Lecture 13: Interfaces and Objects

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

Goals for this Lecture

- Introduce
  - Interfaces
  - Objects
- Examine their associated UML Notations

Interfaces

- An interface is a collection of operations (not data) that specifies a particular service of a class or a component
  - For instance, lists, queues, stacks, and trees typically provide an Iterator interface that allows other classes to cycle through their elements

UML Notation

- The most simple notation for an interface is a labeled circle

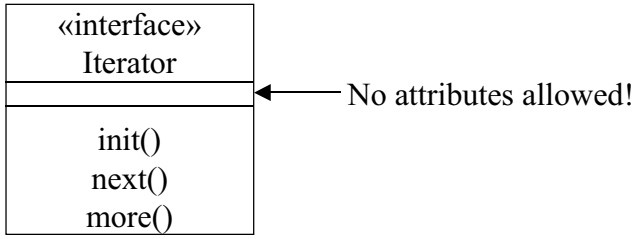
Iterator

Interface names can be grouped using packages

Java::Collection::Iterator

UML Notation

- However, a full class diagram can be used to specify the particular operations associated with an interface

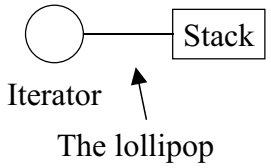


How interfaces are used

- You cannot instantiate an instance of an interface, instead other classes (and thus their objects) choose to implement certain interfaces
  - An interface can act as a type, so you can declare variables that have, for instance, the Iterator type
  - This allows you to point at a class who implements the Iterator interface without knowing (or caring) about what its actual type is

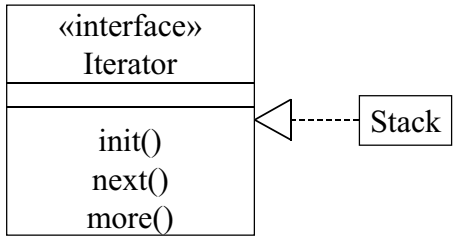
UML Notation

- To indicate that a class implements a particular interface, use the "lollipop" notation
- This is also called "realization"



UML Notation, continued

- When drawing an interface using a class diagram, realization is shown using the following notation



The fact that realization has two notations is, in my opinion, unfortunate.

Roles

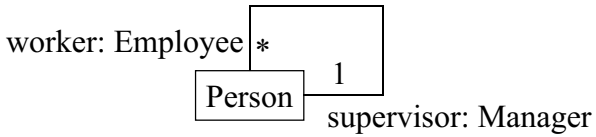
- A class can implement more than one interface
  - each interface represents a role that a class can play
  - we saw how roles can be specified for associations back in lecture 11

Returning to Lecture 12

- In lecture 12, we deferred two advanced association notations
  - interface specifiers
  - interface realization
- We have already covered interface realization

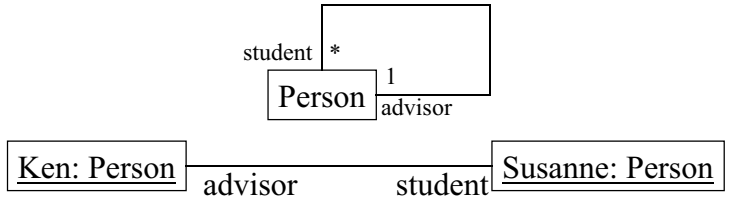
Interface Specifiers

- In an association, a role name can specify the specific interface that it is presenting to the class on the other side of the association



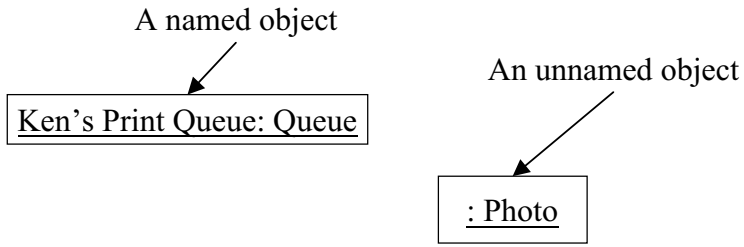
Links

- An association specifies a relationship between two classes
  - A link is an instance of an association



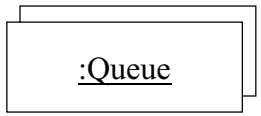
Objects

- Objects are instances of classes
  - an object can be named or unnamed



Multiobjects

- If you need to model a collection of anonymous objects (such as a stack or queue), you can use the multiobject notation



This represents the collection object and all of its individual instances

Orphan Instances

- In some situations, you may need to model an object whose type is unknown
  - This can occur in practice when dynamically loading an object into memory
  - Use the orphan notation to indicate such an object

If you later discover the type of an orphan instance, you can transform it to a named instance using the «become» stereotype (not yet covered)

Active Objects

- Finally, you can indicate that an object has its own flow of control (e.g. it's a Thread object) using the following notation