Representing Designs in UML

• The Unified Modeling Language (UML) is a set of diagram types
• Used to represent software designs
• Most commonly used diagram types are:
  – Structure diagrams
    • A static view of objects and their relationships
    • Inheritance, composition, dependencies
  – Sequence diagrams
    • A view of the dynamic behavior of software
    • Ordering of calls among class instances & their methods
UML and PHP

- UML is designed to represent object-oriented languages
- PHP can be both object-oriented and page-oriented, with procedural code embedded in a page
- Requires some tweaks to UML to represent this correctly
UML Structure Diagrams

- **Classes:**

<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ClassVar1: VarType1</td>
</tr>
<tr>
<td>- ClassVar2: VarType2</td>
</tr>
<tr>
<td>+StaticVar3: VarType3</td>
</tr>
<tr>
<td>+Method1()</td>
</tr>
<tr>
<td>-Method2()</td>
</tr>
<tr>
<td>+Method3(var1: type1, var2: type2) : ret_type</td>
</tr>
</tbody>
</table>

**Visibility** of variables and methods is indicated by putting + (visible) or – (private) before the variable or class name.

**Static** variables are indicated by underlining them.

Can optionally indicate the **signature** (parameters) for methods, and the **return type** of a function.
Representing PHP Pages

• **PHP Page:**

<table>
<thead>
<tr>
<th>[[ Page Name ]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ PageVar1: Type1</td>
</tr>
<tr>
<td>+ PageVar2: Type2</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

| +Function1() |
| +Function2() |
| +Function3(var1: type1, var2: type2) : ret_type |
| ... |

**Basic Idea:** represent a page using the same graphic element as classes (a rectangle)

To indicate this is a page, and not a class, put “[[“ and “]]” around page name (e.g., [[index.php]]).

**Variables** represents those variables defined outside of functions on a page (page global variables).

**Visibility** of variables and functions is always public.

**Functions** represent those functions defined on the page. Classes defined on the page are represented as separate classes in the diagram.
Relationships

• Inheritance
  – A open arrow pointing to parent class
  – PHP pages cannot participate in inheritance relationships

```
ParentClass

ChildClass
```
Relationships (cont’d)

• Association
  – In general, a semantically meaningful link between two classes
  – Represented with a line, with a label on top, with a closed arrow indicating the directionality of the relationship
  – For PHP pages, can use this to indicate “include” relationships among pages
    • Note: use only for PHP pages, not for PHP classes

[[index.php]] \[\[include\]\] [[common-code.php]]

Shows that index.php includes common-code.php.
Relationships (cont’d)

- Aggregation (containment)
  - Represents the case where class instances contain one or more instances of other classes
  - Containment is typically referential, and indicates one instance has pointers to other instances
  - Used for class modeling in PHP
  - A PHP page might contain instances of PHP classes, but would not contain other PHP pages (use association to represent included code)

```
[[index.php]]  ContaineeClass

OR

ContainerClass  ContaineeClass
```