CSSE376 - Lab9 - CaioBalthazar - CM1478

[Question 1 (5 pts)] What is the return value of the following Ruby expressions?

{ [].class

Array

{ 3 / 2

1

{ 3.0 / 2.0

1.5

{ [].nil?

false

{ def h; "Hello world"; end

nil

[Question 2 (5 pts)] What is your personal favorite feature of Ruby? Why?

Dynamic typing and blocks. These make programming a lot more productive and easier most of the time, if used with some care.

[Question 3 (5 pts)] In your opinion, are the advantages of BDD worth the drawbacks (at least in theory)? Why or why not?

It depends. There are scenarios in which client interaction and feedback during development is a very crucial thing, and in these cases BDD presents itself very worth. In other cases, when the system is already very well defined without the user involvement and unit testing is doing its job, BDD might present more overhead than benefits. As most of the techniques for software, you have to know when to properly apply.

[Question 4 (5 pts)] Now that you have used BDD in a development process, do you think it helps

or hinders developers?

I think BDD is more useful for the non-technical staff and clients than it is for the developers mainly for the fact that it takes way too much time. It may help developers to understand an unfamiliar or still undefined system, but there are faster and better ways to do that. I believe BDD is technique to be used mostly when you need to interact heavily with the client during development.

[Question 5 (5 pts)] What is the advantage of using more verbose, complex English in your feature

definitions? What is the drawback?

It helps you describe the high-level features even if you don't know the details yet. It also communicates with the non-technical stakeholders what the application is really doing. Finally, it helps unfamiliar developers to better understand the code.

The drawbacks are time and operation overhead, writing DSLs and one more software layer that can contain errors and throw false positives/negatives.