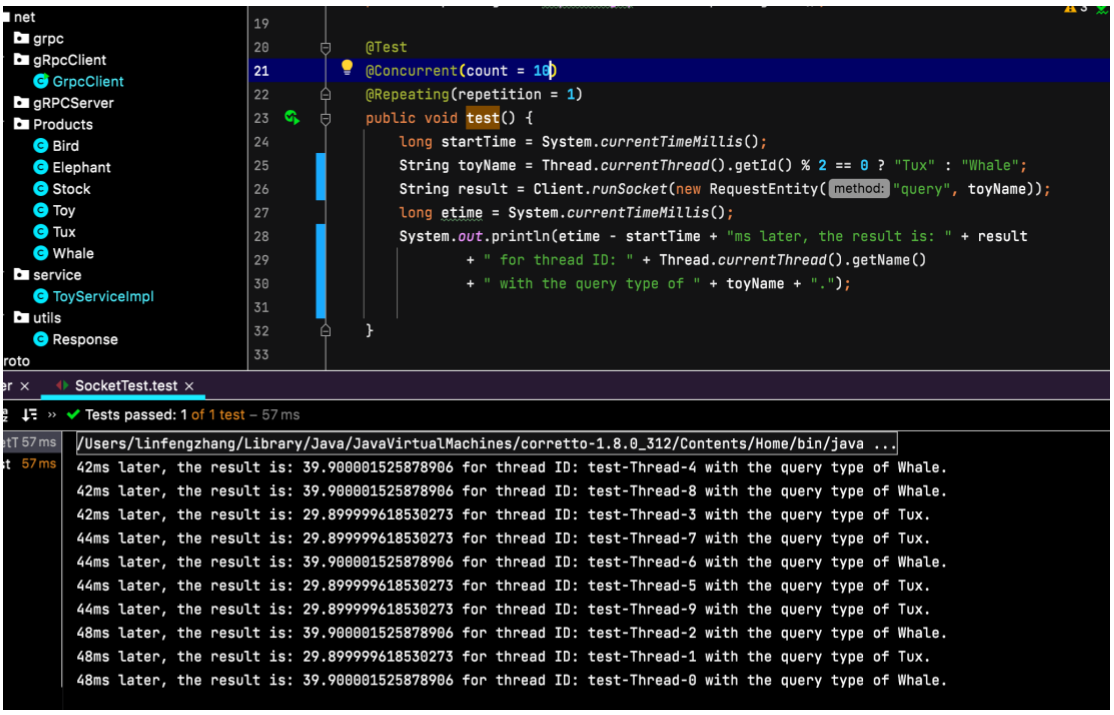
System Outputs

We have written concurrent testing part as showed in our system design, and it will part an important role in this document and the measurements part.

For part1, the requirements don't contain any concurrent issues, so here we just run 10 threads to check out the result. The price is given by the server when stock number is set positive.

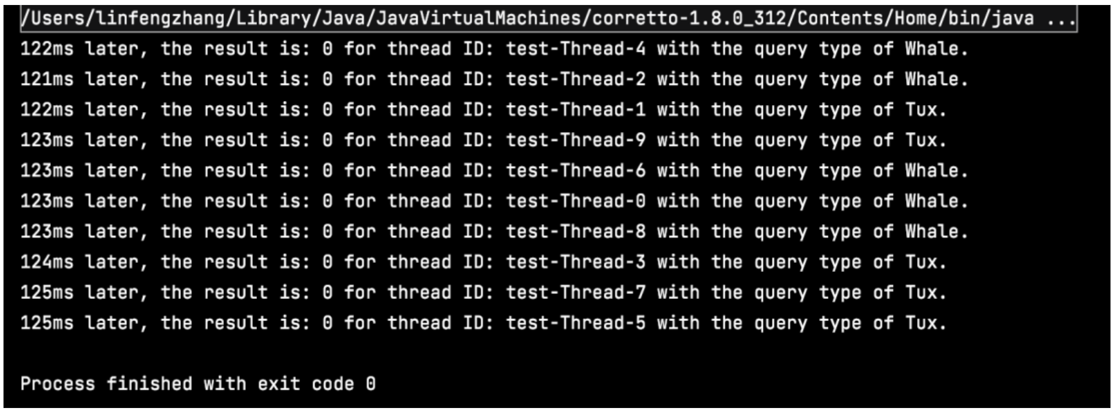


```java
    @Test
    @Concurrent(count = 10)
    @Repeating(repetition = 1)
    public void test() {
        long startTime = System.currentTimeMillis();
        String toyName = Thread.currentThread().getId() % 2 == 0 ? "Tux" : "Whale";
        String result = Client.runSocket(new RequestEntity( method: "query", toyName));
        long etime = System.currentTimeMillis();
        System.out.println(etime - startTime + "ms later, the result is: " + result
                + " for thread ID: " + Thread.currentThread().getName()
                + " with the query type of " + toyName + ".");

    }
```

```
/Users/linfengzhang/Library/Java/JavaVirtualMachines/corretto-1.8.0_312/Contents/Home/bin/java ...
42ms later, the result is: 39.900001525878906 for thread ID: test-Thread-4 with the query type of Whale.
42ms later, the result is: 39.900001525878906 for thread ID: test-Thread-8 with the query type of Whale.
42ms later, the result is: 29.899999618530273 for thread ID: test-Thread-3 with the query type of Tux.
44ms later, the result is: 29.899999618530273 for thread ID: test-Thread-7 with the query type of Tux.
44ms later, the result is: 39.900001525878906 for thread ID: test-Thread-6 with the query type of Whale.
44ms later, the result is: 29.899999618530273 for thread ID: test-Thread-5 with the query type of Tux.
44ms later, the result is: 29.899999618530273 for thread ID: test-Thread-9 with the query type of Tux.
48ms later, the result is: 39.900001525878906 for thread ID: test-Thread-2 with the query type of Whale.
48ms later, the result is: 29.899999618530273 for thread ID: test-Thread-1 with the query type of Tux.
48ms later, the result is: 39.900001525878906 for thread ID: test-Thread-0 with the query type of Whale.
```

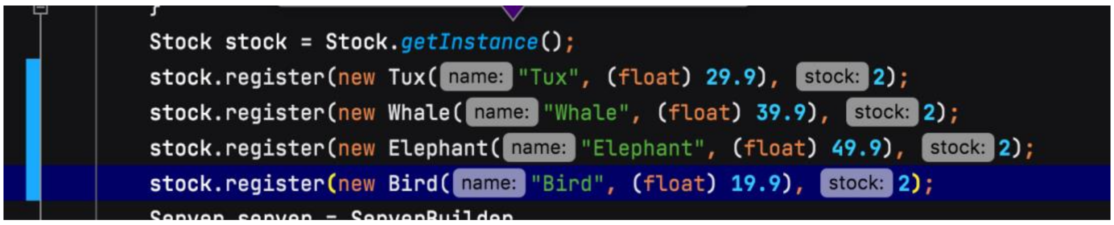
After setting the stock in Server.class to be 0, the result returned thereby becomes 0.



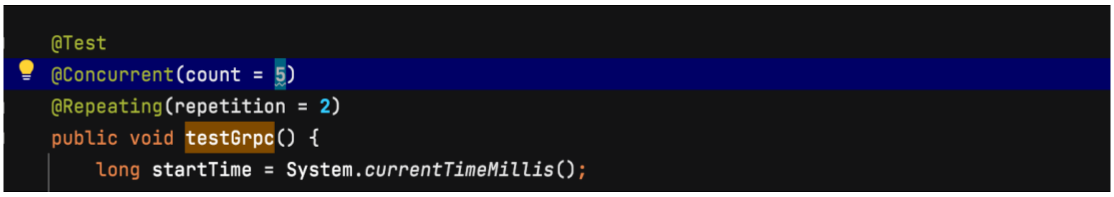
```
/Users/linfengzhang/Library/Java/JavaVirtualMachines/corretto-1.8.0_312/Contents/Home/bin/java ...
122ms later, the result is: 0 for thread ID: test-Thread-4 with the query type of Whale.
121ms later, the result is: 0 for thread ID: test-Thread-2 with the query type of Whale.
122ms later, the result is: 0 for thread ID: test-Thread-1 with the query type of Tux.
123ms later, the result is: 0 for thread ID: test-Thread-9 with the query type of Tux.
123ms later, the result is: 0 for thread ID: test-Thread-6 with the query type of Whale.
123ms later, the result is: 0 for thread ID: test-Thread-0 with the query type of Whale.
123ms later, the result is: 0 for thread ID: test-Thread-8 with the query type of Whale.
124ms later, the result is: 0 for thread ID: test-Thread-3 with the query type of Tux.
125ms later, the result is: 0 for thread ID: test-Thread-7 with the query type of Tux.
125ms later, the result is: 0 for thread ID: test-Thread-5 with the query type of Tux.

Process finished with exit code 0
```

For part2, we set all the stocks for four toys to be 2, running 10 threads to send randomly quests twice to make a clear correctness explanation.

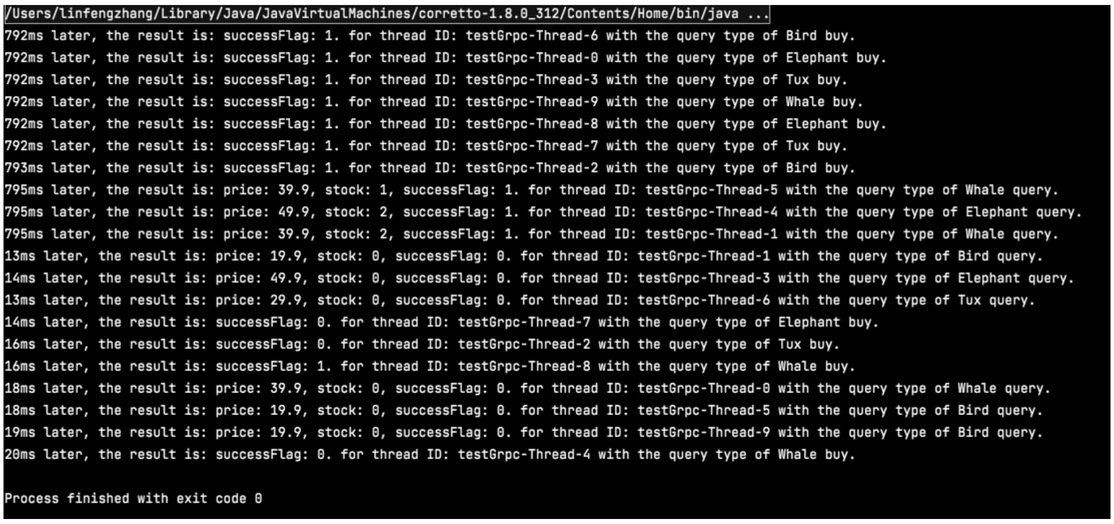


```java
Stock stock = Stock.getInstance();
stock.register(new Tux(name: "Tux", (float) 29.9), stock: 2);
stock.register(new Whale(name: "Whale", (float) 39.9), stock: 2);
stock.register(new Elephant(name: "Elephant", (float) 49.9), stock: 2);
stock.register(new Bird(name: "Bird", (float) 19.9), stock: 2);
Server server = ServerBuilder
```



```java
@Test
@Concurrent(count = 5)
@Repeating(repetition = 2)
public void testGrpc() {
    long startTime = System.currentTimeMillis();
```

The result returns out to be:



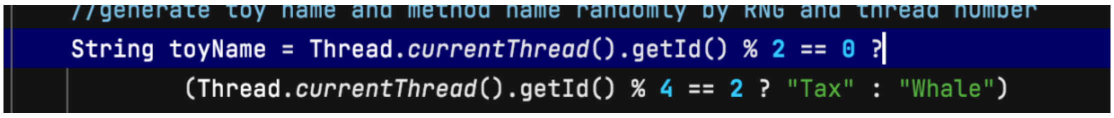
```
/Users/linfengzhang/Library/Java/JavaVirtualMachines/corretto-1.8.0_312/Contents/Home/bin/java ...
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-6 with the query type of Bird buy.
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-0 with the query type of Elephant buy.
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-3 with the query type of Tux buy.
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-9 with the query type of Whale buy.
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-8 with the query type of Elephant buy.
792ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-7 with the query type of Tux buy.
793ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-2 with the query type of Bird buy.
795ms later, the result is: price: 39.9, stock: 1, successFlag: 1. for thread ID: testGrpc-Thread-5 with the query type of Whale query.
795ms later, the result is: price: 49.9, stock: 2, successFlag: 1. for thread ID: testGrpc-Thread-4 with the query type of Elephant query.
795ms later, the result is: price: 39.9, stock: 2, successFlag: 1. for thread ID: testGrpc-Thread-1 with the query type of Whale query.
13ms later, the result is: price: 19.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-1 with the query type of Bird query.
14ms later, the result is: price: 49.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-3 with the query type of Elephant query.
13ms later, the result is: price: 29.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-6 with the query type of Tux query.
14ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-7 with the query type of Elephant buy.
16ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-2 with the query type of Tux buy.
16ms later, the result is: successFlag: 1. for thread ID: testGrpc-Thread-8 with the query type of Whale buy.
18ms later, the result is: price: 39.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-0 with the query type of Whale query.
18ms later, the result is: price: 19.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-5 with the query type of Bird query.
19ms later, the result is: price: 19.9, stock: 0, successFlag: 0. for thread ID: testGrpc-Thread-9 with the query type of Bird query.
20ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-4 with the query type of Whale buy.

Process finished with exit code 0
```

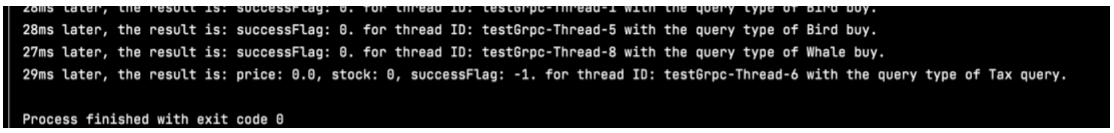
You can notice that the initial buys are all successful with returned flag 1. Note that each thread can have a nonlinear order of printing returning values, so the query result later of Elephant stock can still be 2 even after the buy operation of elephant has been printed. But we can still infer from the buys and queries later that the buy operations for each toy only be carried out twice, which matches our expectation.

You can change the count and repetition number in the annotation on your will to see how the randomly generated operations and our system work, but remember to restart the server if you see the stock is run out because all the data is stored in memory.

We finally tested the situation of queried toy not exist in the stock:



```java
//generate toy name and method name randomly by RNG and thread number
String toyName = Thread.currentThread().getId() % 2 == 0 ?
        (Thread.currentThread().getId() % 4 == 2 ? "Tax" : "Whale")
```



```
28ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-1 with the query type of Bird buy.
28ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-5 with the query type of Bird buy.
27ms later, the result is: successFlag: 0. for thread ID: testGrpc-Thread-8 with the query type of Whale buy.
29ms later, the result is: price: 0.0, stock: 0, successFlag: -1. for thread ID: testGrpc-Thread-6 with the query type of Tax query.

Process finished with exit code 0
```

Seems fine, we don’t sell taxes!