

Assignment 7 Instructions

1. Assignment 07: **37 - 40 points total**

[35pts + 2 E.C. pts = 37 points] or [35pts + 2 e.c pts for being on time and 3 e.c pts for Ask and Answer = 40 points] or [35pts + 3 e.c points for Ask and Answer = 38 pts]

Extra Credit points:

2 E.C. points (for on-time work)

3 E.C. points (for Ask and Answer forum participation),

2. Due Date & Time: **10/26/2020 at 11:59 PM**

WHAT TO SUBMIT

Submit 2 files to iLearn by the deadline and post a reflection on iLearn:

- 1 Java File: Please submit 1 files to iLearn: **CoffeeShopAccountInventoryManagerNew.java[30 points]**
- 1 PDF File: Submit 1 Word/PDF file which is a filled-out, downloaded local copy of this Google page on your local computer, named "firstname-lastname-assignment-7-report.pdf". Fill this out with screenshots then save it as Word or PDF

Then go to Google forms to submit:

- 1 Reflection

Then go to iLearn to ask a question:

- 1 Optional Ask and Answer forum

HOW TO SUBMIT

Please upload all 2 files separately via iLearn Assignments Submission

GUIDELINES FOR ALL ASSIGNMENTS:

1. Each assignment includes a code portion and a non-code portion. Please submit both 2 portions.
 - a. Code portion: Your source code files, only the files which you create and edit.
 - b. Non-code portion: Your assignment report, only 1 **Word** or **PDF** file.
2. Please submit all required files separately, un-zipped, via iLearn Assignments Submission
3. Always **read through the entire assignment before starting and submitting any of it. Missing files or missing requirements will result in deducted points**
4. **a. Include a proper header at the top of every Java file. Figure 1**

Header Format
<pre>/* * Assignment <assignment number> * Description: <program description> * Name: <your name> * ID: <your SFSU ID number> * Class: CSC 210-<section number> * Semester: <current semester> */</pre>

Replace each tag (such as **<assignment number>**) with the appropriate text.

You should adhere to this format as closely as possible. You do not need to include the **<>** symbols in your header fields.

- b. Only if you work with a Study Buddy, include your Buddy's name in your header at the top of every Java file. Figure 1**

Header Format
<pre>/* * Assignment <assignment number></pre>

```
* Description: <program description>

* Name: <your name>

* Teammate: <Study Buddy name>

* ID: <your SFSU ID number>

* Class: CSC 210-<section number>

* Semester: <current semester>

*/
```

Assignment 7

COFFEE SHOP ACCOUNTING OPERATIONS

☐ Part 1: CoffeeShopInventoryManagerNew [10 points]

File Name: CoffeeShopInventoryManagerNew.java

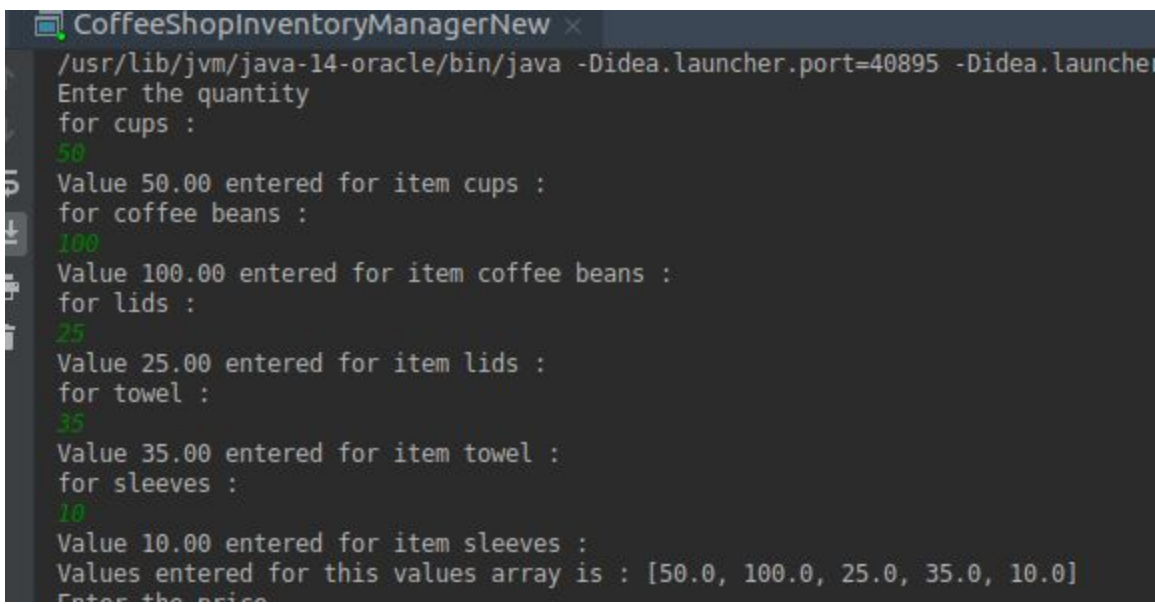
1. (Total 10 points) Refactor previous CoffeeShopInventoryManager.java and combine 2 arrays into 1-2D array and keep your first array “items”
 - a. (1 points) You should already have the first array “items” should contain 5 items {"cups", "coffee beans", "lids", "towel", "sleeves"}. Use Array-Initializer notation for assigning these 5 values to this array.
 - b. (2 points) Create a 1D array “labels” for each row, first element of the array is “quantity”, and second is “price”.
 - c. (4 points) Initialize the secondary array, a 2D array, with length of “items” and “labels”. Do not hardcode the amount of row and columns of the 2D array - points will be taken off for hardcoded array row and column lengths. `int[][] my2D arr = new int [2][5] -> new int [items.length][labels.length]` This second array, named “values” its first row, should contain quantity for each of the corresponding elements in “items”. The second row should contain the price per piece of the corresponding “items”. Obtain these quantities and prices from the user in one loop. Check if it is a valid number (greater than or equal to 0).

```
labels = {quantity, price};
```

```
for int i = 0 items.length
    for int j = 0 label.length
        temp = scan.nextInt();
    do
        if temp < 0
            /print out enter value bigger than 0
        else
            myArr[i][j] = temp
    while(temp < 0)
    System.println(label[i]) // row 0 ,print quantity, row 1 print price
    items[index for items]
```

d. (3 points) Use all arrays in one loop to output a helpful message to the user, prompting entry into each row label and each item name, for each entry into the 2D array.

See output below:

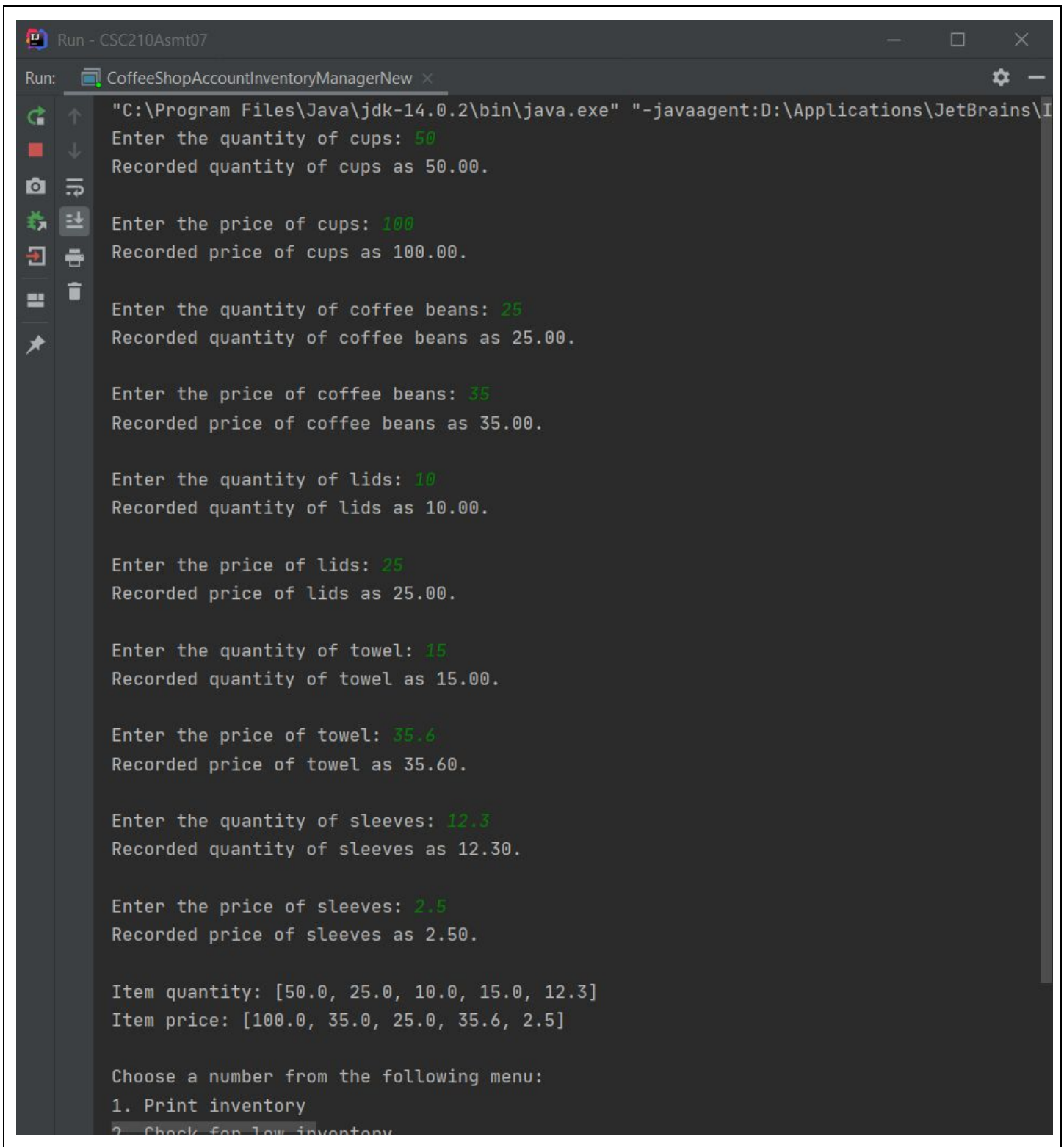


```
CoffeeShopInventoryManagerNew x
/usr/lib/jvm/java-14-oracle/bin/java -Didea.launcher.port=40895 -Didea.launcher
Enter the quantity
for cups :
50
Value 50.00 entered for item cups :
for coffee beans :
100
Value 100.00 entered for item coffee beans :
for lids :
25
Value 25.00 entered for item lids :
for towel :
35
Value 35.00 entered for item towel :
for sleeves :
10
Value 10.00 entered for item sleeves :
Values entered for this values array is : [50.0, 100.0, 25.0, 35.0, 10.0]
Enter the price
```

```
Enter the price
for cups :
25.0
Value 25.00 entered for item cups :
for coffee beans :
12
Value 12.00 entered for item coffee beans :
for lids :
35.6
Value 35.60 entered for item lids :
for towel :
12.3
Value 12.30 entered for item towel :
for sleeves :
2.5
Value 2.50 entered for item sleeves :
Values entered for this values array is : [25.0, 12.0, 35.6, 12.3, 2.5]
```

Please paste your screenshot for this scenario here:

Output :



The screenshot shows a Java IDE console window titled "Run - CSC210Asmt07". The console output displays the execution of a program named "CoffeeShopAccountInventoryManagerNew". The program prompts the user to enter the quantity and price for various items, and then displays the recorded values and a menu.

```
Run: CoffeeShopAccountInventoryManagerNew x
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:D:\Applications\JetBrains\I
Enter the quantity of cups: 50
Recorded quantity of cups as 50.00.

Enter the price of cups: 100
Recorded price of cups as 100.00.

Enter the quantity of coffee beans: 25
Recorded quantity of coffee beans as 25.00.

Enter the price of coffee beans: 35
Recorded price of coffee beans as 35.00.

Enter the quantity of lids: 10
Recorded quantity of lids as 10.00.

Enter the price of lids: 25
Recorded price of lids as 25.00.

Enter the quantity of towel: 15
Recorded quantity of towel as 15.00.

Enter the price of towel: 35.6
Recorded price of towel as 35.60.

Enter the quantity of sleeves: 12.3
Recorded quantity of sleeves as 12.30.

Enter the price of sleeves: 2.5
Recorded price of sleeves as 2.50.

Item quantity: [50.0, 25.0, 10.0, 15.0, 12.3]
Item price: [100.0, 35.0, 25.0, 35.6, 2.5]

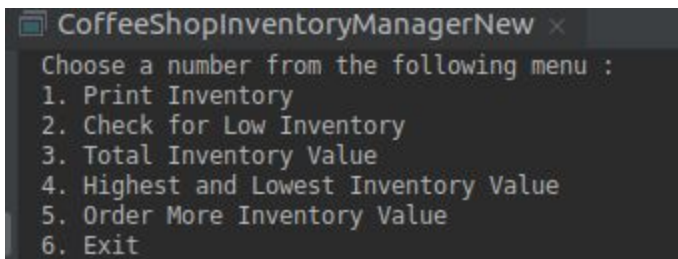
Choose a number from the following menu:
1. Print inventory
2. Check for low inventory
```

☐ Part 2: CoffeeShopInventoryManagerNew [17 points]

2. (Total 17 points) A typical user of this program is the inventory manager. Refactor your previous CoffeeShopInventoryManager file such that this new file has **at least 6 methods**

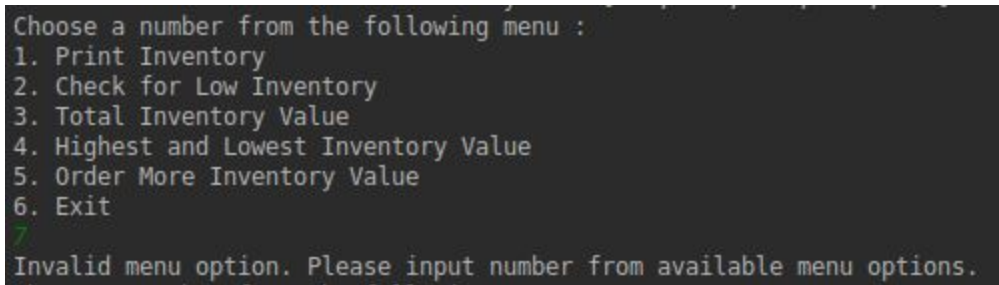
a. (2 points) 1st method: "getMenu". Display to the user the possible operations on the inventory and prompt her/him to choose one. There are 6 possible operations: Print Inventory, Check for low inventory, Total inventory value, Highest and lowest inventory value items, Ordering More Inventory, and Exit.

- **Use a switch case to direct the program to call the other methods.** The details of each of these operations are given below.

A screenshot of a terminal window titled "CoffeeShopInventoryManagerNew". The prompt "Choose a number from the following menu :" is followed by a numbered list: 1. Print Inventory, 2. Check for Low Inventory, 3. Total Inventory Value, 4. Highest and Lowest Inventory Value, 5. Order More Inventory Value, and 6. Exit.

```
CoffeeShopInventoryManagerNew x
Choose a number from the following menu :
1. Print Inventory
2. Check for Low Inventory
3. Total Inventory Value
4. Highest and Lowest Inventory Value
5. Order More Inventory Value
6. Exit
```

- **If the user enters an invalid menu option please prompt them to input the correct numbers**

A screenshot of a terminal window showing the same menu as the previous one. The user has entered the number 7, which is highlighted in green. Below the menu, the text "Invalid menu option. Please input number from available menu options." is displayed.

```
Choose a number from the following menu :
1. Print Inventory
2. Check for Low Inventory
3. Total Inventory Value
4. Highest and Lowest Inventory Value
5. Order More Inventory Value
6. Exit
7
Invalid menu option. Please input number from available menu options.
```

Please paste your screenshot for this invalid scenario here:

Output :

```
Choose a number from the following menu:
1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit
>>> 8

Invalid selection. Please enter a valid option.
```

b. (3 points) 2nd method: "printInventory". This operation prints the inventory in the following format:
Item Name, Quantity, Value then Item Name, Price, Value.. Go back to step a.

```
1
Quantity
Item Name cups : 5.00
Item Name coffee beans : 3.00
Item Name lids : 2.00
Item Name towel : 1.00
Item Name sleeves : 5.00
Price
Item Name cups : 3.50
Item Name coffee beans : 2.00
Item Name lids : 3.50
Item Name towel : 2.50
Item Name sleeves : 1.40
```

Please paste your screenshot for this scenario here:

Output :


```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 1
```

```
Quantity
```

```
Item Name cups : 50.00  
Item Name coffee beans : 25.00  
Item Name lids : 10.00  
Item Name towel : 15.00  
Item Name sleeves : 12.30
```

```
Price
```

```
Item Name cups : 100.00  
Item Name coffee beans : 35.00  
Item Name lids : 25.00  
Item Name towel : 35.60  
Item Name sleeves : 2.50
```

c. (3 points) 3rd method: "checkInventory": This operation checks for items that have 5 or fewer quantity, and prints them in the same format as in option(b).

If there is no such item then print an appropriate message. Finally, go back to step a.

```
2  
Item Name cups has low quantity: 5  
Item Name coffee beans has low quantity: 3  
Item Name lids has low quantity: 2  
Item Name towel has low quantity: 1  
Item Name sleeves has low quantity: 5
```

Please paste your screenshot for this scenario here:

Output :

```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 2
```

```
All stocked up! No items less than 5 in quantity.
```

```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 2
```

```
Item Name cups has low quantity: 1.00  
Item Name coffee beans has low quantity: 1.00  
Item Name lids has low quantity: 1.00
```

d. (3 points) 4th method: "minMaxInventory" This operation finds the item with the highest inventory value (quantity*price-per-piece),

If there is more than one item with the same highest (or lowest) value then display all such items. Finally, go back to step a.

```
4  
The least valued inventory is cups at this total value 1.50.  
The most valued inventory is coffee beans at this total value 125.00.
```

Please paste your screenshot for this scenario here:

Output :

```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 4
```

```
Highest total value items are cups at $5000.00.
```

```
Lowest total value items are sleeves at $30.75
```

```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 4
```

```
Highest total value items are towel, sleeves at $36.00.
```

```
Lowest total value items are cups, coffee beans, lids at $1.00
```

e. (2 points) 5th method: "getTotal". This operation computes the grand total value of the current inventory using the quantity and price per piece information, and prints that grand total value. Finally, go back to step a.

```
6. Exit
```

```
3
```

```
Total value of items is : 23.50
```

Please paste your screenshot for this scenario here:

Output :

```
Choose a number from the following menu:
```

1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit

```
>>> 3
```

```
Total value of items is : 6689.75
```

f. (3 points) 6th method: “orderInventory”. This operation displays the menu of items and asks the user to enter the number they would like to order. This operation also updates the main value 2D array quantity chosen and **returns an array to the main method, which updates the original array**. Finally, go back to step a.

```
5
Enter the number next to the item you would like to order.
1. cups
2. coffee beans
3. lids
4. towel
5. sleeves
4
Okay, enter the quantity you would like to order for towel. 12
Great! We have ordered more quantity, your new quantity for item towel is 47
```

For example if a **towel** was chosen to add quantity of 12, then the next printout towel value has been updated. See towel item below was 35 and now it is 47.

BEFORE

```
Quantity
Item Name cups : 5.00
Item Name coffee beans : 1.00
Item Name lids : 25.00
Item Name towel : 35.00
Item Name sleeves : 12.00
Price
Item Name cups : 1.50
Item Name coffee beans : 2.60
Item Name lids : 3.60
Item Name towel : 4.60
Item Name sleeves : 1.20
```

AFTER

```
2
Quantity
Item Name cups : 5.00
Item Name coffee beans : 1.00
Item Name lids : 25.00
Item Name towel : 47.00
Item Name sleeves : 12.00
Price
Item Name cups : 1.50
Item Name coffee beans : 2.60
Item Name lids : 3.60
Item Name towel : 4.60
Item Name sleeves : 1.20
```

Please paste your screenshot for this scenario here:

Output :

```
Enter the number next to the item you would like to order.
1. cups
2. coffee beans
3. lids
4. towel
5. sleeves
4

Okay, enter the quantity you would like to order for towel: 12
```

Before

```
Quantity
Item Name cups : 5.00
Item Name coffee beans : 1.00
Item Name lids : 25.00
Item Name towel : 35.00
Item Name sleeves : 12.00
Price
Item Name cups : 1.50
Item Name coffee beans : 2.60
Item Name lids : 3.60
Item Name towel : 4.60
Item Name sleeves : 1.20
```

After

```
Quantity
Item Name cups : 5.00
Item Name coffee beans : 1.00
Item Name lids : 25.00
Item Name towel : 47.00
Item Name sleeves : 12.00
Price
Item Name cups : 1.50
Item Name coffee beans : 2.60
Item Name lids : 3.60
Item Name towel : 4.60
Item Name sleeves : 1.20
```

g. (1 point) Exit (e): Exits the program.

```
Choose a number from the following menu :
1. Print Inventory
2. Check for Low Inventory
3. Total Inventory Value
4. Highest and Lowest Inventory Value
5. Order More Inventory Value
6. Exit
6
Thank you. Goodbye!
```

Please paste your screenshot for this scenario here:

Output :

```
Choose a number from the following menu:
1. Print inventory
2. Check for low inventory
3. Total inventory value
4. Highest and lowest inventory value
5. Order more inventory value
6. Exit
>>> 6

Exiting... Goodbye!

Process finished with exit code 0
```

❑ Part 3: Reflect 50 words [8 points]

Let's reflect on what we've learned so far.



[Assignment 7 Part 3: Your own Reflection](#)

The assignment was kind of straightforward but it was pretty tedious. Some of the directions were a bit unclear. For example, it was unclear whether the 2D array had to be of int or double type. Instructed to create an 2D int array, but the examples clearly show that a double type was needed.

Assignment also never specified if fields should/can be used in the program. If the arrays were declared as a field (which is what I did) the bolded requirement in (2)(f) wouldn't make sense to implement. I did it regardless, though. Despite having the return value be redundant.

❑ Part 4 Optional Extra Credit : Ask and Answer [3 e.c]

points]

Ask or Answer a question posted in this form



[Assignment 7 Part 4: Ask and Answer](#)