CIT 597 – Homework 3

Due – Dec 4, 2014 at 12.00pm

Part 1 – Theory (25 points)

1. Project 2.6 from the class textbook (Fox and Patterson) (5 points)
2. Read Joel Spolsky's "The Law of Leaky Abstractions"
   http://www.joelonsoftware.com/articles/LeakyAbstractions.html
   Write a 1-page (normal margins, normal font sizes, single line spacing) response to the above. Some things to think about while writing the response: (10 points)
   a. The article does not specifically talk about MVC. Do the same thoughts apply here? If yes, why? If no, why not?
   b. Are there Leaky Abstractions in the Ruby and Rails API?
   c. Do you agree with the article? If yes, why? If not, why not? Etc.
3. Read Steve Yegge's "Execution in the Kingdom of Nouns"
   http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html
   Write a 1-page (normal margins, normal font sizes, single line spacing) response to the above. Some things to think about while writing the response: (10 points)
   a. How does this fit it with what we've seen about Ruby/Rails and your previous experience programming in other languages?
   b. Do you agree with the article? If yes, why? If not, why not? Etc.

Part 2 – Unit Testing HW2 Java to Ruby Code (35 points)

For this part, you will conduct unit testing on your HW2 Ruby code for Java to Ruby.

You must use MiniTest for this part. For every method in your Ruby file, write at least 3 unit tests. Overall, you must have at least 10 tests for your entire program.

Part 3 – Unit Testing HW2 Jeopardy Code (40 points)

For this part, you will conduct unit testing on your HW2 Ruby code for Jeopardy.

You must use RSpec and SimpleCov for this part. Please do the following:

1. Start by describing the behavior of your Jeopardy code. Keep in mind that is unit testing, so focus on the behavior of small chunks of code, rather than the overall program.
2. Add in SimpleCov to track your coverage.
3. Add behavior and appropriate tests for you code.
4. The goal is to get at least 90% code coverage – add in as many tests as you need to achieve this. If it's not possible to achieve this, please explain why in your readme.txt file.

Note: For both the parts above, if you find any bugs as a result on the testing, you don't need to fix it.

Part 4 – Extra Credit (20 points)

One key aspect of good tests is that they should be **Fast**. For the normal credit, you don't need to worry about this.

The big bottleneck in part 3 is talking to the internet. Your tests will only run when you're connected to the Internet and will still be dependent on the speed of your connection.

RSpec (and other testing frameworks) provide the notion of "Mocks" where you can mock out the functionality of certain things (like the Internet, in this case) to help with testing.

Your task for the EC is to read about Mocks, and update your tests from Part 3 to use Mocks instead. The goal is for your tests to run (and still achieve at least 90% code coverage) without being connected to the Internet. There are many different mocking frameworks available – you can use whichever one you like.

For the EC part, you cannot have any help from the TAs/instructor.

Grading Criteria

90% for testing – Do you have the required number of tests? Are the tests well structured? Do they follow the FIRST principles?
10% for design and style – Test are still code - Is your test code well designed? Do you have good comments? Are your variables named appropriately?

Programming – General Comments

Here are some guidelines wrt programming style for full credit.

Please follow the Ruby style as far as possible. Here's a good coding style guide (with tools) to check your programs.

https://github.com/bbatsov/ruby-style-guide

Submission Instructions

The code should be submitted electronically. Please **do not** print it out.

In addition to the Ruby files, you should also submit a text file titled readme.txt, which should contain a short write-up about your tests – why did you choose these tests over (possibly infinite) others? Did you experience any problems? The readme should also include the answers for Part 1.

Please create a folder called YOUR_PENNKEY. Places all your files inside this – the ruby files, the readme.txt file. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.