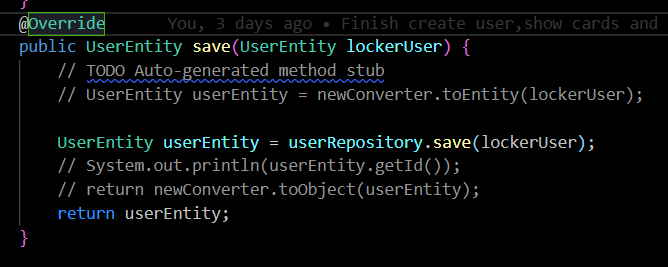
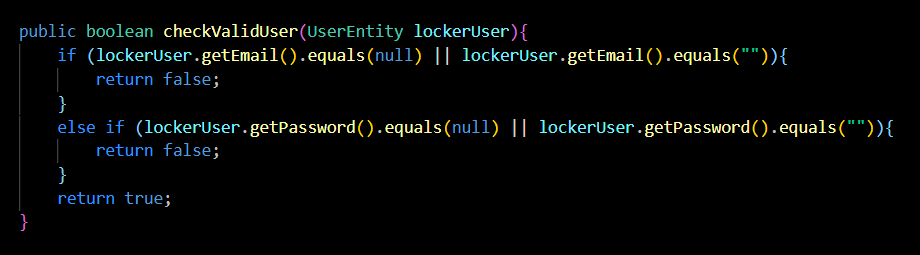
# Unit Testing

### Service Layer Unit Testing:

* Save() method

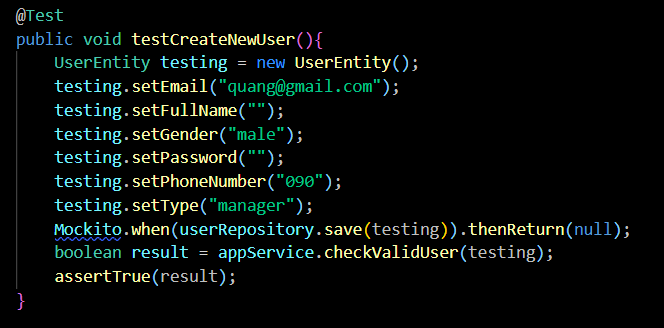


From my perspective, I think we should check the validity of the data when register a new account. So I create another method checkValidUser():

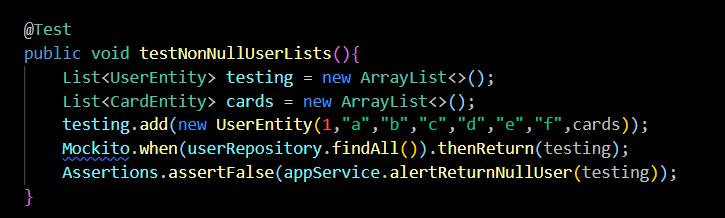


So here are some steps for creating a unit test in this case:

* Create a sample object with data
* Use Mockito to interrupt when the function call in the app service method (in this case the method in the service layer calls to the repository layer)
* After that we assert the result of for the test case.

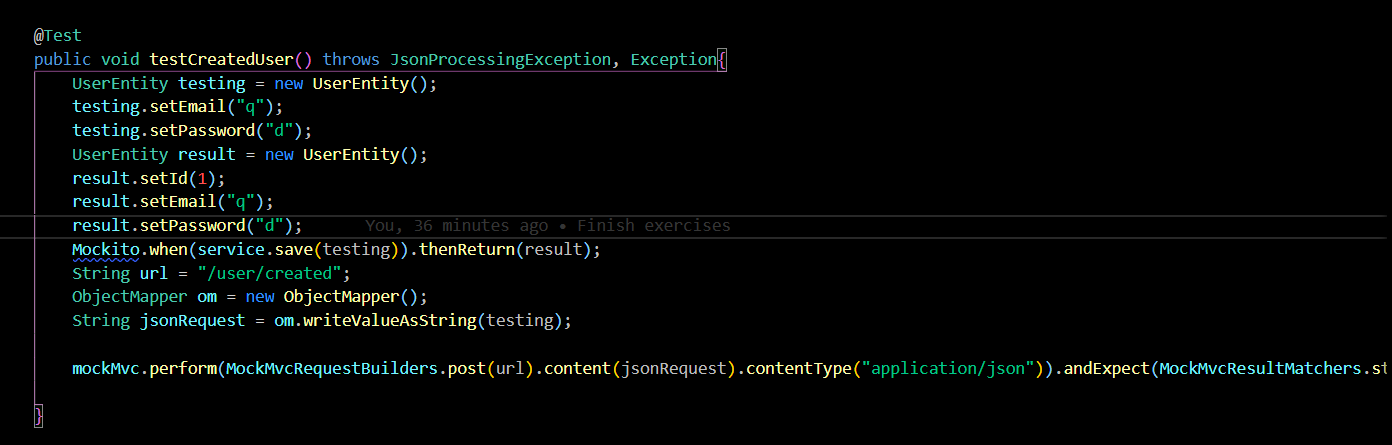


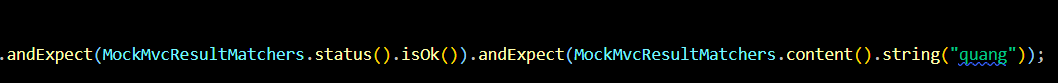
* Get() method



The strategy is the same as above but the function call is different due to different feature of the method.

1. Controller Layer Unit Testing:





I think this is the most complicated section which consumes most of my time. I imagine in this case we have 2 identical object but I assign to 2 different variables. The first variable receive the data as an input and the second one as an expected output. My job in this method is to make sure that the 2 value matched when the mockMvc.perform() executed. So in the mockMvc.perform, we have:

* Identify post,put,get or delete method of restful API
* The content of the input json (jsonRequest)
* The expected status response
* Check if the input pattern is also included in the expected output pattern