Observer Pattern

Context:

An object (the *Subject*) is the source of interesting events. Other objects (*Observers*) want to know when an event occurs.

Solution:

(1) Subject provides a method for Observers to register themselves as interested in the event.

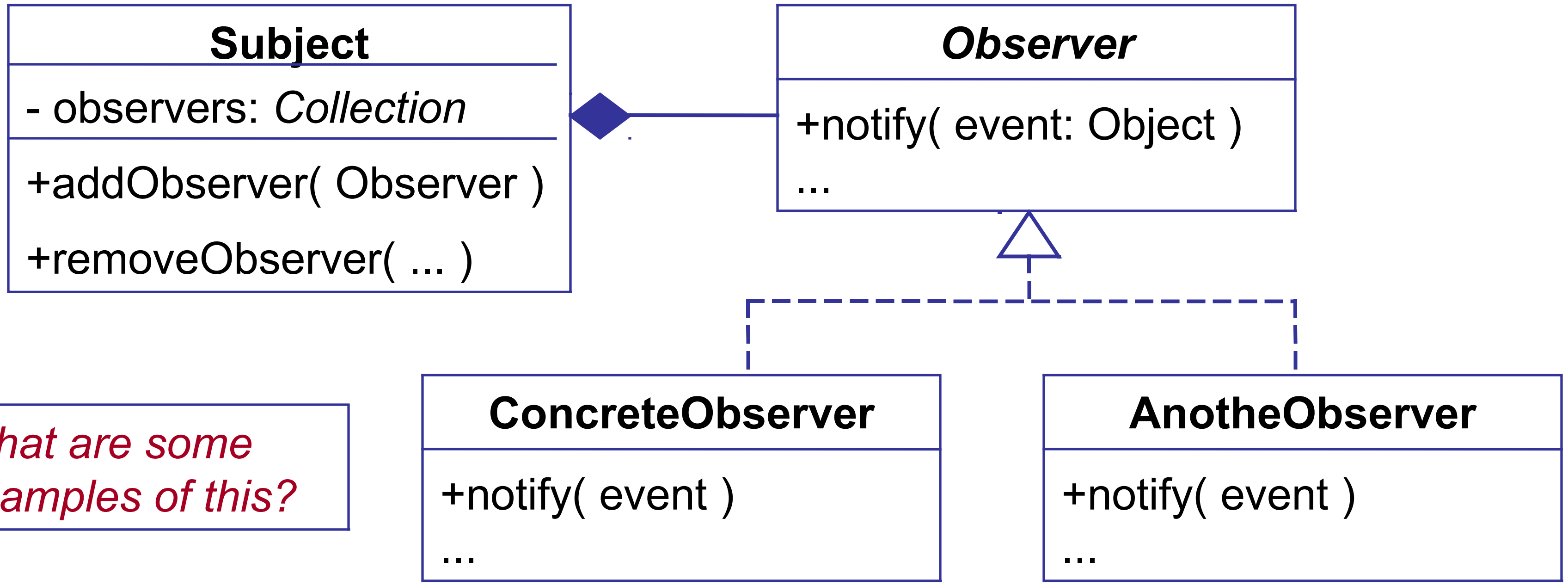
(2) Subject calls a known method (*notify*) of each Observer when event occurs.

Observer Pattern

Context: An object (the *Subject*) is the source of interesting events. Other objects (*Observers*) want to know when an event occurs.

Solution: (1) Subject provides a method for Observers to register themselves as interested in the event.

(2) Subject calls a known method (*notify*) of each Observer when event occurs.



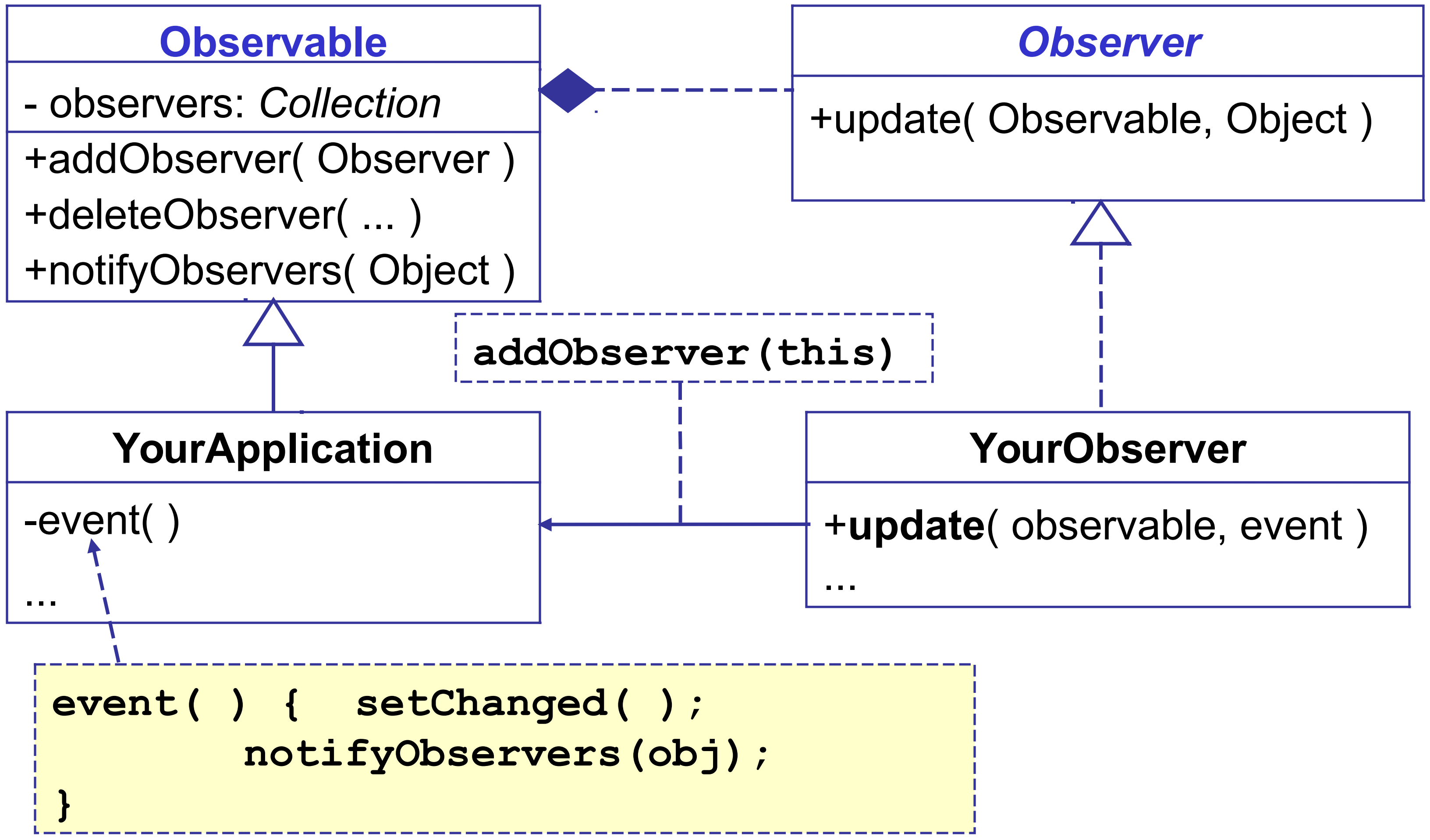
What are some examples of this?

Table for Identifying a Pattern

Name In Pattern	Name in Application: this is for a JButton
Subject *Observer* Concrete Observer	JButton *ActionListener* a class that implements *ActionListener*
addObserver(Observer)	addActionListener( )
notify(Event) [in the observer]	actionPerformed(ActionEvent)

Observer Pattern in Java

Java provides an **Observable** class and **Observer** interface that make it *easy* to use the Observer pattern..



```
event( ) {  setChanged( );
          notifyObservers(obj);
}
```

Using the Observable class

(1) Declare that your class extends Observable

```
public class MySubject extends Observable
{
    Object myinfo;

  /** An event the observers want to know about */
  public void event( ) {
  doSomeWork( );
  // now notify the observers
  setChanged( );
  notifyObservers( ); // can include a parameter
  }
```

(2) When an event occurs, invoke setChanged() and notifyObservers( )

Writing an Observer

(3) Declare that observers *implement* the Observer interface.

```
public class MyObserver implements Observer {
  /* This method receives notification from the
   * subject (Observable) when something happens
   * @param message is value of parameter sent
   *   by subject in notifyObservers. May be null.
   */
  public void update( Observable subject,
                      Object message ) {
    info = ((MySubject)subject).getInfo( );
  ...
  ...
}
```

(4) update takes action using notification from the Subject.

Connecting Observer to Subject

Call addObserver( ) to add Observers to the subject. You can have many Observers.

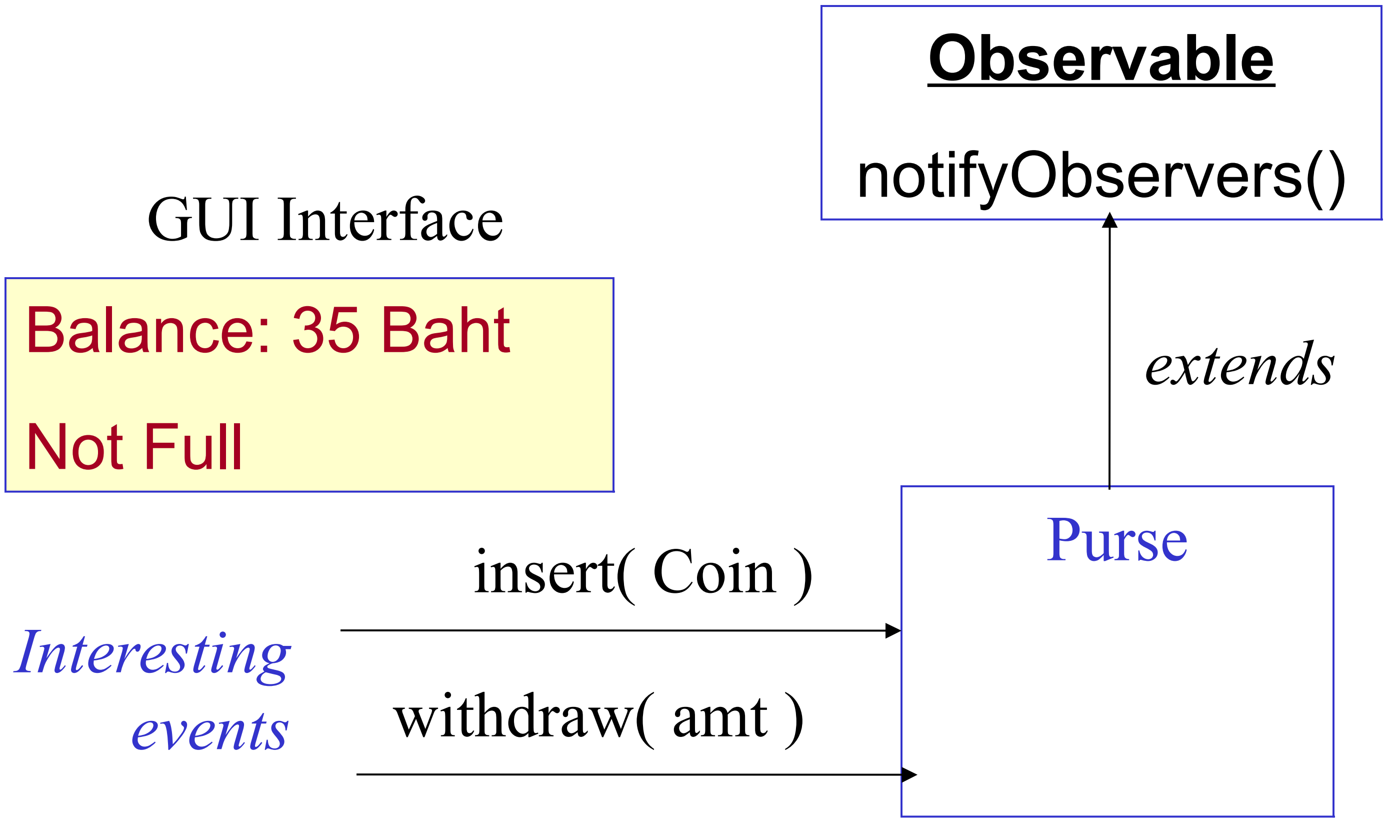
```
public static void main(String [] args) {
    Observable subject = new MySubject( );
    MyObserver observer = new MyObserver( );

    subject.addObserver( observer );

    subject.run( );

}
```

Example for Coin Purse



C# Delegates as Observers

- Delegate is a type in the C# type system.
- It describes a group of functions with same parameters.
- Delegate can act as a collection for observers.

```
/** define a delegate that accepts a string **/
public delegate void WriteTo( string msg );
```

```
/** create some delegates **/
WriteTo observers = new WriteTo( out.WriteLine );
observers += new WriteTo( button.setText );
observers += new WriteTo( textarea.append );
/** call all the observers at once! **/
observers("Wake Up!");
```