Unified Modeling Language (UML), v. 1.3

- Standard notation to
  - specify, visualize and document developing models
  - including their structure and design
- Class Diagram Notation
  - shows classes and
  - relationships between them
Classes

- Attributes are presented as global variables
- Operations are presented as methods
- Accessing attribute values
  - get method to return the value
  - set method to set the value

<table>
<thead>
<tr>
<th>ClassName</th>
</tr>
</thead>
<tbody>
<tr>
<td>-attribute1 : double</td>
</tr>
<tr>
<td>#attribute2 : int</td>
</tr>
<tr>
<td>+method1() : char</td>
</tr>
<tr>
<td>+method2() : &lt;unspecified&gt;</td>
</tr>
</tbody>
</table>

- Visibility
  - `public` (for everybody)
  - `private` (for itself only)
  - `protected` (for subclasses)
  - `package` (inside package)
Abstract Classes

- Are not fully implementation
  - needs to be extended with final implementation
- Notation
  - class name written
    in *Italic*

<table>
<thead>
<tr>
<th>-attribute : int</th>
</tr>
</thead>
<tbody>
<tr>
<td>#AbstractClass()</td>
</tr>
<tr>
<td>#method1() : char</td>
</tr>
<tr>
<td>+method2(in Attribute : int)</td>
</tr>
</tbody>
</table>
Interface

- Same notation as class except:
  - has word “interface”
  - does not have attributes
  - all methods are public

- Additional notation for implementation

Implementation of the interface

Using the interface
Association

- Relationship between the instances of two classes (Class1, Class2)
  - Class1 containing link to Class2
  - Class1 creating an instance of Class2
  - Class1 sending message to Class2
  - Class1 receiving a message containing Class2

```
<table>
<thead>
<tr>
<th>Class</th>
<th>Multiplicity or Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>1 0..1 0..* 0..*</td>
</tr>
<tr>
<td>Address</td>
<td>0..1</td>
</tr>
<tr>
<td></td>
<td>n 1..n 1..*</td>
</tr>
</tbody>
</table>
```
Aggregation and Composition

- Whole-part relationship
  - “whole” class contains “part” class
- Aggregation: instances of both classes can exist independently
- Composition: both instances need each other to exist

![Diagram showing aggregation and composition]
Inheritance

- Is-a relationship (Generalization)
  - Mountain bike is a bicycle
  - Superclass is a general class
  - Subclass is instance of/extends Superclass

- Subclass inherits from Superclass
  - all attributes and operations
  - except private
Useful links

- UML official pages
  http://www.omg.org.uml/

- Specification of UML standard

- Non-commercial drawing program
  http://www.gnome.org/projects/dia/

- Tutorials
  http://www.smartdraw.com/resources/centers.uml.uml.htm
  http://www.auldenfire.com/aitpncc/resources.uml.shtml

- Books
  - B. Oestereich: Developing Software with UML