RubyKoans Sandwich-Code

```
def count_lines(file_name)
  file = open(file_name)
  count = 0
  while file.gets
    count += 1
  end
  count
ensure
  file.close if file
end

def test_counting_lines
  assert_equal 4, count_lines("example_file.txt")
end

def find_line(file_name)
  file = open(file_name)
  while line = file.gets
    return line if line.match(/e/)
  end
ensure
  file.close if file
end
```

If you look at the methods :count_lines and :find_line above, it has one functionality in common.

> *Open file Read file Close file*

It is clear that opening and closing the file functions are same for both.

> *Assume them as buns. They are constant.*

Only the middle function changes.

> *Assume them as meat. Keeps changing.*

Hence the name sandwich-code.

Now to re-factoring the code for efficiency and better understanding,

```
def file_sandwich(file_name)
  file = open(file_name)
  yield(file)
ensure
  file.close if file
end
```

The above method handles both opening and closing the file for us and can be called by other methods(meat). The :file_sandwich is further simplified by Ruby built-in method

:open,

```
  open(file_name) do |file|
```

RubyKoans Proxy Object

```
class Proxy
  def initialize(target_object)
      @object = target_object
  end
end
```

1. Above, we are given a class named Proxy that could act as a proxy for another class object.

   ```
   class Television
   attr_accessor :channel

   def power
    if @power == :on
      @power = :off
    else
      @power = :on
    end
   end

   def on?
    @power == :on
   end
   end
   ```

2. Class Television provided.

```
  tv = Proxy.new(Television.new)
```

3. Above we have managed to,

   > *Create a proxy object for Television Television.new is a Television object which is passed to Proxy class for initiation as shown below.*

   ```
        class Proxy
            def initialize(target_object)
                @object = target_object
            end
        end
   ```

   > *Which now satisfies the assert:*

   ```
        assert tv.instance_of?(Proxy)       => returns true.
   ```

   > *Proving that tv is now a proxy object for Proxy of Television.*

In this project we are simply asked to do,

> *Write methods to override the default methods [:method_missing, :called?, number_of_times_called] so as to satisfy the test cases given. Ruby provides Object.messages to show all the methods of the said object. Since we need to override it for the test cases we use Ruby provided:*

```
   attr_reader: messages
```

Re-Factoring the Proxy class

Refer to https://www.ruby-lang.org/en/documentation/

```
class Proxy
  attr_reader :messages
    def initialize(target_object)
        @object = target_object
        @messages = []
    end

    def called?(msg)
        @messages.include?(msg)
    end

    def number_of_times_called(msg)
        @messages.count(msg)
    end

    def method_missing(method_name, *args, &block)
        @messages << method_name
        @object.send(method_name, *args, &block)
    end
end
```