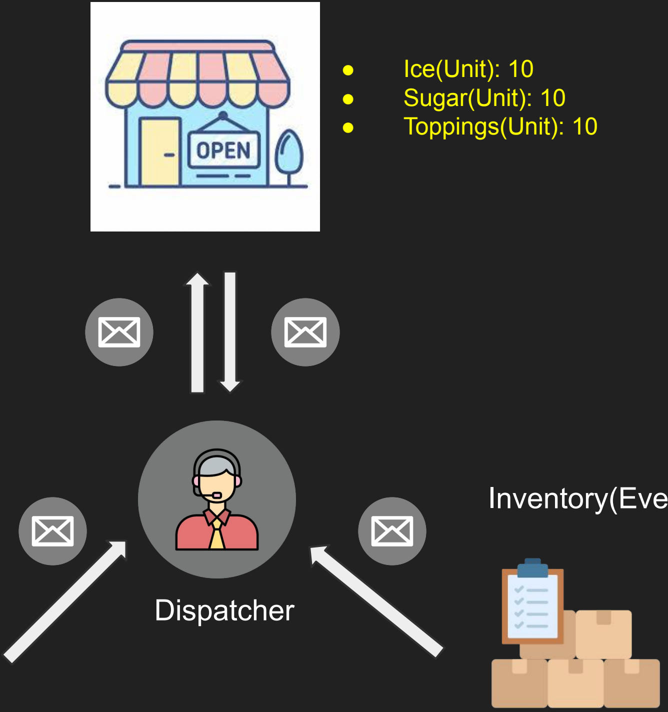
CS665 Final Presentation

- Event Driven Architecture
- Template Method Pattern

Stephen Yang

Event Driven Architecture

- Communicate changes across components of the system through events
- System responds to events through actions
- Loosely-coupled System
- Easy to extend functionality and enhance modularity



Shop

- Ice(Unit): 10
- Sugar(Unit): 10
- Toppings(Unit): 10

Order(Event)

Dispatcher

Inventory(Event)

- Ice(Unit): 2
- Sugar(Unit): 2
- Toppings(Unit): 2

+ Ice(Unit): 5
+ Sugar(Unit): 5
+ Toppings(Unit): 5

Pattern Components

Dispatcher

The central component of the system which dispatches the event.

Event

Occurrences or changes within the system such as data updates, user inputs.
OrderEvent(Class): Event for placing Boba Tea order.
InventoryEvent(Class): Event for reordering right amount of stock.

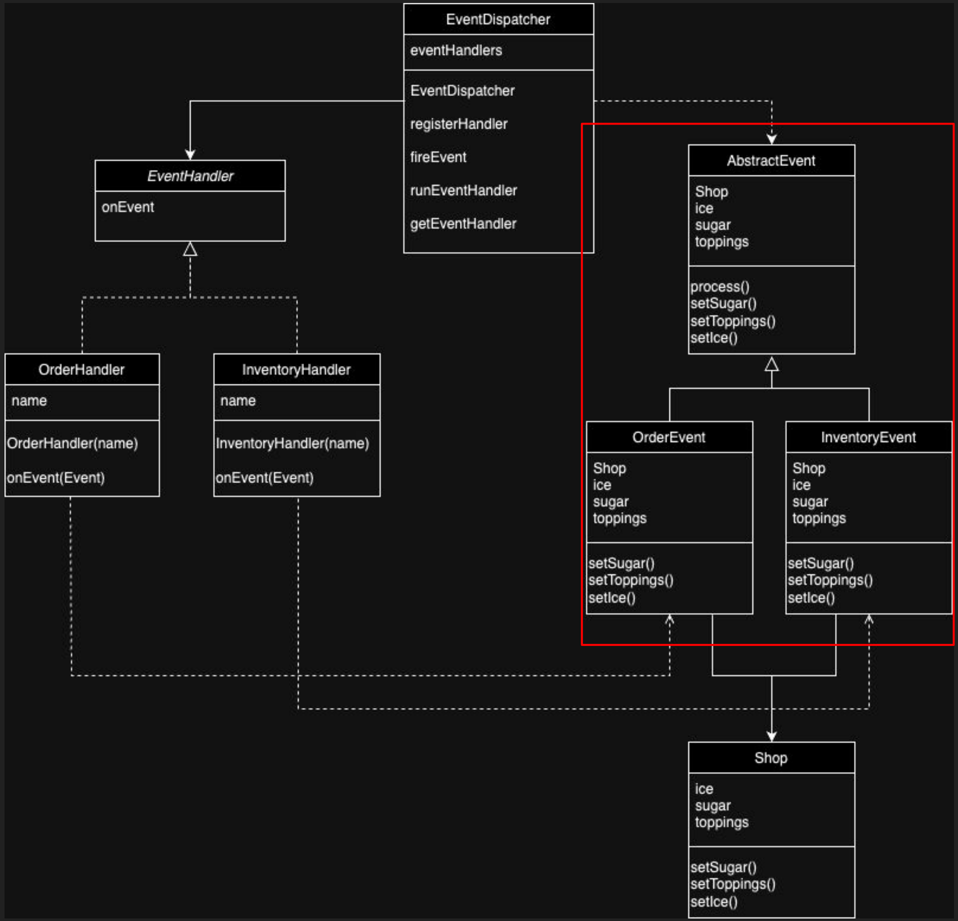
EventHandler

It contains business logic to process the event.
OrderHandler(Class): Event handler for Boba Tea order
InventoryHandler(Class): Event handler for inventory event

Shop

The entity to run the business engaging order and inventory events.

UML Diagram



EventDispatcher
eventHandlers
EventDispatcher
registerHandler
fireEvent
runEventHandler
getEventHandler

EventHandler
onEvent

OrderHandler
name
OrderHandler(name)
onEvent(Event)

InventoryHandler
name
InventoryHandler(name)
onEvent(Event)

AbstractEvent
Shop
ice
sugar
toppings
process()
setSugar()
setToppings()
setIce()

OrderEvent
Shop
ice
sugar
toppings
setSugar()
setToppings()
setIce()

InventoryEvent
Shop
ice
sugar
toppings
setSugar()
setToppings()
setIce()

Shop
ice
sugar
toppings
setSugar()
setToppings()
setIce()

Template Method Pattern:🟥
A skeleton of operation in the superclass.
It allows subclasses to overwrite the
methods without changing the structure.