**OOP Lab08 – Small Business Pt 1 (Item & Inventory only)**

**Note: This lab will eventually have 3 classes! Item, Inventory, & Inventory Tester**

You are an avid buyer and seller of random stuff. You find stuff, you buy it, and then later *you re-sell it*.

However, you occasionally get a little too fond of buying stuff, and your parents are not fond of you accumulating so much stuff in their garage.

To save space, they have imposed a three-item inventory limit on your small business. You can never have more than 3 items in your store inventory.

**Write a program to keep track of your inventory.**

First, write the Item class. An Item is a convenient bundle of a few different variables and methods.

Fields (private instance variables) – describe what an Item is

* String description;
* double cost;
* String dateListed; //String version of date you put item up for sale

Methods/Constructors – Make ONLY these.

* public Item(String d, double c, String date) – constructor, initializes the instance variables
* Include an accessor getCost()
* Include a toString() method returns a String that contains the information for this Item, in the following format (angle brackets indicate the instance variable values for this object):

<description>, $<cost> (Listed: <dateListed>)

**Common question from students:** “How many spaces are between the cost and the list date?” Answer: I’m using the tab escape sequence, so that list dates will line up better when you have lots of items.

Next, write the Inventory class.

Fields (instance variables)

* Item item1;
* Item item2;
* Item item3;

**NOTE: Your instance variables will be Item objects!**

Methods

* public Inventory() – default constructor, sets all Items initially to null

/\* **null** is the default value for a variable that ‘stores’ an object. The null keyword means that there is no object yet to refer to. In other words, the variables item1, item2, and item3, are unset references – they are Item type variables, but there is no Item object yet for these variables to reference \*/

* public Inventory(Item a, Item b, Item c) – parameterized constructor, initializes the instance variables
* public double netWorth() – returns a double value representing the cumulative cost of all three Items currently in your inventory. Your method should check if an Item actually exists (is not null) before adding its cost.
* public boolean roomForNewItem() – returns true if you have room to add an Item to your Inventory (if one of the Item instance variables has a null reference)
* public void setItemAsSold(int itemNumber) – sets the corresponding instance variable to null, indicating that you now have room for another item in your Inventory.

if (itemNumber == 1)

item1 = null;

//etc, for all items

* public void setNewItem(Item newItem, int itemNumber) – sets the value of the corresponding instance variable to the value of the new Item parameter. The method should PRINT an error message if that spot in inventory is unavailable because something else is currently there.
* Include a toString() method returns a String that contains the information about the 3 items in inventory (one per line). Should say “NO ITEM” for any spot of your inventory that is available (null).

Continued on next page.

Once you have the Item and Inventory classes completed, test your code with the InventoryTester class (written for you). Copy / paste the code below into a new class in your BlueJ project.

/\*\* THIS TESTS YOUR INVENTORY AND ITEM CLASSES

\* DO NOT CHANGE THIS CODE!

\* If your output doesn't match, make changes within

\* Inventory and Item. Not here!

\*/

public class InventoryTester

{

public static void main(String[] args)

{

//Make an empty garage. Check the net worth and see if there is room to add an item.

Inventory garage = new Inventory();

System.out.println("Current inventory: \n" + garage);

System.out.println("Net worth of inventory >>> $" + garage.netWorth());

System.out.println("Can add an item to inventory? >>> " + garage.roomForNewItem() + "\n");

//Create 3 items. Print them to test the toString()

Item birdCage = new Item("A decent bird cage", 45, "03/28/16");

Item cookBook = new Item("A new pizza cookbook", 8.75, "03/31/16");

Item hockeyStick = new Item("A new hockey stick", 31.50, "04/02/16");

//Testing toString of Item

System.out.println(birdCage);

System.out.println(cookBook);

System.out.println(hockeyStick + "\n");

//reconstruct garage as a new object, this time using parameters

garage = new Inventory(birdCage, cookBook, hockeyStick);

//Testing toString of Inventory

System.out.println("Current inventory: \n" + garage);

//Testing methods

System.out.println("\nCan add an item to inventory? >>> " + garage.roomForNewItem());

System.out.println("Net worth of inventory >>> $" + garage.netWorth() + "\n");

//Testing more methods: selling items, adding items.

System.out.println("Trying to add grill.");

Item grill = new Item("A small charcoal grill", 20, "04/04/16");

garage.setNewItem(grill, 2); //should fail.

System.out.println("Selling second item so we can add grill.");

garage.setItemAsSold(2);

garage.setNewItem(grill, 2);

System.out.println("NEW net worth of inventory >>> $" + garage.netWorth());

}

}

If the code above does not compile, fix any problems you have with your classes; do not alter this tester code.

Make sure your parameters match what’s expected, you’re initializing the instance variables properly, etc. When the code compiles, run the main() method in the InventoryTester class - the output should look exactly like the expected output on the next page:

Expected Output of InventoryTester:

Current inventory:

NO ITEM

NO ITEM

NO ITEM

Net worth of inventory >>> $0.0

Can add an item to inventory? >>> true

A decent bird cage, $45.0 (Listed: 03/28/16)

A new pizza cookbook, $8.75 (Listed: 03/31/16)

A new hockey stick, $31.5 (Listed: 04/02/16)

Current inventory:

A decent bird cage, $45.0 (Listed: 03/28/16)

A new pizza cookbook, $8.75 (Listed: 03/31/16)

A new hockey stick, $31.5 (Listed: 04/02/16)

Can add an item to inventory? >>> false

Net worth of inventory >>> $85.25

Trying to add grill.

Can't add item to that itemNumber. Currently occupied.

Selling second item so we can add grill.

NEW net worth of inventory >>> $96.5