Everything on MVC Architecture

It is good to adopt a well understood overall organizational pattern as framework around which to build the class diagram.

One such framework is Model-View-Controller (MVC). This is useful for highly interactive applications and is based on the Publisher-Subscriber design pattern where the Model is the Publisher and the View is the Subscriber.

We have a single model that represents the information content of our system of interest.

We have one or more controllers that are boundary classes who receive inputs and send messages to the model. Each of these has one or more associated views which are notified by the model whenever its content changes.

MVC Architecture Representation

Example: A large modern supermarket

Model: Stock Database, Price Database, Customer Database
Controllers: Service tills, warehouse office, barcode scanners
Candidate classes: Till, TillView, Manager, ManagerView

Java Library Classes

Observer is the Subscriber role: it is an interface.

It's only operation is :

Public void update(Observable o, Object obj)

Object obj can be a string, any other object or null

Each View implements Observer. The Model sends a message (update) to signal it has changed. The View's update() implements the appropriate action.

Observable is the Publisher role: it is a class.

Principle operations are:

public void addObserver()	
public void setChanged()	Change boolean flag
public void notifyObservers()	Notifies all observers and contains boolean flag
public void notifyOberservers(Object obj)	Notifies all observers and contains boolean flag

Models extend Observable or include it by aggregation

Implementation decisions

Treat each controller and its views as appearing in one JFrame window (i.e. all in a single screen)

Each view with a controller will be a JPanel.

Controller extends JFrame with a constructor parameter of a Model.

View extends JPanel with constructor parameters of a Controller and a Model.