Источник: <https://prog-cpp.ru/uml-classes/>  
  
UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Словарь UML включает три вида строительных блоков:

* Диаграммы.
* Сущности.
* Связи.

Сущности – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей.  
Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Язык UML включает 13 видов диаграмм, среди которых на первом месте в списке - диаграмма классов, о которой и пойдет речь.  
Диаграммы классов показывают набор классов, интерфейсов, а также их связи. Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. Они предназначены для статического представления системы.  
Большинство элементов UML имеют уникальную и прямую графическую нотацию, которая дает визуальное представление наиболее важных аспектов элемента.

Сущности

Диаграммы классов оперируют тремя видами сущностей UML:

* Структурные.
* Поведенческие.
* Аннотирующие.

Структурные сущности – это «имена существительные» в модели UML. В основном, статические части модели, представляющие либо концептуальные, либо физические элементы. Основным видом структурной сущности в диаграммах классов является класс.  
   
Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели во времени и пространстве. Основной из них является взаимодействие – поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели. Сообщение изображается в виде линии со стрелкой, почти всегда сопровождаемой именем операции.  
   
Поведенческие сущности  
Аннотирующие сущности – это поясняющие части UML-моделей, иными словами, комментарии, которые можно применить для описания, выделения и пояснения любого элемента модели. Главная из аннотирующих сущностей – примечание. Это символ, служащий для описания ограничений и комментариев, относящихся к элементу либо набору элементов. Графически представлен прямоугольником с загнутым углом; внутри помещается текстовый или графический комментарий.  
   


Структурные сущности - классы

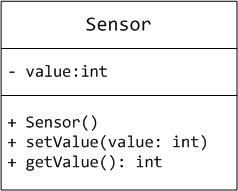
Класс – это описание набора объектов с одинаковыми атрибутами, операциями, связями и семантикой.  
  
Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

* имя класса
* атрибуты (свойства) класса
* операции (методы) класса.

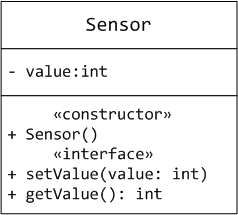
Для атрибутов и операций может быть указан один из трех типов видимости:

* - - private (частный)
* # - protected (защищенный)
* + - public (общий)

Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента.  
  
Каждый класс должен обладать именем, отличающим его от других классов. Имя – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк.  
На практике обычно используются краткие имена классов, взятые из словаря моделируемой системы. Каждое слово в имени класса традиционно пишут с заглавной буквы (верблюжья конвенция), например Sensor (Датчик) или TemperatureSensor (ДатчикТемпературы).



Для абстрактного класса имя класса записывается курсивом.  
  
   
Атрибут (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута. Класс может иметь любое число атрибутов или не иметь ни одного. В последнем случае блок атрибутов оставляют пустым.  
Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может представлять собой текст. На практике для именования атрибута используются одно или несколько коротких существительных, выражающих некое свойство класса, к которому относится атрибут.  
  
Можно уточнить спецификацию атрибута, указав его тип, кратность (если атрибут представляет собой массив некоторых значений) и начальное значение по умолчанию.  
  
Статические атрибуты класса обозначаются подчеркиванием.  
  
Операция (метод) – это реализация метода класса. Класс может иметь любое число операций либо не иметь ни одной. Часто вызов операции объекта изменяет его атрибуты.  
Графически операции представлены в нижнем блоке описания класса.  
Допускается указание только имен операций. Имя операции, как и имя класса, должно представлять собой текст. На практике для именования операции используются короткие глагольные конструкции, описывающие некое поведение класса, которому принадлежит операция. Обычно каждое слово в имени операции пишется с заглавной буквы, за исключением первого, например move (переместить) или isEmpty (проверка на пустоту).  
Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а применительно к функциям – тип возвращаемого значения.  
  
Абстрактные методы класса обозначаются курсивным шрифтом.  
Статические методы класса обозначаются подчеркиванием.  
  
Изображая класс, не обязательно показывать сразу все его атрибуты и операции. Для конкретного представления, как правило, существенна только часть атрибутов и операций класса. В силу этих причин допускается упрощенное представление класса, то есть для графического представления выбираются только некоторые из его атрибутов. Если помимо указанных существуют другие атрибуты и операции, вы даете это понять, завершая каждый список многоточием.  
Чтобы легче воспринимать длинные списки атрибутов и операций, желательно снабдить префиксом (именем стереотипа) каждую категорию в них. В данном случае стереотип – это слово, заключенное в угловые кавычки, которое указывает то, что за ним следует.



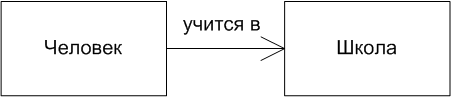
Отношения между классами

Существует четыре типа связей в UML:

* Зависимость
* Ассоциация
* Обобщение
* Реализация

Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.  
  
Первая из них – зависимость – семантически представляет собой связь между двумя элементами модели, в которой изменение одного элемента (независимого) может привести к изменению семантики другого элемента (зависимого). Графически представлена пунктирной линией, иногда со стрелкой, направленной к той сущности, от которой зависит еще одна; может быть снабжена меткой.  
   
Зависимость

Зависимость – это связь использования, указывающая, что изменение спецификаций одной сущности может повлиять на другие сущности, которые используют ее.  
  
   
Ассоциация – это структурная связь между элементами модели, которая описывает набор связей, существующих между объектами.  
Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому.  
Например, класс Человек и класс Школа имеют ассоциацию, так как человек может учиться в школе. Ассоциации можно присвоить имя «учится в». В представлении однонаправленной ассоциации добавляется стрелка, указывающая на направление ассоциации.



Двойные ассоциации представляются линией без стрелок на концах, соединяющей два классовых блока.  
Ассоциация может быть именованной, и тогда на концах представляющей её линии будут подписаны роли, принадлежности, индикаторы, мультипликаторы, видимости или другие свойства.  
  
Множественность ассоциации представляет собой диапазон целых чисел, указывающий возможное количество связанных объектов. Он записывается в виде выражения с минимальным и максимальным значением; для их разделения используются две точки. Устанавливая множественность дальнего конца ассоциации, вы указываете, сколько объектов может существовать на дальнем конце ассоциации для каждого объекта класса, находящегося на ближнем ее конце. Количество объектов должно находиться в пределах заданного диапазона. Множественность может быть определена как единица 1, ноль или один 0..1, любое значение 0..\* или \*, один или несколько 1..\*. Можно также задавать диапазон целых значений, например 2..5, или устанавливать точное число, например 3.



Агрегация – особая разновидность ассоциации, представляющая структурную связь целого с его частями. Как тип ассоциации, агрегация может быть именованной. Одно отношение агрегации не может включать более двух классов (контейнер и содержимое).  
Агрегация встречается, когда один класс является коллекцией или контейнером других. Причём, по умолчанию агрегацией называют агрегацию по ссылке, то есть когда время существования содержащихся классов не зависит от времени существования содержащего их класса. Если контейнер будет уничтожен, то его содержимое — нет.  
Графически агрегация представляется пустым ромбом на блоке класса «целое», и линией, идущей от этого ромба к классу «часть».

Агрегация

Композиция — более строгий вариант агрегации. Известна также как агрегация по значению.  
Композиция – это форма агрегации с четко выраженными отношениями владения и совпадением времени жизни частей и целого. Композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет также уничтожено.  
Графически представляется как и агрегация, но с закрашенным ромбиком.

Композиция  
  
Третья связь – обобщение – выражает специализацию или наследование, в котором специализированный элемент (потомок) строится по спецификациям обобщенного элемента (родителя). Потомок разделяет структуру и поведение родителя. Графически обобщение представлено в виде сплошной линии с пустой стрелкой, указывающей на родителя.

Обобщение

Четвертая – реализация – это семантическая связь между классами, когда один из них (поставщик) определяет соглашение, которого второй (клиент) обязан придерживаться. Это связи между интерфейсами и классами, которые реализуют эти интерфейсы. Это, своего рода, отношение «целое-часть». Поставщик, как правило, представлен абстрактным классом. В графическом исполнении связь реализации – это гибрид связей обобщения и зависимости: треугольник указывает на поставщика, а второй конец пунктирной линии – на клиента.

Реализация