Nhóm 15

1. Tìm hiểu cơ chế hoạt động của Observable trong JavaFX :

1.1. Interface Observable:

Public abstract void addListener(javafx.beans.InvalidationListener arg0);
Public abstract void removeListener(javafx.beans.InvalidationListener arg0);
Interface Observable của JavaFx chỉ có 2 phương thức addListener và removeListener và không có phương thức notify
addListerner :
Thêm một ChangeListener sẽ được thông báo bất cứ khi nào giá trị của ObservableValue thay đổi. Nếu cùng một người nghe được thêm nhiều hơn một lần, thì nó sẽ được thông báo nhiều hơn một lần. Có nghĩa là, không có kiểm tra nào được thực hiện để đảm bảo tính duy nhất.
removeListener :
Xóa Listerner đã cho khỏi danh sách Listerner, được thông báo bất cứ khi nào giá trị của Observable được trở nên không hợp lệ.
Nếu Listerner nhất định chưa được đăng ký trước đó (tức là nó chưa bao giờ được thêm vào) thì cuộc gọi phương thức này là không chọn. Nếu nó đã được thêm trước đó thì nó sẽ bị xóa. Nếu nó đã được thêm nhiều lần, thì chỉ lần xuất hiện đầu tiên sẽ bị xóa.

1.2. Các modul khác : Observable integer, map, collection

InterfaceJavaFX Observable là interface cơ sở cho nhiều lớp vùng chứa và interface trong JavaFX Collection Framework. Ví dụ, ObservableList và ObservableArray là hai triển khai phổ biến của giao diện Observable cơ sở. Java core Collection API đã chứa nhiều lớp vùng chứa hữu ích để đại diện cho cấu trúc dữ liệu chung của lists, set, and maps. Ví dụ, java.util.ArrayList là một thực thi mảng khá lớn của interface java.util.List để chứa một danh sách các đối tượng. Tuy nhiên, chúng không có khả năng hoạt động liền mạch khi chức năng đồng bộ được yêu cầu giữa mô hình danh sách và thành phần chế độ xem trong kịch bản GUI. Các đối tượng Observable và Observer đóng một vai trò quan trọng trong việc triển khai kiến trúc Model-View-Controller trong Java Observable là một class và Observer là một interface. Chúng được đặt trong java.util package. Một lớp muốn được extends lớp Observable phải gọi phương thức setChanged () nếu nó có thay đổi và phải kích hoạt thông báo bằng cách gọi phương thức

notifyObservers () . Điều này lần lượt gọi phương thức update () của lớp observer . Tuy nhiên, nếu một đối tượng gọi phương thức InformObserver() mà không gọi phương thức setChanged () thì sẽ không có hành động nào diễn ra. Do đó, một đối tượng observable phải gọi phương thức setChange () trước khi gọi phương thức notifyObservers () khi có bất kỳ thay đổi nào. Thứ tự này sau đó sẽ gọi phương thức update () của các lớp quan sát một cách liền mạch.

1.3. Sự khác biệt giữa ChangeListenr và InvalidationListener là :

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, vì đối với một giá trị được đánh giá lười biếng, người ta không biết liệu 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.

ChangeListener thực thi tính toán mong muốn ngay cả khi giá trị có thể quan sát được hỗ trợ đánh giá lười biếng.

Nếu bạn muốn tìm thay đổi đã xảy ra trong thuộc tính được quan sát, bạn sử dụng trình nghe thay đổi. Vì không hợp lệ hóa danh sách chỉ giúp chúng tôi biết một số thay đổi đã xảy ra, nếu bạn muốn biết sự khác biệt giữa các giá trị cũ và mới, bạn phải tự tính toán hoặc đơn giản là sử dụng trình nghe thay đổi.

2. So sánh với mẫu thiết kế Observer :
Observable là thuộc mẫu Observer. cả hai đều phụ thuộc vào sự trừu tượng được cung cấp bởi Interface Observable và vì chúng dựa vào Client code sử dụng chúng để kết nối Observer / Listener với Observable.
Điểm khác nhau :

Observer	Observable in JavaFX
Trong mẫu Observer, bạn có một Observer trừu tượng (an interface or a base class) định nghĩa những thứ mà observer có thể quan sát (hoặc lắng	Observable, là đối tượng gửi thông báo, bản thân nó có thể là một phần trừu tượng hoặc một phần của hệ thống phân cấp nào đó. Tức là, mẫu không yêu cầu

nghe). CurrencyListener của bạn là Observer..	phải có các lớp con ConcreteObservable. Nếu ConcreteObserver có thể không cần bất kỳ trạng thái đặc biệt nào hoặc chỉ có thể hoạt động với giao diện được cung cấp bởi lớp cơ sở Observable, thì không cần ConcreteObservable.

3. Xem xét Codebase liệu có thể áp dụng cơ chế Observable hay mẫu thiết kế Observer để cải thiện một cách hợp lí hay không ?
   Codebase đã áp dụng Observer Pattern như sau :
   - Interface Observable là Subject chứa, thông báo dữ liệu khi có thay đổi
   - Interface Observer đóng vai trò là Observer , chứa hàm update, khi nhận được notify sẽ update lại

   Lớp MediaHandler và MediaScreen đã sử dụng mẫu thiết kế này.

   Áp dụng : CartMediaHandler và CartScreenHandler.

   Lớp CartMediaHandler có thể implement Observable