String Calculator TDD Kata – Documentation Report

Author: Janvi Patel

Assessment: Incubyte TDD Evaluation

Language: JavaScript (Node.js with Jest)

Repository: https://github.com/JanviJPatel30/stringCalculator-TDD

Objective

The objective of this task was to build a String Calculator using the principles of Test-Driven Development (TDD). The implementation was completed step-by-step using Red-Green-Refactor cycles, adhering to software craftsmanship principles. Each requirement was verified using unit tests, and additional bonus steps were implemented for extra credit.

Tools Used

- Node.js
- Jest (for Unit Testing)
- VS Code
- Git & GitHub
- TDD methodology

TDD Steps & Evolution

Step 1: Empty String Input

Test Case: expect(add("")) toBe(0);

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

FAIL ./calculator.test.js
  × returns 0 for an empty string (2 ms)

  ● returns 0 for an empty string

    TypeError: add is not a function

      2 |
      3 | test('returns 0 for an empty string', () => {
    > 4 |   expect(add("")).toBe(0);
        |          ^
      5 | });

      at Object.add (calculator.test.js:4:10)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        0.571 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: if (numbers === "") return 0;

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

PASS ./calculator.test.js
  √ returns 0 for an empty string (3 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.622 s, estimated 1 s
Ran all test suites.
PS C:\Users\hp\Pictures\MTech\stringCalculator>
```

Step 2: Single Number

Test Case: expect(add("1")) toBe(1);

```
> stringcalculator@1.0.0 test
> jest

 FAIL  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  × returns number when a single number is provided (3 ms)

  ● returns number when a single number is provided

    expect(received).toBe(expected) // Object.is equality

    Expected: 1
    Received: undefined

       6 |
       7 | test('returns number when a single number is provided', () => {
    >  8 |   expect(add("1")).toBe(1);
         |                    ^
       9 | });
      10 |

      at Object.toBe (calculator.test.js:8:20)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:   0 total
Time:        0.579 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: return parseInt(numbers);

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (2 ms)
  √ returns number when a single number is provided

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.532 s, estimated 1 s
Ran all test suites.
```

Step 3: Two Numbers

Test Case: expect(add("1,2")) toBe(3);

```
> stringcalculator@1.0.0 test
> jest

FAIL ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided
  × returns sum when two comma-separated numbers are provided (2 ms)

  ● returns sum when two comma-separated numbers are provided

    expect(received).toBe(expected) // Object.is equality

    Expected: 3
    Received: 1

      10 |
      11 | test('returns sum when two comma-separated numbers are provided', () => {
    > 12 |   expect(add("1,2")).toBe(3);
         |                      ^
      13 | });
      14 |

      at Object.toBe (calculator.test.js:12:22)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        0.578 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Split by ',' and reduce to sum

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

PASS ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.55 s, estimated 1 s
Ran all test suites.
```

Step 4: Multiple Numbers

Test Case: expect(add("1,2,3,4")) toBe(10);

Implementation Approach: Handled by same split-reduce logic

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.666 s, estimated 1 s
Ran all test suites.
```

Step 5: Newline as Delimiter

Test Case: expect(add("1\n2,3")) toBe(6);

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 FAIL  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  × returns sum when newlines are used as delimiters (2 ms)

  ● returns sum when newlines are used as delimiters

    expect(received).toBe(expected) // Object.is equality

    Expected: 6
    Received: 4

      18 |
      19 | test('returns sum when newlines are used as delimiters', () => {
    > 20 |   expect(add("1\n2,3")).toBe(6);
         |                         ^
      21 | });
      22 |

      at Object.toBe (calculator.test.js:20:25)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 4 passed, 5 total
Snapshots:   0 total
Time:        0.676 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Replace '\n' with ','

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided (1 ms)
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters (1 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        0.674 s, estimated 1 s
Ran all test suites.
```

Step 6: Custom Single Delimiter

Test Case: expect(add("//;\n1;2")) toBe(3);

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 FAIL  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  √ returns sum when newlines are used as delimiters (1 ms)
  × supports custom delimiter defined at the start (3 ms)

  ● supports custom delimiter defined at the start

    expect(received).toBe(expected) // Object.is equality

    Expected: 3
    Received: NaN

      22 |
      23 | test('supports custom delimiter defined at the start', () => {
    > 24 |   expect(add("//;\n1;2")).toBe(3);
         |                           ^
      25 | });
      26 |

      at Object.toBe (calculator.test.js:24:27)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 5 passed, 6 total
Snapshots:   0 total
Time:        0.659 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Extract and use delimiter from string

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (2 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided (1 ms)
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        0.608 s, estimated 1 s
Ran all test suites.
```

Step 7: Count Method Calls

Test Case: expect(getCalledCount()) toBe(2);

```
> stringcalculator@1.0.0 test
> jest

 FAIL  ./calculator.test.js
  √ returns 0 for an empty string (2 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided (1 ms)
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start
  √ throws an exception when a negative number is used (9 ms)
  √ throws an exception when multiple negative numbers are used (1 ms)
  × getCalledCount returns how many times add was called

  ● getCalledCount returns how many times add was called

    TypeError: StringCalculator is not a constructor

      36 |
      37 | test('getCalledCount returns how many times add was called', () => {
    > 38 |   const calculator = new StringCalculator();
         |                      ^
      39 |   calculator.add("1,2");
      40 |   calculator.add("3");
      41 |   expect(calculator.getCalledCount()).toBe(2);

      at Object.<anonymous> (calculator.test.js:38:22)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 8 passed, 9 total
Snapshots:   0 total
Time:        0.678 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Use `this.callCount` and increment

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  √ returns sum when newlines are used as delimiters
  √ supports custom delimiter defined at the start
  √ throws an exception when a negative number is used (11 ms)
  √ throws an exception when multiple negative numbers are used (1 ms)
  √ getCalledCount returns how many times add was called (1 ms)

Test Suites: 1 passed, 1 total
Tests:       9 passed, 9 total
Snapshots:   0 total
Time:        0.583 s, estimated 1 s
Ran all test suites.
```

Step 8: Negative Numbers

Test Case: expect(() => add("1,-2")).toThrow(...);

expect(() => add("1,-2,-5")).toThrow("negative numbers not allowed -2,-5");

```
> stringcalculator@1.0.0 test
> jest

FAIL ./calculator.test.js
  √ returns 0 for an empty string (2 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start
  × throws an exception when a negative number is used (1 ms)
  × throws an exception when multiple negative numbers are used

  ● throws an exception when a negative number is used

    expect(received).toThrow(expected)

    Expected substring: "negative numbers not allowed -2"

    Received function did not throw

      26 |
      27 | test('throws an exception when a negative number is used', () => {
    > 28 |   expect(() => add("1,-2,3")).toThrow("negative numbers not allowed -2");
         |                               ^
      29 | });
      30 |
      31 | test('throws an exception when multiple negative numbers are used', () => {

      at Object.toThrow (calculator.test.js:28:31)

  ● throws an exception when multiple negative numbers are used

    expect(received).toThrow(expected)

    Expected substring: "negative numbers not allowed -2,-5"

    Received function did not throw

      30 |
      31 | test('throws an exception when multiple negative numbers are used', () => {
    > 32 |   expect(() => add("1,-2,-5,3")).toThrow("negative numbers not allowed -2,-5");
         |                                  ^
      33 | });
      34 |

      at Object.toThrow (calculator.test.js:32:34)

Test Suites: 1 failed, 1 total
Tests:       2 failed, 6 passed, 8 total
Snapshots:   0 total
Time:        0.683 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Throw error if any number < 0

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (6 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start (1 ms)
  √ throws an exception when a negative number is used (11 ms)
  √ throws an exception when multiple negative numbers are used (2 ms)

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        0.654 s, estimated 1 s
Ran all test suites.
```

Step 9: Ignore > 1000

Test Case: expect(add("2,1001")) toBe(2);

```
> stringcalculator@1.0.0 test
> jest

 FAIL  ./calculator.test.js
  √ returns 0 for an empty string (2 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided (1 ms)
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters
  √ supports custom delimiter defined at the start
  √ throws an exception when a negative number is used (10 ms)
  √ throws an exception when multiple negative numbers are used (2 ms)
  √ getCalledCount returns how many times add was called (1 ms)
  × ignores numbers greater than 1000 (3 ms)

  ● ignores numbers greater than 1000

    expect(received).toBe(expected) // Object.is equality

    Expected: 2
    Received: 1003

      50 | test('ignores numbers greater than 1000', () => {
      51 |   const calculator = new StringCalculator();
    > 52 |   expect(calculator.add("2,1001")).toBe(2);
         |                                    ^
      53 | });
      54 |

      at Object.toBe (calculator.test.js:52:36)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 9 passed, 10 total
Snapshots:   0 total
Time:        0.606 s, estimated 1 s
Ran all test suites.
```

Implementation Approach: Filter out numbers > 1000

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start (1 ms)
  √ throws an exception when a negative number is used (10 ms)
  √ throws an exception when multiple negative numbers are used (1 ms)
  √ getCalledCount returns how many times add was called (1 ms)
  √ ignores numbers greater than 1000 (1 ms)

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        0.603 s, estimated 1 s
Ran all test suites.
```

Step 10: Multi-character Delimiter

Test Case: expect(add("//[***]\n1***2***3")) toBe(6);

Implementation Approach: Extract delimiters inside [ ] and join with '|'

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided
  √ returns sum when two comma-separated numbers are provided (1 ms)
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters
  √ supports custom delimiter defined at the start
  √ throws an exception when a negative number is used (9 ms)
  √ throws an exception when multiple negative numbers are used (2 ms)
  √ getCalledCount returns how many times add was called (1 ms)
  √ ignores numbers greater than 1000 (1 ms)
  √ supports delimiters of any length (2 ms)

Test Suites: 1 passed, 1 total
Tests:       11 passed, 11 total
Snapshots:   0 total
Time:        0.589 s, estimated 1 s
Ran all test suites.
```

Step 11: Multiple Delimiters
Test Case: expect(add("//[*][%]\n1*2%3")) toBe(6);

Implementation Approach: Parse all and combine as RegExp

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers
  √ returns sum when newlines are used as delimiters
  √ supports custom delimiter defined at the start
  √ throws an exception when a negative number is used (11 ms)
  √ throws an exception when multiple negative numbers are used (2 ms)
  √ getCalledCount returns how many times add was called (1 ms)
  √ ignores numbers greater than 1000
  √ supports delimiters of any length
  √ supports multiple delimiters

Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        0.595 s, estimated 1 s
Ran all test suites.
```

Step 12: Multi-character Multiple Delimiters
Test Case: expect(add("//[**][%%]\n1**2%%3")) toBe(6);

Implementation Approach: Supports multiple custom delimiters with any length

```
PS C:\Users\hp\Pictures\MTech\stringCalculator> npm test

> stringcalculator@1.0.0 test
> jest

 PASS  ./calculator.test.js
  √ returns 0 for an empty string (3 ms)
  √ returns number when a single number is provided (1 ms)
  √ returns sum when two comma-separated numbers are provided
  √ returns sum for multiple comma-separated numbers (1 ms)
  √ returns sum when newlines are used as delimiters (1 ms)
  √ supports custom delimiter defined at the start (1 ms)
  √ throws an exception when a negative number is used (9 ms)
  √ throws an exception when multiple negative numbers are used (1 ms)
  √ getCalledCount returns how many times add was called (1 ms)
  √ ignores numbers greater than 1000 (1 ms)
  √ supports delimiters of any length (1 ms)
  √ supports multiple delimiters (1 ms)
  √ supports multiple delimiters with multiple characters

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:   0 total
Time:        0.579 s, estimated 1 s
Ran all test suites.
```

Conclusion

This assessment was implemented using JavaScript with Jest for unit testing, following a disciplined Test-Driven Development approach. Each feature was developed iteratively using the Red-Green-Refactor cycle, with frequent commits and clear test coverage. The process reflects key principles like extreme ownership, pragmatic quality, and feedback-driven improvement - closely aligned with Incubyte's values of software craftsmanship and continuous mastery.