Java Observable and Observer

Any Java class interested in being observable extends the Observable class. The extended class then can be observed by other interested classes in other parts of the program. The registered classes are notified as soon as any change is triggered in the observable class to which the observer class is registered to. This is the relation between the Observable and Observer classes. Observer classes are nothing but the implementation of the Observer interface. The Observer interface provides an update() method that is called when an observer is notified of a change in the observed object.

A class that wants to be observed or the class that extends Observable must call the setChanged() method if it has changed and must trigger notification by calling the notifyObservers() method. This in turn calls the update() method of the observer class. If, however, an object calls the notifyObserver() method without calling the setChanged() method, no action takes place. Therefore, an observable object must call the setChange() method prior to calling the notifyObservers() method on the event of any change. This order would subsequently call the update() method of the observer classes seamlessly.

The Observable class and the Observer interface provide a invaluable impetus to implement sophisticated program architectures based on the Model-View-Controller paradigm. In fact, the observable and observer concept is more than just MVC. A good understanding of the underlying concept can help create more sophisticated code in Java. If observer and the observable portions of the Java core utility framework show the principle, the JavaFX observable classes and interfaces exactly show how those principles can be put to work.