EECE 276
Embedded Systems
Object-oriented design techniques
Object-oriented Programming

- Data abstraction/encapsulation
  » Data can be manipulated only by specific operations

- Inheritance
  » Object classes are organized into an inheritance (generalization/specialization) hierarchy, where lower-level classes inherit properties of higher-level classes

- Polymorphism
  » Operations on objects can use the base class’ interface, the actual implementation is delegated to the specific object used

- Messaging
  » Objects interact through method calls
Good points of OO

- Open-closed principle: classes are closed to modification but open to extension
- Once and only once: any aspect of a software system is captured only once, in one place.
- Dependency inversion: high-level modules should not depend on low-level modules – both should depend on abstractions of those modules.
- Liskov substition principle:
  - If for each object o1 of type S there is an object o2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o1 is substituted for o2, then S is a subtype of T.
  (Subtypes are interchangeable with their base types!)
OO Design Patterns

- GOF: a prototypical solution to a recurring software design problem.
  - “Design Patterns” book by the ‘Gang-of-Four’
- Typically involves a collection of classes with fragments of behavior
- Categories (GOF)
  - Creational: Abstract Factory, …
  - Behavioral: Observer, …
  - Structural: Composite, …
OO Design using UML

- UML: a collection of graphical languages
UML Diagrams for Real-time Systems

- Activity diagrams
  Like flowcharts, but can represent concurrency
- Class diagram
  (seen before)
UML Diagrams for Real-time Systems

- Collaboration diagrams
  Messages passed between classes along associations
UML Diagrams for Real-time Systems

- Component diagrams
  Components and their interfaces and dependencies

![Component Diagram]

BaseDB.cs
(Source)

SiteConfig.cs
(Source)

MyWebComps.dll
(Assembly)
UML Diagrams for Real-time Systems

- Deployment diagrams
  
  How the components will be deployed (on processing nodes)
UML Diagrams for Real-time Systems

- Object diagrams
  Static objects of the system
  Follows class diagram

```
Bird
Name = "Tweety"
WingSpan = 7.25

Rover : Dog
Name = "Rover"
InDogDaysProgram = true

John : Person
Name = "John"
Address = "100 Main St."
City = "Boston"
State = "MA"
ZipCode = "01621"
Phone = "800-800-8000"
```
UML Diagrams for Real-time Systems

- Sequence diagrams
  - Objects, links, messages
  - Message sequences shown in time

*UML does not provide good facilities expressing real-time properties!* (see ROOM, HRT-HOOD)
## SA vs. OO design

<table>
<thead>
<tr>
<th></th>
<th>SA</th>
<th>OOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>System components</td>
<td>Functions</td>
<td>Objects</td>
</tr>
<tr>
<td>Data processors</td>
<td>Separated through internal decompositions</td>
<td>Encapsulated within objects</td>
</tr>
<tr>
<td>Control Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data stores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Characteristics</td>
<td>Hierarchical structure</td>
<td>Inheritance</td>
</tr>
<tr>
<td></td>
<td>Classification of functions</td>
<td>Classification of objects</td>
</tr>
<tr>
<td></td>
<td>Encapsulation of knowledge within functions</td>
<td>Encapsulation of knowledge within objects</td>
</tr>
</tbody>
</table>