Multiple Observer / Observable
Midterm 2
Using java.util.Observable

• Using Observable is convenient
  – Inherits basic Observable functionality without having to worry about doing it yourself.

• Must subclass observable.
  – Not useful to create direct instantiations.
    • Possible to ask for notification from outside of the class, but no way to call setChanged().

• lec17.v1
What if you can’t

• 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.
Delegating Observable

• Solution lies with delegation.
• Outline of solution:
  – Create a helper class that does subclass Observable.
    • Can be non-public if helper is only going to be used within this class for this purpose.
    • Override notifyObservers to call setChanged() first.
  – Create an instance of this class as part of your object.
  – Delegate observable functions to this instance.
• lec17.v2, lec17.v3
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.

• lec17.v4, lec17.v5
Observing more than one type

• Not very common
• Can be done by using instanceof operator
  – Beware instanceof. This is one of the few places where its use may be appropriate.
  – Generally try to avoid.
• Assignment 7 as an example.