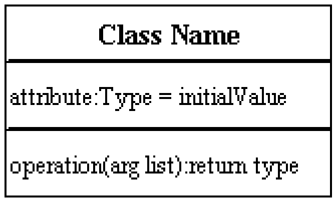
## What is a UML Class Diagram?

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

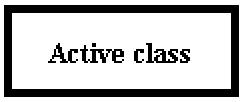
## Basic Class Diagram Symbols and Notations

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.


Class Name
attribute:Type = initialValue
operation(arg list):return type

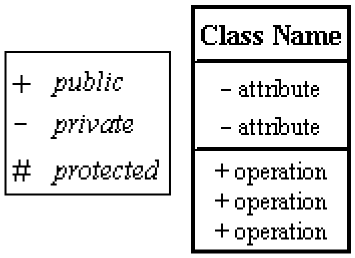
### Classes

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



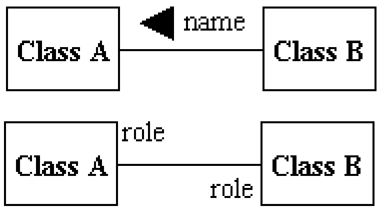
### Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



### Visibility

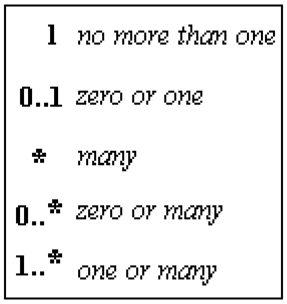
Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.



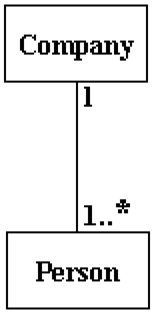
### Associations

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.
*Note:* It's uncommon to name both the

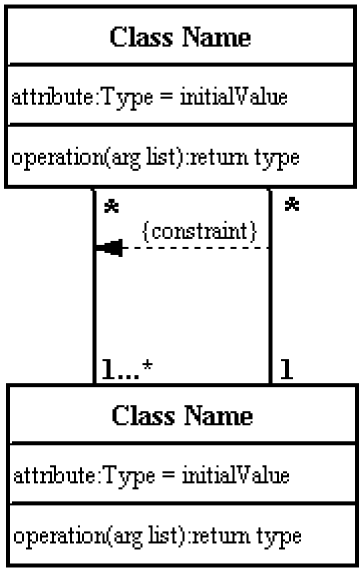
association and the class roles.



| | |
|---|---|
| 1 | *no more than one* |
| 0..1 | *zero or one* |
| * | *many* |
| 0..* | *zero or many* |
| 1..* | *one or many* |

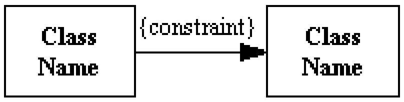


Multiplicity (Cardinality)
Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to *one* instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

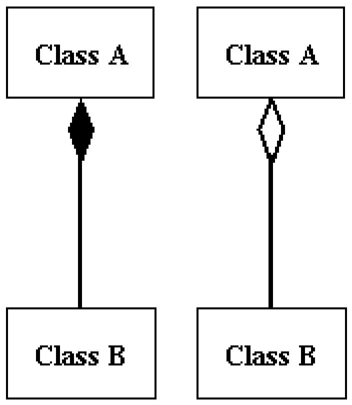

Class Name
attribute:Type = initialValue
operation(arg list):return type
*
*
{constraint}
1...*
1
Class Name
attribute:Type = initialValue
operation(arg list):return type

*Complex Constraint*

Constraint
Place constraints inside curly braces {}.

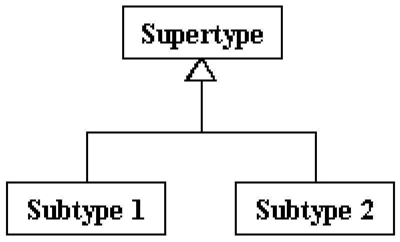


*Simple Constraint*



## Composition and Aggregation

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond.
Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other.
The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate.



## Generalization

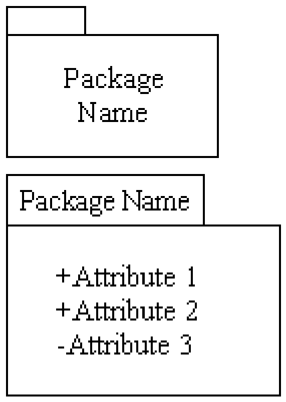
Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.

In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the superclass.

## What is a UML Package Diagram?

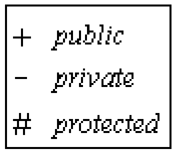
Package diagrams organize the elements of a system into related groups to minimize dependencies among them

## Basic Package Diagram Symbols and Notations



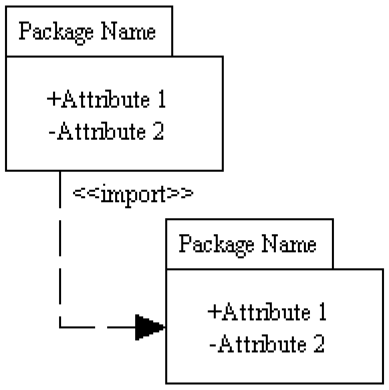
### Packages

Use a tabbed folder to illustrate packages. Write the name of the package on the tab or inside the folder. Similar to classes, you can also list the attributes of a package.



### Visibility

Visibility markers signify who can access the information contained within a package. Private visibility means that the attribute or the operation is not accessible to anything outside the package. Public visibility allows an attribute or an operation to be viewed by other packages. Protected visibility makes an attribute or operation visible to packages that inherit it only.



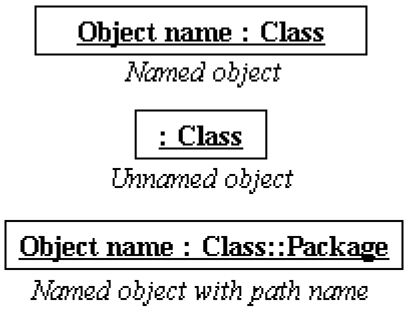
### Dependency

Dependency defines a relationship in which changes to one package will affect another package. Importing is a type of dependency that grants one package access to the contents of another package.

## What is a UML Object Diagram?

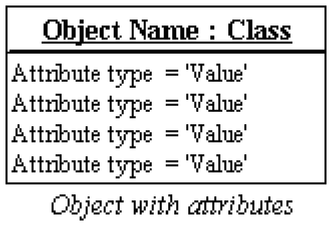
Object diagrams are also closely linked to class diagrams. Just as an object is an instance of a class, an object diagram could be viewed as an instance of a class diagram. Object diagrams describe the static structure of a system at a particular time and they are used to test the accuracy of class diagrams.

## Basic Object Diagram Symbols and Notations



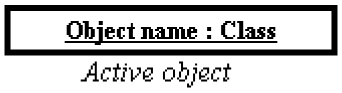
### Object names

Each object is represented as a rectangle, which contains the name of the object and its class underlined and separated by a colon.



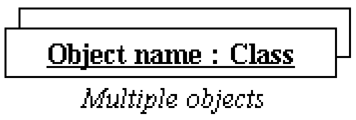
### Object attributes

As with classes, you can list object attributes in a separate compartment. However, unlike classes, object attributes must have values assigned to them.



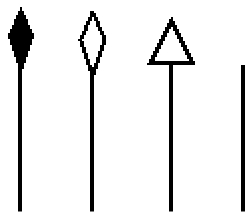
### Active object

Objects that control action flow are called active objects. Illustrate these objects with a thicker border.

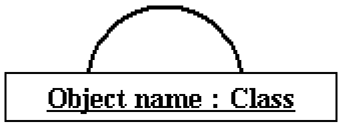


### Multiplicity

You can illustrate multiple objects as one symbol if the attributes of the individual objects are not important.


*Links*

Links
Links are instances of associations. You can draw a link using the lines used in class diagrams.

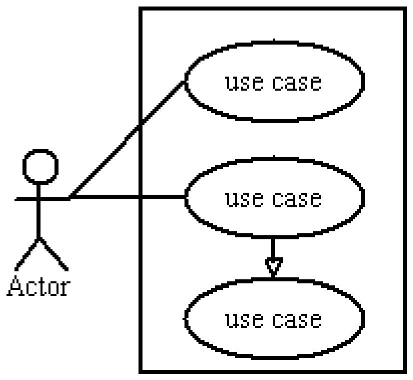


Self-linked
Objects that fulfill more than one role can be self-linked. For example, if Mark, an administrative assistant, also fulfilled the role of a marketing assistant, and the two positions are linked, Mark's instance of the two classes will be self-linked.

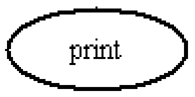
## What is a UML Use Case Diagram?

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

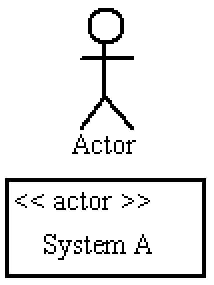
## Basic Use Case Diagram Symbols and Notations



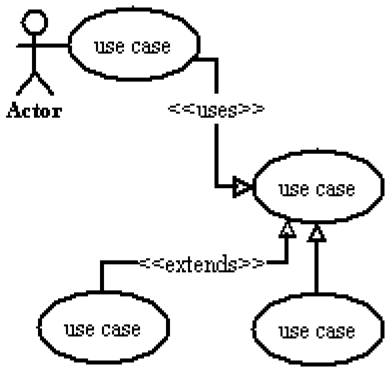
System
Draw your system's boundries using a rectangle that contains use cases. Place actors outside the system's boundries.



Use Case
Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.



Actors
Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



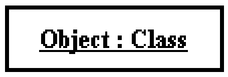
Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.

## What is a UML Sequence Diagram?

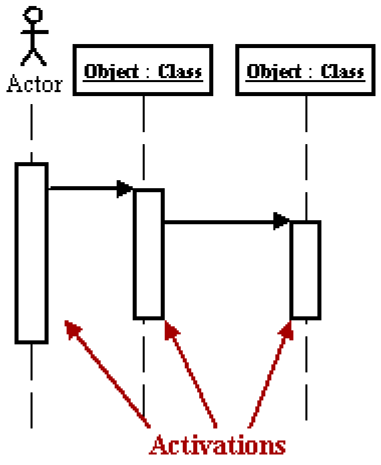
Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

## Basic Sequence Diagram Symbols and Notations



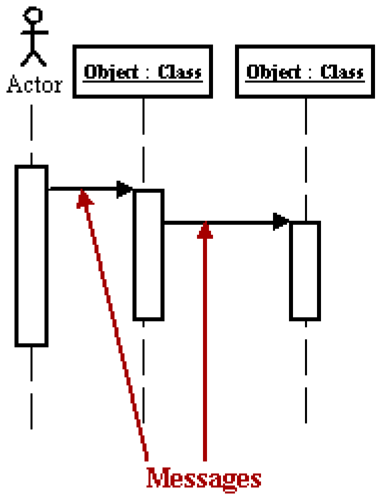
Class roles

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



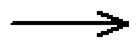
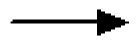
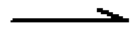
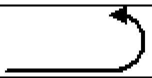
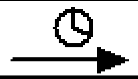
Activation

Activation boxes represent the time an object needs to complete a task.

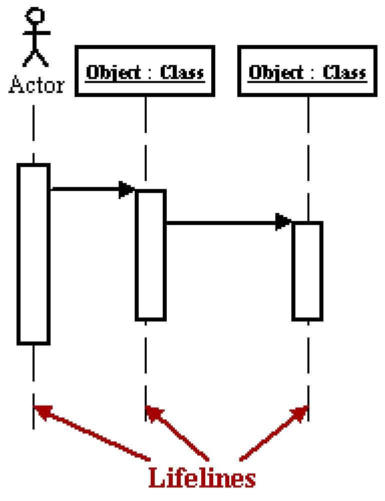


Messages

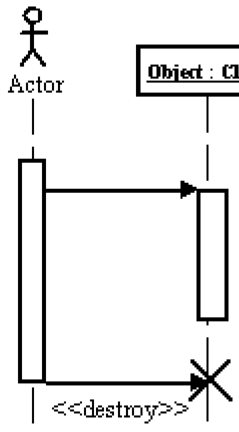
Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

| Arrow | Message type |
|---|---|
|  | Simple |
|  | Synchronous |
|  | Asynchronous |
|  | Balking |
|  | Time out |

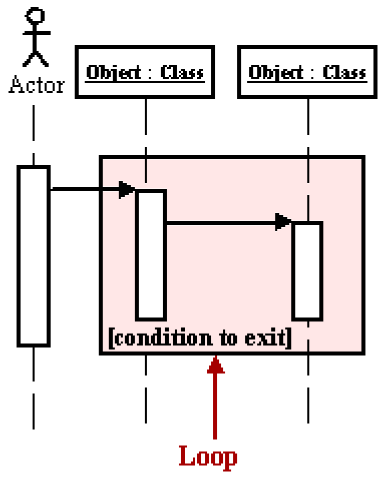
*Various message types for Sequence and Collaboration diagrams*



Lifelines
Lifelines are vertical dashed lines that indicate the object's presence over time.



Destroying Objects
Objects can be terminated early using an arrow labeled "< < destroy > >" that points to an X.



Loops
A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].