Programming Assignment: ShoppingCart

Check Due Dates on Canvas

Objectives:

Write a class named `ShoppingList` that represents a person's list of items to buy from the market and another class named `ShoppingItem` that represents a request to purchase a particular item in each quantity (e.g., four boxes of cookies). *Hint: Write the ShoppingItem class first*

The `ShoppingList` class should use an array field to store the grocery items and keep track of its size (number of items in the list so far). Assume that a shopping list will have no more than 10 items.

ShoppingList Class

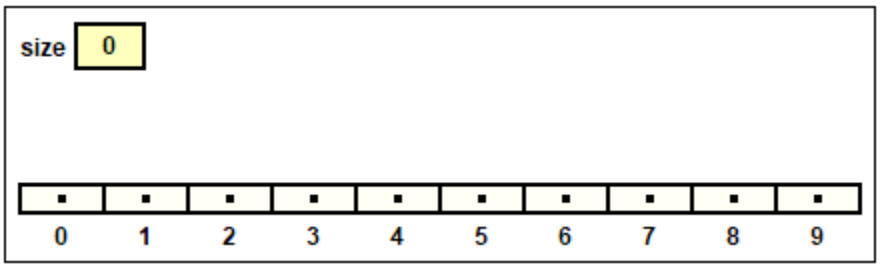
`ShoppingList` class should have the following two fields:

```
private ShoppingItem [] items;      //Array field to store
                                    // the shopping items
private int size;          //number of items in the list
```

A ShoppingList class should have the following methods with other methods you may want to add:

```
// constructs a new empty ShoppingList.
public ShoppingList ()
```

The constructor should generate a size 10 empty shopping list as follows:

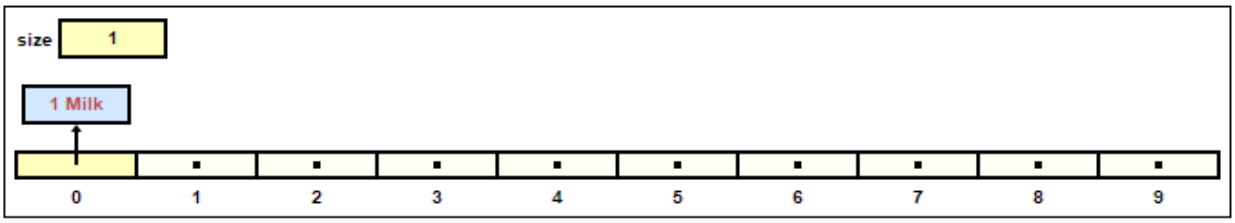



```
// adds the given item to this list if the list has fewer
than 10 items. If the shopping list is full, return false.
Return true if add is successful.
public boolean add (ShoppingItem item)
```

Feel free to use the debugger and visualize the objects as you proceed with the code. For example, to add a shopping item to the shopping list, client will ask the following questions:

```
Press your choice from 1 to 5: 2
Enter the shopping item: Milk
Enter the quantity: 1
Enter the price: 0.99
```

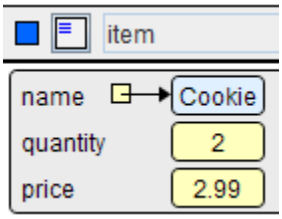
Client creates the ShoppingItem Milk and calls the add method to add it to the ShoppingList as follows:



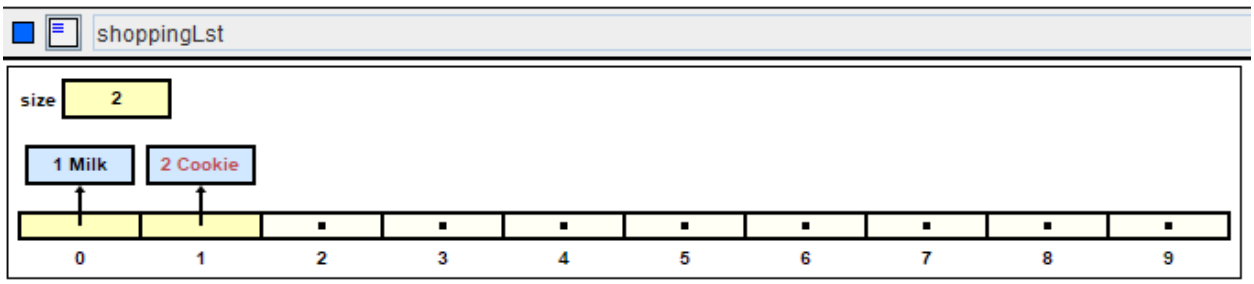
Assume the user wants to add a second item as follows:

```
Press your choice from 1 to 5: 2
Enter the shopping item: Cookie
Enter the quantity: 2
Enter the price: 2.99
```

The following ShoppingItem will be generated:



And the ShoppingItem will be inserted to the ShoppingList as follows:



```
// returns the total sum cost of all shopping items in this
list
public double getTotalCost ()


// returns String form of ShoppingList
// for example: ShoppingList has 3 shopping items: 3 Milk,
// 4 Tissues, 1 Toothpaste
// If the ShoppingList is empty, return "No Items in your
shopping list"
public String toString ()



// searches an item in this list by its name. Return the
item if match is found, otherwise return null.

public ShoppingItem searchByName (String itemName)
```

ShoppingItem Class

A `ShoppingItem` class should have the following three fields:

```
private String name;          // Name of the shopping item
private int quantity;         //quantity
private double price;         //price per unit.
```

`ShoppingItem` class should have the following methods (feel free to add other methods as necessary):

```
// Constructs an item order to purchase the item with the
// given name, in the given quantity, which costs the given
// price per unit

public ShoppingItem (String name, int quantity, double
pricePerUnit)


// returns the name of this item
public String getName()
```

```
// returns the total cost of this item in its given
//quantity. For example, four boxes of cookies that cost
//2.30 per unit have a total cost of 9.20
public double getCost()


// sets this grocery item's quantity to be the given value
public void setQuantity(int quantity)


// returns quantity and the shipping item name separated by
// space. For example, the method will return "3 Eggs"
// given the shopping item name is Eggs, and the quantity
// is 3.
public String toString ()
```

Deliverables

- Submit the following files
    1. `ShoppingList.java`
    2. `ShoppingItem.java`
    3. `ShoppingClient.java`

`Shoppingclient.java` will contain the client program to build the Grocery Shopping application. At minimum, your client program should have the following features:

1. Menu driven loop
    a. Option 1: Display the menu for the user.
    b. Option 2: Add a shopping item (User input from the keyboard: Item name, quantity and the unit price)
    c. Option 3: Change the quantity of the item in the shopping list. First you need to search for the item in the list (`searchByName` method in `ShoppingList`). If the match is found, item object will be returned from the `searchByName()`. If the item is not in the list, display a message such as "`Item is not found`"
    d. Option 4: Display the shopping list. **<u>If the shopping list is empty, display "No items in your shopping list"</u>**. Also display the total cost for the items in the shopping list.
    e. Option 5: Exit the program

2. User input validation
    a. Your program should not crash with invalid user inputs.
        i. For example, if user enters wrong values (i.e., enters a non-numeric value when asked for the quantity), display an error message and prompt the user again until they enter the correct value.
        ii. Also, validate user inputs in the Option menu.

Use plenty of `println` statements as you go to see what is working and what isn't working. Printing out test variables can be very useful as you go. Or use the debug features in jGrasp.

Make the programs work in stages as you go. Test your code **incrementally**. Add one method, run the program to see that it works, then add another.

Style Guidelines

Since this assignment is largely about classes and objects, much of the style grading will be about how well you follow proper programming style we have studied this quarter and also the **object-oriented** programming style. **You should encapsulate the data inside your objects, and you should not declare unnecessary data fields to store information that isn't vital to the state of the object**.

You should follow good general style guidelines such as properly using indentation, good variable names and types. Each method including main should not have more than 80 lines including blank lines. Place comments at the beginning of each class documenting that object's behavior, as well as on top of each method and on any complex code. **Write JavaDoc for your own class implementations (5 pts)** (`ShoppingList.java, ShoppingItem.java`). Place appropriate comments for the client program (JavaDoc is not necessary for the client program).