APPENDIX C

UML Reference

UML Reference

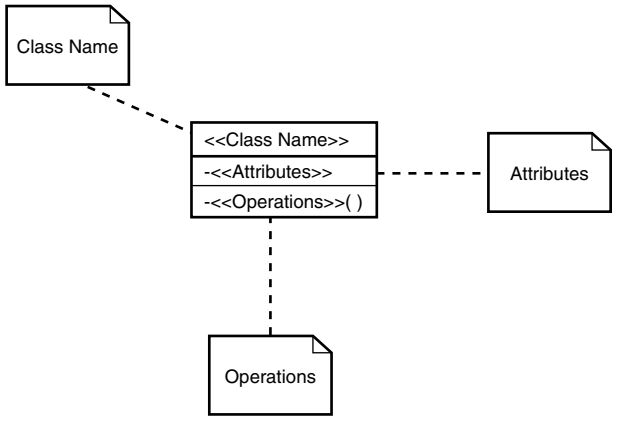
This appendix provides a quick reference to the UML notation used throughout the book.

Classes

The UML represents a class through a box divided into three sections. The top section contains the name, the middle section contains the attributes, and the bottom section contains the methods.

Figure C.1 illustrates how the UML represents a class.

FIGURE C.1
A UML class.



Object

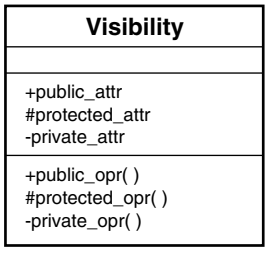
The UML represents objects the same as a class. The only difference is that the object's name is underlined and the methods are omitted. The attributes may also show value.

Visibility

Figure C.2 illustrates how the UML represents attribute and method visibility:

- + represents public visibility
- # represents protected visibility
- - represents private visibility

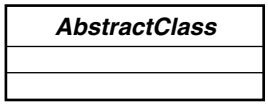
FIGURE C.2
Method and attribute visibility.



Abstract Classes and Methods

Abstract classes and methods are symbolized by italicizing the abstract name. Figure C.3 provides an example of an abstract class.

FIGURE C.3
An abstract class.



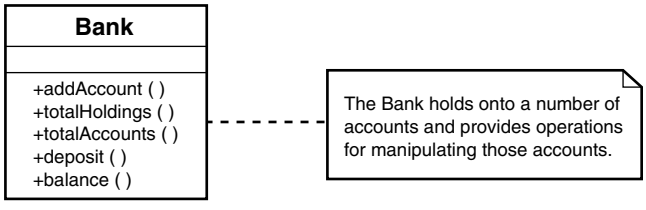
You can also add the `{abstract}` constraint after the name. Using the constraint label helps when drawing a diagram by hand.

Notes

Sometimes a note will make a model more understandable. In the UML, a note resembles a sticky note and is attached to the element being noted with a dashed line.

Figure C.4 illustrates the UML note. You can attach a note to any part of your UML model.

FIGURE C.4
The UML note.

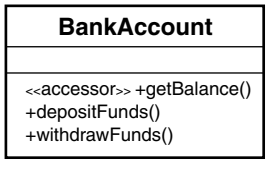


Stereotypes

A *stereotype* is a UML element that allows you to extend the vocabulary of the UML language itself or to classify a marking. A stereotype consists of a word or phrase enclosed in guillemets (<< >>). You place a stereotype above or to the side of an existing element.

Figure C.5 illustrates a stereotype that defines a type of method.

FIGURE C.5
The UML stereotype.



Relationships

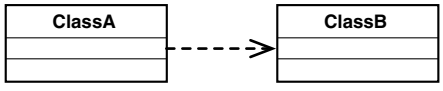
A *relationship* describes how classes interact with one another. In the UML, a relationship is a connection between two or more notational elements. In the UML, a relationship is normally illustrated through a line or an arrow between classes.

Dependency

In a *dependency* relationship, one object is dependent on another object's specification. If the specification changes you will need to update the dependent object.

In the UML you represent a dependency as a dashed arrow between the dependent classes. Figure C.6 illustrates the UML dependency notation. The relationship tells you that `ClassA` depends upon `ClassB`.

FIGURE C.6
A simple dependency relationship.



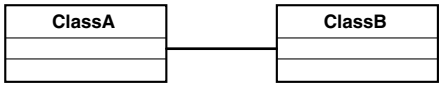
Association

An *association* indicates that one object contains another object. In the UML terms, when in an association relationship one object is connected to another.

In the UML you represent an association as a line that connects the two classes. An association that has no arrow is said to be bidirectional. An arrow signifies that the relationship works only in one way.

Figure C.7 illustrates the UML association notation. The relationship tells you that `ClassA` is associated with `ClassB`.

FIGURE C.7
An association relationship.

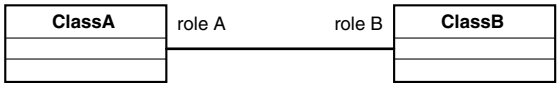


Roles

The UML allows you to denote each class's role in the association. The association *role* is the part that an object plays in a relationship.

Figure C.8 illustrates the UML notation for the role.

FIGURE C.8
The UML role.



Multiplicity

The UML allows you to denote the multiplicity of the association. The *multiplicity* indicates how many objects may take part in the instance of an association.

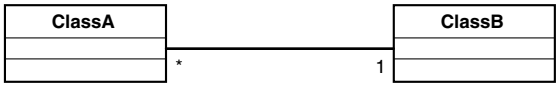
The range of multiplicity values is listed in Table C.1.

TABLE C.1 Multiplicity Values

Notation	*Value*
1	One
*	Any number
1..*	At least one
x..y	Any number of values in the range *x* to *y*

Figure C.9 illustrates the UML notation for multiplicity.

FIGURE C.9
Multiplicity.

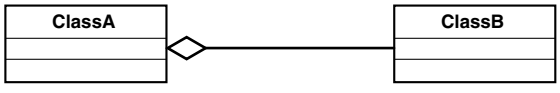


Aggregation

The UML provides notation for aggregation. An *aggregation* is a special type of association that models has-a of whole/part relationships among peers.

You model an aggregation as a line with a hollow diamond on the "whole/part" end. Figure C.10 illustrates the UML notation for aggregation.

FIGURE C.10
Aggregation.

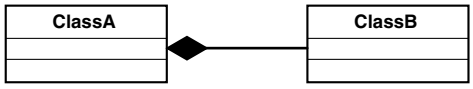


Composition

The UML provides notation for composition. A *composition* is a special type of association that models has-a of whole/part relationships among classes that are not peers. The part is not independent of the whole in a composition relationship.

You model composition as a line with a blackened diamond on the "whole/part" end. Figure C.11 illustrates the UML notation for composition.

FIGURE C.11
Composition.

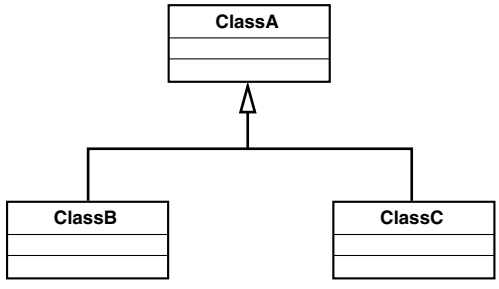


Generalization

A *generalization* relationship exists between the general and the specific. It is inheritance.

Generalization is symbolized through a solid line with a closed hollow arrow head. Figure C.12 illustrates the UML notation for generalization.

FIGURE C.12
Generalization.



Interaction Diagrams

Interaction diagrams model the interactions between objects.

Collaboration Diagrams

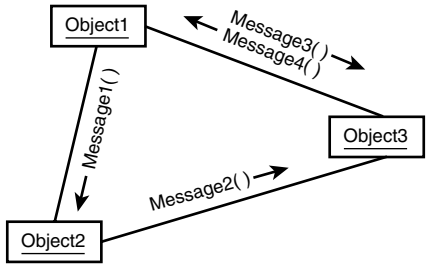
Collaboration diagrams represent the messages that the objects send one another.

Each object in the diagram is symbolized as a box. A line connects each object that interacts. On top of that line, you write the messages that the objects send and the direction of those messages.

Collaboration diagrams highlight the relationships between the actors. Figure C.13 illustrates the UML collaboration diagram.

FIGURE C.13

A collaboration diagram.



Sequence Diagrams

Sequence diagrams model the sequence of events in a scenario over time.

Each object in the diagram is symbolized at the top of the diagram as a box. The lines descending from the boxes represent the object's lifeline. Messages are passed back and forth between the objects and within the objects. Figure C.14 illustrates the UML sequence diagram.

FIGURE C.14

A sequence diagram.