CPSC2108 Data Structures, Spring 2014

ASSIGNMENT 4 – DUE BEFORE MIDNIGHT, WEDNESDAY, APRIL 2

Warehouse Simulation

The purpose of this assignment is to practice the application of the Stack and Queue Abstract Data Types (ADT) for problem solving.

Write a program to simulate the operation of a warehouse by processing orders and shipments over a 24-hour period. The warehouse will have numerous shipments of widgets, and orders for these widgets. Each shipment of widgets has a cost per widget (the price varies because of different suppliers) and a quantity associated with it.

The accountants for the firm have instituted a first-in, first-out (FIFO) system for filling orders. But, because it is easier to retrieve shipment boxes closer to the entrance of the warehouse, a last-in, first-out (LIFO) system of using shipments to fill orders is used by the warehouse manager. This means the most recently arrived widgets are the first ones sent out to fill an order. More than one shipments may be used to meet a single order. This method of inventory can be represented using a queue, **orders-to-be-filled**, and a stack, **widgets-on-hand**.

When a shipment of new widgets is received, any unfilled orders (in the orders-to-be-filled queue) are processed and filled. After all orders are filled, if there are any widgets remaining in the new shipment, the shipment object with these widgets is pushed onto the widgets-on-hand stack.

When a shipment of new widgets is received, any unfilled orders (on the orders-to-be-filled queue) are processed and filled. After all orders are filled, if there are any widgets remaining in the new shipment, a new widget object with these widgets is pushed onto the widgets-on-hand stack.

When an order for new widgets is received, one or more shipment objects are popped from the widgets-on-hand stack until the order has been filled. If the order is completely filled and there are widgets remaining in the last shipment object popped, a modified shipment object with the quantity updated is pushed onto the widgets-on-hand stack. If the order is not completely filled, the order is added to the orders-to-be-filled queue to be sent out later with the remaining quantity of widgets.

Table 1 below shows the shipping(S) and order(O) events with their times, quantities and prices (for shipments only) over a 24-hour period. It also shows the widgets-on-hand stack and the orders-to-be-filled queue growing or shrinking with every event.

Table 1. Simulation test data

Time	Event	Quantity	Price($)	Widgets-on-hand stack	Orders-to-be-filled queue
0	S1	500	5	S1(500)	
1	S2	100	4	S1(500)-S2(100)	
2	O1	250		S1(350)	
3	O2	300		S1(50)	
4	S3	500	4.50	S1(50)-S3(500)	
5	O3	300		S1(50)-S3(200)	
6	O4	400			O4(150)
7	O5	100			O4(150)-O5(100)
8	S4	1000	4	S4(1000) S4(750)	
9	O6	350		S4(650) S4(400)	
10	S5	500	4.75	S4(400)-S5(500)	
11	S6	300	5	S4(400)-S5(500)-S6(300)	
12	O7	750		S4(400)-S5(50)	
13	O8	300		S4(150)	
14	O9	250			O9(100)
15	S7	500	5.25	S7(400)	
16	O10	300		S7(100)	
17	O11	400			O11(300)
18	O12	250			O11(300)-O12(250)
19	O13	150			O11(300)-O12(250)-O13(150)
20	S8	300	4.10		O12(250)-O13(150)
21	O14	50			O12(250)-O13(150)-O14(50)
22	S9	300	4.75		O13(100)-O14(50)
23	S10	100	4.50		O14(50) O13(50)
24	S11	750	4	S11(700)	

After an order is filled, display the time, the quantity sent out and the total cost for all widgets in the order. Also indicate if there are any widgets remaining to be sent out at a later time.
After a shipment is processed, display information about each order that was filled with this shipment and indicate how many widgets, if any, were stored in the shipment object pushed onto the widgets-on-hand stack.

For input data, you should create a text file **warehouse_data.txt** with the data from Table 1.This file will contain one shipment or order event per line. Each shipment or order should have its quantity and price (if applicable) associated with it.

You can implement the 24-hour simulation cycle with a loop. It will maintain a clock, whose time is incremented by an hour at each iteration of the loop, during which the next event is read from the data file and processed.

What to submit:

- A single zipped folder named in the format: *<your last name_initial_Assignment 4>.zip*, containing the following:
  - ✓ The .jar file (including all source code files and your project file)
  - ✓ A Word or .rtf document showing screen captures of test runs.
  - ✓ Data file warehouse_data.txt containing shipments and orders data for testing your program

  It is your responsibility to make sure your jar file can be extracted and the code runs on a different computer.

  Submit your zip file using CougarView Assignment 4 drop box before midnight, Wednesday, April 2. You must mention in the comments box in CougarView any problems encountered by you.

Grading Rubric:

1	Good documentation (follow style given in textbook program listings) • Header documentation • Class and method documentation	10 points
2	Good programming style • Good use of white space • Meaningful variable names • Use of indentation	10 points
3	Robust • Will not crash with unexpected input • Will not crash if a data file is not found	10 points
4	Program • Fulfills all requirements • Solves the problem • Good program logic • Good use of data structures • Efficient algorithm design	60 points
5	Sample runs • Screen shots of sample runs are included to adequately demonstrate the program	5 points
6	Instructions followed	5 points

Maximum points awarded for programs that fail to compile: 50 points
Make sure you are aware of the penalties of not following the **academic honesty** guidelines given in page 6 of the course syllabus.