Диаграмма классов

Диаграмма классов – одна из диаграмм набора UML, предназначенная для демонстрации структуры классов и их взаимодействий.

Диаграмма классов позволяет определить архитектуру объектно-ориентированной системы независимо от языка программирования и внутренней реализации, что позволяет использовать ее как для демонстрации универсальных подходов к проектированию систем, так и для планирования/документирования конкретной системы.

Для создания диаграммы классов может подойти любой инструмент для создания диаграмм UML. Некоторые из инструментов позволяют генерировать заголовки классов на языке программирования из построенной диаграммы. Из наиболее известных инструментов создания таких диаграмм: Rational Rose, Visual Paradigm, ArgoUML, MS Visio. Из них бесплатными являются Visual Paradigm Community Edition и ArgoUML. Для образовательного процесса мы рекомендуем Visual Paradigm Community Edition, однако можно использовать любой инструмент (даже не перечисленный здесь).

Класс

Есть два способа обозначения класса.

Если внутренняя структура класса неважна, он обозначается как прямоугольник, в центре которого указано его имя.

Если внутренняя структура класса важна, он обозначается как прямоугольник, разделенный на три секции. В верхней секции указывается имя класса. В средней – поля и свойства класса (атрибуты). В нижней – методы класса (операции). Если в классе нет полей или методов, секция может быть опущена.

На рисунке 1 изображен класс Cat с полем name, конструктором и методом say().

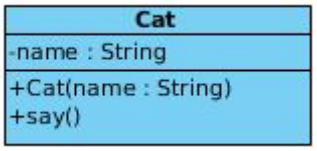

Рисунок 1

Области видимости

Закрытые (private) поля и методы обозначаются знаком - (минус) перед именем сущности.

Защищенные (protected) поля и методы обозначаются знаком # (решетка).

Публичные (public) поля и методы обозначаются знаком + (плюс).

Внутренние (package или internal в зависимости от языка) поля и методы обозначаются знаком ~ (тильда).

Вычисляемые свойства

Свойства, которые не имеют соответствующего им поля и вычисляются на основе других полей (или другой информации), помечаются знаком / (слэш) между указанием области видимости и именем свойства.

На рисунке 2 изображен класс, содержащий вычисляемое свойство full_name, которое вычисляется на основе полей first_name и last_name.

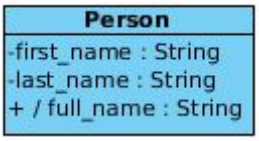

Рисунок 2

Атрибуты

Для указания атрибута достаточно указать только его имя. Однако для полноты описания можно указать тип данных после имени и двоеточия. Можно использовать любой допустимый для языка тип данных, в том числе указатели, ссылки, массивы, шаблонные типы, модификатор const.

Операции

Минимальный синтаксис указания операции: `имя(парамер1, параметр2, ...)`, например, `add(left, right)`. Для полноты описания можно указывать типы данных как параметров, так и возвращаемого значения. Тип параметра указывается непосредственно после имени соответствующего параметра, отделяясь от него двоеточием. Тип возвращаемого значения указывается после закрывающей скобки параметров, также отделяясь от нее двоеточием. Пример:

```
add(left: double, right: double): double
```

Статические поля и методы

Статические элементы класса обозначаются подчеркиванием.

На рисунке 3 изображен класс Date со статическим полем days_in_month и статическим методом isLeap().

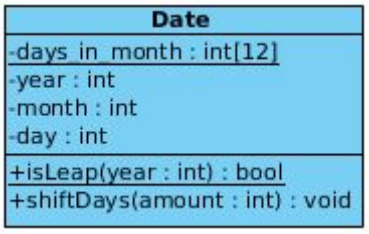

Рисунок 3

Абстрактные методы и классы

Абстрактные методы записываются курсивом. Имена абстрактных классов также указываются курсивом.

На рисунке 4 изображен абстрактный класс Animal с абстрактным методом say().

Рисунок 4

Интерфейс

Интерфейсы обозначаются аналогично классу, за исключением аннотации `<<interface>>`.

На рисунке 5 изображен интерфейс Animal. Для методов интерфейса указание абстрактности метода опционально, т.к. все методы интерфейса абстрактные.

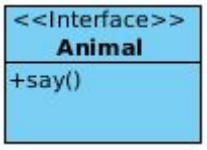

Рисунок 5

Наследование (обобщение)

Связь, при которой супертип (предок) *обобщает* свои подтипы (потомков). При этом говорят, что подтип *расширяет* супертип. Таким образом обозначают отношение наследования. Данный вид связи обозначается стрелкой со сплошной линией и пустым треугольником. Стрелка направляется от потомка к предку.

На рисунке 6 изображены класс Time (супертип) и его наследник DateTime (подтип).

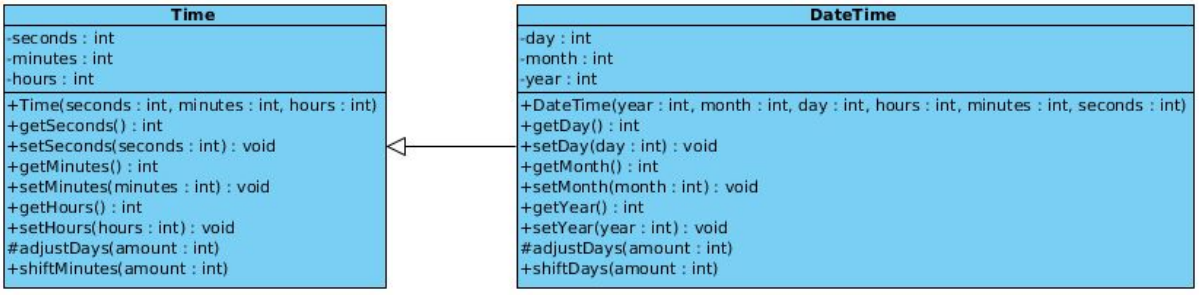

Рисунок 6

Реализация

Связь, при которой подтипы реализуют поведение супертипа-интерфейса. Данный вид связи обозначается пунктирной стрелкой с пустым треугольником. Стрелка направляется в сторону интерфейса.

На рисунке 7 изображен интерфейс Movable и реализующие его классы Point и Circle.

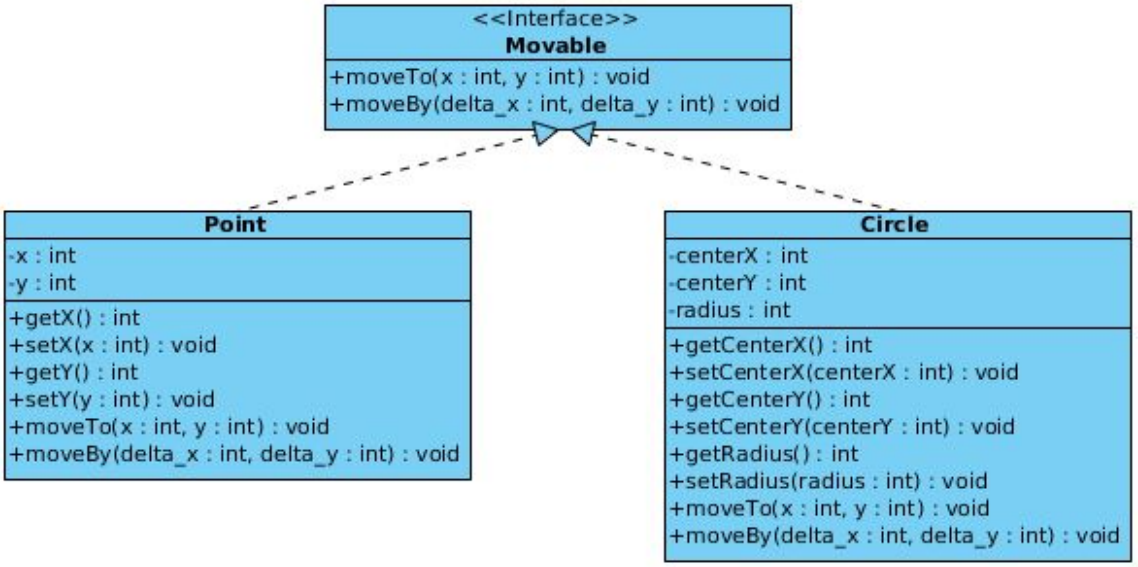

Рисунок 7

Зависимость

Самая слабая связь между классами. Обозначает ситуацию, когда изменение интерфейса одного класса влияет на работу другого, но обратное может быть неверно. Обозначается как пунктирная стрелка с угловым наконечником. Направлена от зависимого класса к независимому.

Технически может быть выражена разнообразными средствами, такими как типы параметров или возвращаемых значений методов, а также локальных переменных методов.

Из-за слабости связи она может возникать между большим количеством классов на схеме, поэтому в чистом виде изображается редко. Тем не менее, есть ситуации, в которых может потребоваться изобразить эту связь.

Использование

Форма связи “Зависимость”, выражающаяся в том, что для совершения какой-либо операции один класс использует другой (или целую иерархию). Обозначается аналогично зависимости, но с аннотацией `<<use>>` (рисунок 8).

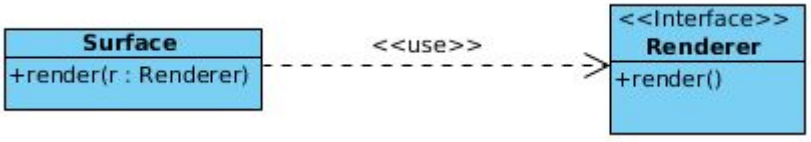

Рисунок 8

Инстанцирование

Форма связи “Зависимость”, выражающаяся в том, что один класс, создает экземпляры другого. Обозначается аналогично зависимости, но с аннотацией `<<instantiate>>`, и направлена в сторону инстанцируемого объекта (рисунок 9).

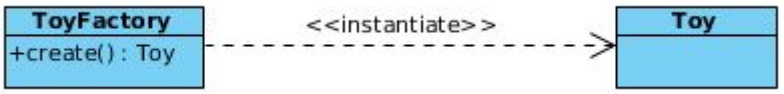

Рисунок 9

Однонаправленная ассоциация

Форма связи между классами, в которой один класс может предоставлять ссылку на другой класс. Обозначается как стрелка со сплошной линией с угловым наконечником. Зачастую подписана наименованием связи. Направлена от ссылающегося класса к классу, на который предоставляется ссылка (рисунок 10).

Технически может выражаться хранением одним классом ссылки на другой класс в качестве поля. Ссылка может быть как слабой (не владеющей), так и сильной (владеющей).

Рисунок 10

Двунаправленная и множественная ассоциация

Форма ассоциации, в которой все классы, входящие в эту связь, предоставляют ссылку друг на друга. Так как связь многонаправленная, она обозначается сплошной линией без стрелок (рисунок 11). Если связь имеет более двух концов, она может разветвляться.

Аналогично однонаправленной ассоциации ссылки на ассоциированные классы могут быть как слабыми, так и сильными.

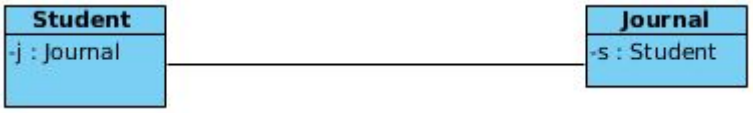

Рисунок 11

Агрегация

Частный случай ассоциации, в котором участвуют два класса, связанных отношением “часть-целое”. Целое может состоять из нескольких разных одиночный частей (в этом случае хранятся ссылки или указатели на экземпляры этих частей), или набора однотипных частей (коллекция, часто массив, ссылок или указателей на экземпляры частей).

Связь обозначается стрелкой со сплошной линией с пустым ромбом на конце. Стрелка направлена в сторону целого (рисунок 12).

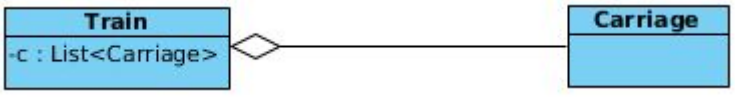

Рисунок 12

Композиция

Более строгая форма агрегации, в которой части неотделимы от целого и зачастую целое не может существовать без частей. Связь обозначается стрелкой со сплошной линией с закрашенным ромбом на конце. Стрелка направлена в сторону целого (рисунок 13).

В языках, в которых поддерживаются соответствующие возможности, реализуется при помощи поля с сильной ссылкой (класс-целое ответственен за освобождение памяти своих частей) или части хранятся в классе-целом по значению.

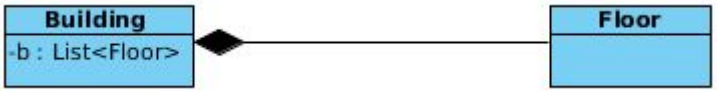

Рисунок 13

Комментарии

Комментарии представляют собой заметки в произвольной форме, оставленные на диаграмме внутри прямоугольников с загнутым углом. Блоки комментариев могут быть оставлены свободно на схеме, либо соединены с каким-то ее элементом при помощи пунктирной линии (рисунок 14).

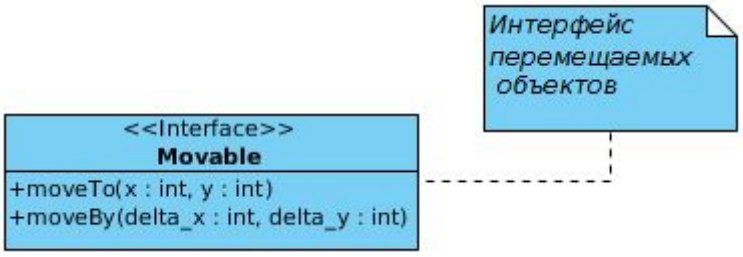

Рисунок 14

Другие обозначения

На диаграмме классов допускаются прочие пометки, предназначенные для группировки элементов. Также может использоваться общий для многих диаграмм UML элемент **пакет**. Он предназначен для группировки классов для обозначения какой-либо подсистемы. В языках с поддержкой пакетов сущность “пакет” транслируется непосредственно в пакеты языка. В других языках это может быть пространство имен, библиотека или просто папка. Пакет обозначается прямоугольником с ярлыком. Пакет на диаграмме может содержать ссылку на другую диаграмму, содержать внутри определения классов, либо быть связанным с определениями классов сплошной линией (рисунок 15).

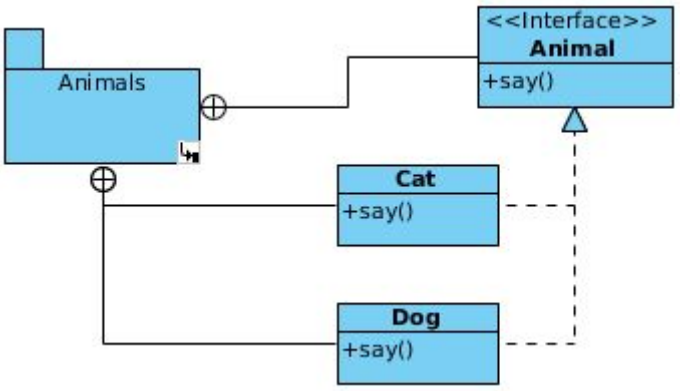

Рисунок 15