Chris Sipola (s1667278)

IJP Assignment 1

[11] Programming experience

I don't have a degree in computer science (it's in economics). However, I took several online computer science courses that required the use of Python. I also programmed professionally in SAS for 1.5 years when I worked in analytics for the US government, and in R for 2 years when I worked as a data scientist in the tech industry.

[12] Tasks for which I submitted code

a. The controller: [2], [3].
    i. Note that the code for [1] was removed in favor of the updated methodology in [3].
b. The cache proxy: [4], [5]
c. The tests: [6], [7], [8]

[13] Questions

1. **Clearly explain any errors that you found in tasks [8] or [9].** For [8], three of the tests failed, but the one that points most directly to the error is `nonequalityTest()`, which printed the message "same picture returned for different subject." The issue is likely that all pictures requests after the first picture simply return the first cached picture. For example, if requesting three different pictures, the cache proxy would return the first (cached) picture for the second and third requests. (We know caching is working correctly because `equalityTest()` passed.)
2. **Explain why `MyController` is hard to test in the same way that you tested the `CacheProxy` (i.e. without relying on external services or views). Suggest how you might make a small modification to the controller class to make this easier.** `CacheProxy` (or `MyCacheProxy`) is easy to test because the unit tests in `MyCacheProxyTest` use the in-class method `getPicture`, which does not require a service request. This lets you be sure that any errors in `CacheProxy` are a problem with the class rather than with the service (say, if the service is down or has a bug). To make `MyController` easier to test, we can use the same "trick" used by

MyCacheProxyTest: create a getPicture method within the unit tests for MyController and have the service take the this object as an argument.

3. **Explain why it is difficult to compose the RandomProxy and the CacheProxy.** Similar to the previous answer, RandomProxy and CacheProxy require that you rely on an outside services. For RandomProxy, you have to rely on *multiple* outside services. As an additional difficulty, CacheProxy must rely on another proxy service (SlowProxy) to test its effect on functionality. (Also, they may be considered “difficult” to compose in that much of the code for the two classes has been repeated.)
4. **Suggest how you might change the way in which the resources are used to make this composition possible.** You can define the base service in the my.properties file to point to a local service, i.e., one that is reliable and does not require a request from something external and potentially buggy/unreliable. Or you can point to a “fake” base service that uses something similar to the getPicture “trick” used in MyCacheProxyTest.