java.io

Class Reader

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

Direct Known Subclasses:

BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

---

public abstract class **Reader**
extends Object

Abstract class for reading character streams. The only methods that a subclass must implement are read(char [], int, int) and close(). Most subclasses, however, will override some of the methods defined here in order to provide higher efficiency, additional functionality, or both.

Since:

JDK1.1

See Also:

BufferedReader, LineNumberReader, CharArrayReader, InputStreamReader, FileReader, FilterReader, PushbackReader, PipedReader, StringReader, Writer

---

Field Summary

protected Object	**lock** The object used to synchronize operations on this stream.

Constructor Summary

protected	**Reader**() Create a new character-stream reader whose critical sections will synchronize on the reader itself.
protected	**Reader**(Object lock) Create a new character-stream reader whose critical sections will synchronize on the given object.

Method Summary

abstract void	**close**() Close the stream.
void	**mark**(int readAheadLimit) Mark the present position in the stream.
boolean	**markSupported**() Tell whether this stream supports the mark() operation.

int	**read**() Read a single character.
int	**read**(char[] cbuf) Read characters into an array.
abstract int	**read**(char[] cbuf, int off, int len) Read characters into a portion of an array.
boolean	**ready**() Tell whether this stream is ready to be read.
void	**reset**() Reset the stream.
long	**skip**(long n) Skip characters.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

lock

```
protected Object lock
```

The object used to synchronize operations on this stream. For efficiency, a character-stream object may use an object other than itself to protect critical sections. A subclass should therefore use the object in this field rather than `this` or a synchronized method.

Constructor Detail

Reader

```
protected Reader()
```

Create a new character-stream reader whose critical sections will synchronize on the reader itself.

Reader

```
protected Reader(Object lock)
```

Create a new character-stream reader whose critical sections will synchronize on the given object.
Parameters:
`lock` - The Object to synchronize on.

Method Detail

read

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

Read a single character. This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

Returns:
The character read, as an integer in the range 0 to 65535 (`0x00-0xffff`), or -1 if the end of the stream has been reached

Throws:
`IOException` - If an I/O error occurs

---

read

```
public int read(char[] cbuf)
         throws IOException
```

Read characters into an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters:
`cbuf` - Destination buffer

Returns:
The number of bytes read, or -1 if the end of the stream has been reached

Throws:
`IOException` - If an I/O error occurs

---

read

```
public abstract int read(char[] cbuf,
                         int off,
                         int len)
                  throws IOException
```

Read characters into a portion of an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

Parameters:
`cbuf` - Destination buffer

`off` - Offset at which to start storing characters

`len` - Maximum number of characters to read

Returns:
The number of characters read, or -1 if the end of the stream has been reached

Throws:
`IOException` - If an I/O error occurs

---

skip

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

Skip characters. This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

Parameters:

n - The number of characters to skip

Returns:

The number of characters actually skipped

Throws:

IllegalArgumentException - If n is negative.

IOException - If an I/O error occurs

---

ready

```
public boolean ready()
              throws IOException
```

Tell whether this stream is ready to be read.

Returns:

True if the next read() is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

Throws:

IOException - If an I/O error occurs

---

markSupported

```
public boolean markSupported()
```

Tell whether this stream supports the mark() operation. The default implementation always returns false. Subclasses should override this method.

Returns:

true if and only if this stream supports the mark operation.

---

mark

```
public void mark(int readAheadLimit)
          throws IOException
```

Mark the present position in the stream. Subsequent calls to reset() will attempt to reposition the stream to this point. Not all character-input streams support the mark() operation.

Parameters:

`readAheadLimit` - Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.

Throws:

IOException - If the stream does not support mark(), or if some other I/O error occurs

---

reset

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

Reset the stream. If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the reset() operation, and some support reset() without supporting mark().

Throws:

`IOException` - If the stream has not been marked, or if the mark has been invalidated, or if the stream does not support reset(), or if some other I/O error occurs

---

close

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

Close the stream. Once a stream has been closed, further read(), ready(), mark(), or reset() invocations will throw an IOException. Closing a previously-closed stream, however, has no effect.

Throws:

`IOException` - If an I/O error occurs

Submit a bug or feature

For further API reference and developer documentation, see Java 2 SDK SE Developer Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Java, Java 2D, and JDBC are trademarks or registered trademarks of Sun Microsystems, Inc. in the US and other countries.
Copyright 1993-2001 Sun Microsystems, Inc. 901 San Antonio Road
Palo Alto, California, 94303, U.S.A. All Rights Reserved.