In this assignment, you will apply design patterns to the design and implementation of an on-line shopping site, using the framework diagrammed below.

The classes are described below:

**Store** – the on-line shopping site itself
- **Attributes**
  - name – the site’s name (e.g., “Amazon.com”)
  - departments – the various sales departments
  - customers – the clients currently using the store
- **Operations**
  - enter(Customer c) - Customer c enters the store
  - exit(Customer c) – Customer c exits the store
  - getShoppingCart() – returns an empty ShoppingCart
  - getWishList() – returns an empty WishList
  - customers() – returns a list of the customers in the store
  - departments() – returns a list of the store’s departments
  - addDepartment() – adds a new department to the store

**Department** – abstract superclass for a department
- **Attributes**
  - name – the department’s name (e.g., “Computer Software”)
  - items – items available for sale in the department
  - customers – the clients currently using the department
  - observers – the clients to be notified of sales or new items
  - departmentID – a unique ID for the department
- **Operations**
  - enter(Customer c) - Customer c enters the department
  - exit(Customer c) – Customer c exits the department
  - getShoppingCart() – returns a list of the customers in the store
  - departments() – returns a list of the store’s departments
  - addDepartment() – adds a new department to the store
  - addItem(item) – add an item to the list
  - removeItem(item) – remove an item from the list
  - items() – returns a list of all current items

**Item** – an item for sale in a department
- **Attributes**
  - name – the item’s name (e.g., “Adobe Photoshop”)
  - ID – a unique ID for the item
  - departmentID – the ID of the department selling the item
  - price – the price of the item

**Customer** – a client of the on-line shopping site
- **Attributes**
  - name – the customer’s name (e.g., “Jane Doe”)
  - shoppingCart – the cart being used by the customer
  - wishList – the wish list generated by the customer
  - department – the department the customer is currently in

**ItemList** – abstract superclass for a list of items
- **Attributes**
  - items – items currently in the item list
- **Operations**
  - addItem(item) – add an item to the list
  - removeItem(item) – remove an item from the list
  - items() – returns a list of all current items

**ShoppingCart, WishList** – subclasses of ItemList
Using the framework described above, implement a working version of the on-line shopping site. (Implement a desktop simulation of an on-line application.)

Your assignment must have at least one customer, at least four departments, and at least six items for sale in each department. The user interface should be simple (e.g., no images or animations) and may be either textual or graphical.

Your program must implement the following design patterns:

- **Iterator**
  Any operation that returns a list should demonstrate the Iterator pattern.

- **Abstract Factory (and/or Factory Method)**
  All departments should be created by means of factories.

- **Singleton**
  Only one instance of any department type should be allowed to be created.

- **Observer**
  Departments are observable (i.e., a customer should be able to request notification whenever a new item is added for sale in a department or a specified old item’s price is reduced). Demonstrate this pattern by having at least one customer register for such notifications with at least one department.

- **Composite**
  Customers can be afforded the opportunity to buy combinations of items within a department (e.g., all six seasons of “The Sopranos” on Blu-ray, all three volumes of Donald Knuth’s “The Art of Computer Programming”).

A short report (no more than five pages) should be submitted with your fully implemented program. This report should describe your use of design patterns in the assignment, including:

- Class diagrams
- Identification of other viable patterns for this program (or reasonable extensions of it)
- Instructions on how to use your submitted software package

Place the project folder and the report in a common folder and submit that zip-compressed folder on your drop-box.

This assignment is due on your drop-box by 9 AM on Thursday, April 16, 2009.