Programming Assignment 8: Create an inventory program
Total Points (50 pts) - Due Monday, November 16th at 11:59 PM

Overview

Create an inventory program that can be used for a range of different products (cds, dvds, software, etc.).

Topic(s):

- Using loops
- Using *if* statements
- Arrays of objects
- Instantiating Objects
- Creating Classes
- Constructors

1. Open the inventory program that was created in **Assignment 6: Creating an inventory project**
2. Modify the **ProductTester** class. Ask the user to enter the number of products they wish to add. Accept a positive integer for the number of products and handle the value of zero.
    a) Create a variable named *maxSize* that can store integers.
    b) Create a prompt at the beginning of your main method that will instruct the user to enter the required value for the number of products they wish to store:

        Enter the number of products you would like to add or
        Enter 0 (zero) if you do not wish to add products:

    c) Use a *do while* loop so that the program will not continue until a valid positive value is entered. If a value less than zero is entered an error message stating "*Incorrect Value entered!*" should be displayed before the user is re-prompted to enter a new value. You should not leave the loop until a value of zero or greater is entered.
3. Modify the **ProductTester** class to handle multiple products using a single dimensional array if a value greater than zero is entered.
    a) Create an if statement that will display the message "*No products required!"* to the console if the value of *maxSize* is zero.
    b) Add an *else* statement to deal with any value other than zero (0).
    c) Create a single one-dimension array named *products* based on the **Product** class that will have the number of elements specified by the user.
4. You are now going to populate the array, getting the values from the user for each field in a product object.
    a) Inside the *else* statement, under where you created the array, write a *for* loop that will iterate through the array from zero (0) to 1 less than *maxSize*.
    b) As the last input you received from the user was numeric, you will need to add a statement that clears the input buffer as the first line in your *for* loop.

c) Copy the code that you used to get input from the user for all a products fields into the *for* loop. This includes *name*, *quantity*, *price* and *item number*.

d) Add a new product object into the array using the index value for the position and the constructer that takes 4 parameters.

e) Use a *for each* loop to display the information for each individual product in the products array.

5. Remove any unnecessary code that's not used in this exercise.
6. Save your project.
7. You are now going to modify your code so that the main class will not do any processing but simply call static methods when required.

a) Create a static method after the end of the main method called **addToInventory.** This method will not return any values and will accept the *products* array and the Scanner as parameters.

b) Copy the code that adds the values to the array from the main method into the new **addToInventory** method.

c) To resolve the errors that you have in your code move the local variables required (*tempNumber, tempName, tempQty, tempPrice*) from the main method into the top of the **addInventory** method.

d) Add a method call in main to the **addToInventory()** method where you removed the *for* loop from.

e) Run and test your code.

f) Create a static method after the end of the main method called **displayInventory**. This method will not return any values and will accept the products array as a parameter. Remember when you pass an array as a parameter you use the class name as the data type, a set of empty square brackets and then the array name (*ClassName*[]*arrayName*).

g) Copy the code that displays the array from the main method into the new **displayInventory** method.

h) Where you removed the display code from main, replace it with a method call to the **displayInventory** method. Remember to include the correct argument list to match the parameter list in the method you are calling.

i) Run and test your code.

Here is a sample run:

```
Enter the number of products you would like to add.
Enter 0 (zero) if you do not wish to add products: 3

Please enter the product name: Pen
Please enter the quantity of stock for this product: 30
Please enter the price for this product: 3.99
Please enter the item number: 8
```

```
Please enter the product name: Pencil
Please enter the quantity of stock for this product: 40
Please enter the price for this product: 3.99
Please enter the item number: 3


Please enter the product name: Notebook
Please enter the quantity of stock for this product: 20
Please enter the price for this product: 9.00
Please enter the item number: 2

Item Number     : 8
Name          : Pen
Quantity in stock: 30
Price         : 3.99
Stock Value     : 119.7
Product Status   : Active


Item Number     : 3
Name          : Pencil
Quantity in stock: 40
Price         : 3.99
Stock Value     : 159.60000000000002
Product Status   : Active

Item Number     : 2
Name          : Notebook
Quantity in stock: 20
Price         : 9.0
Stock Value     : 180.0
Product Status   : Active
```

Submission Instructions

- Execute the program and copy/paste the output that is produced by your program into the bottom of the source code file, making it into a comment. I will run the programs myself to see the output.
- Make sure the run "matches" your source. If the run you submit does not match the source you submit, it will be graded as if you did not submit a run at all.
- Use the 'Assignments' submission link to submit the source code file.
- Submit the following files:
    - **Product.java**
    - **ProductTester.java**
- You will need to label your projects with your first initial, last name, and the name of the project.
- Zip the two files together to create one compressed file. Example: **hibrahim_assignment8.zip**
- Upload the compressed file into Canvas.