Lecture 12: Advanced Class Diagrams

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2001

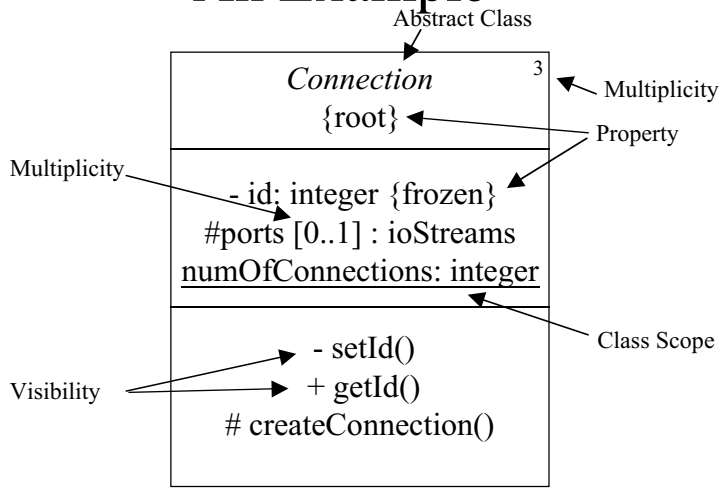
Goals for this Lecture

- Examine Advanced UML Notations
  - for Classes
  - and Associations
- We'll cover interfaces and object diagrams in the next lecture

Advanced UML Class Notations

- UML supports a number of advanced modeling features for classes
  - Class and Attribute Properties
  - Class and Attribute Multiplicity
  - Class-Scope Attributes and Operations
  - Visibility

An Example



Visibility

- The visibility of an attribute or operation specifies whether it can be used by other classes; the default visibility is public
- Three types
  - public (+)
    - Any outside class can access the feature (as long as it has a reference to the class)
  - protected (#)
    - Any descendant of the class can use the feature
  - private (-)
    - Only the host class can access the feature

Scope

- A feature (attribute or operation) can be assigned a scope
  - instance: each instance of a class has its own state for the feature
  - classifier (or class): There is only one value for this feature across all classes
    - numberOfConnections in the previous example
- Classifier scope is indicated by underlining the feature definition

Properties

- A class can be assigned two properties
  - root - the class can have no parents
  - leaf - the class can have no children
- A property is indicated by placing it below the class in brackets, e.g. {leaf}
  - attributes and operations can have properties too (covered later in this lecture)
- A class can also be abstract; which means that no instances can be created of this class
  - This is indicated by placing the class name in italics
  - This is used when the root class is meant to serve as a template for creating various subclasses

Multiplicity

- In the previous lecture, we saw multiplicity used for associations
- On classes, multiplicity constrains the number of instances that can be created for a class
  - The multiplicity for classes is indicated in the top, right corner of the class
- On attributes, it constrains the number of values an attribute can have
  - this lets you specify attributes that can be modeled as arrays: ports[2..*] : Port

Complete Attribute Syntax

- The complete syntax for attributes is

```
[visibility] name [multiplicity] [: type]
  [= initial-value] [{property}]
```

- Example

  + ports [2..*] : Port = null {addOnly}

  id : integer = 0

- Attribute Property Values
  - *changeable*: default, freely modifiable
  - *addOnly*:may add new values;no changes allowed
  - *frozen*: the value may not change after the object is initialized

Complete Operation Syntax

- The complete syntax for operations is

```
[visibility] name [(parameter-list)] [:
  return-type] [{property}]
```

- The complete syntax for a parameter is

```
[direction] name : type [= default-value]
```

- Examples

  + set(n : Name, s : String) {sequential}

  - setId(inout id : integer)

Additional Operation Info

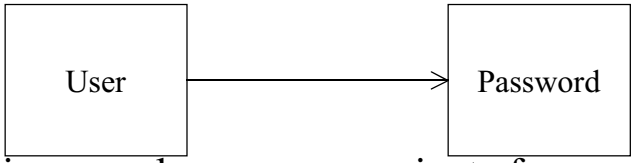
- Possible Direction Values
  - in : An input parameter; may not be modified
  - out : An output parameter; may be modified to communicate with caller
  - inout: An input parameter; may be modified
- Possible Operation Properties
  - isQuery: Does not change state of system
  - sequential: does not protect against multiple threads
  - guarded: does protect against multiple threads
  - concurrent: multiple threads can execute it at the same time

Associations

- Advanced adornments for associations include
  - navigation
  - visibility
  - qualification
  - interface specification {next lecture}
- In addition, we will introduce the notions of
  - association classes
  - association constraints
  - interface realization {next lecture}

Association Navigation

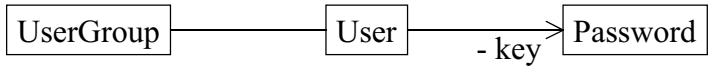
- A direction can be added to an association



- in this example, you can navigate from objects of type User to objects of type Password but not the other way around

Association Visibility

- Visibility can be assigned to an association role
  - public: objects outside the association can navigate the association
  - protected: only an object and its children can access a protected association
  - private: only the objects that participate in the association can navigate it

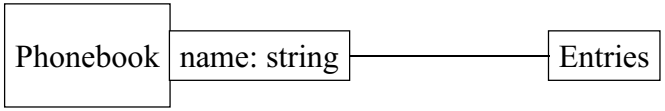


Association Qualification

- Associations sometimes model relationships that involve "lookup"
  - That is, when navigating the relationship, you are looking for a particular object (or set of objects)
- Example
  - A phonebook consists of multiple entries
  - Given a name, we want to look up the associated phone number

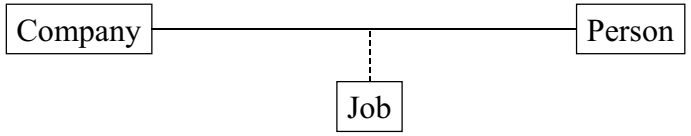
Association Qualification, cont.

- UML can model such a situation using an association qualification
  - the qualification is drawn as a rectangle extending out of its associated class
  - the rectangle contains the attributes used to perform the "look up"



Association Classes

- There are times when it becomes necessary to associate data with an association
  - Employment: should the details of a job be associated with a company object or a person object?



Association Constraints

- UML provides five pre-defined association constraints
  - implicit: the relationship is conceptual
  - ordered: the set of objects at one end of the association are in an explicit order
  - changeable: links between objects can be modified freely
  - addOnly: new links may be added only
  - frozen: a link, once added, cannot be modified
- Constraints are drawn in braces: {frozen}