53 Restaurant Ordering System

Kristopher Neilsen
Samuel Steward
Chandler Baggett

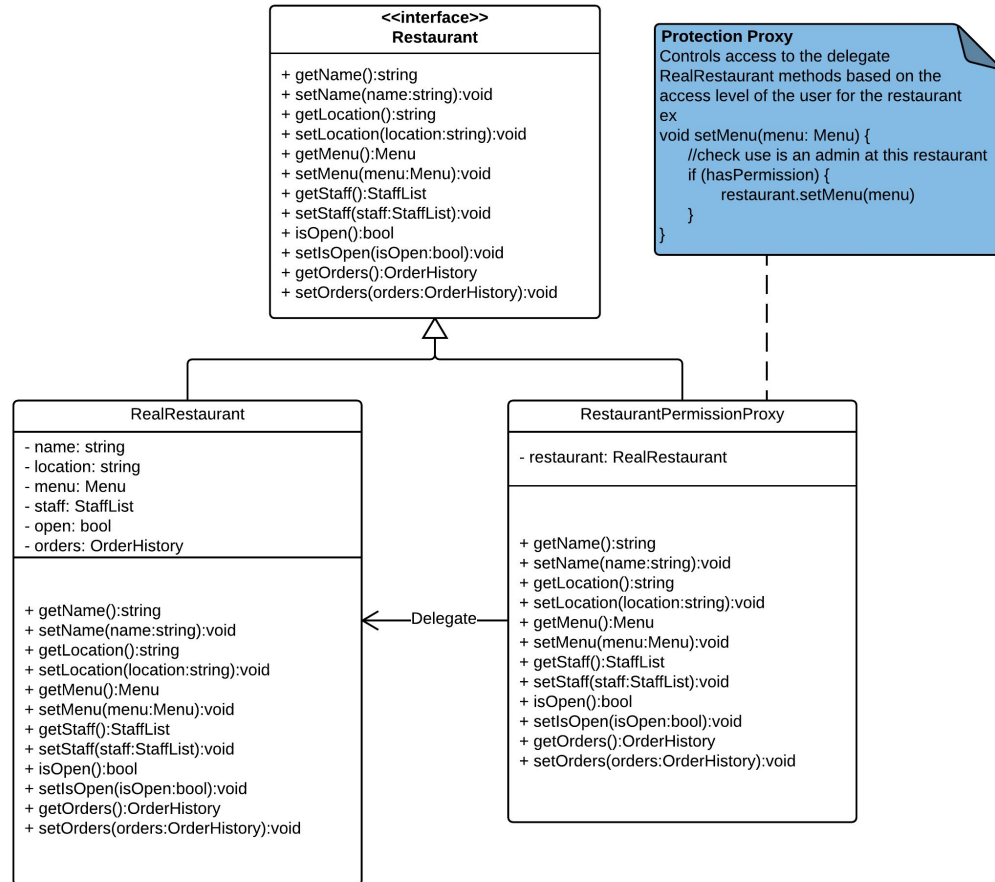CSCI 5448 Spring 2017

Vision

A system that customers can use to view a restaurant menu, select items for an order and place the order to the restaurant staff.
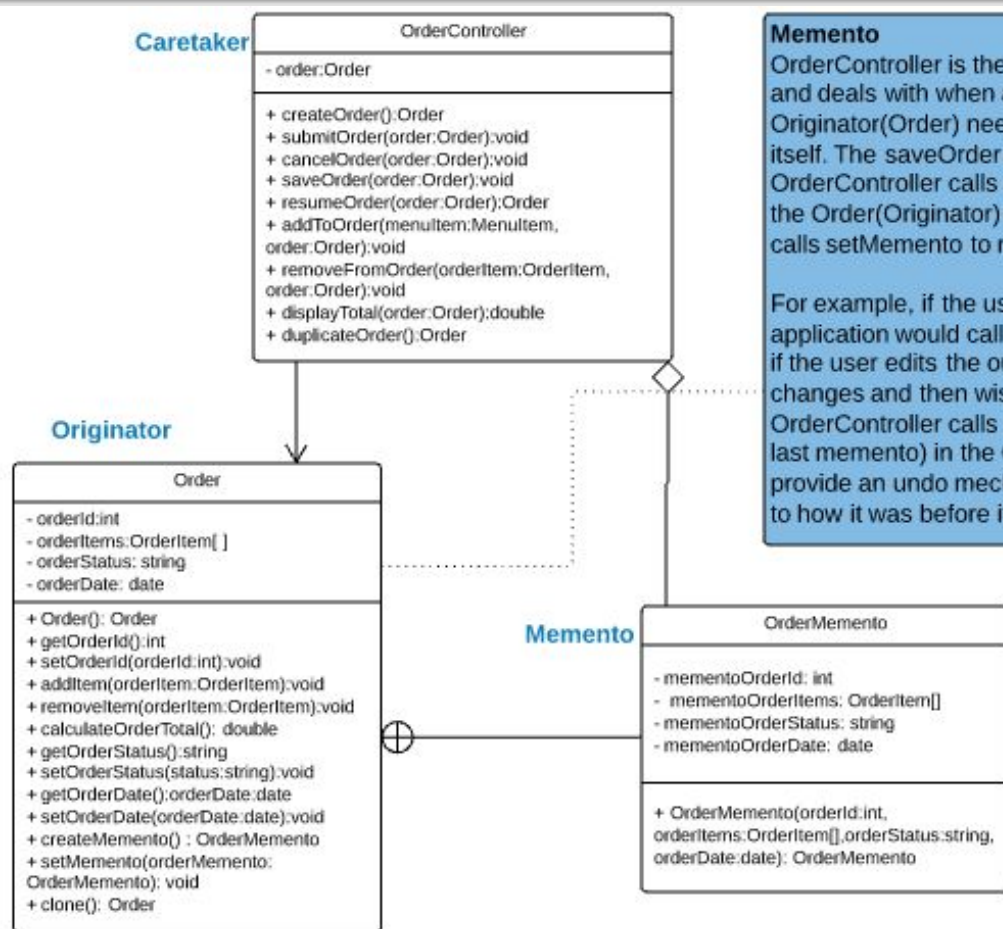
Technology Used

- Spring MVC
- Hibernate
- Mysql database

Live Demo (Use Cases)

Proxy Design Pattern

<<interface>>
Restaurant

+ getName():string
+ setName(name:string):void
+ getLocation():string
+ setLocation(location:string):void
+ getMenu():Menu
+ setMenu(menu:Menu):void
+ getStaff():StaffList
+ setStaff(staff:StaffList):void
+ isOpen():bool
+ setIsOpen(isOpen:bool):void
+ getOrders():OrderHistory
+ setOrders(orders:OrderHistory):void

Protection Proxy
Controls access to the delegate
RealRestaurant methods based on the
access level of the user for the restaurant
ex
```
void setMenu(menu: Menu) {
    //check use is an admin at this restaurant
    if (hasPermission) {
        restaurant.setMenu(menu)
    }
}
```

RealRestaurant

- name: string
- location: string
- menu: Menu
- staff: StaffList
- open: bool
- orders: OrderHistory

+ getName():string
+ setName(name:string):void
+ getLocation():string
+ setLocation(location:string):void
+ getMenu():Menu
+ setMenu(menu:Menu):void
+ getStaff():StaffList
+ setStaff(staff:StaffList):void
+ isOpen():bool
+ setIsOpen(isOpen:bool):void
+ getOrders():OrderHistory
+ setOrders(orders:OrderHistory):void

RestaurantPermissionProxy

- restaurant: RealRestaurant

+ getName():string
+ setName(name:string):void
+ getLocation():string
+ setLocation(location:string):void
+ getMenu():Menu
+ setMenu(menu:Menu):void
+ getStaff():StaffList
+ setStaff(staff:StaffList):void
+ isOpen():bool
+ setIsOpen(isOpen:bool):void
+ getOrders():OrderHistory
+ setOrders(orders:OrderHistory):void

Delegate

Memento Design Pattern

Caretaker

OrderController

- order:Order

+ createOrder():Order
+ submitOrder(order:Order):void
+ cancelOrder(order:Order):void
+ saveOrder(order:Order):void
+ resumeOrder(order:Order):Order
+ addToOrder(menuItem:MenuItem, order:Order):void
+ removeFromOrder(orderItem:OrderItem, order:Order):void
+ displayTotal(order:Order):double
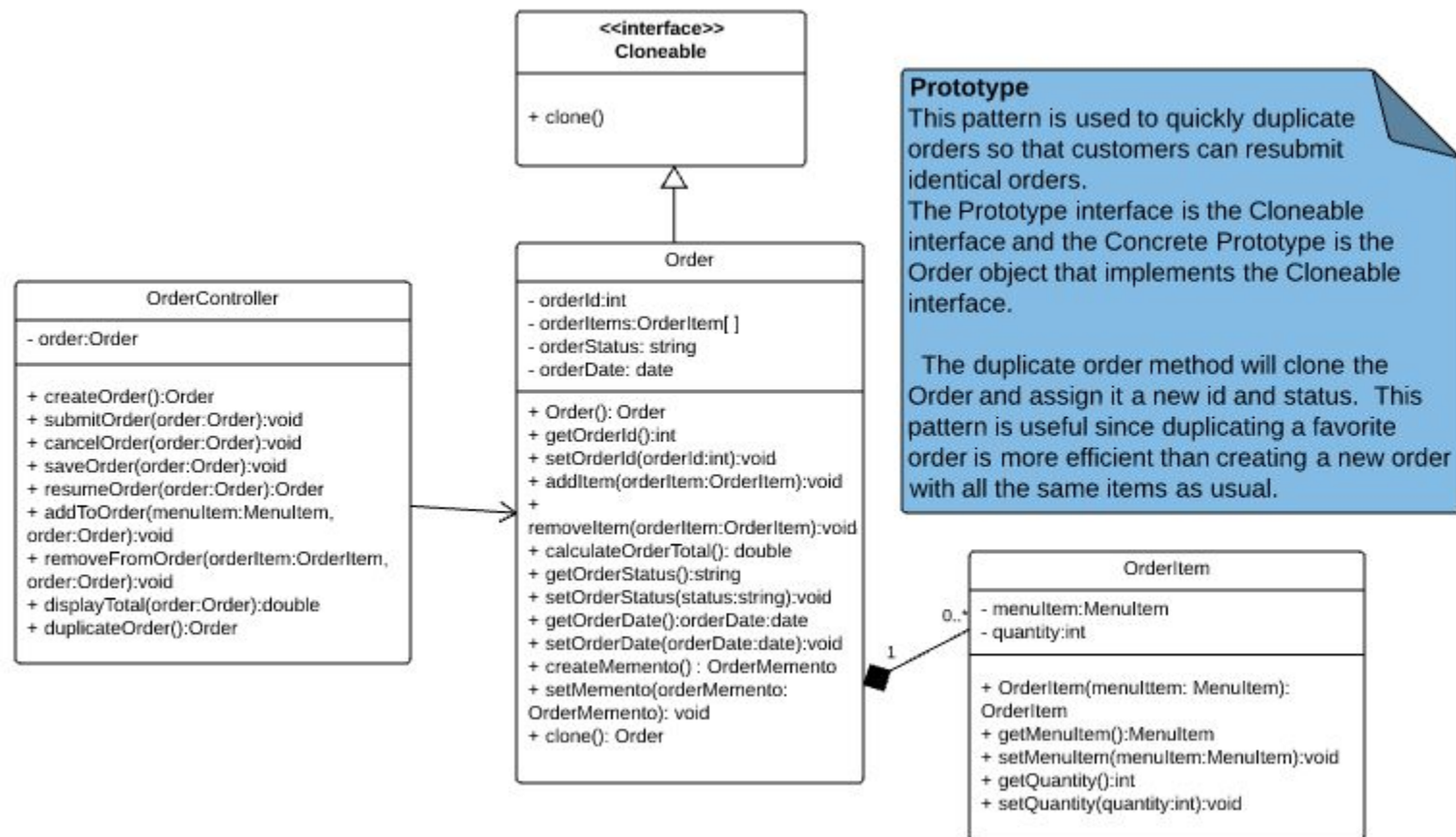+ duplicateOrder():Order

Memento

Memento
OrderController is the Caretaker object and deals with when and why the Originator(Order) needs to save or restore itself. The saveOrder method in the OrderController calls createMemento method in the Order(Originator) class and resumeOrder calls setMemento to restore the Order.

For example, if the user selects to Save then the application would call createMemento(save) and if the user edits the order further without saving changes and then wishes to restore, then the OrderController calls the setMemento(to restore last memento) in the Order object in order to provide an undo mechanism to restore the Order to how it was before it was modified.

Originator

Order

- orderId:int
- orderItems:OrderItem[ ]
- orderStatus: string
- orderDate: date

+ Order(): Order
+ getOrderId():int
+ setOrderId(orderId:int):void
+ addItem(orderItem:OrderItem):void
+ removeItem(orderItem:OrderItem):void
+ calculateOrderTotal(): double
+ getOrderStatus():string
+ setOrderStatus(status:string):void
+ getOrderDate():orderDate:date
+ setOrderDate(orderDate:date):void
+ createMemento() : OrderMemento
+ setMemento(orderMemento: OrderMemento): void
+ clone(): Order

Memento

OrderMemento

- mementoOrderId: int
- mementoOrderItems: OrderItem[]
- mementoOrderStatus: string
- mementoOrderDate: date

+ OrderMemento(orderId:int, orderItems:OrderItem[],orderStatus:string, orderDate:date): OrderMemento

Prototype Design Pattern

<<interface>>
Cloneable

+ clone()

Prototype
This pattern is used to quickly duplicate orders so that customers can resubmit identical orders.
The Prototype interface is the Cloneable interface and the Concrete Prototype is the Order object that implements the Cloneable interface.

 The duplicate order method will clone the Order and assign it a new id and status. This pattern is useful since duplicating a favorite order is more efficient than creating a new order with all the same items as usual.

OrderController

- order:Order

+ createOrder():Order
+ submitOrder(order:Order):void
+ cancelOrder(order:Order):void
+ saveOrder(order:Order):void
+ resumeOrder(order:Order):Order
+ addToOrder(menuItem:MenuItem, order:Order):void
+ removeFromOrder(orderItem:OrderItem, order:Order):void
+ displayTotal(order:Order):double
+ duplicateOrder():Order

Order

- orderId:int
- orderItems:OrderItem[ ]
- orderStatus: string
- orderDate: date

+ Order(): Order
+ getOrderId():int
+ setOrderId(orderId:int):void
+ addItem(orderItem:OrderItem):void
+ removeItem(orderItem:OrderItem):void
+ calculateOrderTotal(): double
+ getOrderStatus():string
+ setOrderStatus(status:string):void
+ getOrderDate():orderDate:date
+ setOrderDate(orderDate:date):void
+ createMemento() : OrderMemento
+ setMemento(orderMemento: OrderMemento): void
+ clone(): Order

OrderItem

- menuItem:MenuItem
- quantity:int

+ OrderItem(menuItem: MenuItem): OrderItem
+ getMenuItem():MenuItem
+ setMenuItem(menuItem:MenuItem):void
+ getQuantity():int
+ setQuantity(quantity:int):void

0..*

1

Link to Recorded Demo

Group GitHub

https://github.com/chandlerbaggett/5448Project

Demo:

https://github.com/chandlerbaggett/5448Project/blob/master/53_Restaurant OrderingSystem_Video_Demo.webm