Observer / Observable

COMP 401 Spring 2013

Lecture 19

3/26/2013

Observer / Observable

- Official Gang of Four description:
  - Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
- Observable
  - The object of "interest"
    - Represents data or state that may change in the future.
- Observer
  - The "interested" objects
    - Represents parts of system that need to be updated or may need to take action if/when observable object changes.

Use Cases

- User Interfaces
  - User interface elements like buttons, scrollbars, etc. are "observable"
    - State changes correspond to clicks, drags, etc.
  - Application objects that must respond to user's interactions with the UI are "observers"
- Asynchronous Programming
  - Also known as "event-based" programming
  - May have well-defined actions corresponding to events that may occur, but can't know in advance which event will occur or when it will occur.

Basic Observer/Observable

```
class Observable {
  List<Observer> observers;

  void addObserver(Observer o) {
    // Adds o to list of observers
    observers.add(o);
  }

  void deleteObserver(Observer o) {
    // Takes o off list of observers
    observers.remove(o);
  }

  void notifyObservers() {
    // Trigger update method on
    // all observers
    for (Observer o : observers) {
       o.update();
    }
  }
}
```

```
interface Observer {
  void update();
}
```

- Defining Observer as an interface allows any class to act as an observer for the Observable class.
- Notice that Observable can have more than one observer.
  - And that they don't know about each other or the order of update.

lec19.v1

- Game acts as Observable class
- GameObserver acts as Observer interface
- Discussion
  - How many games can one fan watch?
  - What if we wanted to have fan objects that could watch more than one game simultaneously?

Observer/Observable Refined

```
class Observable {
  List<Observer> observers;

  void register(Observer o) {
    // Adds o to list of observers
    observers.add(o);
  }

  void unregister(Observer o) {
    // Takes o off list of observers
    observers.remove(o);
  }

  void notifyObservers() {
    // Trigger update method on
    // all observers
    for (Observer o : observers) {
       o.update(this);
    }
  }
}
```

```
interface Observer {
  void update(Observable o);
}
```

- Passing reference to observable as a parameter to update method allows Observer to register with more than one Observable and then detect which one changed.

lec19.v2

- Same observers watching multiple games
  - Game object passed to fan as parameter to update method.
  - So far, fans are reacting to current state of game. But in real life, what do we react to?

Observer/Observable Refined More

```
class Observable {
  List<Observer> observers;

  void register(Observer o) {
    // Adds o to list of observers
    observers.add(o);
  }

  void unregister(Observer o) {
    // Takes o off list of observers
    observers.remove(o);
  }

  void notifyObservers(Info i) {
    // Trigger update method on
    // all observers
    for (Observer o : observers) {
       o.update(this, i);
    }
  }
}
```

```
interface Observer {
  void update(Observable o, Info i);
}
```

- In addition to passing reference to Observable that changed, we can encapsulate *what* changed as additional information for the update method of the Observer.

lec19.v3

- GameObserver update method now takes second argument
  - String who_scored
    - This is the “info” about the change in the game that is being passed to the fans

Observer/Observable in java.util

- Java provides skeleton Observer / Observable that you can extend in java.util
  - Don't have to use them.
    - Our examples so far haven't
  - The pattern is defined by relationship between objects and their interaction.
    - Not the specific method names and/or implementation.
    - Should be able to recognize Observer/Observable by these characteristics:
      - Observer object some how registers with observable.
      - Observable invokes a method on observers in order to signal state changes.

lec19.v4

- Game extends java.util.Observable
  - No longer have to manage our own list since we inherit that from Observable
- UNCFan and DukeFan implement java.util.Observer
  - Parameters to update are no longer context specific.
    - update(Observable o, Object arg)
  - Need contravariant cast to make them useful
- Observable parent class requires state change to be signaled by call to setChanged() before calling notifyObservers.
  - Otherwise, won't do anything.

Delegating java.util.Observable

- Using Observable is convenient
  - Inherits basic Observable functionality without having to worry about doing it yourself.
- Must subclass observable.
  - Suppose your class already has a parent due to design or circumstance.
    - Don't have development control over class hierarchy.
    - Parent class is part of third party library.
- Solution lies with delegation.

Delegating Observable

- Outline of solution:
  - Create a helper class that does subclass Observable
    - Implements same interface as original class via delegation.
    - Override notifyObservers to call setChanged() first.
  - Create an instance of this helper class as part of your object.
  - Provide methods from Observable via delegation to this instance.
- Delegation is being used both ways here.
  - Allows Observer to use "hidden" observable instance as if it were the original object.
  - Allows outside code to interact with original object as if it were Observable.
- lec19.v5

Multiple Observer Lists

- Same delegation trick can be used to support multiple lists of observers.
  - Useful if you want to support more than one type of observable changes/events.
  - Requires providing a way to specify what is being observed.
    - One approach: separate registration methods for each observable change/event.
    - Another approach: additional argument to registration method.
  - May need to provide way of distinguishing between events as part of update info if you want the same observer to be able to observe more than one observable event.
- lec19.v6