Java Fundamentals
Section 7 Part 1: Creating an Inventory Project
Project

Overview

This project will progress with you throughout Sections 4, 5, 6, and 7 of the course. After each section there will be more to add until it builds into a complete Java application to maintain Inventory. For each part, build upon the last part so that both the old and new requirements are met. Include all parts in a package called inventory.
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 (Section 7.3)
  - using parameters in a method (Section 7.1)
  - Return a value from a method (Section 7.1)
- Adding methods(behaviours) to an existing class (Section 7.2)
- Implementing a user interface (Sections 5.1, 5.2, 6.2)

1. Open the inventory program that was updated in **Section 6: Creating an inventory Project**

2. 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 in the **ProductTester** class 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*)

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

   c) 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.

   d) Run and test your code

   e) Create a static method in the **ProductTester** class 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.

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

   g) 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.

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

   i) Run and test your code

   j) Create a method in **ProductTester** that will return an integer value named getNumProducts() that accepts the scanner as a parameter. Move all the code that gets the maximum number of products from the user into this method, put a method call in main to your new method. You will store the returned value in your maxSize variable so you will need to declare 2 of these, one in your main method and one in the getNumProducts() method. You can remove the initial value of -1 from the declaration in main.

   k) Run and test your code

3. 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.

4. 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. Exit
Please enter a menu option:
```

        iii. Only numbers between 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 initialise 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 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 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 0 and above. All input should have both appropriate error handling and error messages.
        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 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, have no return value and 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
```

g) The final stage of this exercise 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:

```
public static void main(String[] args) {
    //create a Scanner object for keyboard input
    Scanner in = new Scanner(System.in);
    int maxSize, menuChoice;

    maxSize = getNumProducts(in);
    if(maxSize ==0) {
        //Display a no products message if zero is entered
        System.out.println("No products required!");
    }else {
        Product[] products = new Product[maxSize];
        addToInventory(products, in);
        do {
            menuChoice = getMenuOption(in);
            executeMenuChoice(menuChoice, products, in);
        }while(menuChoice!=0);
    }//endif

}//end method main
```

h) Run and test your code.

5. Save your project.