Mocking - doubles workshop

The only real class in a unit test should be the class that's being tested. If you are using other classes, the test isn't properly isolated. Replace them with doubles to isolate your test! One way you can check this is by commenting out the internals of a class. Only *that class's* unit tests should break. If unit tests break for other classes as well, you have some mocking to do!

let and double `let` allows us to create a variable, which will reset for each test. Like like this:

```
let(:variable_name) { 10 }
puts variable_name # => 10
```

`double` creates an empty double object, that can stand in for a real object in our tests:

```
# exam_spec.rb

it 'can total correct answers' do
  correct_answer = double('correct answer', :correct? => true)
  incorrect_answer = double('correct answer', :correct? => true)
  exam = Exam.new(Date.today, [correct_answer, incorrect_answer])
  expect(exam.score).to eq 1
end
```

We can combine `let` and `double` to create a new double for each test:

```
# exam_spec.rb

let(:correct_answer) { double('correct answer', :correct? => true) }
let(:incorrect_answer) { double('incorrect answer', :correct? => false) }

it 'can total correct answers' do
  exam = Exam.new(Date.today, [correct_answer, incorrect_answer])
```

```
  expect(exam.score).to eq 1
end
```

`subject` and `let` behave very similarly, and the differences between them are quite advanced. For now, it's enough to know that `subject` typically is used to create an instance of *the class you are testing*, but `let` is used for everything else. You therefore wouldn't use `subject` with `double` because you don't want to mock the class you are testing.

```
# exam_spec.rb

let(:correct_answer) { double('correct answer', :correct? => true) }
let(:incorrect_answer) { double('incorrect answer', :correct? => false) }

subject(:exam) { Exam.new(Date.today, [correct_answer, incorrect_answer]) }

it 'can total correct answers' do
  expect(exam.score).to eq 1
end
```

:symbols vs 'strings' The short version:

> A Symbol is a lightweight String. It uses less memory and can do less stuff.

The long version:

> We use Symbols as

let (:variable_name) do

double (:label, :method ⇒ response)

end