Design Amazon Notification System

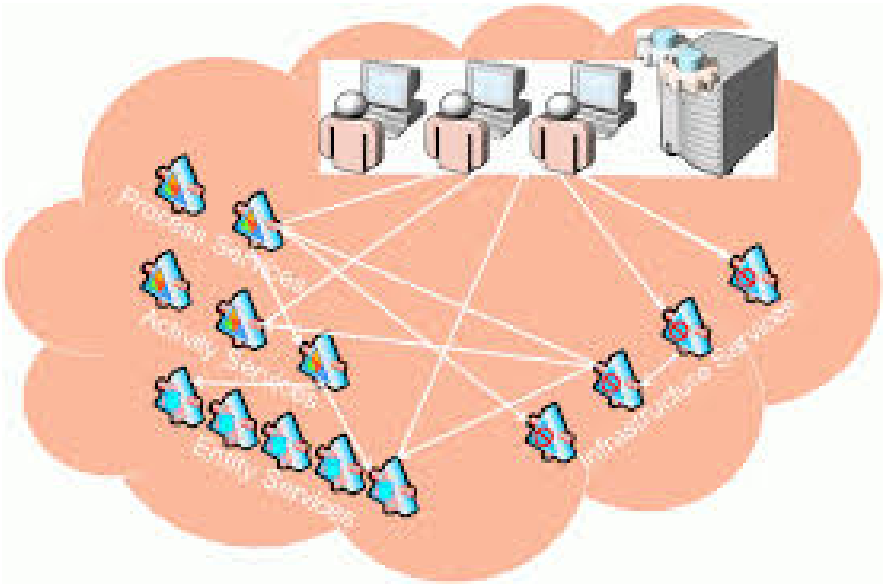
22 June 2024 13:33

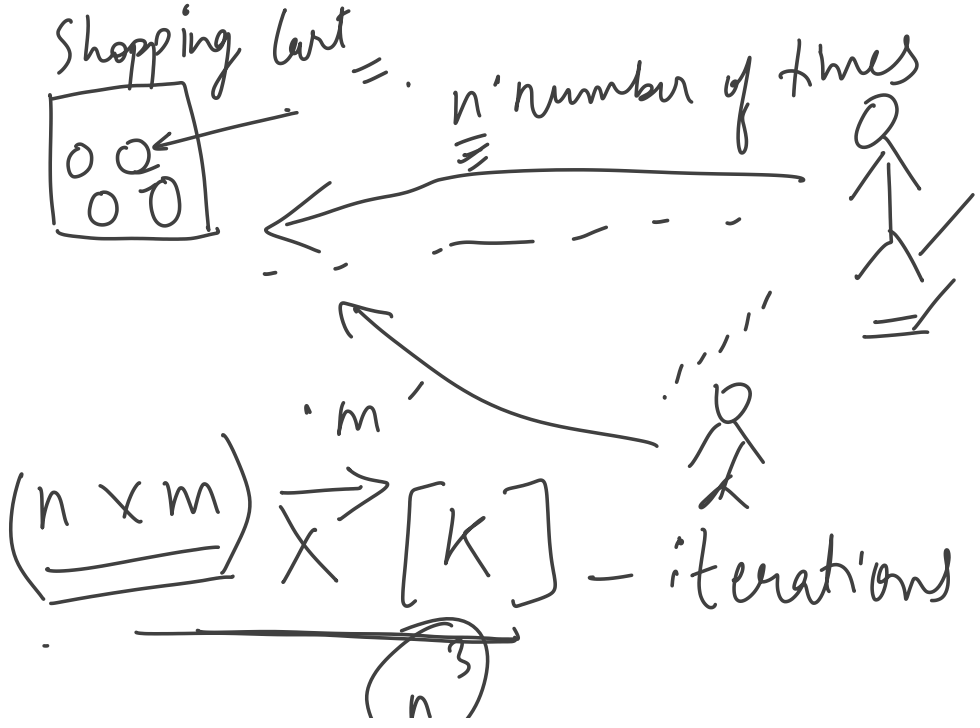
Design classes for a system that sends different types of messages - Email,SMS,Whatsapp.

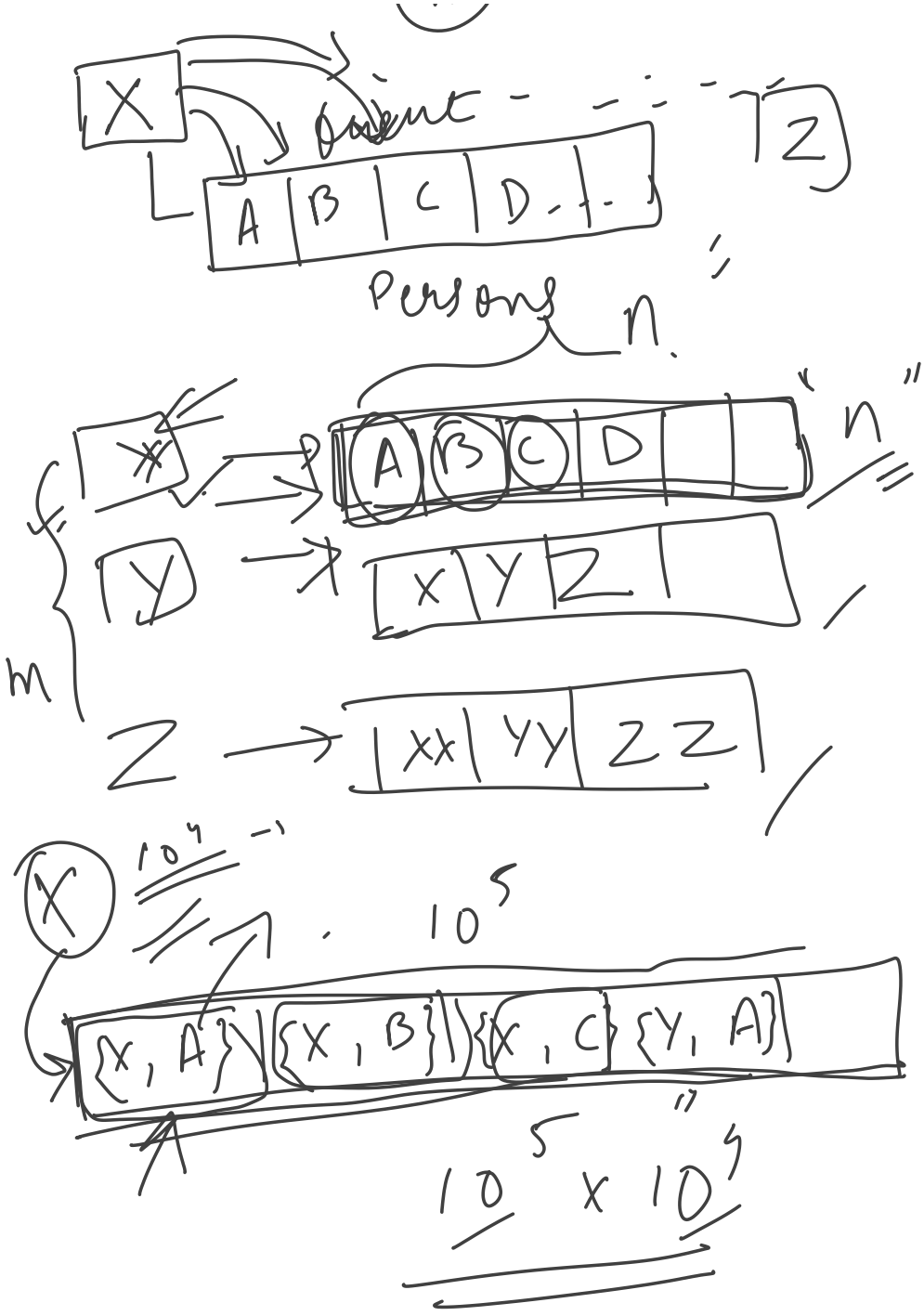
Functional Requirements:

1. **Notification Channels (Subscriber) :** The system should support multiple notification channels, such as email, SMS, push notifications, and in-app notifications. Each channel has its own delivery mechanism and requirements.

2. **Product Types (Publisher)** : Users can subscribe to different types of product, such as Iphone. Each product type may have specific content and type.



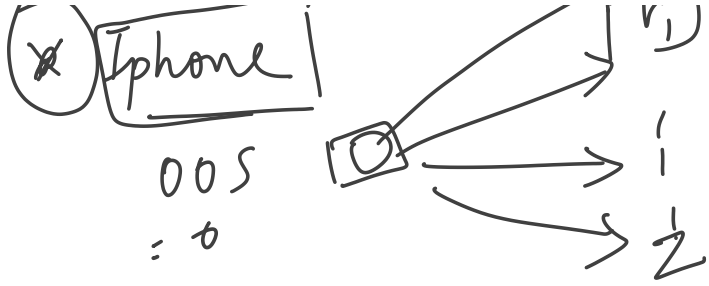




Pub - Sub

Publisher Subscriber

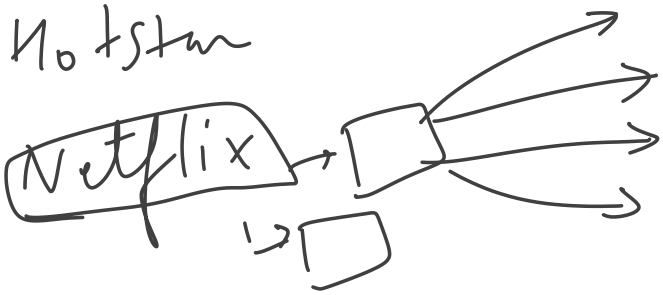
Amazon.



= 10

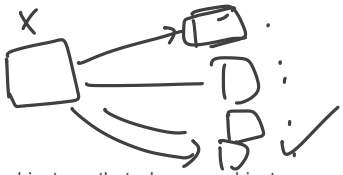
Gibu22

Motstm



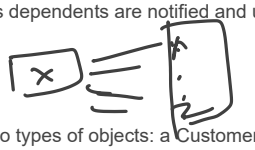
Observer Design Pattern

22 June 2024 13:34



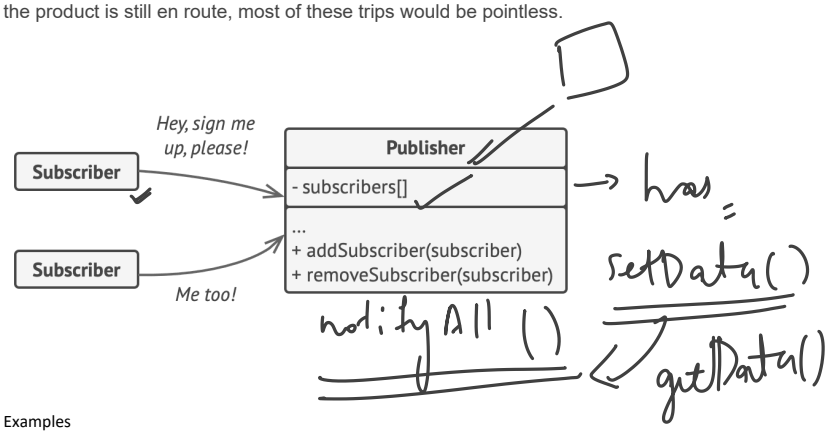
Intent

- Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



Imagine that you have two types of objects: a Customer and a Store. The customer is very interested in a particular brand of product (say, it's a new model of the iPhone) which should become available in the store very soon.

The customer could visit the store every day and check product availability. But while the product is still en route, most of these trips would be pointless.



Examples

- Pub-Sub Models
- Weather Station
- Cricket broadcast
- Notifications System
-

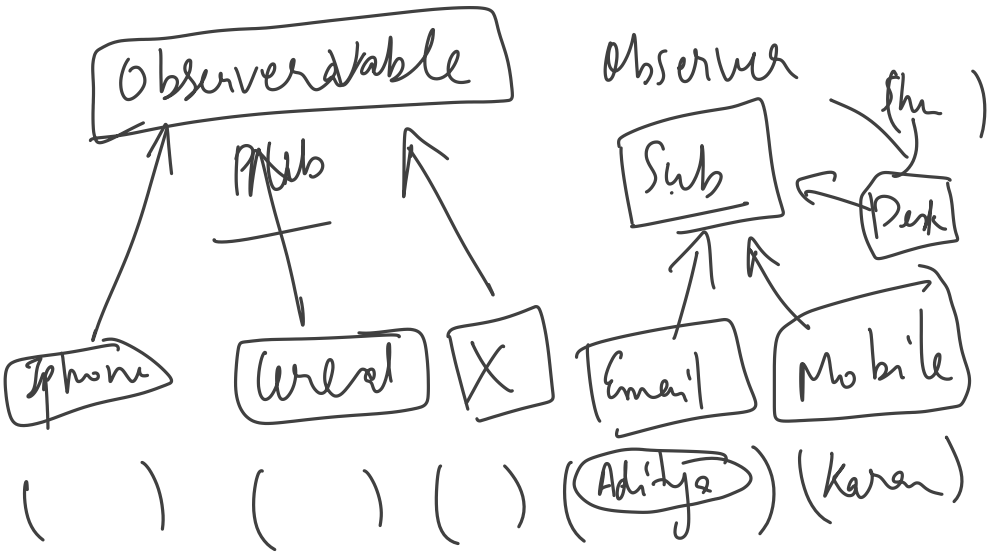
Pros :

Open/Closed Principle. You can introduce new subscriber classes without having to change the publisher's code (and vice versa if there's a publisher interface).

You can establish relations between objects at runtime.

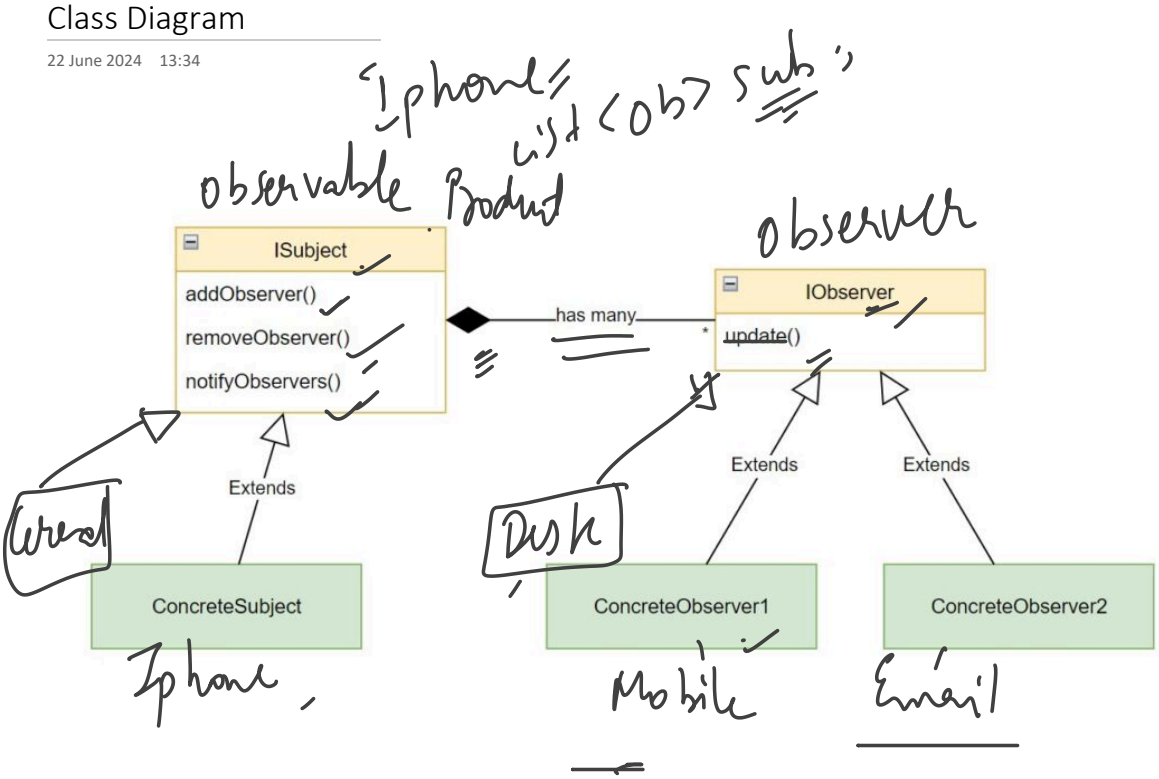
Cons:

Subscribers are notified in random order.



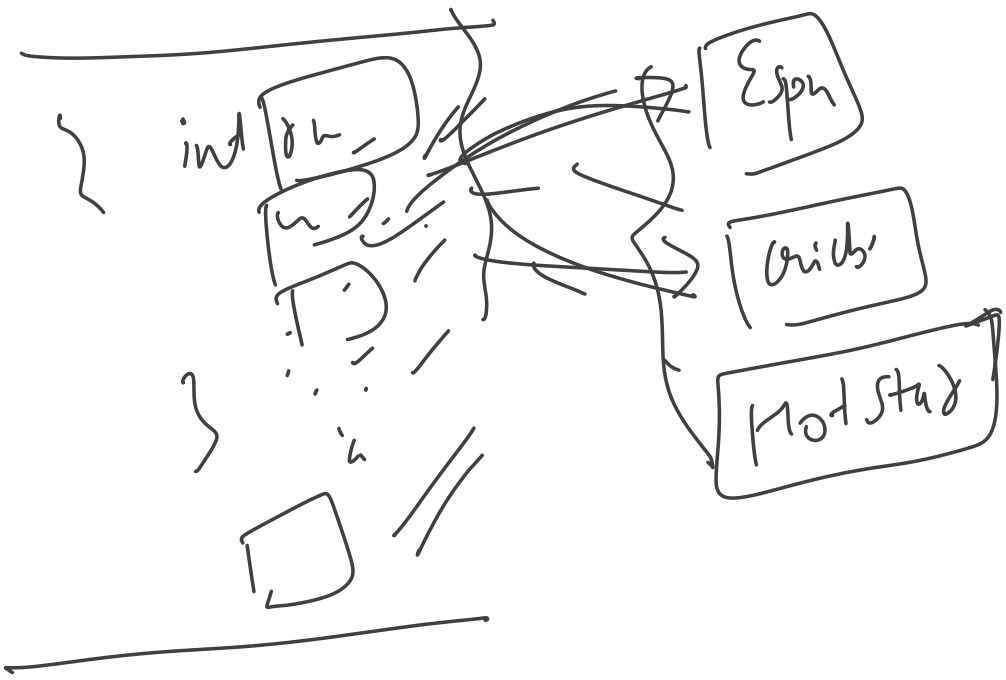
Class Diagram

22 June 2024 13:34

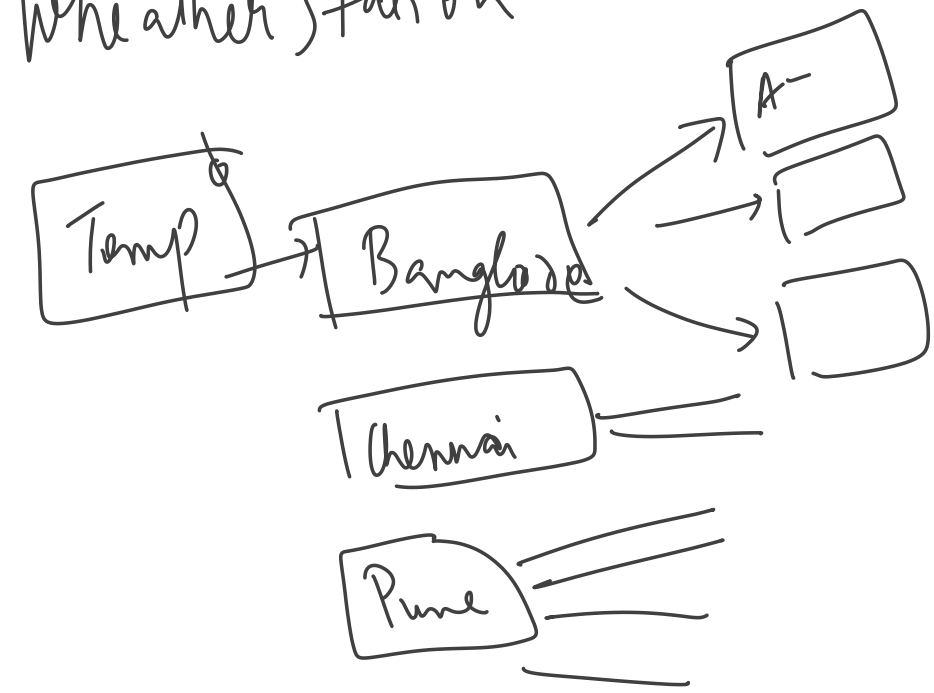



```
class Iphone {
    add();
    ;
}

class Ireal {
    addObserver()
}
```



Weather Station



Design Kafka → Pub Sub