CMSC 132:
Object-Oriented Programming II

Design Patterns II

Department of Computer Science

University of Maryland, College Park

More Design Patterns

- **Marker interface**
  - **Label semantic attributes of a class**
- **Observer**
  - **A way of notifying change to a number of classes**
- **State**
  - **Alter an object's behavior when its state changes**
- **Visitor**
  - **Defines a new operation to a class without changing class**

Marker Interface Pattern

- **Definition**
  - **Label semantic attributes of a class**
- **Where to use & benefits**
  - **Need to indicate attribute(s) of a class**
  - **Allows identification of attributes of objects without assuming they are instances of any particular class**

Marker Interface Pattern

- **Example**
  - **Classes with desirable property GoodProperty**
  - **Original**
    - **Store flag for GoodProperty in each class**
  - **Using pattern**
    - **Label class using GoodProperty interface**

- **Examples from Java**
  - **Cloneable**
  - **Serializable**

Marker Interface Example

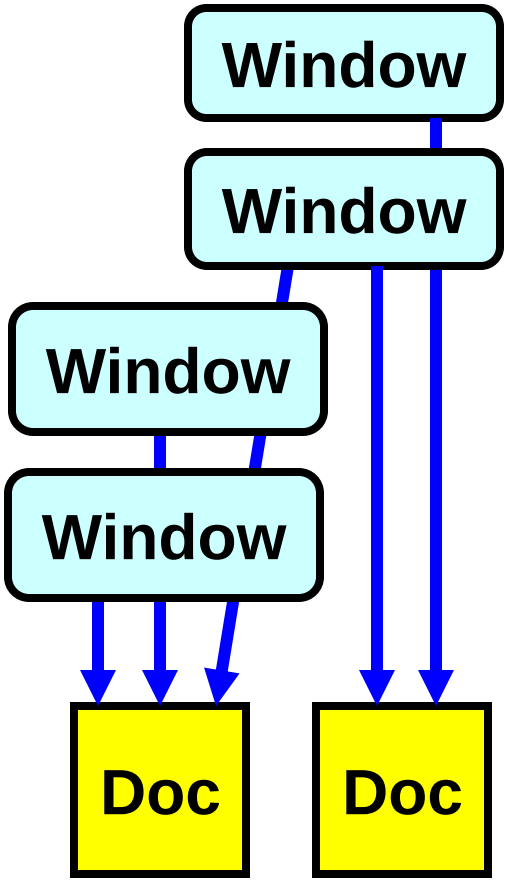
```
public interface SafePet { }   // no methods

class Dog implements SafePet { … }
class Piranha { … }

class d  = new Dog();
class p  = new Piranha();

if (d instanceof SafePet) …   // True
if (p instanceof SafePet) …   // False
```

Observer Pattern

- **Definition**
  - **Updates all dependents of object automatically once object changes state**
- **Where to use & benefits**
  - **One change affects one or many objects**
  - **Many object behavior depends on one object state**
  - **Need broadcast communication**
  - **Maintain consistency between objects**
  - **Observers do not need to constantly check for changes**

Observer Pattern

- **Example**
  - Multiple windows (views) for single document
  - Original
    - Each window checks document
    - Window updates image if document changes
    - Think of window as asking “Are we there yet?”
  - Using pattern
    - Each window registers as observer for document
    - Document notifies all of its observers when it changes



Observer Example

```
public interface Observer {
    public void update(Observable o, Object a)
     // called when observed object o changes
}

public class Observable {
    protected  void setChanged()      // changed
    protected void clearChanged()     // no change
    boolean hasChanged()              // return changed?

    void addObserver(Observer o)      // track observers
    void notifyObservers()            // notify if changed,
    void notifyObservers(Object a)    //  then clear change
}
```

Observer Example

```
public class MyWindow implements Observer {
    public openDoc(Observable doc) {
        doc.addObservers(this); // add window to list
    }
    public void update(Observable doc, Object arg)  {
        redraw(doc);        // display updated document
    }
}
public class MyDoc extends Observable {
    public void edit() {
        …                                  // edit document
        setChanged();                      // mark change
        notifyObservers(arg);              // invokes update()
    }
}
```