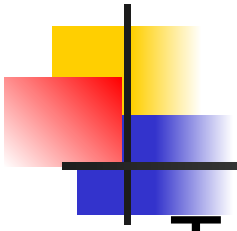
Информационные технологии

- Диаграммы Классов

Диаграммы классов

Унифицированный язык моделирования (UML) – это семейство графических нотаций, в основе которого лежит единая метамодель.

Он помогает в описании и проектировании программных систем, в особенности систем, построенных с использованием объектно-ориентированных технологий.



Диаграммы классов

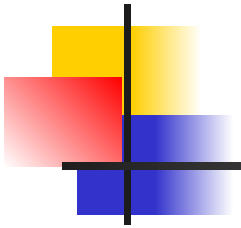
Три режима использования:

- Режим эскиза;
- Режим проектирования;
- Режим языка программирования.

Диаграммы классов

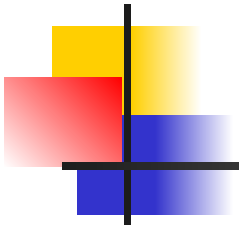
Эскизное моделирование

разработчик делает наброски отдельных элементов программы, которую собирается написать, и обычно обсуждает их с другими разработчиками из команды



Диаграммы классов

UML как средство проектирования нацелен на полноту: работа заключается в построении детальной модели для программиста, который будет выполнять кодирование.



Диаграммы классов

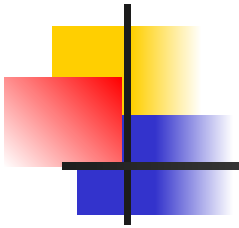
Дизайнер разрабатывает модели проектного уровня в виде интерфейсов подсистем, а затем дает возможность разработчикам работать над реализацией подробностей. В такой модели можно показать все детали класса.

Диаграммы классов

Различия:

Эскизы сознательно выполняются неполными, подчеркивая важную информацию,

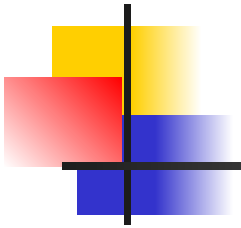
в то время как модели нацелены на полноту.



Диаграммы классов

Режим использования UML в качестве языка программирования:

разработчики рисуют диаграммы, которые компилируются прямо в исполняемый код



Диаграммы классов

Другая точка зрения:

- применением для концептуального моделирования;
- моделирования программного обеспечения;

Диаграммы классов

«Значение моделей в процессе обратной разработки зависит от того, как работает инструментарий.

Если он применяется в качестве динамического броузера, то он может быть очень полезным;

если же он генерирует большой документ, то вся его работа сводится к пустому переводу бумаги»

М.Фаулер

Диаграммы классов

- ***Класс*** (class) — абстрактное описание множества однородных объектов, имеющих одинаковые ***атрибуты***, ***операции*** и отношения с объектами других *классов*.

Диаграммы классов

- ***Класс*** (class) — абстрактное описание множества однородных объектов, имеющих одинаковые ***атрибуты***, ***операции*** и отношения с объектами других *классов*.

Диаграммы классов

- ***Класс*** (class) — абстрактное описание множества однородных объектов, имеющих одинаковые ***атрибуты***, ***операции*** и отношения с объектами других *классов*.

Диаграммы классов

Диаграммы классов

Имя класса

- Должно быть уникальным в пределах пакета
- *имя класса* записывается по центру секции *имени* полужирным шрифтом и должно начинаться с заглавной буквы.
- надо использовать существительные, записанные по практическим соображениям без пробелов
- <Имя пакета>::<*Имя класса*>

Банк::Счет

Диаграммы классов

Класс

- ***Конкретный класс*** (concrete class) — *класс*, на основе которого могут быть непосредственно созданы экземпляры или объекты
- ***Абстрактный класс*** (abstract class) — *класс*, который не имеет экземпляров или объектов

В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом

Атрибуты класса

Атрибут (attribute) — содержательная характеристика *класса*, описывающая множество значений, которые могут принимать отдельные объекты этого *класса*.

<квантор видимости> <имя атрибута> [кратность] :
<тип атрибута> = <исходное значение> {строка-свойство}.

Атрибуты класса

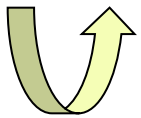
<квантор видимости> -- «видимость» (visibility) — качественная характеристика характеризующая возможность других объектов модели оказывать влияние

+ – общедоступный (**public**).

– типа защищенный (**protected**).

- – закрытый (**private**).

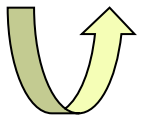
~ - пакетный (package). *Атрибут* с этой областью видимости недоступен или невиден для всех *классов* за пределами пакета, в котором определен *класс*-владелец данного *атрибута*.



Кратность (multiplicity)

[нижняя граница .. верхняя граница]

- [0..1]
- [0..*]
- [1..*]
- [1..3,7..*]
- 1=[1..1]
- *=[0..*]



Типы атрибутов

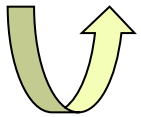
Тип атрибута представляет собой выражение, семантика которого определяется языком спецификации соответствующей модели

цвет: Color = (255, 0, 0)

имя_сотрудника [1..2] : String = Иван

видимость:Boolean

форма:Многоугольник = прямоугольник

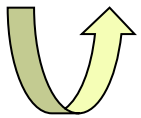


Типы атрибутов

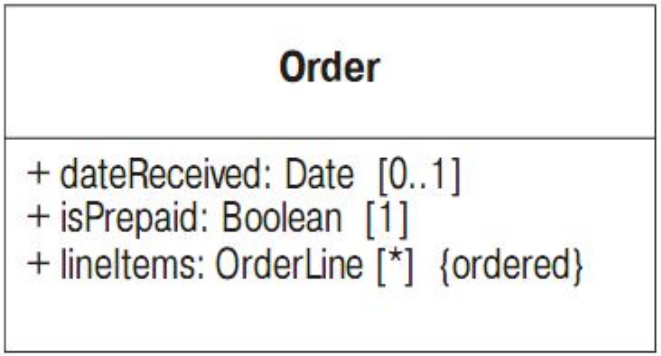
Подчеркивание означает, что атрибут может принимать подмножество значений из области его значений

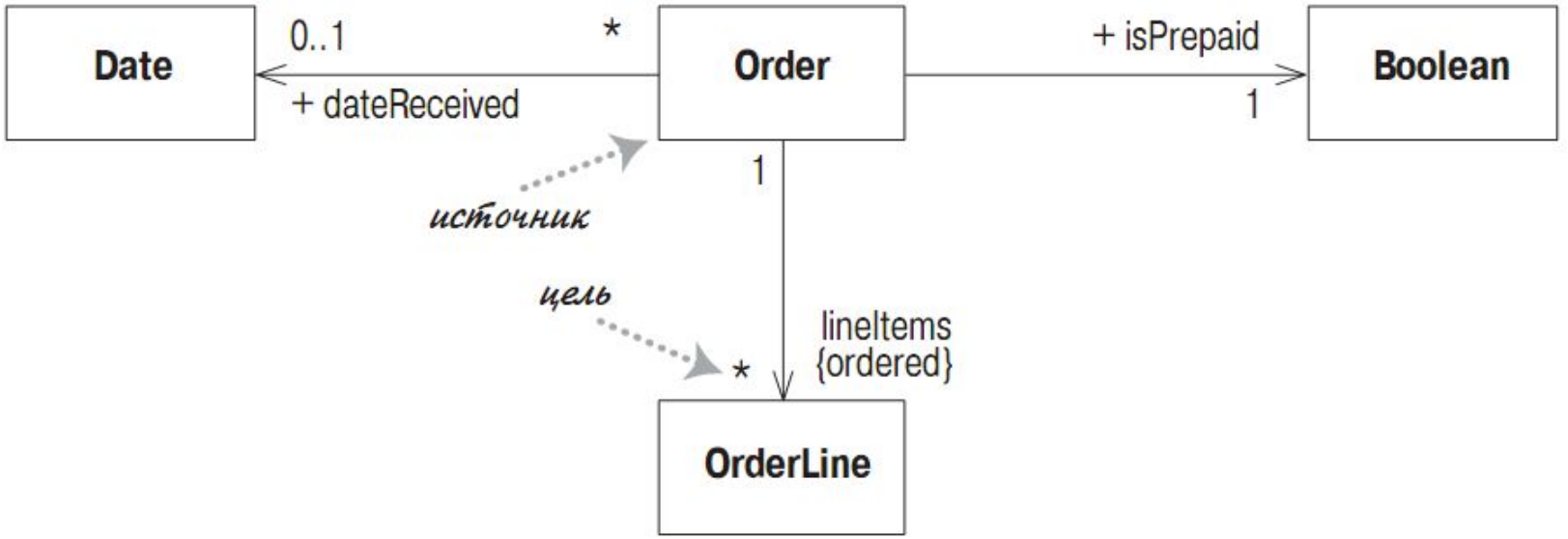
{Строка-свойство} – служит для указания значения атрибута, которое он получает при создании класса

заработная_плата:Currency = {$500}



Типы атрибутов

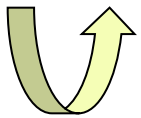




Операции

Операция или метод (operation) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию.

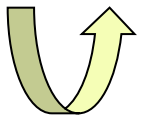
<квантор видимости><имя операции> (список параметров): <выражение типа возвращаемого значения>{строка-свойство}



Параметр

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

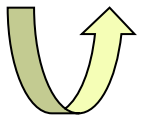
Вид (направление) параметра — есть одно из ключевых слов **in**, **out** или **inout**

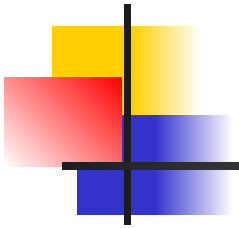


Параметр

Строка-свойство

- Запрос {**query**}
- последовательная (**sequential**)
- параллельная (**concurrent**) - может выполняться параллельно с другими операциями в системе, при этом параллельность должна поддерживаться на уровне реализации модели.
- охраняемая (**guarded**) - все обращения к данной операции должны быть строго упорядочены во времени, при этом могут быть приняты дополнительные меры по контролю исключительных ситуаций на этапе ее выполнения.
- **abstract**
- **signal** (Аналог message)



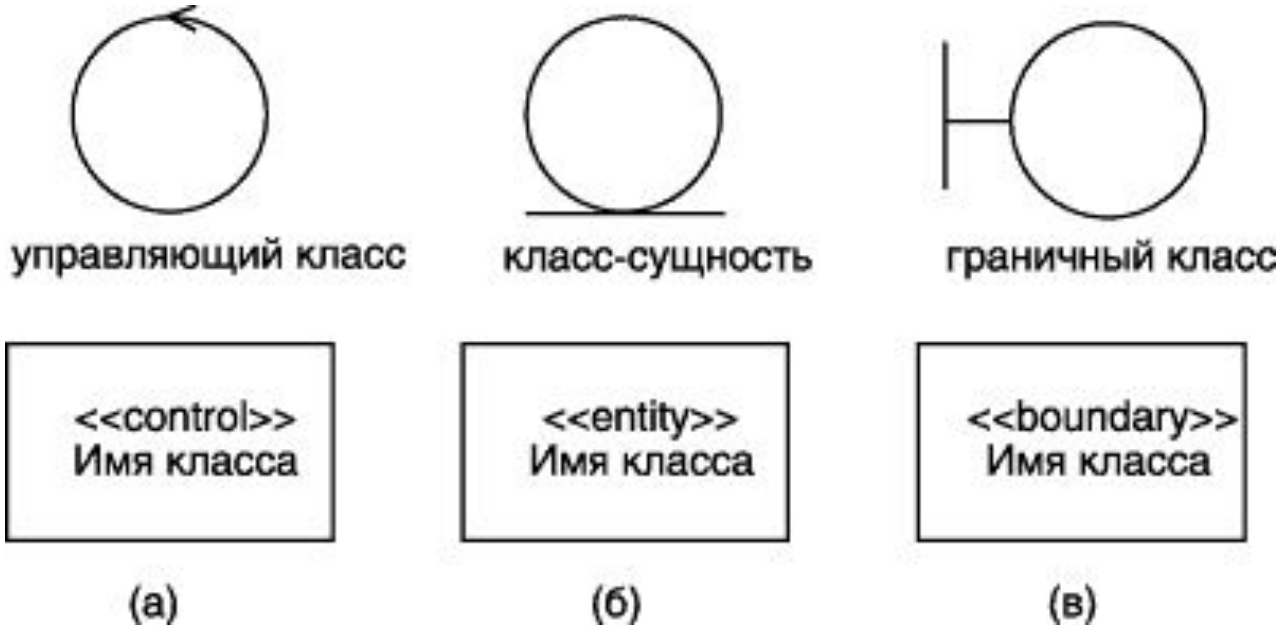


Параметр

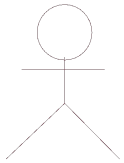
+нарисовать(форма: Многоугольник = прямоугольник, цвет_заливки: Color = (0, 0, 255))

Расширение языка UML

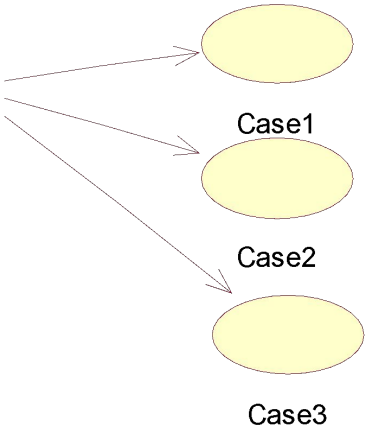
- Управляющий *класс* (control class)
- *Класс*-сущность (entity class)
- Граничный *класс* (boundary class)

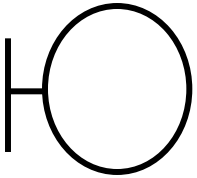


Расширение языка UML

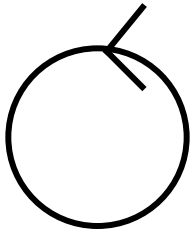


Actor1

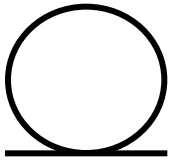




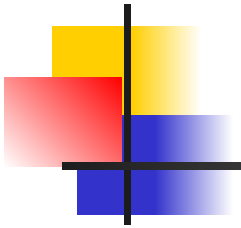
Пограничный объект



Управляющий объект



Объект-сущность

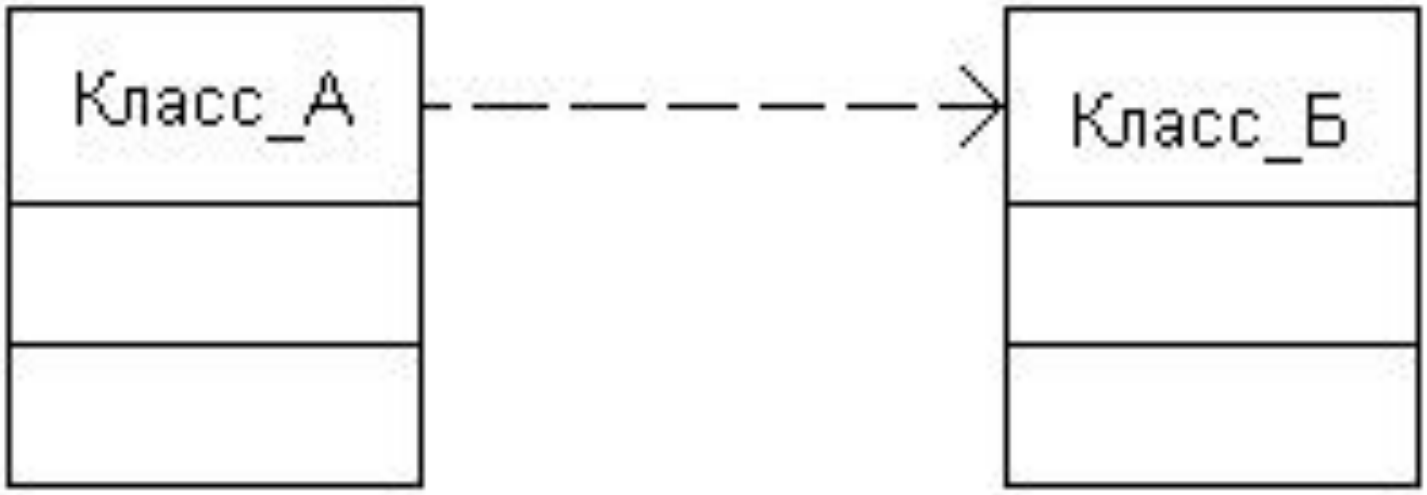


Типы отношений

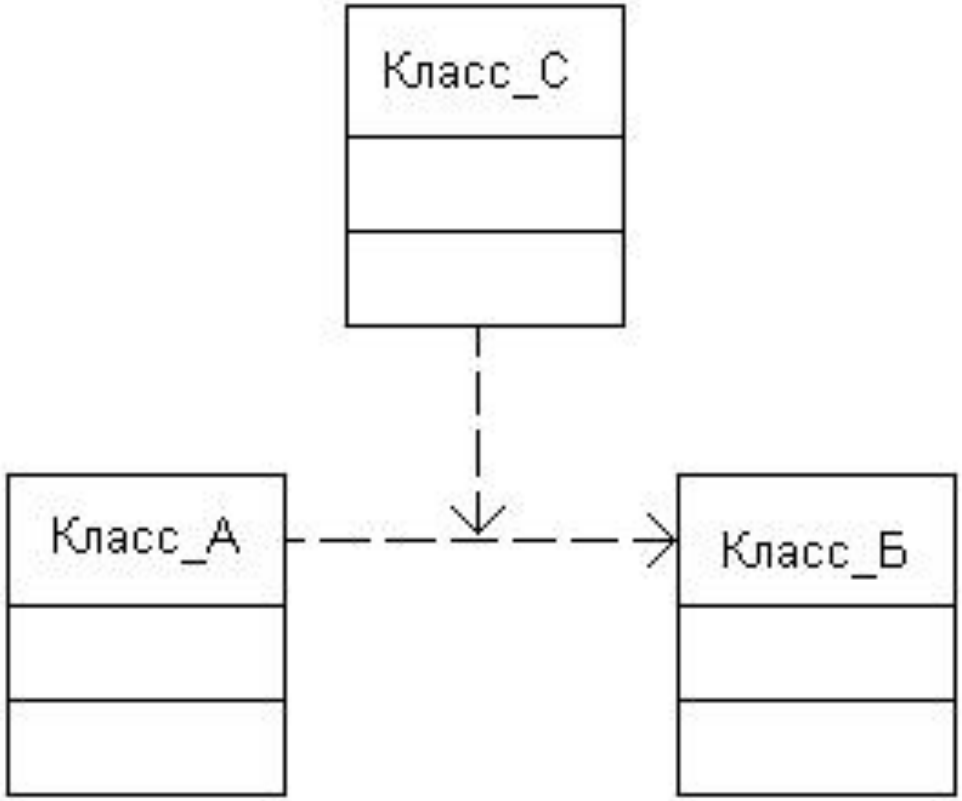
- Отношение зависимости (*dependency relationship*)
- Отношение ассоциации (*association relationship*)
- Отношение обобщения (*generalization relationship*)
- Отношение реализации (*realization relationship*)

Отношение зависимости

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



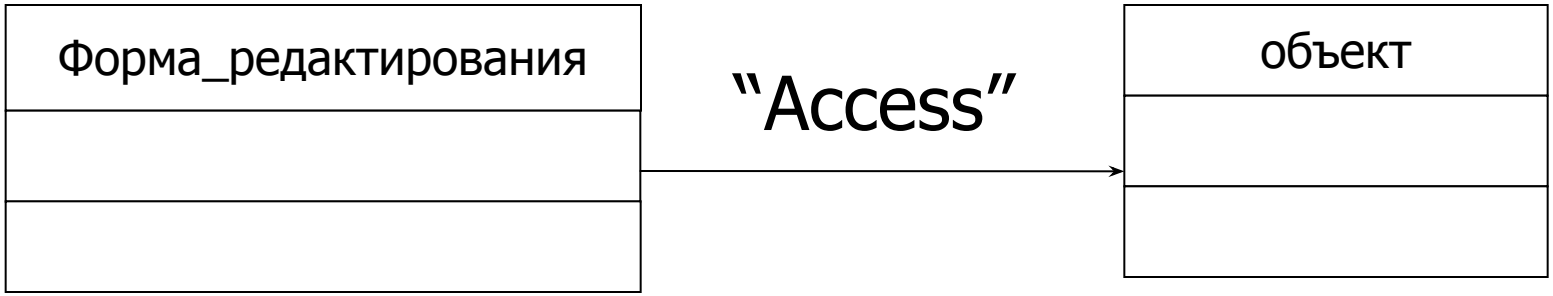
Отношение зависимости



Стереотипы отношения зависимости

- «access» - служит для обозначения доступности открытых атрибутов и операций класса-источника для классов-клиентов;
- «bind» - класс-клиент может использовать некоторый шаблон для своей последующей параметризации;
- «derive» - атрибуты класса-клиента могут быть вычислены по атрибутам класса-источника;
- «import» - открытые атрибуты и операции класса-источника становятся частью класса-клиента, как если бы они были объявлены непосредственно в нем;
- «refine» - указывает, что класс-клиент служит уточнением класса-источника в силу причин исторического характера, когда появляется дополнительная информация в ходе работы над проектом

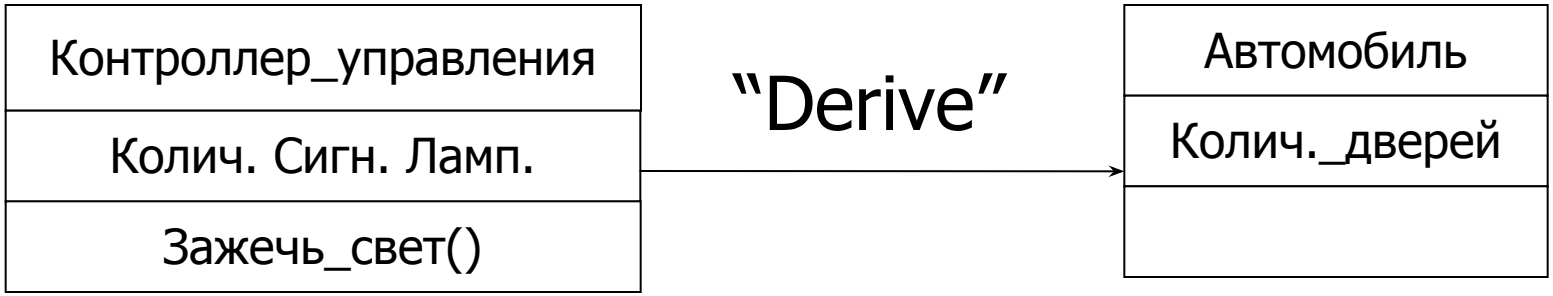
Стереотип “Access”



Класс Клиент
(Зависимый)

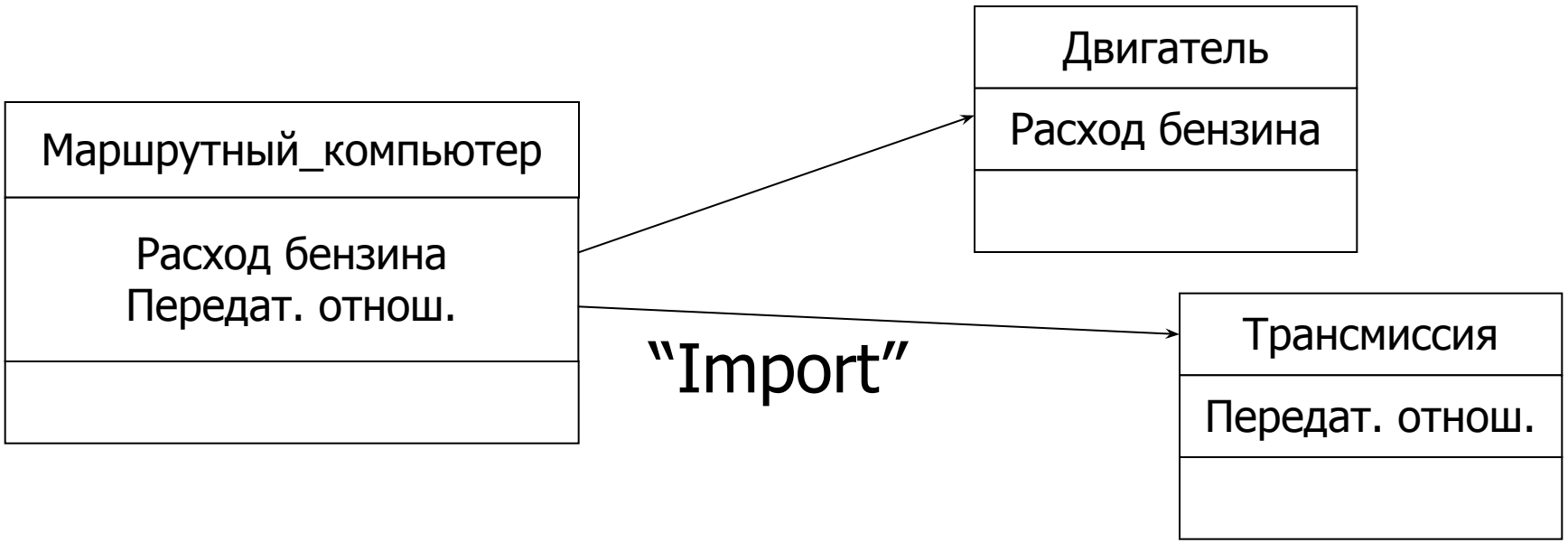
Класс источник
(Независимый)

Стереотип “Derive”



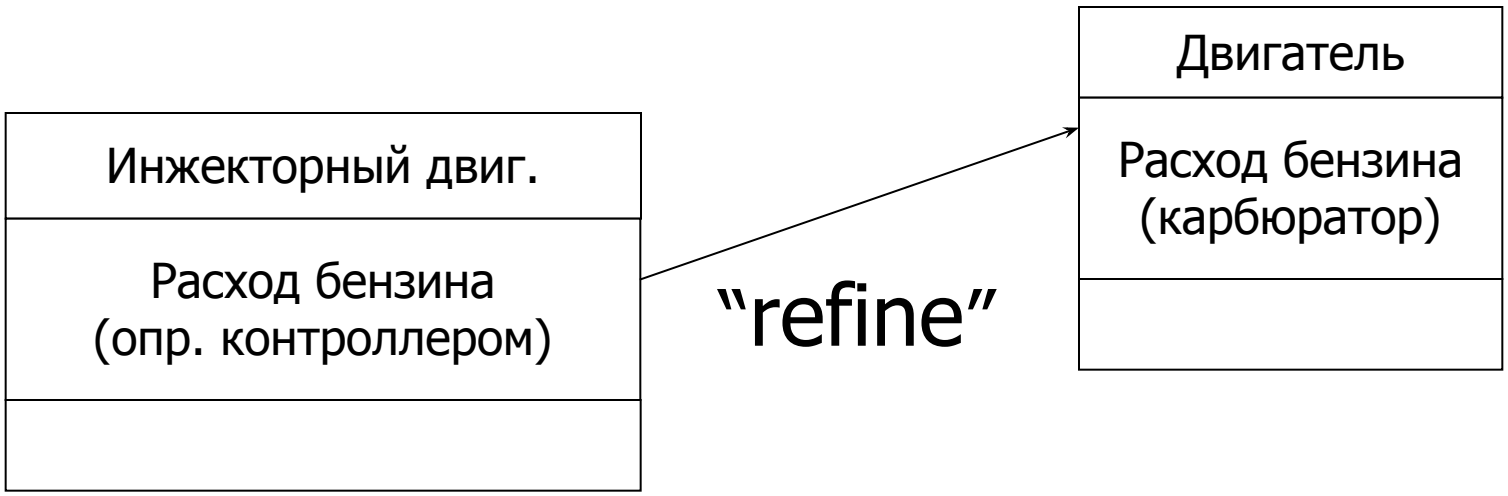
атрибуты класса-клиента могут быть вычислены по атрибутам класса-источника

Стереотип “Import”



открытые атрибуты и операции класса-источника становятся частью класса-клиента, как если бы они были объявлены непосредственно в нем (например, шаблон проектирования Facade)

Стереотип “refine”



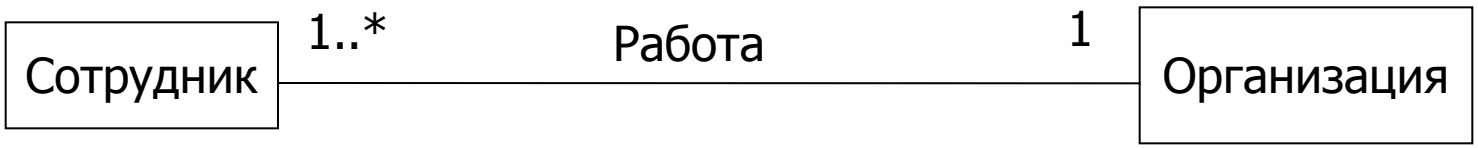
указывает, что класс-клиент служит уточнением класса-источника в силу причин исторического характера, когда появляется дополнительная информация в ходе работы над проектом

Отношение ассоциации

Соответствует наличию некоторого отношения между классами.

Дополнительно характеризуют

- Имя
- Кратность
- Роль

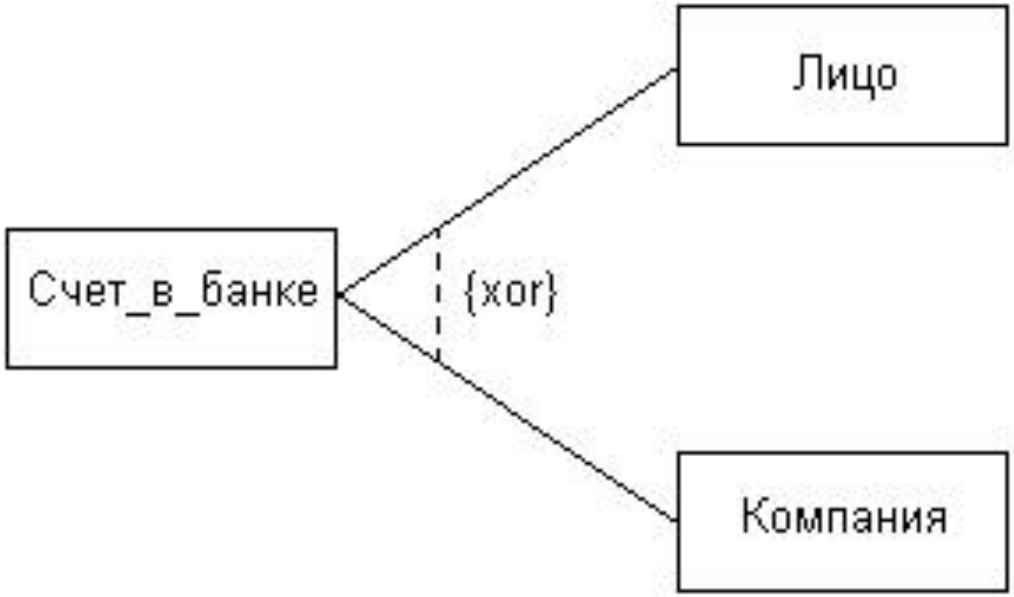


Порядок ассоциации

N-арная ассоциация

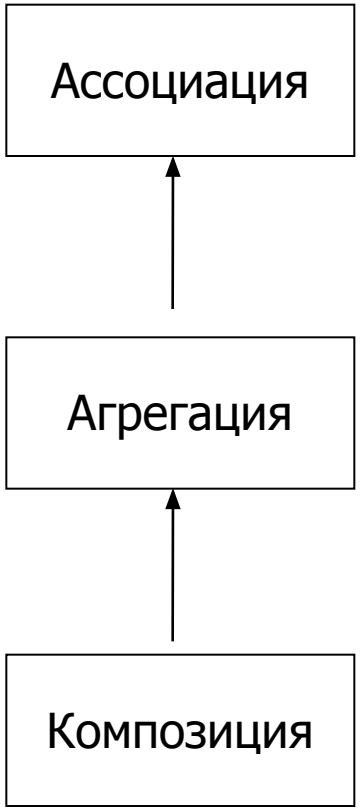
- Ассоциация-класс – класс, реализующий ассоциацию
- Конец ассоциации

XOR-ассоциация



Отношение агрегации

- Классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.



Отношение агрегации

- В связи с рассмотрением данного отношения вполне уместно вспомнить о специальном термине "агрегат", которое служит для обозначения технической системы, состоящей из взаимодействующих составных частей или подсистем

Отношение агрегации

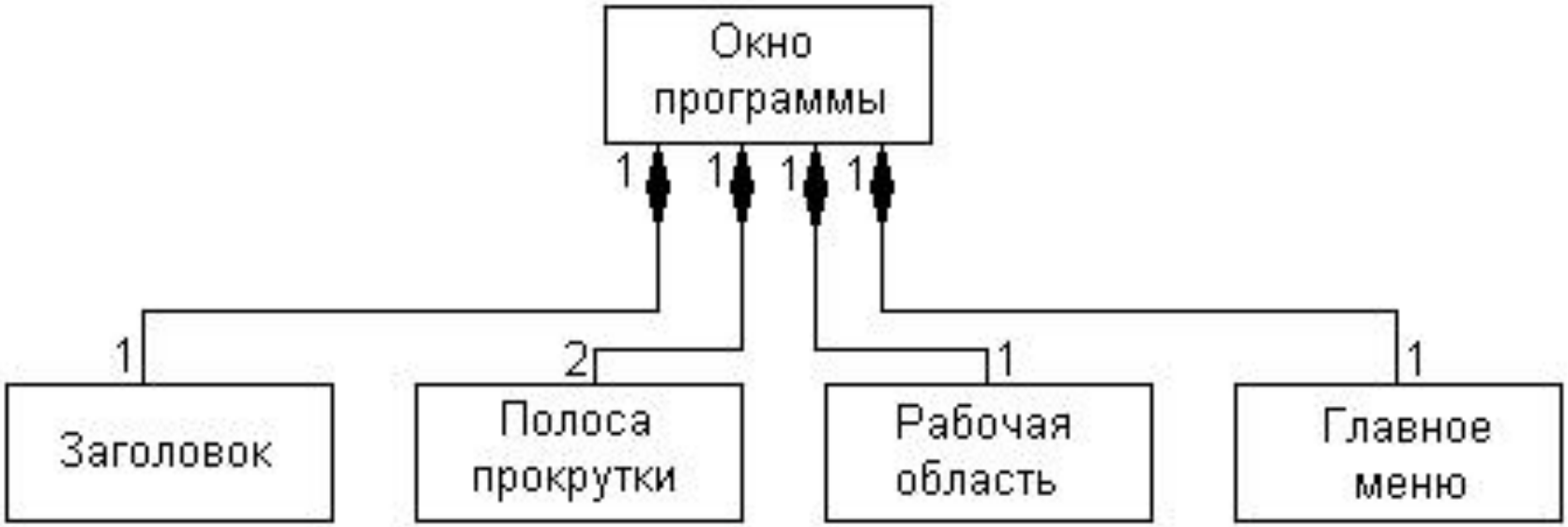
- В связи с рассмотрением данного отношения вполне уместно вспомнить о специальном термине "агрегат", которое служит для обозначения технической системы, состоящей из взаимодействующих составных частей или подсистем

Отношение агрегации

Отношение композиции

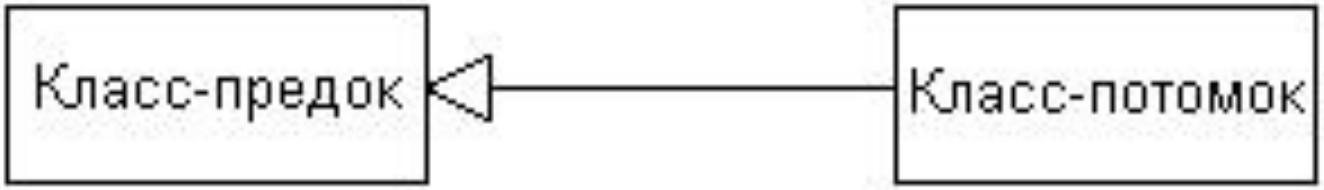
части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

Отношение композиции

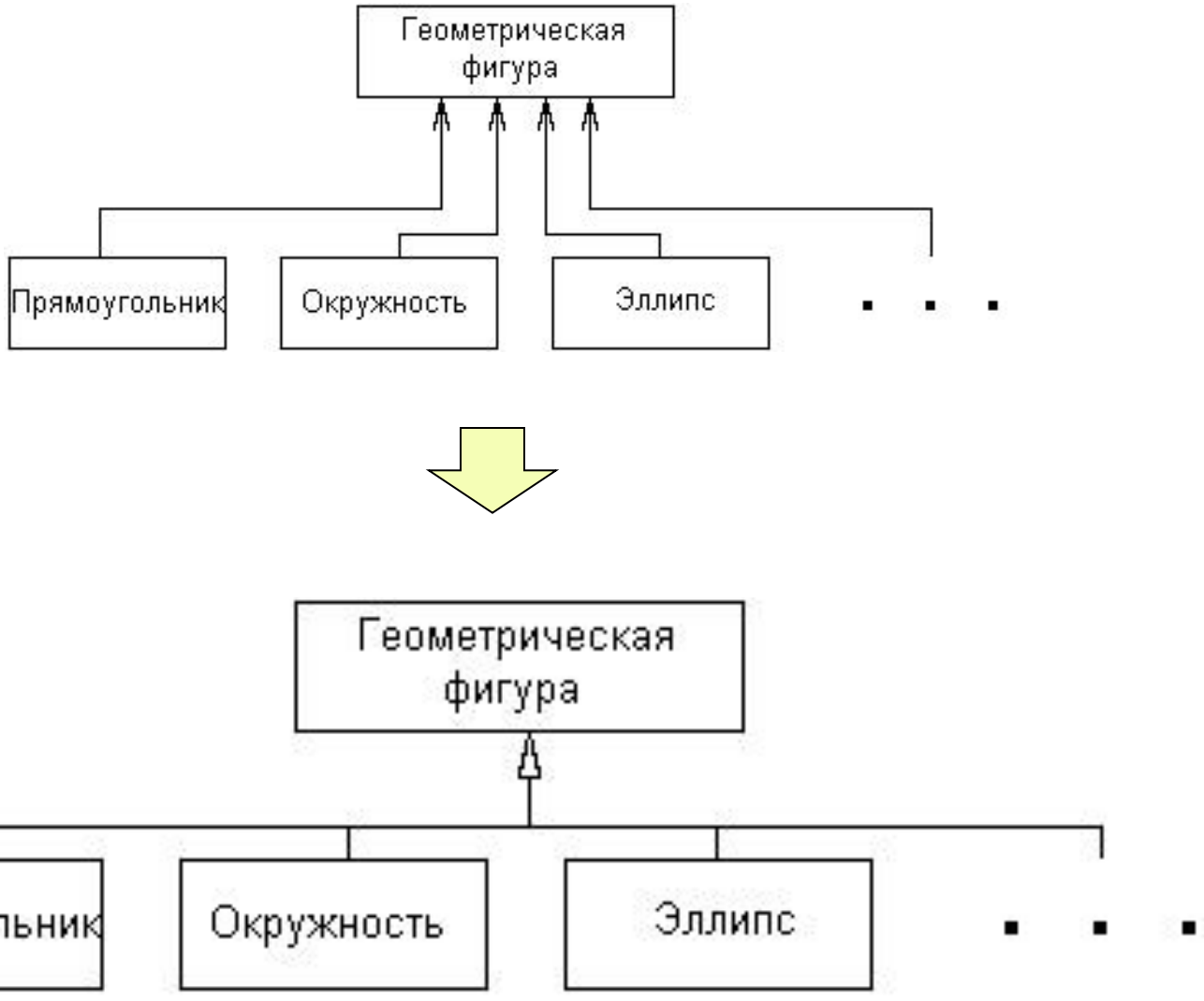


Отношение обобщения

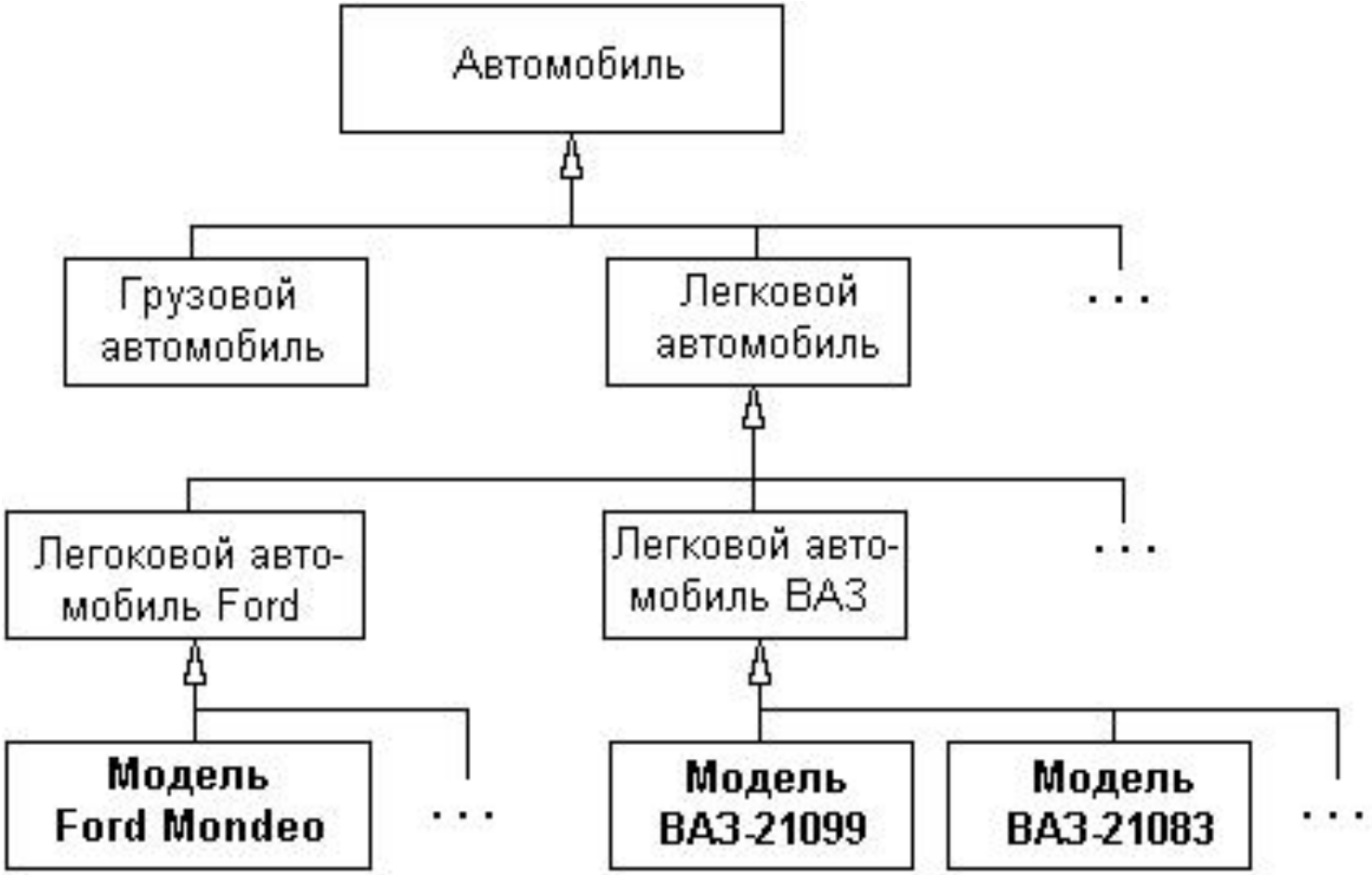
отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).



Отношение обобщения



Ограничения отношения обобщения



Интерфейс