Comp-304 : Class Diagrams
Lecture 9

Alexandre Denault
Original notes by Hans Vangheluwe
Computer Science
McGill University
Fall 2006
Classes vs Objects

- An object is an instance of some class
- Many objects may be instances of the same class
- Classes are static, depict the design and structure at design-time
- Objects are dynamic and are instantiated (from a class) at run-time, they have state
Attributes vs Variables

- Attributes are considered at design-time, are some abstractly defined property.
- Variables are considered at implementation-time, are concretely defined properties.
- An attribute may be known as date at design-time, but at implementation-time, it must be decided if the date variable will be a series of integers or a string.
Objects consist of an object name (underlined with the class name of the class it is an instance of) and its variables (and their values if they have default values).

Object names are written in lowerCamelCase.

Why is there not method names?
Here is a concrete example of an object called o1, an instance of class 2DPoint.

When the object is instantiated, the default value of x and y is 10.

An object with no names is anonymous.
Now, suppose ALL our 2D points must have an (x,y) coordinate such that both x and y are between 0 and 10

This is a constraint.
■ Some attributes are known as derived attributes.
■ They depend on other attributes and are calculated by some formula.
■ Derived attributes are written with a / in front.
■ Derived attributes cannot have set methods.

<table>
<thead>
<tr>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>length:int</td>
</tr>
<tr>
<td>width:int</td>
</tr>
<tr>
<td>/ area:int</td>
</tr>
</tbody>
</table>

```
getL():int {return length}
setL(a:int):void {length = a}
getW():int {return width}
setW(b:int):void {width = b}
getA():int {return area}
```

area = length * width
A set of prefixes for attributes and methods
- + public – visible to any class
- # protected – visible to any subclass
- – private – visible only to class itself
- ~ package – visible to any class within enclosing package

Visibility is a class feature. It is found only in class diagrams.
- We discussed Inheritance extensively the last classes.
- In UML, inheritance is illustrated using a line with a white arrow.
- In this case, Cube is a Rectangle.
In UML, interfaces are often used to represent hardware that needs to interact with this software.

- The ball sign is used to illustrate the require relation.
- The arc sign is used to illustrate the offered service.
In Java, interfaces form a contract between the class and the outside world.

This contract is enforced at build time by the compiler.
- All methods defined by that interface must be implemented by the class.

In UML, these interfaces are described using <<stereotypes>>.

```
<<interface>>
Shape
getWidth(): int
gHeight(): int
```

```
Rectangle
width: int
height: int
Rectangle(width: int, height: int)
gWidth(): int
gHeight(): int
```
Abstract Members

- Abstract methods of written in italic.
- If a class is abstract, it's name is written in italic.
  - Any class with an abstract method is considered abstract.
- Abstract methods cannot be implemented because they are missing part of their implementation.
  - Abstract method are inherited and the missing methods are implemented.

<table>
<thead>
<tr>
<th>Vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td>wheels: Wheel[4]</td>
</tr>
<tr>
<td>body: CarBody</td>
</tr>
<tr>
<td>position: Position</td>
</tr>
</tbody>
</table>

move(float distance): void
turn(float amount): void
getPosition(): Position
As we saw with generacity, Templates are a mechanism to specify the types of objects in a class at declaration time.

In UML, they are defined with a box in the upper left corner of the class.
A package is a collection of classes that, together, perform a certain task.

Classes and objects in package have a prefix:
- `ClassName::PackageName`
- `objectName:ClassName::PackageName`

A package may contain other packages.
Static Members

- Static members of a class (either attributes or methods) exist at the class level.
- They can be used without instantiated an object.
- In UML, these are underlined.

<table>
<thead>
<tr>
<th>VideoUtils</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ getGraphicConfiguration(): Info</td>
</tr>
<tr>
<td>+ getVideoOptions(): Info</td>
</tr>
</tbody>
</table>