Arrange, Act, Assert:

```
describe Oystercard do
it 'something' do
   expect(subject.method).to eq(something)
end
end
```

Eg. Used in Oystercard to say the card starts with a balance of 0

```
it ‘expect the card to have zero balance' do
   expect(subject.balance).to eq(0)
end
```

Error:

```
1) Oystercard has initial balance
    Failure/Error: expect(subject.balance).to eq 0
    NoMethodError:
      undefined method `balance' for #<Oystercard:0x007fb811106ca8>
    # ./spec/oystercard_spec.rb:5:in `block (2 levels) in <top (required)>'
```

Code response to the error:

Before refactoring:

```
def balance
  0
 end
```

After refactoring:

```
attr_reader :balance

  def initialize
    @balance = 0
  end
```

```
describe '#top_up' do
it { is_expected.to respond_to(:method).with(1).argument }
end
```

Eg. Used in Oystercard to tell the Oystercard that top_up has 1 argument

```
it { is_expected.to respond_to(:top_up).with(1).argument }
```

Error:

```
1) Oystercard#top_up should respond to #top_up with 1 argument
    Failure/Error: it {is_expected.to respond_to(:top_up).with(1).argument }
      expected #<Oystercard:0x007f8bf29fd8e8 @balance=0> to respond
to :top_up with 1 argument
    # ./spec/oystercard_spec.rb:10:in `block (3 levels) in <top (required)>'
```

Code response to the error:

```
def top_up(amount)

  end
```

```
describe '#top_up' do
it { is_expected.to respond_to(:method).with(1).argument }

it 'can top up the balance' do
   expect{ subject.top_up 1 }.to change{ subject.balance }.by 1
end
```
Eg. Used in Oystercard to tell the Oystercard to top_up by x amount
```
expect{ subject.top_up 1 }.to change{ subject.balance }.by 1
```
Code to go with test:
```
def top_up(amount)
  @balance = @balance + amount
end
```

```
expect{ subject.method(argument) }.to raise_error ‘what causes the error'
```
Eg Used in Oystercard to set a maximum balance
```
  it 'raises an error if the maximum balance is exceeded' do
    maximum_balance = Oystercard::MAXIMUM_BALANCE
    subject.top_up(maximum_balance)
    expect{ subject.top_up 1 }.to raise_error 'Maximum balance exceeded'
  end
```
Error:
```
1) Oystercard#top_up raises an error if the maximum balance is exceeded
     Failure/Error: maximum_balance = Oystercard::BALANCE_LIMIT
     NameError:
       uninitialized constant Oystercard::BALANCE_LIMIT
     # ./spec/oystercard_spec.rb:18:in `block (3 levels) in <top (required)>'
```