CPD-4414 Assignment #2

Continue Working on the Following Project

Submit the "Quiz" with your GitHub URL before 11:55pm on Saturday, Jan 31st

You have been given the following documents to implement a customer service queue for a nuts and bolts shop: they sell either nuts or bolts.

The major goal of this project is technical: you must make it work.

Using Test-Driven Development (TDD) principles, you need to make effective tests for all of the following behaviours, and then implement the code to pass the tests.

You may continue working as a team, or work solo.

If you are working as a team, you MUST have each team member doing an equal share of the work in terms of lines of code (LOC).

Everyone must submit a link to their own copy of the GitHub repository. The GitHub repo you submit may have submissions from anyone, but at least an equal share of the work submitted must be your work.

New Behaviours (added since Assignment #1) have been highlighted in **bold with yellow background**.

As part of these new behaviours, you will be accessing a MySQL database server. This database server is only accessible on-campus, or through the VMWare Virtual Desktops (https://virtual.lambtoncollege.ca)

Database Credentials

Server Address:	**ipro.lambton.on.ca**
Database:	**inventory**
Table:	**inventory**
Username:	**products**
Password:	**products**

This account only has SELECT privileges on this database.

Starter Files

A set of Starter Files have been provided on D2L. This repository includes some incomplete implementation. You can use it as a reference, or a foundation.

Grading Criteria

Test Effectiveness – Do the tests cover a good example and common edge cases?	**[4]**
Source Control Effectiveness – Are there frequent commits with meaningful messages?	**[2]**
Meets Technical Requirements – Does the finished project work as intended?	**[4]**
Total	**[10]**

Problem Description

Bill Roberts runs FastenIt, a nuts-and-bolts shop that sells nuts and/or bolts to industry in bulk. Right now, his order control system is paper-based and poorly organized. He lost a major client last week because 10% of their orders went unfulfilled, and another 10% were processed late. He wants to make sure orders are dealt with one-by-one, in the order they are received, and that no orders go missing. He also wants to be able to receive a report of past orders including the date they arrived, the date they were processed, the date they were fulfilled (ie- shipped), and a brief summary. The report should include all orders (just-arrived through fulfilled.)

Your Senior Engineer Says

This is a big project, it will involve front-end, back-end, databases, and all sorts of things in-between. You will not be completing this whole project in one go. Your team has been assigned the task of building the order-queuing and reporting system: the "back-end." In the future (in future assignments) you will connect this to a database, and build its front end, but for now we're just building part of the data model. **The inventory system was begun by a co-worker before he got sick. It is now your responsibility to finish integrating the inventory system into your order queue.**

What's In An Order?

- Customer ID
- Customer Name
- Time Received
- Time Processed
- Time Fulfilled
- List of Purchases (Product ID, Quantity)
- Notes

Build (at least) the Following Behaviours

- Given a new order arrives, when either customer ID or customer name exists, and there is a list of purchases, then set the time received to now.
- Given a new order arrives, when neither customer ID nor customer name exists, then throw an exception.
- Given a new order arrives, when there is no list of purchases, then throw an exception.
- Given a request for the next order, when there are orders in the system, then return the order with the earliest time received that does not have a time processed.
- Given a request for the next order, when there are no orders in the system, then return null.
- Given a request to process an order, when the order has a time received **and all of the purchases are in-stock in the inventory table**, then set the time processed to now.
- Given a request to process an order, when the order does not have a time received, then throw an exception.
- Given a request to fulfill an order, when the order has a time processed and a time received **and all of the purchases are in-stock in the inventory table**, then set the time fulfilled to now.

- Given a request to fulfill an order, when the order does not have a time processed, then throw an exception.
- Given a request to fulfill an order, when the order does not have a time received, then throw an exception.
- Given a request for a report, when there are no orders in the system, then return an empty string.
- Given a request for a report, when there are orders in the system, then return a JSON object of the following rough schema:

```
{ “orders” : [
    { “customerId” : “...”,
      “customerName” : “...”,
      “timeReceived” : “2015-01-12T12:25:43.511Z”,
      “timeProcessed” : “2015-01-12T18:16:42.453Z”,
      “timeFulfilled” : “2015-01-13T14:44:44.444Z”,
      “purchases” : [
        { “productId” : 132, “quantity” : 500 }
      ],
      “notes” : “...”
    },
    { “customerId” : “...”,
      “customerName” : “...”,
      “timeReceived” : “2015-01-12T12:25:43.511Z”,
      “timeProcessed” : “”,
      “timeFulfilled” : “”,
      “purchases” : [
        { “productId” : 14, “quantity” : 100 },
        { “productId” : 15, “quantity” : 100 }
      ],
      “notes” : “...”
    }
] }
```

How Do I Submit This?!

There will be a quiz on D2L. It will have one question: What is your GitHub URL?

You must provide the URL to **your** GitHub repo.

When I am grading your work, I will be looking for the version that is closest to, but not beyond, the due date of 11:55pm on Saturday, January 31st. You may continue working on this to complete the described behaviours, but I will only mark the latest version committed and pushed before the due date.

