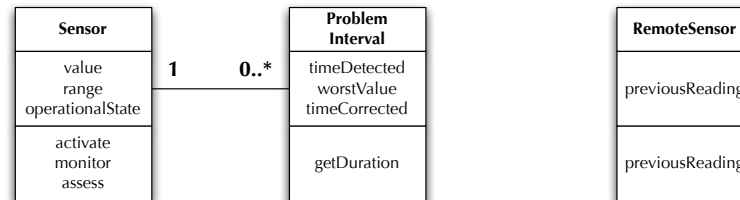


we have sensors
recording problems ...

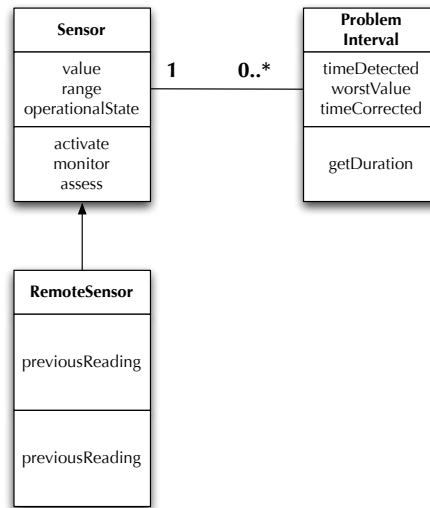
... and wish to add
remote sensors



**add remote sensors
to the design**

There seems to be a relationship...

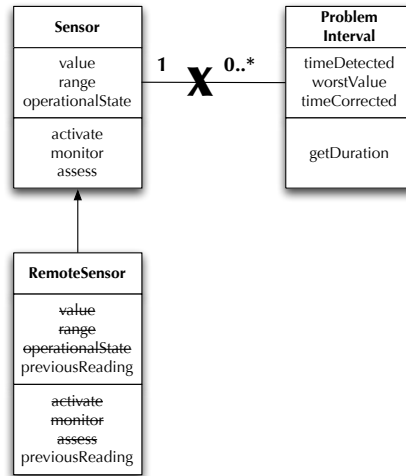
one approach: subclass



A RemoteSensor is a Sensor. But it doesn't have ranges or states that the system can control.

Does a RemoteSensor have ProblemIntervals? In this design it does!

good design?



1. The subclass extends its superclass with new behaviors.

2. The subclass is not merely a utility or helper class.

3. The subclass is a special kind of an object.

4. An object never needs to transmute into an instance of another class.

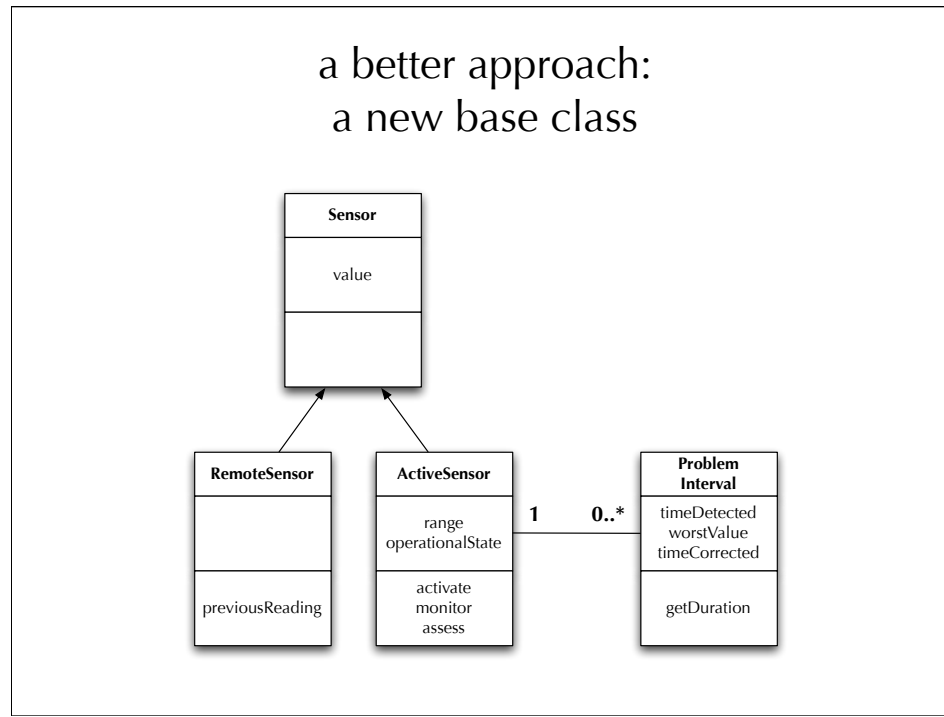
RemoteSensor has an inherited relationship with ProblemInterval that it ignores as well.

Inheritance is useful,
but
we need to redesign
the class structure!

The two kinds of sensor are, well, kinds of sensor. What relationship does that indicate?

A common superclass (or interface).

a better approach:
a new base class



Pull what is common to two kinds of thing into a superclass that represents the common thing.

Inheritance 101.

a common design mod:

extract superclass

You can do this while creating a design from scratch.

You can do this when re-designing a system in response to change. This is a daily practice in the agile approaches to software development. Those folks call it refactoring. We will discuss this practice more next week.

design heuristic

a commonsense rule intended to increase the probability of solving some problem

last time, some heuristics for the use of inheritance

more general...

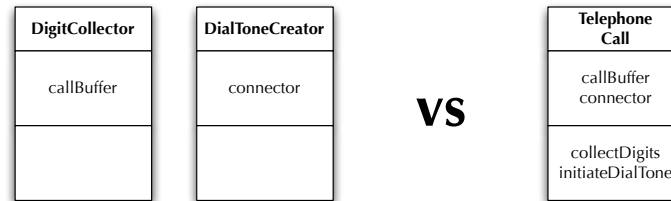
system topology heuristics

a heuristic: distribute intelligence horizontally as uniformly as possible

an example: room temperature monitor

tongue-in-cheek heuristic

beware action as object



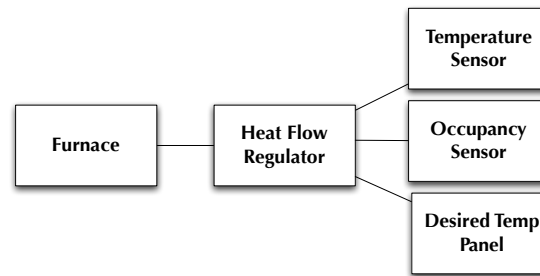
a heuristic: distribute intelligence horizontally as uniformly as possible

an example: room temperature monitor

tongue-in-cheek heuristic

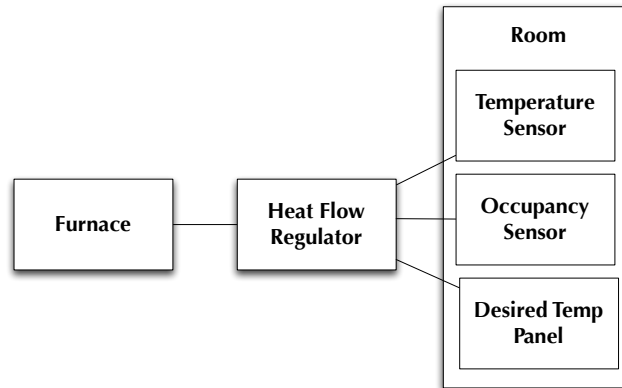
distribute action uniformly

before



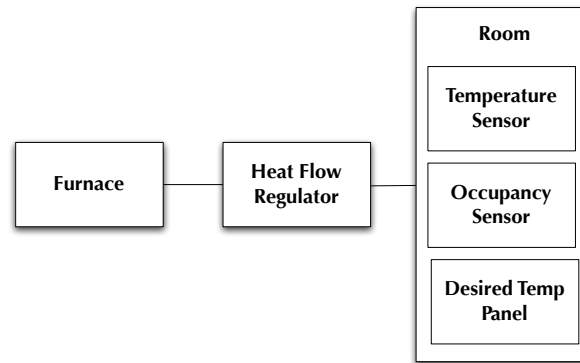
distribute action uniformly

after Step 1



distribute action uniformly

after Step 2



god classes: data or behavior

tongue-in-cheek heuristic: choose n-1

<proxy exercise>

on paper...

dates of interest

10/26/09 → 10/30/09

10/27/09 → 11/03/09

10/29/09 → 11/05/09

*** postponing things by a week ***

Friday: project designs are due (or: iteration 2 is due)

Tuesday: discuss designs in class ... informal presentations

Thursday: midterm exam