Supplement: The StringTokenizer Class

For Introduction to Java Programming
By Y. Daniel Liang

The StringTokenizer class is a legacy class in Java. It can be replaced by the split method in the String class. You may still see this class in some legacy code. This section introduces the StringTokenizer class.

1 The StringTokenizer Class

The java.util.StringTokenizer class can be used to break a string into pieces so that information contained in it can be retrieved and processed. For example, to get all of the words in a string like "I am learning Java now", you create an instance of the StringTokenizer class for the string and then retrieve individual words in the string by using the methods in the StringTokenizer class, as shown in Figure 1.

java.util.StringTokenizer	
+StringTokenizer(s: String)	Constructs a string tokenizer for the string.
+StringTokenizer(s: String, delimiters: String)	Constructs a string tokenizer for the string with the specified delimiters.
+StringTokenizer(s: String, delimiters: String, returnDelimiters: boolean)	Constructs a string tokenizer for the string with the delimiters and returnDelims.
+countTokens(): int	Returns the number of remaining tokens.
+hasMoreTokens(): boolean	Returns true if there are more tokens left.
+nextToken(): String	Returns the next token.
+nextToken(delimiters: String): String	Returns the next token using new delimiters.

Figure 1

The StringTokenizer class provides the methods for processing tokens in a string.

How does the StringTokenizer class recognize individual words? You can specify a set of characters as delimiters when constructing a StringTokenizer object. Each delimiter is a character. The delimiters break a string into pieces known as *tokens*. You can specify delimiters in the StringTokenizer constructors:

[BL] public StringTokenizer(String s, String delim, boolean returnDelims)

Constructs a StringTokenizer for string s with specified delimiters. Each character in the string delim is a delimiter. If returnDelims is true, the delimiters are counted as tokens.

[BL] public StringTokenizer(String s, String delim)

Constructs a StringTokenizer for string s with specified delimiters delim, and the delimiters are not counted as tokens.

[BL] public StringTokenizer(String s)

Constructs a StringTokenizer for string s with default delimiters " \t\n\r" (a space, tab, new line, and carriage return), and the delimiters are not counted as tokens.

The following code creates a string tokenizer for a string using space as delimiters and extracts all the tokens.

<side remark: Line 2: create a string tokenizer>
******PD: Please add line numbers in the following code******

```
String s = "Java is cool.";
StringTokenizer tokenizer = new StringTokenizer(s);

System.out.println("The total number of tokens is " +
  tokenizer.countTokens());

while (tokenizer.hasMoreTokens())
  System.out.println(tokenizer.nextToken());
```

The code displays

```
The total number of tokens is 3
Java
is
cool.
```

Line 2 creates a string tokenizer using the default delimiters. If you create it using delimiters 'a' and 'c' (new StringTokenizer(s, "ac")), the output would be

```
The total number of tokens is 4
J
v
 is
ool.
```

If you want the delimiters to be counted as tokens, create a string tokenizer using new StringTokenizer(s, "ac", ture), the output would be

```
The total number of tokens is 7
J
a
v
a
 is
c
ool.
```

NOTE

<Side Remark: no no-arg constructor>

The StringTokenizer class does not have a no-arg constructor. Normally it is good programming practice to provide a no-arg constructor for each class. On rare occasions, however, a no-arg constructor does not make sense. StringTokenizer is such an example. A

StringTokenizer object must be created for a string that is to be passed as an argument from a constructor.