Ryne Bell

CSSE376

Lab09

1.

```
[].class
=> Array
3/2
=> 1
3.0/2.0
=> 1.5
[].nil?
=> false
def h; "Hello world"; end
=> nil
```

2. I like the object orientedness of it. Everything being a class, and having gets and puts (and other functions) work in any way that makes sense is nice. ie. Math.sqrt doesn’t care if the input is an integer or a Float, and it is easy to accept any input into functions I write, then check what class the object passed in was, then use that to decide what to do.

3. The answer to this question really depends on the problem being solved, in some cases such as buiding a video game, having a Domain-Specific language to program AI, and/or physics might make sense, but in general, I don't think using BDD is worth the drawbacks. There seems to be too much overhead, both in the design, and implementation of the project when using this design strategy.

4. I think BDD helps in a limited number of ways. Having to modify tests every time something in the GUI changes is annoying, but assuming there is a well defined sequence of menus from the beginning, this shouldn't be a problem. It has a limited overall usefulness though, as it can only test the overall

functionality of the program, not the underlying methods, also, the test cases tend to get unnecessarily wordy.

5. The advantage of using more verbose complex English in feature definitions is that anyone can understand what is being tested by a particular test case. The disadvantage of using DBB is that test cases become unnecessarily wordy, and this can cause a lower number of tests than may have been otherwise written to be implemented.