Java.io.BufferedReader Class in Java

Difficulty Level : Easy Last Updated : 31 Jan, 2017

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

- The buffer size may be specified, or the default size may be used. The default is large enough for most purposes.
- In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream.
- It is therefore advisable to wrap a BufferedReader around any Reader whose read() operations may be costly, such as FileReaders and InputStreamReaders.
- Programs that use DataInputStreams for textual input can be localized by replacing each DataInputStream with an appropriate BufferedReader.

Constructors:

- **BufferedReader(Reader in) :** Creates a buffering character-input stream that uses a default-sized input buffer.
- **BufferedReader(Reader in, int sz) :** Creates a buffering character-input stream that uses an input buffer of the specified size.

Methods:

- **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 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.

```
Syntax :public void mark(int readAheadLimit)
          throws IOException
Parameters:
readAheadLimit - Limit on the number of characters that may be read while
```

```
preserving the mark.
Throws:
IllegalArgumentException - If readAheadLimit is < 0
IOException
```

- **boolean markSupported() :** Tells whether this stream supports the mark() operation, which it does.

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

- **int read() :** Reads a single character.

```
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, int off, int len) :** Reads characters into a portion of an array.
  This method implements the general contract of the corresponding read method of the Reader class. As an additional convenience, it attempts to read as many characters as possible by repeatedly invoking the read method of the underlying stream. This iterated read continues until one of the following conditions becomes true:
  - The specified number of characters have been read,
  - The read method of the underlying stream returns -1, indicating end-of-file, or
  - The ready method of the underlying stream returns false, indicating that further input requests would block.

  If the first read on the underlying stream returns -1 to indicate end-of-file then this method returns -1. Otherwise this method returns the number of characters actually read.

```
Syntax :public 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 read
Throws:
IOException
```

- **String readLine() :** Reads a line of text.A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a linefeed.

```
Syntax :public String readLine()
                throws IOException
Returns:
A String containing the contents of the line, not including any
line-termination characters,
or null if the end of the stream has been reached
Throws:
IOException
```

- **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 to the most recent mark.

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

- **long skip(long n) :** Skips characters.

```
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

IOException

```java
//Java program demonstrating BufferedReader methods
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
class BufferedReaderDemo
{
    public static void main(String[] args) throws IOException
    {
        FileReader fr = new FileReader("file.txt");
        BufferedReader br = new BufferedReader(fr);
        char c[]=new char[20];

        //illustrating markSupported() method
        if(br.markSupported())
        {
            System.out.println("mark() method is supported");

            //illustrating mark method
            br.mark(100);
        }

        /*File Contents
        * This is first line
        this is second line
        */

        //skipping 8 characters
        br.skip(8);

        //illustrating ready() method
        if(br.ready())
        {
            //illustrating readLine() method
            System.out.println(br.readLine());

            //illustrating read(char c[],int off,int len)
            br.read(c);
            for (int i = 0; i <20 ; i++)
            {
                System.out.print(c[i]);
            }
            System.out.println();
```

```
                //illustrating reset() method
                br.reset();
                for (int i = 0; i <8 ; i++)
                {
                    //illustrating read() method
                    System.out.print((char)br.read());
                }
            }
        }
    }
```

Output :

```
mark() method is supported
first line
this is second line
This is
```

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.