Course Day 13, June 9, SUN

Weekend challenge - continuation - Sunday morning

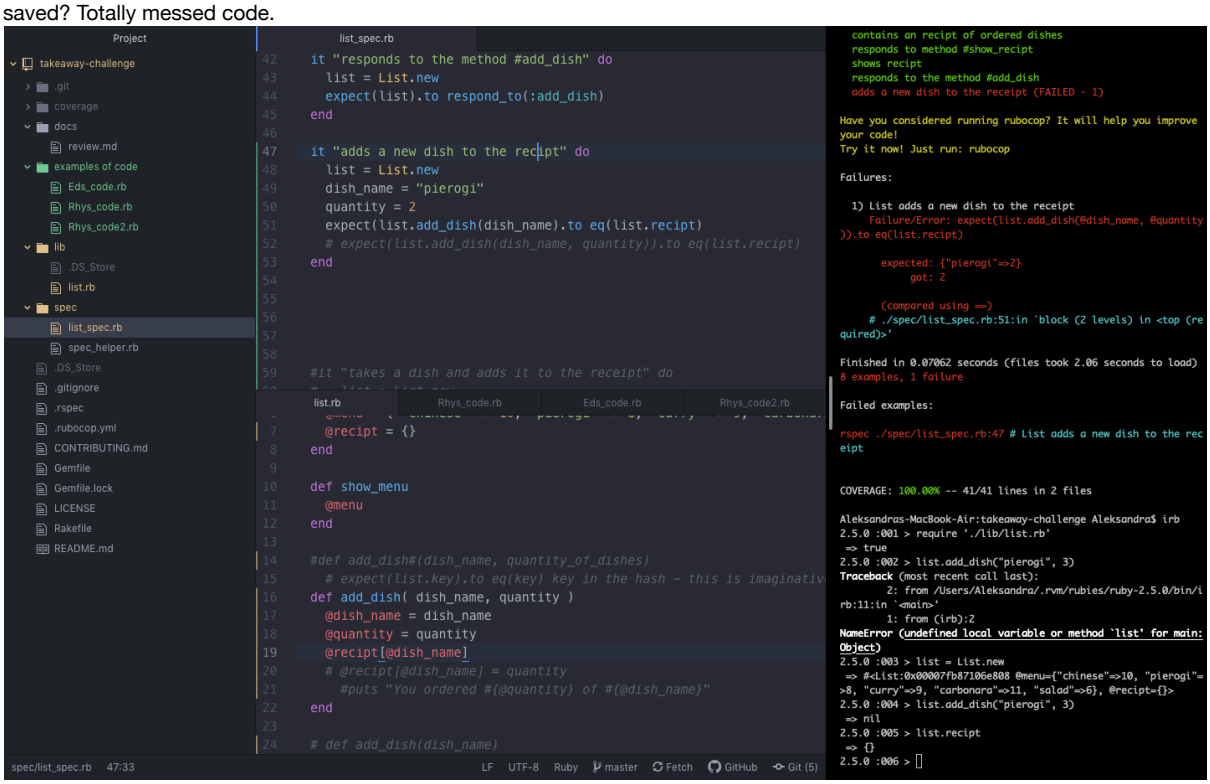
- [x] Working on the second user story

```
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
```

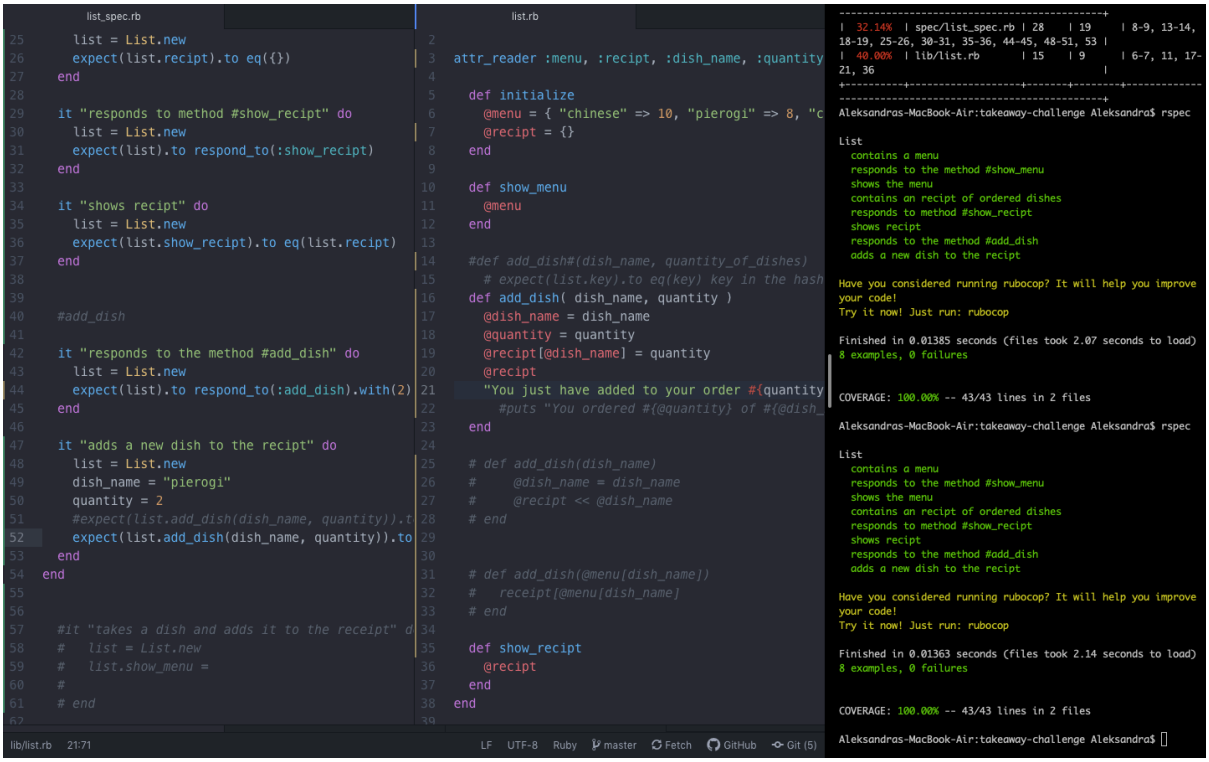
- [x] reflect on how to move the dish to a receipt & how to test it
- [x] review my notes from review of hashes

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

- [x] 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.



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:



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 recipt (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 recipt). I wanted to the outcome of the method .add_dish to be equal to my recipt, 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 recipt 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:

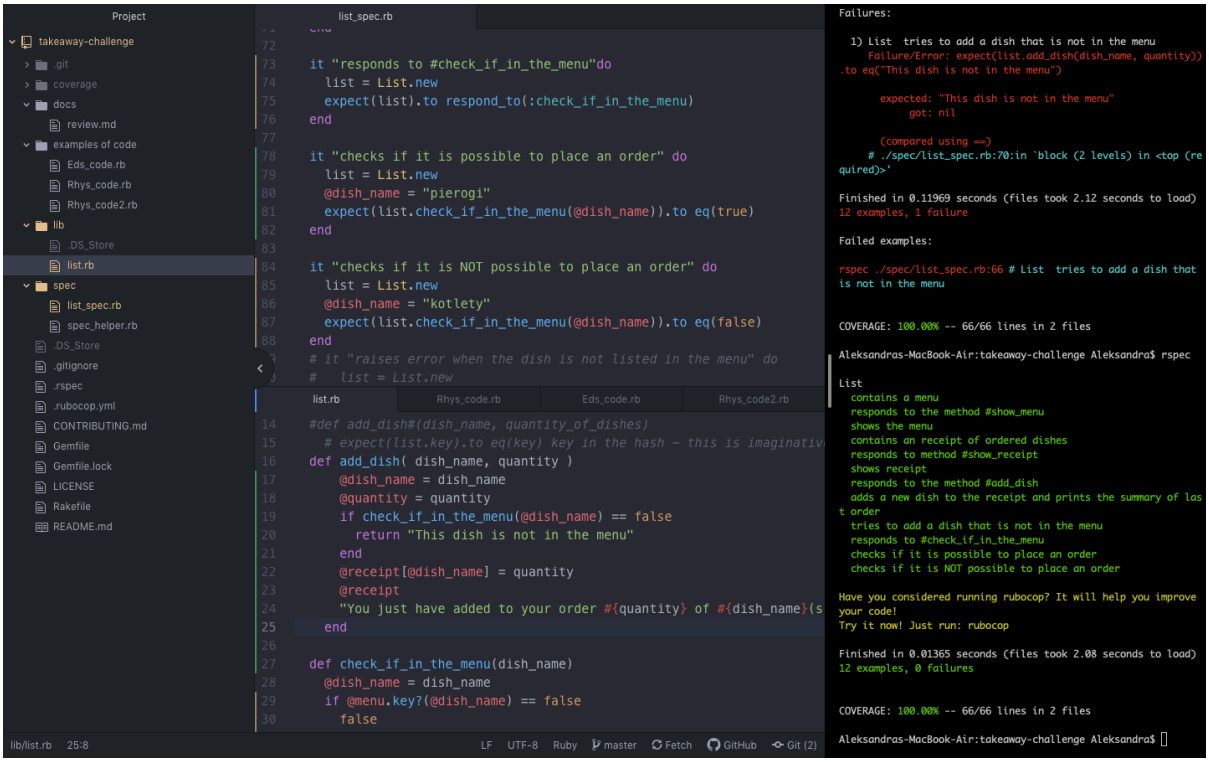
```
  xit "adds a new dish to the recipt" do #I keep this test as temporarly pending for future learning
purposes. It is not to be run in the code at any stage.
    list = List.new
    dish_name = "pierogi"
    quantity = 2
    expect(list.add_dish(dish_name, quantity)).to eq(list.recipt)


    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.recipt)
    #it is ok, because the recipt also changes after the first dish was added so the test passes
  end
```

- [x] The test above has been commented - I was not able to keep it xit-ed because the test coverage was <90%
- [x] found and replaced the "recipt" 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.

- [x] 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.



- [x] 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
```

- [x] I need to sum the values and record the sum
    - [x] 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
```