INTRO TO TDD

boltmade

WHAT IS TDD?

TEST DRIVEN DEVELOPMENT

"first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards."

— Wikipedia

1. WRITE A FAILING TEST

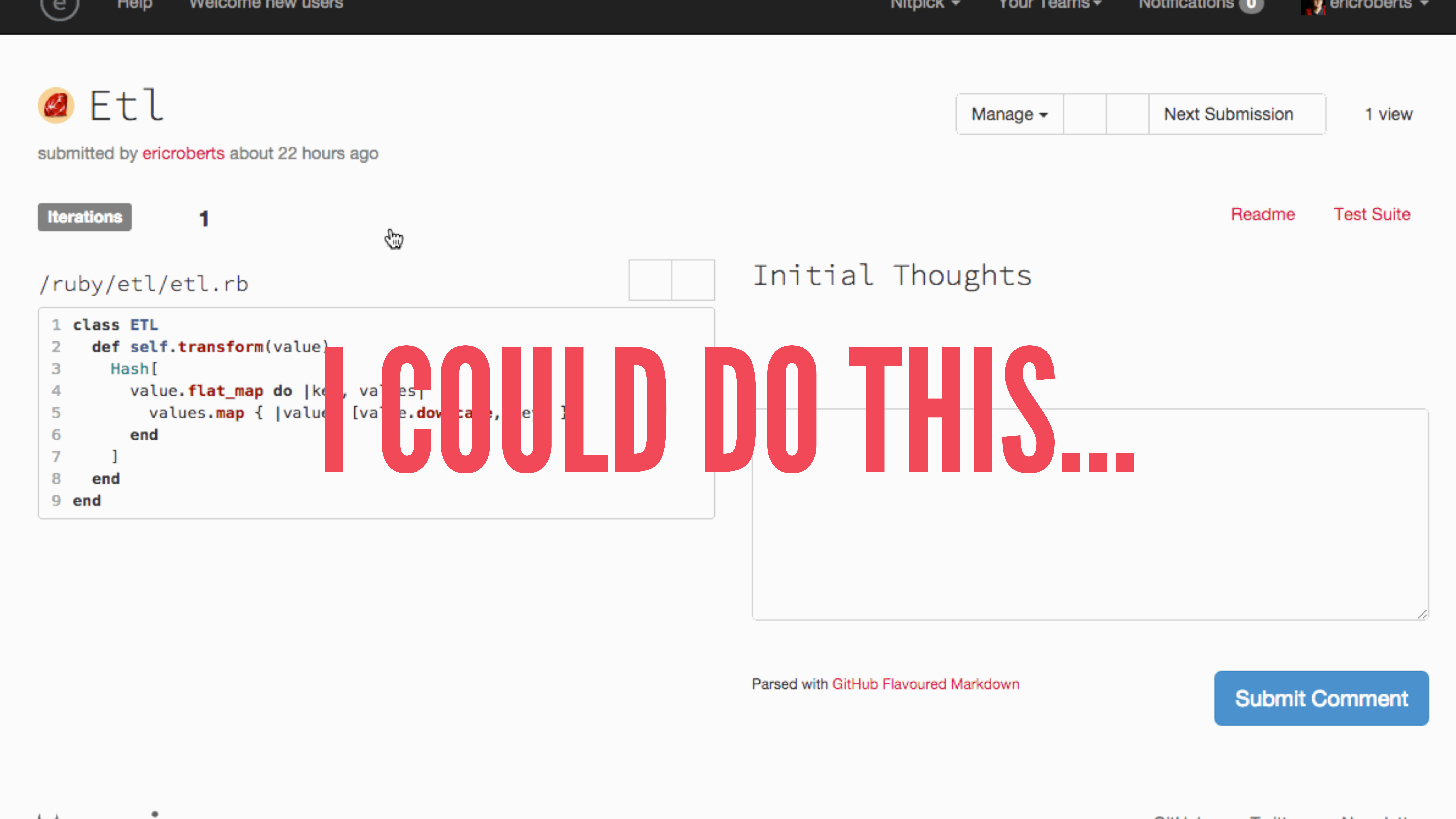
2. PASS THE TEST

3. REFACTOR

RED, GREEN, REFACTOR

WHY DO TDD?

SHORT FEEDBACK CYCLE



Etl

submitted by ericroberts about 22 hours ago

/ruby/etl/etl.rb

```ruby
class ETL
  def self.transform(value)
    Hash[
      value.flat_map do |key, values|
        values.map { |value| [value.downcase, key] }
      end
    ]
  end
end
```

Initial Thoughts

Parsed with GitHub Flavoured Markdown

I COULD DO THIS...

```
Pierrty ~/Code/Presentations/intro-to-tdd [master*]:
```

OR I COULD DO THIS!

SMALLER PIECES

"even if I couldn't imagine an implementation I could almost always figure out how to write a test."

Kent Beck

IMPROVE DESIGN

- Be a client of your own code.
- If it's hard to write the test, rethink your code.

REFACTOR WITH CONFIDENCE

- Stop avoiding old code because you are afraid of breaking it.
  - Rework becomes enjoyable, not stressful.

WHY WRITE THE TESTS FIRST?

- Demonstrates understanding of what it is we are going to do.
- Ensures we are testing the right thing.

LET'S GET STARTED!

There should be a method on Calculator called "add" that takes a string representation of a number ("1") and returns it as a number (1).

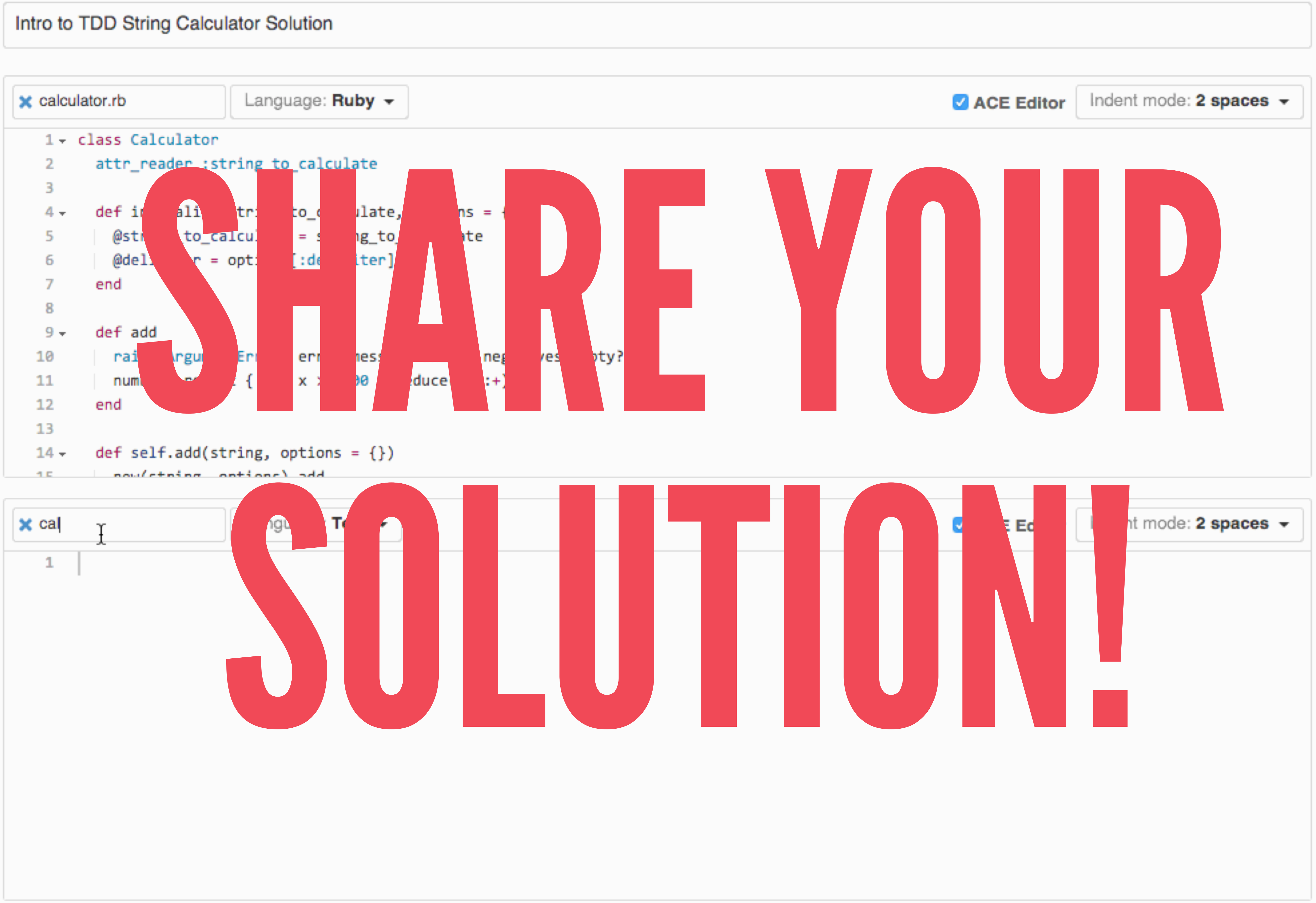
When I give the method an empty string (""), it should return 0.

```ruby
require "minitest/autorun"
require_relative "calculator"

class TestCalculator < MiniTest::Unit::TestCase
  def test_given_string1_returns_1
    assert_equal Calculator.add("1"), 1
  end

  def test_given_string_returns_number
    number = rand(100)
    assert_equal Calculator.add(number.to_s), number
  end
end
```

```ruby
class Calculator
  def self.add(string)
    string.to_i
  end
end
```

YOUR TURN!

SHARE YOUR SOLUTION!



Intro to TDD String Calculator Solution

calculator.rb · Language: **Ruby** · ACE Editor ✓ · Indent mode: **2 spaces**

```ruby
class Calculator
  attr_reader :string_to_calculate

  def i[illegible]ali[illegible]tr[illegible]to_c[illegible]ulate, [illegible]ns = {
    @str[illegible]to_calcu[illegible] = s[illegible]ng_to_[illegible]te
    @del[illegible]r = opt[illegible][:de[illegible]iter]
  end

  def add
    rai[illegible]rgu[illegible]Er[illegible] err[illegible]mess[illegible]neg[illegible]ves[illegible]pty?
    num[illegible] { [illegible] x [illegible]00 [illegible]duce[illegible]:+)
  end

  def self.add(string, options = {})
    new(string, options).add
```

cal · Indent mode: **2 spaces**

FURTHER READING

- Test Driven Development on Wikipedia
- The Three Rules of TDD by Uncle Bob
- Test Driven Development from The Art of Agile Development
- Extreme Programming
- Test Driven Development By Example
- String Calculator Solution by ericroberts