Topics for today:

- Pattern Review
  (structural)
  1. Composite
  2. Adapter

  (behavioral)
  3. Observer
  4. Template Method
  5. Strategy

  (creational)
  6. Factory Method

- Pattern Recognition
  (ie. HW4 warm-up)

15-214 Recitation 6

Pattern Practice

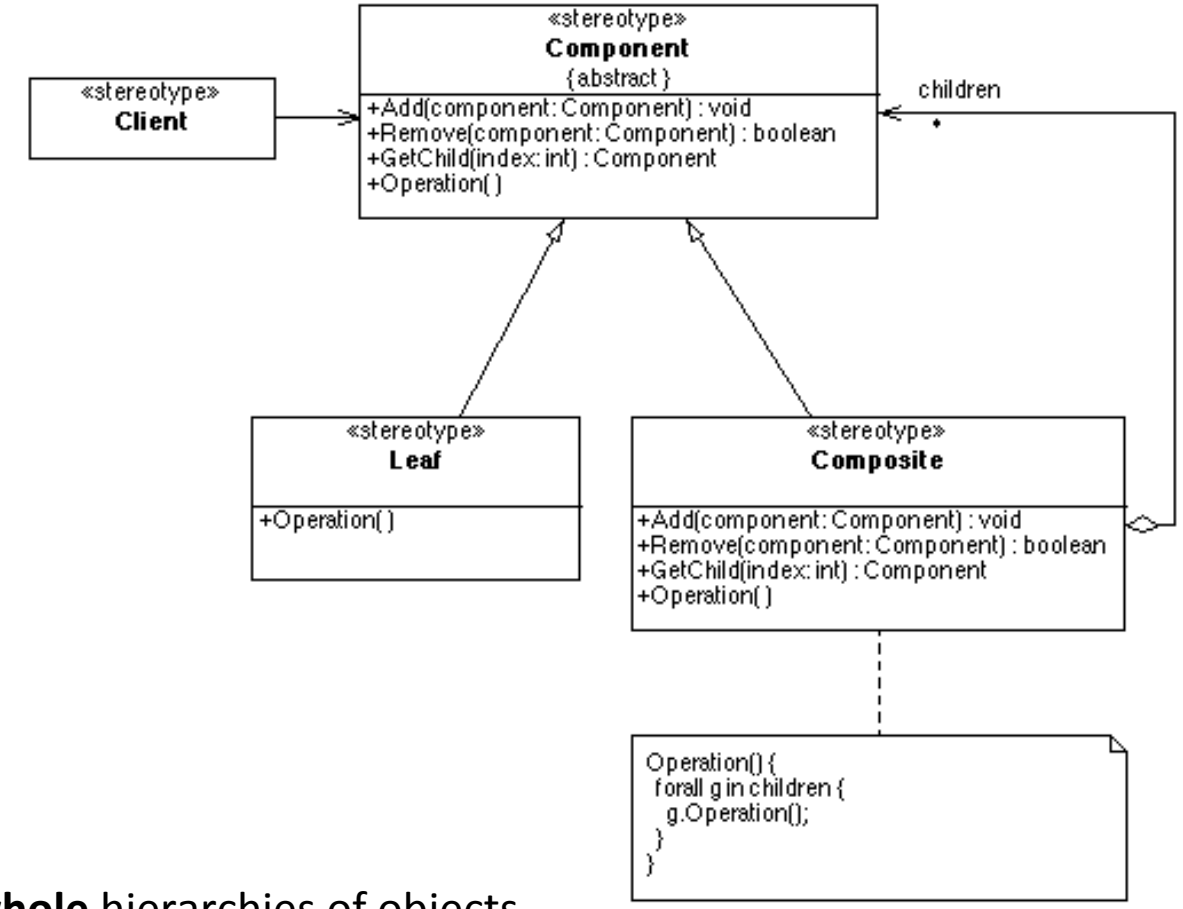
Pick up worksheet

UML from: http://koti.welho.com/pnikande/GoF-models/html/

Part 1

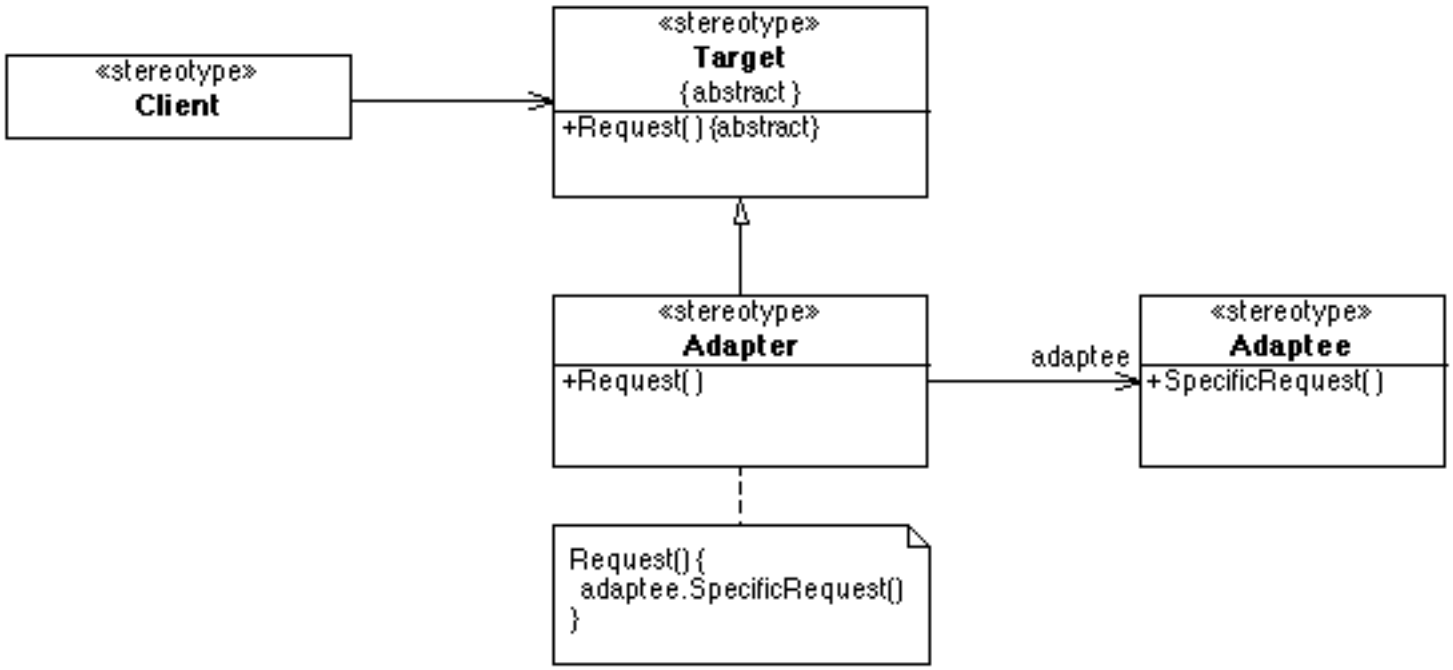
PATTERN REVIEW

Structural - **Composite**



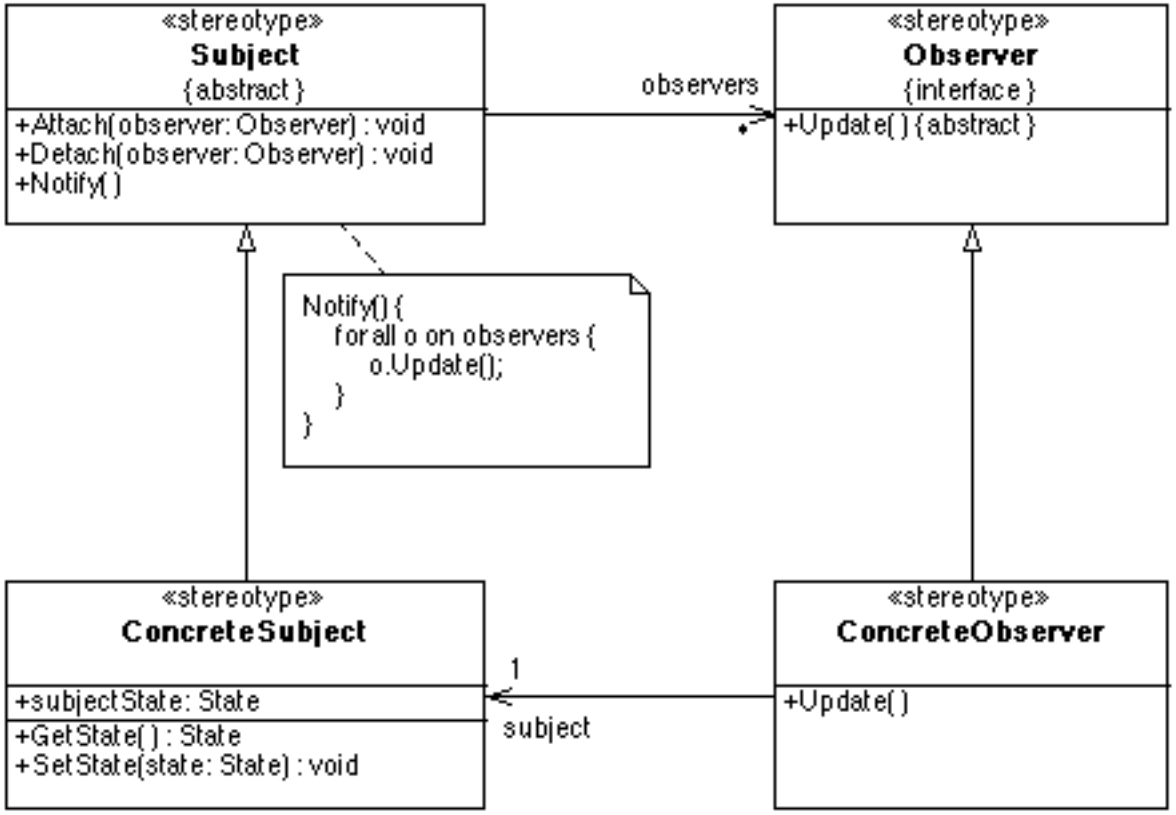
- **part-whole** hierarchies of objects
- *compositions* and *individuals* handled **similarly**

Structural - **Adapter**



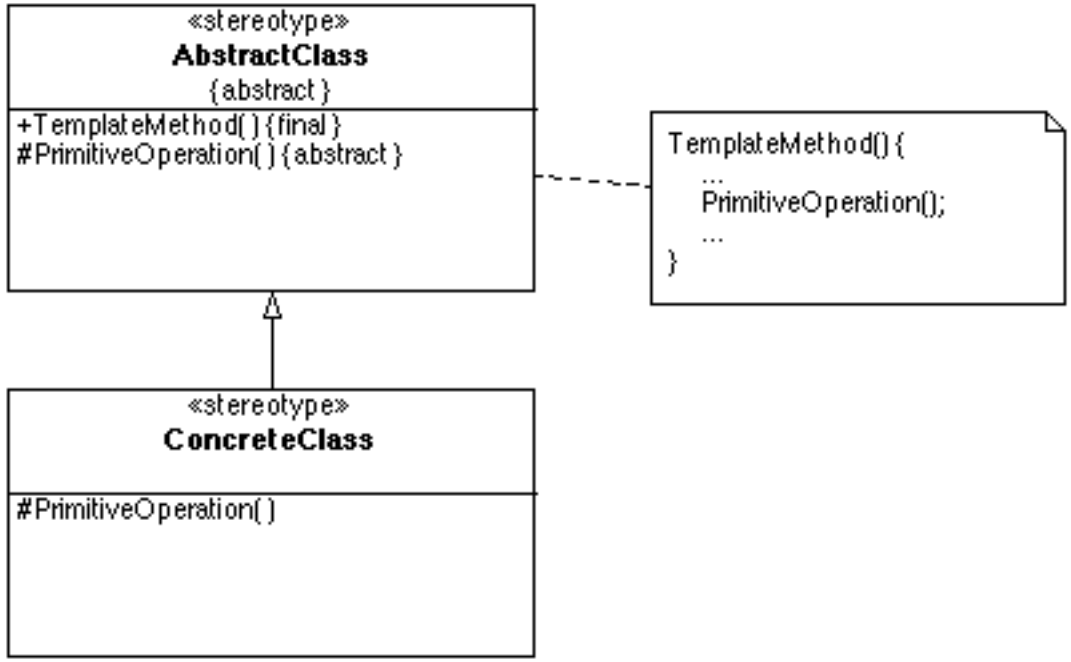
- class with **incompatible interface**
- **Adapt** interface to match your needs

Behavioral - **Observer**



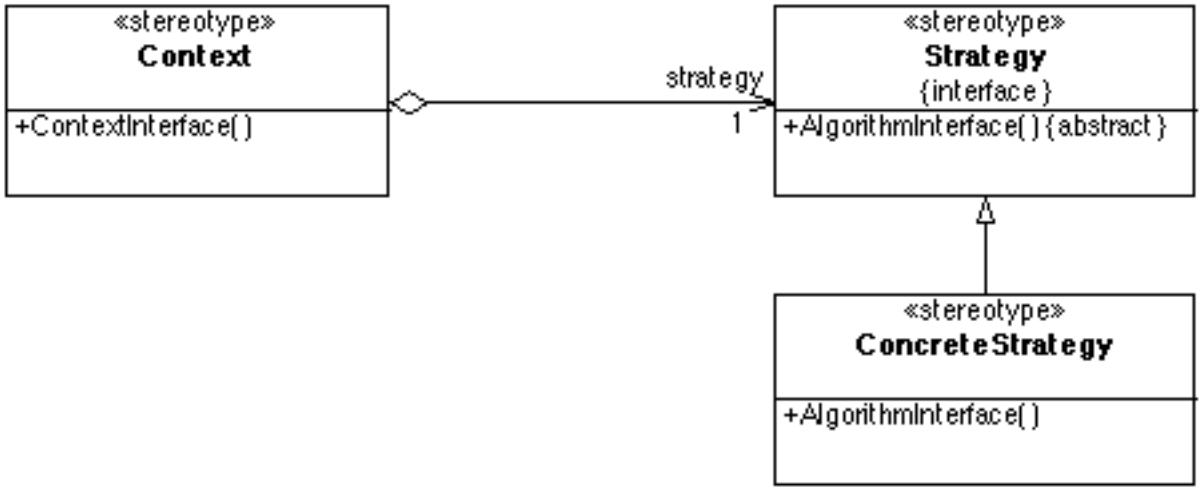
- changing 1 object should **cascade** to changing N others
- **notifies** unknown others of changes

Behavioral - **Template Method**



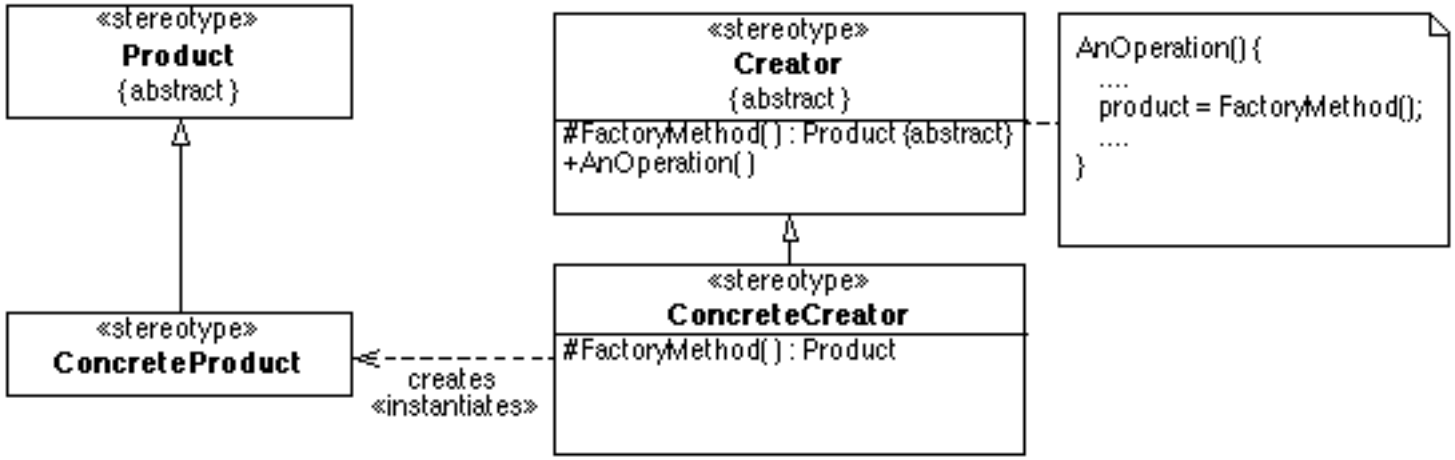
- common subclass **behavior factored and localized**
- subclasses define **specific operations**

Behavioral - **Strategy**



- identical classes **differing only** in algorithm
- different **variants** of an algorithm

Creational - **Factory Method**



- **unknown class** of objects to create
- subclasses specify **created objects**

Part 2

PATTERN RECOGNITION

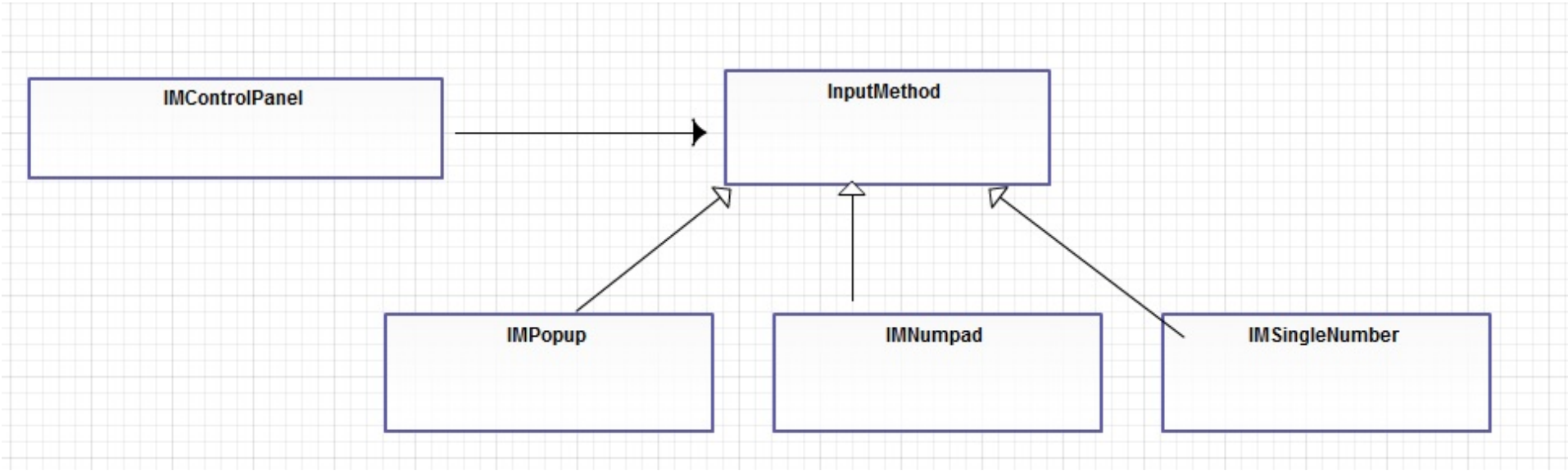
Java SWING

```
typingArea = new JTextField(20);
// all that will listen to key events
typingArea.addKeyListener( autoSaver );
typingArea.addKeyListener( undoHandler );
typingArea.addKeyListener( this );
```

```
// somewhere else
public void keyTyped(KeyEvent e) {
      // handles a KeyEvent (when triggered)
}
```

OpenSudoku

- User chooses **different ways to input numbers** into the cells which is *changeable* at runtime.

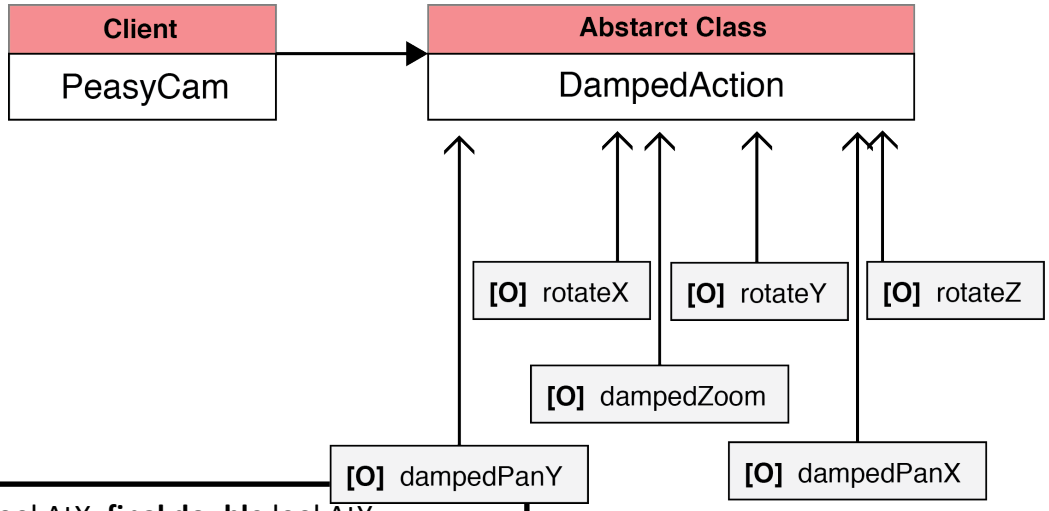


Rabbits & Foxes revisited

```
//in abstract class AbstractImpl

abstract protected Object createOffspring();

abstract protected Command act(World w, Actor a);

public void breed(World w, Direction d) {
        //…
        Object child = createOffspring();
        w.add(child, newLoc);
        //…
}

public void act(World w) {
        //…
        Command c = act(w, this);
        if( c != null ) c.execute(this, w);
        //…
}
```

```
//in class RabbitImpl

protected Object createOffspring() {
        return new RabbitImpl( this.getEnergy()/2 );
}

protected Command act(World w, Actor a) {
                return RabbitAI.getAI().act(w,a);
}
```

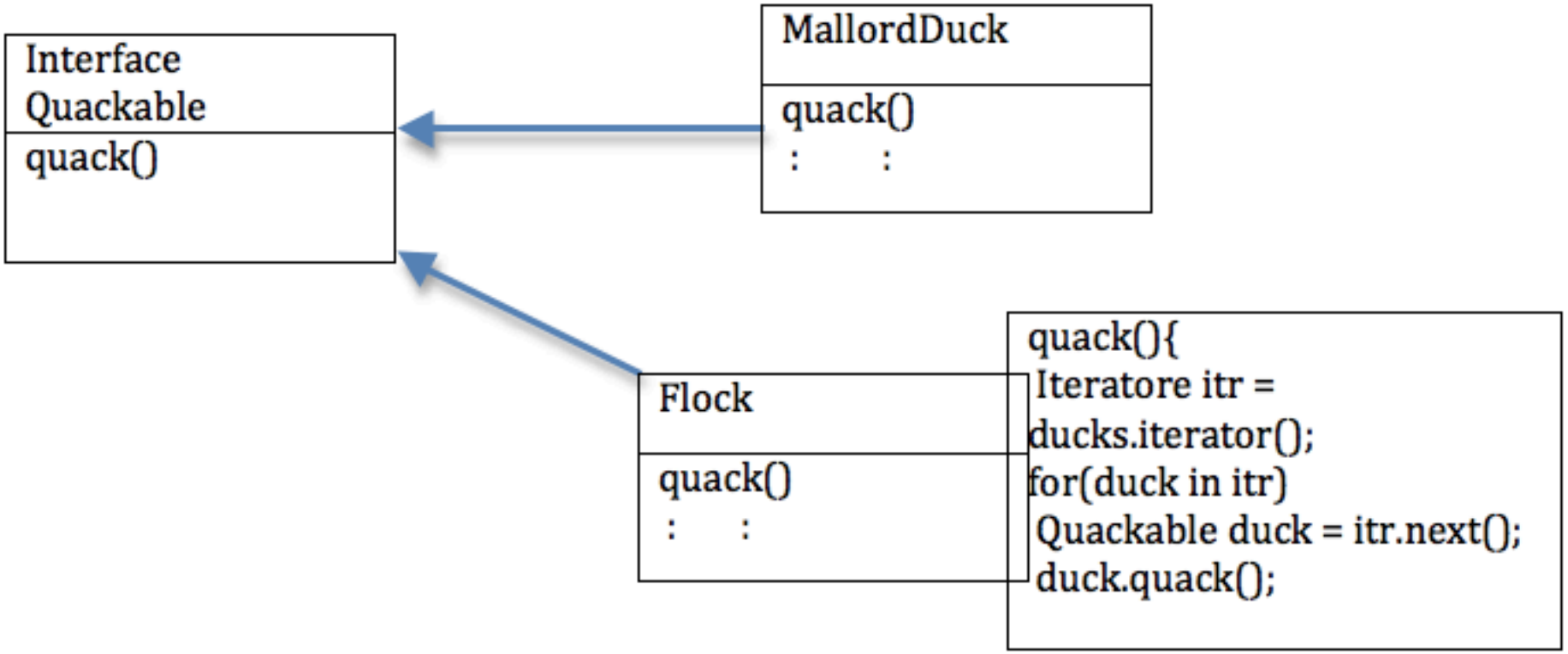
Peasy Cam




```
public PeasyCam(final PApplet parent, final double lookAtX, final double lookAtY,
                final double lookAtZ, final double distance) {

    rotateY = new DampedAction(this) {
        @Override
        protected void behave(final double velocity) {
            rotation = rotation.applyTo(new Rotation(Vector3D.plusJ, velocity));
        } };

    rotateZ = new DampedAction(this) {
        @Override
        protected void behave(final double velocity) {
            rotation = rotation.applyTo(new Rotation(Vector3D.plusK, velocity));
        } };
}
```

Flocks

- Quacking flock of birds



Goose

- Making a goose quack