But first, Traffic Light

■ A traffic light has a green, a yellow, and a red light.
  - Lights can be on and off

■ The traffic light has a normal mode of operation.
  - At first, the green light is on.
  - After 0.5 sec, the green light turns off and the yellow light turns on.
  - After 0.2 sec, the yellow light turns off and the red light turns on.
  - After 0.7 sec, the red light turns off and the green light turns on.

■ A police officer can switch the traffic light into and out of emergency mode.
  - In emergency mode, the yellow light blinks every 0.5 seconds.
  - When exiting emergency mode, the light goes back to its normal operation.

■ The traffic light can be turned off and turned on again.
  - When turned back on, the traffic light resumes the previous mode of operation.
Quality of Design

- What is a good object-oriented design?
- How do I determine if design X is good?
- Are there characteristics?
- Are there metrics?
Analysis of Design

- **Domains**
  - Domains of classes
  - Reusability & Sophistication

- **Encumbrance**
  - What is it? and example
  - It's use, Law of Demeter

- **Class cohesion**
  - Mixed – instance/domain/role
Classes in an HR System

- Employee
- Date / Time
- Salary
- Performance Review
- Job Position
- Job Offer
- Recruit
- Currency
- Bonus
- Location/Office
Classes in an Inventory System

- Equipment
- Bar code
- Loan History
- Date/Time
- Employee
- Location/Office
- Repair Order
- Repair History
- Purchase Order
- Currency
Classes in an Accounting System

- Client
- Account
- Invoice
- Date / Time
- Currency
- Employee
- Bar code
- Delivery
- Pickup
<table>
<thead>
<tr>
<th>Similarities</th>
<th>HR</th>
<th>Inventory</th>
<th>Accounting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Employee</strong></td>
<td><strong>Equipment</strong></td>
<td><strong>Client</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Date / Time</strong></td>
<td><strong>Bar code</strong></td>
<td><strong>Account</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Salary</strong></td>
<td><strong>Loan History</strong></td>
<td><strong>Invoice</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Performance Review</strong></td>
<td><strong>Date/Time</strong></td>
<td><strong>Date / Time</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Job Position</strong></td>
<td><strong>Employee</strong></td>
<td><strong>Currency</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Recruit</strong></td>
<td><strong>Location/Office</strong></td>
<td><strong>Employee</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Currency</strong></td>
<td><strong>Repair Order</strong></td>
<td><strong>Bar code</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Location/Office</strong></td>
<td><strong>Purchase Order</strong></td>
<td><strong>Delivery</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Currency</strong></td>
<td><strong>Pickup</strong></td>
</tr>
</tbody>
</table>
Domains of Classes

- Application Domain
  - classes valuable for one application
    
    Edit Salary, New Loan History, Delete Purchase Order

- Business Domain
  - classes valuable for an industry
    
    Employee, Location

- Architecture Domain
  - classes valuable for an implementation architecture
    
    Currency

- Foundation Domain
  - classes valuable for all businesses and architectures
    
    Date / Time
More Examples

- Application Domain
  - Event recognizer class
- Business Domain
  - Role class, relationship class
- Architecture Domain
  - Human interface class
- Foundation Domain
  - Fundamental class, structural class
Reusability & Sophistication

Application domain

Business domain

Architecture domain

Foundation domain

LOW REUSE

MEDIUM REUSE

HIGH REUSE

HIGH SOPHISTICATION

LOW SOPHISTICATION
Encumbrance

- Quantitative measure of how far a class is from the foundation domain (i.e. it's sophistication)
- Encumbrance: If we take a class c1 and measure the number of classes c1 depends on and measure the number of classes that those classes depend on and so on...
- This may be very large, so we talk about direct and indirect class reference sets
Direct and Indirect

- Direct class reference set refers to the set of classes that a given class c1 directly refers to (via inheritance, association, ...), call these c2, c3, c4, ...
- Indirect class reference set of c1 is the union of its direct class reference set (c2, c3, c4, ...) and the indirect class reference sets of c2, c3, c4, ...
- This leads to direct and indirect encumbrance, which is just the size of the respective class reference set
This is a recursive definition...so when does it stop?

We say that the direct class reference set of classes of the foundation domain is the empty set.
Simple Example