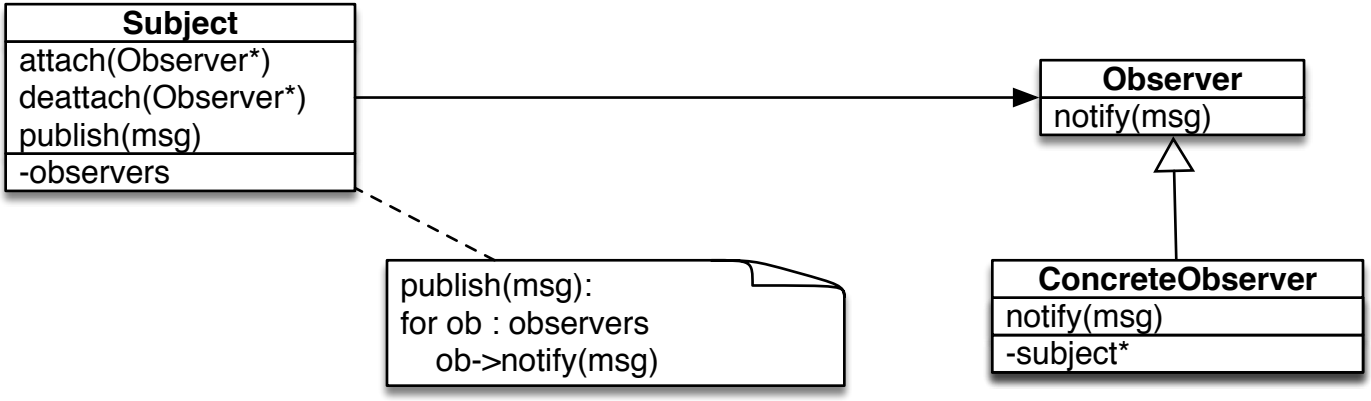
Observer / Publish-Subscribe

Intent: to model a pub-sub relationship between objects; a subject may have any number of subscribers who are interested. Who its subscribers are is of no interest to the publisher. Each subscriber is independent from each other.



In the GoF example, *notify* is called with no param, and *ConcreteObserver* maintains a reference to *Subject*, who has a *getState* call: when an *observer* is notified of a change, call *subject*'s *getState* method to know what's changed.
This difference can be summarized as that of a **push model** (this diagram, where *Subject* has an idea of what it should report) and **pull model** (where *Observer* can and need to query any state it's interested in from the subject). The former (potentially) increases coupling and efficiency, while the latter decreases both. An enum *aspect* can be added to the push model to increase efficiency (so that *observers* know what's changed) without too much coupling.

Consequence of observer is that a subject is decoupled from those interested in it: they can evolve independently. This would also cause a subject to not know the cost of its subscriber's notification handling.