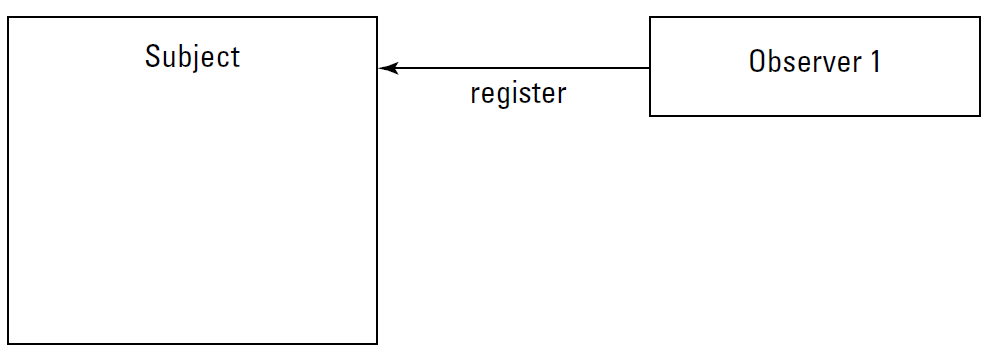
**OBSERVER PATTERN**

Trong phần này chúng ta sẽ xem qua nguyên tắc hoạt động và cách thức cài đặt Observer Pattern như thế nào.

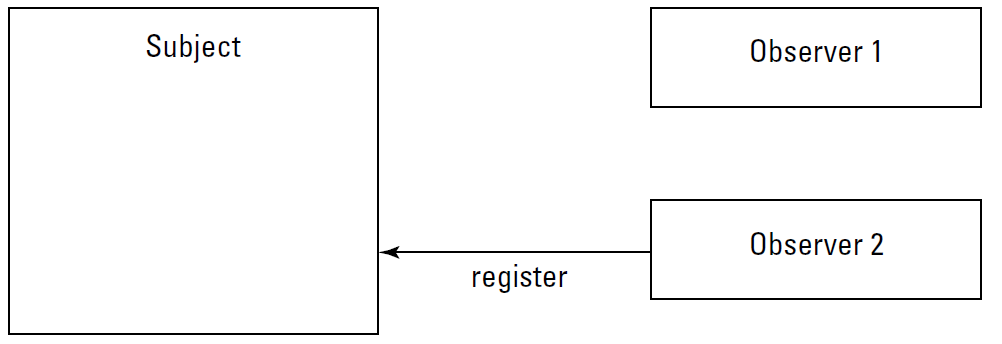
Đầu tiên ta coi qua nguyên tắc hoạt động của Observer Pattern. Trong Observer Pattern sẽ có 2 đối tượng chính, 1 đối tượng là Observer sẽ lắng nghe đối tượng Subject. Nếu Subject có bất cứ thay đổi gì thì đối tượng Observer sẽ được ghi nhận sự kiện đó. Tất nhiên đối tượng Observer phải được đăng ký với Subject rằng sẽ được thông báo khi có sự kiện xảy ra.

Những hình ảnh sau sẽ mô tả hoạt động của Observer Pattern

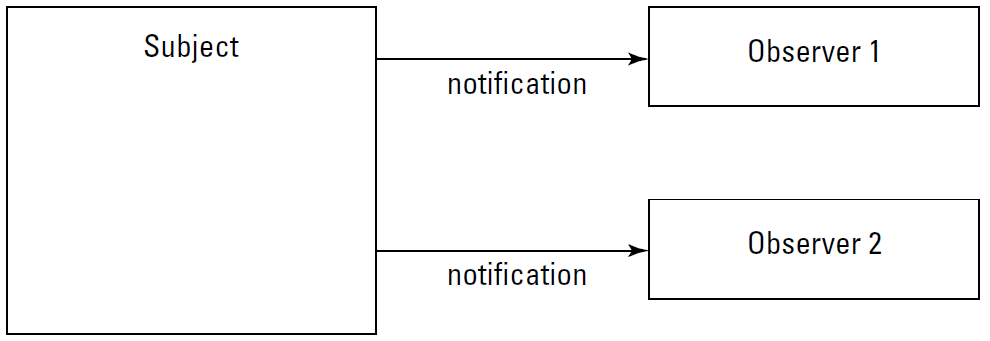
Đầu tiên là đối tượng Observer sẽ đăng ký với Subject



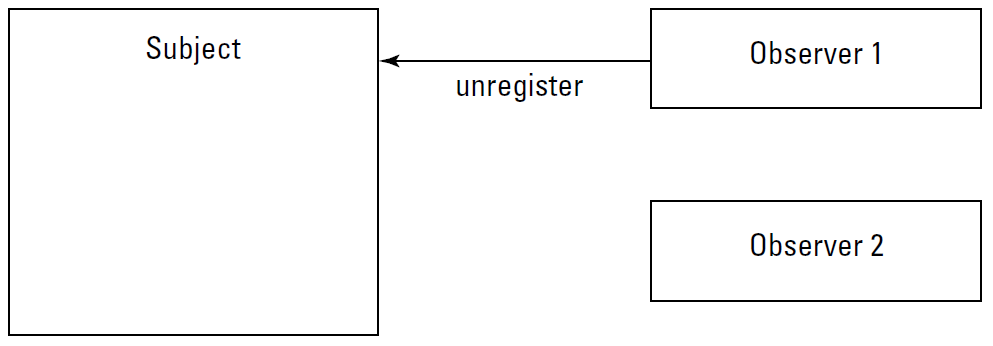
Giả thử ta có thêm 1 Observer được đăng ký



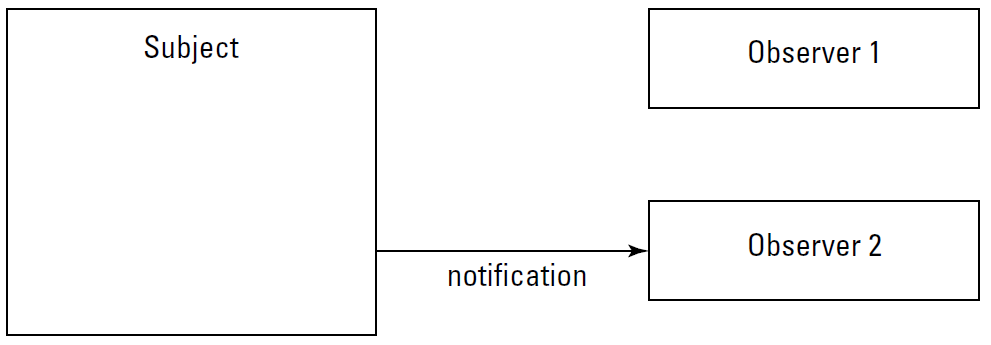
Khi có bất kỳ sự kiện nào thì Subject sẽ thông báo với các Observer được đăng ký



Bây giờ giả sử Observer 1 tắt kết nối với Subject



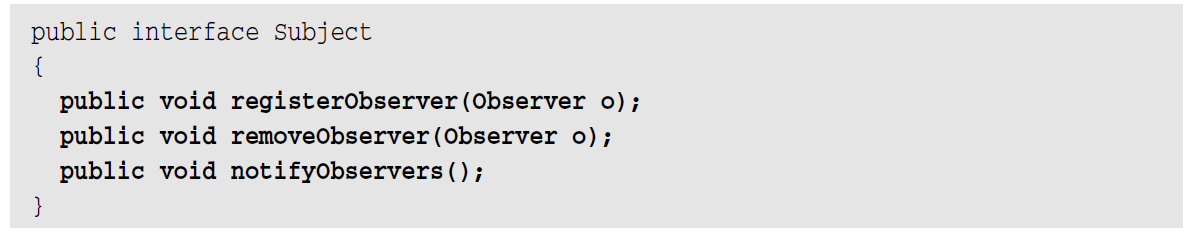
Bây giờ khi có bất cứ sự kiện nào thì Subject sẽ chuyển cho các Observer còn được đăng ký



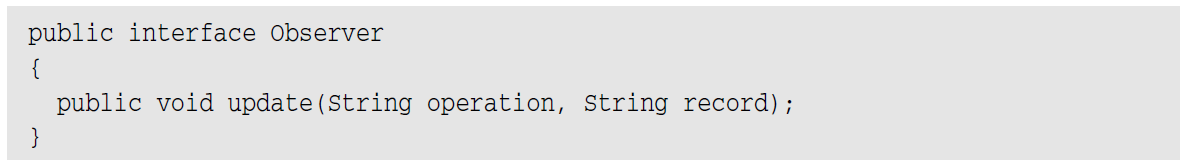
Tuy nhiên Observer 1 cũng có thể quay lại đăng ký bất cứ lúc nào.

Bây giờ chúng ta sẽ bắt đầu xây dựng code, ta để ý kỹ ở đây có 2 thành phần Observer và subject, Chúng ta sẽ xây dựng interface với các thành phần cơ bản trong 2 interface Observer và Subject phục vụ cho việc xây dựng Observer Pattern.

Đầu tiên chúng ta coi qua Subject Interface

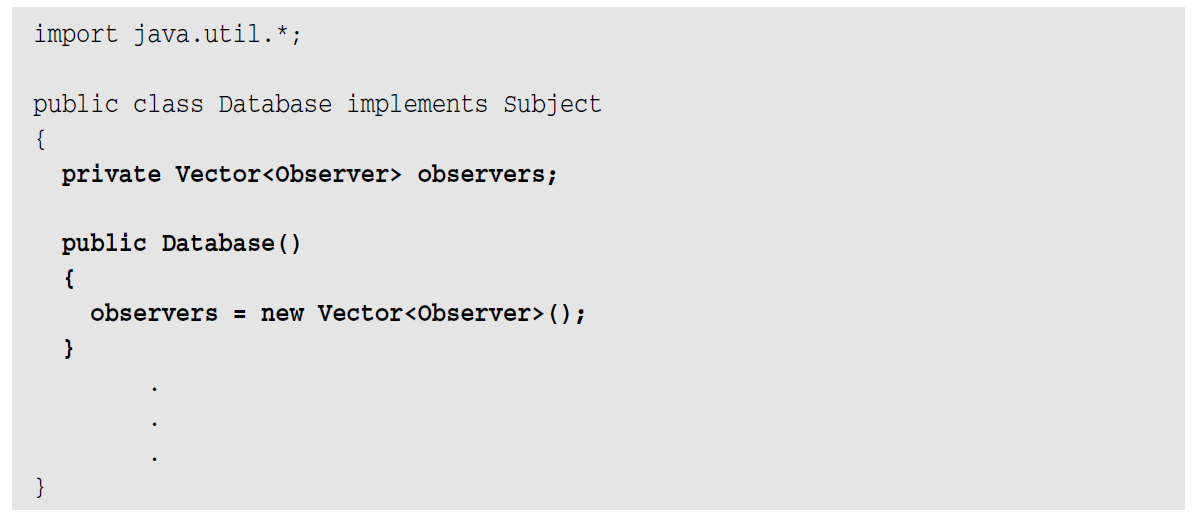


Ở đây chúng ta thấy được 3 thành phần cơ bản phải có trong Subject là đăng ký, hủy đăng ký và thông báo cho Observer, bất cứ đối tượng Subject nào hỗ trợ Observer Pattern cũng phải cần 3 hàm này. Tiếp theo là interface Observer



Observer chỉ cần 1 hàm update để nhận các thông tin từ phía Subject.

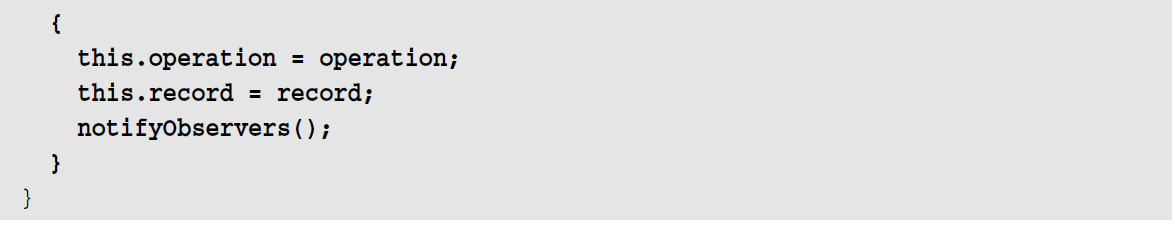
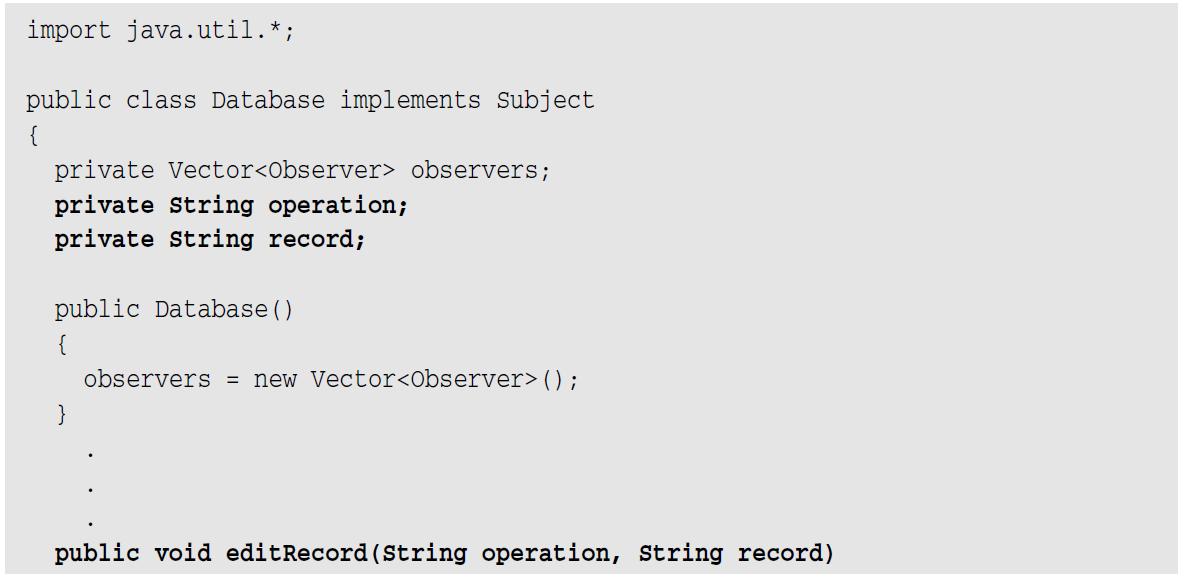
Tiếp theo ta coi cách xây dựng 1 subject cụ thể như thế nào, 1 subject phải có khả năng cho nhiểu observer đăng ký và trả thông tin lại cho observer khi có sự kiện xảy ra, để có thể lưu trữ một loạt các observer thì ta nên sử dụng cấu trúc dữ liệu vector. Ta coi code sau



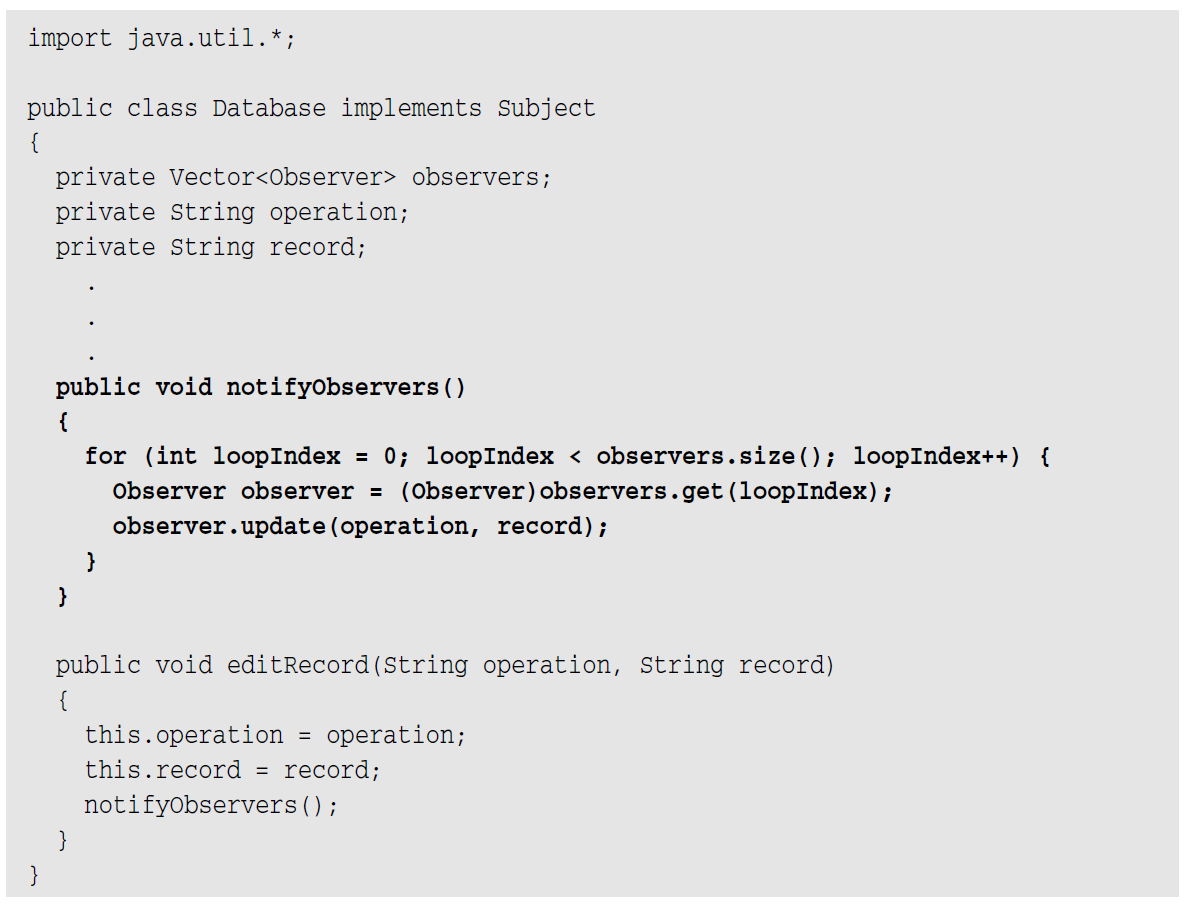
Như vậy ta có thể add observer vào vector này và xóa nó khỏi vector nếu hủy đăng ký.



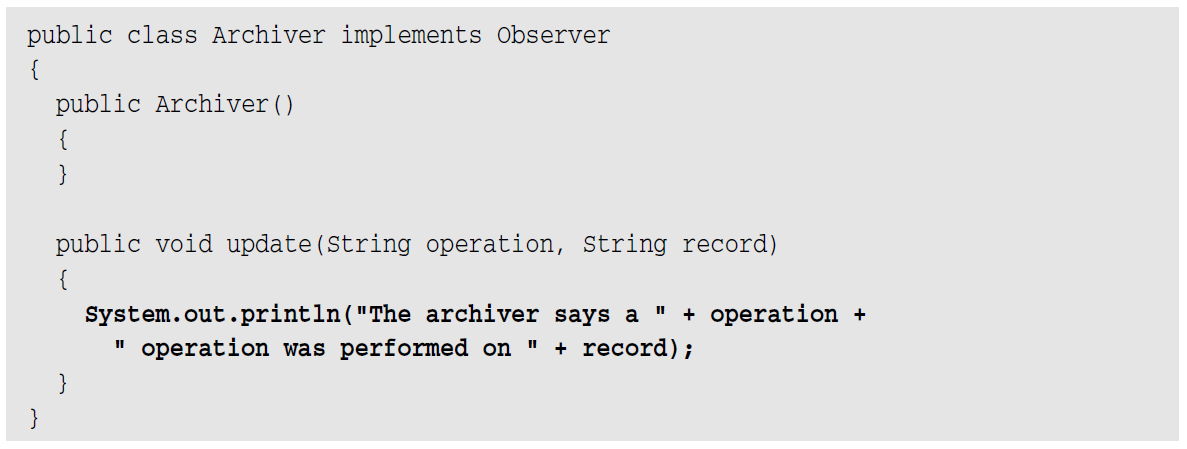
Tiếp theo chúng ta dùng 1 hàm để mô phỏng sự thay đổi trong 1 subject gọi la editRecord, hàm này sẽ thay đổi operator và record.



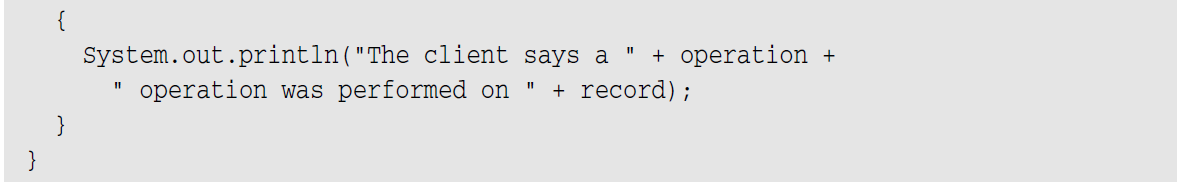
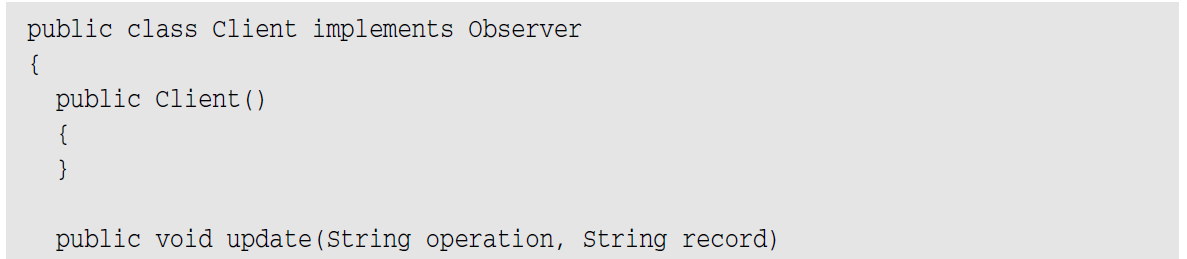
Hàm thay đổi này sẽ gọi cho Observer khi nó được thực thi. Tiếp theo chúng ta sẽ coi hàm notifyObservers thông báo cho các Observer đã đăng ký biết sự thay đổi trong Subject. Chú ý là hàm này sẽ được gọi khi editRecord c hạy



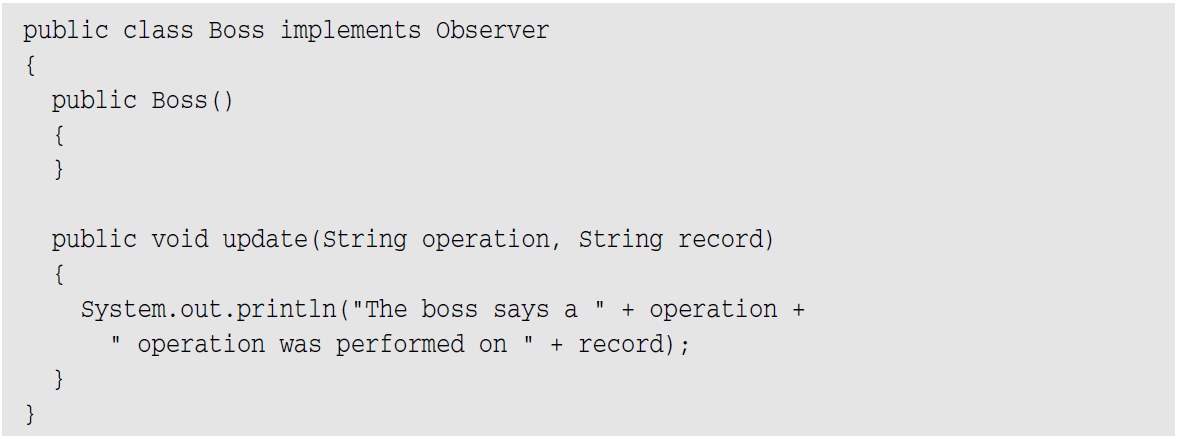
Tiếp theo chúng ta coi cách xây dựng Observer, Observer thì rất đơn giản chỉ cần xử lý hàm update cập nhật các thông tin từ phía Subject



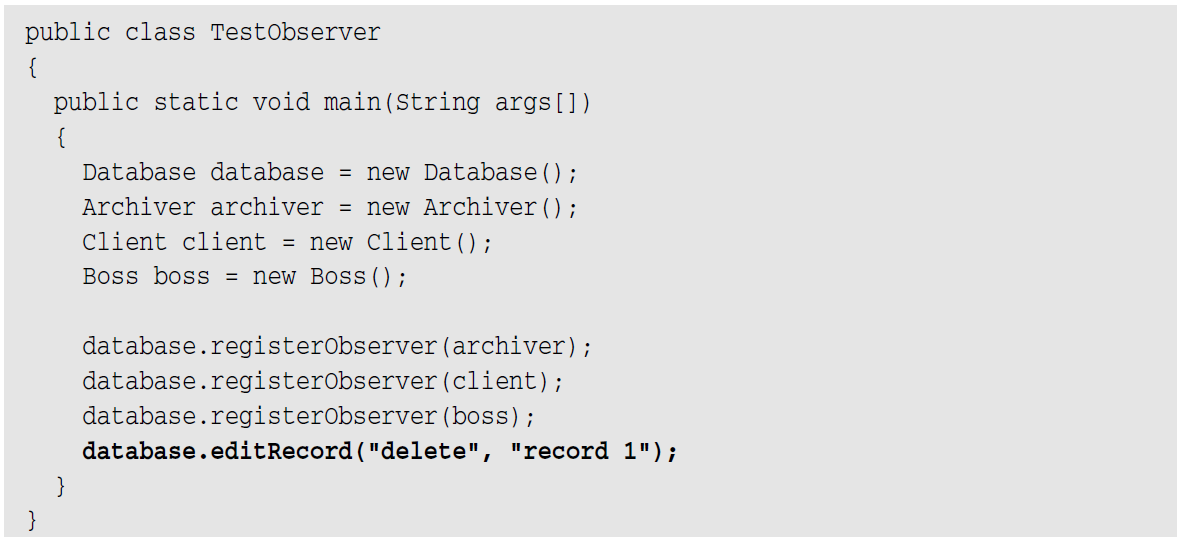
Như trên là một ví dụ về Archiver (đóng vai trò Observer) chúng ta có thể thực hiện nhiều Observer kiểu khác như



Hay



Bây giờ chúng ta test thử với 3 Observer đã đăng ký là archiver, client, boss và 1 sự thay đổi editRecord



Kết quả là 3 Observer sẽ bật lên thông báo

