Object-Oriented Design

Processes:
- Thermostat - regulates temp
- Volume regulator
- Master control

burner control valve

meter
valve
0-0 Design approach

1. Start with "typical" example
2. Identify methods required for various objects
3. Group into classes based on similar methods
4. Create a hierarchy of classes
   Cons: Push methods as high as possible
   $\Rightarrow$ inheritance (reuse) + polymorphism
5. Look for places to exploit polymorphism
   Can a type be considered a special case of another type w/ benefits?
1. Study objects — identify probable methods
   * thermometer — getTemp()
   + valve adjuster — setValveOpening(v)
   * setpoint dial — getSetting()
   + burner regulator — setFlameHeight(v)
   * float level — getHeight()
   * thermostat — constructor(setpointDial, thermometer, burner reg.)
   * volume reg — ”,getSetting()...
     all ⇒ power(boolean), reset()
   * get readings from these: getValue() “Sensors”
     + things that act on the environment: setting, ”Actuators”
   o things that make decisions in a feedback loop “Control Units”
What might the thermostat code look like?

```java
int burnerSetting = 0;
while (true) {
    if (dialog.getSetting() > thermometer.getTemp())
        burnerSetting = Math.min(100, burnerSetting + 1);
    else
        burnerSetting = Math.max(0, burnerSetting - 1);

    burnerValve.set(burnerSetting);
    delay 1000 ms Thread.sleep(1000);
}
```

setPoint, sensor device, actuator, logic
Relationships:

Control units, sensors, actuators

→ use: sensors & actuators

OJ Meter ─ Gas Meter ─ Sensors

Actuators ─ Control Unit

Thermostat

Actuator / Set Value(v)

Gas Valve ─ Control Unit ─ Thermostat