Using iterators and tokenizing strings

Read Chapter 5
An iterator helps you repeat over a group of values

- It is easy to write for loops over sequences of integer values, but ...

- There are many other cases where you would like to loop over a group of values

- An **iterator** is an object that allows you to cycle through some group of values using two methods:
  - **hasNext()** tells you if you have reached the end of the group
  - **next()** gets the next object and advances one step further through the group
Let’s use an iterator to write `wordContaining()`

- From the `SpellChecker` class ...
An enumeration also allows you to repeat

- `java.util.Iterator` is an interface that defines `hasNext()` and `next()`

- `java.util Enumeration` is a similar interface that also provides methods for repeating over a group of objects:
  - `hasMoreElements()` tells you if you have reached the end of the group
  - `nextElement()` gets the next object and advances one step further through the group

- An enumeration is read-only, but the `Iterator` interface also provides a `remove()` method

- Note: adding elements while iterating usually causes exceptions or bugs
You can use a StringTokenizer to iterate over words

- So far, we have not done much with breaking up sentences into individual words

- Java provides a helpful tool for this: java.util.StringTokenizer

- Uses the same basic two-method pattern to let you iterate over all the “words” in a string

- You can pick the delimiters yourself—the default is whitespace characters
StringTokenizers are great for splitting strings

```java
StringTokenizer tokenizer =
    new StringTokenizer( "This is a sentence." );

while ( tokenizer.hasMoreTokens() )
{
    System.out.println( tokenizer.nextToken() );
}

Output (4 "words"):
This
is
a
Sentence.
```
Let’s add to the SpellChecker class

- Using a `StringTokenizer`, we can break lines up into words and spell-check each word ...
You can control the delimiters that define words

- By default, a `StringTokenizer` uses any whitespace character as a delimiter (spaces, tabs, newlines, etc.)

- You can provide your own set of delimiter characters that mark word boundaries instead:

```java
StringTokenizer tokenizer = new StringTokenizer("Hey--too! Much punctuation? For 'you'?", " ,.;:-+=[]{}|><'"!");

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

- Skips over sequences of delimiter characters