**CSC 3020 – Java Programming**

**Homework 5 – [your name]**

**25 points – Due April 4, 10am**

**Late deadline is April 6, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

You’ve been hired by *Camping Critters* to write a Java console application that manages their product inventory. The application has the following two classes:

**Product.java**

Each object created from this class represents one product in their inventory and includes the following fields and methods:

**Fields**

● (static) productCount – count of all distinct products; initialize to 0 in declaration.

● (static) inventoryValue – total inventory value of all products; initialize to 0 in declaration.

● (static) inventoryCount – total count of all products in inventory; initialize to 0 in declaration.

● code – two-digit product code of the product.

● name – name of the product.

● cost – cost of the product in dollars.

● count – current count of the product in stock.

**Methods**

● A constructor with no parameters that sets the fields, respectively, to these values:

productCount = productCount + 1

code = -1

name = "(not set)"

cost = -1

count = -1

● A constructor with four parameters that sets the fields, respectively, to these values:

productCount = productCount + 1

inventoryValue = inventoryValue + (cost \* count)

inventoryCount = inventoryCount + count

code – set from parameter

name – set from parameter

cost – set from parameter

count – set from parameter

● Getter methods for each field (declare the getters for productCount, inventoryValue, and inventoryCount static).

● Setter methods for each field (declare the setters for productCount, inventoryValue, and inventoryCount static).

● *equals* method that compares the product codes from two objects for equality.

● *toString* method that returns instance variable values only.

**HW5.java**

This class contains the main method and uses the Product class to store product data. Create text file *ProductInventoryIn.txt*, paste the following data into it, and place the file in your project folder. It has the following file specification:

|  |  |  |
| --- | --- | --- |
| Field | Type | Start-End |
| Product code | integer | 1-7 |
| Product name | string | 8-31 |
| Product cost | real | 32-41 |
| Product quantity | integer | 42-51 |
| The file does not contain a header row. | | |

**ProductInventoryIn.txt**

**80 Daypack 110.00 50**

**81 Duffel Bag 35.00 60**

**82 Hammock 70.00 70**

**83 Cot 155.00 80**

**84 Tent 430.00 90**

**85 Stove 100.00 40**

**86 Cooler 350.00 30**

**87 Sleeping Bag 320.00 20**

**88 Blanket 140.00 10**

**89 Camp Chair 120.00 100**

Read the data from file *ProductInventoryIn.txt* into an array of Product objects called **products**. Note that to call a method for any array element object, you use, for example:

products[i].getCount()

Present the following menu to the user:

Camping Critters Menu

1 – Sell product

2 – Order product

3 – List product inventory

4 – Exit

Enter an option:

Here are what the options do:

● **Sell product –** use a validation loop to prompt for and get from the user the code of the product to be sold (it has to be a valid code). Then use a validation loop to prompt for and get from the user the quantity of the product to be sold. Insure that the quantity is not greater than the current inventory for that product. Update the following fields:

✓ (static) inventoryValue

✓ (static) inventoryCount

✓ count for the product

Print a "product sold" message that includes:

✓ Code

✓ Quantity

✓ Revenue from the sale

Format the message in two columns with the first column containing a label and the second column containing a value. Format real numbers to two decimal places.

● **Order Product –** use a validation loop to prompt for and get from the user the code of the product to be ordered (it has to be a valid code). Then use a validation loop to prompt for and get from the user the quantity of the product to be ordered. Insure that the quantity is greater than zero. Update the following fields:

✓ (static) inventoryValue

✓ (static) inventoryCount

✓ count for the product

Print a "product ordered" message that includes:

✓ Code

✓ Quantity ordered

✓ Cost of the order

Format the message in two columns with the first column containing a label and the second column containing a value. Format real numbers to two decimal places.

● **List product inventory** shows all product data in formatted columns. It then lists the product count, inventory value, and inventory count.

● **Exit** closes the menu.

Java doesn't handle multiple keyboard objects gracefully so declare one keyboard object as a field (global) and close it at the end of method *main*. Continue to process menu options until the user enter 4. Then write the data to file *ProductInventoryOut.txt* in the same layout as the input file. Use these menu options and inputs for your last run:

**Option Code Quantity**

**3**

**1 80 10**

**3**

**2 81 40**

**3**

**1 82 20**

**3**

**2 83 25**

**3**

**1 10,84 100,10**

**3**

**4**

**Product.java**

*[your Product.java class here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**HW5.java**

*[your HW5.java class here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Program output**

*[your program output here (just the last two screens)]\*\**

**ProductInventoryOut.txt**

*[your ProductInventoryOut.txt file here]\*\*\**

\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.

\*\*\* **Copying-and-pasting text file contents to a Word document**

1) From the text editor screen, maximize the window.

2) From the window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.