Concurrency Test - PASSED
Setup: all 3 servers up and running, chmod permissions set, run `./test_concurrent.sh`
Goal: Verify that concurrent buy requests lead to correct buy/fail messages from the client, and that no clients make illegal stock decrements.
Description: This test spawns 5 clients which each concurrently make 100 buy requests to item 1. The stock of item 1 is initialized to 300. Periodic stock update is turned off. To verify that our system can handle concurrent requests, the script does a `grep` command to count how many 'failed' and 'bought' lines were printed by clients. With 500 buys and 300 initial stock, this program works correctly if 200 'fail' lines are printed, 300 'bought' lines are printed, and the ending stock is 0. The test output is below.

```
Setting stock to 300, 5 clients concurrently making 100 buys each
{"How to get a good grade in 677 in 20 minutes a
day":{"COST":120.0,"QUANTITY":300,"SUCCESS":true}}
Average runtime is 0.167263169289
Sequential buy results written to
../test/experiment_results/3conctestbuy_1_100
Average runtime is 0.168271756172
Sequential buy results written to
../test/experiment_results/1conctestbuy_1_100
Average runtime is 0.168353865147
Sequential buy results written to
../test/experiment_results/5conctestbuy_1_100
Average runtime is 0.168610415459
Sequential buy results written to
../test/experiment_results/2conctestbuy_1_100
Average runtime is 0.168879029751
Sequential buy results written to
../test/experiment_results/4conctestbuy_1_100
200 buys failed and 300 buys succeeded

Name: How to get a good grade in 677 in 20 minutes a day
Cost: 120.0
Quantity: 0
```

From the highlighted portion above, we note that the test is successful.

Catalog Server Unit Tests -PASSED

Setup: all 3 servers up and running. Run `python3 TestCatalog.py`. Please note to use python3, otherwise python2 will have library error.

Goal: verify that catalog server functionalities are working

Description:

1. Query by topic test: Assert that querying by topic returns the correct dictionaries
2. Query by item test: Assert that querying by item returns cost and quantity fields, and that the field types are float and int respectively.
3. Update test: Assert that increase operations result in correct value increases, decrease operations result in correct value decreases, and set operations result in correct value set.

```
Query message by item, passed
Return types are correct
.Query message by topic, passed
.Increase operation, passed
Decrease operation, passed
Set operation, passed
.
----------------------------------------------------------------------
-----
Ran 3 tests in 0.083s

OK
```

Order Server Unit Tests -PASSED

Setup: all 3 servers up and running. Run `python3 TestOrder.py`. Please note to use python3, otherwise python2 will have library error.

Goal: verify that order server functionalities are working

Description:

1. Buy item title test: Assert that buy operation fetches the correct book titles
2. Buy item return type test: Assert that the the buy operation returns a dictionary with the correct keys and correct value types
3. Buy success test: Assert that a single client can successfully buy with positive stock, and unsuccessfully buy with 0 stock.

```
Buy fetches correct title, passed
.Successful buy passed
Unsuccessful buy passed
.Buy return fields, passed
Return types are correct
.
----------------------------------------------------------------------
Ran 3 tests in 0.212s

OK
```