CS 157 Lab 13 November 12, 2009
Text File Processing

Create a new class called **Lab13**.

You can have a Scanner object "tokenize" a String variable. For example:

```
String text = "15  3.2 hello   9  27.5";
Scanner scan = new Scanner(text);
```

Your Scanner object **scan**, in this case, has a String called **text** inside it (instead of a File variable). But the Scanner methods work exactly the same as for a file. So `scan.nextInt()` would return then number 15 the first time you try it. If you do it again, it would generate an exception, because 3.2 is not an integer.

Write JUnit tests for each of the methods below.

1. Write a static method named `numberOfNumbers` that accepts a string of text as a parameter. Assume that the text is a series of integers. Return the total number of numbers contained in the string. For example, if the string is the following:

   `"5 7 2 8 9 10 12 98 7 14 20 22"`

   Your method should return **12**.

2. Write a static method named `sumOfNumbers` that accepts a string of text as a parameter as in number 1. Return the sum of the numbers contained in the string. For example, with the same string as above, your method should return **214**.

3. As above, write a method named `numberOfEvens` that returns the number of even numbers contained in the string. For example, with the string as above, your method should return **8**.

4. As above, write a method named `percentEvens` that returns the percentage of even numbers contained in the string. For example, with the string as above, your method should return `66.66666666666667`.

5. Write a static method named `negativeSum` that accepts a string of text as a parameter. Assume that the text is a series of integers, and determine whether or not the cumulative sum starting from the first number is ever negative. The method should return `true` if a negative sum can be reached and false if it can't be reached. For example, if the string contains the following text,

   `"38 4 19 -27 -15 -3 4 19 38"`

   your method will consider the sum of just one number (38), the sum of the first two numbers (38 + 4), the sum of the first three numbers (38 + 4 + 19), and so on up to the sum of all of the numbers. None of these sums is negative, so the method would return `false`. If the text is the following,

   `"14 7 -10 9 -18 -10 17 42 98"`
   the method finds that a negative sum is reached after adding 6 numbers together
   `(14 + 7 + -10 + 9 + -18 + -10)` and returns `true`.

6. Write a static method named `lineCount` that takes a `Scanner` representing a File as a parameter and that returns the number of lines in the file. For example, if the file contains the following text:

```
Twas brillig and the slithy toves
did gyre and gimble in the wabe.
All mimsey were the borogroves,
and the mome raths outgrabe.

"Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch."
```

   the program would find that there are 9 lines (blank lines count).

7. Write a method named `characterCount` that is similar to the one above, but returns the total number of characters (not counting any new-line `\n` characters) in the file. For example, with the file as above, your method should return 254.

8. Write a method named `longestLine` that returns the length of the longest line. You may assume that the input file has at least one line of input. For example, with the file above your method should return 41.