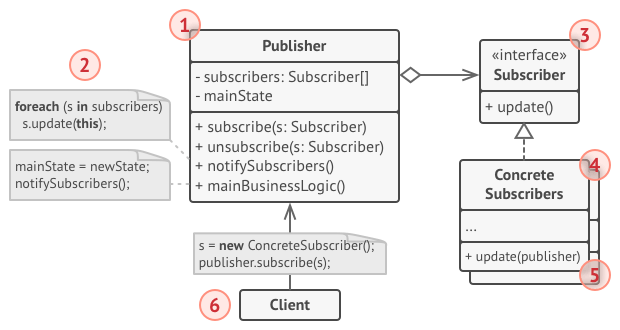
**НАБЛЮДАТЕЛЬ** - определяет отношение "1 ко многим" между объектами таким образом, что при изменении состояния одного объекта происходит автоматическое оповещение и обновление (опционально) всех зависимых объектов. Субъект ничего не знает о наблюдателях, кроме того, что они реализуют интерфейс Observer. При использовании паттерна возможен как запрос, так и активная доставка данных от субъекта (запрос считается более правильным). Работа кода не должна зависеть от порядка оповещения наблюдателей. Java содержит несколько реализаций паттерна, включая обобщенную реализацию java.util.Observable (имеет недостатки - например, java.util.abstract\_observer является классом, а не интерфейсом). Многие GUI-инфраструктуры широко применяют данный паттерн

ПРИМЕНЕНИЕ:

- когда после изменения состояния одного объекта требуется что-то сделать в других, но вы не знаете наперёд, какие именно объекты должны отреагировать

- когда одни объекты должны наблюдать за другими, но только в определённых случаях

****РЕАЛИЗАЦИЯ:

**1. ИЗДАТЕЛЬ**: владеет внутренним состоянием, изменение которого интересно отслеживать подписчикам. Издатель содержит механизм подписки: список подписчиков и методы подписки/отписки.

**2**. Когда внутреннее состояние издателя меняется, он оповещает своих подписчиков. Для этого издатель проходит по списку подписчиков и вызывает их метод оповещения, заданный в общем интерфейсе подписчиков.

**3. ПОДПИСЧИК**: определяет интерфейс, которым пользуется издатель для отправки оповещения. В большинстве случаев для этого достаточно единственного метода.

**4. КОНКРЕТНЫЕ ПОДПИСЧИКИ**: выполняют что-то в ответ на оповещение, пришедшее от издателя. Эти классы должны следовать общему интерфейсу подписчиков, чтобы издатель не зависел от конкретных классов подписчиков.

**5**. По приходу оповещения подписчику нужно получить обновлённое состояние издателя. Издатель может передать это состояние через параметры метода оповещения. Более гибкий вариант — передавать через параметры весь объект издателя, чтобы подписчик мог сам получить требуемые данные. Как вариант, подписчик может постоянно хранить ссылку на объект издателя, переданный ему в конструкторе.

**6. КЛИЕНТ** создаёт объекты издателей и подписчиков, а затем регистрирует подписчиков на обновления в издателях.

ПЛЮСЫ:

- издатели не зависят от конкретных классов подписчиков и наоборот

- вы можете подписывать и отписывать получателей на лету

- реализует принцип открытости/закрытости

МИНУСЫ:

- подписчики оповещаются в случайном порядке