Assignment #3: Revise a Java program

Manage Inventory for an Online Store

CS 4354 Fall 2012
Instructor: Jill Seaman

Due: in class **Wednesday, 10/10/2012** (upload electronic copy by 11:30am)

Problem:

Write a Java program that will allow a user to manage the inventory of a store that sells DVDs, books, some games and puzzles, and used cell phones online (the video store owner has expanded his inventory).

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

SKU	(stock-keeping unit, an integer)
quantity	(number of copies in inventory)
price	(dollars and cents)
title	(may contain spaces in it)

For **movies** (dvds) a upc (universal product code) is also stored
For **books**, an isbn (international standard book number) is also stored, along with the author, publisher, and year of publication.
For **games** and **puzzles**, the weight (in ounces) is also stored
For **cell phones**, the weight (in ounces), the manufacturer, and model number are also stored (the description of the model is stored in the title field).

You will modify your inventory program from assignment 2 (or the solution) to accommodate the new product types.

The program should offer the user a menu with the following options (changes from the previous version in **bold**):

1. Add a product to the inventory (prompt user for **product category** and input 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 information available for the product item (this will differ for each category product).

For #4, display the product category, sku, price, quantity, and title for each product item.

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. attempt to remove a product not in the inventory) your program should display a message.

Do not change the menu numbers associated with the operations.

Your program should store the inventory in a file 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

To process the sale of a certain product, ask the user to input the following:
sku
quantity sold
date
shipping cost (the actual cost of shipping all the item(s), eg postage)

Make sure there are enough items in the inventory to meet the order (check the quantity in the inventory). If so, decrement the inventory quantity appropriately and proceed. If not, provide a message to the user and abort the operation. Do NOT remove the product from the inventory if the quantity is 0.

For each sale on the online venue, the venue collects a shipping credit, which it passes on to the seller, and it charges a commission from the seller. The rules for calculating the shipping credit and the commission vary by product type, and are shown in the table below.

Product Type	Shipping credit	Commission
Movie (dvd)	$2.98	12% of sale price
Book	$3.99	15% of sale price
Toys	$4.49 + .50/lb	15% of sale price
Electronics	$4.49 + .50/lb	8% of sale price

To process the sale, your program should compute and list the following values:

Date
Sku
Sale price
Shipping credit
Commission
Shipping cost
Profit

These values should reflect the total quantity of items sold. So the Sale price is really the price times the quantity sold. The same for the Shipping credit and Commission. The shipping cost is input from the user and should be the total cost to ship all of the items sold.

The Profit is calculated as follows:
Sale price + Shipping credit - 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.

- Prices should be output in standard money format: ($7.95).

- When designing classes: try to move the common attributes and methods into the superclass as much as possible. Use abstract classes when you can. You might find an extra abstract class to be useful as well.

- **Style guidelines:** comment the following: definitions of variables, constants, class members, sections of code, and functions (description+parameter descriptions+what it returns, if anything). Mind your line length and indentation. Use named constants as needed. Use static elements only as truly necessary.

Logistics:

Please submit your java files in a single zip file (assign3_xxxxxx.zip). The xxxxx is your TX State NetID (mine is js236).

Submit: an electronic copy only, using the Assignments tool on the TRACS website for this class.