Observer

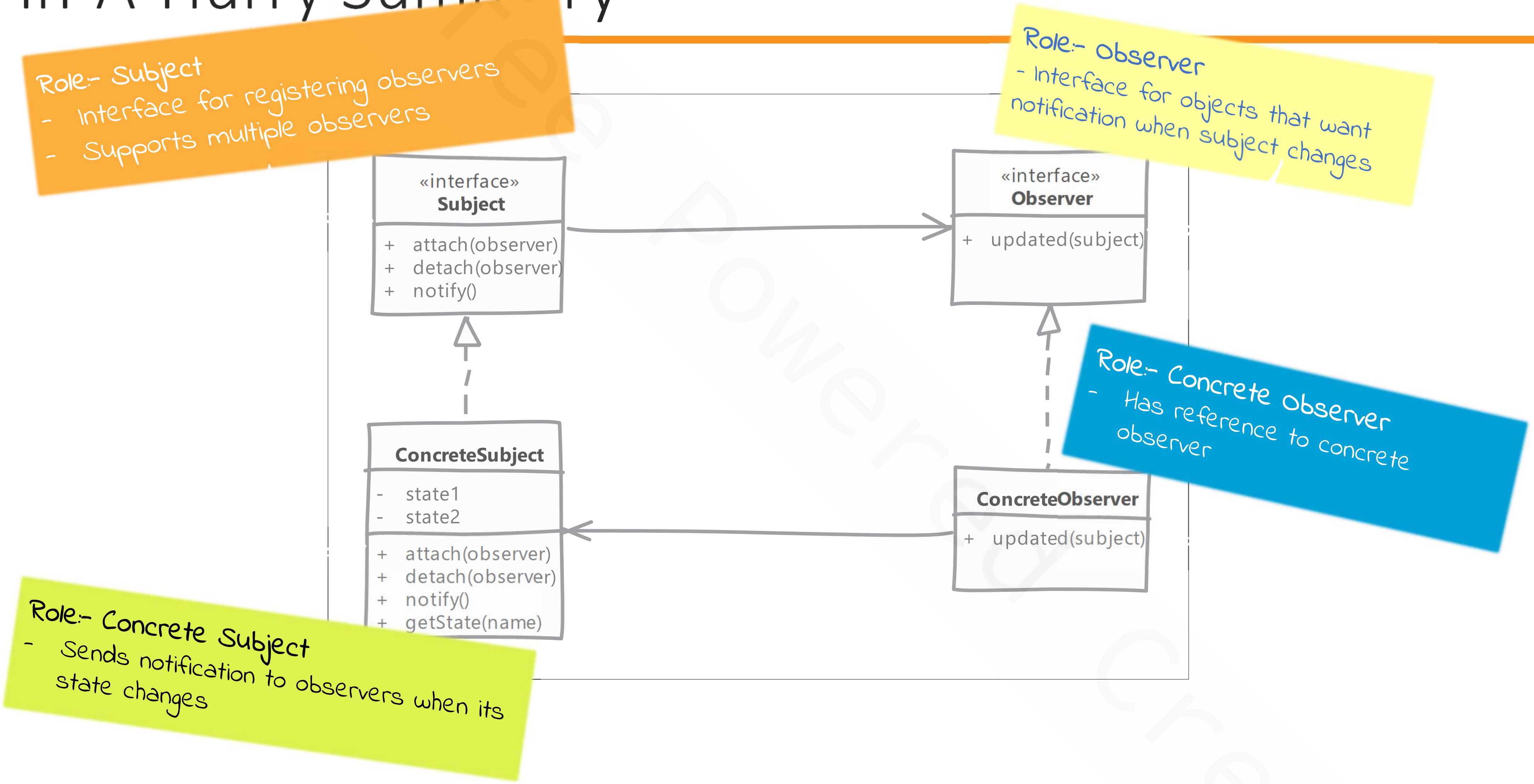
Behavioral Design Patterns

Design Patterns in Java

In-A-Hurry Summary

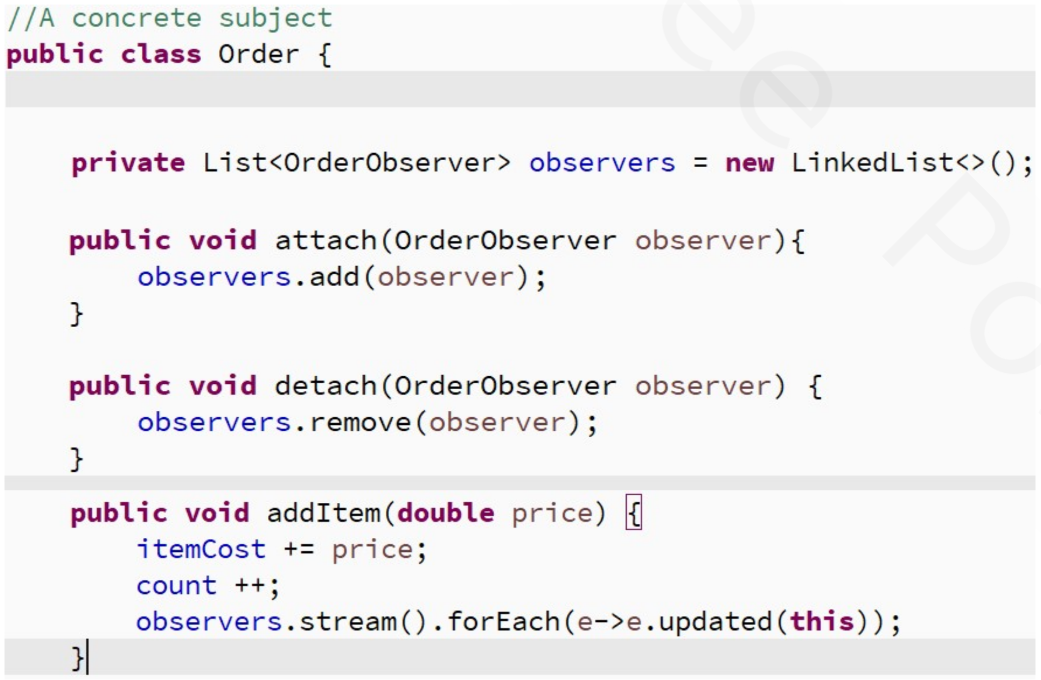
- Observer pattern allows to define one-to-many dependency between objects where many objects are interested in state change of a object.
- Observers register themselves with the subject which then notifies all registered observers if any state change occurs.
- In the notification sent to observers it is common to only send reference of subject instead of state values. Observers will call the subject back for more information if needed.
- We can also register observers for a specific event only, resulting in improved performance of sending notifications in the subject.
- This design pattern is also known as publisher-subscriber pattern. Java messaging uses this pattern but instead of registering with subject, listeners register with a JMS broker, which acts as a middleman.

In-A-Hurry Summary



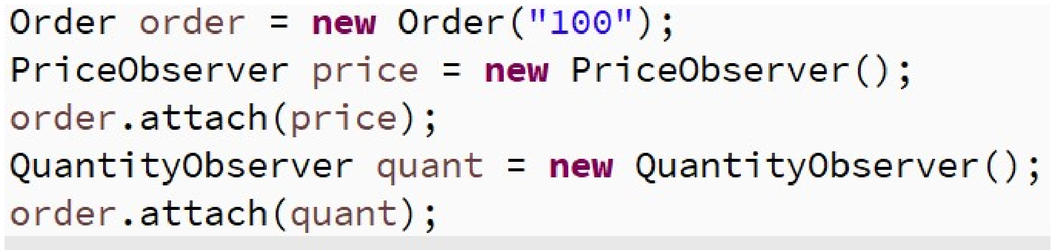
In-A-Hurry Summary

Subject



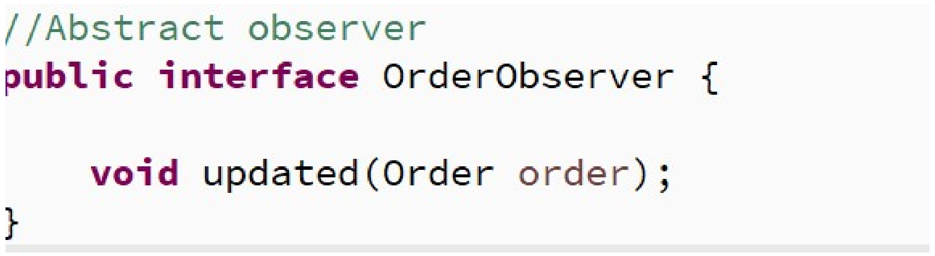
```
//A concrete subject
public class Order {


    private List<OrderObserver> observers = new LinkedList<>();

    public void attach(OrderObserver observer){
        observers.add(observer);
    }

    public void detach(OrderObserver observer) {
        observers.remove(observer);
    }
    public void addItem(double price) {
        itemCost += price;
        count ++;
        observers.stream().forEach(e->e.updated(this));
    }
```

Attaching Observers to Subject



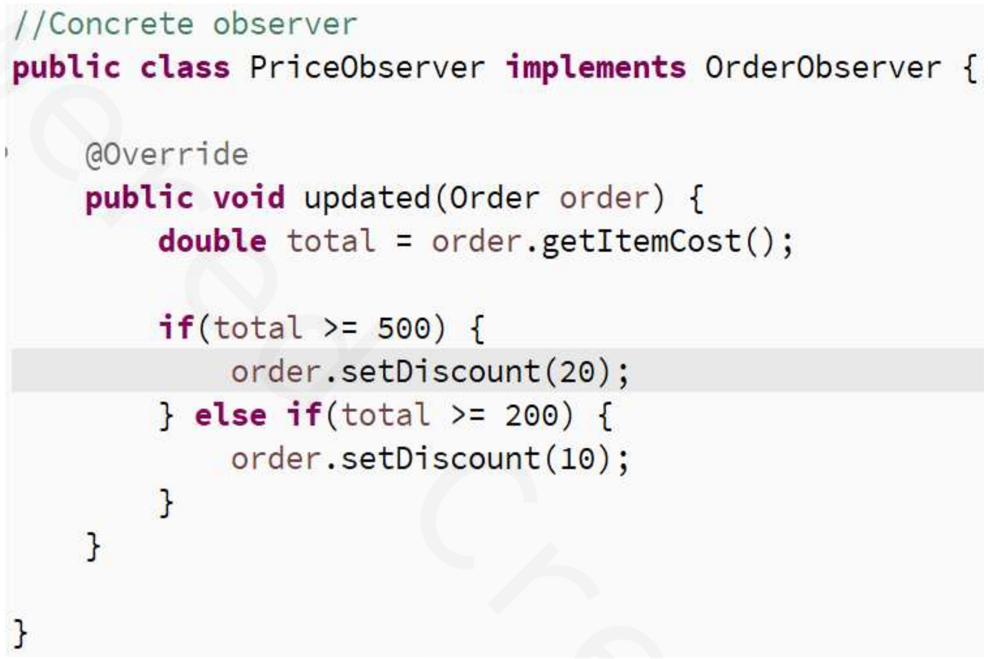
```
Order order = new Order("100");
PriceObserver price = new PriceObserver();
order.attach(price);
QuantityObserver quant = new QuantityObserver();
order.attach(quant);
```

Observer



```
//Abstract observer
public interface OrderObserver {

    void updated(Order order);
}
```

Concrete Observer



```
//Concrete observer
public class PriceObserver implements OrderObserver {

    @Override
    public void updated(Order order) {
        double total = order.getItemCost();

        if(total >= 500) {
            order.setDiscount(20);
        } else if(total >= 200) {
            order.setDiscount(10);
        }
    }

}
```