Class BasicTokenizer

```
java.lang.Object
  └ BasicTokenizer
```

All Implemented Interfaces:
java.util.Enumeration<java.lang.Object>

```
public class BasicTokenizer
extends java.lang.Object
implements java.util.Enumeration<java.lang.Object>
```

The string tokenizer class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the `StreamTokenizer` class. The `StringTokenizer` methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments.

The set of delimiters (the characters that separate tokens) may be specified either at creation time or on a per-token basis.

An instance of `StringTokenizer` behaves in one of two ways, depending on whether it was created with the `returnDelims` flag having the value `true` or `false`:

- If the flag is `false`, delimiter characters serve to separate tokens. A token is a maximal sequence of consecutive characters that are not delimiters.
- If the flag is `true`, delimiter characters are themselves considered to be tokens. A token is thus either one delimiter character, or a maximal sequence of consecutive characters that are not delimiters.

A `StringTokenizer` object internally maintains a current position within the string to be tokenized. Some operations advance this current position past the characters processed.

A token is returned by taking a substring of the string that was used to create the `StringTokenizer` object.

The following is one example of the use of the tokenizer. The code:

```
StringTokenizer st = new StringTokenizer("this is a test");
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

prints the following output:

```
this
is
a
test
```

`StringTokenizer` is a legacy class that is retained for compatibility reasons although its use is discouraged

in new code. It is recommended that anyone seeking this functionality use the `split` method of `String` or the java.util.regex package instead.

The following example illustrates how the `String.split` method can be used to break up a string into its basic tokens:

```
String[] result = "this is a test".split("\\s");
for (int x=0; x<result.length; x++)
    System.out.println(result[x]);
```

prints the following output:

```
this
is
a
test
```

Since:

JDK1.0

See Also:

`java.io.StreamTokenizer`

Constructor Summary

BasicTokenizer(java.lang.String str) Constructs a string tokenizer for the specified string.
BasicTokenizer(java.lang.String str, java.lang.String delim) Constructs a string tokenizer for the specified string.
BasicTokenizer(java.lang.String str, java.lang.String delim, boolean returnDelims) Constructs a string tokenizer for the specified string.

Method Summary

int	**countTokens**() Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.
boolean	**hasMoreElements**() Returns the same value as the `hasMoreTokens` method.
boolean	**hasMoreTokens**() Tests if there are more tokens available from this tokenizer's string.
java.lang.Object	**nextElement**() Returns the same value as the `nextToken` method, except that its declared return value is `Object` rather than `String`.
java.lang.String	**nextToken**() Returns the next token from this string tokenizer.

java.lang.String	nextToken(java.lang.String delim) Returns the next token in this string tokenizer's string.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BasicTokenizer

```
public BasicTokenizer(java.lang.String str)
```

Constructs a string tokenizer for the specified string. The tokenizer uses the default delimiter set, which is " \t\n\r\f": the space character, the tab character, the newline character, the carriage-return character, and the form-feed character. Delimiter characters themselves will not be treated as tokens.

Parameters:
str - a string to be parsed.
Throws:
java.lang.NullPointerException - if str is null

BasicTokenizer

```
public BasicTokenizer(java.lang.String str,
                      java.lang.String delim)
```

Constructs a string tokenizer for the specified string. The characters in the delim argument are the delimiters for separating tokens. Delimiter characters themselves will not be treated as tokens.

Note that if delim is null, this constructor does not throw an exception. However, trying to invoke other methods on the resulting StringTokenizer may result in a NullPointerException.

Parameters:
str - a string to be parsed.
delim - the delimiters.
Throws:
java.lang.NullPointerException - if str is null

BasicTokenizer

```
public BasicTokenizer(java.lang.String str,
                      java.lang.String delim,
                      boolean returnDelims)
```

Constructs a string tokenizer for the specified string. All characters in the delim argument are the

delimiters for separating tokens.

If the `returnDelims` flag is `true`, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is `false`, the delimiter characters are skipped and only serve as separators between tokens.

Note that if `delim` is `null`, this constructor does not throw an exception. However, trying to invoke other methods on the resulting `StringTokenizer` may result in a `NullPointerException`.

Parameters:

`str` - a string to be parsed.
`delim` - the delimiters.
`returnDelims` - flag indicating whether to return the delimiters as tokens.

Throws:

`java.lang.NullPointerException` - if str is `null`

Method Detail

countTokens

```
public int countTokens()
```

Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception. The current position is not advanced.

Returns:

the number of tokens remaining in the string using the current delimiter set.

See Also:

`StringTokenizer.nextToken()`

---

hasMoreElements

```
public boolean hasMoreElements()
```

Returns the same value as the `hasMoreTokens` method. It exists so that this class can implement the `Enumeration` interface.

Specified by:

`hasMoreElements` in interface `java.util.Enumeration<java.lang.Object>`

Returns:

`true` if there are more tokens; `false` otherwise.

See Also:

`Enumeration`, `StringTokenizer.hasMoreTokens()`

---

hasMoreTokens

```
public boolean hasMoreTokens()
```

Tests if there are more tokens available from this tokenizer's string. If this method returns `true`, then a subsequent call to `nextToken` with no argument will successfully return a token.

Returns:
`true` if and only if there is at least one token in the string after the current position; `false` otherwise.

---

nextElement

```
public java.lang.Object nextElement()
```

Returns the same value as the `nextToken` method, except that its declared return value is `Object` rather than `String`. It exists so that this class can implement the `Enumeration` interface.

Specified by:
`nextElement` in interface `java.util.Enumeration<java.lang.Object>`

Returns:
the next token in the string.
Throws:
`java.util.NoSuchElementException` - if there are no more tokens in this tokenizer's string.
See Also:
`Enumeration`, `StringTokenizer.nextToken()`

---

nextToken

```
public java.lang.String nextToken()
```

Returns the next token from this string tokenizer.

Returns:
the next token from this string tokenizer.
Throws:
`java.util.NoSuchElementException` - if there are no more tokens in this tokenizer's string.

---

nextToken

```
public java.lang.String nextToken(java.lang.String delim)
```

Returns the next token in this string tokenizer's string. First, the set of characters considered to be delimiters by this `StringTokenizer` object is changed to be the characters in the string `delim`. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters:
`delim` - the new delimiters.
Returns:
the next token, after switching to the new delimiter set.

Throws:

`java.util.NoSuchElementException` - if there are no more tokens in this tokenizer's string.
`java.lang.NullPointerException` - if delim is `null`
