Contents

[CoolShop 2](#_Toc143029254)

[Domain driven development 2](#_Toc143029255)

[Factory Method 2](#_Toc143029256)

[Mediator Pattern 2](#_Toc143029257)

[Strategy Pattern, Behavioral pattern 2](#_Toc143029258)

[Repository 3](#_Toc143029259)

[Adapter Pattern 3](#_Toc143029260)

[Lazy Loading 3](#_Toc143029261)

[Rip 3](#_Toc143029262)

[Simple factory, Creational Pattern 3](#_Toc143029263)

[IOC, Inversion of Control 3](#_Toc143029264)

[Unit of work 3](#_Toc143029265)

[Decorator 3](#_Toc143029266)

[Iterator Pattern 3](#_Toc143029267)

[Prototype 3](#_Toc143029268)

[Memento 3](#_Toc143029269)

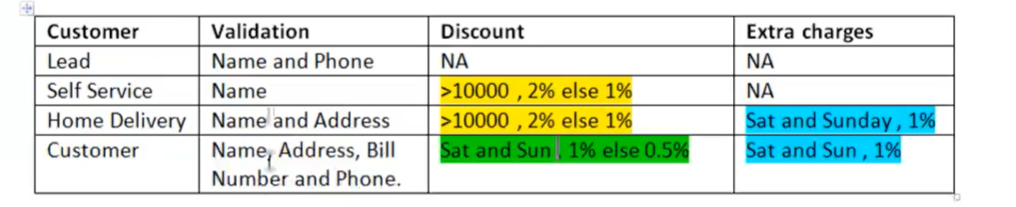
[Ado Connection Example 4](#_Toc143029270)

# Type of Desing Pattern

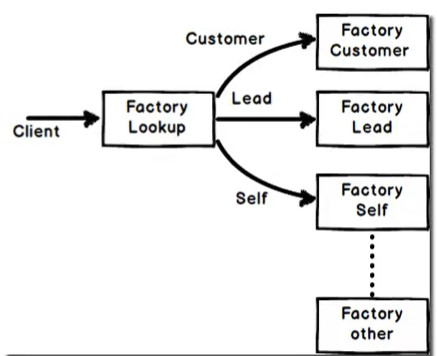
Creational, Structural and Behavioral Patterns

# CoolShop

Customer login



Object creation



# Domain driven development

Follow the vocabulary of you end user (ubiquitous language)

# Factory Method

The Factory Method defines an interface for creating objects, but lets subclasses decide which classes to instantiate

# Mediator Pattern

Handle all the communication between difference components.

# Strategy Pattern, Behavioral pattern

This is a behavioral pattern witch help to select algorithms on runtime. It chooses algorithms dynamically.

# Repository

It creates a commons interface, a common base point for creating a differences type of concrete data access methods.

# Adapter Pattern

Helps to work in a unified way when we have incompatible interfaces from tree party. For example, we retrieved data in Json and xml format but we mapped this both types into a common object.

# Lazy Loading

Load static information just ones.

# Rip

Replacing If condition with polymorphism

# Simple factory, Creational Pattern

Create a new object in a difference class.

# IOC, Inversion of Control

Solid Principle, take this unnecessary logic for this class and put it in another place.

# Unit of work

Solve the problems of repository pattern, create a single transaction for difference repositories

# Decorator

Adding behavior by using separate class

# Iterator Pattern

When just need to iterate over a list, do not allow add, remove end other method, just iterate. **Enumerable or Enumerator** encapsulate ADD, REMOVE methods.

# Prototype

Create a new copy of the object (Clone an object)

# Memento

To reverse back to any old estate of one object

# Ado Connection Example

