

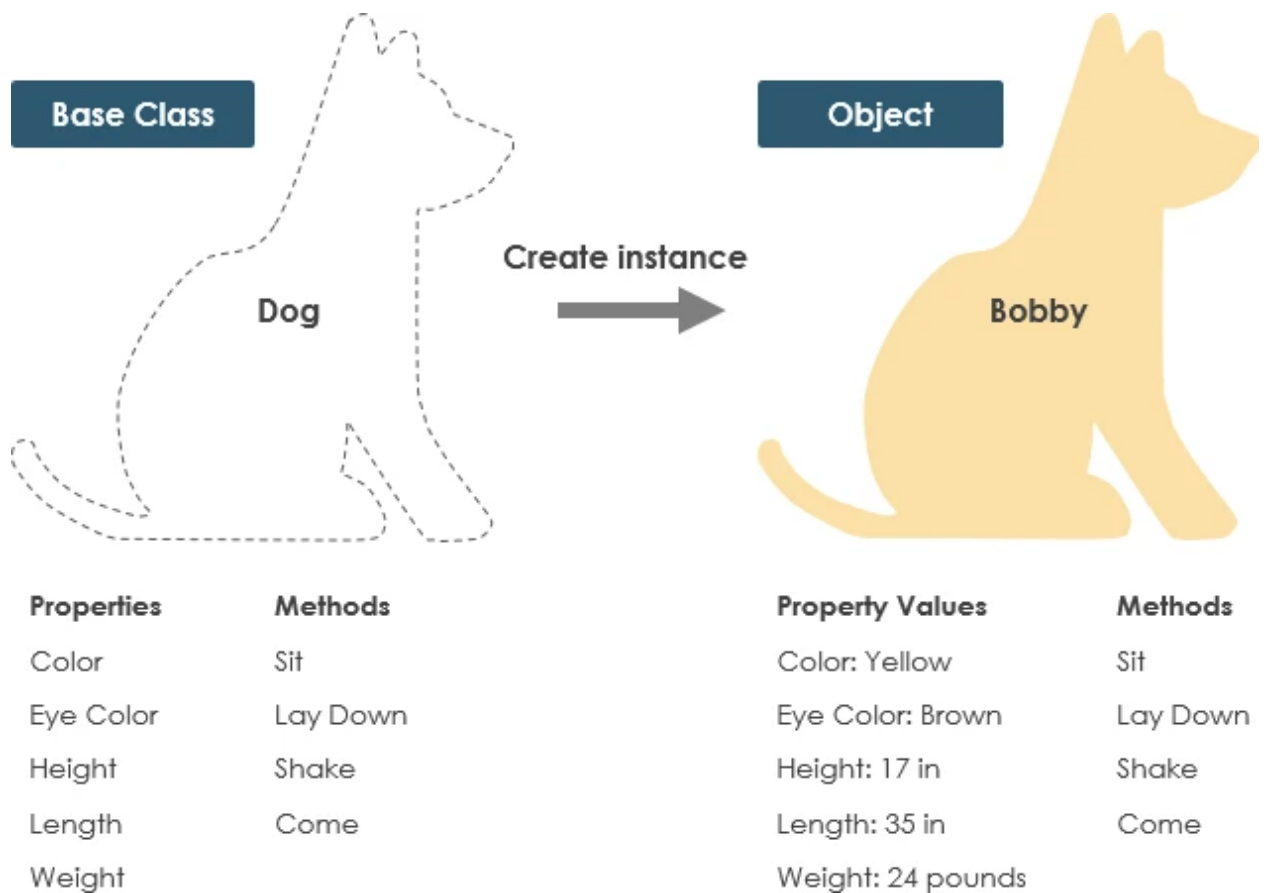
Урок 101. UML-диаграммы классов

Что такое диаграмма классов?

Диаграмма классов на унифицированном языке моделирования (UML) — это диаграмма статической структуры, которая описывает структуру системы, показывая ее классы, их атрибуты, операции (или методы) и отношения между объектами. Диаграмма классов — это план системы или подсистемы. Диаграммы классов можно использовать для моделирования объектов, составляющих систему, для демонстрации отношений между объектами и для описания ролей этих объектов и предоставляемых ими услуг.

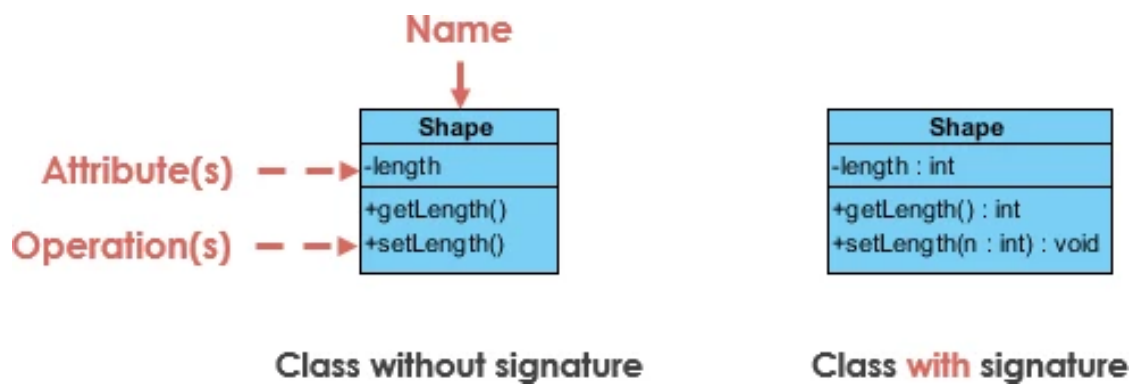
Пример класса

У собаки есть состояние (характеристики) — цвет, цвет глаз, высота, длина, вес, а также поведение (команды, которые она выполняет) — «сидеть», «лежать», «дай лапу», «ко мне». Объект является экземпляром класса.



Обозначение класса UML

Класс представляет концепцию, которая инкапсулирует состояние (**атрибуты**) и поведение (**операции** или **методы**). Каждый атрибут имеет тип. Каждая операция имеет **подпись** (**сигнатуру**). Имя класса является единственной обязательной информацией.



Имя класса:

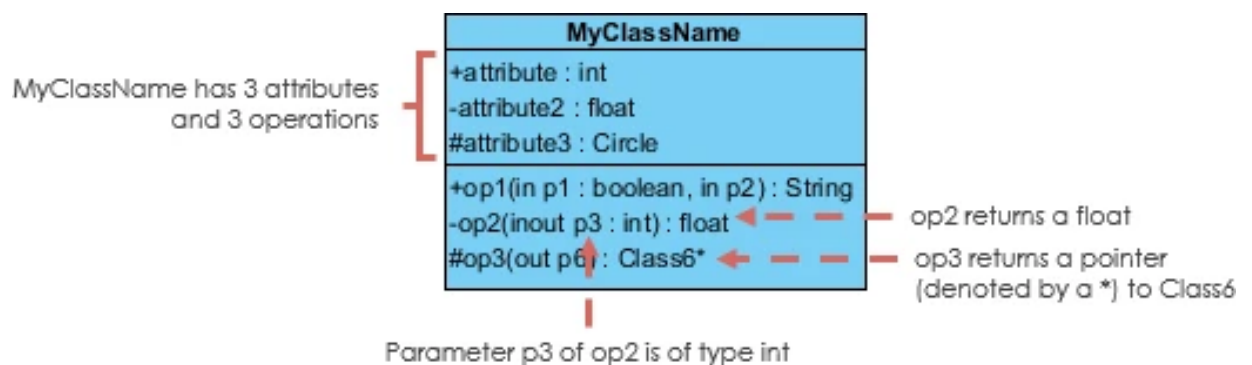
- Имя класса появляется в первом разделе.

Атрибуты класса:

- Атрибуты показаны во втором разделе.
- Тип атрибута отображается после двоеточия.
- Атрибуты сопоставляются с переменными-членами (элементами данных) в коде.

Классовые операции (методы):

- Операции показаны в третьем разделе. Это услуги, предоставляемые классом.
- Тип возвращаемого значения метода отображается после двоеточия в конце подписи метода.
- Возвращаемый тип параметров метода отображается после двоеточия после имени параметра. Операции сопоставляются с методами класса в коде.



Видимость атрибутов и операций класса

В объектно-ориентированном дизайне есть нотация видимости атрибутов и операций. UML определяет четыре типа видимости: **public**, **protected**, **private** и **package**.

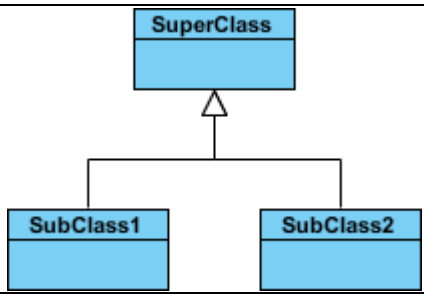
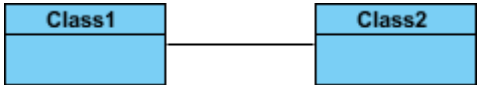
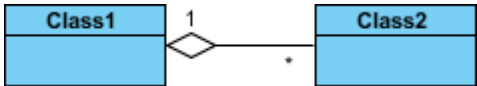
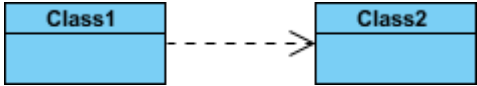
Символы +, -, # и ~ перед именем атрибута и операции в классе обозначают видимость атрибута и операции.

- + обозначает общедоступные атрибуты или операции
- - обозначает частные атрибуты или операции
- # обозначает защищенные атрибуты или операции
- ~ обозначает атрибуты пакета или операции

Примечание. В языке PascalABC.Net тип видимости **package** обозначается словом **internal**.

Классовые отношения

Класс может быть вовлечен в одно или несколько отношений с другими классами.

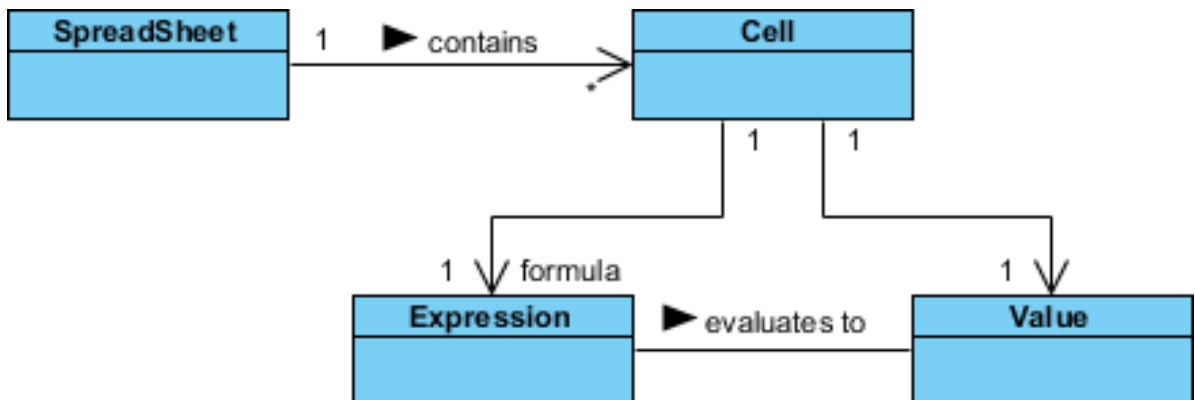
Тип отношений	Графическое представление
Наследование (или обобщение): - Представляет отношение «есть», «является». - Имя абстрактного класса обозначается курсивом. - SubClass1 и SubClass2 являются специализациями SuperClass. - Сплошная линия с полой стрелкой указывает от дочернего к родительскому классу.	
Простая ассоциация: - Структурная связь между двумя одноранговыми классами. - Существует связь между классом 1 и классом 2. - Сплошная линия, соединяющая два класса.	
Агрегация — особый тип ассоциации. Он представляет собой отношение «является частью», «целое - часть», «контейнер - содержимое». - Class2 является частью Class1. - Многие экземпляры (обозначенные *) класса 2 могут быть связаны с классом 1. - Объекты Class1 и Class2 имеют разное время жизни. - Сплошная линия с незакрашенным ромбом на конце ассоциации, связанным с составным классом (агрегатом).	
Композиция — особый тип агрегации, при котором части разрушаются при разрушении целого. - Объекты класса 2 живут и умирают вместе с классом 1. - Класс 2 не может существовать сам по себе. - Сплошная линия с закрашенным ромбом в ассоциации, связанной с составным классом (композицией).	
Зависимость: - Существует между двумя классами, если изменения в определении одного могут вызвать изменения в другом (но не наоборот). - Class1 зависит от Class2 - Пунктирная линия с открытой стрелкой	

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.

Имена отношений

- Имена отношений пишутся посередине ассоциативной строки.
- Названия хороших отношений имеют смысл, если прочитать их вслух: «каждая электронная таблица (SpreadSheet) **содержит (contains)** некоторое количество ячеек (Cell)», «выражение (Expression) **вычисляется в (evaluates to)** значение (Value)».
- Они часто имеют небольшую стрелку, указывающую направление, в котором следует читать отношения. Например, выражения вычисляются в значения, но значения не вычисляются в выражения.



Отношения — Роли

- Роль – это направленная цель ассоциации.
- Роли написаны на концах ассоциативной линии и описывают цель, которую играет этот класс в отношениях.

Например (см. рисунок выше), ячейка (Cell) связана с выражением (Expression). Характер связи таков, что выражение является **формулой (formula)** ячейки.

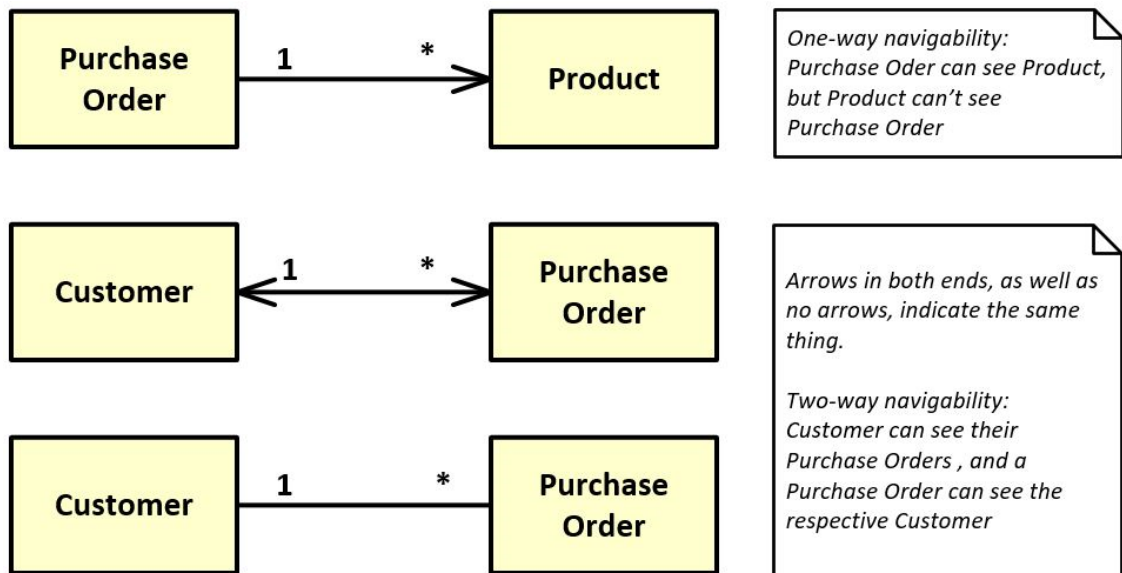
Отношения — Возможность навигации

Стрелки указывают, возможно ли при наличии одного экземпляра, участвующего в отношении, определить связанные с ним экземпляры другого класса.

Диаграмма выше предполагает, что,

- Имея электронную таблицу, мы можем найти все содержащиеся в ней ячейки, но мы не можем определить по ячейке, в какой электронной таблице она содержится.
- Имея ячейку, мы можем получить соответствующее выражение и значение, но имея значение (или выражение), мы не можем найти ячейку, к которой они относятся.

Стрелки в обе стороны, а также отсутствие стрелок обозначают возможность навигации в обе стороны.



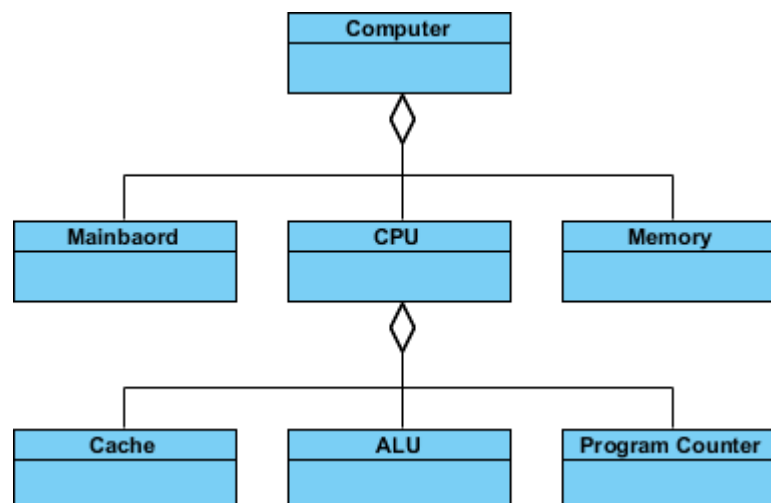
Множественность (мощность отношений, кратность)

Сколько объектов каждого класса принимают участие в отношениях и множественности можно выразить как:

- Ровно один — **1**
- Ноль или единица — **0..1**
- Многие — **0..*** или *****
- Один или несколько — **1..***
- Точное число — например, **3..4** или **6**
- Или сложное отношение — например, **0..1**, **3..4**, **6..*** будет означать любое количество объектов, кроме 2 или 5.

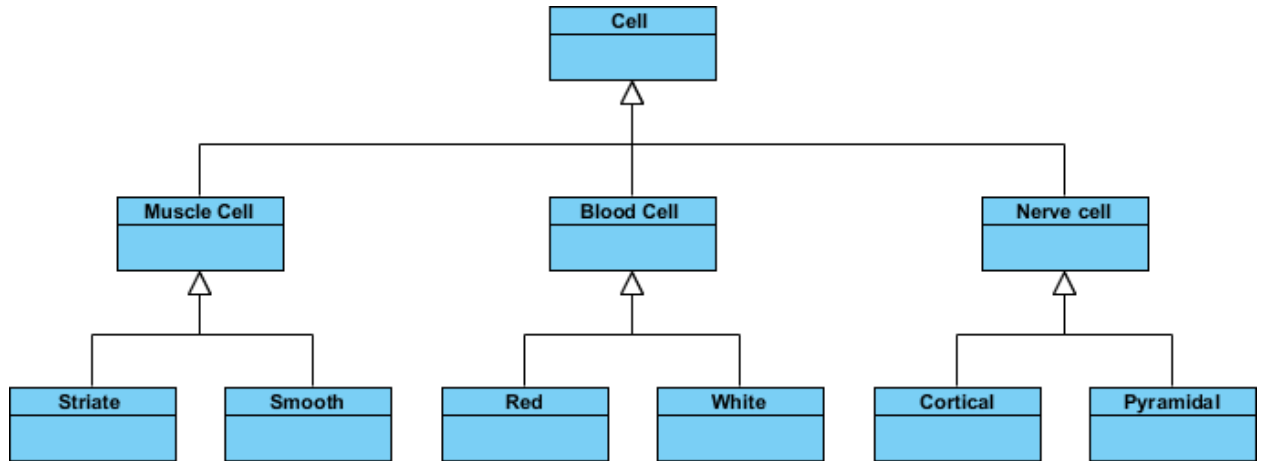
Пример агрегации — компьютер и комплектующие

- Агрегация — это частный случай ассоциации, обозначающий иерархию «состоит из».
- Агрегат — это родительский класс, компоненты — это дочерние классы.



Пример наследования — таксономия (система классификации) клеток

- Наследование — еще один частный случай ассоциации, обозначающей «своего рода» иерархию.
- Наследование упрощает модель анализа, вводя таксономию.
- Дочерние классы наследуют атрибуты и операции родительского класса.

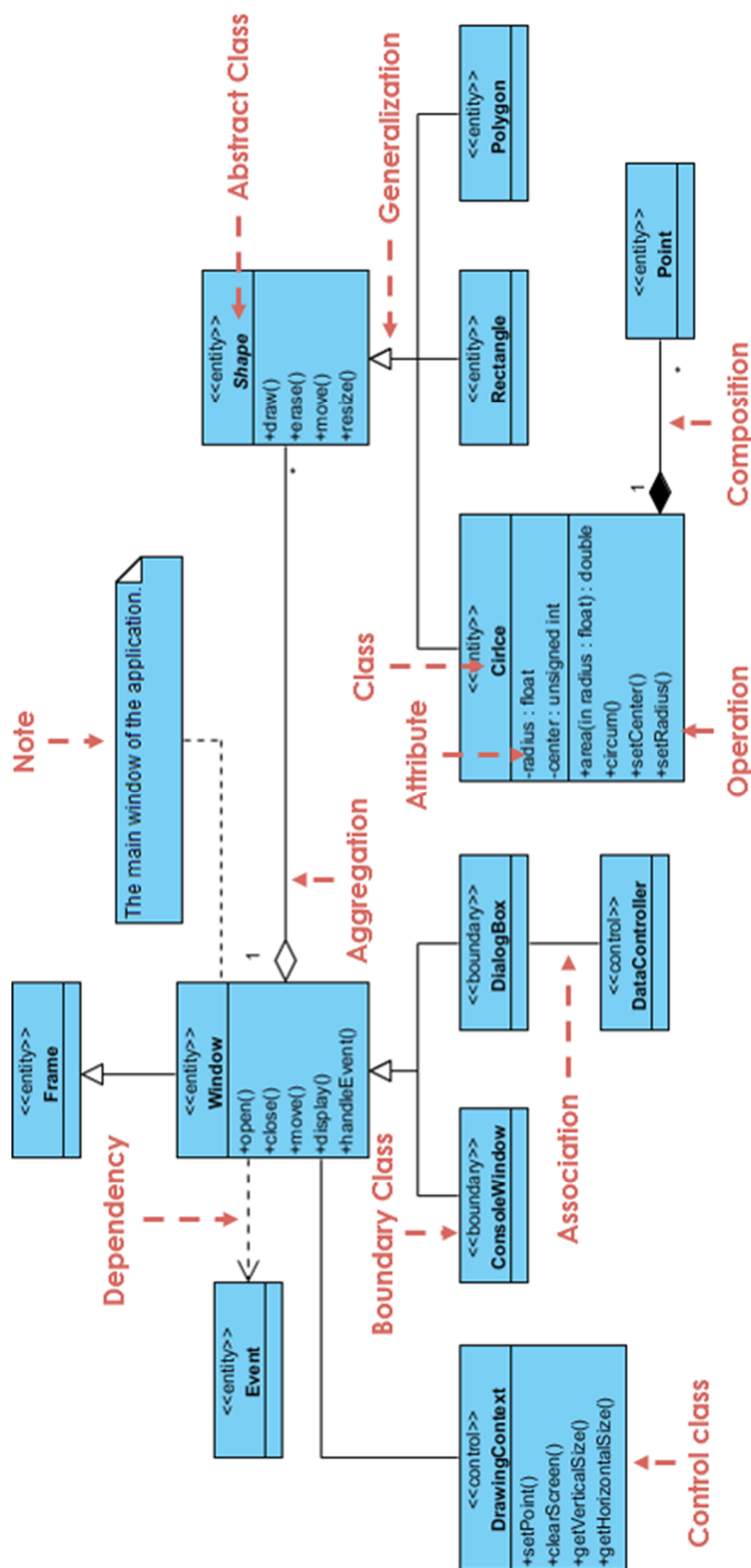


Пример диаграммы классов — графический интерфейс

Диаграмма классов может также иметь примечания, прикрепленные к классам или отношениям.

В приведенном ниже примере:

- Форма — это абстрактный класс. Он показан курсивом.
- Форма — это суперкласс. Круг, прямоугольник и многоугольник являются производными от формы. Другими словами, Круг — это Форма. Это отношение обобщения/наследования.
- Существует связь между DialogBox и DataController.
- Форма является частью окна. Это агрегационные отношения. Shape может существовать без Window.
- Точка является частью Круга. Это композиционные отношения. Точка не может существовать без Окружности.
- Окно зависит от события. Однако Event не зависит от Window.
- Атрибутами Circle являются радиус и центр. Это класс сущности.
- Имена методов Circle: area(), circ(), setCenter() и setRadius().
- Радиус параметра в Circle является параметром in типа float.
- Метод area() класса Circle возвращает значение типа double.
- Атрибуты и имена методов Rectangle скрыты. У некоторых других классов на диаграмме также скрыты атрибуты и имена методов.



Пример диаграммы классов — система заказов

