Use case #1

Can chain to `allow` call to make sure that for certain cases original implementation of the method is called.

E.g. we have the following spec which will succeed

```
require 'calculator'

RSpec.describe "and_call_original" do
  it "can be overridden for specific arguments using #with" do
    allow(Calculator).to receive(:add).and_call_original
    allow(Calculator).to receive(:add).with(2, 3).and_return(-5)

    expect(Calculator.add(2, 2)).to eq(4)
    expect(Calculator.add(2, 3)).to eq(-5)
  end
end
```

It tells Rspec to use original implementation for any call of `.add` method, **except** when args are `2,3` . Because for that case we provide fake answer - `-5`.

Note

If we remove the `.and_call_original` line we will get the following error from Rspec

```
Please stub a default value first if message might be received with other args as
well.
```

Meaning if we call a method more than once with diff args and we fake the answer for only one case, we must let Rspec know what to do in other cases.

This also means the test will pass if we comment out the `.and_call_original` line **AND** leave only the second expect like so:

```
require "../and_call_original/calculator"

RSpec.describe "and_call_original" do
        it "trying to figure out what it does" do
                # allow(Calculator).to receive(:add).and_call_original
                allow(Calculator).to receive(:add).with(2, 3).and_return(-5)
```

```
                # expect(Calculator.add(2, 2)).to eq(4)
                expect(Calculator.add(2, 3)).to eq(-5)
        end
end
```