

Course Day 13, June 9, SUN

Weekend challenge - continuation - Sunday morning

☒ Working on the second user story

As a customer

So that I can order the meal I want

I would like to be able to select some number of several available dishes

☒ reflect on how to move the dish to a receipt & how to test it

☒ review my notes from review of hashes

Got distracted with reading bits of conversation on slack from friday. Someone posted tutorial for doubles: https://www.tutorialspoint.com/rspec/rspec_test_doubles.htm

☒ Moment of confusion and exploration - how to write test that checks if the dishes and number of the ordered has been saved? Totally messed code.

```
Project
├── takeaway-challenge
│   ├── .git
│   ├── coverage
│   ├── docs
│   ├── review.md
│   ├── examples of code
│   │   ├── Eds_code.rb
│   │   ├── Rhys_code.rb
│   │   └── Rhys_code2.rb
│   ├── lib
│   │   ├── .DS_Store
│   │   ├── list.rb
│   │   └── spec
│   │       ├── list_spec.rb
│   │       └── spec_helper.rb
│   ├── .DS_Store
│   ├── .gitignore
│   ├── .rspec
│   ├── .rubocop.yml
│   ├── CONTRIBUTING.md
│   ├── Gemfile
│   ├── Gemfile.lock
│   ├── LICENSE
│   ├── Rakefile
│   └── README.md
├── list_spec.rb
├── spec_helper.rb
├── .DS_Store
├── .gitignore
├── .rspec
├── .rubocop.yml
├── CONTRIBUTING.md
├── Gemfile
├── Gemfile.lock
├── LICENSE
├── Rakefile
└── README.md
```

```
list_spec.rb
42 it "responds to the method #add_dish" do
43   list = List.new
44   expect(list).to respond_to(:add_dish)
45 end
46
47 it "adds a new dish to the receipt" do
48   list = List.new
49   dish_name = "pierogi"
50   quantity = 2
51   expect(list.add_dish(dish_name).to eq(list.recipe)
52     # expect(list.add_dish(dish_name, quantity)).to eq(list.recipe)
53 end
54
55
56
57
58
59 #it "takes a dish and adds it to the receipt" do
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
list.rb
7 @receipt = {}
8 end
9
10 def show_menu
11   @menu
12 end
13
14 #def add_dish(dish_name, quantity_of_dishes)
15   # expect(list.key).to eq(key) key in the hash - this is imaginative
16 def add_dish( dish_name, quantity )
17   @dish_name = dish_name
18   @quantity = quantity
19   @receipt[@dish_name]
20   # @receipt[@dish_name] = quantity
21   #puts "You ordered #{@quantity} of #{@dish_name}"
22 end
23
24 # def add_dish(dish_name)
```

```
contains an receipt of ordered dishes
responds to method #show_receipt
shows receipt
responds to the method #add_dish
adds a new dish to the receipt (FAILED - 1)

Have you considered running rubocop? It will help you improve
your code!
Try it now! Just run: rubocop

Failures:

1) List adds a new dish to the receipt
   Failure/Error: expect(list.add_dish(@dish_name, @quantity
)).to eq(list.recipe)
     expected: {"pierogi"=>2}
     got: 2

(compared using ==)
# ./spec/list_spec.rb:51:in 'block (2 levels) in <top (re
quired)>'

Finished in 0.07862 seconds (files took 2.06 seconds to load)
8 examples, 1 failure

Failed examples:

rspec ./spec/list_spec.rb:47 # List adds a new dish to the rec
eipt

COVERAGE: 100.00% -- 41/41 lines in 2 files

Aleksandras-MacBook-Air:takeaway-challenge Aleksandra$ irb
2.5.0 :001 > require './lib/list.rb'
=> true
2.5.0 :002 > list.add_dish("pierogi", 3)
Traceback (most recent call last):
  2: from /Users/Aleksandra/.rvm/rubies/ruby-2.5.0/bin/ir
rb:11:in 'main'
  1: from (irb):2
NameError (undefined local variable or method 'list' for main:
Object)
2.5.0 :003 > list = List.new
=> #<List:0x00007fb87106e808 @menu={"chinese"=>10, "pierogi"=
>8, "curry"=>9, "carbonara"=>11, "salad"=>6}, @receipt={}>
2.5.0 :004 > list.add_dish("pierogi", 3)
=> nil
2.5.0 :005 > list.recipe
=> {}
2.5.0 :006 >
```

This one took some time and some proper “fight” or perhaps I should say “vigorous exploration. NOW - here is the solution moved one step further:

```

list_spec.rb
25 list = List.new
26 expect(list.recipe).to eq({})
27 end
28
29 it "responds to method #show_recipe" do
30 list = List.new
31 expect(list).to respond_to(:show_recipe)
32 end
33
34 it "shows recipe" do
35 list = List.new
36 expect(list.show_recipe).to eq(list.recipe)
37 end
38
39 #add_dish
40
41 it "responds to the method #add_dish" do
42 list = List.new
43 expect(list).to respond_to(:add_dish).with(2)
44 end
45
46
47 it "adds a new dish to the receipt" do
48 list = List.new
49 dish_name = "pierogi"
50 quantity = 2
51 #expect(list.add_dish(dish_name, quantity)).to
52 expect(list.add_dish(dish_name, quantity)).to
53 end
54 end
55
56 #it "takes a dish and adds it to the receipt" do
57 # list = List.new
58 # list.show_menu =
59 #
60 #
61 # end
62
list.rb
2 attr_reader :menu, :recipe, :dish_name, :quantity
3
4
5 def initialize
6 @menu = { "chinese" => 10, "pierogi" => 8, "c
7 @recipe = {}
8 end
9
10 def show_menu
11 @menu
12 end
13
14 #def add_dish(dish_name, quantity_of_dishes)
15 # expect(list.key).to eq(key) key in the hash
16 def add_dish( dish_name, quantity )
17 @dish_name = dish_name
18 @quantity = quantity
19 @recipe[@dish_name] = quantity
20 @recipe
21 "You just have added to your order #{@quantity
22 #puts "You ordered #{@quantity} of #{@dish_
23 end
24
25 # def add_dish(dish_name)
26 # @dish_name = dish_name
27 # @recipe << @dish_name
28 # end
29
30
31 # def add_dish(@menu[dish_name])
32 # receipt[@menu[dish_name]
33 # end
34
35 def show_recipe
36 @recipe
37 end
38 end
39

```

The green looks good.

So what were the issues I needed to work on?

1. The method `add_dish` wasn't returning the hash. It was returning the value of the hash. I wanted to test the successful addition to the receipt (should be receipt - I clearly misspelled the name of this variable and I am going to change it in both documents, for now it is `recipe`). I wanted to the outcome of the method `.add_dish` to be equal to my receipt, that is the hash container for the ordered dishes. And now when I think about this... that might not work when I have more dishes added. Or would it? Let's check in `irb`. —> Yes it would. Because `recipe` changes too after the first addition, so later the base for the second addition is the same. So, it works. I copied the code below:

```

xit "adds a new dish to the receipt" do #I keep this test as temporarily pending for future learning
  purposes. It is not to be run in the code at any stage.
    list = List.new
    dish_name = "pierogi"
    quantity = 2
    expect(list.add_dish(dish_name, quantity)).to eq(list.recipe)

    dish_name = "pierogi2" #here I check what would happen if two dishes were added
    quantity = 4
    expect(list.add_dish(dish_name, quantity)).to eq(list.recipe)
    #it is ok, because the receipt also changes after the first dish was added so the test passes
  end

```

- ☒ The test above has been commented - I was not able to keep it `xit`-ed because the test coverage was <90%
- ☒ found and replaced the "recipe" to "receipt" - yes, importance of good names. <https://atom.io/packages/find-and-replace>

2. Even on the earlier step I needed to reflect on the nature of hashes. I always need a pair key-value, if I would like to store a single value without assigned key, I should do it with the use of an array.

☒ I went on the exploration outside of the scope of the assignment - **I know that I should not have done that as we should always stick to the specs**. I wanted to check how to forbid the customer to order the dish that is not in the menu. Something what in my imagination was easy if statement turned out to be MUCH more complicated if one wants to have 100% test coverage. But here it is. I wrote a test for a case that is true, and one that is false - that worked. I learnt.

```

72
73 it "responds to #check_if_in_the_menu" do
74   list = List.new
75   expect(list).to respond_to(:check_if_in_the_menu)
76 end
77
78 it "checks if it is possible to place an order" do
79   list = List.new
80   @dish_name = "pierogi"
81   expect(list.check_if_in_the_menu(@dish_name)).to eq(true)
82 end
83
84 it "checks if it is NOT possible to place an order" do
85   list = List.new
86   @dish_name = "kotlety"
87   expect(list.check_if_in_the_menu(@dish_name)).to eq(false)
88 end
89 # it "raises error when the dish is not listed in the menu" do
90 #   list = List.new
91
92   list.rb      Rhys_code.rb      Eds_code.rb      Rhys_code2.rb
93
94   14 #def add_dish(dish_name, quantity_of_dishes)
95   15 # expect(list.key).to eq(key) key in the hash - this is imaginative
96   16 def add_dish(dish_name, quantity)
97   17   @dish_name = dish_name
98   18   @quantity = quantity
99   19   if check_if_in_the_menu(@dish_name) == false
100   20     return "This dish is not in the menu"
101   21   end
102   22   @receipt[@dish_name] = quantity
103   23   @receipt
104   24   "You just have added to your order #{quantity} of #{@dish_name}(s)"
105   25 end
106
107   27 def check_if_in_the_menu(dish_name)
108   28   @dish_name = dish_name
109   29   if @menu.key?(@dish_name) == false
110   30     false

```

Failures:

```

1) List tries to add a dish that is not in the menu
   Failure/Error: expect(list.add_dish(dish_name, quantity))
   .to eq("This dish is not in the menu")
     expected: "This dish is not in the menu"
     got: nil

   (compared using ==)
   # ./spec/list_spec.rb:70:in `block (2 levels) in <top (re
   quired)>'

Finished in 0.11969 seconds (files took 2.12 seconds to load)
12 examples, 1 failure

Failed examples:

rspec ./spec/list_spec.rb:66 # List tries to add a dish that
is not in the menu

COVERAGE: 100.00% -- 66/66 lines in 2 files

Aleksandras-MacBook-Air:takeaway-challenge Aleksandra$ rspec
list
  contains a menu
  responds to the method #show_menu
  shows the menu
  contains an receipt of ordered dishes
  responds to method #show_receipt
  shows receipt
  responds to the method #add_dish
  adds a new dish to the receipt and prints the summary of las
  t order
  tries to add a dish that is not in the menu
  responds to #check_if_in_the_menu
  checks if it is possible to place an order
  checks if it is NOT possible to place an order

Have you considered running rubocop? It will help you improve
your code!
Try it now! Just run: rubocop

Finished in 0.01365 seconds (files took 2.08 seconds to load)
12 examples, 0 failures

COVERAGE: 100.00% -- 66/66 lines in 2 files

Aleksandras-MacBook-Air:takeaway-challenge Aleksandra$

```

☒ Now, back to the user stories

As a customer

So that I can verify that my order is correct

I would like to check that the total I have been given matches the sum of the various dishes in my order

☒ I need to sum the values and record the sum

☒ side note on raising errors for this point

```

# it "raises error when the dish is not listed in the menu" do
#   list = List.new
#   dish
#   expect(list.add_dish('jjjj', 1)).to raise_exception
# end

```