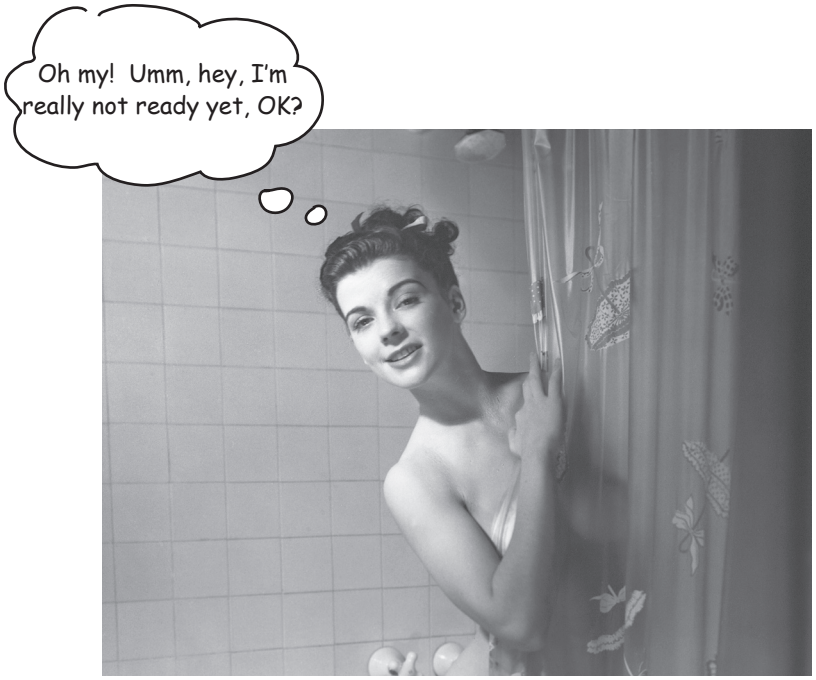
UML for the Java Associate



UML is Cool. Rumor has it there are some **UML** questions on Sun's new Java Associates exam. Sadly, the new *Head First* book on OO isn't quite ready yet (be sure to check hfoobook.com toward the end of 2005!), but we've taken a few random pages (and parts of pages) from it and put them together here. So, if you don't mind the lack of organization, you should find everything you need to know about UML (for the exam) in the following pages. Oh, and did we mention that the author of this book wrote most of the UML questions for the exam? :-)

What's with all the rectangles?

NOTE: The "facing page" is not included in this little UML booklet!

The four boxes full of text over there on the facing page are our first examples of **UML**. UML stands for Unified Modeling Language, and it's the most common notation used for creating diagrams that describe object-oriented systems.

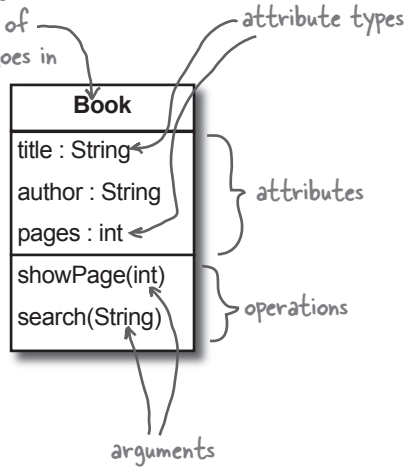
It's beyond the scope of this book to teach you everything there is to know about UML. However, we'll be sprinkling in a lot of the really critical stuff throughout the course of the book. If you're going to be moving to Objectopia, you should definitely have a reasonable understanding of UML.

In this, our first look at UML, we'll look at the notation used to represent classes (probably the single most commonly used aspect of UML) as well as the notation used to represent individual objects.

UML 101

Classes

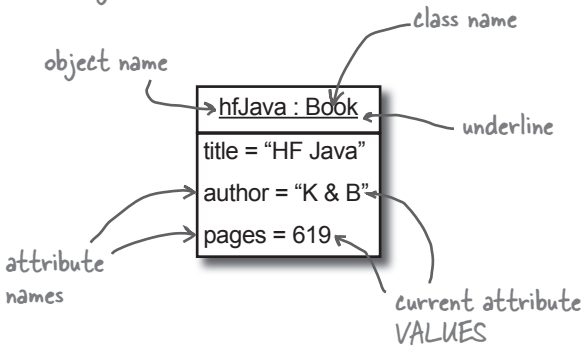
A class is drawn as a rectangle with three sections. The name of the class (in bold) goes in the first section.



The 2nd section contains the attributes and the last contains the operations.

Objects

It's less common to draw object diagrams than it is to draw class diagrams. But they can be useful for describing specific examples or complex relationships between objects.



Notice that we don't bother listing the operations when we're drawing a single object instance. All objects of a given type (class) have the same set of operations, so it would be redundant to list them here.

Associations

Here's another little UML 101 section taken from a Chapter 1 page of HFOO, plus a few extra notes (since Chapter 9 hasn't been written yet)...

Associations are drawn in UML by connecting two classes with a line. Depending on the type of relation, there may be a diamond at one end of the line. In this example, the diamond is filled in, which means we're showing a special type of association called "composition." Don't worry, we'll have much more to say about the different types of associations in Chapter 8!

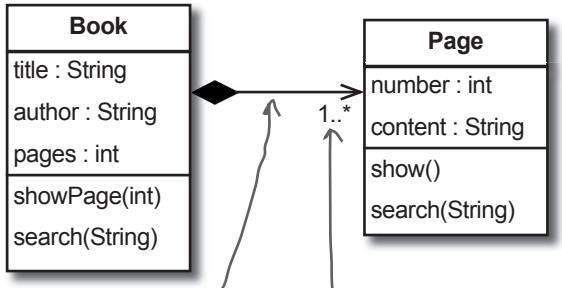
The UML 101 lesson on the right mentions Chapter 9. Well, the truth is, Chapter 9's not written yet! But there are a couple of key things you need to understand if you're taking the Java Associates exam...

Composition vs. Association

The little diamond indicates a "composition" relationship. The easiest way to think about a composition relationship is as a "part-of" relationship: a Page is a part of a Book. If you look at the Book/Page example on the right (which uses composition), the key point here is that if you get rid of a Book, all of its Pages go away too! The Book is composed of Pages.

Navigation

Another key thing to understand about UML relationship notation is navigation. If a line between two classes has an arrow at one end of it (not a triangle, just an arrow like the one on the right), this is telling you that the relationship can only be traversed in one direction. In this example, we're saying that a Book knows about its Pages, but a Page doesn't know what Book it's in.



Read this as "a Book is composed of one or more Pages."

This little "1..*" is a multiplicity indicator. It's shorthand for "one or more." You can also use:

* = zero or more

1 = one (duh!)

0..1 = zero or one (optional)

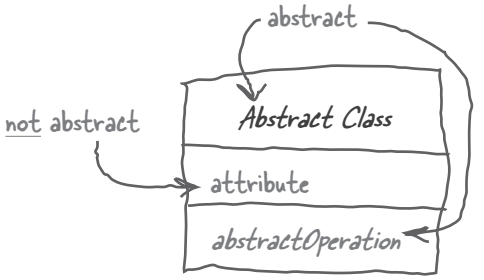
2..7 = between two & seven

More UML 101 Goodies...

Here are a few more random bits of UML goodness from the pages of HFOO...

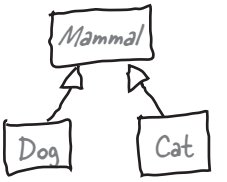
UML 101

Anything in a UML diagram that's abstract is written using *italics*. In our chess example, we've seen both abstract classes (Piece, Player) and abstract operations (canMoveTo).

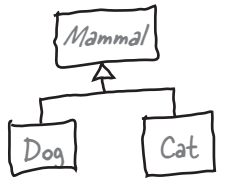


UML 101

Generalization is shown in UML using an open triangle that points from the more specific class to the more general one.



If you prefer, it's also valid UML to use a single triangle with multiple lines connected to it, like this:



UML 101

Operations are shown in the bottom section of the class box. The full, precise specification of what you can include when describing a method is beyond the scope of this book. For our purposes, we'll use the syntax:

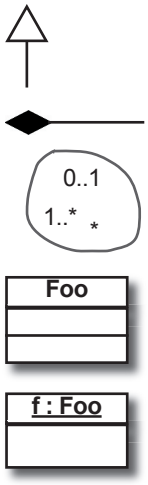
name(parameter types) : return type

So, if you had an operation that added two numbers together, it might look like this:

add(int, int) : int

UML Match-Up

Draw lines to connect the UML notation with its meaning.



Class

Generalization

Object

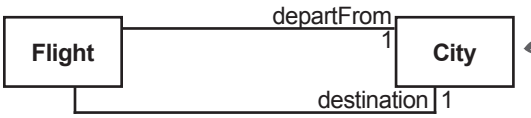
Association (Composition)

Multiplicities

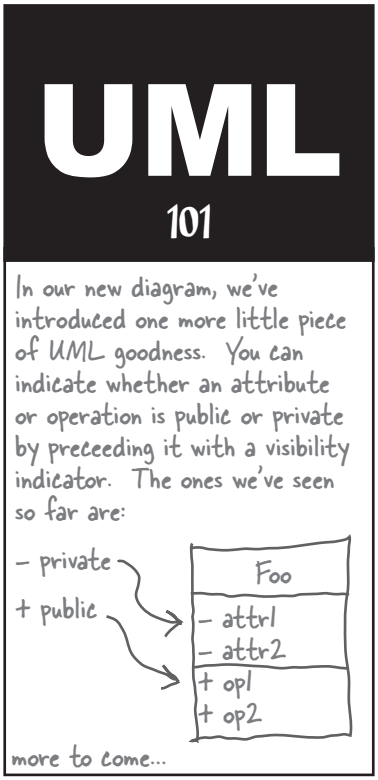
Roles Tell Us The Job of Each Class in an Association

In addition to multiplicities, you can add "roles" that serve to name associations in cases where it may not be clear what role a class is playing in a particular association. Adding roles to every association can be tedious. You should add them only when they help to clarify the diagram.

Role names are written in the same general area as multiplicities. Be sure to put the role name next to the class that it represents. For example:



Here, we have 2 associations between the same two classes. Without the roles, it would be hard to know what the associations mean. With the roles, it's clear that one assoication represents the departure city and the other represents the destination.



World's Easiest UML Quiz

Match the access modifier with its UML notation.

public	-
private	+

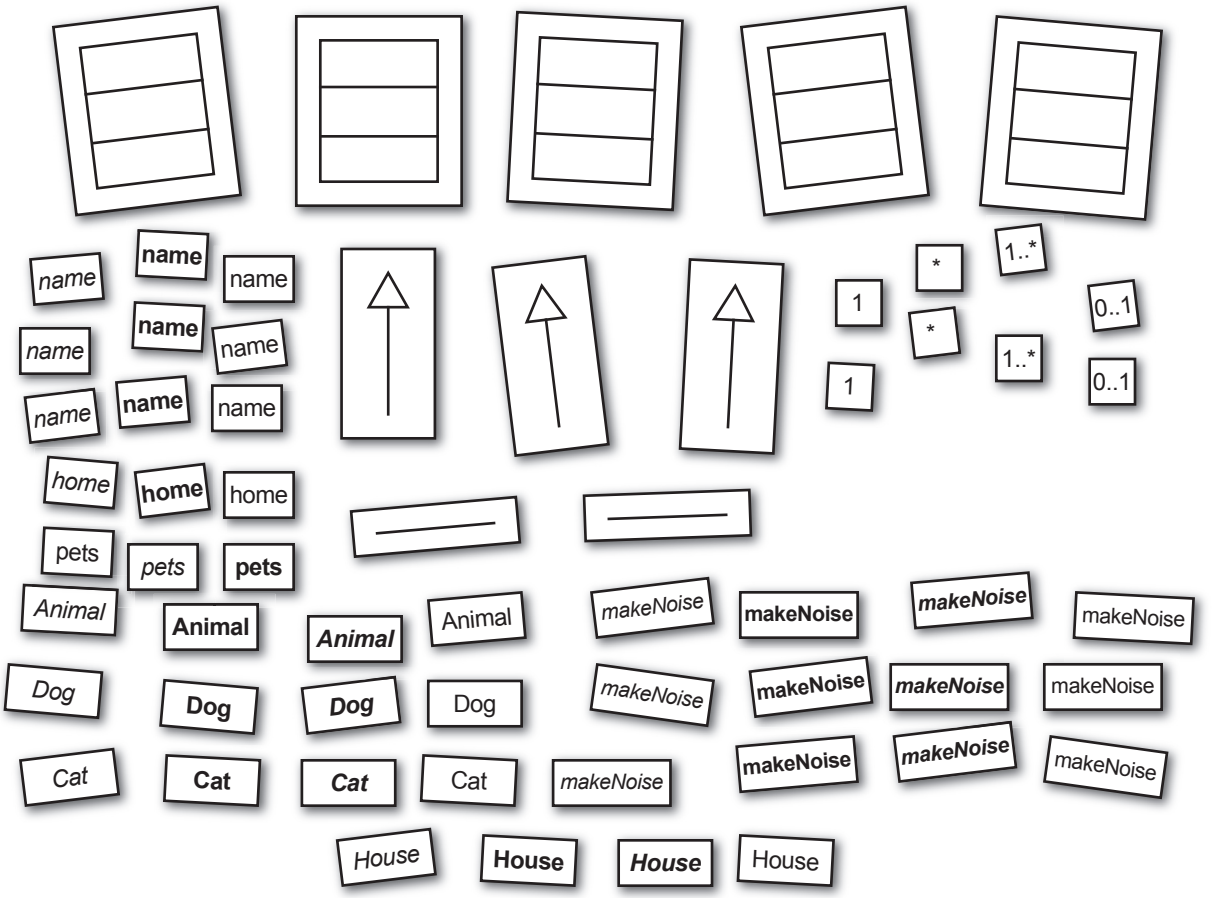
Exercise

Piece together the UML magnets (you won't need all of them) shown below to create a class diagram that takes into account all of the following information:

- ✦ A house may have any number of animals (pets) living it.
- ✦ The two possibile types of animals that can live in a house are dogs and cats.
- ✦ Each dog or cat has a name.
- ✦ An animal's house is its one and only home.
- ✦ You can tell an animal to make noise and it will do its thing.

UML 101

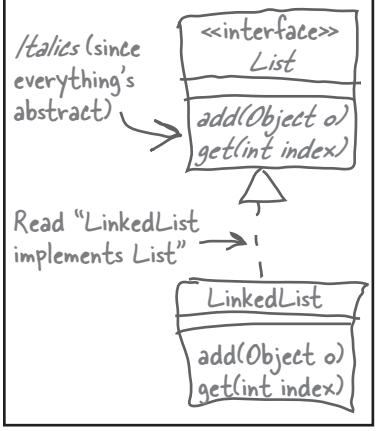
Quiz Time!



UML 101

Interfaces are drawn just like classes. The only difference is the addition of the word "interface" at the top (surrounded by those little guillemet characters: << >>).

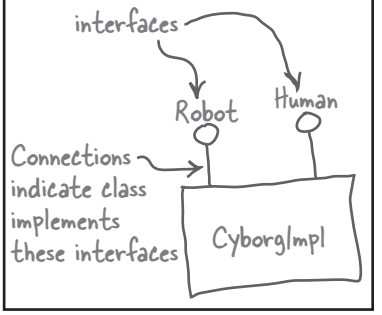
A class that implements an interface is connected to that interface via a dashed line with a triangle pointing to the interface



UML 101

When you're drawing class diagrams with classes that implement lots of interfaces, drawing every interface implemented by every class can become unmanageable.

UML provides an alternate notation for indicating interface implementation. It's sometimes referred to as the "lollipop" notation because the interface is represented by a small circle attached to the class by a solid line. Note that with this notation, you do not list the operations defined by the interface.



Exercises

Use the UML magnets below to construct a class diagram consistent with the following declarations:

(you won't use all of the magnets)

```
interface Atom extends Brain

class Comet extends Diode implements Brain, Entropy
```

these ones are in italics, in case you couldn't tell!

Comet *Brain* *Atom* *Diode* *Entropy*

Comet Brain Atom Diode Entropy

You can use as many of these ones as you need...

«interface»

Exercises

Draw the class diagram from the previous exercise using the lollipop notation.

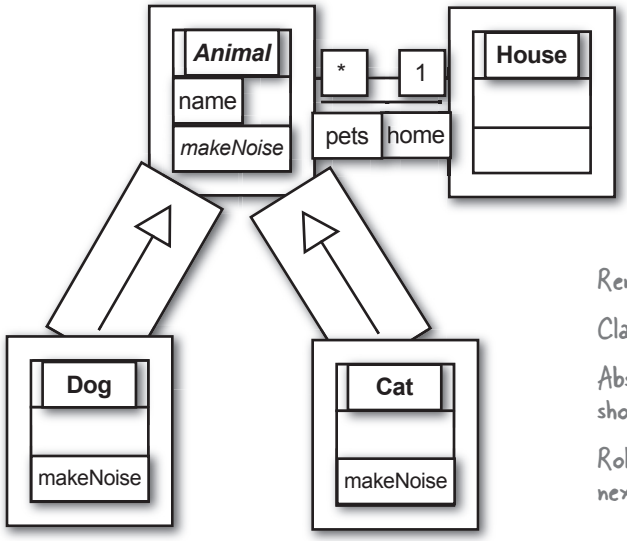
Exercise Solution

Piece together the UML magnets (you won't need all of them) shown below to create a class diagram that takes into account all of the following information:

- A house may have any number of animals (pets) living it.
- The two possibile types of animals that can live in a house are dogs and cats.
- Each dog or cat has a name.
- An animal's house is its one and only home.
- You can tell an animal to make noise and it will do its thing.

UML 101

Quiz Time!



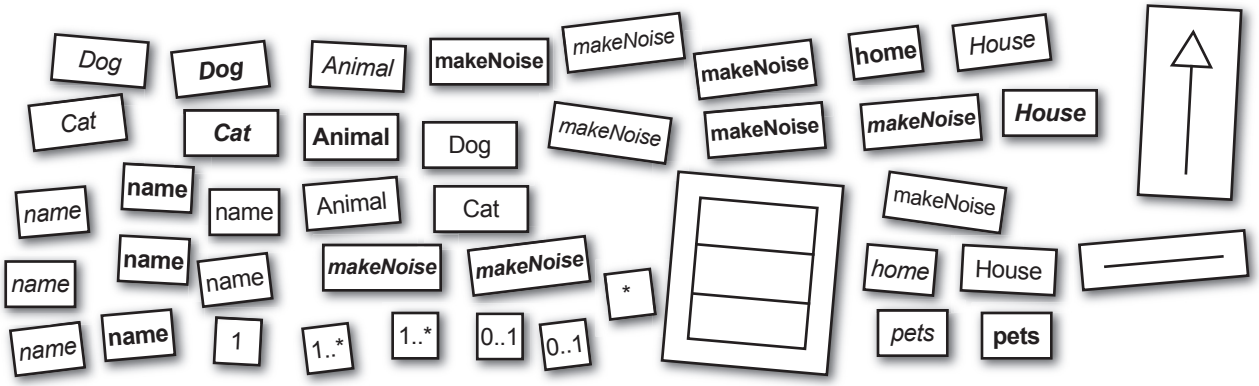
Remember:

Class names should be bold.

Abstract class and operation names should be in italics.

Role names and multiplicities go next to the things they describe.

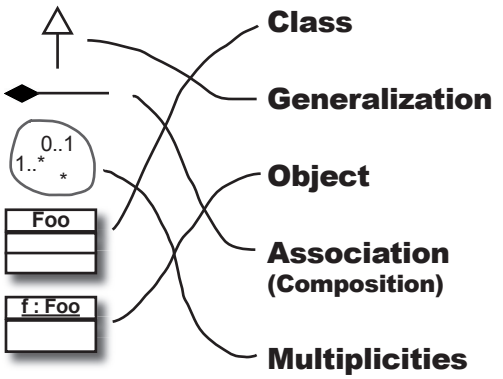
the leftovers...



Exercises

THE ANSWERS

UML Match-Up

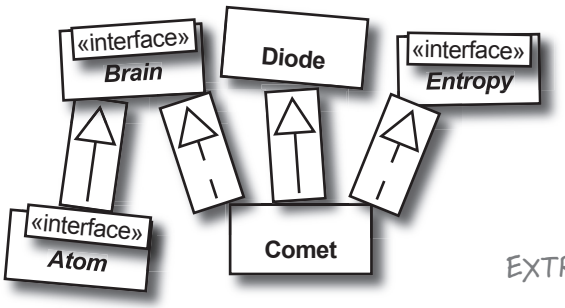


Use the UML magnets below to construct a class diagram consistent with the following declarations:

(you won't use all of the magnets)

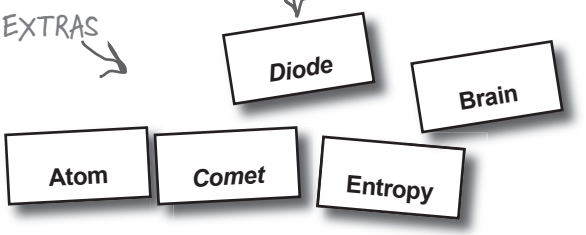
```
interface Atom extends Brain

class Comet extends Diode implements Brain, Entropy
```

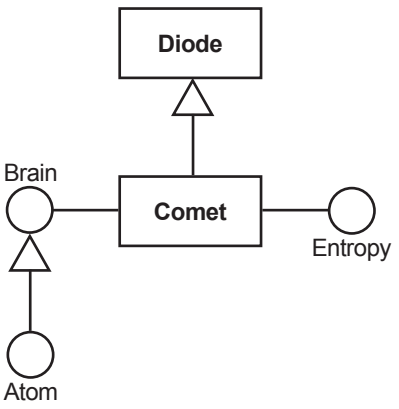


You could also have used this one (the abstract class) for Diode since we can't tell from the given information whether or not it's abstract.

EXTRAS



Draw the class diagram from the previous exercise using the lollipop notation.



THE END

That's all for now. These somewhat randomly thrown-together pages include all the UML you'll need to know to answer the UML questions on Sun's Java Associate exam. Good luck!

Don't forgot to come visit us here...

www.hfoobook.com