OO Design Examples

Mikael Svahnberg*

2023-02-01

Contents

1 <2023-02-01 ons> BurgerOrderer

2 Use Case Order Food

Use Case Order Food **Actors** Customer **Description** A customer arrives at the BurgerOrderer, selects a meal and configures it. **Related Use Case** Pay for Order

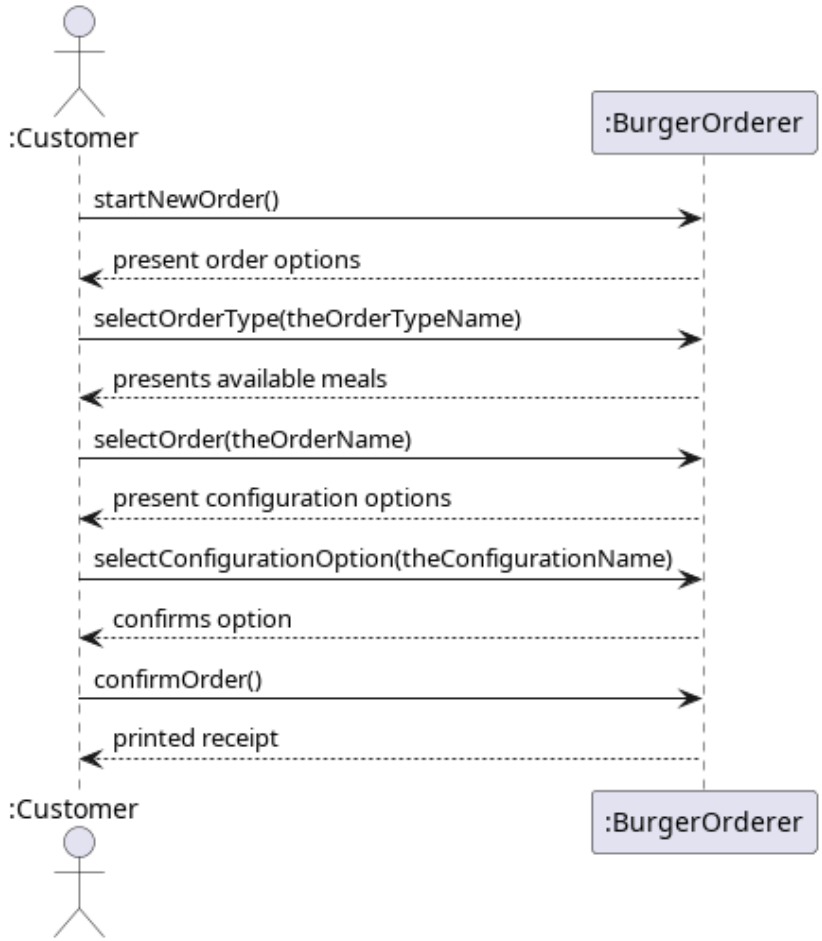
Main Course of Events

*Mikael.Svahnberg@bth.se

Actor	System
1. Customer arrives at BurgerOrderer and starts a new order	
	2. System presents order options [single burger, meal, dessert, drink]
3. Customer selects “meal”	
	4. System presents available meals
5. Customer selects a specific meal.	
	6. System adds meal to current order.
	7. System presents configuration options fo
8. Customer selects “no onions” and “more bacon”	
	9. System adds “no onions” and “more bac
10. Customer confirms order.	
	11. System initiates use case pay for order
	12. System places order to kitchen and pri

3 System Sequence Diagram

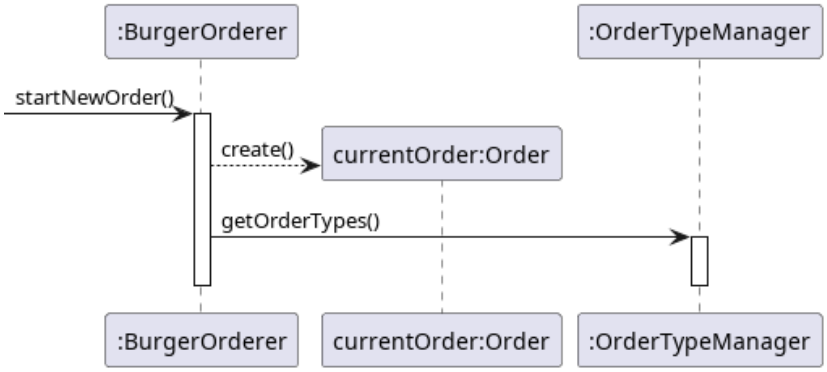
```
actor ":Customer" as cus
participant ":BurgerOrderer" as sys

cus -> sys : startNewOrder()
sys --> cus : present order options

cus -> sys : selectOrderType(theOrderTypeName)
sys --> cus : presents available meals

cus -> sys : selectOrder(theOrderName)
sys --> cus : present configuration options

cus -> sys : selectConfigurationOption(theConfigurationName)
sys --> cus : confirms option

cus -> sys : confirmOrder()
sys --> cus : printed receipt
```



4 Interaction Diagrams (Sequence Diagrams)

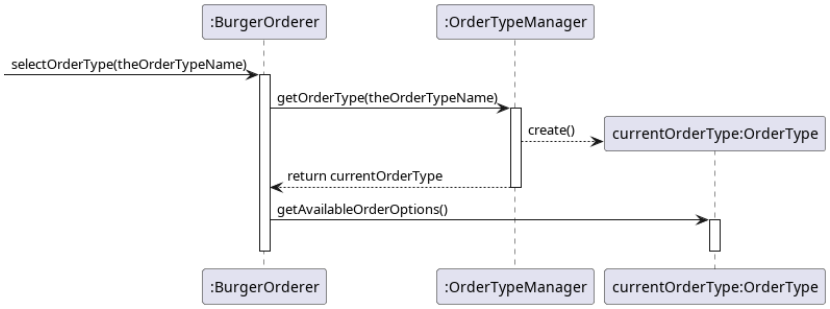
4.1 startNewOrder()

```
[-> ":BurgerOrderer" : startNewOrder()
activate ":BurgerOrderer"

create participant "currentOrder:Order"
":BurgerOrderer" --> "currentOrder:Order" : create()

":BurgerOrderer" -> ":OrderTypeManager" : getOrderTypes()
activate ":OrderTypeManager"
deactivate ":OrderTypeManager"

deactivate ":BurgerOrderer"
```



4.2 selectOrderType()

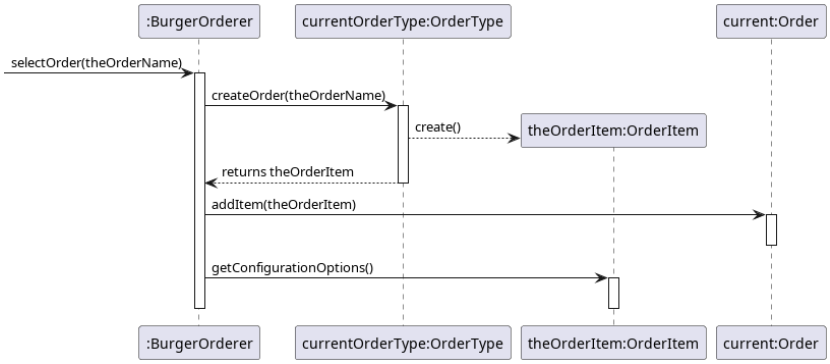
```
[-> ":BurgerOrderer" : selectOrderType(theOrderTypeName)
activate ":BurgerOrderer"

":BurgerOrderer" -> ":OrderTypeManager" : getOrderType(theOrderTypeName)
activate ":OrderTypeManager"
create participant "currentOrderType:OrderType"
":OrderTypeManager" --> "currentOrderType:OrderType" : create()
":OrderTypeManager" --> ":BurgerOrderer" : return currentOrderType
deactivate ":OrderTypeManager"

":BurgerOrderer" -> "currentOrderType:OrderType" : getAvailableOrderOptions()
activate "currentOrderType:OrderType"
deactivate "currentOrderType:OrderType"
deactivate ":BurgerOrderer"
```



4.3 selectOrder()

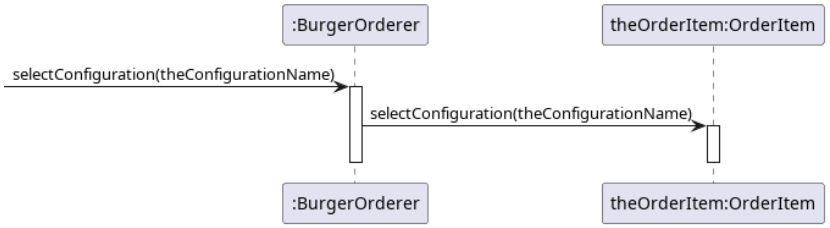
```
[-> ":BurgerOrderer" : selectOrder(theOrderName)
activate ":BurgerOrderer"

":BurgerOrderer" -> "currentOrderType:OrderType" : createOrder(theOrderName)
activate "currentOrderType:OrderType"

create participant "theOrderItem:OrderItem"
```

```
"currentOrderType:OrderType" --> "theOrderItem:OrderItem" : create()
"currentOrderType:OrderType" --> ":BurgerOrderer" : returns theOrderItem

deactivate "currentOrderType:OrderType"

":BurgerOrderer" -> "current:Order" : addItem(theOrderItem)
activate "current:Order"
deactivate "current:Order"

":BurgerOrderer" -> "theOrderItem:OrderItem" : getConfigurationOptions()
activate "theOrderItem:OrderItem"
deactivate "theOrderItem:OrderItem"

deactivate ":BurgerOrderer"
```



4.4 selectConfiguration()

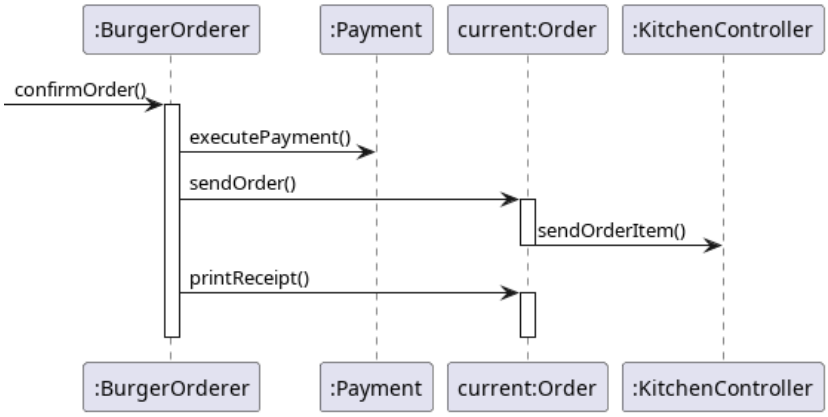
```
[-> ":BurgerOrderer" : selectConfiguration(theConfigurationName)
activate ":BurgerOrderer"
":BurgerOrderer" -> "theOrderItem:OrderItem" : selectConfiguration(theConfigurationName)
activate "theOrderItem:OrderItem"
deactivate "theOrderItem:OrderItem"

deactivate ":BurgerOrderer"
```



4.5 confirmOrder()

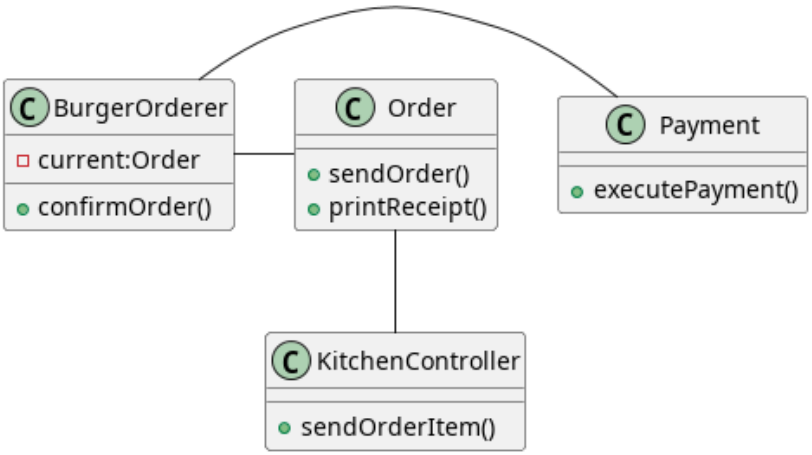
```
[-> ":BurgerOrderer" : confirmOrder()
```

```
activate ":BurgerOrderer"
":BurgerOrderer" -> ":Payment" : executePayment()

":BurgerOrderer" -> "current:Order" : sendOrder()
activate "current:Order"
"current:Order" -> ":KitchenController" : sendOrderItem()
deactivate "current:Order"

":BurgerOrderer" -> "current:Order" : printReceipt()
activate "current:Order"
deactivate "current:Order"


deactivate ":BurgerOrderer"
```



5 Class Diagram

```
BurgerOrderer : +confirmOrder()
BurgerOrderer : -current:Order

Payment : +executePayment()

Order : +sendOrder()
Order : +printReceipt()

KitchenController : +sendOrderItem()

BurgerOrderer - Payment
BurgerOrderer - Order
Order -- KitchenController
```



. . . och så vidare. Man fyller på dett enda klassdiagram med klasser och metoder från alla interaktionsdiagram så att man till slut har ett enda stort klassdiagram för hela systemet och alla use cases.