Ian Parkison
CSE 598
Summer 2018
Assignment 2
IDE Used: Visual Studio 2017 Community

Output from E-commerce chicken application:
The italicized text below is a the output from one price cut sale event and further explanation is below.

Chickens Are On Sale for $9!
Bob's Chickens placed order for 7 chickens @ $9
Order confirmation for Bob's Chickens returned in 00:00:00.0059933

Confirmation for: Bob's Chickens
- *Line Item 7 chickens @ $9*
- *Line Total $63*
- *Tax $5.04*
- *Shipping $10*
- *Total $78.04*

Chickens R Us placed order for 9 chickens @ $9
Credit Card -5580638- is invalid for Chickens R Us.
Turkey Jerky placed order for 13 chickens @ $9
Order confirmation for Turkey Jerky returned in 00:00:00.0009991

Confirmation for: Turkey Jerky
- *Line Item 13 chickens @ $9*
- *Line Total $117*
- *Tax $9.36*
- *Shipping $10*
- *Total $136.36*

JamJam inc placed order for 4 chickens @ $9
Order confirmation for JamJam inc returned in 00:00:00.0009996

Confirmation for: JamJam inc
- *Line Item 4 chickens @ $9*
- *Line Total $36*
- *Tax $2.88*
- *Shipping $10*
- *Total $48.88*

Enterprise Chickens placed order for 5 chickens @ $9
Order confirmation for Enterprise Chickens returned in 00:00:00.0009994

Confirmation for: Enterprise Chickens
- *Line Item 5 chickens @ $9*
- *Line Total $45*

Tax $3.6
Shipping $10
Total $58.6

The output is consistent and expected as the confirmations all match to what the original orders were. Printing this before and after was an effective way to make sure that the multiple threads were preserving the data. In my early testing, I noticed that this output data could be corrupted and create confirmations that didn't match to orders or didn't add up to the correct total. These issues were addressed, and the data is intact and correct.

Another factor I used in testing was to watch the time it took for the confirmations to be returned. This was part of the assignment as we were asked to take a timestamp just before the order was sent and when the confirmation as received. Then print the calculated diff. I noticed that this was very consistent for the most part. The first confirmation was always the slowest and there were a few outliers, but otherwise very consistent.

There is one credit card error in the output, which is by design. I generate random credit card numbers of 8 single digit numbers and then test that the credit card number is 8 digits in the order processing object. To build the random number I use a string type and then convert to an integer. In the conversion from a string to an integer, a leading zero will be lost, making the number only seven digits. This ensures that the verification function will be used when this condition occurs and will print an error.

I did some experimenting on the sleep time between price cut events, but still used 500 milliseconds in the final project. 500 milliseconds were used in the original chicken farm and retailer program and I thought it best to stick with it. However, what I expected to find was some type of corruption if I lowered the sleep time to 100 milliseconds or less. I found that it didn't make a difference until I lowered the time to below 50 milliseconds. At this point the thread terminated too quickly to print all the price cuts.