# CSD

Agilar

## String Calculator KATA

La siguiente ejercicio es una TDD KATA codificando, refactorizando y muy practica para aprender a practicar TDD.

## Antes de empezar

Intentar leer en orden, no leas mas de lo necesario.
Haz una tarea al mismo tiempo, la idea es que aprendas a trabajar de forma incremental.
Intenta probar solo las pruebas validas , no hace falta probar los inputs incorrectos, si te sobra tiempo entonces hazlas.

## String Calculator

1. Crear una calculadora de Strings con un metodo int Add(strings numbers)

    1. El método puede tener 0 , 1 o 2 números , y volverá su suma (para una cadena vacía devolverá 0 ) , por ejemplo, "" o " 1 " o " 1,2 “
    2. Comenzar con el caso de la prueba más simple de una cadena vacía y pasar a 1 y dos números
    3. Recuerda que debe resolver las cosas lo más simple posible para obligarte a escribir pruebas que no has pensado
    4. Recuerda refactorizar cuando ya has pasado el test.

2. Permitir que el método Add debe manejar un número indeterminado de operadores.
3. Permitir que el método Add maneje nuevas lineas entre números ( en lugar de comas)

    ej:
        Introducimos “1\n2,3” y el resultado = 6
        Introducimos 1,\n” y el resultado no es correcto.

**4.- Soportando diferentes delimitadores**

1. Para cambiar un delimitador , el principio de la cadena contendrá una línea separada que tiene este aspecto : "// [ delimitador ] \ n [números ... ] ",

    ej:
        "// ; \ n 1 ; 2 “ = 3 donde el delimitador es ‘;’

2. La primera línea es opcional . todos los escenarios existentes deben ser contemplados

5.- Si llamas a la función Add con un numero negativo debe responder con una excepción “Los números negativos no son permitidos” y especificar el número negativo , si hay mas de un número negativo debe presentarlos todos.

    ej:
        “-2,-5” = “Los numeros negativos no son permitidos -2,-5”

***Para aquí si es la primera vez que realizas TDD o si no has llegado a los 30 minutos para ejecutar los casos :***)

6.- Los números mayores de 1000 deben ser ignorados

ej:

1,1001 = 1

7.- Los delimitadores pueden ser de cualquier longitud siempre que sigan el siguiente formato "//[delimiter]\n"

ej: "//[***]\n1***2***3" = 6

8 .- Permitir multiples delimitadores siempre que sigan el siguiente formato "//[delim1][delim2]\n"

ej: "//[*][%]\n1*2%3" = 6

Asegurar que los delimitadores multiples pueden aceptar mas de una longitud