# Online Marketplace

## Akhil Nayabu

## 2000075395

## Assignment #2

## Course Number: CSCI 50700 – 23706

## Course Name: Object-Oriented Design and Programming

### Overview

The main motive of assignment #2 is to implement application control pattern to the MVC architecture that we designed in assignment #1. The pattern we will be looking at are Front controller to dispatch respective views depending on based on which user logged-in (customer/admin). We will also use Abstract factory pattern and Command pattern to achieve these in our assignment #2.

### Assignment #1 Feedback

* No Comments to address.

### Domain Model

A domain model illustrates meaningful conceptual classes in problem domain. The following diagram illustrates the domain model for this assignment. It lists all the conceptual classes that will be used to build the framework for this assignment.

No changes to domain model from assignment #1

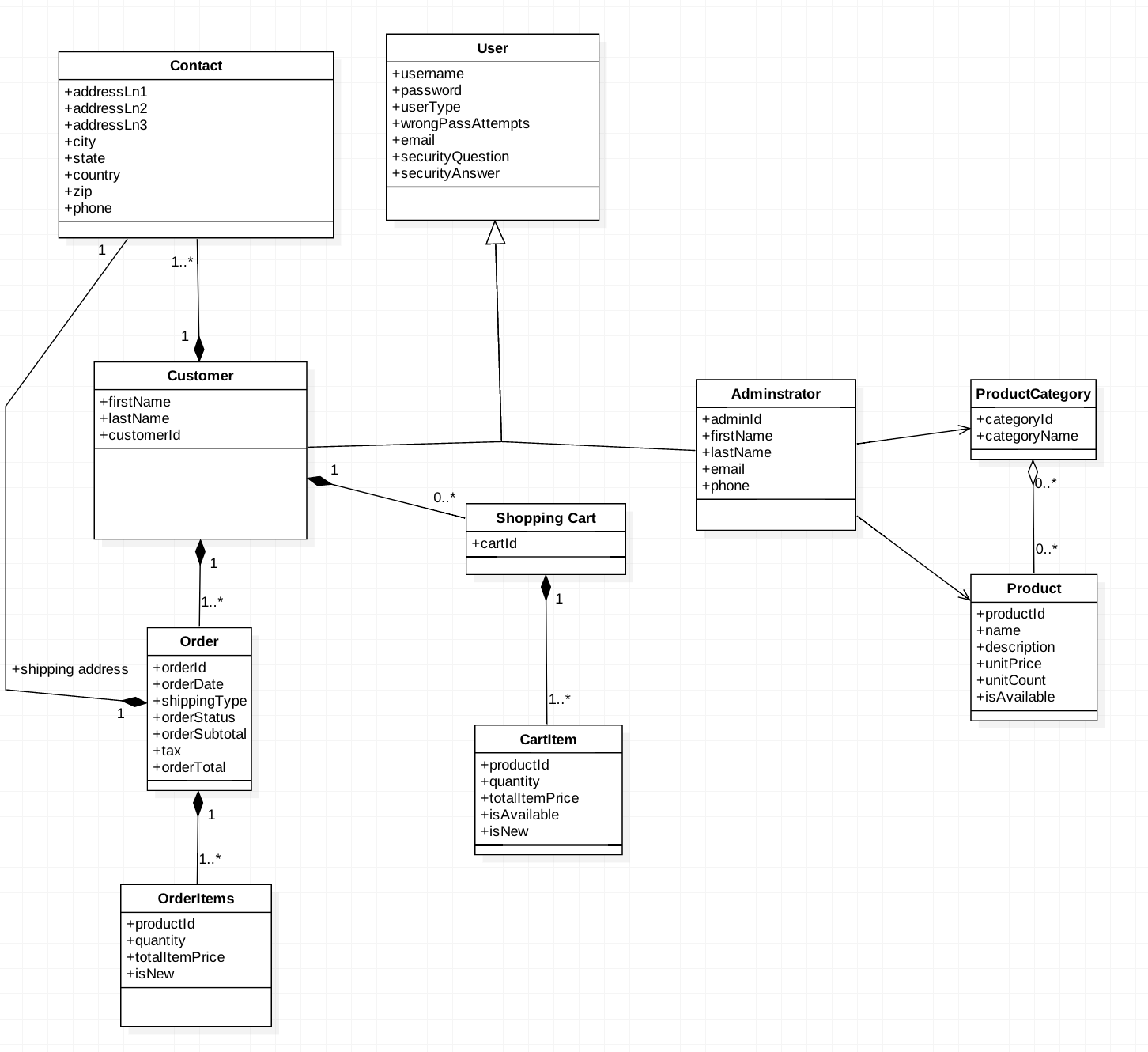


Fig 1: Domain model

### Front Controller Pattern

Front controller is a single handler which will be used to process all the requests. We will use front controller to authenticate and dispatch respective view based on which user logged in (customer/admin).

Components for this pattern are:  
-Front Controller  
-Dispatcher  
-View

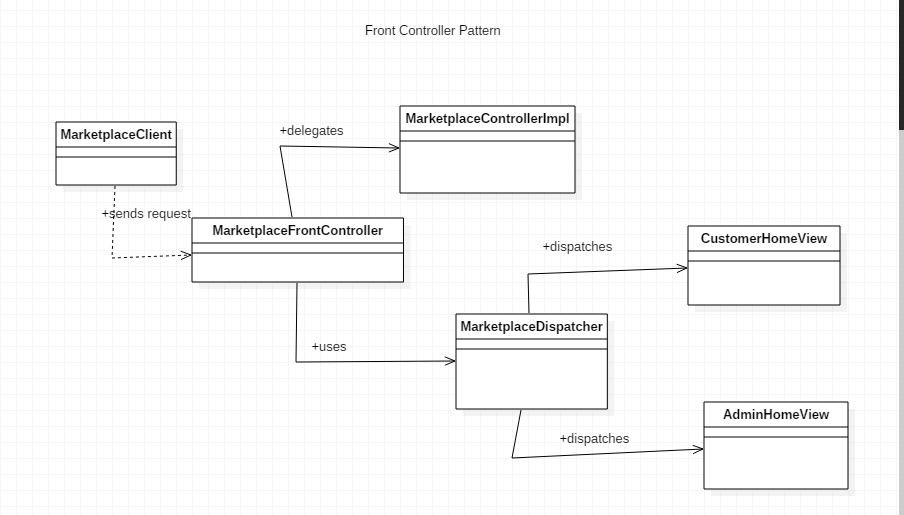


Fig 2: Front Controller

In our application client passes username and password to authenticate method of front controller and front controller calls handleLogin method in application controller (i.e. MarketplaceControllerImpl) where authentication is done and returns the account details to front-controller from where account type is sent is as request to dispatcher and dispatcher will decide which view is to be dispatched.

### Abstract Factory Pattern

This is a creational pattern, used to create objects. Definition according to Gang of four book “Provides an interface for creating families of related or dependent objects without specifying their concrete classes.” AbstractFactory defines an interface that is implemented by all the concrete factories.

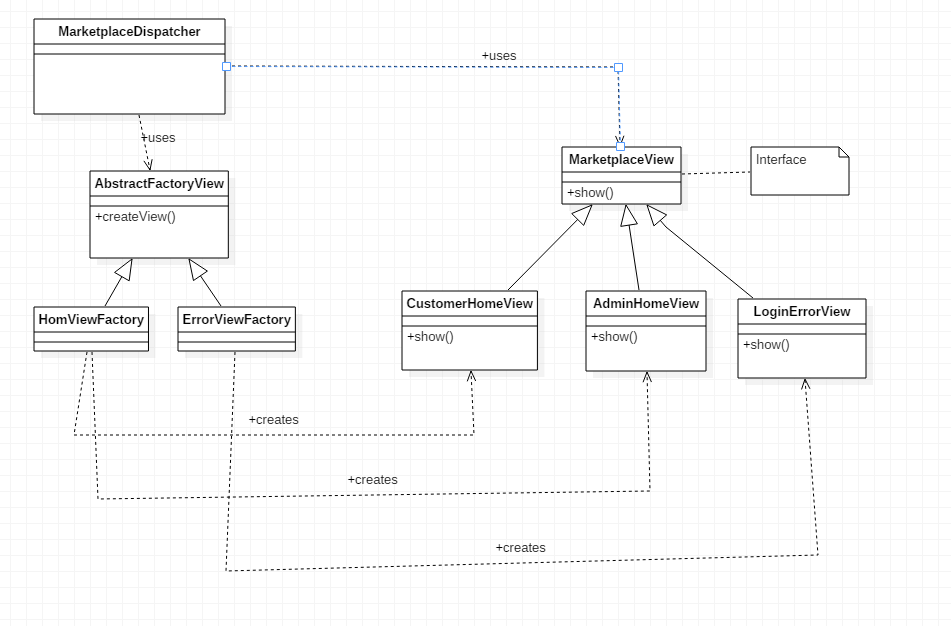


Fig 3: Abstract Factory

In our application dispatcher uses AbstractFactoryView to create views based on the account type that has been dispatched. Abstract factory in here has 2 concrete factories: 1.HomeViewFactory 2.ErrorViewFactory. Based on the request object of respective concrete class is generated. And based on account type concrete factory decides which type of view is to be called.

### Command Pattern

Command Pattern is a behavioral pattern. In this pattern a request is wrapped with an object which server as a command and passed to invoker object.

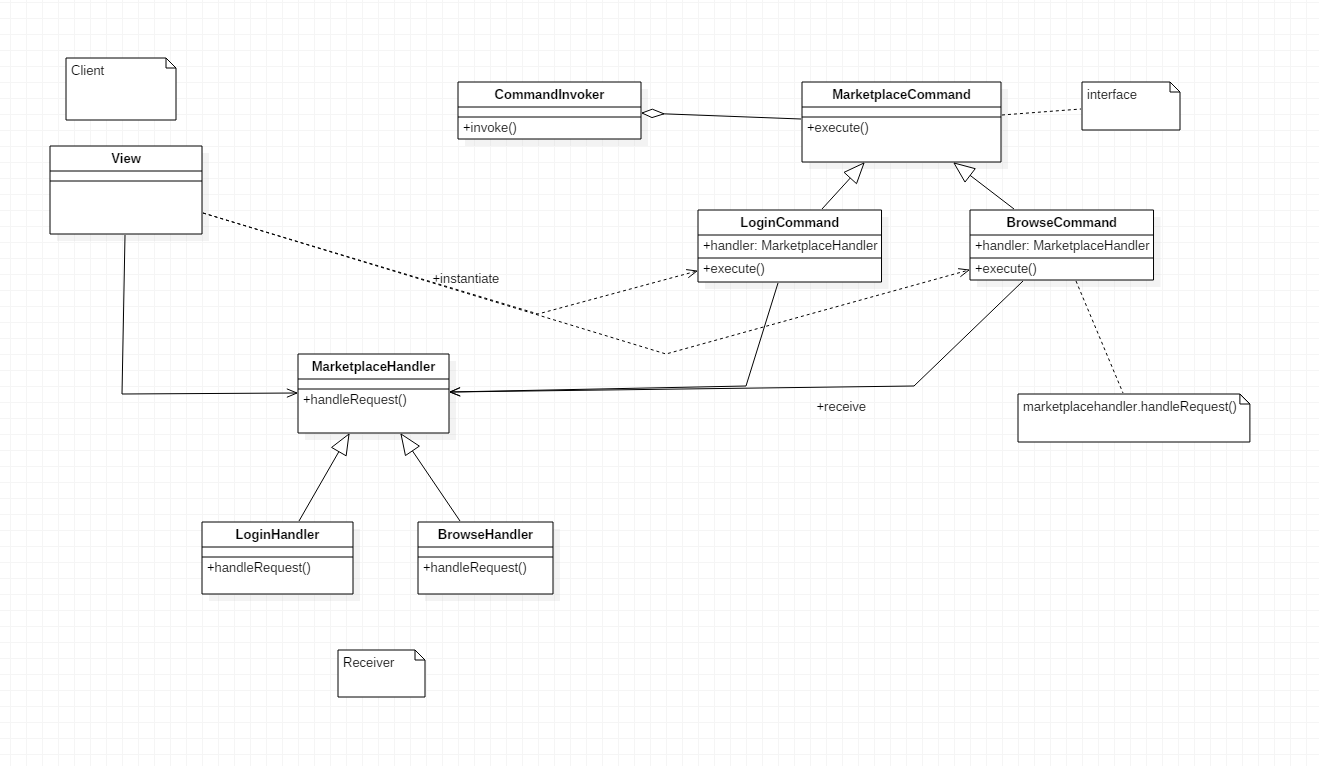
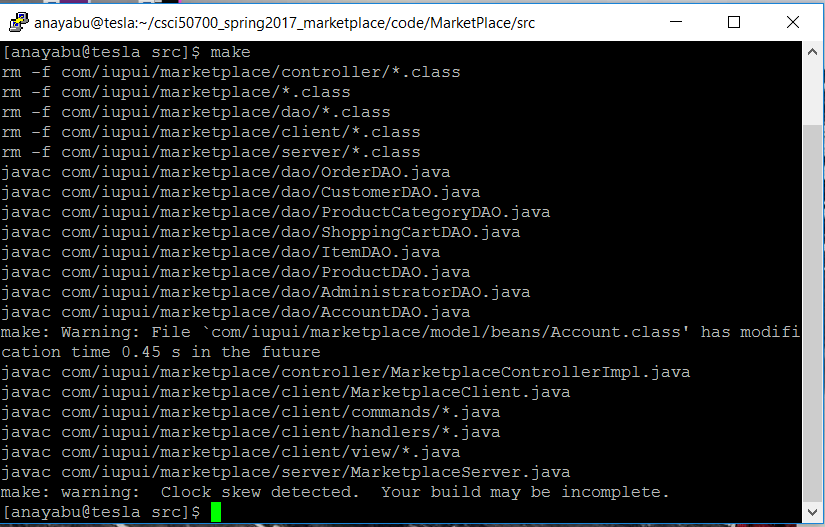


Fig 4: Command Pattern

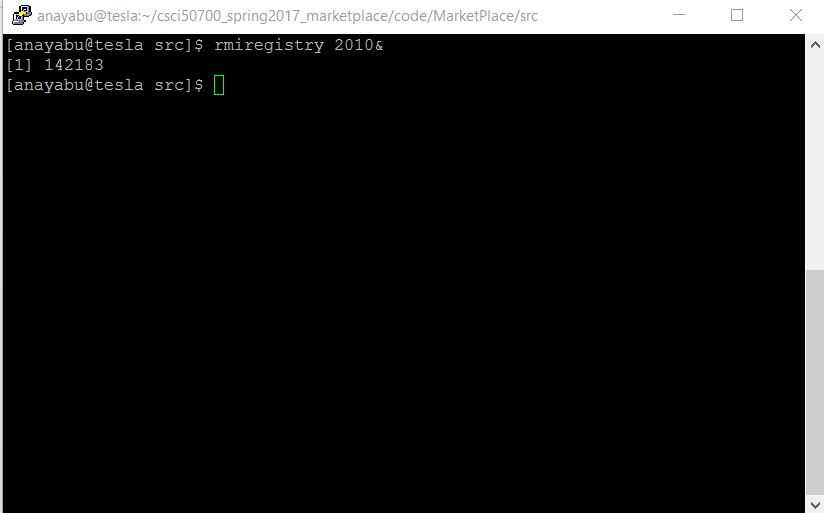
MarketplaceCommand is the interface which acts as a command. MarketplaceHandler acts as a requester. We will use LoginHandler and BrowseHanlder which will process the command. CommandInvoker servers as an invoker object. It will invoke the commands. CommandInvoker object uses client to identify which oject will execute which command.

Sample runs:

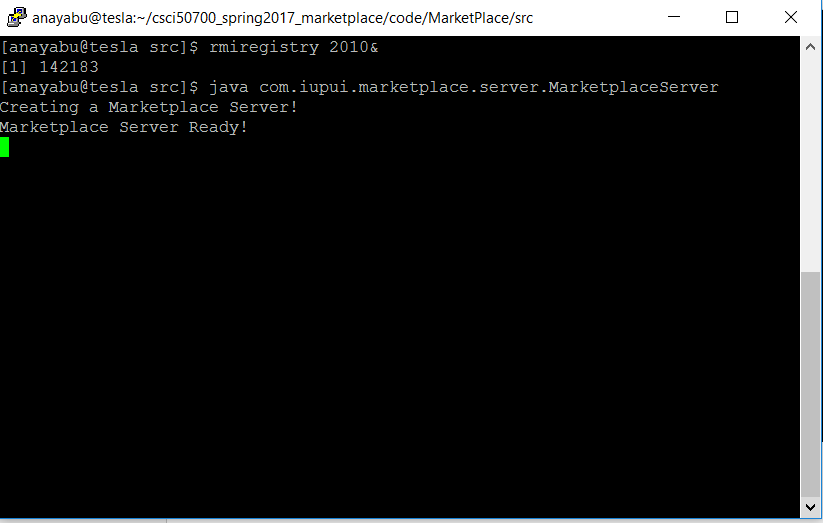
Compilation:



RMIREGISTRY



Server:



Client:

