Lecture 16
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.
Basic Observer/Observable

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

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

class Observable {
    ArrayList<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();
        }
    }
}
```
Notice how Game class inherits basic Observable characteristics from parent class.

– Discussion points:
  • What happens if we want to create a fan that is watching more than one game?
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.
• Same observers watching multiple observables.
  – Notice cast from Observable to Game
    • Observer/Observable mechanisms defined for general use by interface and parent class.
    • Access to specific behavior requires casting to specific class.
      – But this is OK, since we know that the UNCFan and DukeFan objects will only by observing instances of Game.
  – So far, fans are reacting to current state of game. But in real life, what do we react to?
In addition to passing who changed, we can encapsulate what changed as additional info to be passed to the update method of the Observer.
Observer/Observable in Java

- Java provides skeleton Observer / Observable that you can extend.
  - Don’t have to use them.
  - May contain more functionality than you really need.
  - 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 somehow registers with observable.
      - Observable invokes a method on observers in order to signal state changes.