

**Khi sử dụng mẫu Observer**

**Ví dụ :**

1. **Interface Observer:** Xác định phương thức update để các đối tượng lắng nghe nhận thông báo khi có sự thay đổi.
2. **Interface Subject:** Xác định các phương thức để thêm, xóa và thông báo các observer.
3. **Class CaculatorEntity:** Triển khai Subject, chứa logic tính toán và quản lý danh sách các observer.
4. **Class CaculatorBoundary:** Triển khai Observer, tạo giao diện người dùng và cập nhật kết quả khi có thay đổi.
5. **Class CaculatorControl:** Triển khai Observer và ActionListener, nhận lệnh từ giao diện và cập nhật kết quả tính toán từ CaculatorEntity.
   * Với ví dụ trên Các thành phần tương tác với nhau thông qua các interface Observer và Subject, cho phép các đối tượng trong hệ thống được thông báo và cập nhật khi có sự thay đổi. Đây là một minh chứng rõ ràng cho mẫu thiết kế Observer

**Khi sử dụng mẫu Observer với Java Beans và PropertyChangeListener:**

**Java Beans và PropertyChangeListener cung cấp một cơ chế tích hợp sẵn để các đối tượng có thể đăng ký và nhận thông báo khi một thuộc tính của một đối tượng khác thay đổi, giúp tách biệt rõ ràng giữa logic nghiệp vụ và giao diện người dùng, đồng thời hỗ trợ việc cập nhật tự động và đồng bộ hóa trạng thái.**

**Ví dụ:**

Java Beans và PropertyChangeListener. Các thành phần sẽ tương tác như sau:

* CaculatorEntity là nguồn phát sự kiện (Subject). Khi kết quả thay đổi, nó sẽ thông báo cho tất cả các listener (observer) đã đăng ký.
* CaculatorControl là observer, lắng nghe các thay đổi từ CaculatorEntity và cập nhật giao diện thông qua CaculatorBoundary.
* CaculatorBoundary cập nhật hiển thị khi có thay đổi từ CaculatorControl.