Behavioural Design Patterns.

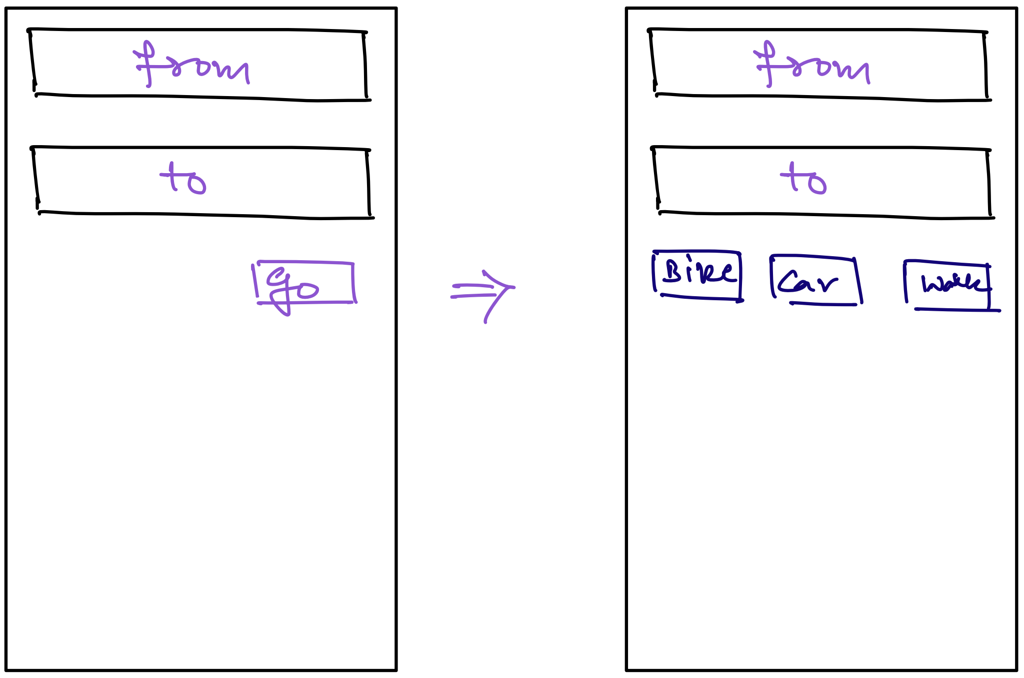
↳ Actions | Methods.

⇒ There are some special kind of behaviours that needs to be implemented for an entity.

① Strategy

② Observer.

STRATEGY.



⇒ When we search path from point Ⓐ to Ⓑ on Google Maps, it suggests us different path based on mode of transportation we are using.

⇒ GoogleMap {

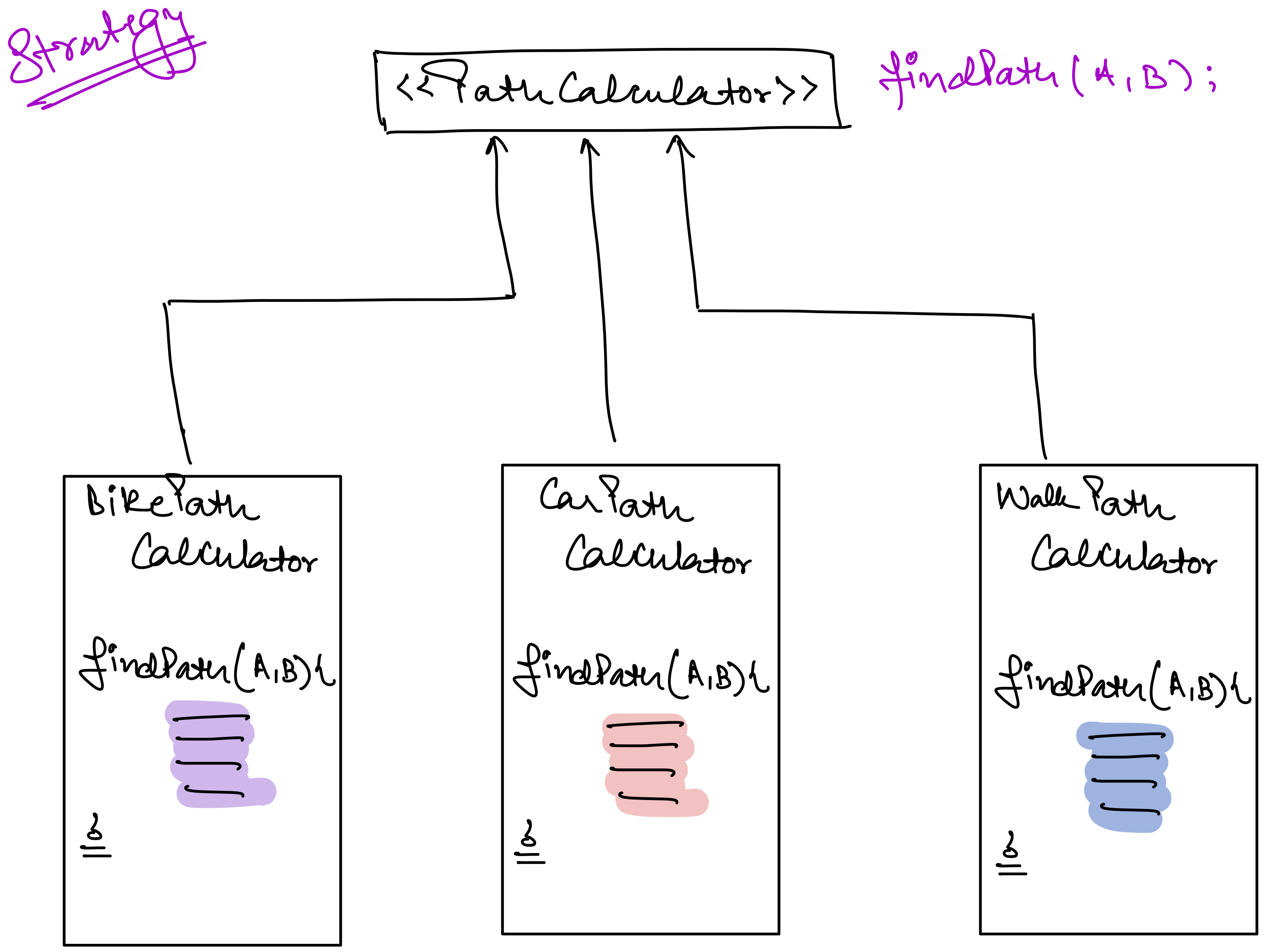
```
    findPath (from, to, mode) {
        if (mode == Car) {
                          CPC = new CPC();
                          cpc.findPath(—);
        }
        else if (mode == Bike) {
                          BPC = new BPC();
                          bpc.findPath();
        }
        else if (mode == ——) {

        }
}
```

SRP ×
OCP ×

⇒ When there are multiple ways of doing something.

$$\begin{tikzcd}
 & & \text{Sort} \arrow[dll] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{Quick} & & \text{Merge} & \text{Heap} & \text{Bubble.}
\end{tikzcd}$$

⇒ Often we see violation of SRP | OCP if there multiple ways of implementing a method because of multiple if-else conditions.



```
GoogleMap
--------------------------------
PathCalculator PC;
findPath(A, B, mode){
    if(mode == Car){
        PC = new CPC();
    }
    else if (mode == "Bike"){
        PC = new BPC();
    }
    ____
    ____
    ____
    ____
}
```

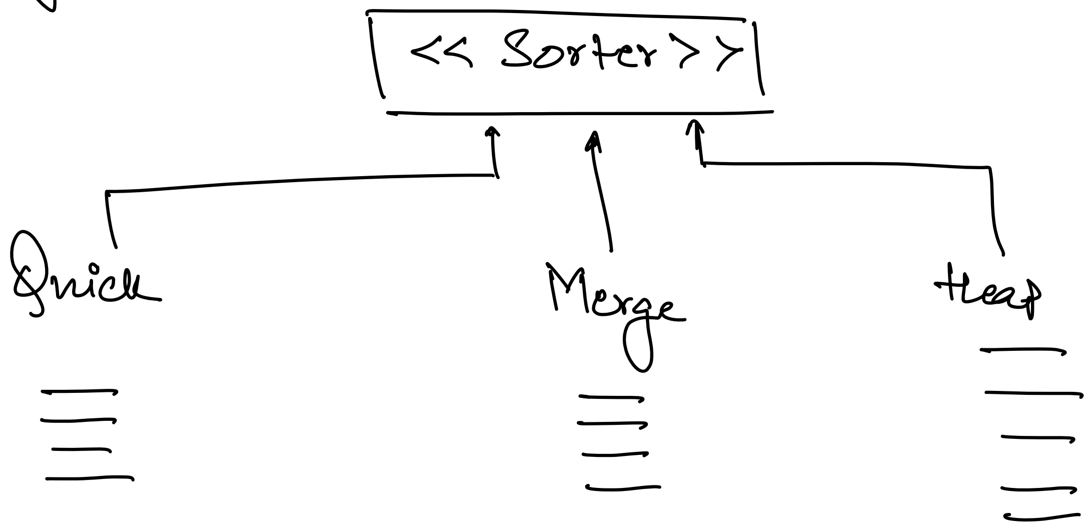
OCP ×

SRP ×

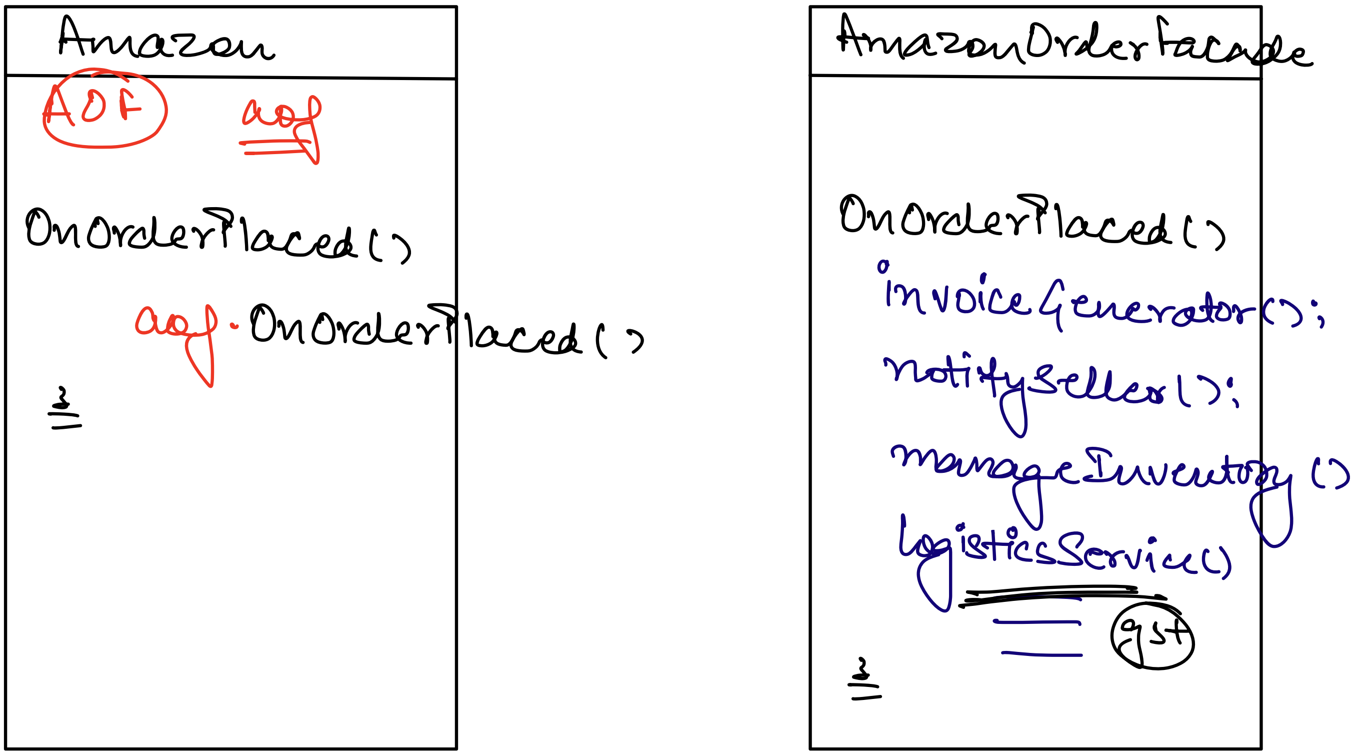
```
PathCalculatorFactory
--------------------------------
static
getPathCalculatorForMode(String mode){
    if(mode == Car){
        PC = new CPC();
    }
    else if (mode == "Bike"){
        PC = new BPC();
    }
    ____
    ____
    ____
    ____
}
```

```
GoogleMap
---------------------------
PathCalculator PC;

findPath (A, B, mode) {
    PC = PCF.getPCforMode(mode);
    PC.findPath(A, B);
}
```

Sorting



⇒ Observer.



```
Amazon
  AOF      aof

OnOrderPlaced()
    aof.OnOrderPlaced()
}
```

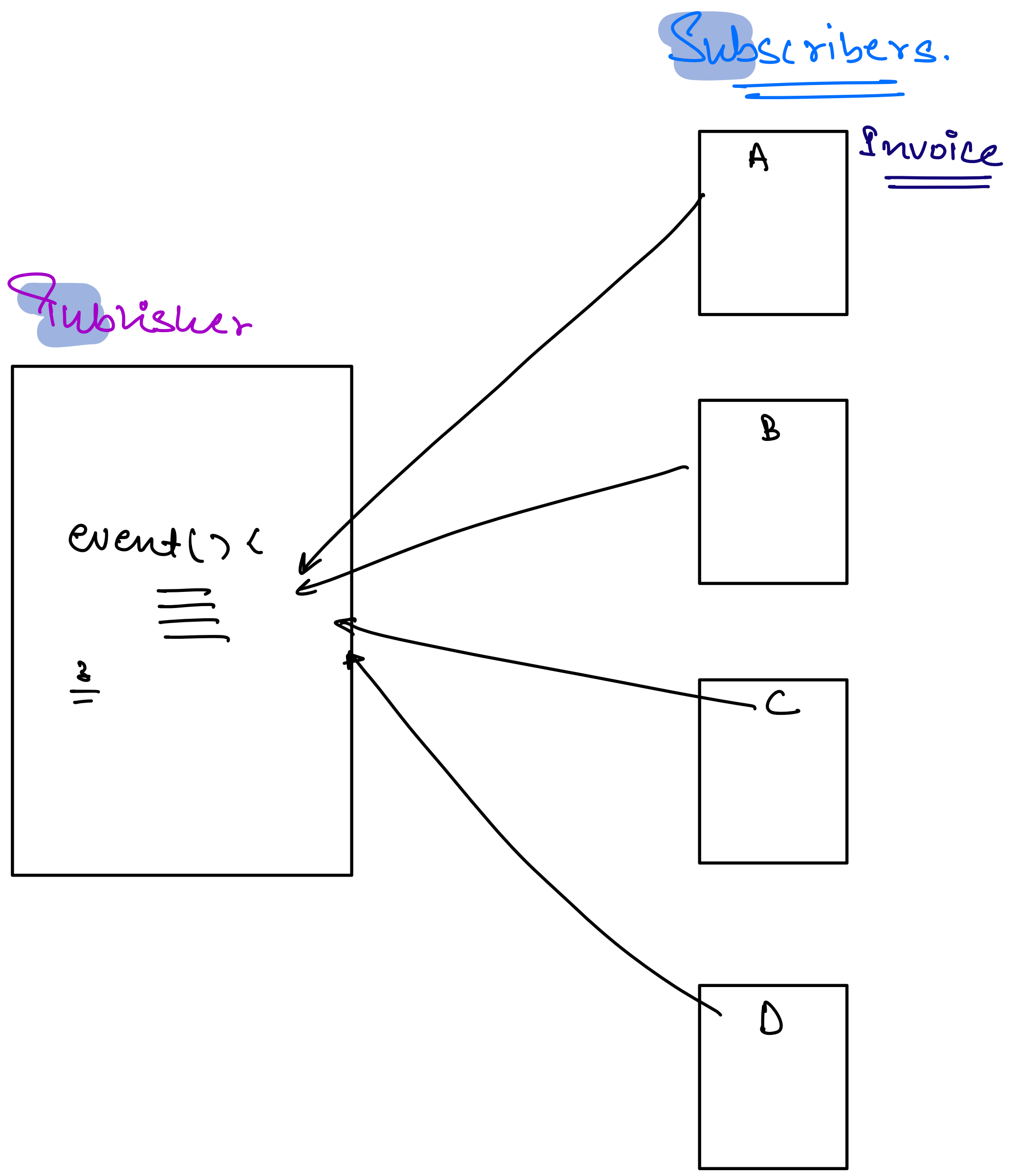
```
AmazonOrderFacade

OnOrderPlaced()
  invoiceGenerator();
  notifySeller();
  manageInventory()
  logisticsService()
                  gst
}
```

Problem Statement

⇒ When some event happens, we might want to do a lot of things internally. Every time we want to add/remove something from the list of actions, we'll have to do the code changes.

⇒ We can't add/remove actions at runtime.

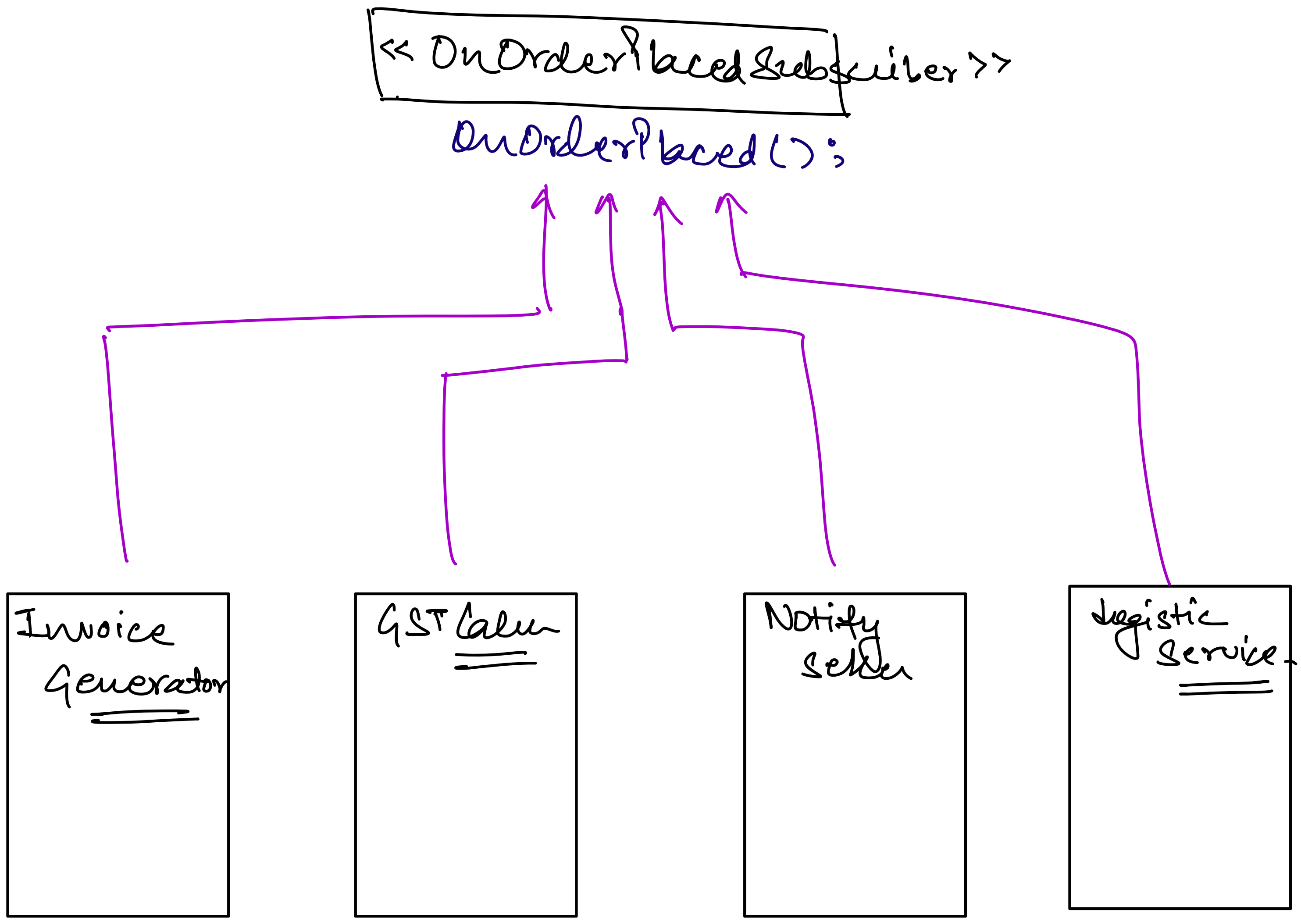
OBSERVER.



⇒ "Event Driven Architecture".

⇒ When an event gets published, multiple Subscribers would like to get executed.

1. Create different classes for each subscriber.
2. Make all the subscribers implement a common interface.
3. Publisher should provide a functionality to add | remove a subscriber for the event.



```
AmzOnOrderPlaced
List<OnOrderPlacedSubsciber> subscribers;

Static
addSubsciber( ↓OOPS — ){
    subscibers.add(↓)
}

OnOrderPlaced(){
    for(Subs sub : subscibers){
        sub.OnOrderPlaced();
    }
}
```

=>

```
InvoiceGenerator.

InvoiceGenerator(){
    AmazonOnOrderPlaced.subscribe(this);
}
```

—— * ——

Implementation.