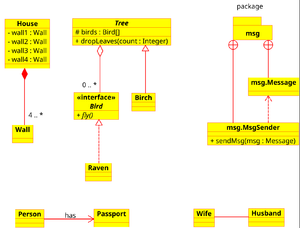
# Диаграмма классов



**Диаграмма классов** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *Static Structure diagram*) — структурная [диаграмма](https://ru.wikipedia.org/wiki/Диаграмма_(UML)) языка моделирования [UML](https://ru.wikipedia.org/wiki/UML), демонстрирующая общую структуру иерархии [классов](https://ru.wikipedia.org/wiki/Класс_(программирование)) системы, их коопераций, [атрибутов](https://ru.wikipedia.org/wiki/Поле_класса) (полей), [методов](https://ru.wikipedia.org/wiki/Метод_(языки_программирования)), интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

## Введение

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, [типов](https://ru.wikipedia.org/wiki/Тип_данных) и т. п.) Она содержит в себе также некоторые элементы поведения (например — операции), однако их динамика должна быть отражена на диаграммах других видов ([диаграммах коммуникации](https://ru.wikipedia.org/wiki/Диаграмма_коммуникации), диаграммах состояний). Для удобства восприятия диаграмму классов можно также дополнить представлением [пакетов](https://ru.wikipedia.org/wiki/Пакет_(UML)), включая вложенные[[2]](https://ru.wikipedia.org/wiki/Диаграмма_классов" \l "cite_note-_00071d72222e96c1-2).

При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от маловажных деталей и концепций, которые не относятся к реальности (производительность, [инкапсуляция](https://ru.wikipedia.org/wiki/Инкапсуляция_(программирование)), [видимость](https://ru.wikipedia.org/wiki/Видимость_(UML)) и т. п.). Классы можно рассматривать с позиции различных уровней. Как правило, их выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации:

* на уровне анализа класс содержит в себе только набросок общих контуров системы и работает как логическая концепция предметной области или программного продукта.
* на уровне проектирования класс отражает основные проектные решения касательно распределения информации и планируемой функциональности, объединяя в себе сведения о состоянии и операциях.
* на уровне реализации класс дорабатывается до такого вида, в каком он максимально удобен для воплощения в выбранной среде разработки; при этом не воспрещается опустить в нём те общие свойства, которые не применяются на выбранном языке программирования.

## Элементы диаграммы

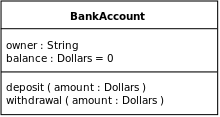


Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента:

* В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный — то его имя пишется полужирным курсивом.
* Посередине располагаются поля (атрибуты) класса. Они выровнены по левому краю и начинаются с маленькой буквы.
* Нижняя часть содержит методы класса. Они также выровнены по левому краю и пишутся с маленькой буквы.

## Члены

Язык UML предоставляет механизмы для представления членов класса, например атрибутов и методов, а также дополнительной информации о них.

### **Видимость**

Для задания [видимости](https://ru.wikipedia.org/wiki/Видимость_(UML)) членов класса (то есть — любым атрибутам или методам), эти обозначения должны быть размещены перед именем участника:[[](https://ru.wikipedia.org/wiki/Диаграмма_классов" \l "cite_note-HOL07-4)

|  |  |
| --- | --- |
| + | Публичный (Public) |
| - | Приватный (Private) |
| # | Защищённый (Protected) |
| / | Производный (Derived) (может быть совмещён с другими) |
| ~ | Пакет (Package) |

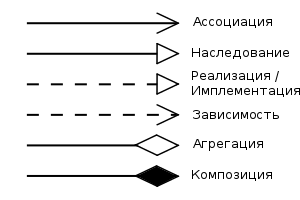
### **Области действия**

UML определяет два типа областей действия для членов: *экземпляр* и *классификатор*, последние имеют подчёркнутые имена.

* **Члены классификаторы**. Во многих языках называются static. Область действия — сам класс.
  + Значения полей одинаковы для всех экземпляров в данной единице трансляции
  + Вызов метода не меняет состояние объекта
* **Члены экземпляры**. Область действия — объект.
  + Значения полей могут отличаться в разных объектах
  + Методы могут изменять поля

Чтобы показать принадлежность к классификатору, имя подчёркивается, в противном случае область действия полагается областью действия по-умолчанию.

## Взаимосвязи[[править](https://ru.wikipedia.org/w/index.php?title=Диаграмма_классов&veaction=edit&section=6) | [править код](https://ru.wikipedia.org/w/index.php?title=Диаграмма_классов&action=edit&section=6)]



Взаимосвязь — это особый тип логических отношений между сущностями, показанных на [диаграммах](https://ru.wikipedia.org/wiki/Диаграмма) классов и [объектов](https://ru.wikipedia.org/wiki/Диаграмма_объектов). В UML представлены следующие виды отношений:

### **Взаимосвязи объектов классов**

[**Зависимость**](https://ru.wikipedia.org/w/index.php?title=Зависимость_(UML)&action=edit&redlink=1) обозначает такое отношение между классами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса, но не наоборот.

[**Ассоциация**](https://ru.wikipedia.org/wiki/Ассоциация_(объектно-ориентированное_программирование))показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому. Является общим случаем композиции и агрегации.

Например, класс Человек и класс Школа имеют ассоциацию, так как человек может учиться в школе. Ассоциации можно присвоить имя «учится в».

Двойные ассоциации представляются линией без стрелочек на концах, соединяющей два классовых блока. Ассоциации более высокой степени имеют более двух концов и представляются линиями, один конец которых идёт к классовому блоку, а другой к общему ромбику. В представлении однонаправленной ассоциации добавляется стрелка, указывающая на направление ассоциации.

Ассоциация может быть именованной, и на концах представляющей её линии могут быть подписаны роли, принадлежности, индикаторы, мультипликаторы, видимости или другие свойства.



Диаграмма классов, показывающая Агрегацию между двумя классами

[***Агрегация***](https://ru.wikipedia.org/wiki/Агрегирование_(программирование))— это разновидность ассоциации при отношении между целым и 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, и линией, идущей от этого ромба к содержащемуся классу.

[***Композиция***](https://ru.wikipedia.org/w/index.php?title=Композиция_(объектно-ориентированное_программирование)&action=edit&redlink=1)— более строгий вариант агрегации. Известна также как агрегация по значению.

*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.

#### **Различия между композицией и агрегацией**

Приведём наглядный пример. Комната является частью квартиры, следовательно здесь подходит композиция, потому что комната без квартиры существовать не может. А, например, мебель не является неотъемлемой частью квартиры, но в то же время, квартира содержит мебель, поэтому следует использовать агрегацию.

### **Взаимосвязи классов**



**Обобщение** (Generalization) показывает, что один из двух связанных классов (*подтип*) является частной формой другого (*надтипа*), который называется **обобщением** первого. На практике это означает, что любой экземпляр подтипа является также экземпляром надтипа. Например: животные — супертип млекопитающих, которые, в свою очередь, — супертип приматов, и так далее. Эта взаимосвязь легче всего описывается фразой «А — это Б» (приматы — это млекопитающие, млекопитающие — это животные).

Графически обобщение представляется линией с пустым треугольником у супертипа.

Обобщение также известно как наследование или «[is a](https://ru.wikipedia.org/w/index.php?title=Is-a&action=edit&redlink=1)» взаимосвязь (или отношение «является»).

**Реализация** — отношение между двумя элементами модели, в котором один элемент (*клиент*) реализует поведение, заданное другим (*поставщиком*). Реализация — отношение целое-часть. Графически реализация представляется так же, как и наследование, но с пунктирной линией.

Поставщик, как правило, является абстрактным классом или классом-интерфейсом.

### **Общая взаимосвязь**

**Зависимость** (dependency) — это слабая форма отношения использования, при котором изменение в спецификации одного влечёт за собой изменение другого, причём обратное не обязательно. Возникает, когда объект выступает, например, в форме параметра или локальной переменной.

Графически представляется штриховой стрелкой, идущей от зависимого элемента к тому, от которого он зависит.

Существует несколько именованных вариантов.

Зависимость может быть между экземплярами, классами или экземпляром и классом.

#### **Уточнения отношений**

Уточнение имеет отношение к уровню детализации. Один пакет уточняет другой, если в нём содержатся те же самые элементы, но в более подробном представлении. Например, при написании книги вы наверняка начнете с формулировки предложения, в котором кратко будет представлено содержание каждой главы. Предположим, что резюме к каждой главе в качестве отдельного элемента входит в пакет «Предложение». Допустим также, что «Завершённая книга» — это пакет, элементами которого являются законченные главы. В этом контексте пакет «Завершённая книга» является уточнением пакета «Предложение».

#### **Мощность отношений (Кратность)**

Мощность отношения (мультипликатор) означает число связей между каждым экземпляром класса (объектом) в начале линии с экземпляром класса в её конце. Различают следующие типичные случаи:

|  |  |  |
| --- | --- | --- |
| *нотация* | *объяснение* | *пример* |
| **0..1** | Ноль или один экземпляр | кошка имеет или не имеет хозяина |
| **1** | Обязательно один экземпляр | у кошки одна мать |
| **0..\*** или **\*** | Ноль или более экземпляров | у кошки могут быть, а может и не быть котят |
| **1..\*** | Один или более экземпляров | у кошки есть хотя бы одно место, где она спит |