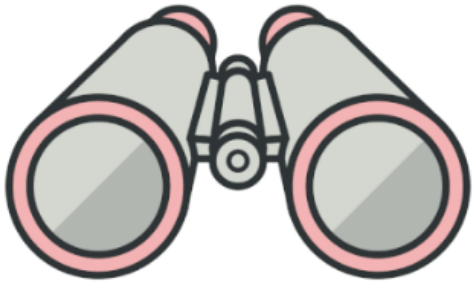
CSci 3081W: Program Design and Development

Lecture 11 – Observer Pattern

Behavioral Design Patterns

Behavioral design patterns are concerned with algorithms and the assignment of responsibilities between objects.

Behavioral Design Patterns

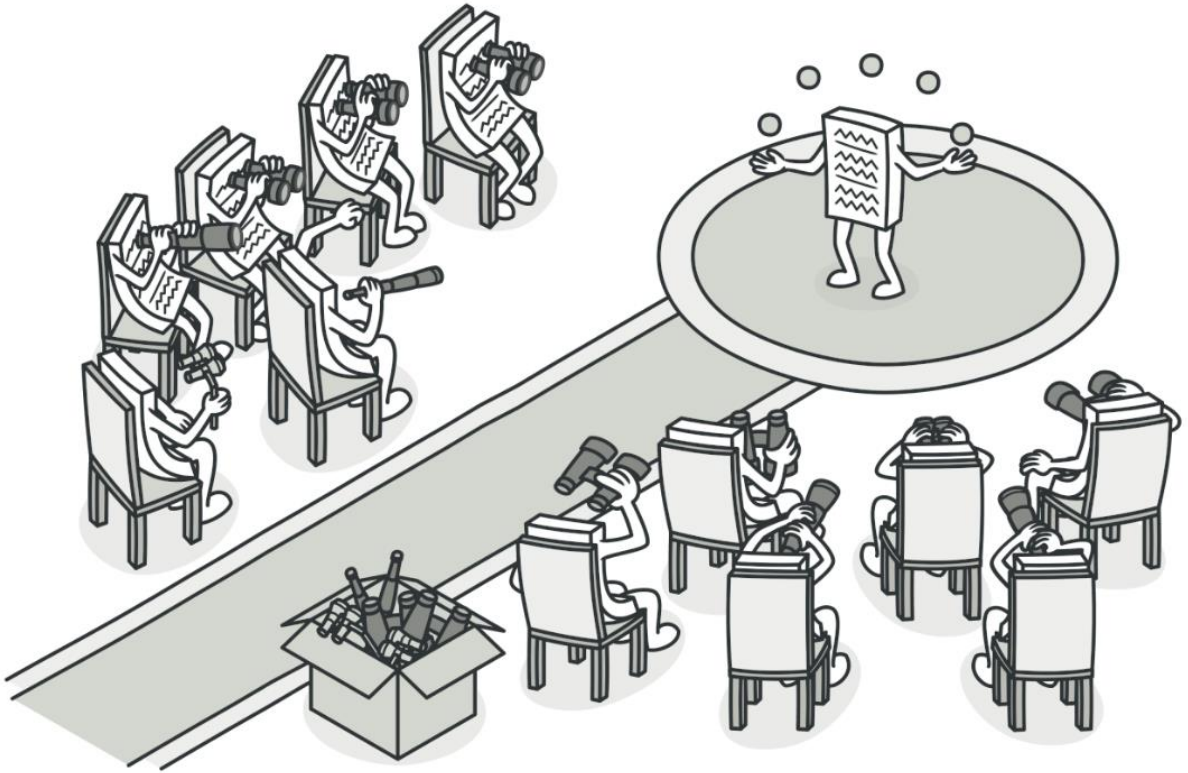


Observer

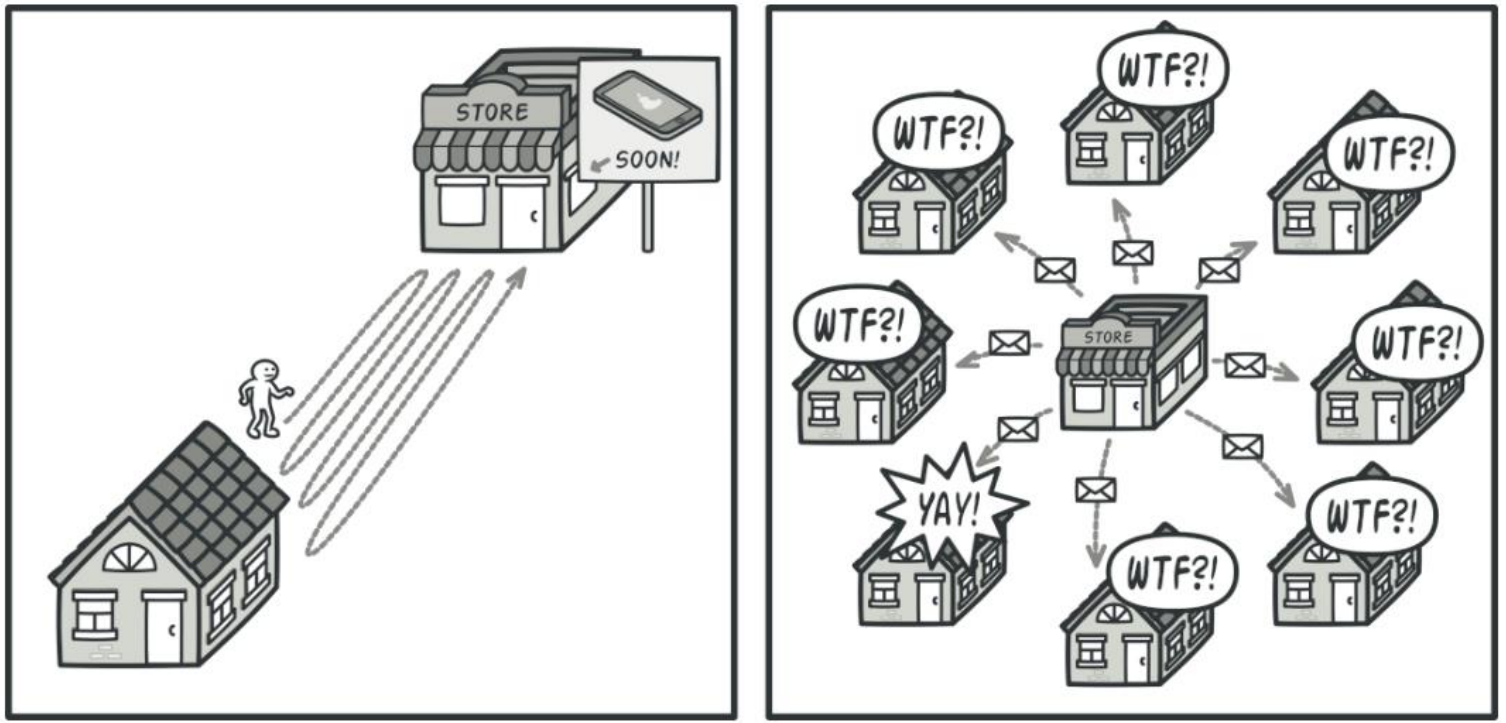
Lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing.

Observer Pattern

Observer is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they’re observing.

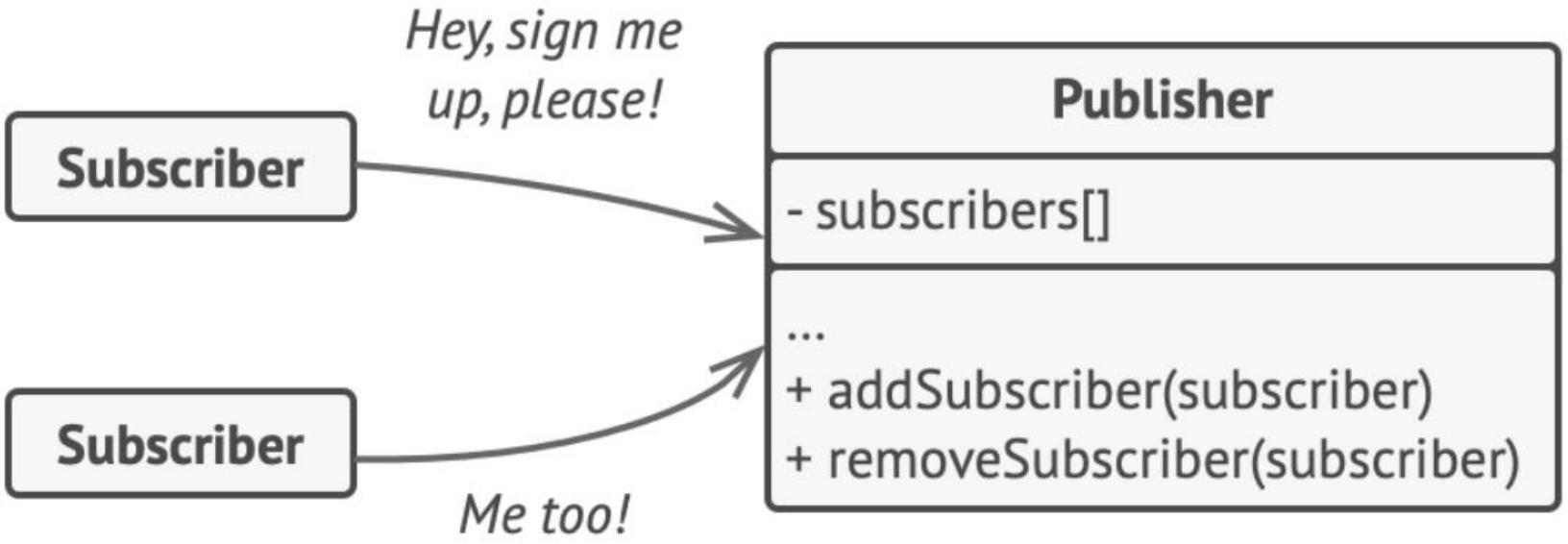


Problem



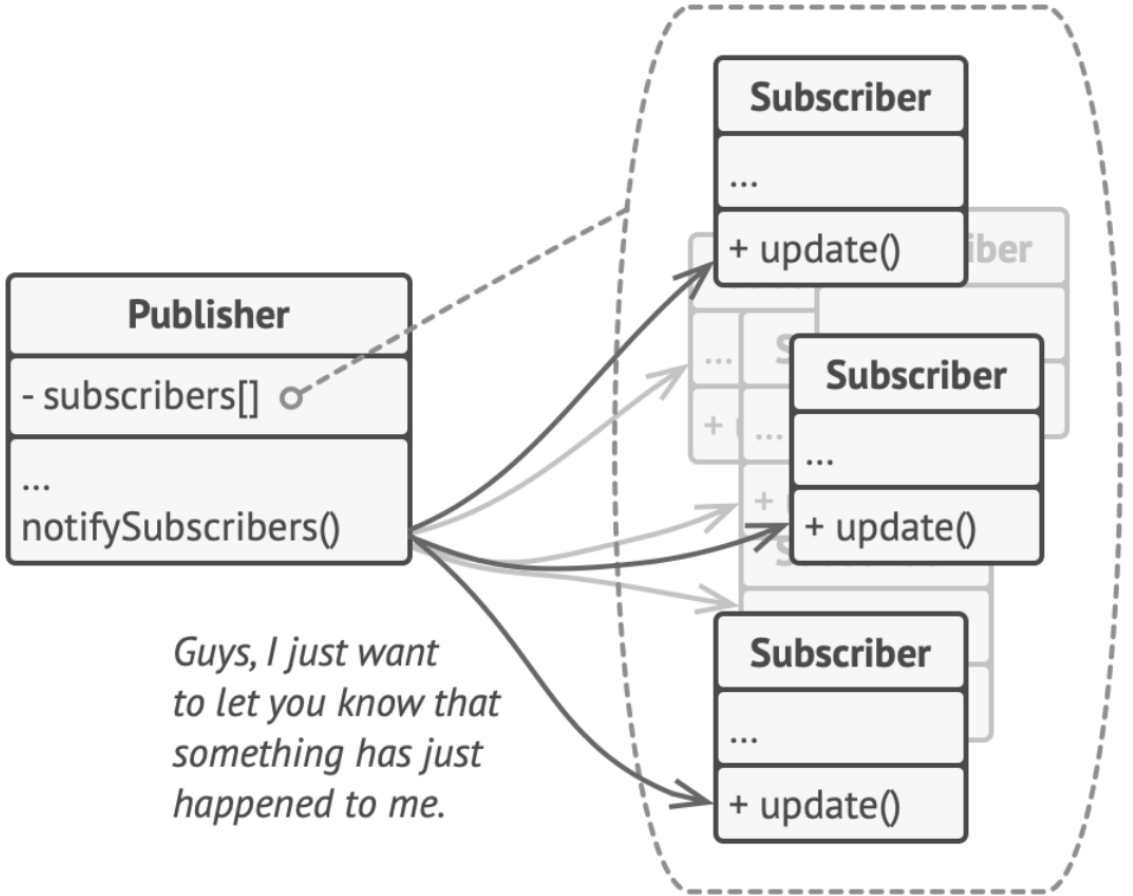
Visiting the store vs. sending spam

Solution



A subscription mechanism lets individual objects subscribe to event notifications.

Solution



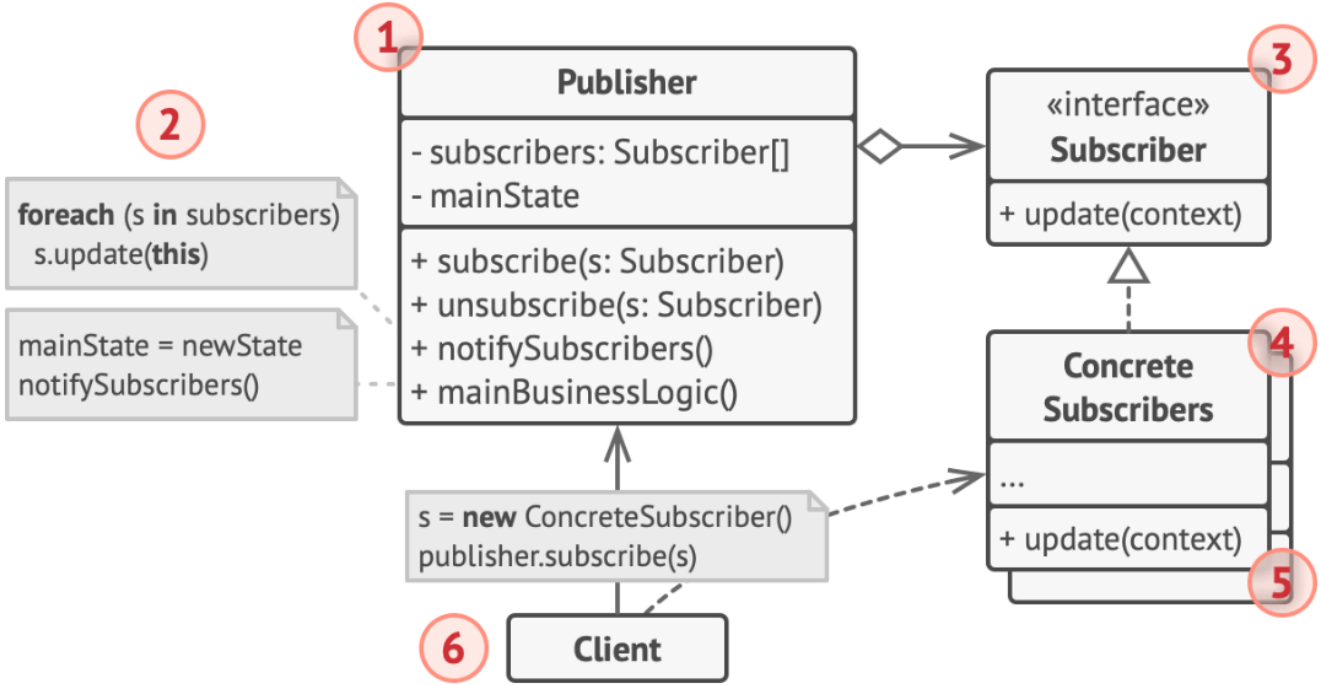
Publisher notifies subscribers by calling the specific notification method on their objects.

Examples

Magazine and newspaper subscriptions.

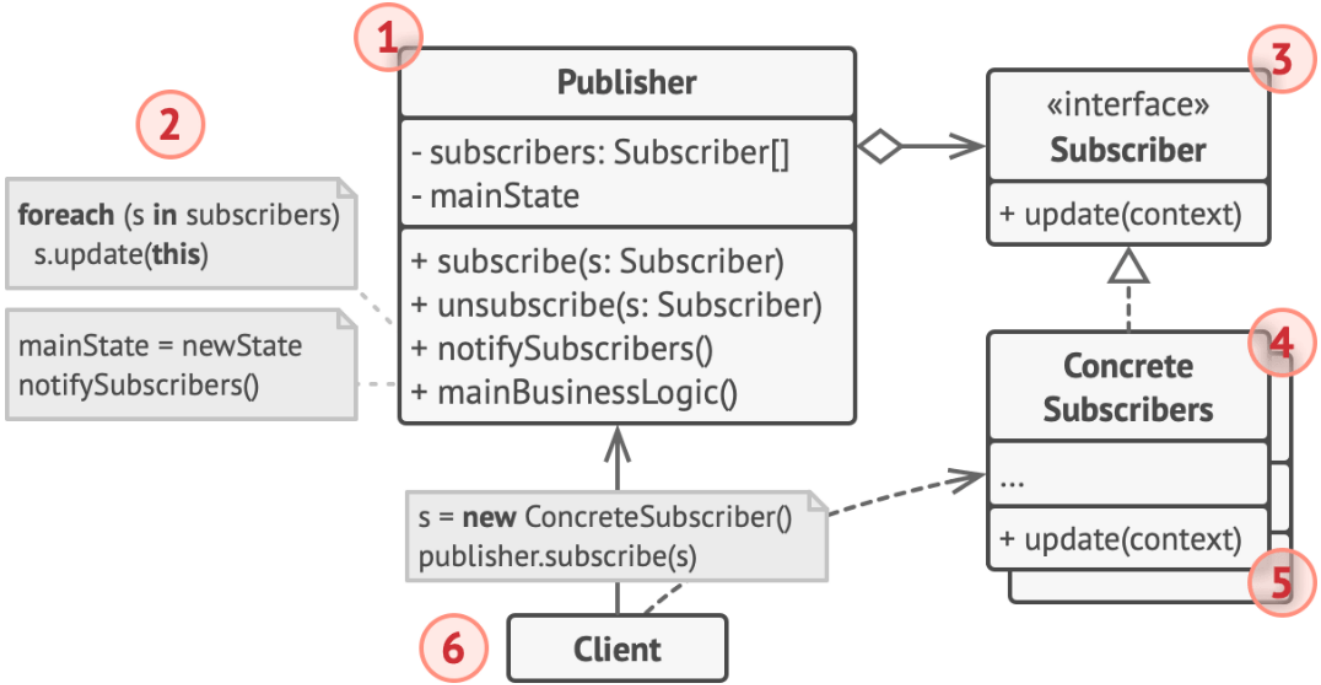
UML Class Diagram

1. The **Publisher** issues events of interest to other objects. These events occur when the publisher changes its state or executes some behaviors.



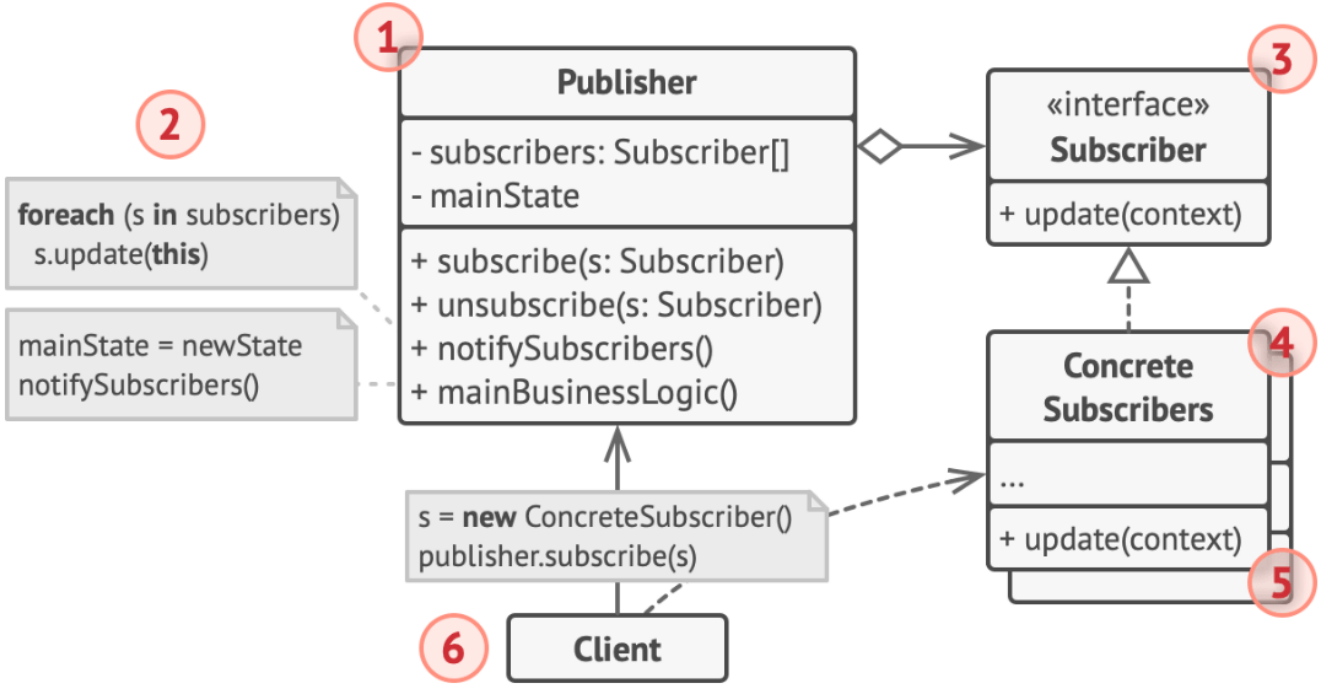
UML Class Diagram

2. When a new event happens, the publisher goes over the subscription list and calls the notification method declared in the subscriber interface on each subscriber object.



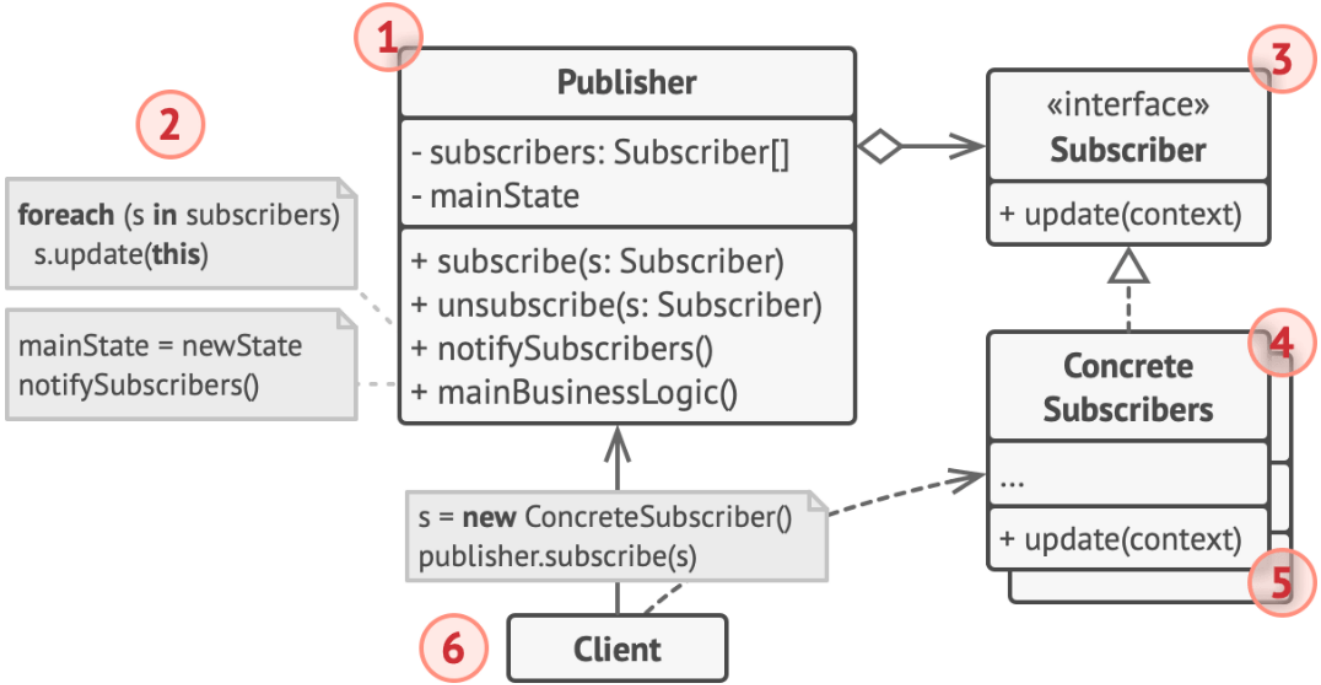
UML Class Diagram

3. In most cases, the **Subscriber** consists of a single update method. The method may have several parameters that let the publisher pass some event details along with the update.



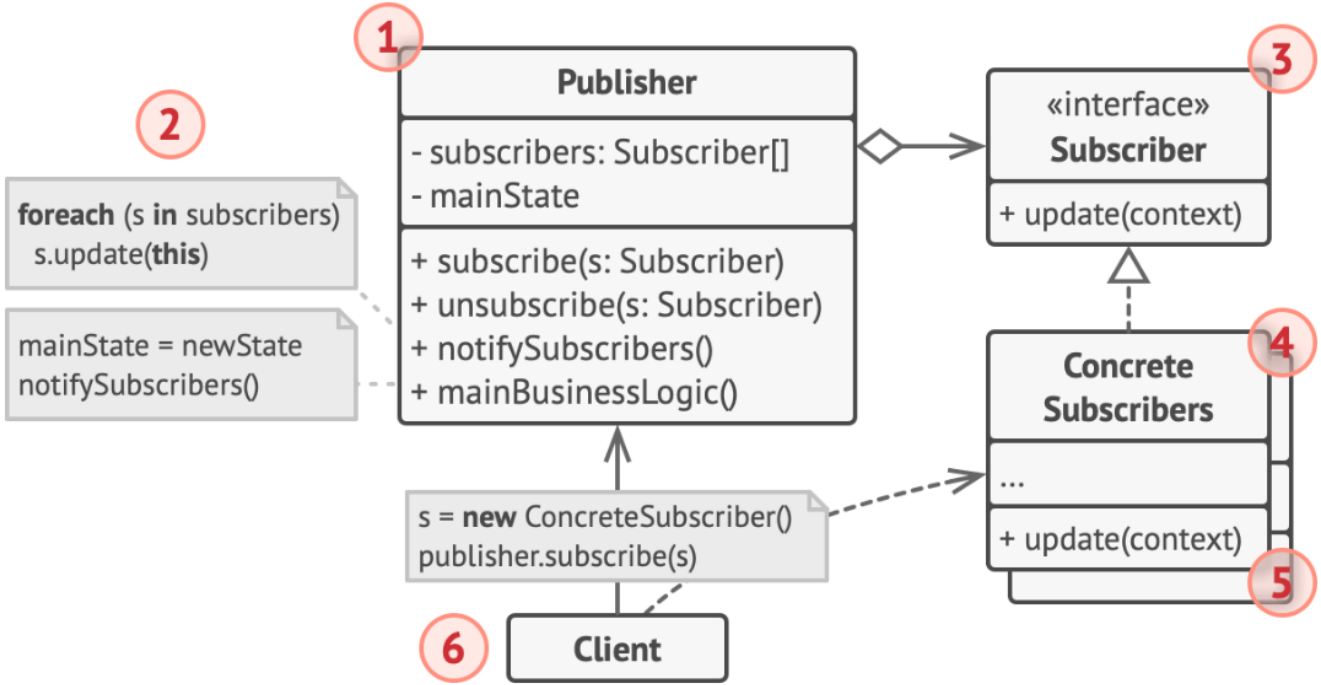
UML Class Diagram

4. **Concrete Subscribers** perform some actions in response to notifications issued by the publisher.



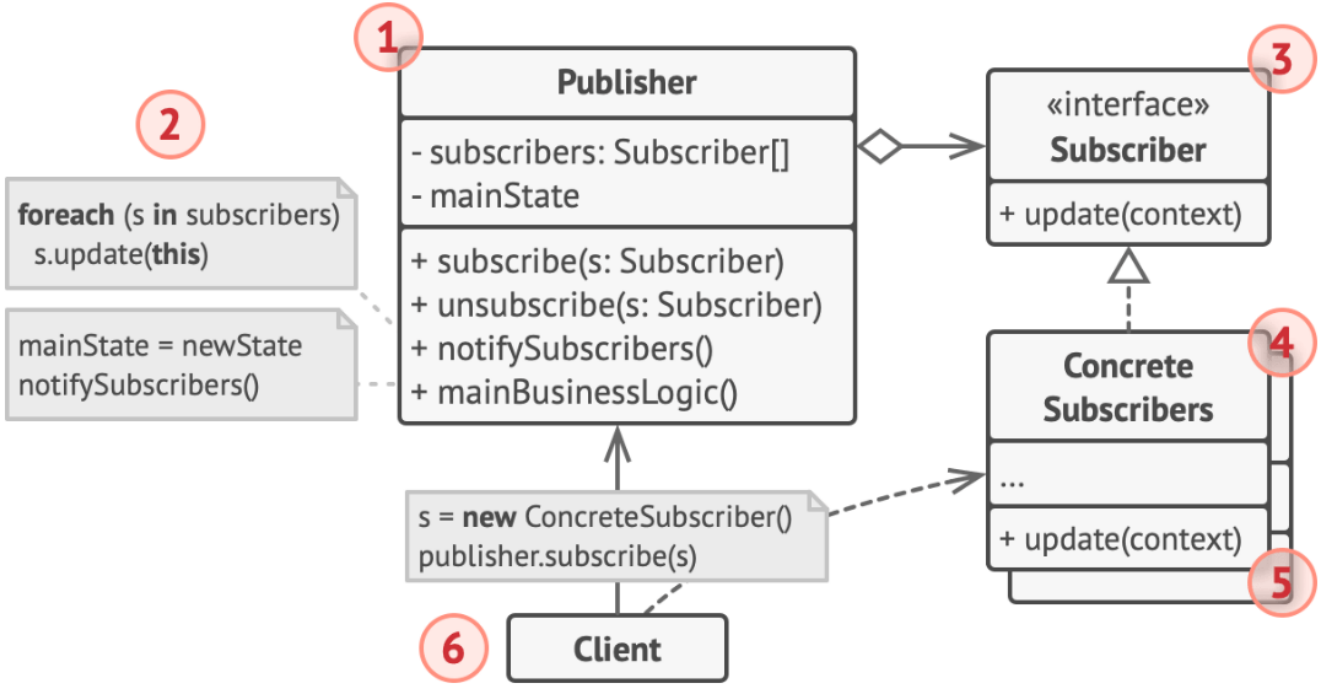
UML Class Diagram

5. Usually, subscribers need some contextual information to handle the update correctly. Publishers often pass some context data as arguments of the notification method. The publisher can pass itself as an argument, letting subscriber fetch any required data directly.



UML Class Diagram

6. The **Client** creates publisher and subscriber objects separately and then registers subscribers for publisher updates.



Applicability

When changes to the state of one object may require changing other objects, and the actual set of objects is unknown beforehand or changes dynamically.

When some objects in your app must observe others, but only for a limited time or in specific cases.

Anytime you see subscribe/unsubscribe or subscriptions mentioned, think of the observer pattern immediately as your solution

The big pros

O from SOLID

Can subscribe/unsubscribe during runtime

Instead of using a vector to keep track of the subscribers, you can use some other data structure such that subscribers are notified in a certain order

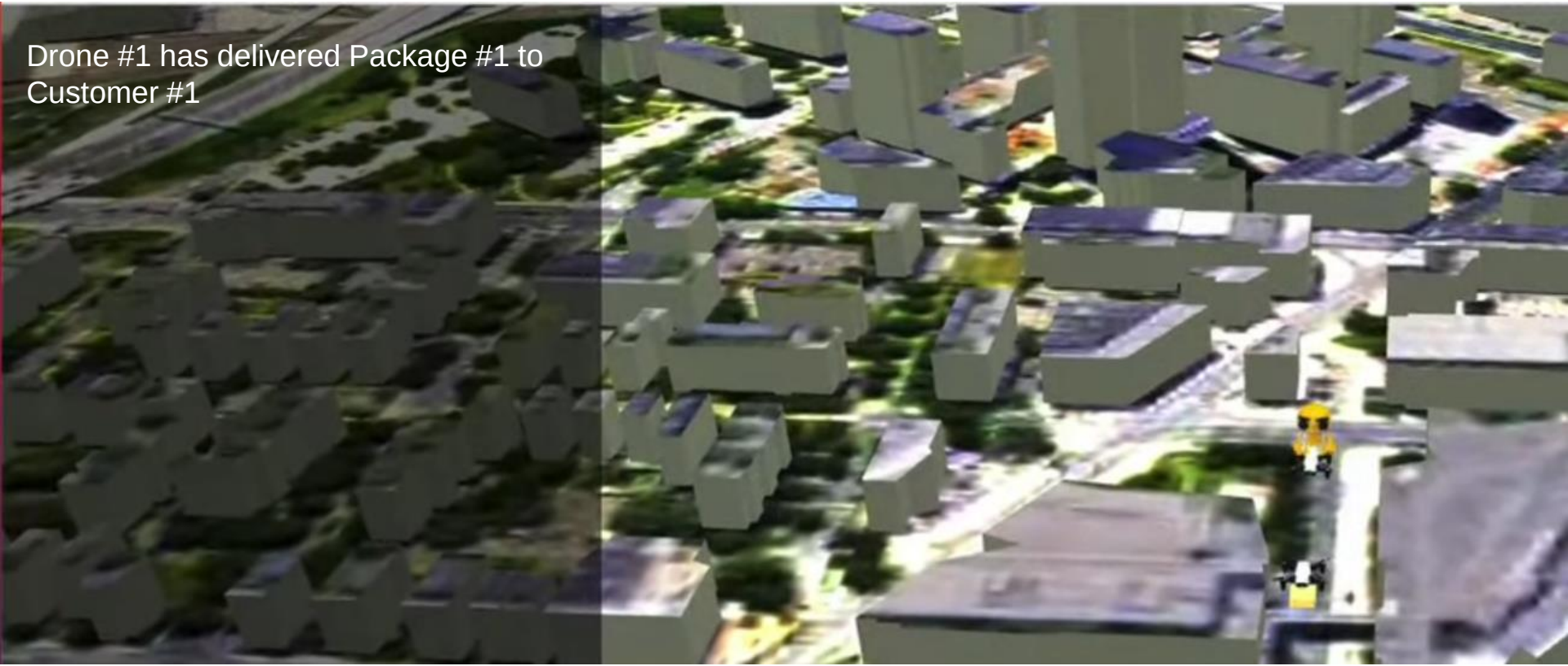
Most similar design patterns to observer

Chain of responsibility – keeps passing a request down a chain until one of the links receives it

Command – unidirectional connections between senders and receivers

Mediator – no connections between senders and receivers directly, all communication done indirectly via mediator object

Observer – subscribe/unsubscribe


Drone #1 has delivered Package #1 to
Customer #1


Drone #1 has delivered Package #1 to Customer #1
Drone #2 has delivered Package #2 to Customer #1

PS5 Code Example

New UML Diagram

Sequence Diagram

Shows process interactions in sequence (by time)