Java.io.Reader class in Java

Difficulty Level : Hard Last Updated : 30 Jan, 2017

It is an 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.

Constructors:

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

Methods:

- **abstract void close() :** Closes the stream and releases any system resources associated with it. Once the stream has been closed, further read(), ready(), mark(), reset(), or skip() invocations will throw an IOException. Closing a previously closed stream has no effect.

```
Syntax :public abstract void close()
                    throws IOException
Throws:
IOException
```

- **void mark(int readAheadLimit) :** Marks 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.

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

- **boolean markSupported() :** Tells whether this stream supports the mark()

operation.The default implementation always returns false. Subclasses should override this method.

```
Syntax :public boolean markSupported()
Returns:
true if and only if this stream supports the mark operation.
```

- **int read() :** Reads 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.

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

- **int read(char[] cbuf) :** Reads 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.

```
Syntax :public int read(char[] cbuf)
         throws IOException
Parameters:
cbuf - Destination buffer
Returns:
The number of characters read, or -1 if the end of the stream has been reac
Throws:
IOException
```

- **abstract int read(char[] cbuf, int off, int len) :** Reads 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.

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

- **int read(CharBuffer target) :** Attempts to read characters into the specified character buffer.The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

```
Syntax :public int read(CharBuffer target)
         throws IOException
Parameters:
target - the buffer to read characters into
Returns:
The number of characters added to the buffer,
or -1 if this source of characters is at its end
Throws:
IOException
NullPointerException
ReadOnlyBufferException
```

- **boolean ready() :** Tells whether this stream is ready to be read.

```
Syntax :public boolean ready()
              throws IOException
Returns:
True if the next read() is guaranteed not to block for input, false otherwi
Note that returning false does not guarantee that the next read will block.
Throws:
IOException
```

- **void reset() :** Resets 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().

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

Throws:
IOException

- **long skip(long n) :** Skips characters.This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

  Syntax :public long skip(long n)
  throws IOException
  Parameters:
  n - The number of characters to skip
  Returns:
  The number of characters actually skipped
  Throws:
  IllegalArgumentException - If n is negative.
  IOException

```
//Java program demonstrating Reader methods
import java.io.*;
import java.nio.CharBuffer;
import java.util.Arrays;
class ReaderDemo
{
    public static void main(String[] args) throws IOException
    {
        Reader r = new FileReader("file.txt");
        PrintStream out = System.out;
        char c[] = new char[10];
        CharBuffer cf = CharBuffer.wrap(c);

        //illustrating markSupported()
        if(r.markSupported()) {
            //illustrating mark()
            r.mark(100);
            out.println("mark method is supported");
        }
        //skipping 5 characters
        r.skip(5);

        //checking whether this stream is ready to be read.
        if(r.ready())
        {
            //illustrating read(char[] cbuf,int off,int len)
            r.read(c,0,10);
            out.println(Arrays.toString(c));

            //illustrating read(CharBuffer target )
            r.read(cf);
```

```
            out.println(Arrays.toString(cf.array()));

            //illustrating read()
            out.println((char)r.read());
        }
        //closing the stream
        r.close();
    }
}
```

Output :

```
[f, g, h, i, g, k, l, m, n, o]
[p, q, r, s, t, u, v, w, x, y]
z
```

This article is contributed by Nishant Sharma. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.