**JavaFx Observable**

The JavaFX Observable interface is the base interface for many container classes and interfaces in the JavaFX Collection Framework. For example, ObservableList<E> and ObservableArray<T> are two common implementations of the base Observable interface. The Java core Collection API already contains many useful container classes to represent the generic data structure of lists, set, and maps. For example, java.util.ArrayList<E> is a re-sizable array implementation of the java.util.List<E> interface to contain a list of objects. However, they are incapable of working seamlessly when synchronous functionality is required between the list model and the view component in a GUI scenario.

Before JavaFX, Swing developers relied on ArrayList to contain a list of objects and subsequently display them in a list-like UI control, such as JList. But, ArrayList is too generic and was not built keeping in mind the requirement of synchronization when associated with a view component. As a result, it is quite difficult to update, add, or remove objects from the model list and at the same time reflect changes in the view component. To overcome this problem, JavaFX uses observable interfaces and their implementation, such as ObserverList<E>.

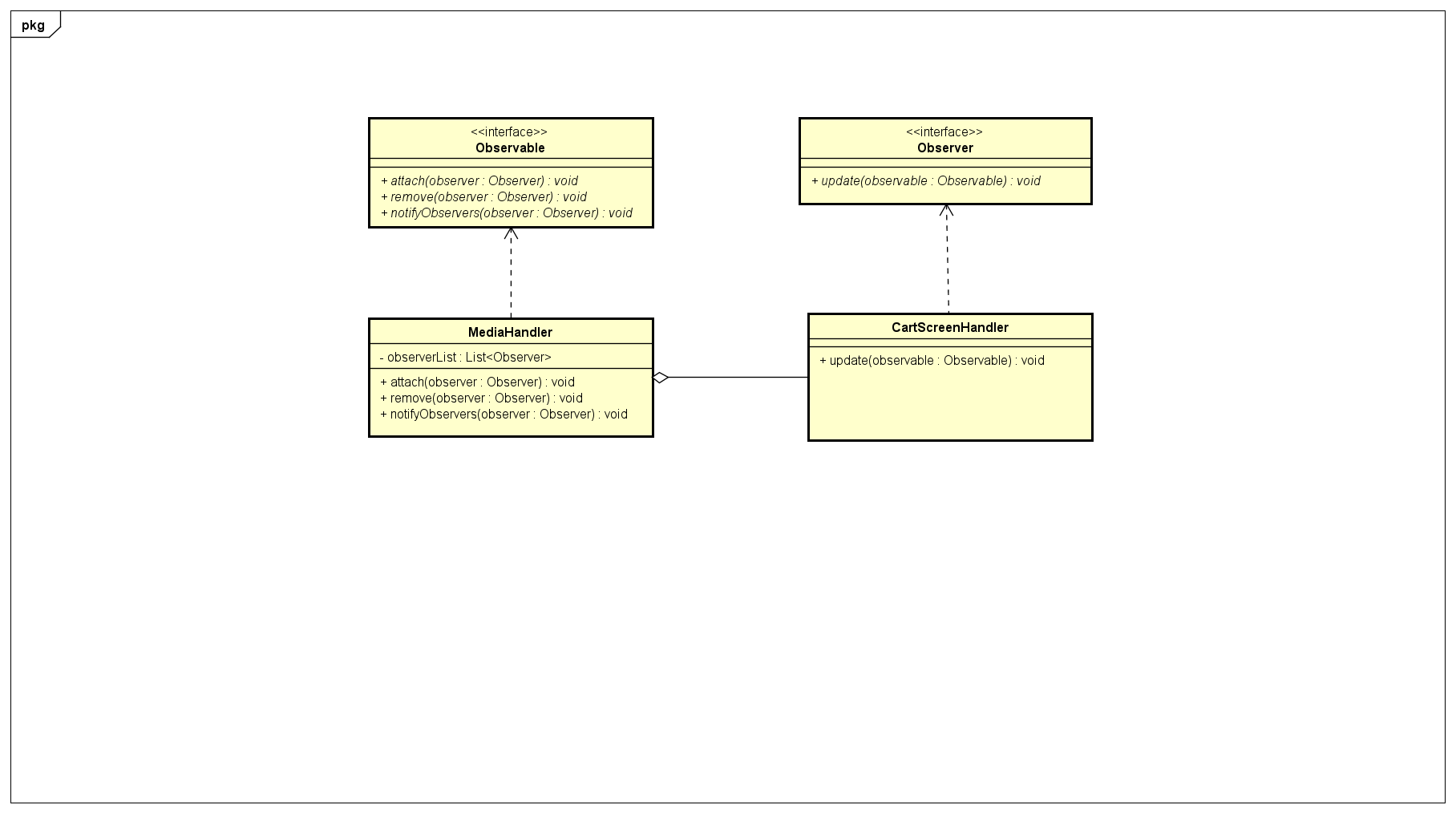
Because ObservableList<E> adheres to the rules of the observable and observer paradigm of MVC, such as providing notification to its interested observer regarding any updation, addition or removal of objects from the model list, it became a de-facto container for using any lists in the JavaFX arena. Here, data representation in the model and view are synchronized seamlessly. JavaFX ObserverList<E> is typically used in UI controls such as ListView and TableView. Let’s go through a quick example to see how ObservableList<E> is actually used.

JavaFx Observable implemented based on Observer pattern including:

+ Subject : Ex: ObservableList

+ Observer: Ex: ObserverList

* **Áp dụng Observer Pattern vào CodeBase**

****

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