Design Patterns Assignment: Requirements

1 Overview

In this assignment you will apply design patterns to the design and implementation of an simplified on-line shopping system.

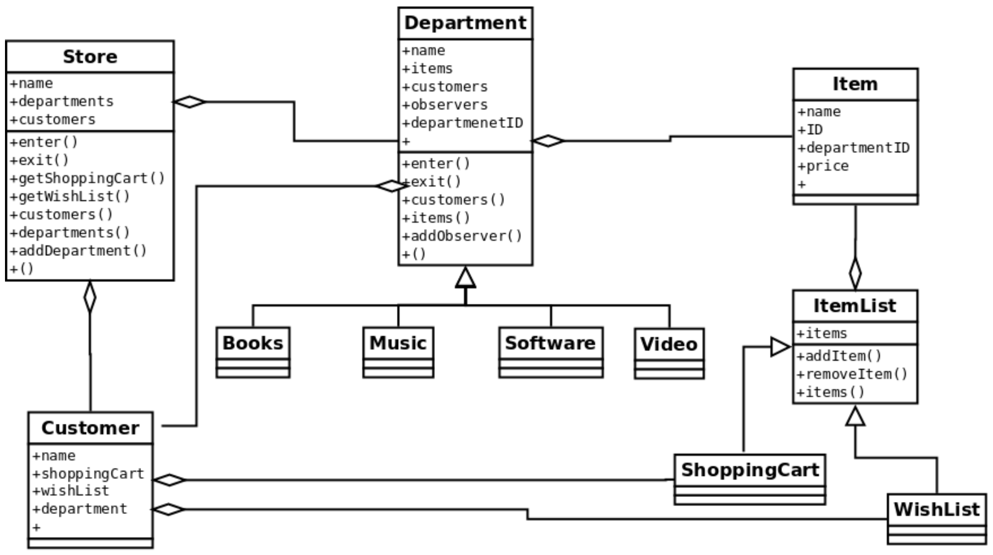

Figure 1: UML Diagram: Simple Store

2 Background

Figure 1 describes the framework of the back-end for on-line shopping site. The classes in this framework are as follows:

2.1 Class Descriptions

2.1.1 Store: The backend representation of the site

The `Store` class is the model for the store. The web server passes requests through this object.

Attribute	
name	The site's name
departments	The various departments in the store
customers	The customers currently using the store

Operations	
enter(Customer c)	Customer enters the store
exit(Customer c)]	Customer exits the store
getShoppingCart()	Returns an empty shopping cart
getWishList()	Returns an empty wish list
cusomters()	Returns a list of customers in the store
departments()	Returns a list of the store's department
addDepartment()	Adds a new department to the store

2.1.2 Department

Abstract superclass for a department.

Attributes	
name	The department's name
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 enters the store
exit(Customer c)	Customer exits the store
customers()	Return a list of customers in the department
items()	Return a list of items for sale in the department
addObserver()	Add a new observer to the department

The classes `BookDepartment` and so on are possible sub-classes of the `Department` abstract class.

2.1.3 Item

An item for sale in a department.

Attributes	
name	The item's name
ID	A unique ID for the item
departmentID	The ID of the department selling the item
price	The price of the item

2.1.4 Customer

A client of the on-line shopping site.

Attributes	
name	The customer's name
shoppingCart	The cart being used by the customer
wishList	The wish list generated by the customer
department	The department in which the customer is currently shopping

2.1.5 ItemList

An abstract superclass for a list of items. This class has (at least) two specializations: `ShoppingCart` and `WishList`.

Attributes	
items	Items currently in the item list

Operations	
addItem(Item i)	Add an item to the list
removeItem(Item i)	Remove an item from the list
items()	Returns a list of all current items

2.2 Initial Requirements

1. Customer transactions start when the enter the store.

2. Customer's checkout when they exit the store.

3. A customer should be able to request notification whenever a new item is added for sale in a department or a specified item's price is reduced.

4. Customers can be afforded the opportunity to buy combinations of all items within a department

5. The user interface should be simple (no images or animations) and may be either textual or graphical.

6. A testing framework is required for the back-end that will be independent of the user interface (opportunity for use of a unit test framework).

3 Problem

Your task is to build a prototype of this shopping site back-end system. This can be done using any of the three languages we have discussed in class. You are expected to refactor the design as you go through the implementation process. However, design changes must be documented by updating the class diagram provided with the assignment.

Note that the requirements require you to implement at least seven design patterns: Iterator, Abstract Factory, Factory Method, Singleton, Observer, and Composite. There will be others that may be needed as well depending upon revisions required to the design and implementation decisions you make.