CSc 2720 - Data Structures: Assignment 3

How to Submit:

Turn the .java files in Folder Assignment 3 in iCollege no later than **11:59 p.m. on 03/17/2018**.

Note:

- Handle all the possible exceptions regarding insertion, deletion or update operations in the list using the given ProductException class.

Requirements:

You have a new project in your company and your client is a small shopkeeper that needs to manage the shop inventory. Your job is to design and implement **ProductNode** and **ProductInventory** classes that manage the **inventory** using a linked List ADT. The specifications of the two classes are as follows:

ProductNode Class:

For each product, you should store the following info:

1. **Product Name** (i.e. “Apple iphone 7 plus 32gb rose gold”, may contain spaces in it, not unique)
2. **Locator** (string used to physically locate product, not unique)
3. **Quantity** (how many of this product in stock, greater than or equal to 0)
4. **Price** (in dollars and cents, greater than 0)

**Note**: For a given product, the product name **AND** locator together must be unique. So you may have two entries for “iPhone 5c“ if one has “box3” for the locator and the other has “shelf12”.

You should implement two constructors:

- A constructor that takes no arguments (quantity and price are initialized to 0, product name and locator are initialized to empty strings)
- A constructor that takes a value for each of the four-member variables.

ProductInventory Class:

- **showInventory**: displays a listing of the product inventory to the screen, one product entry per line. Output locator, then quantity, then price, then product name.
- **getTotalQuantity**: returns the total number of units of **all** of the products in the inventory.
- **removeMaximum**: finds the maximum product in the list based in the quantity. It removes the node of the maximum from the list and returns a pointer to this node.
- **sortInventory**: reorders the products in the list based on their quantity, using **removeMaximum**. Here is the algorithm you must use for implementing the sort function. It is a form of the selection sort. It selects the next (maximum) element from the current list and removes its node from the current list, and then inserts the node at the front of a new list (pointed to by a temporary pointer). When finished, it makes the old head pointer point to the new list. “inventoryHead” is the head pointer and while
inventoryHead is not empty find the maximum product in inventoryHead list and remove it (i.e. call removeMaximum) add the maximum node to the front of a new temporary list end and finally make inventoryHead point to the new temporary list.

addProduct: takes a product and adds it to the inventory. If a product with the same name and locator already exists in the inventory or if the quantity or price are invalid, the function throws a ProductException.

removeProduct: takes a product name and locator and removes any matching product from the inventory.

countProduct: since the same product can exist in the inventory with different locators. The function should take the name of a given product, traverse the inventory list and count the quantity of the same product.

countNeededQuantity: the store keeper needs to order products in the inventory to avoid any future shortage. The function should take the name of the product and the desired quantity of a given product as an argument. The function calls countProduct to deduce the needed quantity.
public class ProductNode {
    ProductNode next;

    //Constructor # 1
    ProductNode()
    {
        // Initialize other variables here
        next = null;
    }
    //Constructor # 2
}

public class ProductInventory {

    ProductNode inventoryHead = new ProductNode();

    public void showInventory()
    {
    }

    public int getTotalQuantity()
    {
        return 0;
    }

    public ProductNode removeMaximum()
    {
        return null;
    }

    public void sortInventory()
    {
    }

    public void addProduct(String productName, String locator, int quantity, float price)
    {
    }

    public void removeProduct(String productName, String locator)
    {
    }

    public int countProduct(String productName)
    {
        return 0;
    }

    public int countNeededQuantity(String productName, int neededQuantity)
    {
        return 0;
    }

    class ProductException extends RuntimeException {
        public ProductException(String s)
        {
            super(s);
        }
    } // end ProductException
}

// End ProductInventory
public class InventoryTester {

public static void main(String[] args){
ProductInventory inventory = new ProductInventory();

// Add Products to inventory
inventory.addProduct("Apple iphone 7 plus 32gb rose gold", "box256", 10, 699.80);
inventory.addProduct("Apple iphone 7 plus 32gb rose gold", "shelf4", 4, 699.80);
inventory.addProduct("Apple macbook pro", "box15", 2, 1500.87);
inventory.addProduct("Dell Monitor", "shelf10", 12, 221.54);

// Show inventory
inventory.showInventory();
/
Product Name | Locator | Quantity | Price
------------------------
Apple iphone 7 plus 32gb rose gold | box256 | 10 | 699.80
Apple iphone 7 plus 32gb rose gold | shelf4 | 4 | 699.80
Apple macbook pro | box14 | 2 | 1500.87
Dell Monitor | shelf10 | 12 | 221.54
*/

// Sort Products in inventory
inventory.sortInventory();

// Show inventory After sorting
inventory.showInventory();
/
Product Name | Locator | Quantity | Price
------------------------
Dell Monitor | shelf10 | 12 | 221.54
Apple iphone 7 plus 32gb rose gold | box256 | 10 | 699.80
Apple iphone 7 plus 32gb rose gold | shelf4 | 4 | 699.80
Apple macbook pro | box14 | 2 | 1500.87
*/

System.out.println(inventory.countProduct("Apple iphone 7 plus 32gb rose gold");
// Should return 14
System.out.println(inventory.countNeededQuantity("Dell Monitor", 15)); // Should return 3
inventory.removeMaximum();
inventory.removeProduct("Apple iphone 7 plus 32gb rose gold", "shelf4");
System.out.println(inventory.getTotalQuantity()); // Should return 12
}
}