CSCI E-97 Software Design: Principles, Models, and Patterns – Fall 2014
Assignment #5 – Results
Roland Galibert
December 19, 2014

Following is a summary of my overall experience in assignment #5, the final project for this course, designing and implementing the Drone Package Delivery System consisting of a number of different services.

Design Patterns

I was fortunately able to apply a number of design patterns to my system. These included:
Façade and Singleton – used to implement APIs
Singleton – For the all-encompassing service classes which implemented the corresponding APIs and private methods required for the service
Factory – I have a DroneTypeFactory for creating DroneTypes
Flyweight – The DroneTypeFactory mentioned above as well as a number of enum classes
Observer – I implemented Observer for communication between active drones (drones not in a hanger and out on delivery) and my DroneManager service (drones in my system actually communicate with the DroneManager and not directly with the ACS; the DroneManager acts as a mediator). Drones are the subject, and the DroneManager attaches itself as an Observer when the Drone is sent out on delivery, and detaches itself when the Drone returns to the hanger.
Strategy – As Professor Giesecke suggested on the discussion board, I managed to implement the routing algorithm as a strategy (RoutingStrategy interface with one implementation, DefaultRoutingStrategy) so that the ACS could switch easily between different routing strategies.
Command – As Professor Giesecke suggested on the discussion board, I managed to implement my ScheduledDelivery object as a command, although, due to reasons of time, it wasn't the cleanest implementation of the Command pattern. In addition, the execute() method is not always called, only in the case where the assigned drone assigned to the delivery is docked in a hanger (if the assigned drone is completing another delivery, the new delivery is initiated when the drone reports it has completed the previous delivery).
Proxy – My Drone objects are handled somewhat like proxies, though again, because of time, it is actually not a strict implementation of the Proxy pattern. It was actually somewhat later this week that I realized the difference between an actual drone and its representation in my DroneManager object.

Modular Approach

The modular approach was a very big help in implementing my design. Especially with the discussion in the previous assignment regarding the intent of the authentication service and the lecture on service interaction and level 5 modularity, I got a very good idea of what was expected in implementing a modular service.

Also, after I realized that an ideal implementation would be to just have services send messages with status/command updates, my APIs turned out to be fairly small, just a sendMessage() method and relatively few other public methods.