Java handles the contents of a file as separate **tokens**, individual pieces of data filing sequentially along in a stream of information. The same **Scanner** class methods you used to accept user input from the keyboard (i.e., **next()**, **nextLine()**, **nextInt()**, and **nextDouble()**) can be used to pick tokens out of an input stream from a file. As with keyboard input, white space is the default delimiter (separator) for individual tokens in a sequential file.



This diagram is analogous to a stream of tokens in a file separated by white space. Notice that there is also an EOF (end-of-file) marker signifying the end of the stream. Removing the beads one at a time until the EOF marker (the knot) is encountered is like reading the tokens from a text file.

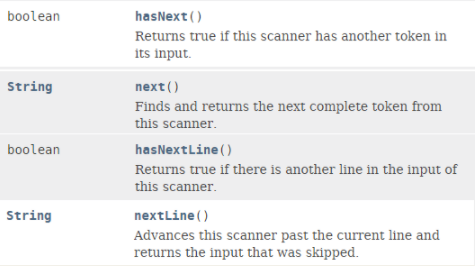
From your programmer's perspective, you probably suspect that the tokens in a file can be read with a **while** loop. The condition simply needs to recognize when there are no more tokens to be read. The EOF marker is the sentinel that forces the loop to stop looking for more tokens to read.

### **Using Scanner to Read Text Files**

Reading information from text files is very simple, but Java is very picky, so follow the pattern and you won’t have any trouble.

Information from an input stream can be read sequentially from a text file, using familiar methods of the **Scanner class**. The important concept to envision is that a stream of tokens is being read in from a file with a **while** loop, which continues to iterate as long as there are more tokens to be read.

Each of the following pairs of methods work in concert, to read in a stream of **String** data from a text file. Study the excerpts from the **Scanner** class API to gain an overview of how each of these methods performs.



The **boolean** methods **hasNext()** and **hasNextLine()** are used in the while loop condition to determine when the loop should terminate. The **hasNext()** method deals with individual tokens delimited by white space, whereas the hasNextLine() method deals with entire lines of tokens separated by a carriage return. As long as there are more tokens to read from the file, these methods will be evaluated as true and more information will be read by the **next()** and **nextLine()** methods, respectively. When the end-of-file (EOF) marker is encountered, the file contains no more tokens and the condition will evaluate to **false**, terminating the loop.

Each of the following pairs of methods works together to read in a stream of numeric data from a text file. Carefully study the excerpts from the **Scanner** class API.

