

Programming Assignment #9: Creating an inventory project

Total Points (50 points) - Due Monday, November 23rd at 11:59 PM

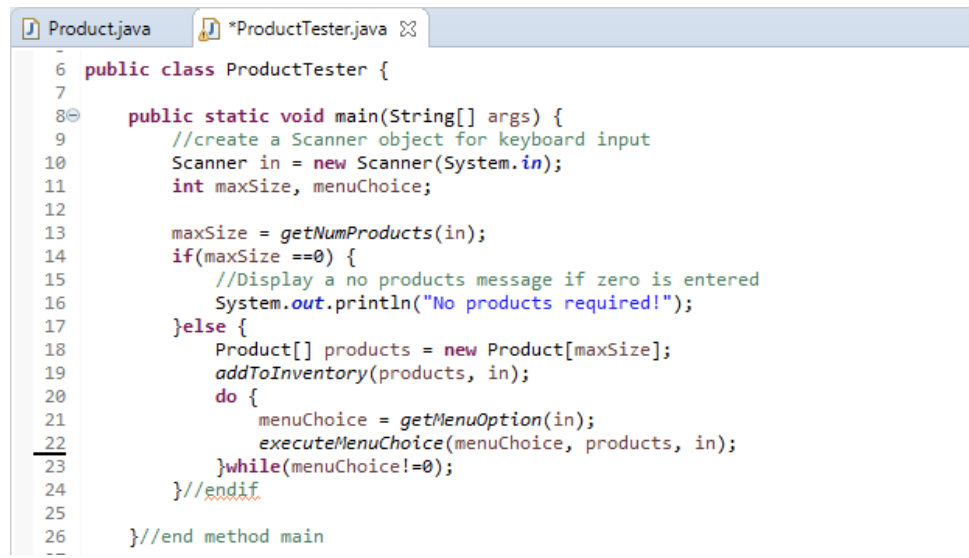
Overview

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

Topic(s):

- Modifying programs:
 - Creating static methods
 - Using parameters in a method
 - Return a value from a method
 - Adding methods (behaviors) to an existing class.
 - Implementing a user interface.
1. Open the inventory program that was updated in **Assignment 8: Creating an inventory project**
 2. Create two new methods in the **Product** class, one that will allow the user to add to the number of units in stock (**addToInventory**), and one that will allow the user to deduct from the number of units in stock (**deductFromInventory**). Both methods should accept a parameter (quantity) that holds the number of items to add/deduct. Place these under your constructors.
 3. Modify the **ProductTester** class so that the user can view, modify or discontinue the products through a user interface that is based on a menu system.
 - a) Display a menu system that will display options and return the menu choice entered by the user.
 - I. The method should be called **getMenuOption**, return an integer value, and take a **Scanner** object as a parameter. Write the code under main.
 - II. The menu should look like the following:
 - 1) View Inventory
 - 2) Add Stock
 - 3) Deduct Stock
 - 4) Discontinue Product
 0. ExitPlease enter a menu option:
 - III. Only numbers between zero (0) and 4 should be accepted, any other input should force a re- prompt to the user. Remember when adding a try catch statement, you have to initialize the variable to something that will fail the *while* condition!
 - b) Create a method that will display the index value of the array and name of each product allowing the user to select the product that they want to update (add/deduct).
 - I. The method should be called **getProductNumber**, return an integer value and take the products array and a Scanner object as parameters. It should have a single local variable named **productChoice** of type integer that is initialized to -1. Write the code under main.
 - II. A traditional *for* loop should be used to display the index value and the product name. Use the length of the array to terminate the loop. The name for each product can be accessed through its appropriate getter method
 - III. The user should only be allowed to enter values between zero (0) and 1 less than the length of the array. All input should have appropriate error handling.

- c) Create a method that will add stock values to each identified product.
 - I. The method should be called **addInventory**, have no return value and take the `products` array and a `Scanner` object as parameters. It should have two local variables named **productChoice** that does not need to be initialized and another named **updateValue** that should be initialized to -1. Both local variables should be able to store integer values. Write the code under main.
 - II. Add a method call to the **getProductNumber** method passing the correct parameters and saving the result in the **productChoice** variable.
 - III. The user should be prompted with the message *"How many products do you want to add?"* and only be allowed to enter positive values of zero (0) and above. All input should have both appropriate error handling and error message.
 - IV. Once a valid update value has been added then the selected product stock levels should be updated through the **addToInventory** method that you created earlier. The **productChoice** variable is used to identify the index value of the product in the array and the **updateValue** is the amount of stock to be added.
 - d) Create a method that will deduct stock values to each identified product.
 - I. Follow the same procedure as you did to add stock but name your method **deductInventory**. The restrictions on his input is that the value must be zero (0) or greater and cannot be greater than the current quantity of stock for that product. All input should have both appropriate error handling and error messages. Use the **deductFromInventory** method to make the change to the *product* object in the array.
 - e) The final menu option to implement is the ability to mark stock as discontinued.
 - I. The method should be called **discontinueInventory**, have no return value and take the `products` array and a `Scanner` object as parameters. It should have a single local variable named **productChoice** that stores an integer value and does not need to be initialized. Write the code under main.
 - II. Add a method call to the **getProductNumber** method passing the correct parameters and saving the result in the **productChoice** variable.
 - III. Now use the **setActive** method to set the value of active to false for the chosen product object.
 - f) You now need to create a method that will bring it all together.
 - I. The method should be called **executeMenuChoice** and have no return value. It should also take the menu choice, `products` array, and a **Scanner** object as parameters. It does not require any local variables. Write the code under main.
 - II. Use a switch statement to execute the methods that you have created in this exercise. For each case statement, use an output statement that will display one of the following heading before executing the appropriate method:
 - View Product List
 - Add Stock
 - Deduct Stock
 - Discontinue Stock
4. The final stage is to update the main method to make use of the new functionality. Update your code so that your main method matches the following code:



```
Product.java *ProductTester.java
6 public class ProductTester {
7
8     public static void main(String[] args) {
9         //create a Scanner object for keyboard input
10        Scanner in = new Scanner(System.in);
11        int maxSize, menuChoice;
12
13        maxSize = getNumProducts(in);
14        if(maxSize == 0) {
15            //Display a no products message if zero is entered
16            System.out.println("No products required!");
17        }else {
18            Product[] products = new Product[maxSize];
19            addToInventory(products, in);
20            do {
21                menuChoice = getMenuOption(in);
22                executeMenuChoice(menuChoice, products, in);
23            }while(menuChoice!=0);
24        }//endif
25
26    }//end method main
--
```

5. Run and test your code
6. Save your project.

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: 3.75
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: 7.75
Please enter the item number: 9

Please enter the product name: Notebook
Please enter the quantity of stock for this product: 20
Please enter the price for this product: 10.50
Please enter the item number: 6

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product

0. Exit

Please enter a menu option: 1
View Product List

CS 111B | Assignment #7

Spring 2021

Item Number : 8
Name : Pen
Quantity in stock: 30
Price : 3.75
Stock Value : 112.5
Product Status : Active

Item Number : 9
Name : Pencil
Quantity in stock: 40
Price : 7.75
Stock Value : 310.0
Product Status : Active

Item Number : 6
Name : Notebook
Quantity in stock: 20
Price : 10.5
Stock Value : 210.0
Product Status : Active

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit
Please enter a menu option: 2
Add Stock
0 : Pen
1 : Pencil
2 : Notebook
Please enter the item number of the product you want to update: 2
How many products do you want to add? 4

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit
Please enter a menu option: 1
View Product List

Item Number : 8
Name : Pen
Quantity in stock: 30
Price : 3.75
Stock Value : 112.5
Product Status : Active

Item Number : 9
Name : Pencil

CS 111B | Assignment #7

Spring 2021

Quantity in stock: 40
Price : 7.75
Stock Value : 310.0
Product Status : Active

Item Number : 6
Name : Notebook
Quantity in stock: 24
Price : 10.5
Stock Value : 252.0
Product Status : Active

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit
Please enter a menu option: 3
Deduct Stock
0 : Pen
1 : Pencil
2 : Notebook
Please enter the item number of the product you want to update: 1
How many products do you want to deduct? 3

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit
Please enter a menu option: 1
View Product List

Item Number : 8
Name : Pen
Quantity in stock: 30
Price : 3.75
Stock Value : 112.5
Product Status : Active

Item Number : 9
Name : Pencil
Quantity in stock: 37
Price : 7.75
Stock Value : 286.75
Product Status : Active

Item Number : 6
Name : Notebook
Quantity in stock: 24
Price : 10.5

CS 111B | Assignment #7

Spring 2021

Stock Value : 252.0
Product Status : Active

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit

Please enter a menu option: 4

Discontinue Stock

0 : Pen

1 : Pencil

2 : Notebook

Please enter the item number of the product you want to update: 0

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit

Please enter a menu option: 1

View Product List

Item Number : 8
Name : Pen
Quantity in stock: 30
Price : 3.75
Stock Value : 112.5
Product Status : Discontinued

Item Number : 9
Name : Pencil
Quantity in stock: 37
Price : 7.75
Stock Value : 286.75
Product Status : Active

Item Number : 6
Name : Notebook
Quantity in stock: 24
Price : 10.5
Stock Value : 252.0
Product Status : Active

1. View Inventory
2. Add Stock
3. Deduct Stock
4. Discontinue Product
0. Exit

Please enter a menu option: 0

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 could not have come from 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_assignment9.zip**
- Upload the compressed file into Canvas.