Assignment #2: Revise a Java program

Manage Inventory for a Books, Toys, and Video Store

CS 4354 Summer II 2016
Instructor: Jill Seaman

Due: before class **Wednesday, 7/20/2016** (upload electronic copy by 9:30am)

Problem:

The video store has expanded and now also sells used books and toys in addition to DVDs. Modify your Java program from assignment 1 (or the solution) to handle these different types of products in the inventory.

The inventory for the store will contain the following information for each product in the inventory:

SKU	(stock-keeping unit, an integer, must be unique)
quantity	(number of copies in inventory, greater than or equal to 0)
price	(dollars and cents, greater than 0)
title	(may contain spaces in it)

In addition to these attributes, the following are stored for each product type:
For **movies** (dvds) a upc (universal product code) is stored
For **books**, an isbn (international standard book number) and author name are stored.
For **toys**, the weight (in total ounces!!) is stored

The program should offer the user a menu with the following options (changes from assignment 1 are in **bold**):

1. Add a product to the inventory (user should enter a letter (M or B or T) to indicate the **product category** and then input corresponding values).
2. Remove a product from the inventory (by sku).
3. Display the information for a product (given the sku).
4. Display the inventory in a table, **sorted by sku.**
5. **Process a sale.**
6. Quit

For #3, display **all** the attributes for the product (this will differ for each category).

For #4, display **only** the product category (Movie, Book, or Toy), sku, price, quantity, and title for each product.

The details of #5 Process a sale are given below.

The program should perform the operation selected by number and then re-display the menu. If the operation fails (i.e. invalid input or attempt to remove a product not in the inventory) your program should display an error message and return to the menu.

Your program should store the inventory in a file called "inventory.dat" between executions of the program, so that when the program is run again it will start up with the same inventory contents as when it last terminated.

5. Process a sale

The products are offered for sale on Amazon, which charges a commission on each sold item, and refunds a shipping credit to the store for each sold item (the store must ship the items offline and pay their own shipping costs).

To process the sale of a certain product, the user must input the following:

sku of the product sold
quantity of items sold to the customer
shipping cost (the amount paid by the store to ship these item(s))

Your program should update the quantity attribute for that product in the inventory, if there are enough items of that product in the inventory. If not, it should issue an error message and abort the operation. Do NOT remove the product from the inventory.

The shipping credit and commission that Amazon charges for each type of item of are shown below. Note that for toys, the shipping credit depends on the weight (in pounds, rounded up) of the item. So for a toy weighing 18 ounces, the shipping credit is $4.49 + ($0.50*2) = $5.49. (1 pound is 16 ounces, so 18 ounces rounds up to 2).

Product Type	Shipping credit per item	Commission
Movie	$2.98	12% of price
Book	$3.99	15% of price
Toys	$4.49 + $0.50 / lb	15% of price

To complete the process, compute and output the following four values:

Total price (the product's price times the quantity sold)
Total shipping credit (the product's shipping credit times the quantity sold)
Total commission (the product's commission times the quantity sold)

Profit = Total price + Total shipping credit - (Total commission + shipping cost)

NOTES:

- You may use an IDE (Eclipse, netbeans, etc) or just an editor and command line operations (javac, java) in unix or windows/dos to develop your program.

- This assignment is to be done with your partner (in groups of 2).

- Use inheritance to implement the different product types.
  - Make a superclass called Product
  - Make three subclasses: Movie, Book, and Toy

- Use good design with respect to inheritance:
  - Put the **common** attributes and methods into the superclass whenever appropriate. (Avoid duplicate code in the subclasses).
  - Subclass-specific attributes (not common) should go in the proper subclass.
  - Use abstract classes and methods when you can.
  - Make the instance variables **private**. (May need public getters).
  - Use methods (functions!) to compute (or return) the different values needed to process the sale of a given product. Some of these might be defined in the superclass, some might need subclass-specific definitions (polymorphism).
  - Note: if your method has a switch statement (or if) to determine if the product is a book/toy/video, it's probably NOT polymorphic!

- Implement the Comparable<T> (or Comparator<T>) interface in order to sort the Products (do NOT write your own sort function!)

- Use a package for your classes and put your files in the appropriate directories.

- Follow the style guidelines from the class website. **Use javadoc comments for all of your public elements.**

Logistics:

Please submit your files in a single zip file (assign2_xxxxxx_yyyyyy.zip). The xxxxxx and yyyyyy are your TX State NetIDs (mine is js236, you have one for each partner).

Submit: an electronic copy only, using the Assignments tool on the TRACS website for this class. Submit using the TRACS account of just ONE member of your partnership.