Strings in Java – The `StringTokenizer` Class

Java's Application Programming Interface (API) is a collection of hundreds of classes designed to add power and flexibility to the language. The `StringTokenizer` class is one of these, providing a number of methods for dealing with strings as a sequence of *tokens* that are separated by *delimiters*.

The most common delimiter is the blank space. Consider the following string:

```
String quote = "To be or not to be";
```

If blanks are used as delimiters, then this string contains six tokens: "To", "be", "or", "not", "to", and "be".

Consider the following code fragment, which assumes the variable `quote` has already been declared as shown above:

```
StringTokenizer st = new StringTokenizer(quote); //1
while (st.hasMoreTokens())                       //2
     System.out.println(st.nextToken());         //3
```

The output will be as shown to the right.

```
To
be
or
not
to
be
```

The explanation of each line of code is as follows:

1. Creates a new `StringTokenizer` object using the contents of the string `quote`. Since no delimiters are specified, the default delimiter of a blank space is used.
2. The `hasMoreTokens` method returns a boolean value to indicate of there are any more tokens to be processed.
3. The `nextToken` method *returns* the first available token from `st` and effectively removes it from the object.

`StringTokenizer` Constructors

The class has three overloaded constructors, each of which has a `String` parameter which is used to create a `StringTokenizer` object. The differences come in their treatment of delimiters.

- `public StringTokenizer (String s)`
  Creates a `StringTokenizer` object for the string `s`. It uses the default delimiter, which is a blank space. It also uses any other white space (newlines, tabs, carriage returns, form feeds) as delimiters.
- `public StringTokenizer (String s, String d)`
  Creates a `StringTokenizer` object for the string `s`. It uses *any* of the characters from the string `d` as delimiters. The characters in `d` will *not* be treated as tokens.
- `public StringTokenizer (String s, String d, boolean flag)`
  Same as above, using the string `s` to create the object, and `d` to define the delimiters. If the `flag` is set to `true`, the characters in `d` *will* also be treated as tokens (each characters will be a string of length one). If the `flag` is `false`, the delimiters are *not* treated as tokens.

To illustrate the second and third versions of the constructors, consider the following examples.

```
StringTokenizer st2 = new StringTokenizer("http://www.java.sun.com",":/.");
```

Delimiters are the colon (`:`), backslash (`/`) and period (`.`).

Tokens are: `"http"`, `"www"`, `"java"`, `"sun"`, `"com"`.

```
StringTokenizer st3 = new StringTokenizer("12*(345+6789)","*/+-()",true);
```

Delimiters are the arithmetic operators (`*, /, +, -`) and the left/right parentheses `()`.

Tokens will include the delimiters, since the flag is set to `true`.
Tokens: `"12" "*" "(" "345" "+" "6789" ")"`

`StringTokenizer` Methods

- public String nextToken()

  Returns the next available token from the *implicitly* referenced StringTokenizer object. Each call causes a reference to move along the string to the position of the next token. If there are no more tokens, calling this method will cause an exception.

- public boolean hasMoreTokens()

  Checks to see if there are any more tokens available. Returns true if and only if there is at least one token in the string after the current position.

- public int countTokens()

  Returns the number of tokens remaining in the implicit object.

Accessing the `StringTokenizer` Class

Almost all classes that we have discussed are part of the core package called `java.lang`. The contents of this package are available to our programs by default.

The `StringTokenizer` class is part of the `java.util` package, and must be specified for use in our programs. One way to accomplish this is through the use of an `import` statement at the beginning of a program.

```
import java.util.StringTokenizer;
```

We can also write a broader import which allows the use of all classes in the `java.util` package (which can be more useful if we require multiple packages from `java.util`).

```
import java.util.*;
```

Exercises

1. Assuming that the string `s` contains sentences that consist of words, blanks, and punctuation marks (period, comma, exclamation mark, question mark, colon, and semi-colon), construct a `StringTokenizer` object whose tokens consist only of the words in `s`.

2. Write a class method `wordCount` that has a single `String` parameter, `s`. Assuming that `s` consists of words separated by whitespace, the method should return the number of words in `s`.

3. Write a class method `value` that will return, as an `int`, the value of a simple arithmetic expression contained in a string. The string should contain two integers surrounding one of the operators `*, /, %, +, or -`. The string may also contain blanks. As an example, given that
   ```
   s = "  13 +    2"
   ```
   the method should return the value 15. The method should assume that its argument contains a valid expression.