

UpGrad - Content Strategist Assignment

Design Pattern

Date: 15th June 2017

Md Syed Ahamad
IIIT Guwahati.

Design Pattern is technique used by experienced Software Developers for software development designing in easy and true way. It is a tried and tested solution to a general software problems which helps the developers to design similar problems in the future. Design patterns can even improve the documentation and maintenance of existing systems by furnishing an explicit specification of class and object interactions and their underlying intent. Learning these patterns helps unexperienced developers to learn software design in an easy and faster way.

Designing a problem should be specific to the problem keeping that it should be general enough to address future problems and requirements. That design should be reused for similar problems instead of redesign. Recurring patterns of classes and communicating objects in many object-oriented systems solves specific design problems and makes the object-oriented designs more flexible, elegant and ultimately reusable.

Studying Design Pattern is the good way to build our skills as a Software Developer's point of view. It is advice from developers who is expert in building such things for long time. It is consequently to be their most common and simple. **Design Patterns: Elements of Reusable Object-Oriented Software** is a Software Engineering book describing Software Design Patterns published by four authors Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides (they are collectively known as **Gang of Four (GOF)**). There are 23 design patterns which can be classified in three categories: Creational, Structural and Behavioural patterns in the book.

Creational Patterns: It concerns the process of object creation. It make a system independent of how its objects are created, composed and represented thus hiding the creational logics. This gives program more flexibility in deciding which objects need to be created for a given use case. Creational patterns ensure that your system is written in terms of interfaces, not implementations.

Structural Patterns: It deal with the composition of classes or objects. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

Behavioural patterns: It characterize the ways in which classes or objects interact and distribute responsibility.

Purpose	Design Patterns	Aspect
Creational	Abstract Factory	families of product objects
	Builder	how a composite object gets created
	Factory Method	subclass of object that is instantiated
	Prototype	class of object that is instantiated
	Singleton	the sole instance of a class
Structural	Adapter	interface to an object
	Bridge	implementation of an object
	Composite	structure and composition of an object

	Decorator	responsibilities of an object without sub classing
	Facade	interface to a subsystem
	Flyweight	storage costs of objects
	Proxy	how an object is accessed; its location
Behavioural	Chain of Responsibility	object that can fulfil a request
	Command	when and how a request is fulfilled
	Interpreter	grammar and interpretation of a language
	Iterator	how an aggregate's elements are accessed, traversed
	Mediator	how and which objects interact with each other
	Memento	what private information is stored outside an object, and when
	Observer	number of objects that depend on another object; how the dependent objects stay up to date
	State	states of an object

Table 1.1 Design aspects of design patterns

Discussing complete design patterns at once would be heavy and difficult to have visual ideas of it. To make easier and make concept clearer, I will discuss a Software Application “Item Delivery Application” and its associated design pattern.

Designing an online item delivery application requires the following use cases:

1. User can search different stores nearby
2. User can select a store
3. User can see items in a store
4. User can add an item from a store
5. User can order the items
6. User can cancel the order
7. User can pay for the order

I have implemented the application by using suitable design patterns according to its modules.

1. Store Search – Interpreter Design Pattern
2. Item list in a store – Iterator Pattern
3. Adding item for order – Builder Pattern
4. Order and Cancellation of items – Command Pattern
5. Order status in real time – Observer Pattern

Module 1: Store Search – Interpreter Pattern

This pattern involves implementing an expression interface which tells to interpret a particular context. It comes under behavioural pattern. It is suitable for pattern matching, customer can search things based on given attributes such as location, name, items, PIN, etc. I have used this design pattern for the searching of stores based on location.

Module 2: Item list in a store- Iterator Pattern

The Iterator Pattern provides a way to access elements of an aggregate object sequentially without exposing the underlying structure of the object. It is a behavioural pattern. I have used this design

pattern for the displaying of item list of a particular store. It does not necessarily have to expose its internal representation such as item list matching of the store.

Module 3: Adding item for order – Builder Pattern

It is used for building complex object from simple objects by using a step-by-step approach. It is a creational pattern used to configure and assemble complex objects. I have used this design pattern for adding items because different kinds of items can create an order.

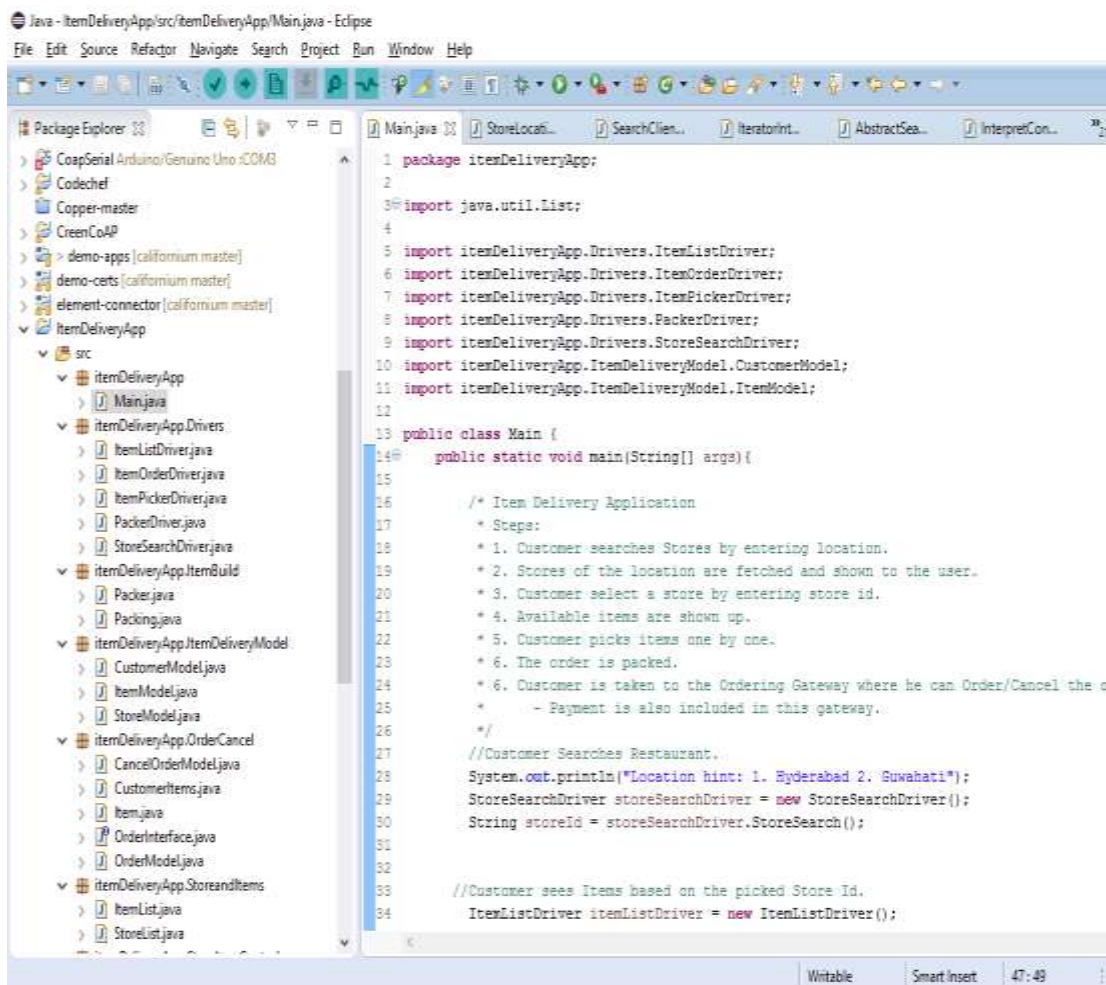
Module 4: Order and Cancellation of items - Command Pattern

Command design pattern is a data driven design pattern which under behavioural design pattern. A request is wrapped under an object as Command and it is being passed to invoker object. Invoker object looks for the appropriate object which can handle the command. I have used this design pattern for ordering or cancelling items for delivery.

Module 5: Order status in Real Time - Observer Pattern

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. It comes under behavioural pattern. This pattern is used for the development of tracking an order in real time because whenever the food delivery states something like "item is packed for delivery," "delivery boy is on the way," "It will be there in two minutes," "item delivered" or "item cancelled," the user will be notified automatically.

Code Implementation:



```
Java - ItemDeliveryApp/src/ItemDeliveryApp/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  CoapSerial Arduino/Genuino Uno :COM3
  Codechef
  Copper-master
  CreenCoAP
  demo-apps [californium master]
  demo-certs [californium master]
  element-connector [californium master]
  ItemDeliveryApp
    src
      itemDeliveryApp
        Main.java
      itemDeliveryApp.Drivers
        ItemListDriver.java
        ItemOrderDriver.java
        ItemPickerDriver.java
        PackerDriver.java
        StoreSearchDriver.java
      itemDeliveryApp.ItemBuild
        Packer.java
        Packing.java
      itemDeliveryApp.ItemDeliveryModel
        CustomerModel.java
        ItemModel.java
        StoreModel.java
      itemDeliveryApp.OrderCancel
        CancelOrderModel.java
        CustomerItems.java
        Item.java
        OrderInterface.java
        OrderModel.java
      itemDeliveryApp.StoreandItems
        ItemList.java
        StoreList.java

Main.java
1 package itemDeliveryApp;
2
3 import java.util.List;
4
5 import itemDeliveryApp.Drivers.ItemListDriver;
6 import itemDeliveryApp.Drivers.ItemOrderDriver;
7 import itemDeliveryApp.Drivers.ItemPickerDriver;
8 import itemDeliveryApp.Drivers.PackerDriver;
9 import itemDeliveryApp.Drivers.StoreSearchDriver;
10 import itemDeliveryApp.ItemDeliveryModel.CustomerModel;
11 import itemDeliveryApp.ItemDeliveryModel.ItemModel;
12
13 public class Main {
14     public static void main(String[] args){
15
16         /* Item Delivery Application
17          * Steps:
18          * 1. Customer searches Stores by entering location.
19          * 2. Stores of the location are fetched and shown to the user.
20          * 3. Customer select a store by entering store id.
21          * 4. Available items are shown up.
22          * 5. Customer picks items one by one.
23          * 6. The order is packed.
24          * 6. Customer is taken to the Ordering Gateway where he can Order/Cancel the or
25          *    - Payment is also included in this gateway.
26          */
27         //Customer Searches Restaurant.
28         System.out.println("Location hint: 1. Hyderabad 2. Gwahati");
29         StoreSearchDriver storeSearchDriver = new StoreSearchDriver();
30         String storeId = storeSearchDriver.StoreSearch();
31
32         //Customer sees Items based on the picked Store Id.
33         ItemListDriver itemListDriver = new ItemListDriver();
34     }
35 }
```

Figure 1.1: Snapshot of main program of the Item Delivery Application.