



Диаграммы UML

Диаграмма классов *(Class Diagram)*

ДИАГРАММЫ КЛАССОВ АНАЛИЗА

Назначение и состав диаграммы классов анализа

Фундаментальными понятиями объектно-ориентированного подхода являются понятия объекта и класса, которые представляются абстракциями реальной или воображаемой сущности (набора сущностей). Класс анализа – еще более абстрактная сущность, чем просто класс, представляет собой набор из одного или более классов. Таким образом, **класс анализа** – это укрупненная абстракция, которая на концептуальном уровне (без точного определения атрибутов и операций) описывает некоторый фрагмент системы.

Существует три вида классов анализа:

- граничный;
- управляющий;
- сущности.

Диаграмма классов анализа по существу является прообразом классической диаграммы классов. Элементами, отображаемыми на диаграмме, являются классы и отношения между ними. При этом для отображения классов можно воспользоваться стандартным обозначением класса (прямоугольник) с указанием внутри него соответствующего стереотипа или значком-стереотипом.

Варианты отображения	Граничный класс	Управляющий класс	Класс сущности
Графический стереотип	 Диалоговое окно «Нормативы»	 Расчет Удол	 План
Стандартное обозначение со строкой-стереотипом	«boundary» Диалоговое окно «Нормативы»	«control» Расчет Удол	«entity» План

Таблица 1. Варианты отображения классов анализа

Назначение классов анализа:

- **границный класс** – используется для моделирования взаимодействия между системой и актерами (пользователями, внешними системами или устройствами). Взаимодействие часто включает в себя получение или передачу информации, запросы на предоставление услуг и т. д. Границные классы являются абстракциями диалоговых окон, форм, панелей, коммуникационных интерфейсов, интерфейсов периферийных устройств, интерфейсов API (англ. application program interface – интерфейс прикладных программ) и т. д. Каждый границный класс должен быть связан как минимум с одним актером;
- **управляющий класс** – отвечает за координацию, взаимодействие и управление другими объектами, выполняет сложные вычисления, управляет безопасностью, транзакциями и т. п.
- **класс сущности** – используется для моделирования долгоживущей, нередко сохраняемой информации. Классы сущности являются абстракциями основных понятий предметной области – людей, объектов, документов и т. д., как правило, хранимых в табличном или ином виде.

Рассмотренное разбиение классов анализа на три группы согласуется с популярным шаблоном проектирования «Модель-Представление-Контроллер» (англ. MVC, Model-View-Controller). При этом под классом сущности - понимается модель, граничным классом - представление и управляющим классом - контроллер.

Связи между классами анализа отображаются с использованием **отношений** пяти видов:

- ассоциаций;
- агрегаций;
- композиций;
- обобщения;
- зависимостей.

Отношение ассоциации применительно к диаграмме классов анализа показывает, что объекты одного класса содержат информацию о существовании (наличии в памяти) объектов другого класса и между ними имеется некоторая логическая или семантическая связь. Если ассоциация указана сплошной линией без стрелок или в виде двунаправленной стрелки, то объекты одного класса будут содержать ссылку на объекты другого и наоборот. Если ассоциация указана односторонней стрелкой, то ссылка будет содержаться только в объектах класса, из которого исходит стрелка. Отношение ассоциации может указываться между классами анализа как одного, так и разных типов.

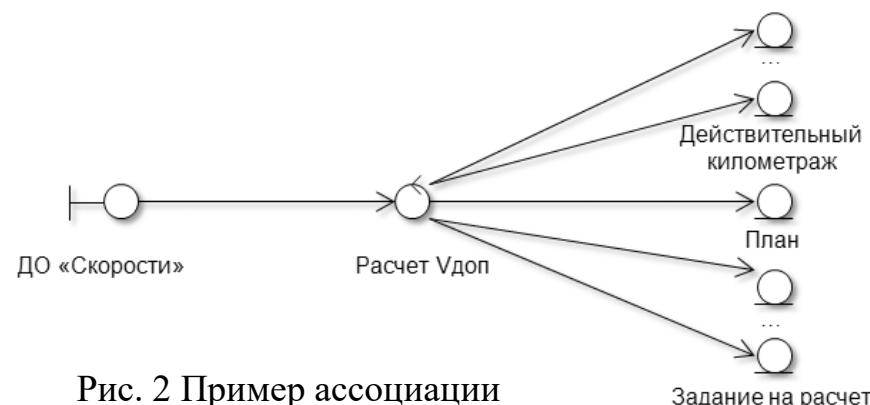


Рис. 2 Пример ассоциации



Отношение агрегации указывает на отношение «часть–целое» и отображается сплошной линией с незакрашенным ромбиком со стороны «целого». Данное отношение, как и ассоциация, означает, что «объект–целое» содержит ссылку на «объект–часть». «Объект–часть» также может содержать ссылку на «объект–целое». Агрегация может указываться только между классами одного типа.

Отношение композиции аналогично агрегации, в которой «части» не могут существовать отдельно от «целого». Применительно к классам (объектам) это означает, что при уничтожении «объекта–целого» должны быть уничтожены все связанные с ним «объекты–части». При этом допускается создание «объектов–частей» намного позже или уничтожение намного ранее «объекта–целого». Применительно к диалоговому окну «Нормативы» фрагмент диаграммы классов анализа может выглядеть следующим образом.

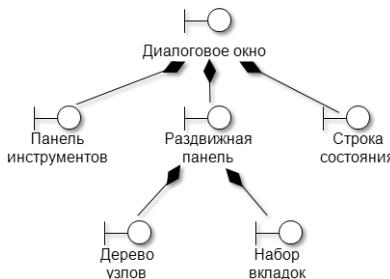


Рис.4 Пример композиции

Отношение обобщения является обычным таксонометрическим отношением между более общим (абстрактным) классом (родителем или предком) и его частным случаем (дочерним классом или потомком). Графически данное отношение обозначается сплошной линией со стрелкой, в виде незакрашенного треугольника, от потомка к родителю. Отношение обобщения может быть только между классами одного вида.



Рис. 5 Пример обобщения

Отношение зависимости применительно к диаграмме классов анализа означает, что в спецификации или теле методов объектов одного класса (зависимого) выполняется обращение к атрибутам, методам или непосредственно к объектам другого класса (независимого). Графически данное отношение обозначается штриховой стрелкой от зависимого класса к независимому. Данное отношение может указываться между классами анализа как одного, так и разных типов. Например, в теле метода инициализации `init()` класса `IskraPUT` выполняется обращение к методу `getProperties()`, определенному в классе `System` ичитывающему системные параметры компьютера.



Рис. 6. Пример зависимости

Правила и рекомендации по разработке диаграмм классов анализа

При разработке диаграммы следует придерживаться следующих правил и рекомендаций.

1. При выделении классов анализа следует учитывать тот факт, что они являются обобщенными (уточненными) сущностями, которые в дальнейшем подлежат уточнению и возможному разбиению на несколько более мелких классов. Можно провести аналогию между ними и накопителями данных на DFD, которые, как правило, являются крупными концептуальными объектами (совокупностями тесно взаимосвязанных таблиц), подлежащими уточнению на стадии информационного проектирования с помощью одной из методологий ERD.

2. Для выделения классов сущностей необходимо определить все реальные либо воображаемые объекты, имеющие существенное значение для рассматриваемой предметной области, информация о которых подлежит хранению. При этом из спецификаций вариантов использования следует выделить все объекты, которые могут существовать независимо от других. Например, объект «билет» является независимой сущностью, потому что любой билет существует независимо от того, знаем мы его номер, стоимость или нет. Т. е. при выделении классов-сущностей действуют те же правила, что при построении концептуальной модели БД.

3. Для каждого актера следует предусмотреть, как минимум, один границный класс в целях организации интерфейса между ним и системой. Аналогично для каждого класса сущности, как правило, должен быть границный класс – ведь по каждому объекту класса сущности должна быть предусмотрена возможность просмотра, ввода и/или корректировки информации через определенную форму ввода/вывода или чтения/записи через определенный интерфейс.

4. Для управления, обеспечения взаимодействия и координации работы объектов, реализующих одну из функций системы (обычно, вариант использования), необходимо предусмотреть, как минимум, один управляющий класс. Как правило, взаимодействие между границным классом и классом сущности происходит через управляющий класс.

5. В целях облегчения восприятия специфики связей между классами рекомендуется использовать отношения агрегации, композиции и обобщения.

6. При разработке диаграммы основное внимание должно быть уделено определению и детализации классов сущностей, управляющих и границных классов, обеспечивающих взаимодействие с внешними системами. Границные классы, обеспечивающие взаимодействие с пользователями, не требуют излишней детализации до уровня отдельного поля ввода или ниспадающего списка, так как современные среды программирования обладают богатыми возможностями по быстрому созданию пользовательского интерфейса.

На следующем рисунке показан фрагмент диаграммы классов анализа.

В правой части диаграммы (начиная с класса «Дорога») расположены классы, описывающие структуру БД на сервере. Несмотря на то, что исходный программный код не будет содержать этих классов, их детальная разработка не менее важна, чем разработка остальных классов. В данном случае правая часть диаграммы представляет концептуальную модель БД, которую можно было бы вынести на отдельную диаграмму и описать с помощью одной из методологий ERD.

Во время работы программы информация из БД считывается в объекты класса «Таблица» (кэшируется), обрабатывается объектами других классов и при необходимости записывается в БД. Каждый такой объект, помимо данных конкретной таблицы БД, содержит описание свойств таблицы в целом, полей (имя поля, его тип, значение по умолчанию и т. д.), описание структуры заголовка таблицы, отображаемого на экране, и другую необходимую информацию. За каждым диалоговым окном, предназначенным для работы с БД, закрепляется определенный набор объектов класса «Таблица».

Аналогично классу «Диалоговое окно "Дороги"» в левую часть диаграммы должны быть нанесены классы «Диалоговое окно "Справочники"» и «Диалоговое окно "Нормативы"», а аналогично классу «Диалоговое окно "Скорости"» – класс «Диалоговое окно "Возвышения"», связанный с управляющим классом «Расчет возвышений». Концептуальная модель БД должна быть дополнена таблицами, описывающими нормативно-справочную информацию и остальные данные дорожного уровня, необходимые для работы системы.

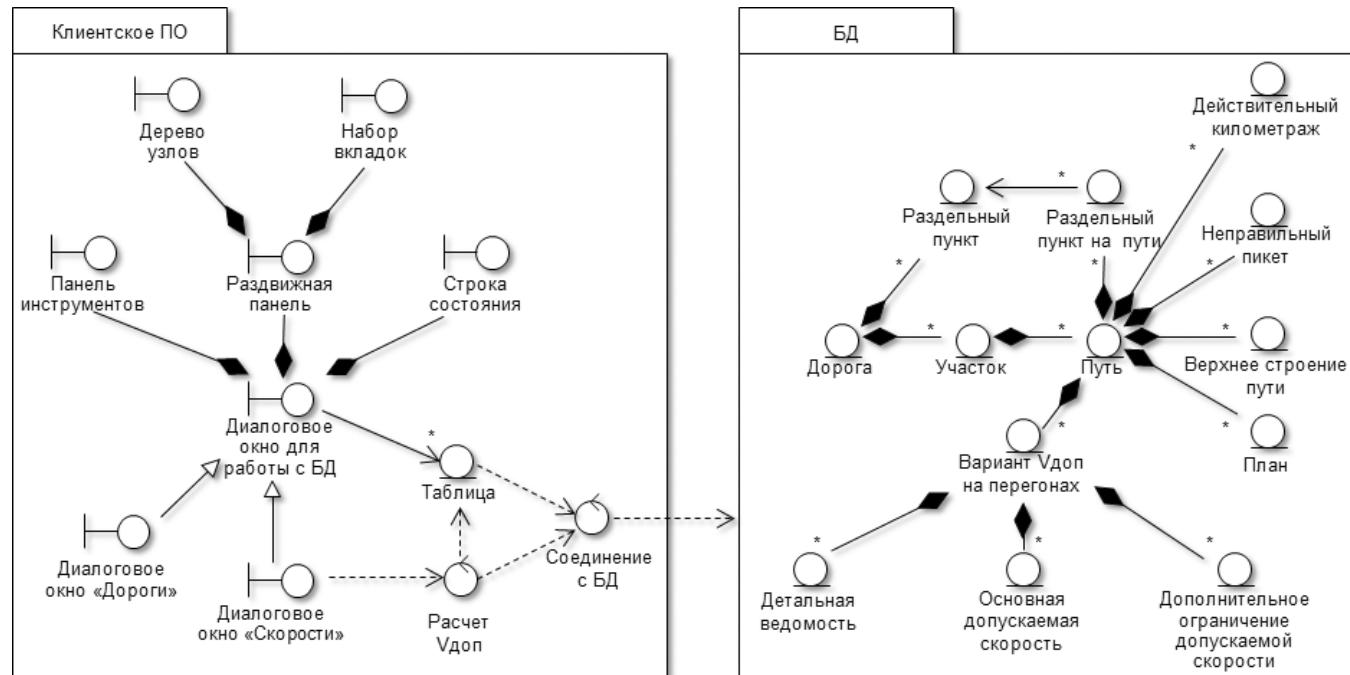


Рис. 7 Фрагмент диаграммы классов анализа

Основные вопросы

- Что такое диаграмма классов
- Компоненты диаграммы классов и их назначение
- Пример диаграммы классов
- Расширение языка UML для построения моделей программного обеспечения и бизнес-систем

Диаграмма классов

- Является центральным звеном объектно-ориентированного подхода
- Содержит информацию об объектах системы и статических связях между объектами
- Отражает *декларативные знания* о предметной области
- Оперирует понятиями *класса, объекта, отношения, пакета*

Класс

- **Класс** – это множество объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов.

Имя_класса



Простейший вид класса состоит только из секции имени

Имя_класса
атрибуты класса



Класс с указанием атрибутов (переменных)

Полное описание класса, состоящее из 3 разделов (секций) – секции имени, секции атрибутов, секции операций



Имя_класса
атрибуты класса
операции класса()

Класс

- *Имя класса* должно быть уникально
- Имя класса должно начинаться с заглавной буквы.
- Класс может не иметь экземпляров или объектов. В этом случае он называется **абстрактным классом**, а для обозначения его имени используется *курсив*

Атрибуты класса

- Атрибут = **свойство**, которое является общим для всех объектов данного класса
- Общий формат записи атрибутов:
<квантор видимости> <имя атрибута> [кратность]: <тип атрибута> = <исходное значение> {строка-свойство}

Атрибуты класса. Квантор видимости

- Квантор видимости может принимать одно из следующих значений: **+, #, - , ~**.
- «**+**» - атрибут с областью видимости типа **общедоступный (public)**.
- «**#**» - атрибут с областью видимости типа **защищенный (protected)**.
- «**-**» - атрибут с областью видимости типа **закрытый (private)**.
- «**~**» - атрибут с областью видимости типа **пакетный (package)**.

Атрибуты класса.

Имя атрибута

- Представлено в виде 的独特的 строки текста
- Имя атрибута является единственным 必需的 элементом в синтаксическом обозначении атрибута
- Должно начинаться со 小写 буквы
- По практическим соображениям записывается без пробелов

Атрибуты класса. Кратность атрибута

- *Кратность атрибута* характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса.
- Формат: [нижняя граница . . верхняя граница]
- Примеры: [0..1], [0..*], [1..3,5..7]

Атрибуты класса. Тип атрибута

- Выражение, определяемое некоторым типом данных (например, в зависимости от языка программирования)
- В простейшем случае – *осмысленная строка текста*.
- Пример:
цвет: Color
имяСотрудника[1..2]: String;
видимость: Boolean

Атрибуты класса. Исходное значение

- Служит для задания некоторого начального значения в момент создания отдельного экземпляра класса
- Пример:

цвет: Color = (255, 0, 0)

*имяСотрудника[1..2]: String = 'Иван
Иванов';*

видимость: Boolean = истина

Атрибуты класса. Строка-свойство

- Служит для указания **дополнительных свойств атрибута**, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения соответствующей программы.
- Это значение принимается за **исходное значение атрибута**, которое не может быть изменено в дальнейшем.
- Пример:
заработкаPlата: Currency = \$500 {frozen}

Операции класса

- Представляют собой некоторый сервис, который предоставляет каждый экземпляр класса или объект по требованию своих клиентов.
- Правила записи операций:
<квантор видимости> <имя операции> (список параметров): <выражение типа возвращаемого значения> {строка-свойство}

Операции класса. Список параметров

- Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:

*<вид параметра> <имя параметра> :
<выражение типа> = <значение
параметра по умолчанию>*

Операции класса. Строка-свойство

- **Строка-свойство** служит для указания значений свойств, которые могут быть применены к данной операции.
- Например, для указания последовательности действий будет использована строка-свойство вида:

{concurrency = имя} ,

где **имя** может принимать одно из следующих значений:

- **sequential** (последовательная),
- **concurrent** (параллельная),
- **guarded** (охраняемая)

Операции класса. Примеры

- +нарисовать (форма : Многоугольник = прямоугольник, цветЗаливки : Color = (0, 0, 255));
- -изменитьСчетКлиента (номерСчета : Integer) : Currency;
- #выдатьСообщение() : ('Ошибка деления на ноль').

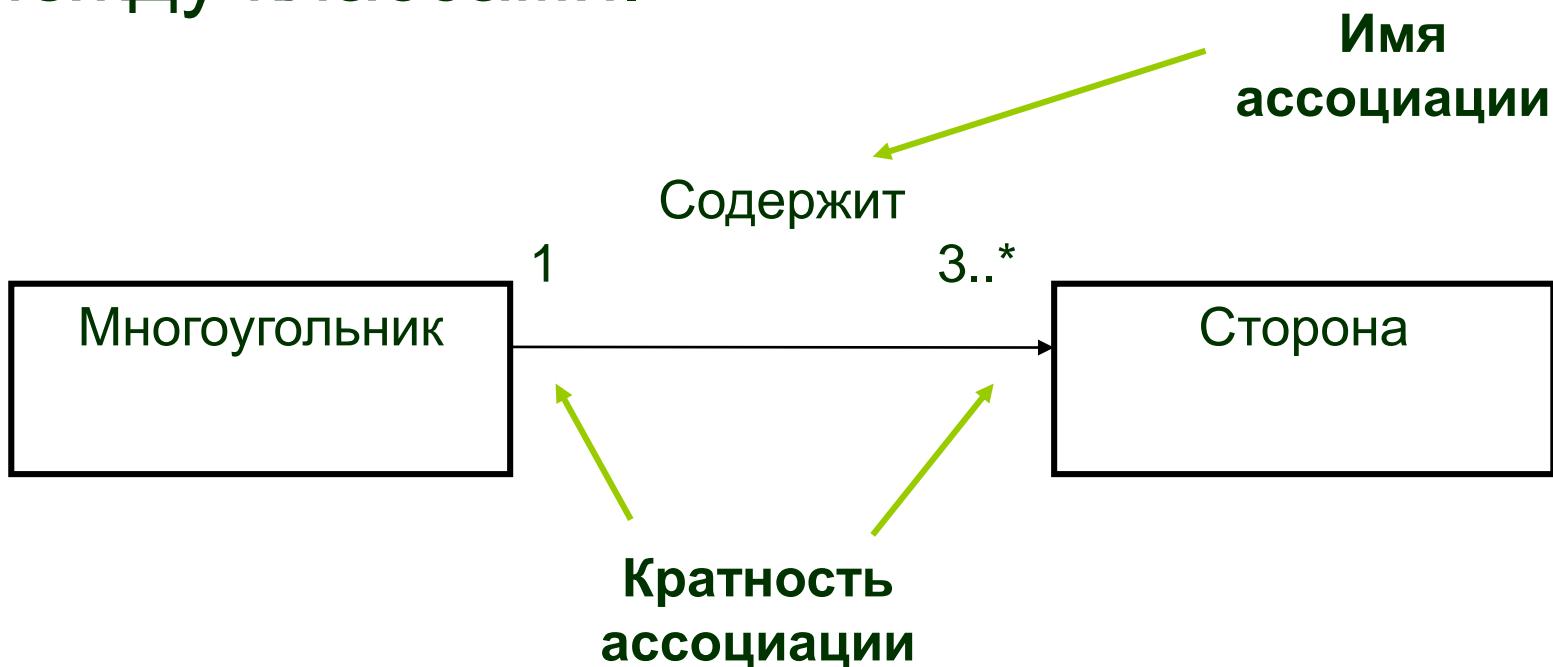
Отношения между классами

Базовыми отношениями на диаграмме классов являются:

- отношения **ассоциации** (*association*);
- отношения **обобщения** (*generalization*);
- отношения **агрегации** (*aggregation*);
- отношения **композиции** (*composition*);
- отношения **зависимости** (*dependency*).

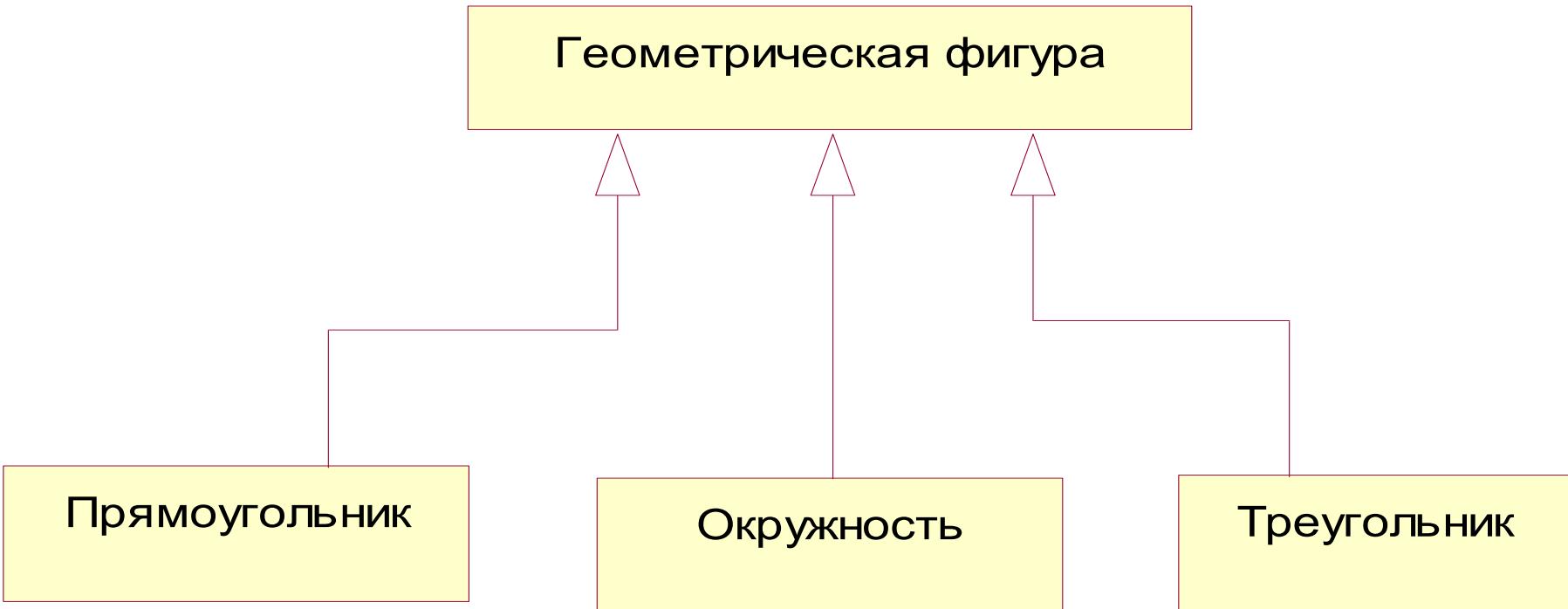
Отношение ассоциации

- Отношение ассоциации свидетельствует о наличии произвольного отношения между классами.



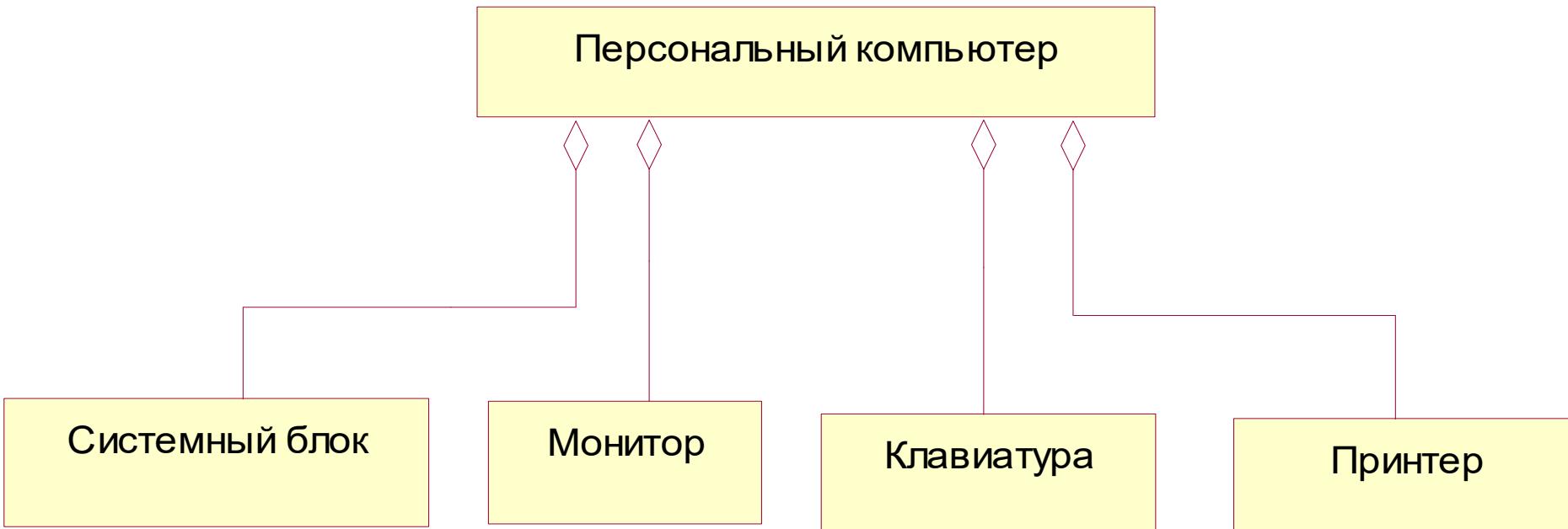
Отношение обобщения

- Является отношением классификации между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком)



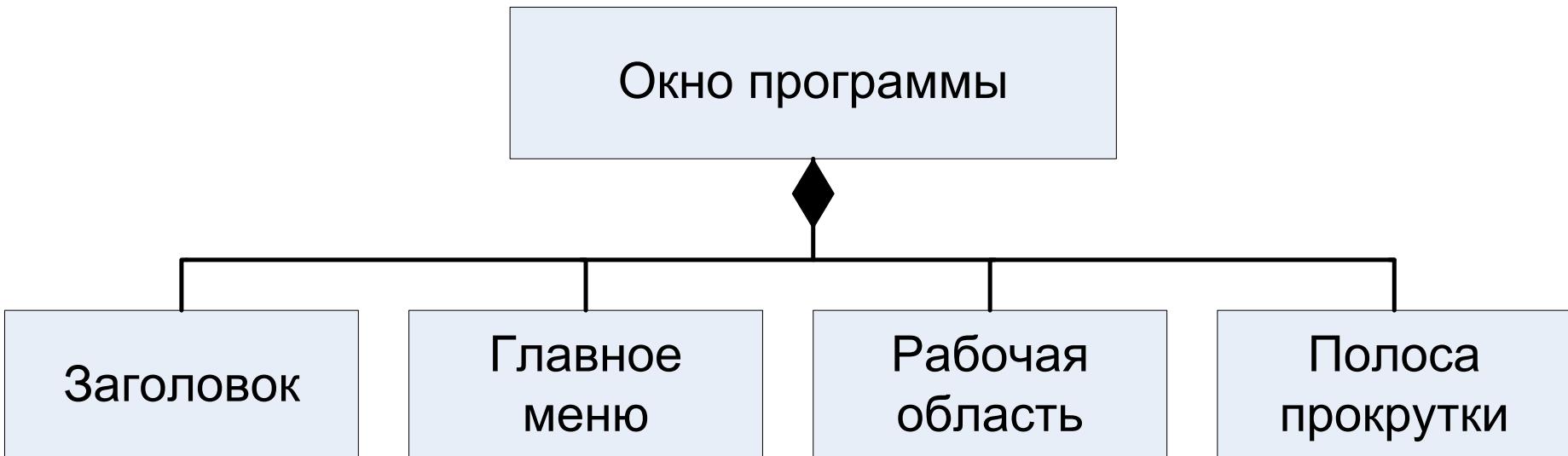
Отношение агрегации

- Смысл: один из классов представляет собой некоторую сущность, которая **включает** в себя в качестве составных частей другие сущности.
- Применяется для представления системных взаимосвязей типа «часть-целое».



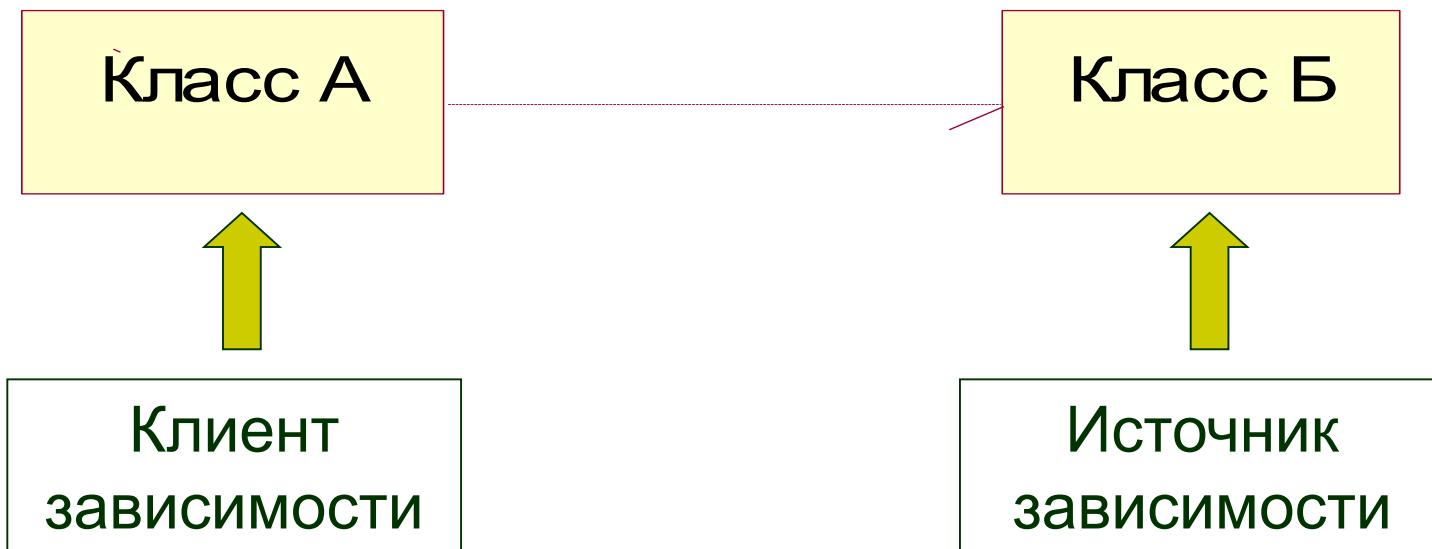
Отношение композиции

- Является частным случаем отношения агрегации.
- Части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются составные части.



Отношение зависимости

- Используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого элемента.



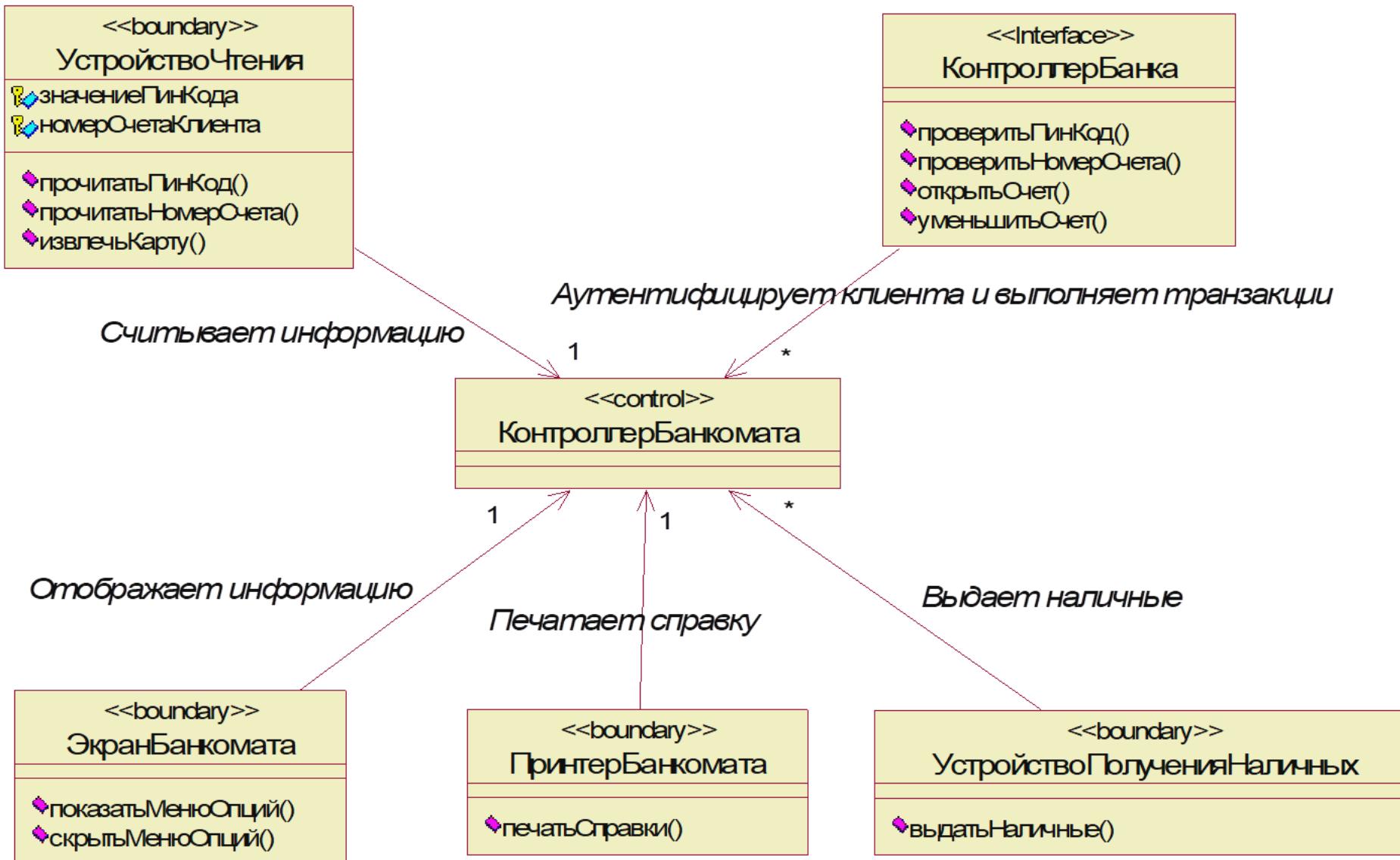
Пакеты

- служат для **группировки** элементов модели
- Любой пакет владеет своими элементами
- любой элемент может принадлежать *только одному пакету*



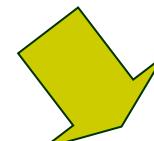
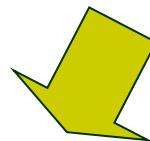
имя пакета

Пример диаграммы классов



Расширения языка UML

Расширения языка
UML

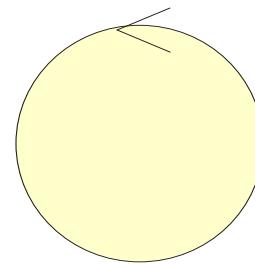


Профиль для процесса
разработки ПО
(The UML Profile for
Software Development)

Профиль для бизнес-
моделирования (The
UML Profile for Business
Modeling)

Профиль для процесса разработки ПО

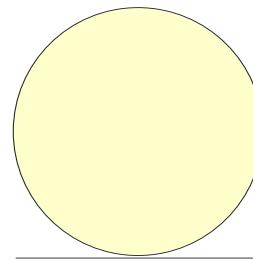
- Управляющий класс (**control**) – отвечает за координацию действий других классов.



NewClass

Профиль для процесса разработки ПО

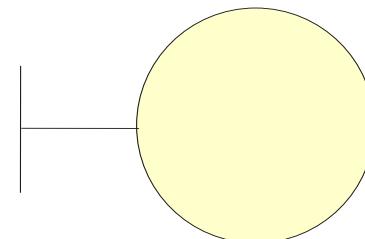
- Класс-сущность (**entity**) содержит информацию, которая должна храниться **постоянно и не уничтожаться** с уничтожением объектов данного класса или прекращением работы моделируемой системы.



NewClass2

Профиль для процесса разработки ПО

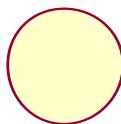
- Границный класс (**boundary**) – располагается на границе системы с внешней средой, но является составной частью системы.



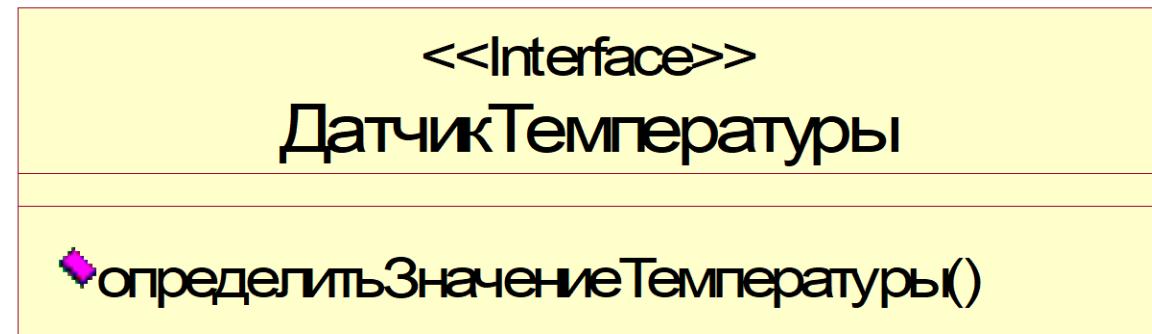
NewClass3

Интерфейс (interface)

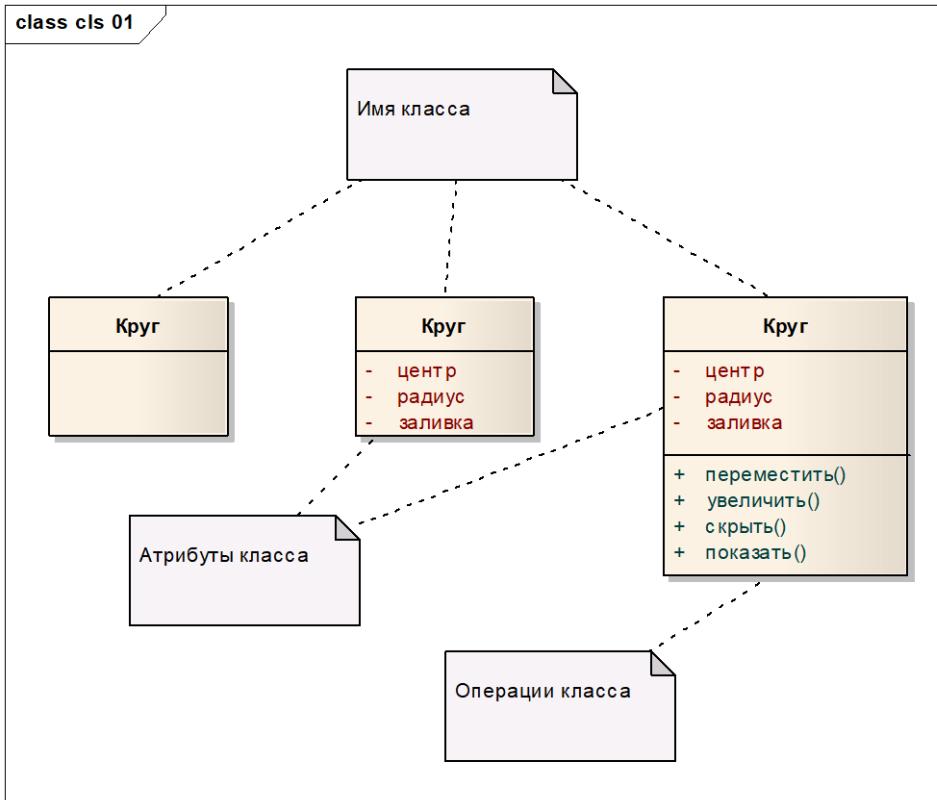
- в контексте языка UML является специальным случаем класса, у которого имеются только операции и отсутствуют атрибуты.



ДатчикТемпературы



Классы



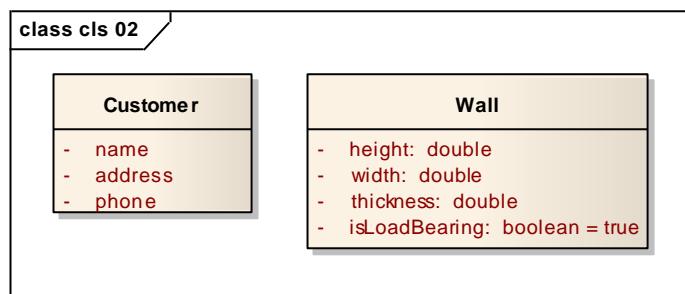
Класс (class) служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции. В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

Имя класса

- **Имя класса** должно быть уникальным в пределах проекта, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Имя класса указывается в верхней секции прямоугольника полужирным шрифтом и должно начинаться с прописной буквы.
- **Абстрактный класс** – это класс, который не может иметь экземпляров (объектов). Для обозначения его имени используется наклонный шрифт.

Атрибуты класса

Атрибут - это именованное свойство класса, общее для всех его объектов, включающее описание множества значений, которые могут принимать экземпляры этого свойства

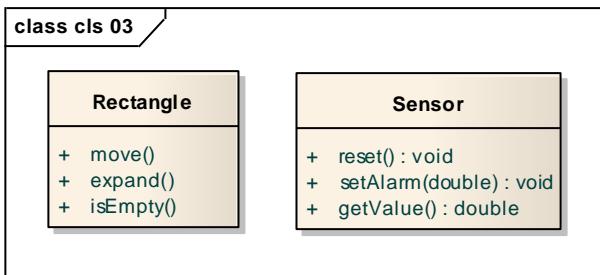


- Класс может иметь любое число атрибутов или не иметь их вовсе
- Атрибут является абстракцией данных объекта или его состояния
- В каждый момент времени любой атрибут объекта, принадлежащего данному классу, обладает вполне определенным значением
- Атрибуты представлены в разделе, который расположен под именем класса

При описании атрибута можно явным образом указывать его тип и значение, принимаемое по умолчанию

Операции класса

Операция – это реализация услуги, которую можно запросить у любого объекта класса для воздействия на поведение.
Операция - это абстракция того, что может делать объект.



- У всех объектов класса имеется общий набор операций
- Класс может содержать любое число операций или не содержать их вовсе
- Как правило (хотя не всегда) обращение к операции объекта изменяет его состояние или его данные
- Операции класса изображаются в разделе, расположеннном ниже раздела с атрибутами. При этом можно ограничиться только именами.

Изображая класс в UML, придерживайтесь следующих правил:

- показывайте только те свойства класса, которые важны для понимания абстракции в данном контексте
- разделяйте длинные списки атрибутов и операций на группы в соответствии с их категориями
- показывайте взаимосвязанные классы на одной и той же диаграмме

Отношения между классами

Помимо внутреннего устройства классов на диаграмме классов указываются различные отношения (связи) между ними. Совокупность типов отношений фиксирована.

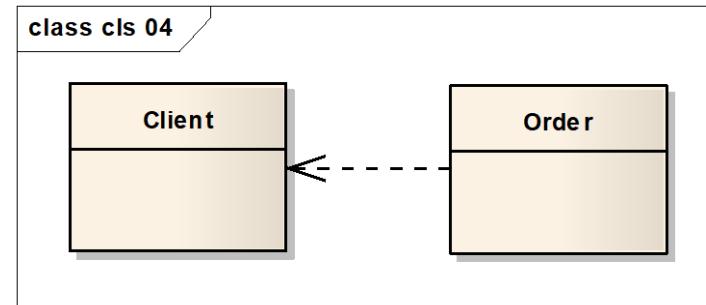
Базовые отношениями в языке UML:

- Отношение зависимости (dependency relationship)
- *Отношение ассоциации (association relationship)*
- Отношение обобщения (generalization relationship)

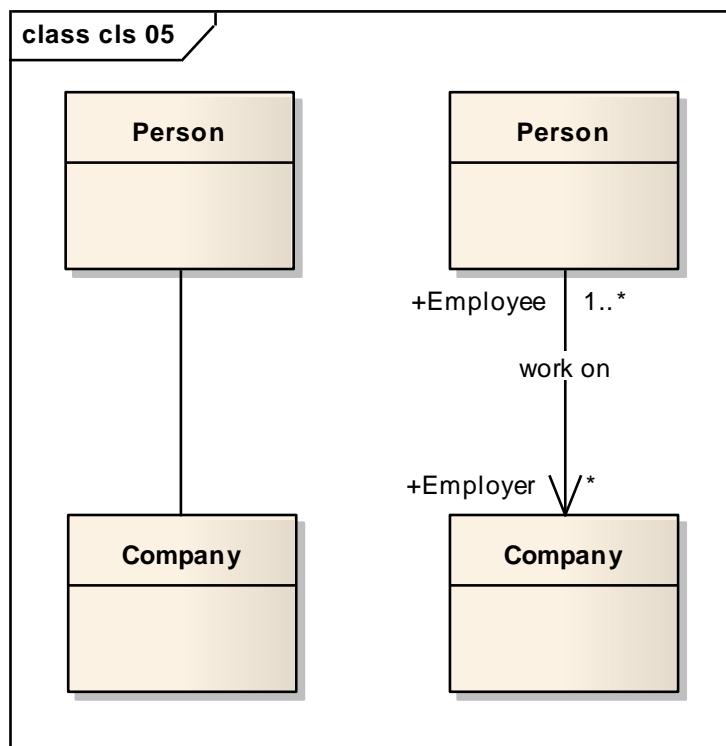
Отношение зависимости

Зависимость (dependency) - отношение использования, согласно которому изменение в спецификации одного элемента может повлиять на другой элемент, его использующий, причем обратное не обязательно.

Графически зависимость изображается пунктирной линией со стрелкой, направленной от зависимого элемента на тот, от которого он зависит.



Отношение ассоциации

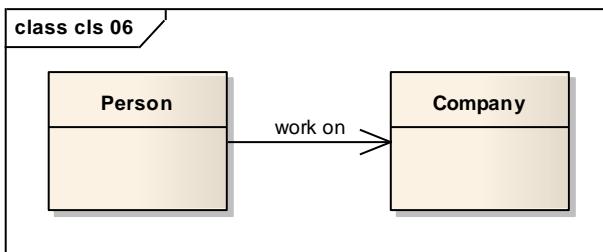


Ассоциация (association) - структурное отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа.

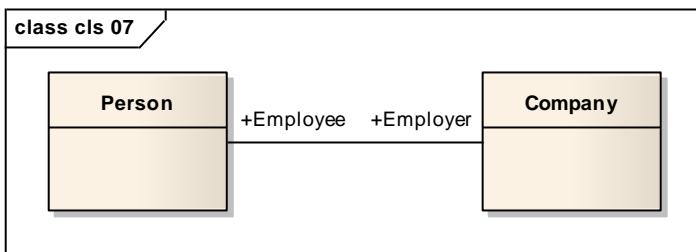
Ассоциация, связывающая два класса, называется бинарной. Можно создавать ассоциации, связывающие сразу несколько классов; они называются *n*-арными.

У ассоциаций могут быть: имя, роль, кратность и агрегирование.

Отношение ассоциации



Имя описывает природу отношения. Чтобы избежать возможных двусмысленностей в понимании имени, указывается направление, в котором оно должно читаться.

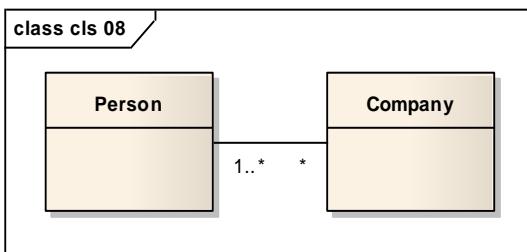


Роль – это «лицо», которым класс, находящийся на одной стороне ассоциации, обращен к классу с другой ее стороны. Один класс может играть в разных ассоциациях как одну и ту же роль, так и различные.

Кратность - количество объектов, которое может быть связано посредством одного экземпляра ассоциации.

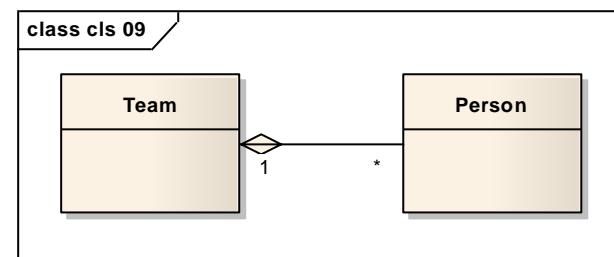
Кратность записывается либо как выражение, значением которого является диапазон значений, либо в явном виде.

Кратность можно задать равной единице (1), можно указать диапазон: «ноль или единица» (0..1), «много» (0..*), «единица или больше» (1..*), «любое число объектов, кроме 2 и 5» (0 .. 1, 3..4, 6..*).

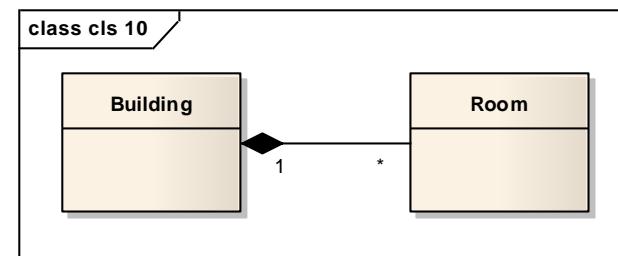


Отношение ассоциации

Агрегирование (aggregation) показывает отношение типа «часть/целое», в котором один класс состоит из нескольких меньших классов



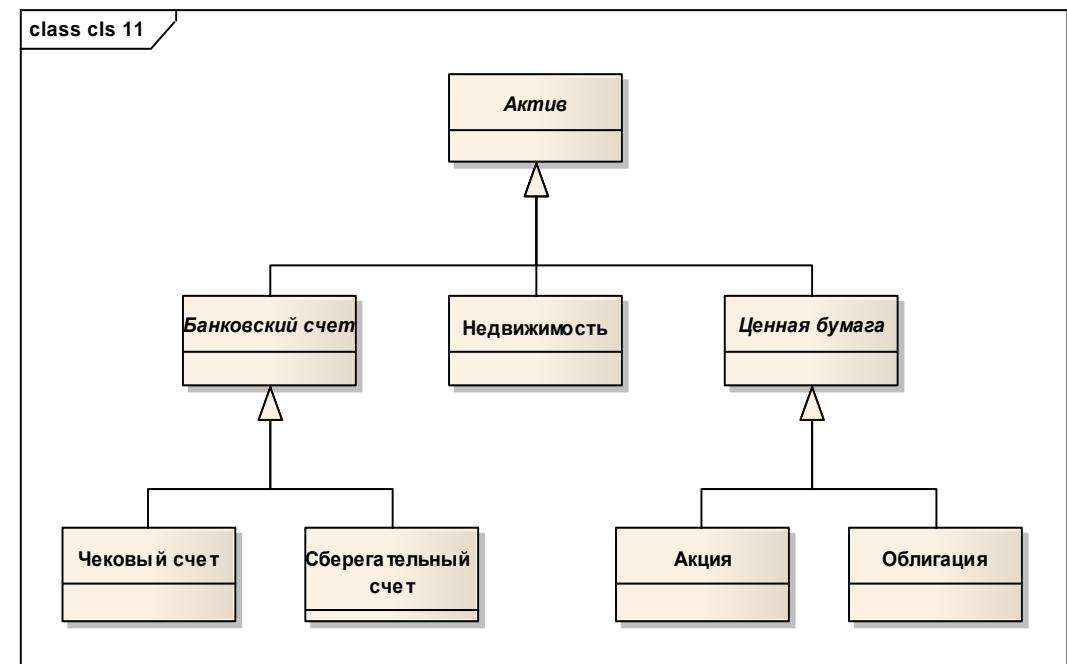
Композиция (composition) служит для выделения специальной формы отношения «часть/целое». Специфика связи заключается в том, что части не могут выступать в отрыве от целого, то есть с уничтожением целого уничтожаются и все его составные части.



Отношение обобщения

Обобщение (generalization) - это отношение между общей сущностью (суперклассом, или родителем) и ее конкретным воплощением (подклассом, или потомком).

Данное отношение описывает иерархию классов и наследование их свойств и поведения. Предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.



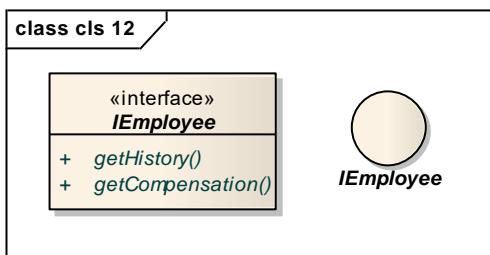
Советы по отношениям между классами

При моделировании отношений в UML соблюдайте следующие правила:

- используйте зависимость, только если моделируемое отношение не является структурным
- используйте обобщение, только если имеет место отношение типа «является»
- избегайте множественного наследования
- иерархия наследования не должна быть ни слишком глубокой (желательно не более пяти уровней), ни слишком широкой
- применяйте ассоциации прежде всего там, где между объектами существуют структурные отношения

Интерфейсы

Интерфейс (interface) - набор операций, используемый для спецификации услуг, предоставляемых классом или компонентом.

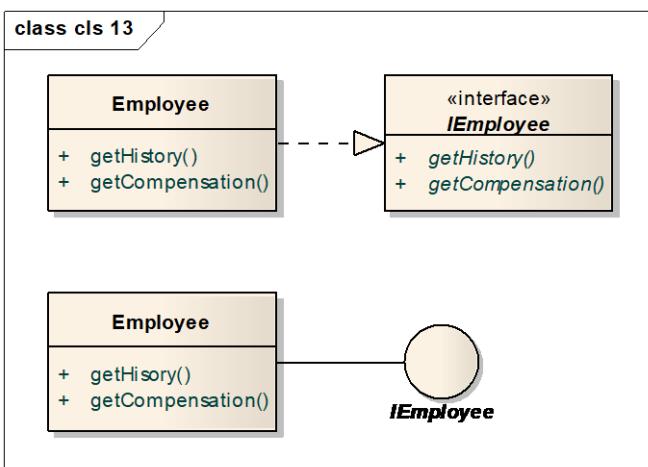


Интерфейс должен иметь уникальное имя.

Интерфейс может включать любое число операций.

Особенности:

- Интерфейсы не содержат атрибутов
- Интерфейсы не содержат реализующих операции методов

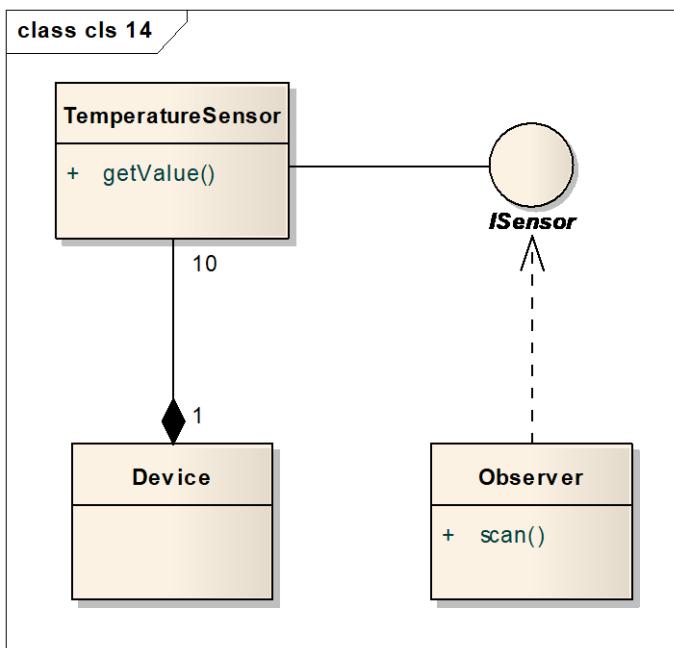


Интерфейс специфицирует контракт класса, но не накладывает никаких ограничений на свою реализацию.

Связь интерфейса с реализующим его элементом можно графически представить двумя способами:

- интерфейс представляют в виде стереотипного класса и связывают его с классом **отношением реализации**. Отношение реализации объединяет отношения обобщения и зависимости
- отношение между интерфейсом и его реализацией изображается кружочком с одной стороны класса.

Интерфейсы



Интерфейсы могут принимать участие в отношениях обобщения, ассоциации и зависимости.

Интерфейс представляет собой стыковочный узел в системе. Он определяет условия контракта, после чего обе стороны - клиент и поставщик - могут действовать независимо друг от друга, полностью полагаясь на взаимные обязательства.

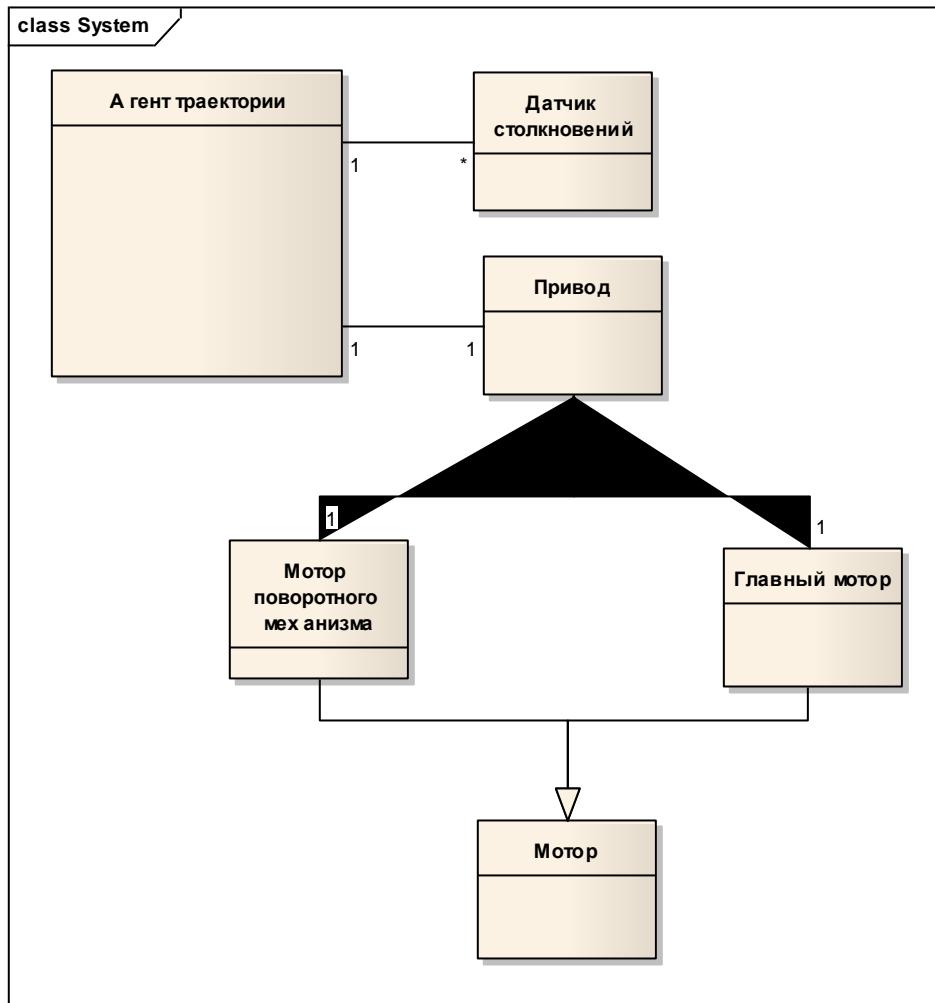


Советы по интерфейсам

Изображая интерфейс на языке UML, руководствуйтесь приведенными ниже правилами:

- используйте сокращенную нотацию, если надо просто показать наличие стыковочного узла в системе
- используйте форму, если надо визуализировать детали самого сервиса

Кооперации

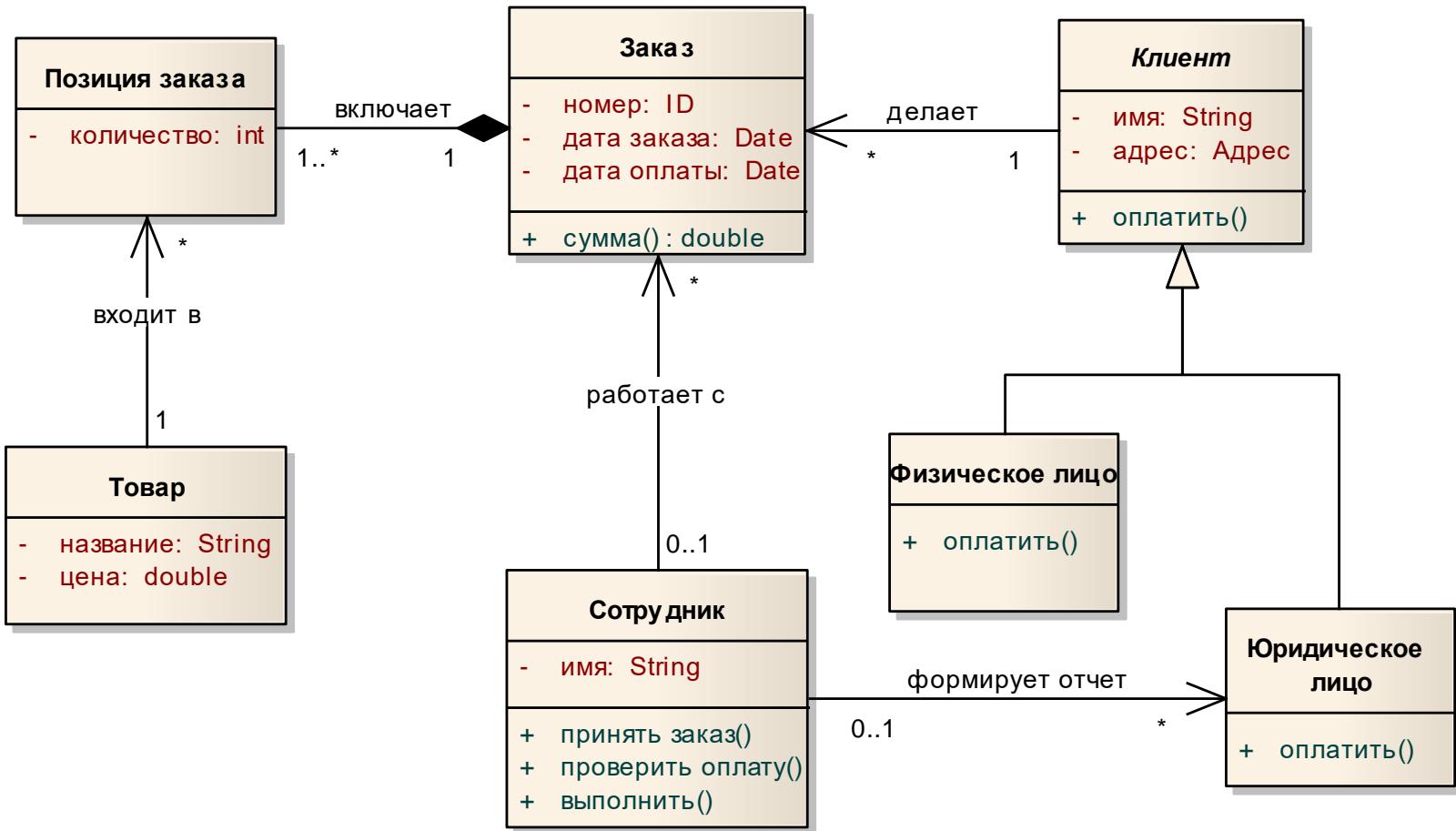


Кооперация (collaboration)

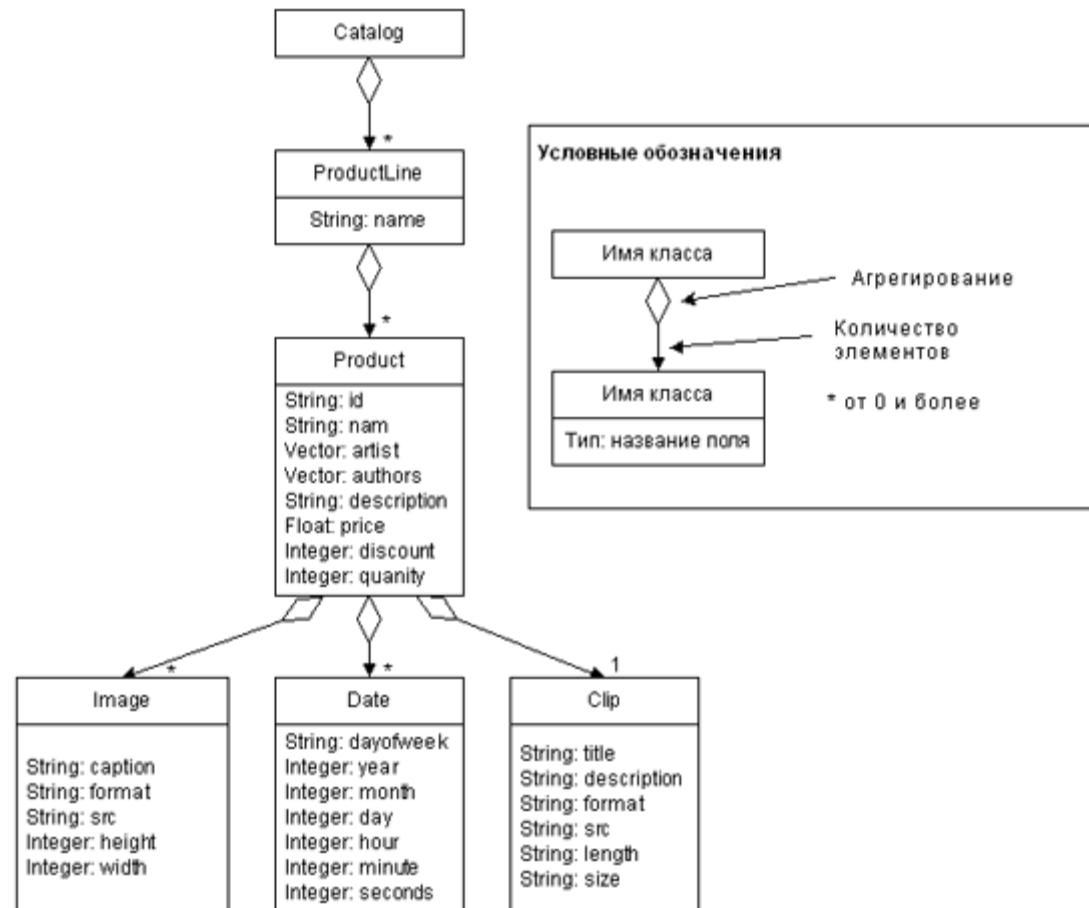
- это сообщество классов, интерфейсов и других элементов, которые работают совместно для обеспечения кооперативного поведения, более значимого, чем сумма его составляющих.

Пример

class cls 16



UML



UML Jokes

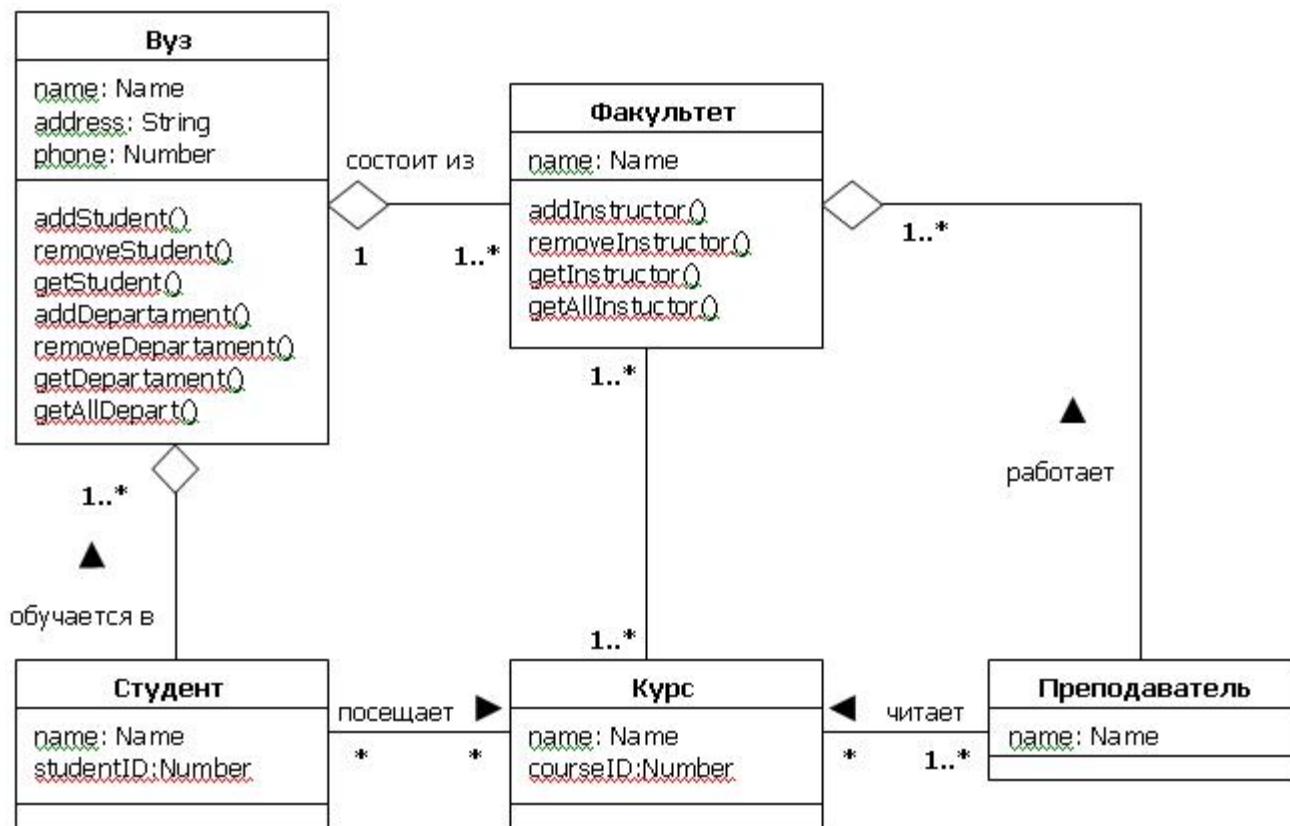


РИС.3.9