TDD Kata 1 - String Calculator

http://osherove.com/kata

Before you start:

- Try not to read ahead .
- Do one task at a time. The trick is to learn to work incrementally.
- Make sure you only test for **correct inputs**. there is no need to test for invalid inputs for this kata
- Test First!

String Calculator

1. In a test-first manner, create a simple class **`class StringCalculator`**
   with a method **`public int Add(string numbers)`**
   1. The method can take 0, 1 or 2 numbers, and will return their sum
      (for an `empty string` it will return 0)
      for example
      `“” == 0 ,   “1” == 1 ,  “1,2” == 3`
   2. Start with the simplest test case of an empty string and move to one & two numbers
   3. Remember to solve things as `simply as possible` so that you force yourself to write tests you did not think about
   4. Remember to refactor after each passing test

2. Allow the Add method to handle an unknown amount of numbers

3. Allow the Add method to handle new lines between numbers (instead of commas).
   1. the following input is ok: `“1\n2,3”` == 6
   2. the following is INVALID input so do not expect it : `“1,\n”` (not need to write a test for it)

4. Support different delimiters:
   to change a delimiter, the beginning of the string will contain a separate line
   that looks like this:

`“//[delimiter]\n[numbers…]”`
for example
`“//;\n1;2”` == 3
since the default delimiter is `‘;’` .

Note: All existing scenarios and tests should still be supported

5. Calling Add with a negative number will throw an exception `“negatives not allowed”` - and the negative that was passed.

6. If there are multiple negatives, show all of them in the exception message

7. Using TDD, Add a method to `StringCalculator`
called `public int GetCalledCount()`
that returns how many times Add() was invoked.
Remember - Start with a failing (or even non compiling) test.

8. (.NET Only) Using TDD, Add an `event` to the `StringCalculator` class named
    `public event Action<string, int> AddOccured` ,
    that is triggered after every Add() call.

Hint:
    Create the `event` declaration first:
    then write a failing test that listens to the event
    and proves it should have been triggered when calling `Add()`.

Hint 2:
    Example:

```
 string giveninput = null;
sc.AddOccured += delegate(string input,
                          int result)
{
    giveninput = input;
};
```

9. Numbers bigger than 1000 should be ignored, for example:
   `2 + 1001 == 2`

10. Delimiters can be of any length with the following format:
    `“//[delimiter]\n”`
    for example:
    `“//[***]\n1***2***3”` == 6

11. Allow multiple delimiters like this:
    `“//[delim1][delim2]\n”`
    for example
    `“//[*][%]\n1*2%3”` == 6.

12. make sure you can also handle multiple delimiters with length longer than one char
    for example
    `“//[**][%%]\n1**2%%3”` == 6.

For more info visit https://osherove.com or email roy@osherove.com