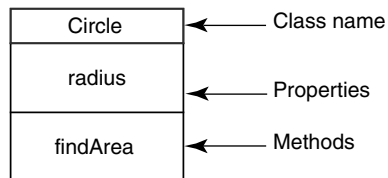


## UML GRAPHICAL NOTATIONS

This appendix summarizes the UML notations used in this book.

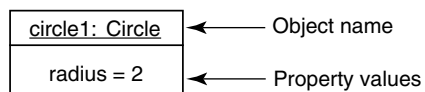
### Classes and Objects

A class is described using a rectangle box with three sections.



The top section gives the class name, the middle section describes the fields, and the bottom section describes the methods. The middle and bottom sections are optional, but the top section is required.

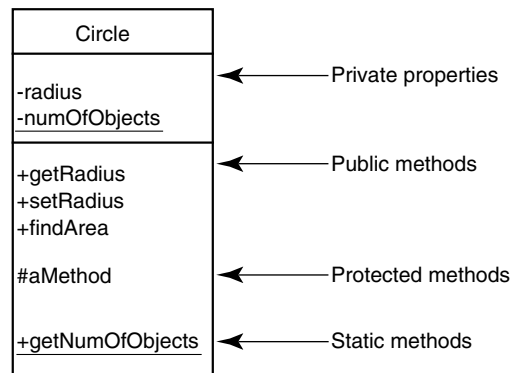
An object is described using a rectangle box with two sections.



The top section is required. It gives the object's name and its defining class. The second section is optional; it indicates the object's field values.

### The Modifiers *public*, *private*, *protected*, and *static*

The symbols +, -, and # are used to denote, respectively, *public*, *private*, and *protected* modifiers in the UML. The static fields and methods are underlined.



## Class Relationships

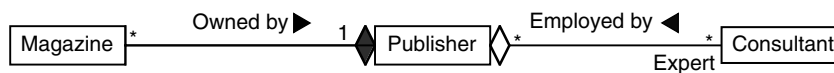
The relationships of the classes are association, aggregation, and inheritance.

An association is illustrated using a solid line between the two classes with an optional label that describes their relationship.

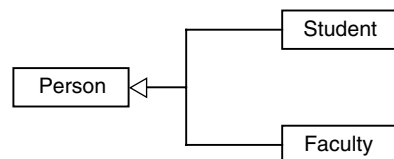


Each class involved in an association may specify a multiplicity. A multiplicity is a number or an interval that specifies the number of objects of the class that are involved in the relationship. The character `*` means that the number of objects is unlimited, and an interval `1..u` means that the number of objects should be between 1 and `u`, inclusive.

A filled diamond is attached to the composed class to denote the composition relationship, and a hollow diamond is attached to the aggregated class to denote the aggregation relationship, as shown below.

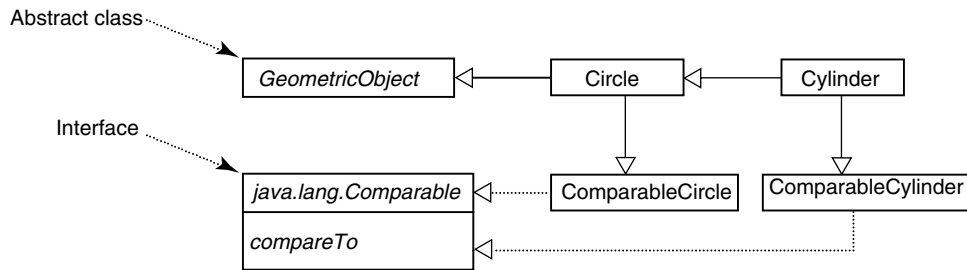


*Inheritance* models the is-a relationship between two classes, as shown below. An open triangle pointing to the superclass is used to denote the inheritance relationship between the two classes involved.



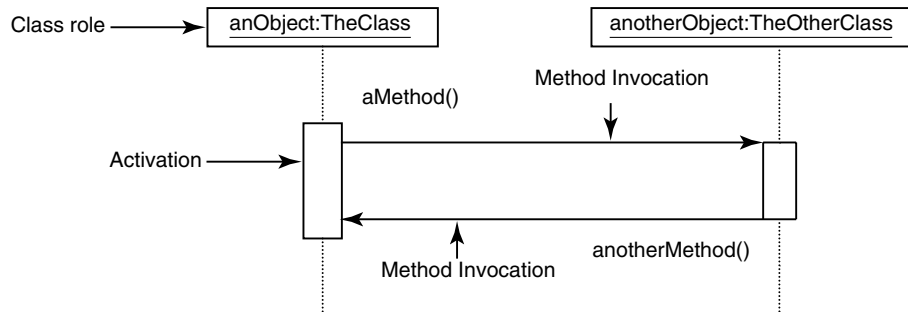
## Abstract Classes and Interfaces

Abstract class names, interface names, and abstract methods are italicized. Dashed lines are used to link to the interface, as shown below:



## Sequence Diagrams

Sequence diagrams describe interactions among objects by depicting the time ordering of method invocations. The sequence diagram shown below consists of the following elements:

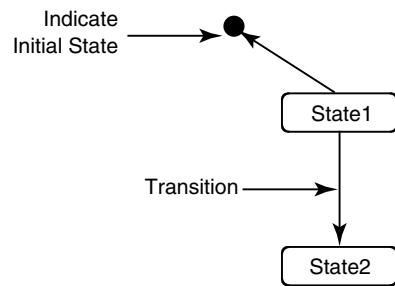


- Class role represents the role an object plays. The objects at the top of the diagram represent class roles.
- Lifeline represents the existence of an object over a period of time. A vertical dashed line extending from the object is used to denote a lifeline.
- Activation represents the time during which an object is performing an operation. Thin rectangles placed on lifelines are used to denote activations.
- Method invocation represents communications between objects. Horizontal arrows labeled with method calls are used to denote method invocations.

## Statechart Diagrams

Statechart diagrams describe the flow of control of an object. The statechart diagram shown below contains the following elements:

## APPENDIXES



- State represents a situation during the life of an object in which it satisfies some condition, performs some action, or waits for some event to occur. Every state has a name. Rectangles with rounded corners are used to represent states. The small filled circle is used to denote the initial state.
- Transition represents the relationship between two states, indicating that an object will perform some action to transfer from one state to the other. A solid arrow with appropriate method invocation is used to denote a transition.