Принципы проектирования и дизайна ПО

Лекция №10

Агошков Илья 2016

Observer

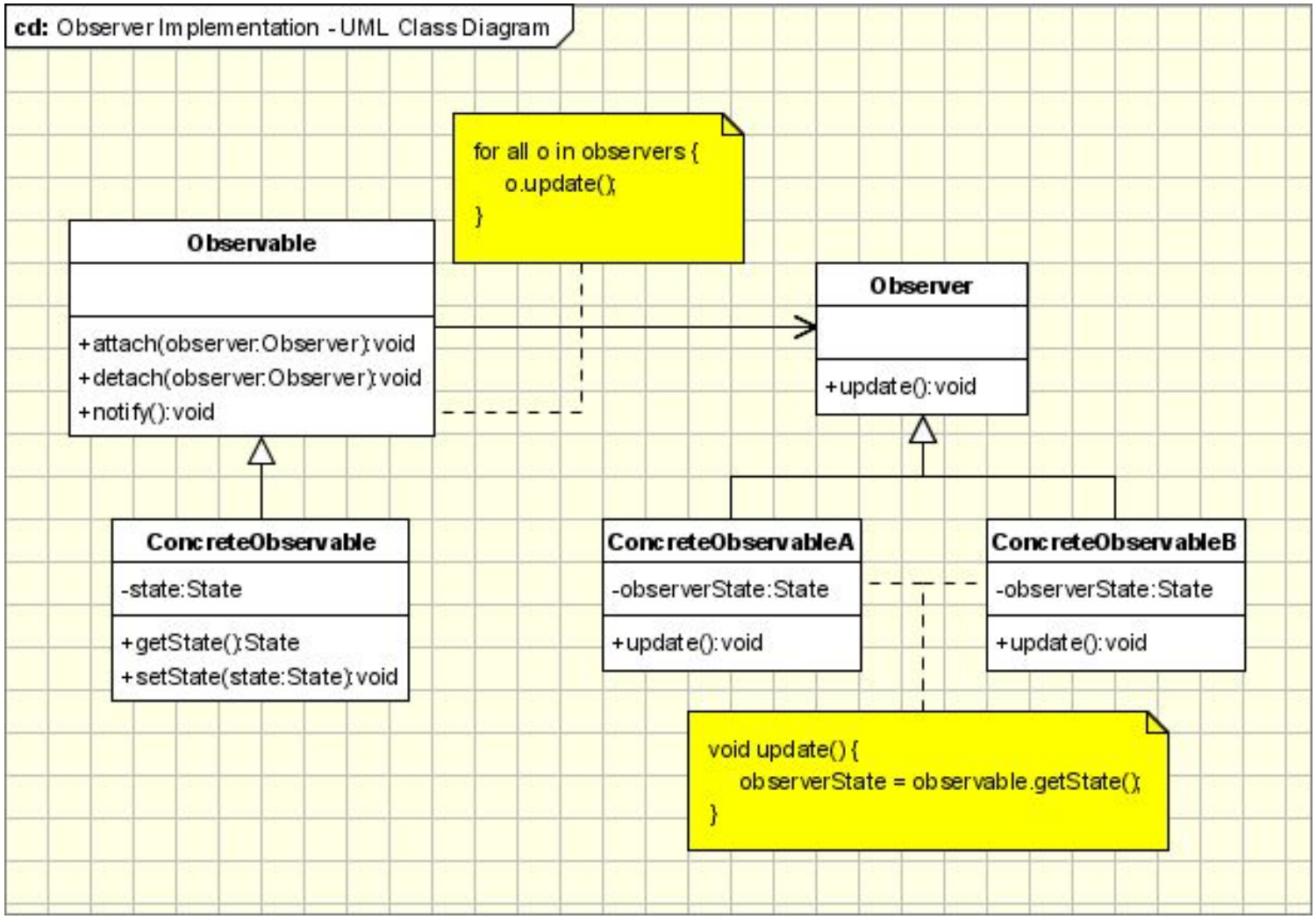
Intent

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

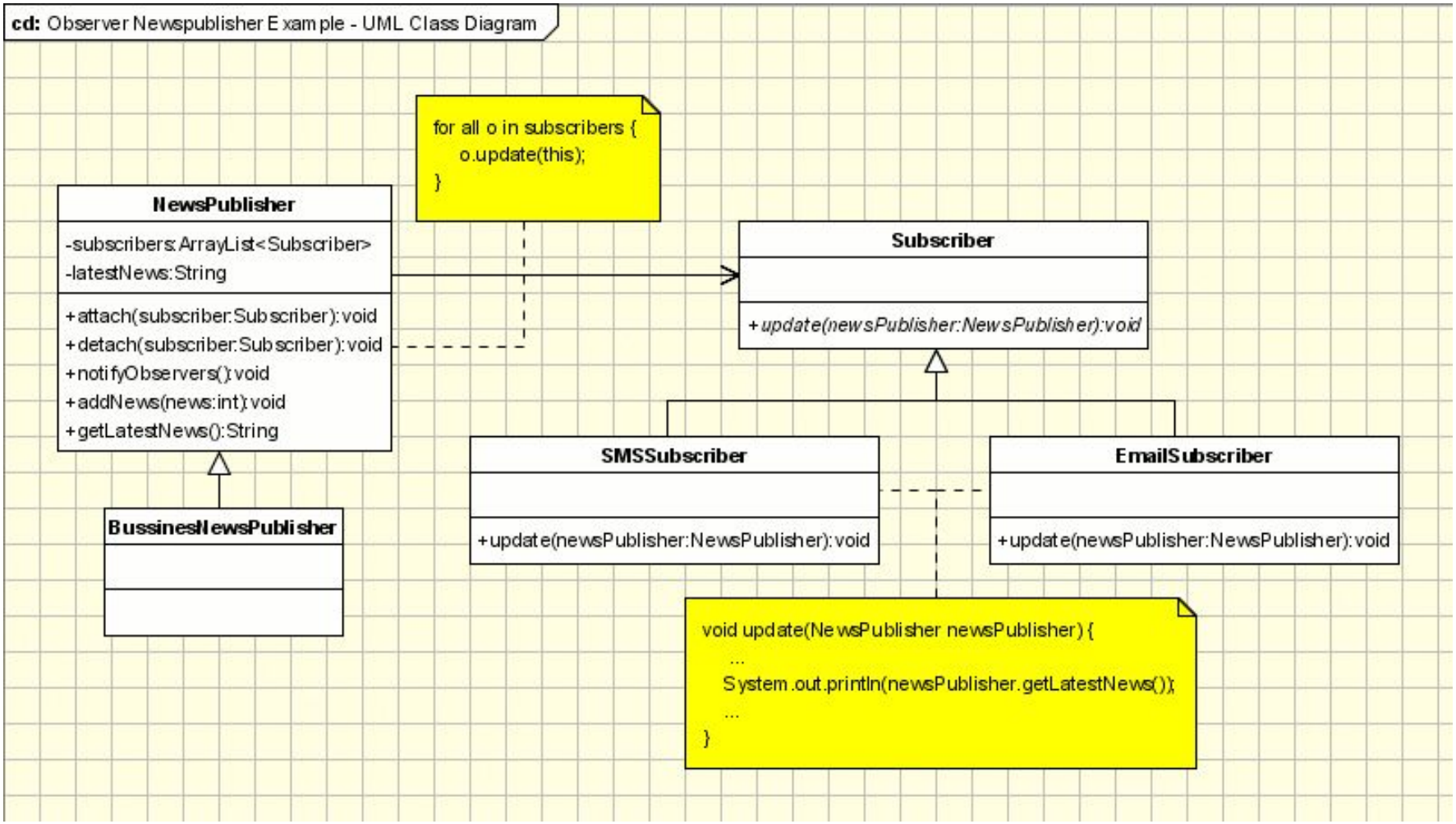
Also known as

Listener, Publish-Subscribe

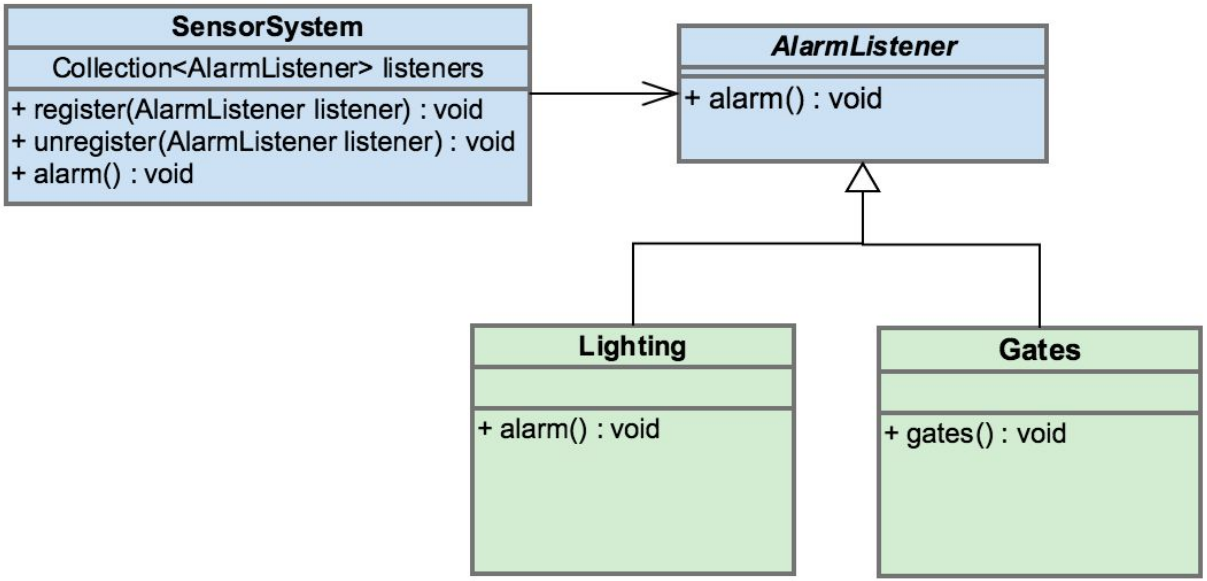
Observer


cd: Observer Implementation - UML Class Diagram
for all o in observers {
o.update();
}
Observable
+attach(observer:Observer):void
+detach(observer:Observer):void
+notify():void
Observer
+update():void
ConcreteObservable
-state:State
+getState():State
+setState(state:State):void
ConcreteObservableA
-observerState:State
+update():void
ConcreteObservableB
-observerState:State
+update():void
void update() {
observerState = observable.getState();
}

Observer


cd: Observer Newspublisher Example - UML Class Diagram
for all o in subscribers {
o.update(this);
}
NewsPublisher
-subscribers:ArrayList<Subscriber>
-latestNews:String
+attach(subscriber:Subscriber):void
+detach(subscriber:Subscriber):void
+notifyObservers():void
+addNews(news:int):void
+getLatestNews():String
Subscriber
+update(newsPublisher:NewsPublisher):void
BussinesNewsPublisher
SMSSubscriber
+update(newsPublisher:NewsPublisher):void
EmailSubscriber
+update(newsPublisher:NewsPublisher):void
void update(NewsPublisher newsPublisher) {
...
System.out.println(newsPublisher.getLatestNews());
...
}

Observer


SensorSystem
Collection<AlarmListener> listeners
+ register(AlarmListener listener) : void
+ unregister(AlarmListener listener) : void
+ alarm() : void
AlarmListener
+ alarm() : void
Lighting
+ alarm() : void
Gates
+ gates() : void

Observer

1. Mapping subjects to their observers.
2. Observing more than one subject.
3. Who triggers the update?

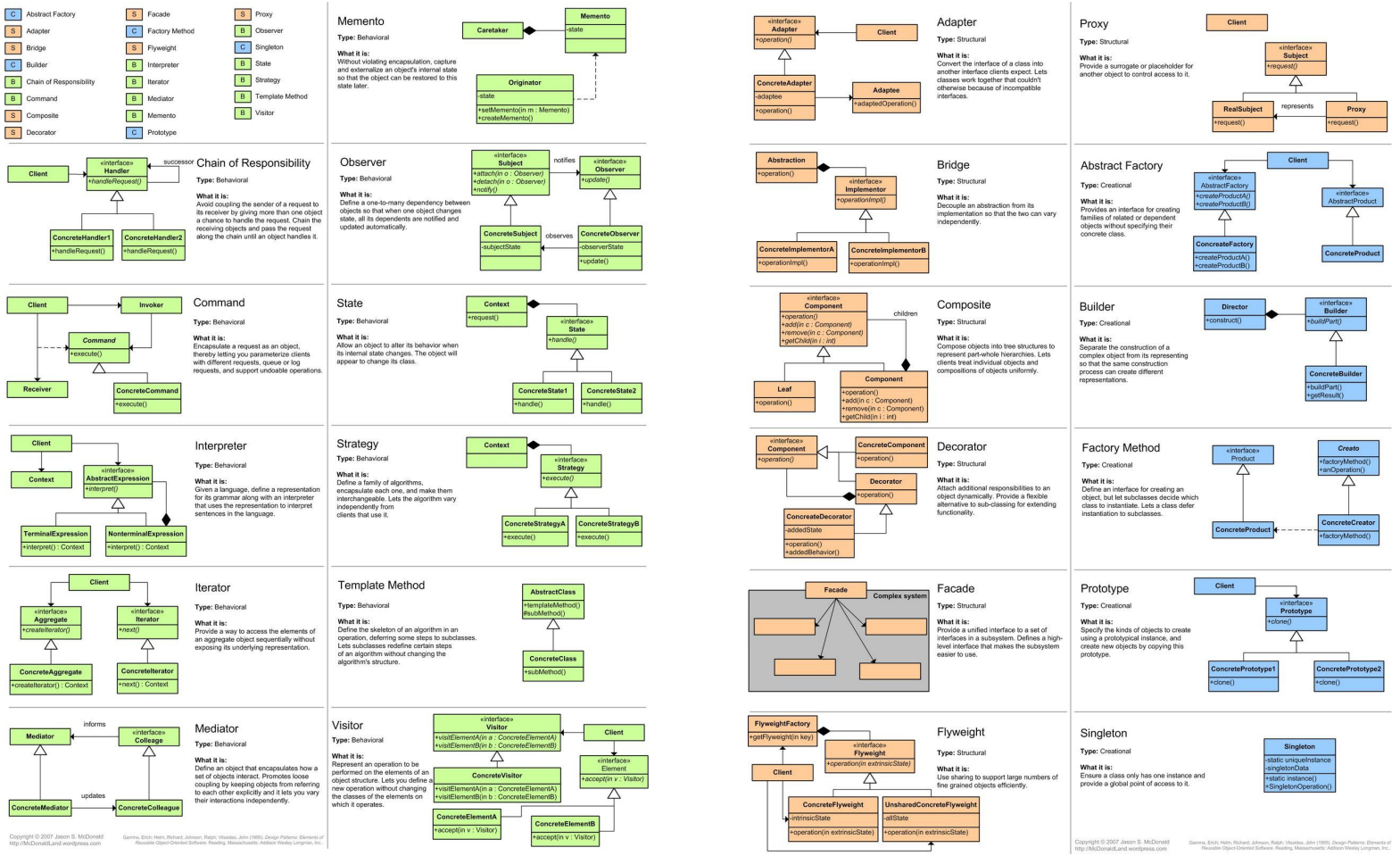
(a) Have state-setting operations on Subject call Notify after they change the subject's state.

(b) Make clients responsible for calling Notify at the right time.

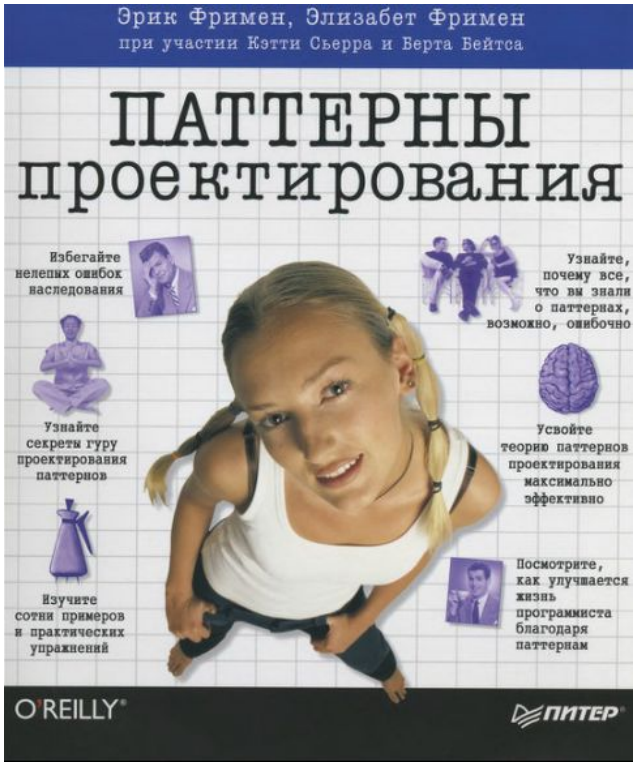
4. Dangling references to deleted subjects. Deleting a subject should not produce dangling references in its observers.
5. Making sure Subject state is self-consistent before notification.
6. Update protocols: the push and pull models.
7. Specifying modifications of interest explicitly.
8. Combining the Subject and Observer classes.

Design patterns cheat sheet

http://viralpatel.net/blogs/download/design-pattern-scard.pdf



Additional materials



Useful links

http://www.oodesign.com/

http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns

http://citforum.ru/SE/project/pattern/