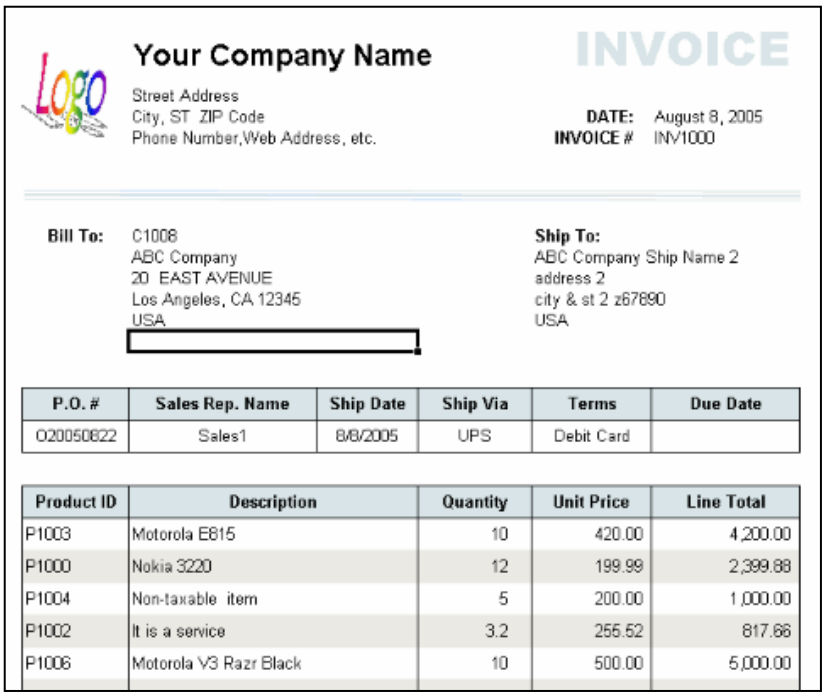
Review/Warm-up
Composition, Classes and Modularity
60 points



Your Company Name

INVOICE

Street Address
City, ST ZIP Code
Phone Number,Web Address, etc.

DATE: August 8, 2005
INVOICE # INV1000

Bill To: C1008
ABC Company
20 EAST AVENUE
Los Angeles, CA 12345
USA

Ship To:
ABC Company Ship Name 2
address 2
city & st 2 z67890
USA

P.O. #	Sales Rep. Name	Ship Date	Ship Via	Terms	Due Date
O20050822	Sales1	8/8/2005	UPS	Debit Card	

Product ID	Description	Quantity	Unit Price	Line Total
P1003	Motorola E815	10	420.00	4,200.00
P1000	Nokia 3220	12	199.99	2,399.88
P1004	Non-taxable item	5	200.00	1,000.00
P1002	It is a service	3.2	255.52	817.66
P1006	Motorola V3 Razr Black	10	500.00	5,000.00

Invoice Application

This assignment uses OOP design, modularity, composition and file handling to simulate a sales invoice as found in an order-entry system.

The application will be data-driven from two files:

`Items.txt`

and

`Order.txt`

Your solution should first read the Item.txt file to load an ArrayList with the available item objects. It should then read the Order.txt file to assemble a subset of Items that make up a customer's order.

Create an **Item** class that contains:

- Stock Keeping Unit ('SKU') – type Integer (with a big 'I')
- Item description – String
- Unit price – Double
- Quantity on hand ('QOH')
- Unit weight – Double (with a big 'D')

All data items are private.

The Item objects will simulate an inventory database in an actual OE system.

The Item class should have:

- A constructor ('ctor') that accepts the SKU, description, unit price, QOH and weight
- Sets and Gets as needed

- A toString method that returns a String with SKU, description, QOH and price (simulates an inventory inquiry capability)

Create a **LineItem** class that contains:

- Item – Type Item
- Quantity ordered – Integer

The LineItem class should have:

- A ctor that accepts an Item object and a quantity ordered
- Sets and Gets as required
- A getExtensionPrice method that returns (quantity ordered * unit price)
- A getExtensionWeight that returns (quantity ordered * weight)
- A toString method that returns a String formatted as

SKU	Description	Quantity	Unit price	Extension	Weight

- A compareTo method that compares LineItem objects by SKU

Create an **Invoice** class that contains:

- Order number – Integer
- Customer number – Integer
- Order date - Date
- An ArrayList collection of type <LineItem>

The Invoice class should have:

- A ctor that accepts an order number and a customer number, and creates an empty ArrayList of type LineItem
- Sets and Gets as required
- An AddItem method that accepts an Item object and quantity ordered, and adds it to the ArrayList
- A toString method that returns the order number, customer number, order date and each LineItem's data

A driver class named **InvoiceTester** that contains:

A readItems method that returns an ArrayList of inventory items (Item objects.) This method should read the Items.txt file and load the first ArrayList.

A createInvoice method that will read the Order.txt file. This method should:

- Accept the ArrayList of available Items
- Create an Invoice object
- Read the Order.txt file and add a new LineItem object to the ArrayList for each item found in the file (if an item SKU is not in inventory or if the QOH is 0, do not add a LineItem)
- Return an object of type Invoice

A printInvoice method that will:

- Access an Invoice object (we'll talk about how to approach this…)
- Print the order header and line item information for that Invoice

---

The Items.txt file will consist of:

Item number, description, unit price, quantity on hand and unit weight – each on a 'line' by itself.

An example follows – (Grader's actual data will vary, but will follow this format.):

- SKU
- Description
- Unit price
- QOH
- Unit weight

```
10001
Lumia 900
450.00
18
1.5
10002
Samsung Galaxy
375.00
0
1.75
10003
iPhone 4S
199.00
4
1.6
10004
iPhone 5
```

```
499.00
1
1.5
```

A given item will be represented in five lines of text data.

The Order.txt file will consist of a customer number and order number on the first two lines, and line item data (SKU and quantity - each on a line by itself) thereafter:

- Customer number
- Order number
- SKU
- Qty ordered
- SKU
- Qty ordered

```
42
2013001
10001
1
10002
5
10004
11
```

Example run:

```
My Company Name


Customer#       Invoice#        Order Date
42              2013001         Wed Apr 10 09:50:14 PDT 2013

     SKU       Description             Quantity Extension

     10001     Lumia 900                     1    450.00
     10002     Samsung Galaxy                5   1875.00
     10004     iPhone 5                     11   5489.00
```

Turn in all source code in a zip file named with your last name, followed by the first initial of your first name, followed by hw1 (ex: peterschw1.zip)

Get started ASAP!