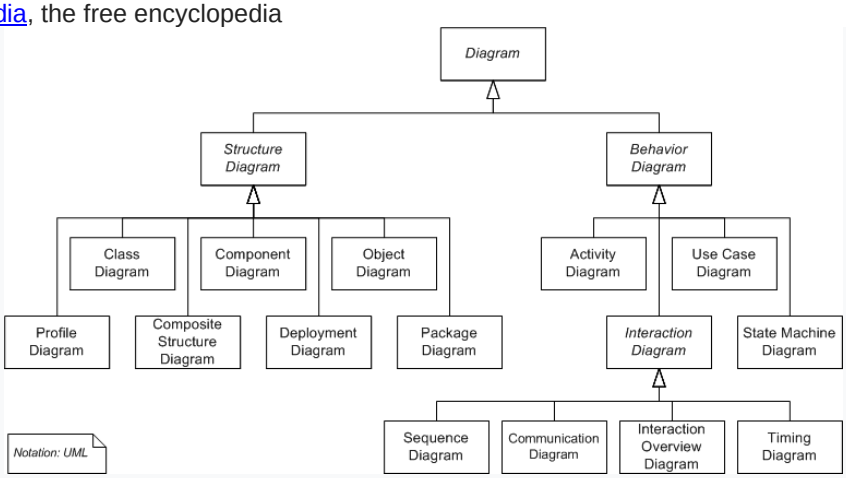
Class diagram

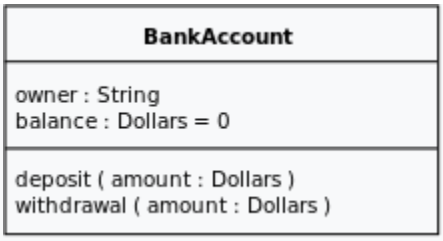
Adapted from Wikipedia, the free encyclopedia



Hierarchy of UML 2.5 Diagrams, shown as a class diagram. The individual classes are represented just with one compartment, but they often contain up to three compartments.

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. This document documents a subset of the UML class diagram notation as documented and used in the books *Head First Software Development* and *Head First Design Patterns.*

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



A class with three compartments.

In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized. If the class is actually an interface, then the additional indicator <<interface>> should be shown above the name.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase. Types are optional.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase. Type information is optional.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behaviour of systems, these class diagrams can be complemented by a state diagram or UML state machine.[2]

UML provides mechanisms to represent class members, such as attributes and methods, and additional information about them.

Members

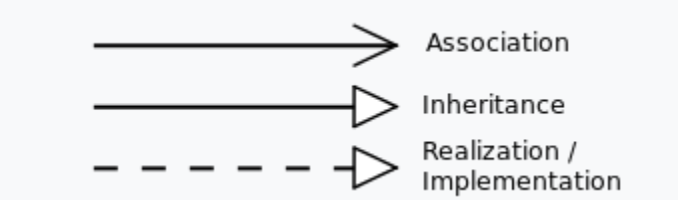
UML provides mechanisms to represent class members, such as attributes and methods, and additional information about them.

Visibility

To specify the visibility of a class member (i.e. any attribute or method), the following notations are placed before the member's name:[3]

+	Public
−	Private
#	Protected

Relationships

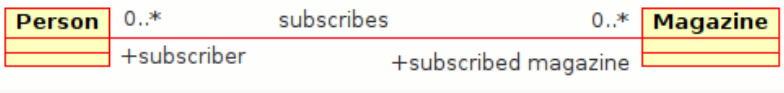


UML relations notation (abbreviated)

A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML defines the following relationships:

Instance-level relationships

Association

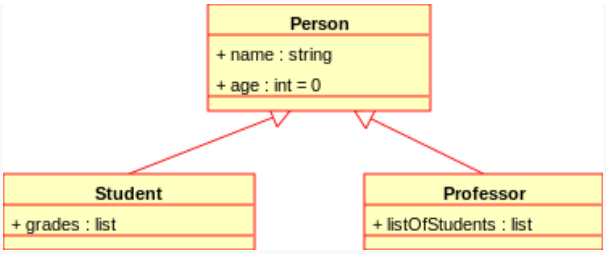


Class diagram example of association between two classes

An *association* represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes. An association with three links is called a ternary association. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.
There are four different types of association: bi-directional, uni-directional, aggregation (includes composition aggregation) and reflexive. Bi-directional and uni-directional associations are the most common ones.
For instance, a flight class is associated with a plane class bi-directionally. Association represents the static relationship shared among the objects of two classes.

Class-level relationships

Generalization



Class diagram showing generalization between the superclass *Person* and the two subclasses *Student* and *Professor*

It indicates that one of the two related classes (the *subclass*) is considered to be a specialized form of the other (the *super type*) and the superclass is considered a Generalization of the subclass. In practice, this means that any instance of the subtype is also an instance of the superclass. An exemplary tree of generalizations of this form is found in biological classification: humans are a subclass of simian, which is a subclass of mammal, and so on. The relationship is most easily understood by the phrase 'an A is a B' (a human is a mammal, a mammal is an animal).

The UML graphical representation of a Generalization is a hollow triangle shape on the superclass end of the line (or tree of lines) that connects it to one or more subtypes.

The generalization relationship is also known as the *inheritance* or *"is a"* relationship.

The *superclass* (base class) in the generalization relationship is also known as the *"parent"*, *superclass*, *base class*, or *base type*.

The *subtype* in the specialization relationship is also known as the *"child"*, *subclass*, *derived class*, *derived type*, *inheriting class*, or *inheriting type*.

Note that this relationship bears no resemblance to the biological parent–child relationship: the use of these terms is extremely common, but can be misleading.

A is a type of B
For example, "an oak is a type of tree", "an automobile is a type of vehicle"

Generalization can only be shown on class diagrams and on use case diagrams.

Realization

In UML modelling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behavior that the other model element (the supplier) specifies.

The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the *dashed* line (or tree of lines) that connects it to one or more implementers. A plain arrow head is used on the interface end of the dashed line that connects it to its users. In component diagrams, the ball-and-socket graphic convention is used (implementors expose a ball or lollipop, whereas users show a socket).

Realizations can only be shown on class or component diagrams.

A realization is a relationship between classes, interfaces, components, and packages that connects a client element with a supplier element. A realization relationship between classes and interfaces and between components and interfaces shows that the class realizes the operations offered by the interface.

Multiplicity

This association relationship indicates that (at least) one of the two related classes make reference to the other. This relationship is usually described as "A has a B" (a mother cat has kittens, kittens have a mother cat).

The UML representation of an association is a line connecting the two associated classes. At each end of the line there is optional notation. For example, we can indicate, using an arrowhead that the pointy end is visible from the arrow tail. We can indicate ownership by the placement of a ball, the role the elements of that end play by supplying

a name for the role, and the *multiplicity* of instances of that entity (the range of number of objects that participate in the association from the perspective of the other end).

0	No instances (rare)
0..1	No instances, or one instance
1	Exactly one instance
0..*	Zero or more instances
*****	Zero or more instances
1..*	One or more instances

See also

- Executable UML
- List of UML tools
- Object-oriented modeling

Related diagrams

- Domain model
- Entity–relationship model
- Object diagram

References[edit]

1. *Sparks, Geoffrey. "Database Modelling in UML". Retrieved 8 September 2011.*
2. Scott W. Ambler (2009) UML 2 Class Diagrams. Webdoc 2003-2009. Accessed Dec 2, 2009
3. *UML Reference Card, Version 2.1.2, Holub Associates, August 2007, retrieved 12 March 2011*
4. OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.3: May 2010. Retrieved 23 September 2010.
5. Fowler (2003) UML Distilled: A Brief Guide to the Standard Object Modeling Language
6. UML Tutorial part 1: class diagrams
7. *Goodwin, David. "Modelling and Simulation, p. 26" (PDF). The University of Warwick. Retrieved 28 November 2015.*

External links[edit]

Wikimedia Commons has media related to ***Class diagram***.

- Introduction to UML 2 Class Diagrams
- UML 2 Class Diagram Guidelines
- IBM Class diagram Introduction

- OMG UML 2.2 specification documents
- UML 2 Class Diagrams

This page is adapted from https://en.wikipedia.org/wiki/Class_diagram