**CS 305: HW 0**
**Fall 2019**

## During the semester, there are free tutoring sessions. Wednesdays, Fridays and Sundays @7pm in Shiley 208 (days and hours might change). If you don't show up at 7pm tutors will go home

**Due Date: Friday September 1st NLT midnight**. Submit your code files (.java files) and short answer questions as a single .zip file to Moodle.

This is the only assignment this semester in Java. This assignment is meant to *review* what you learned in CS 203. It reviews classes, instance variables, constructors, methods, arrays, loops, and selection. If this assignment is difficult for you, please take the time during the first week of class to practice your programming skills. If you have hard time with the assignment, make standing appointments with the tutors, or see me to get more practice and to discuss strategies in order to be successful in CS 305.

Your project **MUST** compile and run on the university managed Ubuntu18 Linux VDI machines accessed at desktop.up.edu

This assignment is to be completed **individually**.

**Specification**

For this project, implement a record keeping application of a store inventory.

The implementation should include 3 Java classes:

- Main (provided to you)
- Item (you should implement this class)
- Inventory (you should implement this class)

Here are the class diagrams for the two classes you should implement:

| Inventory |
|---|
| -inventory : Item[]<br>-stock : int[]<br>-numItems : int<br>-maxInventorySize : int |
| +Inventory(maxInventorySize : int)<br>+addItem(it : Item, count: int) : int<br>+soldItem(id : int) : int<br>+printInventory() : void |

| Item |
|---|
| -price: double<br>-name : String<br>-ID : int |
| +Item(price : double, name : String, ID : int)<br>+print() : void<br>+getPrice() : double<br>+getID() : int |

Recall that – means private and + means public. Thus, the instance variables in the classes should be private and the constructors and methods should be public.

*Inventory Class*

This project implements Inventory as a collection of *Item*(s) who are located int the store. The Inventory object has *maxInventorySize* = the maximum number of items that can be found in the store. The *numItems* is the current number of items that are located in the inventory.

The constructor for the *Inventory* class should check the parameter *maxInventorySize*. If *maxInventorySize* <= 0, then set the *maxInventorySize* to 10. If the parameter is > 0, set the variable *maxInventorySize* is set to the value passed to the constructor. Then, create a new array of *Item*(s) objects of size *maxInventorySize*, create an integer array *stock* of the size *maxInventorySize*, and set *numItems* to 0.

The *addItem* method will check to see if *numItems* >= *maxInventorySize*. If so, this item cannot be added to the inventory since the maximum inventory size was reached. Print "*Cannot add another item to the Inventory: Maximum number of items reached.*" If this is the case, the method should return the value -1. Otherwise, add the item to the inventory array at position *numItems*, increment *numItems* by 1. Next, if the number of items added to the inventory is < 0, set the item's stock count in the stock array to 0, otherwise set it the count passed in. Return 1.

The *soldItem* method will locate the item in the inventory by the *ID*:int, and if the *item* is in the *inventory*, it's *stock* count is decremented by one and the following message is printed "*Sold Item and inventory stock decremented*". If the *item* is not in the inventory, print "*Could not sell item: not in inventory*", if the item's count is <1, print "*Count not sell item: Item's inventory count is 0*".

The *printInventory* method should print "*Store has the following items in the inventory: *" followed by a newline. Then, it should print every *item* in the store's *inventory*, using the *print()* method in the *Item* class and adding the *item*'s current *stock* in the following format: Item <price> <tab> <name> <tab> <ID> <tab> <stock> The *print* method should also calculate the total value of inventory as the sum of each item's price (unit price) multiplied by the number of items in stock, and the item with the min and max price in the inventory. Use the *getPrice()* method in the *Item* class to retrieve item's *price*. Include items with zero stock count when looking for the cheapest and most expensive items in the inventory.

*Item*

The *Item* class represents individual Item that could be placed in the store's inventory. For this project, we will keep it simple. Each *Item* has a *price*, a *name*, and an *ID*.

The constructor should take the item's price, name and ID and assign these instance variables accordingly.

The *print* method should print the Item's details in the format:
Item <price> <tab> <name> <tab><ID>
(see example output below for the format)

The *getPrice* and *getID* methods will return the value of the instance variable *price* and *ID* respectively.

*Error-checking*
All methods should do proper error-checking of the parameters. If the parameter value does not make sense (for example, a negative *stock*), then the code should set an appropriate value such as 0.

*Documentation*
The code should be well-commented, well indented, and include enough whitespace to be readable.

**Example Output**
The following should be printed to the console/screen when the main method is executed. (Check the driver Main.java, this is the output it generates)

```
Store has the following items in the inventory:
Item price      name            id      stock
Item 10.25      WarmFingers     1       10
Item 5.75       WarmEars        2       2
Item 35.22      KeepOnStomping  3       0
Item 67.04      PlankOnWheels   4       0
The total inventory is worth: 114.00
The cheapest item store carries is: 5.75
The most expensive item store carries is: 67.04

Store has the following items in the inventory:
Item price      name            id      stock
Item 10.25      WarmFingers     1       10
Item 5.75       WarmEars        2       2
Item 35.22      KeepOnStomping  3       0
Item 67.04      PlankOnWheels   4       0
Item 352.0      QuickPlanks     5       5
Item 223.47     Fortress007     6       2
Item 125.17     WarmDreams      7       10
Item 63.58      SoftDreams      8       12
The total inventory is worth: 4335.60
The cheapest item store carries is: 5.75
The most expensive item store carries is:
352.00

Store has the following items in the inventory:
Item price      name            id      stock
Item 10.25      WarmFingers     1       10
Item 5.75       WarmEars        2       2
Item 35.22      KeepOnStomping  3       0
Item 67.04      PlankOnWheels   4       0
Item 352.0      QuickPlanks     5       5
Item 223.47     Fortress007     6       2
Item 125.17     WarmDreams      7       10
Item 63.58      SoftDreams      8       12
Item 458.46     QuickDownTheRiver 9     4
Item 25.14      Burn Hot        10      6
The total inventory is worth: 6320.28
The cheapest item store carries is: 5.75
The most expensive item store carries is:
458.46

Cannot add another item to the Inventory:
Maximum number of items reached.
Cannot add another item to the Inventory:
Maximum number of items reached.
Store has the following items in the inventory:
Item price      name            id      stock
Item 10.25      WarmFingers     1       10
Item 5.75       WarmEars        2       2
Item 35.22      KeepOnStomping  3       0
Item 67.04      PlankOnWheels   4       0
Item 352.0      QuickPlanks     5       5
Item 223.47     Fortress007     6       2
Item 125.17     WarmDreams      7       10
Item 63.58      SoftDreams      8       12
Item 458.46     QuickDownTheRiver 9     4
Item 25.14      Burn Hot        10      6
The total inventory is worth: 6320.28
The cheapest item store carries is: 5.75
The most expensive item store carries is:
458.46

Sold Item and inventory stock decremented
Sold Item and inventory stock decremented
Store has the following items in the inventory:
Item price      name            id      stock
Item 10.25      WarmFingers     1       10
Item 5.75       WarmEars        2       1
Item 35.22      KeepOnStomping  3       0
```

Item 67.04 PlankOnWheels 4 0
Item 352.0 QuickPlanks 5 4
Item 223.47 Fortress007 6 2
Item 125.17 WarmDreams 7 10
Item 63.58 SoftDreams 8 12
Item 458.46 QuickDownTheRiver 9 4
Item 25.14 Burn Hot 10 6
The total inventory is worth: 5962.53
The cheapest item store carries is: 5.75
The most expensive item store carries is: 458.46

Sold Item and inventory stock decremented
Count not sell item: Item's inventory count is 0
Count not sell item: Item's inventory count is 0
Store has the following items in the inventory:
Item price name id stock
Item 10.25 WarmFingers 1 10
Item 5.75 WarmEars 2 0
Item 35.22 KeepOnStomping 3 0
Item 67.04 PlankOnWheels 4 0
Item 352.0 QuickPlanks 5 4
Item 223.47 Fortress007 6 2
Item 125.17 WarmDreams 7 10
Item 63.58 SoftDreams 8 12
Item 458.46 QuickDownTheRiver 9 4
Item 25.14 Burn Hot 10 6
The total inventory is worth: 5956.78
The cheapest item store carries is: 5.75
The most expensive item store carries is: 458.46

Could not sell item: not in inventory
Store has the following items in the inventory:
Item price name id stock
Item 223.47 Fortress007 6 5
Item 352.0 QuickPlanks 5 10
The total inventory is worth: 4637.35
The cheapest item store carries is: 100.00
The most expensive item store carries is: 352.00

**Optional - Additional requirements**

Only attempt these if you have completed the required project functionality listed above. Feel free to add the components to your program. Please document the additional features in your code and in your written summary. This is not extra credit, but a chance for you to do some exploration.

- Add a `deleteItem` method to the `Inventory` class. This should delete the Item based on the item's ID number and shift the Items so the inventory array stays "packed".
- Add more attributes to the `Item` class, such as `description`, `category` etc.). You may need to modify the Main class to account for these new attributes.
- Make sure that the Item with an exiting ID in the inventory cannot be added for the second time.
- If you have `Item` with attributes, print the inventory according to category instead of the order that the items are added to the inventory.
- Print the inventory in the sorted order from cheapest to most expensive item.

**Logistics**

1. Download the starter code `Main.java` located on Moodle. You may use BlueJ, Eclipse, Notepad++ or another IDE to develop your Java classes. You will need to add two more classes. You MUST name them `Inventory.java` and `Item.java` and the methods must have the same signature and names as stated in the above specifications.
2. Your project must compile on the Ubuntu 18 VDI (desktop.up.edu) from command line using: `javac Main.java` and must be executable by `java Main`

3. When you are finished and ready to submit:
    1. Create a new folder called username_HW0.
        1. (For example, my would be cenek_HW0.)
    2. Copy your Inventory.java file into this folder.
    3. Copy your Item.java file into this folder.
    4. Copy your Main.java file into this folder.
    5. Copy your HW0Summary.txt file into this folder.
    6. Zip the folder by right-clicking and Send To->Compressed Folder (on Windows or Linux). Very similar to the compression utilities on Mac.
4. **What to turn in:** Submit your **username_HW0.zip** file to Moodle before the due date and time. After logging into learning.up.edu, navigate to CS 305. You will find a link to submit this file. You do not need to print anything.

**Grading Guidelines (total of 20 points)**

Your code files will be graded on a scale from 0 to 7 in two separate categories:

- (0-7pts) Code Quality: Design, commenting, whitespace, readability, using private and public correctly
- (0-7pts) Code Operation: Does code do what is listed in the specification?

Your summary report will be graded on a scale from 0 to 6 based on completeness and clarity.

**HW 0 Report Guidelines and Format – use the template provided below**

**----------**

**Name:**
**CS 305 HW 0**

**1. (2 pts) Testing:** Describe how you tested and evaluated your program. If your program does not meet the specifications, please note these differences. Include a representative printout from executing your program with the original main method.

**2. (1 pt) Evaluation**: Evaluate the quality of your code. Here are some questions you may want to address: Is it readable and easy to follow? Do the parts make sense? Are your comments helpful?

**3. (3 pts) Questions**:
1a. At the bottom of the main method, uncomment the printing of the objects `rei` and `amazon`. Run the main method. What is printed for these objects?

1b. Slightly change the driver file Main.java by commenting out some “addItem” calls. Re-compile and re-run the code again. What is printed for these objects?

1c. What do these values represent?

2. In the main method, you will see a comment after the rei store inventory – draw what the inventory looks like after the items were added to the inventory. Draw the inventory array, the item objects stored in the array and the stock at this point of the code execution. If an array cell is empty, label it with the word “null”.

3a. Describe the most challenging aspect(s) for you when writing this program. (Hopefully, this assignment is review for you from CS 203.)

3b. How much time did you spend in total on this homework assignment (including the report)?

**Appendix A**: Copy and paste your java code here (use Courier New 9pt font so the characters line up correctly)

**Appendix B**: I certify that the submitted work is my own work. Any help I received is documented in code comments.