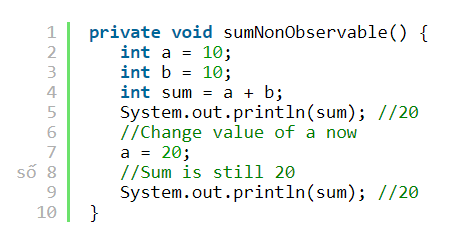
**1.Tìm hiểu về Observable trong JavaFX**

JavaFX Observable là một thực thể bao bọc nội dung và cho phép quan sát nội dung để tìm các trường hợp không hợp lệ. Nói một cách dễ hiểu, Observable cho phép quan sát dữ liệu trong ứng dụng của bạn. Trong lập trình hàng ngày, chúng tôi đưa ra các trường hợp phải liên tục kiểm tra xem giá trị của biến có thay đổi hay không. Với JavaFX có thể quan sát được, chúng ta có thể ràng buộc hoặc đính kèm người nghe vào dữ liệu để tất cả các thay đổi sẽ được theo dõi.

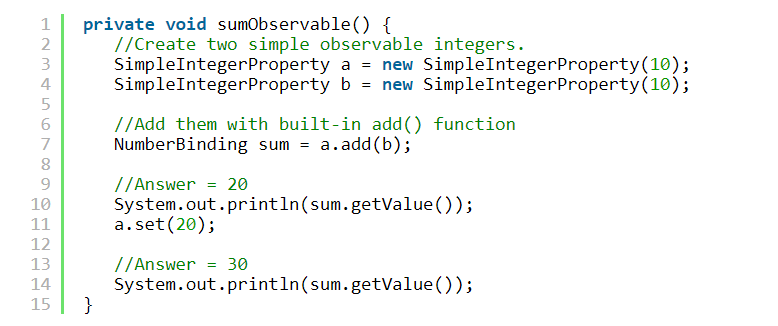
Hãy xem xét trường hợp tính tổng hai số



Đoạn code trên hiển thị tổng 2 số. Hai biến a = 10 và b = 10 được cộng và gán thành tổng. Sau đó, a được đổi thành 20. Tổng sẽ không thay đổi.

**Observable Addition**

Điều gì sẽ xảy ra nếu bạn muốn tổng để theo dõi các thành phần của nó. Giả sử khi thay đổi a thành 20, bạn cần tổng tự động trở thành 30 mà không cần thực hiện thêm một lần nữa ?.JavaFX có giải pháp phù hợp.

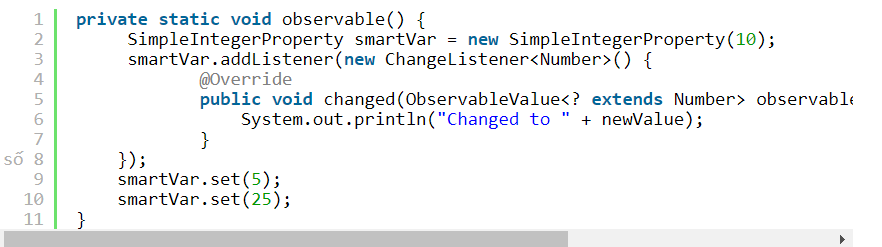


SimpleIntegerProperty là một lớp có thể quan sát được tích hợp sẵn để lưu trữ giá trị số nguyên với sự hỗ trợ có thể quan sát được. Sau đó  a.add (b) sẽ trả về một NumberBinding giữ tổng của a và b. Ưu điểm là, tổng bây giờ đang nghe trên a và b và bất kỳ thay đổi nào được thực hiện đối với a hoặc b hoặc cả hai, nó sẽ được phản ánh ngay lập tức thành tổng.

### Adding Listener:

Chúng ta đã thấy các vật có thể quan sát hữu ích như thế nào trong số học. Đó chỉ là phần nổi của tảng băng chìm. Hãy xem cách chúng ta có thể sử dụng những người nghe có khả năng quan sát.

Ví dụ mã sau sẽ in một thông báo bất cứ khi nào giá trị của một biến được thay đổi.



### Kết quả đầu ra :

### 

Vì vậy, bất cứ khi nào giá trị được thay đổi, ChangeListener sẽ được gọi.

Bạn có thể thêm hai loại người nghe vào một người có thể quan sát được.

ChangeListener  
Một ChangeListener được thông báo bất cứ khi nào giá trị của ObservableValue thay đổi.

InvalidationListener  
An InvalidationListener được thông báo bất cứ khi nào một Observable trở nên không hợp lệ.

**Sự khác biệt giữa ChangeListener và InvalidationListener là gì?**

Một ObservableValue tạo ra hai loại sự kiện; sự kiện thay đổi và sự kiện làm mất hiệu lực. Một sự kiện thay đổi chỉ ra rằng giá trị đã thay đổi. Một sự kiện vô hiệu được tạo ra, nếu giá trị hiện tại không còn hợp lệ nữa. Sự phân biệt này trở nên quan trọng, nếu ObservableValue hỗ trợ đánh giá lười biếng. Bởi vì đối với một giá trị được đánh giá một cách lười biếng, người ta không biết liệu một giá trị không hợp lệ có thực sự thay đổi hay không cho đến khi nó được tính lại. Vì lý do này, việc tạo các sự kiện thay đổi yêu cầu đánh giá háo hức trong khi các sự kiện vô hiệu có thể được tạo cho các triển khai háo hức và lười biếng.

ChangeListists ngược lại với InvalidationListists yêu cầu giá trị mới phải được chuyển cho chúng. Vì vậy, việc thêm bất kỳ ChangeListener nào sẽ dẫn đến việc get () được gọi và mọi lệnh gọi không hợp lệ sẽ kích hoạt một lệnh gọi computeValue.

**2.So sánh trong source codebase**

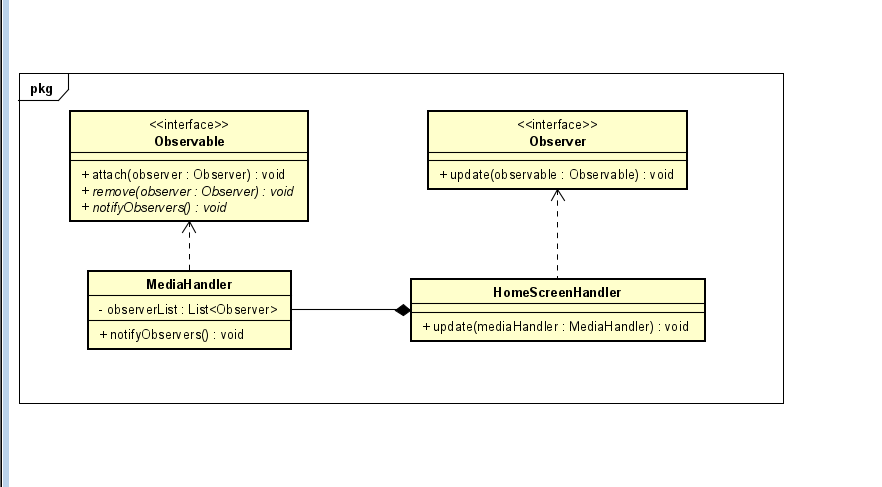
Hiện tại trong source codebase có sử dụng thiết kế Observer pattetn trong các module sau:

* Interface Observable đóng vai trò như Subject cung cấp các phương thức có thể thêm, loại bỏ các observer
* Interface Observer đóng vai trò như Observer định nghĩa một phương thức update() cho các đối tượng sẽ được subject thông báo đến khi có sự thay đổi trạng thái
* Cả hai interface đều nằm trong module common/interface

Trong module views/screen có 2 class sử dụng mẫu thiết kế Observer pattern :

* MediaHandler có nhiệm vụ notify, nó như 1 item hay mặt hàng trên trang home
* HomeScreenHandler sẽ update trạng thái khi nhận được thông báo thông qua hàm update, cụ thể sẽ hiển thị thông báo khi mặt hàng( item) được thêm thành công vào giỏ hàng

Sơ đồ codebase đang sử dụng Observer pattern



**Áp dụng Observer vào codebase**

Diagram

Description automatically generated

Trong hình trên MediaHandler sẽ đóng vai trò là Subject để update thông tin trên các Observer đã được subscribe mỗi khi có thay đổi xảy ra còn CartScreenHandler sẽ đóng vai trò là Observer, mỗi khi nhận được thông báo thay đổi từ MediaHandler, sẽ update thông tin Cart thông qua hàm update () updateCart(), updateCartAmount().