TrackerViewHelper

User Interface

Creates and uses these

<<interface>>
Observer

+ update()

<<interface>>
Subject

+ addSubscription(observer: Observer)
+ removeSubscription(observer: Observer)
+ nofityObservers(shipment: Shipment)

Assumption: The strategy pattern can be implemented with regards to the type of algorithm used for shipments, such as creating and changing the state of it.

Static class

you will want addShipment to be public because it will make doing the strategy pattern a little easier

runSimulation reads the file and handles the updates to the corresponding shipment. You can probably just call it when the app launches.

First assumptions for the observer pattern: The TrackerViewHelper is a type of observer, and when we click 'track' on our GUI, it creates a new instance of the TrackerViewHelper, calls trackShipment with the expected ID (subscribes to a Shipment), and will be notified of any updates through this subscription.

TrackerViewHelper

+ shipmentId: State<String>
+ shipmentNotes: State<String[]>
+ shipmentUpdateHistory: State<String[]>
+ expectedShipmentDeliveryDate: State<String[]>
+shipmentStatus: State<String>
+location: State<String>

*all attributes have a private setter in this class

+ trackShipment(id: String)
+ stopTracking()

Notifies this of changes

TrackingSimulator

- shipments: Shipment[]
- observers: Observers[]
- updates: map<String, Update>

+ findShipment(id: String): Shipment?
+ addShipment(shipment: Shipment)
+ runSimulation()

1

1

0..*

TrackerViewHelper has the responsibility of preparing data for the using interface to display

When the user clicks on the shipment it is already tracking to stop tracking, stopTracking inside of the specific TrackerViewHelper is called, which notifies the Shipment that we do not need to be notified anymore of changes, and then once that method is complete, we deleted the specific ViewHelper for that Shipment and thus it is removed from the GUI.

Is it supposed to be of type Update or ShippingUpdate for addUpdate?

Shipment

+ status: String
+ id: String
+ notes: String[]
private set;
+ updateHistory: ShippingUpdate[]
private set;
+ expectedDeliveryDateTimeStamp: Long
+ currentLocation: String

+ addNote(note: String)
+ addUpdate(update: Update)

ShippingUpdate

+ previousStatus: String
+ newStatus: String
+ timestamp: Long

8

<<interface>>
Update

+ performUpdate()

Assumption: The observer pattern will be useful for notifying (from the TrackingSimulator) all of the needed areas that a shipment has changed

NOTE: Might remove Create from the update interface. This is because it feels as if create is not an update and should be stand alone

Create

Delayed

Shipped

Lost

Location

Canceled

Delivered

NoteAdded