int)

Marks the current position in the input stream.

markSupported()

Returns a boolean indicating if this stream type supports mark/reset.

•<u>read(</u>)

Reads a byte of data.

• **read**(byte[], int, int)

Reads into an array of bytes.

• <u>reset(</u>)

Repositions the stream to the last marked position.

skip(long)

Skips n bytes of input.

Variables

🧶 buf

protected byte buf[]

The buffer where data is stored.

🤍 count

protected int count

The number of bytes in the buffer.

🤍 pos

protected int pos

The current position in the buffer.

🧶 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

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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>read</u> in class <u>FilterInputStream</u>

🄍 read

 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: <u>available</u> in class <u>FilterInputStream</u>

🤍 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 <u>IOException</u>

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

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.BufferedOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----<u>java.io.FilterOutputStream</u> | +----java.io.BufferedOutputStream

 $\label{eq:public class} \begin{array}{l} \textbf{BufferedOutputStream} \\ \textbf{extends} \ \underline{FilterOutputStream} \end{array}$

A buffered output stream. This stream lets you write characters to a stream without causing a write every time. The data is first written into a buffer. Data is written to the actual stream only when the buffer is full, or when the stream is flushed.

Variable Index

• <u>buf</u> The buffer where data is stored. • <u>count</u>

The number of bytes in the buffer.

Constructor Index

<u>BufferedOutputStream</u>(OutputStream)

Creates a new buffered stream with a default buffer size. **<u>BufferedOutputStream</u>**(OutputStream, int) Creates a new buffered stream with the specified buffer size.

Method Index

•<u>flush</u>() Flushes the stream. •<u>write</u>(int) Writes a byte. • <u>write</u>(byte[], int, int) Writes a subarray of bytes.



🤍 buf

protected byte buf[]

The buffer where data is stored.

🤍 count

protected int count

The number of bytes in the buffer.



➡ BufferedOutputStream

public BufferedOutputStream(OutputStream out)

Creates a new buffered stream with a default buffer size. **Parameters:** out – the output stream

BufferedOutputStream

Creates a new buffered stream with the specified buffer size. **Parameters:** out – the output stream

size - the buffer size

Methods

🔍 write

public synchronized void write(int b) throws IOException

Writes a byte. This method will block until the byte is actually written.

Parameters: b – the byte to be written Throws:<u>IOException</u> If an I/O error has occurred. Overrides: write in class FilterOutputStream

🤍 write

Writes a subarray of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>FilterOutputStream</u>

🌻 flush

public synchronized void flush() throws <u>IOException</u>

Flushes the stream. This will write any buffered output bytes. **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>flush</u> in class <u>FilterOutputStream</u>

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.ByteArrayInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----java.io.ByteArrayInputStream

public class **ByteArrayInputStream** extends <u>InputStream</u>

This class implements a buffer that can be used as an InputStream.

Variable Index

<u>• buf</u>

The buffer where data is stored.

<u>• count</u>

The number of characters to use in the buffer.

<u>pos</u>

The current position in the buffer.

Constructor Index

ByteArrayInputStream(byte[])

Creates an ByteArrayInputStream from the specified array of bytes. • <u>ByteArrayInputStream</u>(byte[], int, int)

Creates an ByteArrayInputStream from the specified array of bytes.

Method Index

<u>available()</u>
 Returns the number of available bytes in the buffer.
 <u>read()</u>
 Reads a byte of data.
 <u>read(byte[], int, int)</u>
 Reads into an array of bytes.

 <u>reset()</u> Resets the buffer to the beginning.
 <u>skip(long)</u> Skips n bytes of input.

Variables

🧕 buf

protected byte buf[]

The buffer where data is stored.

🏓 pos

protected int pos

The current position in the buffer.

🤍 count

protected int count

The number of characters to use in the buffer.

Constructors

🥥 ByteArrayInputStream

public ByteArrayInputStream(byte buf[])

Creates an ByteArrayInputStream from the specified array of bytes. **Parameters:** buf – the input buffer (not copied)

ByteArrayInputStream

Creates an ByteArrayInputStream from the specified array of bytes. **Parameters:**

buf - the input buffer (not copied)
offset - the offset of the first byte to read
length - the number of bytes to read

Methods

🤍 read

public synchronized int read()

Reads a byte of data. **Returns:** the byte read, or -1 if the end of the stream is reached. **Overrides:** <u>read</u> in class <u>InputStream</u>

🤍 read

Reads into an array of bytes.

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.

Overrides:

read in class InputStream

🌻 skip

public synchronized long skip(long n)

Skips n bytes of input. Parameters: n – the number of bytes to be skipped Returns: the actual number of bytes skipped. Overrides: skip in class InputStream

🧶 available

public synchronized int available()

Returns the number of available bytes in the buffer. **Overrides:** <u>available</u> in class <u>InputStream</u>

🧶 reset

public synchronized void reset()

Resets the buffer to the beginning. **Overrides:** <u>reset</u> in class <u>InputStream</u>

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.ByteArrayOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----java.io.ByteArrayOutputStream

public class **ByteArrayOutputStream** extends <u>OutputStream</u>

This class implements a buffer that can be used as an OutputStream. The buffer automatically grows when data is written to the stream. The data can be retrieved using toByteArray() and toString().

Variable Index

<u>buf</u>
The buffer where data is stored.

<u>count</u>

The number of bytes in the buffer.

Constructor Index

 <u>ByteArrayOutputStream()</u> Creates a new ByteArrayOutputStream.
 <u>ByteArrayOutputStream(int)</u> Creates a new ByteArrayOutputStream with the specified initial size.

Method Index

• <u>reset(</u>)

Resets the buffer so that you can use it again without throwing away the already allocated buffer.

• <u>size()</u>

Returns the current size of the buffer.

toByteArray() Returns a copy of the input data.
toString() Converts input data to a string.
toString(int) Converts input data to a string.
write(int) Writes a byte to the buffer.
write(byte[], int, int) Writes bytes to the buffer.
writeTo(OutputStream) Writes the contents of the buffer to another stream.



🤍 buf

protected byte buf[]

The buffer where data is stored.

🤍 count

protected int count

The number of bytes in the buffer.



🥥 ByteArrayOutputStream

public ByteArrayOutputStream()

Creates a new ByteArrayOutputStream.

ByteArrayOutputStream

public ByteArrayOutputStream(int size)

Creates a new ByteArrayOutputStream with the specified initial size.

Parameters:

size – the initial size

Methods

🔍 write

public synchronized void write(int b)

Writes a byte to the buffer. **Parameters:** b – the byte **Overrides:** write in class OutputStream

🧶 write

Writes bytes to the buffer. Parameters: b – the data to be written off – the start offset in the data len – the number of bytes that are written Overrides: write in class OutputStream

🔍 writeTo

public synchronized void writeTo(OutputStream out) throws IOException

Writes the contents of the buffer to another stream. **Parameters:** out – the output stream to write to

🔍 reset

public synchronized void reset()

Resets the buffer so that you can use it again without throwing away the already allocated buffer.

🔵 toByteArray

public synchronized byte[] toByteArray()

Returns a copy of the input data.

🧶 size

public int size()

Returns the current size of the buffer.

🔍 toString

```
public <u>String</u> toString()
```

Converts input data to a string. **Returns:** the string. **Overrides:** <u>toString</u> in class <u>Object</u>

🧶 toString

public <u>String</u> toString(int hibyte)

Converts input data to a string. The top 8 bits of each 16 bit Unicode character are set to hibyte.

Parameters: hibyte – the bits set

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Interface java.io.DataInput

public interface **DataInput** extends <u>Object</u>

DataInput is an interface describing streams that can read input in a machine-independent format.

See Also:

DataInputStream, DataOutput

Method Index

• readBoolean() Reads in a boolean. readByte() Reads an 8 bit byte. readChar() Reads a 16 bit char. readDouble() Reads a 64 bit double. readFloat() Reads a 32 bit float. • readFully(byte[]) Reads bytes, blocking until all bytes are read. readFully(byte[], int, int) Reads bytes, blocking until all bytes are read. readInt() Reads a 32 bit int. • readLine() readLong() Reads a 64 bit long. readShort() Reads a 16 bit short. • readUTF() • readUnsignedByte() Reads an unsigned 8 bit byte. readUnsignedShort() Reads an unsigned 16 bit short. • skipBytes(int) Skips bytes, block until all bytes are skipped.

Methods

🔍 readFully

public abstract void readFully(byte b[]) throws <u>IOException</u>

Reads bytes, blocking until all bytes are read. **Parameters:** b – the buffer into which the data is read **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🔍 readFully

Reads bytes, blocking until all bytes are read. Parameters: b – the buffer into which the data is read off – the start offset of the data len – the maximum number of bytes to read Throws:EOFException If end of file is reached. Throws:IOException If other I/O error has occurred.

🧶 skipBytes

public abstract int skipBytes(int n) throws <u>IOException</u>

Skips bytes, block until all bytes are skipped. **Parameters:** n – the number of bytes to be skipped **Returns:** the actual number of bytes skipped. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🤍 readBoolean

public abstract boolean readBoolean() throws <u>IOException</u>

Reads in a boolean. **Returns:** the boolean read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🤍 readByte

public abstract byte readByte() throws <u>IOException</u>

Reads an 8 bit byte. **Returns:** the 8 bit byte read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

readUnsignedByte

public abstract int readUnsignedByte() throws <u>IOException</u>

Reads an unsigned 8 bit byte. **Returns:** the 8 bit byte read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🛢 readShort

public abstract short readShort() throws <u>IOException</u>

Reads a 16 bit short. **Returns:** the 16 bit short read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🔍 readUnsignedShort

public abstract int readUnsignedShort() throws <u>IOException</u>

Reads an unsigned 16 bit short.

Returns: the 16 bit short read. Throws:EOFException If end of file is reached. Throws:IOException If other I/O error has occurred.

🜻 readChar

public abstract char readChar() throws <u>IOException</u>

Reads a 16 bit char. **Returns:** the 16 bit char read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🔍 readInt

public abstract int readInt() throws IOException

Reads a 32 bit int. **Returns:** the 32 bit integer read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🧶 readLong

public abstract long readLong() throws <u>IOException</u>

Reads a 64 bit long. **Returns:** the read 64 bit long. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🔍 readFloat

public abstract float readFloat() throws <u>IOException</u>

Reads a 32 bit float. **Returns:**

the 32 bit float read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🜻 readDouble

public abstract double readDouble() throws <u>IOException</u>

Reads a 64 bit double. **Returns:** the 64 bit double read. **Throws:**<u>EOFException</u> If end of file is reached. **Throws:**<u>IOException</u> If other I/O error has occurred.

🧶 readLine

public abstract <u>String</u> readLine() throws <u>IOException</u>

🜻 readUTF

public abstract <u>String</u> readUTF() throws <u>IOException</u>

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.DataInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----<u>java.io.FilterInputStream</u> | +----java.io.DataInputStream

public class **DataInputStream** extends <u>FilterInputStream</u> implements <u>DataInput</u>

A data input stream that lets you read primitive Java data types from a stream in a portable way. Primitive data types are well understood types with associated operations. For example, Integers are considered primitive data types.

See Also: DataOutputStream

Constructor Index

•<u>DataInputStream</u>(InputStream) Creates a new DataInputStream.

Method Index

• <u>read</u>(byte[]) Reads data into an array of bytes. • <u>read</u>(byte[], int, int) Reads data into an array of bytes. • <u>readBoolean</u>() Reads a boolean. • <u>readByte()</u> Reads an 8 bit byte. • <u>readChar()</u> Reads a 16 bit char. • readDouble() Reads a 64 bit double. readFloat() Reads a 32 bit float. readFully(byte[]) Reads bytes, blocking until all bytes are read. • readFully(byte[], int, int) Reads bytes, blocking until all bytes are read. readInt() Reads a 32 bit int. • readLine() Reads in a line that has been terminated by a n, r, r nor EOF. readLong() Reads a 64 bit long. readShort() Reads a 16 bit short. readUTF() Reads a UTF format String. readUTF(DataInput) Reads a UTF format String from the given input stream. • readUnsignedByte() Reads an unsigned 8 bit byte. readUnsignedShort() Reads 16 bit short. skipBytes(int) Skips bytes, blocks until all bytes are skipped.

Constructors

🥯 DataInputStream

public DataInputStream(InputStream in)

Creates a new DataInputStream. **Parameters:** in – the input stream

Methods

🧶 read

public final int read(byte b[]) throws <u>IOException</u>

Reads data into an array of bytes. This method 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 FilterInputStream

🄍 read

Reads data into an array of bytes. This method 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

🔍 readFully

public final void readFully(byte b[]) throws <u>IOException</u>

Reads bytes, blocking until all bytes are read. **Parameters:** b – the buffer into which the data is read **Throws:**<u>IOException</u> If an I/O error has occurred. **Throws:**<u>EOFException</u> If EOF reached before all bytes are read.

🤍 readFully

Reads bytes, blocking until all bytes are read. **Parameters:**

b – the buffer into which the data is read

off – the start offset of the data len – the maximum number of bytes read **Throws:**<u>IOException</u> If an I/O error has occurred. **Throws:**<u>EOFException</u> If EOF reached before all bytes are read.

skipBytes

public final int skipBytes(int n) throws <u>IOException</u>

Skips bytes, blocks until all bytes are skipped. **Parameters:** n – the number of bytes to be skipped **Returns:** the actual number of bytes skipped. **Throws:**<u>IOException</u> If an I/O error has occurred.

🌻 readBoolean

public final boolean readBoolean() throws <u>IOException</u>

Reads a boolean. **Returns:** the boolean read.

🤍 readByte

public final byte readByte() throws <u>IOException</u>

Reads an 8 bit byte. **Returns:** the 8 bit byte read.

readUnsignedByte

public final int readUnsignedByte() throws <u>IOException</u>

Reads an unsigned 8 bit byte. Returns: the 8 bit byte read.

🤍 readShort

public final short readShort() throws <u>IOException</u>

Reads a 16 bit short. **Returns:** the 16 bit short read.

readUnsignedShort

public final int readUnsignedShort() throws <u>IOException</u>

Reads 16 bit short. Returns: the 16 bit short read.

🜻 readChar

public final char readChar() throws <u>IOException</u>

Reads a 16 bit char. **Returns:** the read 16 bit char.

🔍 readInt

public final int readInt() throws <u>IOException</u>

Reads a 32 bit int. **Returns:** the 32 bit integer read.

🧶 readLong

public final long readLong() throws <u>IOException</u>

Reads a 64 bit long. **Returns:** the 64 bit long read.

🤍 readFloat

public final float readFloat() throws <u>IOException</u>

Reads a 32 bit float. Returns: the read 32 bit float.

🤍 readDouble

public final double readDouble() throws <u>IOException</u>

Reads a 64 bit double. **Returns:** the 64 bit double read.

🔍 readLine

public final <u>String</u> readLine() throws <u>IOException</u>

Reads in a line that has been terminated by a $n, r, r \in OF$. **Returns:** a String copy of the line.

🧶 readUTF

public final <u>String</u> readUTF() throws <u>IOException</u>

Reads a UTF format String. **Returns:** the String.

🔍 readUTF

public final static String readUTF (DataInput in) throws IOException

Reads a UTF format String from the given input stream. **Returns:** the String.

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Interface java.io.DataOutput

public interface **DataOutput** extends <u>Object</u>

DataOutput is an interface describing streams that can write output in a machine–independent format.

See Also:

DataOutputStream, DataInput

Method Index

• write(int) Writes a byte. write(byte[]) Writes an array of bytes. write(byte[], int, int) Writes a subarray of bytes. • writeBoolean(boolean) Writes a boolean. writeByte(int) Writes an 8 bit byte. • writeBytes(String) Writes a String as a sequence of bytes. writeChar(int) Writes a 16 bit char. writeChars(String) Writes a String as a sequence of chars. • writeDouble(double) Writes a 64 bit double. writeFloat(float) Writes a 32 bit float. • writeInt(int) Writes a 32 bit int. writeLong(long) Writes a 64 bit long. writeShort(int) Writes a 16 bit short. • writeUTF(String) Writes a String in UTF format.

Methods

🤍 write

public abstract void write(int b) throws IOException

Writes a byte. Will block until the byte is actually written. **Parameters:** b – the byte to be written **Throws:**<u>IOException</u> If an I/O error has occurred.

🤍 write

public abstract void write(byte b[]) throws <u>IOException</u>

Writes an array of bytes. **Parameters:** b – the data to be written **Throws:**<u>IOException</u> If an I/O error has occurred.

🤍 write

Writes a subarray of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred.

🧶 writeBoolean

public abstract void writeBoolean(boolean v) throws <u>IOException</u>

Writes a boolean. **Parameters:** v – the boolean to be written

🧶 writeByte

public abstract void writeByte(int v) throws <u>IOException</u>

Writes an 8 bit byte. **Parameters:** v – the byte value to be written

🌻 writeShort

public abstract void writeShort(int v) throws <u>IOException</u>

Writes a 16 bit short. **Parameters:** v – the short value to be written

🧶 writeChar

public abstract void writeChar(int v) throws <u>IOException</u>

Writes a 16 bit char. **Parameters:** v – the char value to be written

🤍 writeInt

public abstract void writeInt(int v) throws <u>IOException</u>

Writes a 32 bit int. **Parameters:** v – the integer value to be written

🤍 writeLong

public abstract void writeLong(long v) throws <u>IOException</u>

Writes a 64 bit long. **Parameters:** v – the long value to be written

🤍 writeFloat

public abstract void writeFloat(float v) throws <u>IOException</u>

Writes a 32 bit float. **Parameters:** v – the float value to be written

🔍 writeDouble

public abstract void writeDouble(double v) throws <u>IOException</u>

Writes a 64 bit double.

Parameters:

v – the double value to be written

🔍 writeBytes

public abstract void writeBytes(String s) throws IOException

Writes a String as a sequence of bytes. **Parameters:** s – the String of bytes to be written

🤍 writeChars

public abstract void writeChars(<u>String</u> s) throws <u>IOException</u>

Writes a String as a sequence of chars. **Parameters:** s – the String of chars to be written

🔍 writeUTF

public abstract void writeUTF(<u>String str</u>) throws <u>IOException</u>

Writes a String in UTF format. **Parameters:** str – the String in UTF format

All Packages Class Hierarchy This Package Previous Next Index

<u>All Packages Class Hierarchy This Package Previous Next Index</u>

Class java.io.DataOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----<u>java.io.FilterOutputStream</u> | +----java.io.DataOutputStream

public class **DataOutputStream** extends <u>FilterOutputStream</u> implements <u>DataOutput</u>

This class lets you write primitive Java data types to a stream in a portable way. Primitive data types are well understood types with associated operations. For example, an Integer is considered to be a good primitive data type. The data can be converted back using a DataInputStream.

Variable Index

• <u>written</u> The number of bytes written so far.

Constructor Index

•<u>DataOutputStream</u>(OutputStream) Creates a new DataOutputStream.

Method Index

<u>flush()</u>
 Flushes the stream.
 <u>size()</u>
 Returns the number of bytes written.
 <u>write(int)</u>
 Writes a byte.

write(byte[], int, int) Writes a sub array of bytes. writeBoolean(boolean) Writes a boolean. writeByte(int) Writes an 8 bit byte. • writeBytes(String) Writes a String as a sequence of bytes. writeChar(int) Writes a 16 bit char. • writeChars(String) Writes a String as a sequence of chars. • writeDouble(double) Writes a 64 bit double. writeFloat(float) Writes a 32 bit float. • writeInt(int) Writes a 32 bit int. writeLong(long) Writes a 64 bit long. writeShort(int) Writes a 16 bit short. writeUTF(String) Writes a String in UTF format.

Variables

🤍 written

protected int written

The number of bytes written so far.

Constructors

DataOutputStream

public DataOutputStream(OutputStream out)

Creates a new DataOutputStream. **Parameters:** out – the output stream

Methods

🤍 write

public synchronized void write(int b) throws <a>IOException

Writes a byte. Will block until the byte is actually written. **Parameters:** b – the byte to be written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>FilterOutputStream</u>

🤍 write

Writes a sub array of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>FilterOutputStream</u>

🤍 flush

public void flush() throws <u>IOException</u>

Flushes the stream. This will write any buffered output bytes. **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>flush</u> in class <u>FilterOutputStream</u>

🤍 writeBoolean

public final void writeBoolean(boolean v) throws <u>IOException</u>

Writes a boolean. **Parameters:** v – the boolean to be written

🤍 writeByte

public final void writeByte(int v) throws <u>IOException</u>

Writes an 8 bit byte. **Parameters:** v – the byte value to be written

🤍 writeShort

public final void writeShort(int v) throws **IOException**

Writes a 16 bit short. **Parameters:** v – the short value to be written

🤍 writeChar

public final void writeChar(int v) throws <u>IOException</u>

Writes a 16 bit char. **Parameters:** v – the char value to be written

🤍 writeInt

public final void writeInt(int v) throws <u>IOException</u>

Writes a 32 bit int. **Parameters:** v – the integer value to be written

🤍 writeLong

public final void writeLong(long v) throws <u>IOException</u>

Writes a 64 bit long. **Parameters:** v – the long value to be written

🤍 writeFloat

public final void writeFloat(float v) throws <u>IOException</u>

Writes a 32 bit float. **Parameters:** v – the float value to be written

🤍 writeDouble

public final void writeDouble(double v) throws <u>IOException</u>

Writes a 64 bit double. **Parameters:**

v – the double value to be written

🔍 writeBytes

public final void writeBytes(String s) throws IOException

Writes a String as a sequence of bytes. **Parameters:** s – the String of bytes to be written

🌻 writeChars

public final void writeChars(String s) throws IOException

Writes a String as a sequence of chars. **Parameters:** s – the String of chars to be written

🔍 writeUTF

public final void writeUTF(String str) throws IOException

Writes a String in UTF format. **Parameters:** str – the String in UTF format

🤍 size

public final int size()

Returns the number of bytes written. Returns: the number of bytes written thus far.

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Class java.io.EOFException

java.lang.Object

+----<u>java.lang.Throwable</u> | +----<u>java.lang.Exception</u> | +----<u>java.io.IOException</u> | +----java.io.EOFException

public class **EOFException** extends <u>IOException</u>

Signals that and EOF has been reached unexpectedly during input.

See Also:

IOException, DataInputStream

Constructor Index

EOFException()

Constructs an EOFException with no detail message.

EOFException(String)

Constructs an EOFException with the specified detail message.

Constructors

EOFException

public EOFException()

Constructs an EOFException with no detail message. A detail message is a String that describes this particular exception.

EOFException

public EOFException(String s)

Constructs an EOFException with the specified detail message. A detail message is a String that describes this particular exception. **Parameters:**

s – the detail message

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.File

<u>java.lang.Object</u> |

+----java.io.File

public class **File** extends <u>Object</u>

This class represents a file name of the host file system. The file name can be relative or absolute. It must use the file name conventions of the host platform.

The intention is to provide an abstraction that deals with most of the system-dependent file name features such as the separator character, root, device name, etc. Not all features are currently fully implemented.

Note that whenever a file name or path is used it is assumed that the host's file name conventions are used.

Variable Index

pathSeparator

The system dependent path separator string.

pathSeparatorChar

The system dependent path separator character.

separator

The system dependent file separator String.

separatorChar

The system dependent file separator character.

Constructor Index

<u>File</u>(String)
 Creates a File object.
 <u>File</u>(String, String)
 Creates a File object from the specified directory.
 <u>File</u>(File, String)
 Creates a File object (given a directory File object).

Method Index

canRead()

- Returns a boolean indicating whether or not a readable file exists.
- ocanWrite()
 - Returns a boolean indicating whether or not a writable file exists.
- <u>delete()</u>

Deletes the specified file.

• equals(Object)

Compares this object against the specified object.

exists()

Returns a boolean indicating whether or not a file exists.

getAbsolutePath()

Gets the absolute path of the file.

getName()

Gets the name of the file.

getParent()

Gets the name of the parent directory.

getPath()

Gets the path of the file.

hashCode()

Computes a hashcode for the file.

isAbsolute()

Returns a boolean indicating whether the file name is absolute.

isDirectory()

Returns a boolean indicating whether or not a directory file exists.

•<u>isFile</u>()

Returns a boolean indicating whether or not a normal file exists.

lastModified()

Returns the last modification time.

length()

Returns the length of the file.

•<u>list(</u>)

Lists the files in a directory.

• list(FilenameFilter)

Uses the specified filter to list files in a directory.

• <u>mkdir</u>()

Creates a directory and returns a boolean indicating the success of the creation.

mkdirs()

Creates all directories in this path.

• renameTo(File)

Renames a file and returns a boolean indicating whether or not this method was successful.

• toString()

Returns a String object representing this file's path.

Variables

🥯 separator

public final static <u>String</u> separator

The system dependent file separator String.

🧕 separatorChar

public final static char separatorChar

The system dependent file separator character.

🧕 pathSeparator

public final static <a>String pathSeparator

The system dependent path separator string.

오 pathSeparatorChar

public final static char pathSeparatorChar

The system dependent path separator character.

Constructors

🎴 File

public File(String path)

Creates a File object. **Parameters:** path – the file path **Throws:**<u>NullPointerException</u> If the file path is equal to null.

🎴 File

```
public File(<u>String</u> path,
        <u>String</u> name)
```

Creates a File object from the specified directory. **Parameters:** path – the directory path name – the file name

🥥 File

public File(File dir, String name)

> Creates a File object (given a directory File object). **Parameters:** dir – the directory name - the file name

Methods

🔍 getName

public <u>String</u> getName()

Gets the name of the file. This method does not include the directory. **Returns:** the file name.

🤍 getPath

public <u>String</u> getPath()

Gets the path of the file. **Returns:** the file path.

getAbsolutePath

public <u>String</u> getAbsolutePath()

Gets the absolute path of the file. **Returns:** the absolute file path.

🤍 getParent

public <u>String</u> getParent()

Gets the name of the parent directory. **Returns:** the parent directory, or null if one is not found.

🌻 exists

public boolean exists()

Returns a boolean indicating whether or not a file exists.

🔍 canWrite

```
public boolean canWrite()
```

Returns a boolean indicating whether or not a writable file exists.

🧶 canRead

```
public boolean canRead()
```

Returns a boolean indicating whether or not a readable file exists.

🧶 isFile

```
public boolean isFile()
```

Returns a boolean indicating whether or not a normal file exists.

🔍 isDirectory

```
public boolean isDirectory()
```

Returns a boolean indicating whether or not a directory file exists.

🧶 isAbsolute

```
public boolean isAbsolute()
```

Returns a boolean indicating whether the file name is absolute.

🔍 lastModified

```
public long lastModified()
```

Returns the last modification time. The return value should only be used to compare modification dates. It is meaningless as an absolute time.

🧶 length

```
public long length()
```

Returns the length of the file.

🌻 mkdir

```
public boolean mkdir()
```

Creates a directory and returns a boolean indicating the success of the creation.

🤍 renameTo

public boolean renameTo(<u>File</u> dest)

Renames a file and returns a boolean indicating whether or not this method was successful.

Parameters:

dest - the new file name

🔍 mkdirs

```
public boolean mkdirs()
```

Creates all directories in this path. This method returns true if all directories in this path are created.

🧶 list

public <u>String[]</u> list()

Lists the files in a directory. Works only on directories.

Returns:

an array of file names. This list will include all files in the directory except the equivalent of "." and "..".

🧶 list

```
public <u>String[] list(FilenameFilter</u> filter)
```

Uses the specified filter to list files in a directory. **Parameters:** filter – the filter used to select file names **Returns:** the filter selected files in this directory. **See Also:** <u>FilenameFilter</u>

🔍 delete

public boolean delete()

Deletes the specified file. Returns true if the file could be deleted.

hashCode

public int hashCode()

Computes a hashcode for the file. Overrides: <u>hashCode</u> in class <u>Object</u>

🔍 equals

public boolean equals(Object obj)

Compares this object against the specified object. **Parameters:** obj – the object to compare with **Returns:** true if the objects are the same; false otherwise. **Overrides:** <u>equals</u> in class <u>Object</u>

🔍 toString

```
public <u>String</u> toString()
```

Returns a String object representing this file's path. **Overrides:** <u>toString</u> in class <u>Object</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.FileDescriptor

java.lang.Object

+----java.io.FileDescriptor

public final class **FileDescriptor** extends \underline{Object}

Variable Index

• err Handle to standard error. • in Handle to standard input. • out Handle to standard output.



FileDescriptor()

Method Index

• valid() Determines whether the file descriptor object is valid.

Variables

🤍 in

public final static <a>FileDescriptor in

Handle to standard input.

🤍 out

public final static <a>FileDescriptor out

Handle to standard output.

🥥 err

public final static <u>FileDescriptor</u> err

Handle to standard error.

Constructors

FileDescriptor

public FileDescriptor()

Methods

🧶 valid

public boolean valid()

Determines whether the file descriptor object is valid.

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Class java.io.FileInputStream

java.lang.Object

+

----<u>java.io.InputStream</u> | +----java.io.FileInputStream

public class **FileInputStream** extends <u>InputStream</u>

File input stream, can be constructed from a file descriptor or a file name.

See Also: <u>FileOutputStream</u>, <u>File</u>

Constructor Index

<u>FileInputStream</u>(String)
 Creates an input file with the specified system dependent file name.

 <u>FileInputStream</u>(File)
 Creates an input file from the specified File object.

 <u>FileInputStream</u>(FileDescriptor)



available()

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

• <u>close(</u>)

Closes the input stream.

finalize()

Closes the stream when garbage is collected.

•<u>getFD(</u>)

Returns the opaque file descriptor object associated with this stream.

- •<u>read(</u>)
 - Reads a byte of data.
- read(byte[])

Reads data into an array of bytes. • <u>read</u>(byte[], int, int) Reads data into an array of bytes. • <u>skip</u>(long) Skips n bytes of input.



🥯 FileInputStream

public FileInputStream(String name) throws FileNotFoundException

Creates an input file with the specified system dependent file name. **Parameters:** name – the system dependent file name **Throws:**<u>IOException</u> If the file is not found.

FileInputStream

public FileInputStream(File file) throws FileNotFoundException

Creates an input file from the specified File object. **Parameters:** file – the file to be opened for reading **Throws:**<u>IOException</u> If the file is not found.

🥥 FileInputStream

public FileInputStream(FileDescriptor fdObj)

Methods

🤍 read

public int read() throws <u>IOException</u>

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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>read</u> in class <u>InputStream</u>

🧶 read

public int read(byte b[]) throws <u>IOException</u>

Reads data into an array of bytes. This method 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 if the end of stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

🛢 read

Reads data into an array of bytes. This method 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:<u>IOException</u>

If an I/O error has occurred.

Overrides:

read in class InputStream

🌻 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: <u>skip</u> in class <u>InputStream</u>

🜻 available

public int available() throws <u>IOException</u>

Returns the number of bytes that can be read without blocking. **Returns:** the number of available bytes, which is initially equal to the file size. **Overrides:** available in class InputStream

🧶 close

public void close() throws <u>IOException</u>

Closes the input stream. This method must be called to release any resources associated with the stream. **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>close</u> in class <u>InputStream</u>

🔍 getFD

public final <u>FileDescriptor</u> getFD() throws <u>IOException</u>

Returns the opaque file descriptor object associated with this stream. Returns: the file descriptor.

🌻 finalize

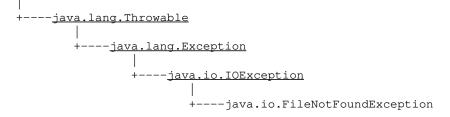
protected void finalize() throws IOException

Closes the stream when garbage is collected. **Overrides:** <u>finalize</u> in class <u>Object</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.FileNotFoundException

java.lang.Object



 $public \ class \ \textbf{FileNotFoundException} \\ extends \ \underline{IOException} \\$

Signals that a file was not found.

Constructor Index

<u>FileNotFoundException()</u>
 Constructs a FileNotFoundException with no detail message.

 <u>FileNotFoundException(String)</u>
 Constructs a FileNotFoundException with the specified detail message.



FileNotFoundException

public FileNotFoundException()

Constructs a FileNotFoundException with no detail message. A detail message is a String that describes this particular exception.

FileNotFoundException

public FileNotFoundException(String s)

Constructs a FileNotFoundException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

 ${\rm s}$ – the detail message

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.FileOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----java.io.FileOutputStream

public class **FileOutputStream** extends <u>OutputStream</u>

File output stream, can be constructed from a file descriptor or a file name.

See Also: <u>FileInputStream</u>, <u>File</u>

Constructor Index

<u>FileOutputStream</u>(String)
 Creates an output file with the specified system dependent file name.

 <u>FileOutputStream</u>(File)
 Creates an output file with the specified File object.

 <u>FileOutputStream</u>(FileDescriptor)



oclose()

Closes the stream.

• finalize()

Closes the stream when garbage is collected.

• getFD()

Returns the file descriptor associated with this stream.

• write(int)

Writes a byte of data.

write(byte[])

Writes an array of bytes.

• write(byte[], int, int)

Writes a sub array of bytes.



🥥 FileOutputStream

public FileOutputStream(String name) throws IOException

Creates an output file with the specified system dependent file name. **Parameters:** name – the system dependent file name **Throws:**<u>IOException</u> If the file is not found.

🥪 FileOutputStream

public FileOutputStream(File file) throws IOException

Creates an output file with the specified File object. **Parameters:** file – the file to be opened for reading **Throws:**<u>IOException</u> If the file is not found.

🥥 FileOutputStream

public FileOutputStream(FileDescriptor fdObj)

Methods

🧶 write

public void write(int b) throws <a>IOException

Writes a byte of data. This method will block until the byte is actually written. **Parameters:**

b – the byte to be written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** write in class <u>OutputStream</u>

```
🤍 write
```

public void write(byte b[]) throws <u>IOException</u>

Writes an array of bytes. Will block until the bytes are actually written. **Parameters:** b – the data to be written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** write in class <u>OutputStream</u>

🄍 write

Writes a sub array of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** write in class <u>OutputStream</u>

🧶 close

public void close() throws **IOException**

Closes the stream. This method must be called to release any resources associated with the stream.

Throws:<u>IOException</u>

If an I/O error has occurred.

Overrides:

close in class OutputStream

🤍 getFD

public final <a>FileDescriptor getFD() throws <a>IOException

Returns the file descriptor associated with this stream. Returns: the file descriptor.

🤍 finalize

protected void finalize() throws IOException

Closes the stream when garbage is collected.

Overrides:

<u>finalize</u> in class <u>Object</u>

All Packages Class Hierarchy This Package Previous Next Index

Interface java.io.FilenameFilter

public interface **FilenameFilter** extends <u>Object</u>

A filter interface for file names.

See Also: <u>File</u>

Method Index

• <u>accept</u>(File, String) Determines whether a name should be included in a file list.



🧶 accept

public abstract boolean accept(<u>File</u> dir, <u>String</u> name)

Determines whether a name should be included in a file list. **Parameters:** dir – the directory in which the file was found name – the name of the file **Returns:** true if name should be included in file list; false otherwise.

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.FilterInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----java.io.FilterInputStream

public class **FilterInputStream** extends <u>InputStream</u>

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.

Variable Index

• <u>in</u> The actual input stream.

Constructor Index

• <u>FilterInputStream</u>(InputStream) Creates an input stream filter.

Method Index

- available()
 - Returns the number of bytes that can be read without blocking.
- **close**()

Closes the input stream.

• <u>mark</u>(int)

Marks the current position in the input stream.

markSupported()

Returns true if this stream type supports mark/reset.

•<u>read(</u>)

Reads a byte.

 <u>read</u>(byte[]) Reads into an array of bytes.
 <u>read</u>(byte[], int, int) Reads into an array of bytes.
 <u>reset()</u> Repositions the stream to the last marked position.
 <u>skip</u>(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 <u>IOException</u>

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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>read</u> in class <u>InputStream</u>

🔍 read

public int read(byte b[]) throws <u>IOException</u>

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. Returns -1 when the end of the stream is reached.

Throws: IOException

If an I/O error has occurred.

Overrides:

read in class InputStream

🧶 read

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. Returns -1 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 <u>IOException</u>

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: <u>skip</u> in class <u>InputStream</u>

🔍 available

public int available() throws <a>IOException

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

the number of available bytes Overrides: available in class InputStream

🧶 close

public void close() throws <u>IOException</u>

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

Throws:<u>IOException</u> If an I/O error has occurred. Overrides: <u>close in class InputStream</u>

🄍 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

🎈 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. If this happens within readlimit bytes, it allows the outer code to reset the stream and try another parser.

Overrides:

reset in class InputStream

markSupported

public boolean markSupported()

Returns true if this stream type supports mark/reset. **Overrides:** <u>markSupported</u> in class <u>InputStream</u>

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Class java.io.FilterOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----java.io.FilterOutputStream

public class **FilterOutputStream** extends <u>OutputStream</u>

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

Variable Index

• <u>out</u> The actual output stream.

Constructor Index

• <u>FilterOutputStream</u>(OutputStream) Creates an output stream filter.

Method Index

<u>close</u>() Closes the stream.
<u>flush</u>() Flushes the stream.
<u>write</u>(int) Writes a byte.
<u>write</u>(byte[]) Writes an array of bytes.
<u>write</u>(byte[], int, int) Writes a subarray of bytes.

Variables

🤍 out

protected <u>OutputStream</u> out

The actual output stream.



FilterOutputStream

public FilterOutputStream(OutputStream out)

Creates an output stream filter. **Parameters:** out – the output stream



🄍 write

public void write(int b) throws IOException

Writes a byte. Will block until the byte is actually written. **Parameters:** b – the byte **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** write in class OutputStream

🧶 write

public void write(byte b[]) throws <u>IOException</u>

Writes an array of bytes. Will block until the bytes are actually written. **Parameters:** b – the data to be written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>OutputStream</u>

🧶 write

Writes a subarray of bytes. To be efficient it should be overridden in a subclass. **Parameters:**

b - the data to be written
off - the start offset in the data
len - the number of bytes that are written
Throws:<u>IOException</u>
If an I/O error has occurred.
Overrides:
write in class <u>OutputStream</u>

🌻 flush

public void flush() throws <a>IOException

Flushes the stream. This will write any buffered output bytes. **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>flush</u> in class <u>OutputStream</u>

🧶 close

public void close() throws <u>IOException</u>

Closes the stream. This method must be called to release any resources associated with the stream.

Throws:<u>IOException</u> If an I/O error has occurred. Overrides: <u>close in class OutputStream</u>

<u>All Packages Class Hierarchy This Package Previous Next Index</u>

Class java.io.IOException

java.lang.Object

+----<u>java.lang.Throwable</u> | +----<u>java.lang.Exception</u> | +----java.io.IOException

public class **IOException** extends <u>Exception</u>

Signals that an I/O exception has occurred.

See Also:

InputStream, OutputStream

Constructor Index

<u>IOException()</u>
 Constructs an IOException with no detail message.
 <u>IOException(String)</u>
 Constructs an IOException with the specified detail message.

Constructors

IOException

public IOException()

Constructs an IOException with no detail message. A detail message is a String that describes this particular exception.

IOException

```
public IOException(String s)
```

Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s – the detail message

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.InputStream

java.lang.Object

+----java.io.InputStream

public class **InputStream** extends <u>Object</u>

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

See Also:

<u>OutputStream</u>, <u>FilterInputStream</u>, <u>BufferedInputStream</u>, <u>DataInputStream</u>, <u>ByteArrayInputStream</u>, <u>PushbackInputStream</u>

Constructor Index

InputStream()

Method Index

available()

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

• <u>close(</u>)

Closes the input stream.

• **mark**(int)

Marks the current position in the input stream.

markSupported()

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

•<u>read(</u>)

Reads a byte of data.

• <u>read</u>(byte[])

Reads into an array of bytes.

<u>read(byte[]</u>, int, int)

Reads into an array of bytes.

• **reset**()

Repositions the stream to the last marked position. • <u>skip</u>(long) Skips n bytes of input.

Constructors

🥥 InputStream

public InputStream()



🄍 read

public abstract int read() throws <u>IOException</u>

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:**<u>IOException</u> If an I/O error has occurred.

🧶 read

public int read(byte b[]) throws <u>IOException</u>

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:<u>IOException</u> If an I/O error has occurred.

🧶 read

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:<u>IOException</u> If an I/O error has occurred.

🔍 available

public int available() throws <a>IOException

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

the number of available bytes.

🧶 close

public void close() throws <u>IOException</u>

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

Throws:<u>IOException</u> 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.

🧶 reset

public synchronized void reset() throws <a>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.

markSupported

public boolean markSupported()

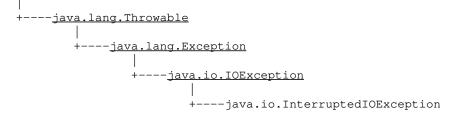
Returns a boolean indicating whether or not this stream type supports mark/reset. **Returns:**

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

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.InterruptedIOException

java.lang.Object



 $\begin{array}{l} \mbox{public class } {\bf Interrupted IOException} \\ \mbox{extends } \underline{\rm IOException} \end{array} \end{array} \\$

Signals that an I/O operation has been interrupted.

See Also:

InputStream, OutputStream

Variable Index

<u>bytesTransferred</u>

Reports how many bytes had been transferred as part of the IO operation before it was interrupted.

Constructor Index

<u>InterruptedIOException()</u>
 Constructs an IOException with no detail message.
 <u>InterruptedIOException(String)</u>
 Constructs an IOException with the specified detail message.

Variables

🥯 bytesTransferred

public int bytesTransferred

Reports how many bytes had been transferred as part of the IO operation before it was interrupted.

Constructors

InterruptedIOException

public InterruptedIOException()

Constructs an IOException with no detail message. A detail message is a String that describes this particular exception.

InterruptedIOException

public InterruptedIOException(String s)

Constructs an IOException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

s – the detail message

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.LineNumberInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----<u>java.io.FilterInputStream</u> | +----java.io.LineNumberInputStream

 $public \ class \ \textbf{LineNumberInputStream} \\ extends \ \underline{FilterInputStream} \\$

An input stream that keeps track of line numbers.

Constructor Index

LineNumberInputStream(InputStream)

Constructs a new LineNumberInputStream initialized with the specified input stream.

Method Index

available()

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

getLineNumber()

Returns the current line number.

• **mark**(int)

Marks the current position in the input stream.

• <u>read(</u>)

Reads a byte of data.

• **read**(byte[], int, int)

Reads into an array of bytes.

• <u>reset(</u>)

Repositions the stream to the last marked position.

• setLineNumber(int)

Sets the current line number.

skip(long)

Skips n bytes of input.

Constructors

单 LineNumberInputStream

public LineNumberInputStream(InputStream in)

Constructs a new LineNumberInputStream initialized with the specified input stream.

Parameters:

in - the input stream



🔍 read

public int read() throws <u>IOException</u>

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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>read</u> in class <u>FilterInputStream</u>

🧶 read

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

🧶 setLineNumber

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

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

🧶 getLineNumber

```
public int getLineNumber()
```

Returns the current line number.

🧶 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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>skip</u> in class <u>FilterInputStream</u>

🧶 available

public int available() throws <u>IOException</u>

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

Returns:

the number of available bytes Overrides: available in class FilterInputStream

🛢 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

🧶 reset

public void reset() throws <u>IOException</u>

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

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.OutputStream

<u>java.lang.Object</u>

+----java.io.OutputStream

public class **OutputStream** extends <u>Object</u>

Abstract class representing an output stream of bytes. All OutputStreams are based on this class.

See Also:

<u>InputStream, FilterOutputStream, BufferedOutputStream, DataOutputStream, ByteArrayOutputStream</u>

Constructor Index

OutputStream()

Method Index

<u>close</u>() Closes the stream.
<u>flush</u>() Flushes the stream.
<u>write</u>(int) Writes a byte.
<u>write</u>(byte[]) Writes an array of bytes.
<u>write</u>(byte[], int, int) Writes a sub array of bytes.

Constructors

OutputStream

public OutputStream()

Methods

🄍 write

public abstract void write(int b) throws <a>IOException

Writes a byte. This method will block until the byte is actually written. **Parameters:** b – the byte **Throws:**<u>IOException</u> If an I/O error has occurred.

🔍 write

public void write(byte b[]) throws <u>IOException</u>

Writes an array of bytes. This method will block until the bytes are actually written.

Parameters: b – the data to be written **Throws:**<u>IOException</u> If an I/O error has occurred.

🤍 write

Writes a sub array of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred.

🧶 flush

public void flush() throws <u>IOException</u>

Flushes the stream. This will write any buffered output bytes. **Throws:**<u>IOException</u> If an I/O error has occurred.

🤍 close

public void close() throws <u>IOException</u>

Closes the stream. This method must be called to release any resources associated with the stream.
Throws:IOException
If an I/O error has occurred.

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.PipedInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----java.io.PipedInputStream

public class **PipedInputStream** extends <u>InputStream</u>

PipedInputStream must be connected to a PipedOutputStream to be useful. A thread reading from a PipedInputStream recieves data from a thread writing to the PipedOutputStream it is connected to.

See Also: <u>PipedOutputStream</u>

Constructor Index

• **<u>PipedInputStream</u>**(PipedOutputStream)

Creates an input file from the specified PiledOutputStream.

<u>PipedInputStream()</u>

Creates an input file that isn't connected to anything (yet).

Method Index

• <u>close(</u>)

Closes the input stream.

• connect(PipedOutputStream)

Connects this input stream to a sender.

•<u>read()</u>

Reads a byte of data.

• read(byte[], int, int)

Reads into an array of bytes.

Constructors

PipedInputStream

public PipedInputStream (PipedOutputStream src) throws IOException

Creates an input file from the specified PiledOutputStream. **Parameters:** src – the stream to connect to.

PipedInputStream

public PipedInputStream()

Creates an input file that isn't connected to anything (yet). It must be connected to a PipedOutputStream before being used.

Methods

🤍 connect

public void connect (PipedOutputStream src) throws IOException

Connects this input stream to a sender. **Parameters:** src – The OutputStream to connect to.

🔍 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:**<u>IOException</u> If the pipe is broken. **Overrides:** <u>read</u> in class <u>InputStream</u>

🌻 read

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:<u>IOException</u> If an I/O error has occurred.

Overrides:

read in class InputStream

🤍 close

public void close() throws <u>IOException</u>

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:
<u>close</u> in class <u>InputStream</u>

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.PipedOutputStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----java.io.PipedOutputStream

public class **PipedOutputStream** extends <u>OutputStream</u>

Piped output stream, must be connected to a PipedInputStream. A thread reading from a PipedInputStream receives data from a thread writing to the PipedOutputStream it is connected to.

See Also: <u>PipedInputStream</u>

Constructor Index

<u>PipedOutputStream</u>(PipedInputStream)

Creates an output file connected to the specified PipedInputStream.

<u>PipedOutputStream()</u>

Creates an output file that isn't connected to anything (yet).

Method Index

<u>close</u>()
 Closes the stream.
 <u>connect</u>(PipedInputStream)
 Connect this output stream to a receiver.
 <u>write</u>(int)
 Write a byte.
 <u>write</u>(byte[], int, int)
 Writes a sub array of bytes.

Constructors

PipedOutputStream

public PipedOutputStream(PipedInputStream snk) throws IOException

Creates an output file connected to the specified PipedInputStream. **Parameters:** snk – The InputStream to connect to.

<mark>ə</mark> PipedOutputStream

public PipedOutputStream()

Creates an output file that isn't connected to anything (yet). It must be connected before being used.

Methods

🤍 connect

public void connect(PipedInputStream snk) throws IOException

Connect this output stream to a receiver. **Parameters:** snk – The InputStream to connect to.

🔍 write

```
public void write(int b) throws <a>IOException</a>
```

Write a byte. This method will block until the byte is actually written. **Parameters:** b – the byte to be written **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>OutputStream</u>

🧶 write

Writes a sub array of bytes.

Parameters:

b - the data to be written
 off - the start offset in the data
 len - the number of bytes that are written
 Throws:IOException
 If an I/O error has occurred.
 Overrides:
 write in class OutputStream

🧶 close

public void close() throws **IOException**

Closes the stream. This method must be called to release any resources associated with the stream. **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>close</u> in class <u>OutputStream</u>

All Packages Class Hierarchy This Package Previous Next Index

<u>All Packages Class Hierarchy This Package Previous Next Index</u>

Class java.io.PrintStream

java.lang.Object

+----<u>java.io.OutputStream</u> | +----<u>java.io.FilterOutputStream</u> | +----java.io.PrintStream

public class **PrintStream** extends <u>FilterOutputStream</u>

This class implements an output stream that has additional methods for printing. You can specify that the stream should be flushed every time a newline character is written.

The top byte of 16 bit characters is discarded.

Example:

```
System.out.println("Hello world!");
System.out.print("x = ");
System.out.println(x);
System.out.println("y = " + y);
```

Constructor Index

 <u>PrintStream</u>(OutputStream) Creates a new PrintStream.
 <u>PrintStream</u>(OutputStream, boolean) Creates a new PrintStream, with auto flushing.

Method Index

<u>checkError()</u>

Flushes the print stream and returns whether or not there was an error on the output stream.

• **close**()

Closes the stream. flush() Flushes the stream. **print**(Object) Prints an object. print(String) Prints a String. **print**(char[]) Prints an array of characters. print(char) Prints an character. **print**(int) Prints an integer. print(long) Prints a long. print(float) Prints a float. print(double) Prints a double. **print**(boolean) Prints a boolean. **println**() Prints a newline. **println**(Object) Prints an object followed by a newline. **println**(String) Prints a string followed by a newline. println(char[]) Prints an array of characters followed by a newline. **println**(char) Prints a character followed by a newline. **println**(int) Prints an integer followed by a newline. **println**(long) Prints a long followed by a newline. println(float) Prints a float followed by a newline. • **println**(double) Prints a double followed by a newline. println(boolean) Prints a boolean followed by a newline. • write(int) Writes a byte. • write(byte[], int, int)

Writes a sub array of bytes.

Constructors

🥪 PrintStream

public PrintStream(OutputStream out)

Creates a new PrintStream. **Parameters:** out – the output stream

🥥 PrintStream

Creates a new PrintStream, with auto flushing. **Parameters:** out – the output stream autoflush – if true the stream automatically flushes its output when a newline character is printed



🧶 write

public void write(int b)

Writes a byte. This method will block until the byte is actually written. **Parameters:** b – the byte **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:** <u>write</u> in class <u>FilterOutputStream</u>

🤍 write

Writes a sub array of bytes.

Parameters:

b – the data to be writtenoff – the start offset in the datalen – the number of bytes that are written

```
Throws:<u>IOException</u>
If an I/O error has occurred.
Overrides:
write in class <u>FilterOutputStream</u>
```

🌻 flush

public void flush()

Flushes the stream. This will write any buffered output bytes. Overrides: <u>flush</u> in class <u>FilterOutputStream</u>

🧶 close

public void close()

Closes the stream. **Overrides:** <u>close</u> in class <u>FilterOutputStream</u>

🤍 checkError

```
public boolean checkError()
```

Flushes the print stream and returns whether or not there was an error on the output stream. Errors are cumulative; once the print stream encounters an error this routine will continue to return true on all successive calls.

Returns:

true if the print stream has ever encountered an error on the output stream.

🧶 print

public void print(Object obj)

Prints an object. **Parameters:** obj – the object to be printed

🧶 print

public synchronized void print(String s)

Prints a String. **Parameters:** s – the String to be printed

🧶 print

```
public synchronized void print(char s[])
```

Prints an array of characters. **Parameters:** s – the array of chars to be printed

🧶 print

```
public void print(char c)
```

Prints an character. **Parameters:** c – the character to be printed

🧶 print

```
public void print(int i)
```

Prints an integer. **Parameters:** i – the integer to be printed

🧶 print

```
public void print(long l)
```

Prints a long. **Parameters:** l – the long to be printed.

🔍 print

```
public void print(float f)
```

Prints a float. **Parameters:** f – the float to be printed

🔍 print

```
public void print(double d)
```

Prints a double. **Parameters:** d – the double to be printed

🤍 print

```
public void print (boolean b)
```

Prints a boolean. **Parameters:** b – the boolean to be printed

🤍 println

```
public void println()
```

Prints a newline.

🤍 println

public synchronized void println(Object obj)

Prints an object followed by a newline. **Parameters:** obj – the object to be printed

🔍 println

public synchronized void println(String s)

Prints a string followed by a newline. **Parameters:** s – the String to be printed

🤍 println

```
public synchronized void println(char s[])
```

Prints an array of characters followed by a newline. **Parameters:** s – the array of characters to be printed

🌻 println

```
public synchronized void println(char c)
```

Prints a character followed by a newline. **Parameters:** c – the character to be printed

🤍 println

```
public synchronized void println(int i)
```

Prints an integer followed by a newline.

Parameters:

i – the integer to be printed

🜻 println

public synchronized void println(long 1)

Prints a long followed by a newline. **Parameters:** 1 – the long to be printed

🜻 println

public synchronized void println(float f)

Prints a float followed by a newline. **Parameters:** f – the float to be printed

🔍 println

public synchronized void println(double d)

Prints a double followed by a newline. **Parameters:** d – the double to be printed

🤍 println

public synchronized void println(boolean b)

Prints a boolean followed by a newline. **Parameters:** b – the boolean to be printed

All Packages Class Hierarchy This Package Previous Next Index

<u>All Packages Class Hierarchy This Package Previous Next Index</u>

Class java.io.PushbackInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----<u>java.io.FilterInputStream</u> | +----java.io.PushbackInputStream

public class **PushbackInputStream** extends <u>FilterInputStream</u>

An input stream that has a 1 byte push back buffer.

Variable Index

• <u>pushBack</u> Push back character.

Constructor Index

• <u>PushbackInputStream</u>(InputStream) Creates a PushbackInputStream.

Method Index

<u>available()</u>
 Returns the number of bytes that can be read.
 <u>markSupported()</u>
 Returns true if this stream type supports mark/reset.
 <u>read()</u>
 Reads a byte of data.
 <u>read(byte[], int, int)</u>
 Reads into an array of bytes.
 <u>unread(int)</u>

Pushes back a character.

Variables

单 pushBack

protected int pushBack

Push back character.



PushbackInputStream

public PushbackInputStream(InputStream in)

Creates a PushbackInputStream. **Parameters:** in – the input stream

Methods

🤍 read

public int read() throws <u>IOException</u>

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:**<u>IOException</u> If an I/O error has occurred. **Overrides:** read in class FilterInputStream

🧶 read

Reads into an array of bytes. This method 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

🤍 unread

public void unread(int ch) throws **IOException**

Pushes back a character. **Parameters:** ch – the character to push back. **Throws:**<u>IOException</u> If an attempt to push back more than one character is made.

🌻 available

public int available() throws <u>IOException</u>

Returns the number of bytes that can be read. without blocking. Overrides: <u>available</u> in class <u>FilterInputStream</u>

markSupported

public boolean markSupported()

Returns true if this stream type supports mark/reset. **Overrides:** <u>markSupported</u> in class <u>FilterInputStream</u>

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

<u>All Packages Class Hierarchy This Package Previous Next Index</u>

Class java.io.RandomAccessFile

java.lang.Object

+----java.io.RandomAccessFile

public class RandomAccessFile extends Object implements DataOutput, DataInput

Random access files can be constructed from file descriptors, file names, or file objects. This class provides a sense of security by offering methods that allow specified mode accesses of read-only or read-write to files.

Constructor Index

RandomAccessFile(String, String)

Creates a RandomAccessFile with the specified system dependent file name and the specified mode.

RandomAccessFile(File, String) Creates a RandomAccessFile from a specified File object and mode ("r" or "rw").

Method Index

• close()

Closes the file.

getFD()

Returns the opaque file descriptor object.

getFilePointer()

Returns the current location of the file pointer.

length()

Returns the length of the file.

• **read**()

Reads a byte of data.

• read(byte[], int, int)

Reads a sub array as a sequence of bytes.

• read(byte[])

Reads data into an array of bytes.

• readBoolean() Reads a boolean. readByte() Reads a byte. readChar() Reads a 16 bit char. readDouble() Reads a 64 bit double. readFloat() Reads a 32 bit float. • readFully(byte[]) Reads bytes, blocking until all bytes are read. • readFully(byte[], int, int) Reads bytes, blocking until all bytes are read. readInt() Reads a 32 bit int. readLine() Reads a line terminated by a n or EOF. readLong() Reads a 64 bit long. • readShort() Reads 16 bit short. readUTF() Reads a UTF formatted String. • readUnsignedByte() Reads an unsigned 8 bit byte. • readUnsignedShort() Reads 16 bit short. seek(long) Sets the file pointer to the specified absolute position. • **skipBytes**(int) • write(int) Writes a byte of data. write(byte[]) Writes an array of bytes. • write(byte[], int, int) Wrotes a sub array of bytes. writeBoolean(boolean) Writes a boolean. writeByte(int) Writes a byte. writeBytes(String) Writes a String as a sequence of bytes. writeChar(int) Writes a character. • writeChars(String) Writes a String as a sequence of chars. • writeDouble(double) writeFloat(float)

writeInt(int)
 Writes an integer.
 writeLong(long)
 Writes a long.
 writeShort(int)
 Writes a short.
 writeUTF(String)
 Writes a String in UTF format.



RandomAccessFile

Creates a RandomAccessFile with the specified system dependent file name and the specified mode. Mode "r" is for read-only and mode "rw" is for read+write. **Parameters:**

name – the system dependent file name mode – the access mode **Throws:**<u>IOException</u> If an I/O error has occurred.

RandomAccessFile

public RandomAccessFile(<u>File</u> file, <u>String</u> mode) throws <u>IOException</u>

Creates a RandomAccessFile from a specified File object and mode ("r" or "rw"). **Parameters:**

file – the file object mode – the access mode

Methods

🧕 getFD

public final <a>FileDescriptor getFD() throws <a>IOException

Returns the opaque file descriptor object. Returns: the file descriptor.

🧶 read

public int read() throws <u>IOException</u>

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[],
                int off,
                int len) throws IOException
```

Reads a sub array as a sequence of bytes. **Parameters:** b – the data to be written off - the start offset in the data len – the number of bytes that are written **Throws:**IOException If an I/O error has occurred.

🔍 read

public int read(byte b[]) throws <u>IOException</u>

Reads data into an array of bytes. This method blocks until some input is available.

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.

🔍 readFully

public final void readFully(byte b[]) throws <u>IOException</u>

Reads bytes, blocking until all bytes are read. **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.

```
🧶 readFully
```

🧶 skipBytes

public int skipBytes(int n) throws <u>IOException</u>

🌻 write

public void write(int b) throws IOException

Writes a byte of data. This method will block until the byte is actually written. **Parameters:** b – the byte to be written **Throws:**<u>IOException</u> If an I/O error has occurred.

🧶 write

public void write(byte b[]) throws <u>IOException</u>

Writes an array of bytes. Will block until the bytes are actually written. **Parameters:** b – the data to be written **Throws:**<u>IOException</u> If an I/O error has occurred.

🧶 write

Wrotes a sub array of bytes. **Parameters:** b – the data to be written off – the start offset in the data len – the number of bytes that are written **Throws:**<u>IOException</u> If an I/O error has occurred.

🧶 getFilePointer

public long getFilePointer() throws <u>IOException</u>

Returns the current location of the file pointer.

🔍 seek

public void seek(long pos) throws <a>IOException

Sets the file pointer to the specified absolute position. **Parameters:** pos – the absolute position

🤍 length

public long length() throws <u>IOException</u>

Returns the length of the file.

🤍 close

public void close() throws <u>IOException</u>

Closes the file. **Throws:**<u>IOException</u> If an I/O error has occurred.

🤍 readBoolean

public final boolean readBoolean() throws <u>IOException</u>

Reads a boolean.

🤍 readByte

public final byte readByte() throws <u>IOException</u>

Reads a byte.

readUnsignedByte

public final int readUnsignedByte() throws <u>IOException</u>

Reads an unsigned 8 bit byte.

Returns:

the 8 bit byte read.

🤍 readShort

public final short readShort() throws <u>IOException</u>

Reads 16 bit short. **Returns:** the read 16 bit short.

readUnsignedShort

public final int readUnsignedShort() throws <u>IOException</u>

Reads 16 bit short. **Returns:** the read 16 bit short.

🔍 readChar

public final char readChar() throws <u>IOException</u>

Reads a 16 bit char. **Returns:** the read 16 bit char.

🔍 readInt

public final int readInt() throws <u>IOException</u>

Reads a 32 bit int. **Returns:** the read 32 bit integer.

🜻 readLong

public final long readLong() throws <u>IOException</u>

Reads a 64 bit long. Returns: the read 64 bit long.

🤍 readFloat

public final float readFloat() throws <u>IOException</u>

Reads a 32 bit float. **Returns:** the read 32 bit float.

🜻 readDouble

public final double readDouble() throws <u>IOException</u>

Reads a 64 bit double. **Returns:** the read 64 bit double.

🔍 readLine

public final <u>String</u> readLine() throws <u>IOException</u>

Reads a line terminated by a '\n' or EOF.

🧶 readUTF

public final <u>String</u> readUTF() throws <u>IOException</u>

Reads a UTF formatted String.

🌻 writeBoolean

public final void writeBoolean (boolean v) throws **IOException**

Writes a boolean. **Parameters:** v – the boolean value

🧶 writeByte

public final void writeByte(int v) throws <u>IOException</u>

Writes a byte. **Parameters:** v – the byte

🤍 writeShort

public final void writeShort(int v) throws <u>IOException</u>

Writes a short. **Parameters:** v – the short

🤍 writeChar

public final void writeChar(int v) throws <u>IOException</u>

Writes a character. Parameters: v – the char

🤍 writeInt

public final void writeInt(int v) throws <u>IOException</u>

Writes an integer. Parameters: v – the integer

🔍 writeLong

public final void writeLong(long v) throws IOException

Writes a long. **Parameters:** v – the long

🤍 writeFloat

public final void writeFloat(float v) throws <u>IOException</u>

🔍 writeDouble

public final void writeDouble(double v) throws **IOException**

🧶 writeBytes

public final void writeBytes (String s) throws IOException

Writes a String as a sequence of bytes. **Parameters:** s – the String

🤍 writeChars

public final void writeChars(String s) throws IOException

Writes a String as a sequence of chars. **Parameters:** s – the String

🤍 writeUTF

public final void writeUTF (String str) throws IOException

Writes a String in UTF format. **Parameters:** str – the String

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.SequenceInputStream

java.lang.Object

----<u>java.io.InputStream</u> | +----java.io.SequenceInputStream

public class **SequenceInputStream** extends <u>InputStream</u>

Converts a sequence of input streams into an InputStream.

Constructor Index

<u>SequenceInputStream</u>(Enumeration)

Constructs a new SequenceInputStream initialized to the specified list.

<u>SequenceInputStream</u>(InputStream, InputStream)

Constructs a new SequenceInputStream initialized to the two specified input streams.

Method Index

oclose()

Closes the input stream; flipping to the next stream, if an EOF is reached. • read()

Reads a stream, and upon reaching an EOF, flips to the next stream.

<u>read(byte[]</u>, int, int)

Reads data into an array of bytes, and upon reaching an EOF, flips to the next stream.

Constructors

🤜 SequenceInputStream

```
public SequenceInputStream(Enumeration e)
```

Constructs a new SequenceInputStream initialized to the specified list. **Parameters:**

e – the list

🥯 SequenceInputStream

Constructs a new SequenceInputStream initialized to the two specified input streams.

Parameters:

s1 – the first input stream

s2 – the second input stream

Methods

🧶 read

```
public int read() throws <u>IOException</u>
```

Reads a stream, and upon reaching an EOF, flips to the next stream. **Overrides:**

read in class InputStream

🧶 read

Reads data into an array of bytes, and upon reaching an EOF, flips to the next stream.

Parameters:

buf – the buffer into which the data is read pos – the start position of the data len – the maximum number of bytes read **Throws:**<u>IOException</u> If an I/O error has occurred. **Overrides:**

read in class InputStream

🤍 close

```
public void close() throws <u>IOException</u>
```

Closes the input stream; flipping to the next stream, if an EOF is reached. This

method must be called to release any resources associated with the stream. Throws: $\underline{IOException}$

If an I/O error has occurred.

Overrides:

<u>close</u> in class <u>InputStream</u>

<u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Class java.io.StreamTokenizer

java.lang.Object

+----java.io.StreamTokenizer

public class **StreamTokenizer** extends <u>Object</u>

A class to turn an input stream into a stream of tokens. There are a number of methods that define the lexical syntax of tokens.

Variable Index

• TT EOF The End-of-file token. • TT EOL The End-of-line token. • TT NUMBER The number token. • TT WORD The word token. • nval The number value. • sval The Stream value.

The type of the last token returned.

Constructor Index

<u>StreamTokenizer</u>(InputStream)

Creates a stream tokenizer that parses the specified input stream.

Method Index

<u>commentChar(int)</u>

Specifies that this character starts a single line comment.

eolIsSignificant(boolean)

If the flag is true, end-of-lines are significant (TT_EOL will be returned by nexttoken).

lineno()

Return the current line number.

lowerCaseMode(boolean)

Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

nextToken()

Parses a token from the input stream.

ordinaryChar(int)

Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character.

• ordinaryChars(int, int)

Specifies that characters in this range are 'ordinary'.

parseNumbers()

Specifies that numbers should be parsed.

pushBack()

Pushes back a stream token.

quoteChar(int)

Specifies that matching pairs of this character delimit String constants.

resetSyntax()

Resets the syntax table so that all characters are special.

slashSlashComments(boolean)

If the flag is true, recognize C++ style(//) comments.

slashStarComments(boolean)

If the flag is true, recognize C style(/*) comments.

toString()

Returns the String representation of the stream token.

• whitespaceChars(int, int)

Specifies that characters in this range are whitespace characters.

• wordChars(int, int)

Specifies that characters in this range are word characters.

Variables

🔍 ttype

public int ttype

The type of the last token returned. It's value will either be one of the following TT_* constants, or a single character. For example, if '+' is encountered and is not

a valid word character, ttype will be '+'

🔍 TT_EOF

public final static int TT_EOF

The End-of-file token.

🔵 TT_EOL

public final static int TT_EOL

The End-of-line token.

🔍 TT_NUMBER

public final static int TT_NUMBER

The number token. This value is in nval.

TT_WORD

public final static int TT_WORD

The word token. This value is in sval.

🤍 sval

public <u>String</u> sval

The Stream value.

🏓 nval

public double nval

The number value.



🥯 StreamTokenizer

public StreamTokenizer(InputStream I)

Creates a stream tokenizer that parses the specified input stream. By default, it recognizes numbers, Strings quoted with single and double quotes, and all the alphabetics.

Parameters:

I – the input stream

🔍 resetSyntax

```
public void resetSyntax()
```

Resets the syntax table so that all characters are special.

🔍 wordChars

Specifies that characters in this range are word characters.

Parameters:

low – the low end of the range hi – the high end of the range

🤍 whitespaceChars

Specifies that characters in this range are whitespace characters.

Parameters:

low – the low end of the range hi – the high end of the range

🧶 ordinaryChars

Specifies that characters in this range are 'ordinary'. Ordinary characters mean that any significance as words, comments, strings, whitespaces or number characters are removed. When these characters are encountered by the parser, they return a ttype equal to the character.

Parameters:

low – the low end of the range hi – the high end of the range

🤍 ordinaryChar

```
public void ordinaryChar(int ch)
```

Specifies that this character is 'ordinary': it removes any significance as a word, comment, string, whitespace or number character. When encountered by the parser, it returns a ttype equal to the character.

Parameters:

ch - the character

🤍 commentChar

```
public void commentChar(int ch)
```

Specifies that this character starts a single line comment. **Parameters:**

ch – the character

🎈 quoteChar

```
public void quoteChar(int ch)
```

Specifies that matching pairs of this character delimit String constants. When a String constant is recognized, ttype will be the character that delimits the String, and sval will have the body of the String.

Parameters:

ch - the character

parseNumbers

```
public void parseNumbers()
```

Specifies that numbers should be parsed. This method accepts double precision floating point numbers and returns a ttype of TT_NUMBER with the value in nval.

🜻 eolIsSignificant

```
public void eolIsSignificant(boolean flag)
```

If the flag is true, end-of-lines are significant (TT_EOL will be returned by nexttoken). If false, they will be treated as whitespace.

🔍 slashStarComments

public void slashStarComments(boolean flag)

If the flag is true, recognize C style(/*) comments.

🧶 slashSlashComments

```
public void slashSlashComments(boolean flag)
```

If the flag is true, recognize C++ style(//) comments.

IowerCaseMode

```
public void lowerCaseMode(boolean fl)
```

Examines a boolean to decide whether TT_WORD tokens are forced to be lower case.

Parameters: fl – the boolean flag

🤍 nextToken

public int nextToken() throws <u>IOException</u>

Parses a token from the input stream. The return value is the same as the value of ttype. Typical clients of this class first set up the syntax tables and then sit in a loop calling nextToken to parse successive tokens until TT_EOF is returned.

🔍 pushBack

```
public void pushBack()
```

Pushes back a stream token.

🔍 lineno

```
public int lineno()
```

Return the current line number.

🔍 toString

public <u>String</u> toString()

Returns the String representation of the stream token. Overrides: <u>toString</u> in class <u>Object</u>

All Packages Class Hierarchy This Package Previous Next Index

Class java.io.StringBufferInputStream

java.lang.Object

+----<u>java.io.InputStream</u> | +----java.io.StringBufferInputStream

public class **StringBufferInputStream** extends <u>InputStream</u>

This class implements a String buffer that can be used as an InputStream.

Variable Index

<u>buffer</u>

The buffer where data is stored.

<u>• count</u>

The number of characters to use in the buffer.

<u>pos</u>

The position in the buffer.

Constructor Index

<u>StringBufferInputStream</u>(String)

Creates an StringBufferInputStream from the specified array of bytes.

Method Index

available()

Returns the number of available bytes in the buffer.

•<u>read(</u>)

Reads a byte of data.

• **read**(byte[], int, int)

Reads into an array of bytes.

• <u>reset(</u>)

Resets the buffer to the beginning.

• <u>skip</u>(long) Skips n bytes of input.



🤍 buffer

protected <u>String</u> buffer

The buffer where data is stored.



protected int pos

The position in the buffer.

🤍 count

protected int count

The number of characters to use in the buffer.

Constructors

StringBufferInputStream

public StringBufferInputStream(String s)

Creates an StringBufferInputStream from the specified array of bytes. **Parameters:** s – the input buffer (not copied)

Methods

🤍 read

public synchronized int read()

Reads a byte of data. **Returns:** the byte read, or -1 if the end of the stream is reached. **Overrides:**

read in class InputStream

🤍 read

Reads into an array of bytes.

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.

Overrides:

read in class InputStream

🎈 skip

public synchronized long skip(long n)

Skips n bytes of input. Parameters: n – the number of bytes to be skipped Returns: the actual number of bytes skipped. Overrides: <u>skip</u> in class <u>InputStream</u>

🤍 available

public synchronized int available()

Returns the number of available bytes in the buffer. **Overrides:** <u>available</u> in class <u>InputStream</u>

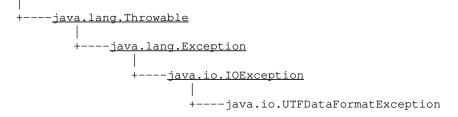
🔍 reset

public synchronized void reset()

Resets the buffer to the beginning. Overrides: reset in class InputStream <u>All Packages</u> <u>Class Hierarchy</u> <u>This Package</u> <u>Previous</u> <u>Next</u> <u>Index</u>

Class java.io.UTFDataFormatException

java.lang.Object



public class **UTFDataFormatException** extends <u>IOException</u>

Signals that a malformed UTF-8 string has been read in a DataInput stream.

See Also:

IOException, **DataInput**

Constructor Index

UTFDataFormatException()

Constructs an UTFDataFormatException with no detail message.

<u>UTFDataFormatException</u>(String)

Constructs an UTFDataFormatException with the specified detail message.

Constructors

UTFDataFormatException

public UTFDataFormatException()

Constructs an UTFDataFormatException with no detail message. A detail message is a String that describes this particular exception.

UTFDataFormatException

public UTFDataFormatException(String s)

Constructs an UTFDataFormatException with the specified detail message. A detail message is a String that describes this particular exception. **Parameters:**

s – the detail message

All Packages Class Hierarchy This Package Previous Next Index