TASK4 program testing with RSpec

So for this task I chose to test my rectangle program. I also chose RSpec to test my program, and I chose RSpec because I was kind of familiar with the syntax coming from javascript testing with jest and enzyme.

RSpec is a testing tool for for Ruby applications. It is considered a behavior driven development framework. The tests look at a ruby object or class, and describes what that object should be doing. It is intended to be very intuitive, like the rest of ruby. It's used in a lot of ruby and rails enterprise level applications.

Here's a link for RSpec documentation.

To get started with RSpec first we need to install rspec:

```
gem install rspec
```

Then to set it up in a project, all we need to do is initialize it with this command:

```
rspec --init
```

Doing this task I've learnt that I made a few mistakes in TASK3 and corrected them in this task. First for it to work, I removed the constructor in the Rectangle class, so rectangle class looked something like this now:

```ruby
class Rectangle

  # method to get a perimeter of a rectangle
  def get_perimeter(width, length)
    (2 * length) + (2 * width)
  end

  # method to get area of a rectangle
  def get_area(width, length)
    width * length
  end

  # method to calculate rectangles diagonal length
  def get_diagonal_length(width, length)
    Math.sqrt((length * length) + (width * width))
  end

  # method to get x coordinate
  def get_coordinate_x(coordinate_x, length)
    coordinate_x + length / 2
  end

  # method to get y coordinate
  def get_coordinate_y(coordinate_y, width)
    coordinate_y + width / 2
  end
end
```

And I also fixed my object when creating it in calculations file. So I fixed it from this:

```ruby
rectangle = RectangleClass.new(length, width, x_coordinate, y_coordinate)
```

To this:

```ruby
rectangle = Rectangle.new
```

So in my test file I had to import rspec and the class file I want to test which I've done right here:

```
require 'rspec/autorun'
require_relative './rectangle.rb'
```

Now I got to write some tests.

```
describe Rectangle do
  subject(:rectangle) {Rectangle.new}
  context "perimeter tests" do
    it "get perimeter of 6 & 8 = 28" do
      expect(subject.get_perimeter( width 6, length 8)).to eq(28)
    end
    it "get perimeter of 8 & 10 != 28" do
      expect(subject.get_perimeter( width 8,  length 10)).not_to eq(28)
    end
  end

  context "area tests"  do
    it "get area of 2 & 4 = 8" do
      expect(subject.get_area( width 2, length 4)).to eq(8)
      end
    it "get area of 2 & 9 != 8" do
      expect(subject.get_area( width 2, length 9)).not_to eq(8)
    end
  end
end
```

I could have written let instead of subject and then it test cases I would have just written the name of it. But I found out that even better method is to write a subject. A subject is basically another version of let. The only difference is that you can only have one subject, and it's meant to be an instance of the main object you're testing.

When running these tests I get the output that I pass them:

```
Finished in 0.00853 seconds (files took 0.2598 seconds to load)
4 examples, 0 failures
```

But what if we want to fail some of them to check it really works?

What I'm going to do for a test to fail, is that I'm going to just change up a few numbers. What I did is change perimeter numbers from 6 to 7 and it still expects for it to be equal to 28, but in fact it's not equal to 28 anymore, it's equal to 30.

```
it "get perimeter of 6 & 8 = 28" do
  expect(subject.get_perimeter( width 7, length 8)).to eq(28)
end
```

```
Failures:

  1) Rectangle perimeter tests get perimeter of 6 & 8 = 28
     Failure/Error: expect(subject.get_perimeter(7,8)).to eq(28)

       expected: 28
            got: 30

       (compared using ==)
     # ./test.rb:8:in `block (3 levels) in <main>'

Finished in 0.06731 seconds (files took 1.17 seconds to load)
4 examples, 1 failure
```