Assignment #3 – COMP 2500

Comment: The system designs from assignment 2 were poorly done. In assignment 3 you will redo your system designs. However, when you implement your java code use collections. Note that the use of collections does not impact your UML design. Some guidelines are suggested for the design below.

Question

A bakery named Z located in Barbados has daily orders for each of its customers. An order is associated with one customer but can have many products associated with it. That is, a customer order can be made up of more than one product from the bakery. Once an order is entered into the computer system it can be used to produce an invoice which is identical to an order but has an invoice number assigned.

Although an invoice and order may appear similar, the computer system at Z bakery has separate storage areas for orders and invoices. Therefore, when an order is converted into an invoice the order information must be copied over into the invoice storage area and removed from the order storage area.

The clerical staff needs to be able to print both and invoice and an order. Also both an order and an invoice should have the ability to calculate the total price of the products on them.

You are required to do download ***ordersystem_files.zip*** from myElearning and study the structure of the order system files. You are required to write a java program which:

1. Allows order information to be loaded into a system of classes from the files given
2. Allows a user to convert an order to an invoice – this process is called posting the order
3. Allows a user to create and store data in a file system for the invoices similar to that of the order system
4. Allows a user to print both an invoice and order.

Start by drawing a UML diagram for the system and paying attention to the fact that data for both the invoices and orders class systems must be ***loaded*** and ***written*** to the respective file systems.

Hints for Design:

- Use the concept of composition to group objects. Example of objects for composition are (not limited to): Orders, Order, Items
- Use an interface called IOperations which forces object with common operations to implement them.
- For classes like Order which only use the customer no. and not the entire customer object, use a knows-a relationship
- Your Programme should contain a class called Bakery which basically contains collection objects called Orders and Invoices