Информационные технологии

- Диаграммы классов (продолжение)

Типы отношений

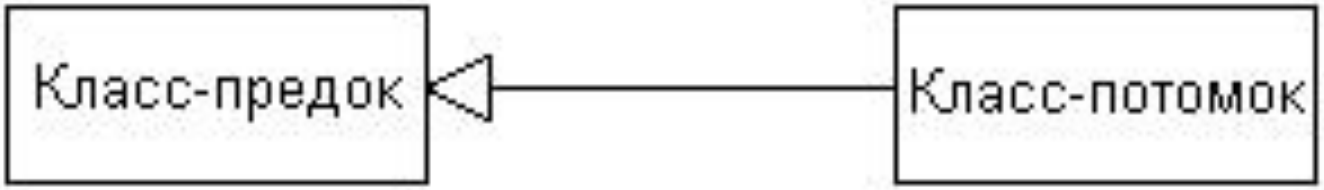
- Отношение зависимости (*dependency relationship*)
- Отношение ассоциаций (*association relationship*)
- Отношение обобщения (*generalization relationship*)
- Отношение реализации (*realization relationship*)

Типы отношений

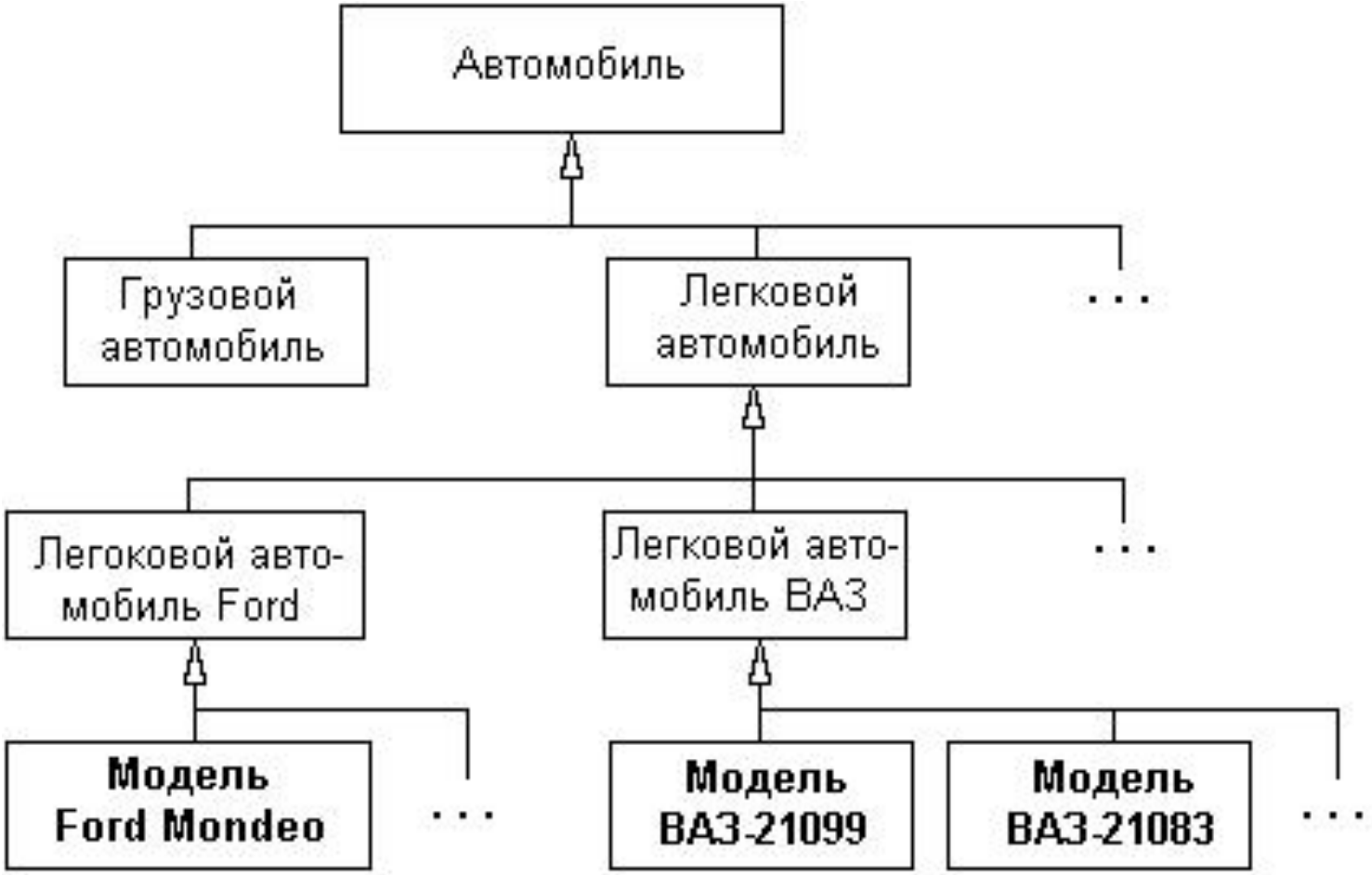
- Зависит от
- Знает о 
- Наследование: является предком (потомком) 
- Реализует 

Отношение обобщения

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



Ограничения отношения обобщения



Отношение обобщения

Осторожно использовать термин «является»:

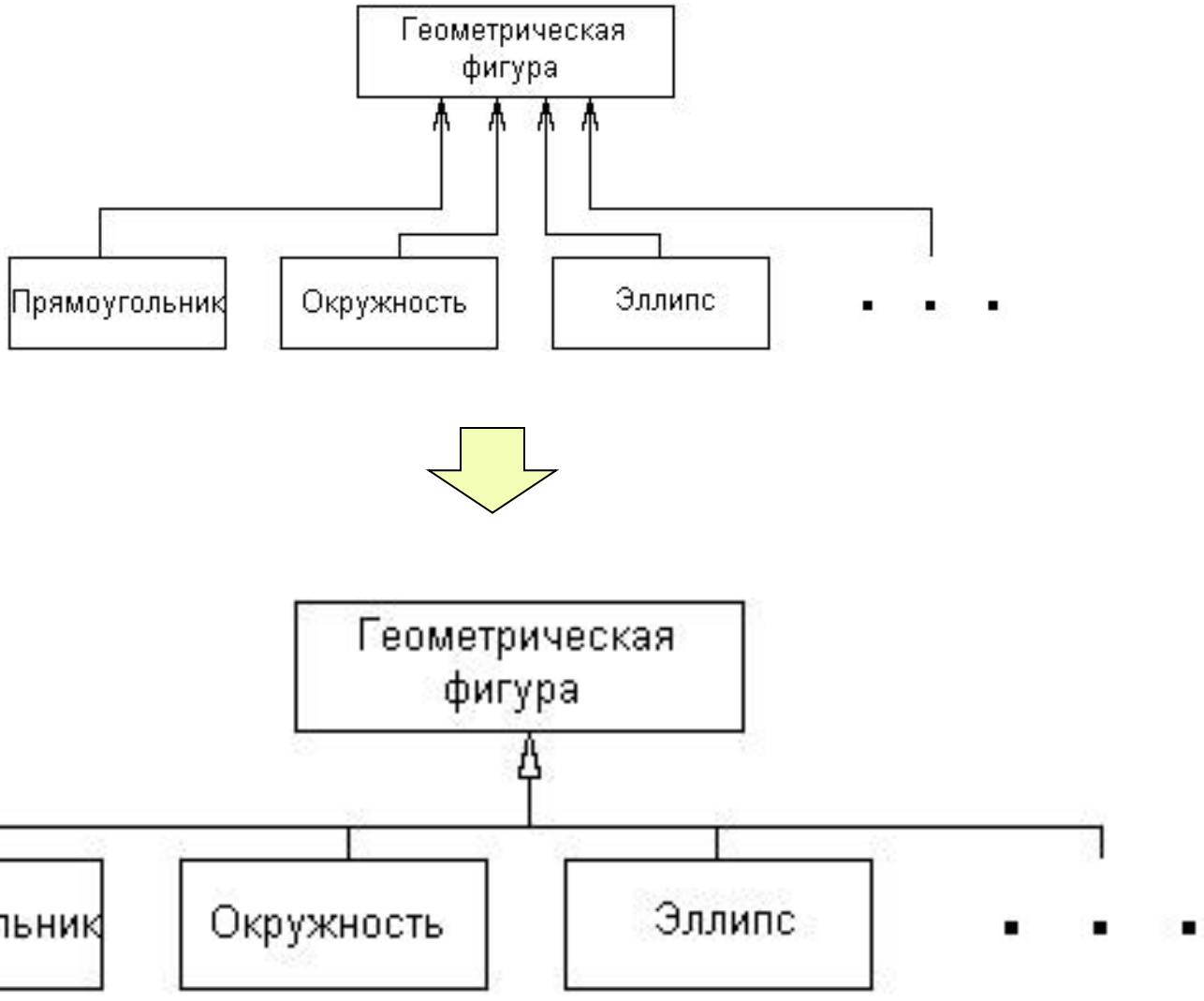
1. Шеп – это бордер-колли.
2. Бордер-колли – это собака.
3. Собаки являются животными.
4. Бордер колли – это порода собак.
5. Собака – биологический вид.

(1)+(2) => Шеп – это собака

(2)+(3) => бордер-колли – это животное

(1)+(4) => Шеп – это порода собак (*неверно!*)

Отношение обобщения

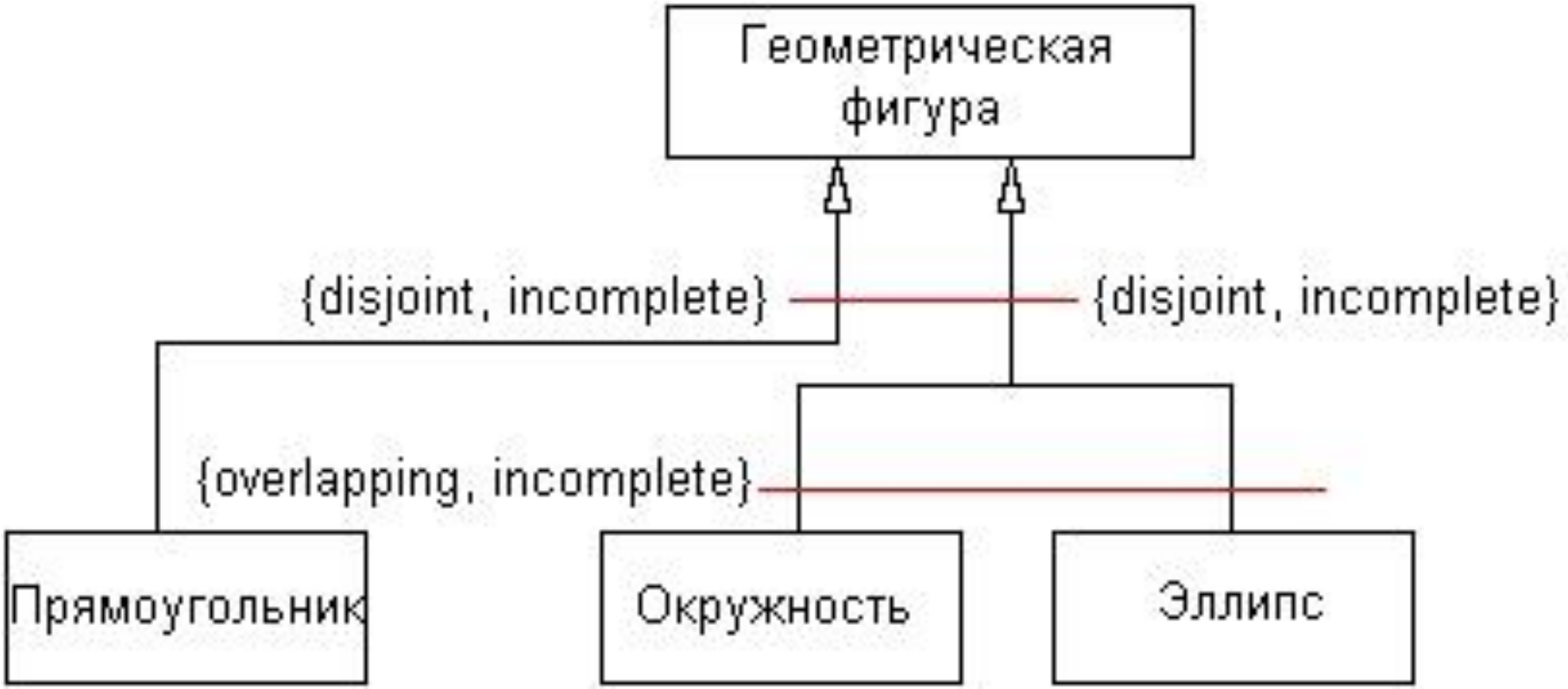


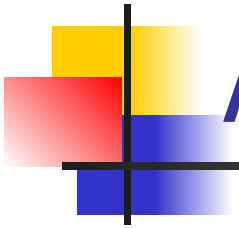
Ограничения отношения обобщения

строка текста, указывающая на некоторые дополнительные свойства этого отношения

- {complete} -- определены все классы-потомки
- {disjoint} -- классы-потомки не содержат объектов, одновременно являющихся экземплярами двух или более классов
- {incomplete}
- {overlapping} -- экземпляры классов-потомков могут принадлежать одновременно нескольким классам

Ограничения отношения обобщения

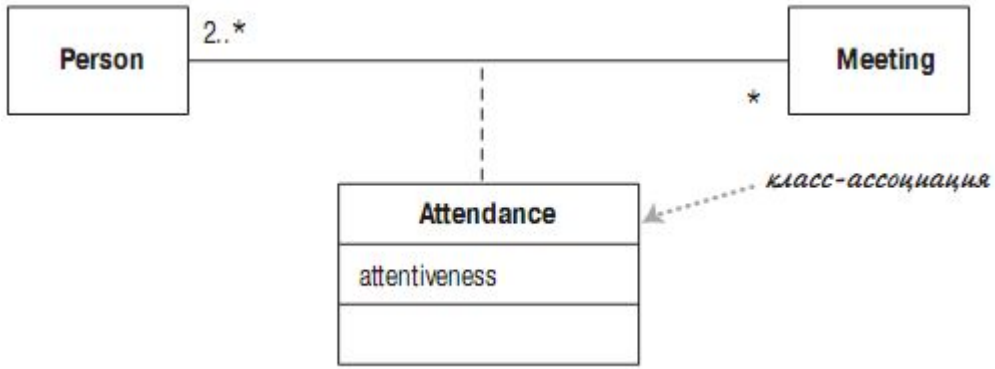


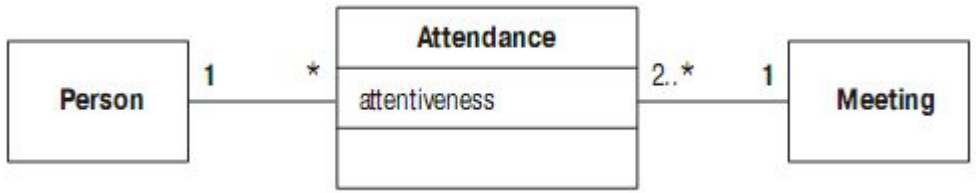


Ассоциация

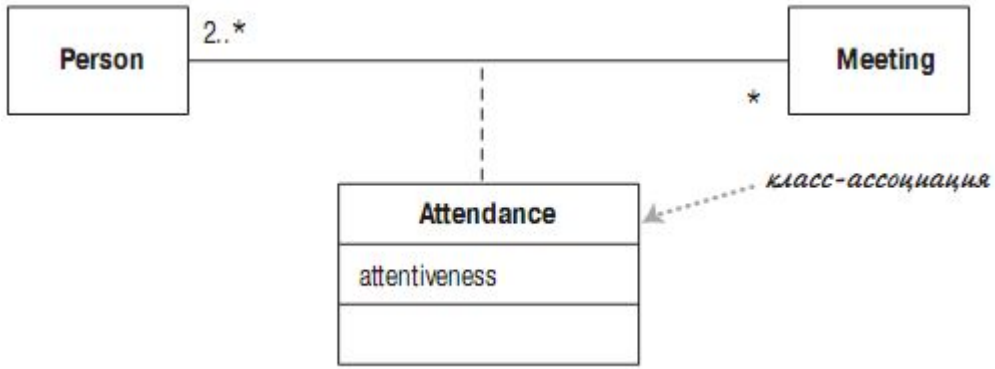
- Дополнительные понятия

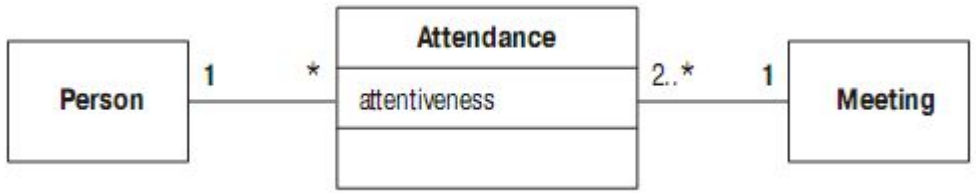
Класс-ассоциация





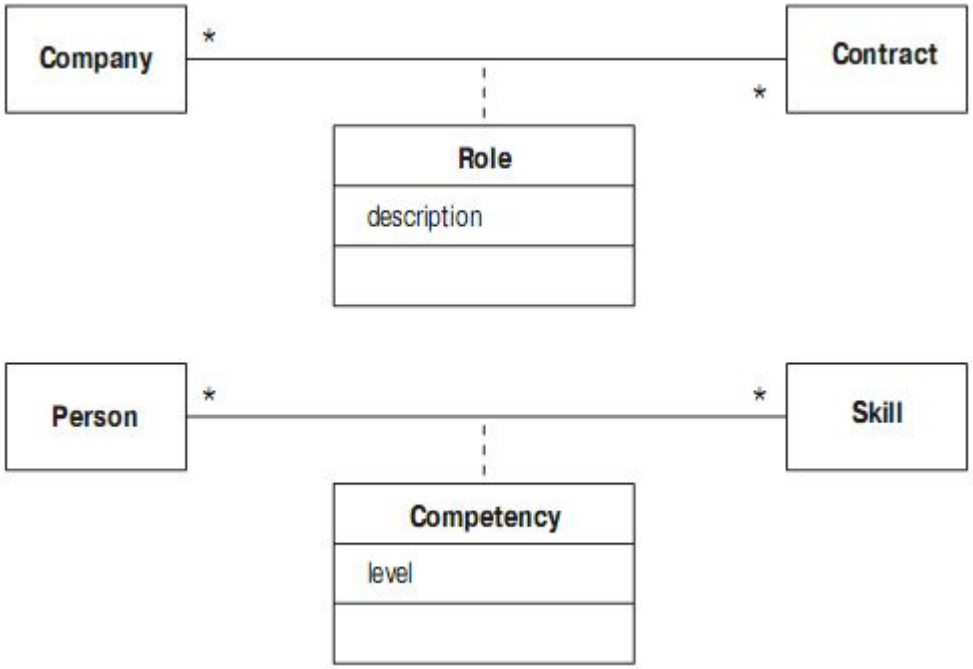
Класс-ассоциация





Класс-ассоциация

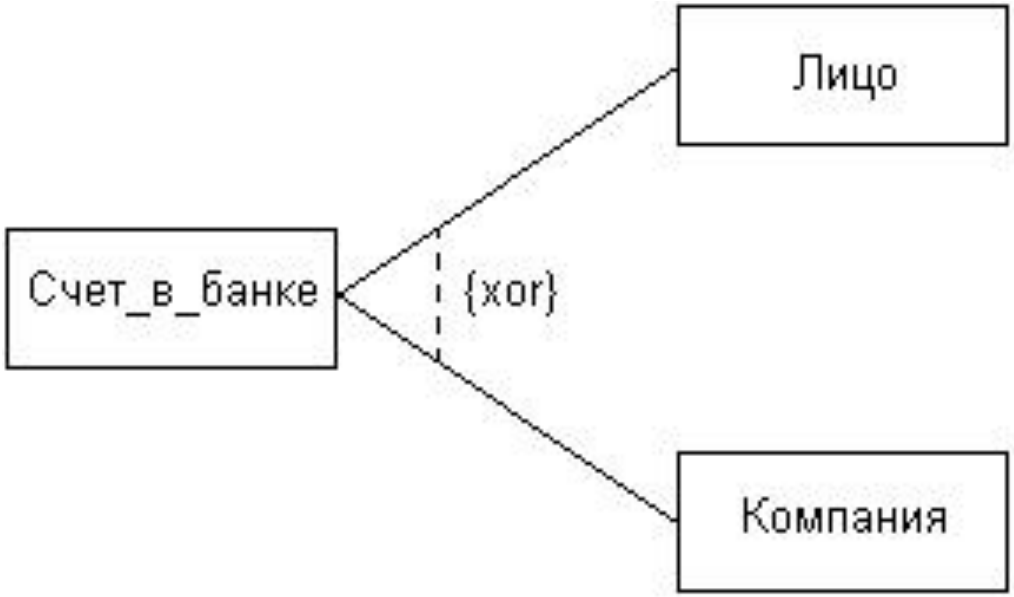
В UML для каждого клиента подразумевается только одно отношение (последний вариант):



N-арная ассоциация

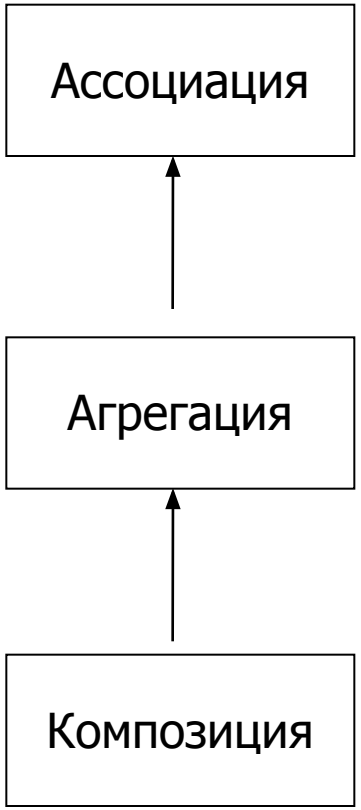
- Ассоциация-класс – класс, реализующий ассоциацию
- Конец ассоциации

XOR-ассоциация



Отношение агрегации

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



Отношение агрегации

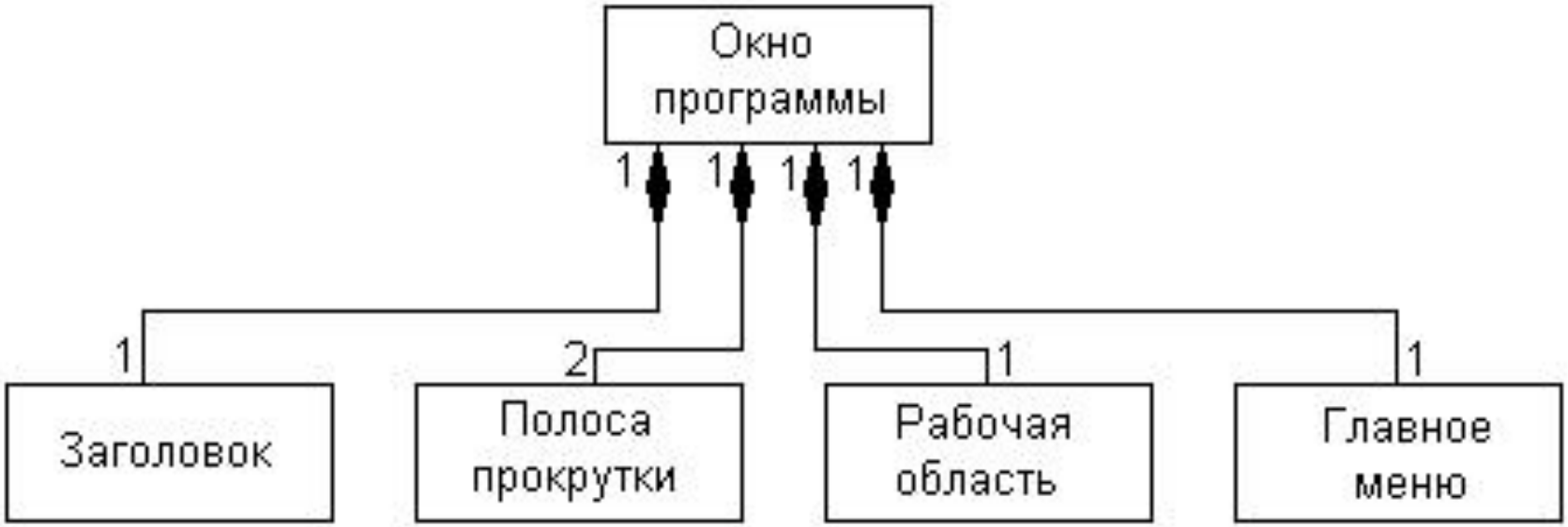
- В связи с рассмотрением данного отношения вполне уместно вспомнить о специальном термине "агрегат", которое служит для обозначения технической системы, состоящей из взаимодействующих составных частей или подсистем

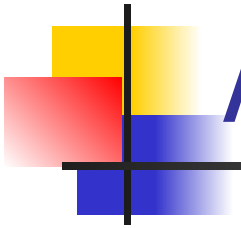
Отношение агрегации

Отношение композиции

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

Отношение композиции





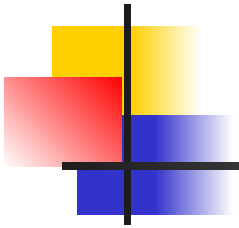
Ассоциация

- Различие между **агрегированием** и **осведомленностью**(acquaintance)
- Агрегирование подразумевает, что один объект владеет другим или несет за него ответственность.
- В общем случае мы говорим, что объект содержит другой объект или является его частью.

GoF, 95

Отношение агрегации

- ***Агрегирование*** и ***осведомленность*** легко спутать, поскольку они часто реализуются одинаково.
- определяется, скорее, предполагаемым использованием, а не языковыми механизмами.

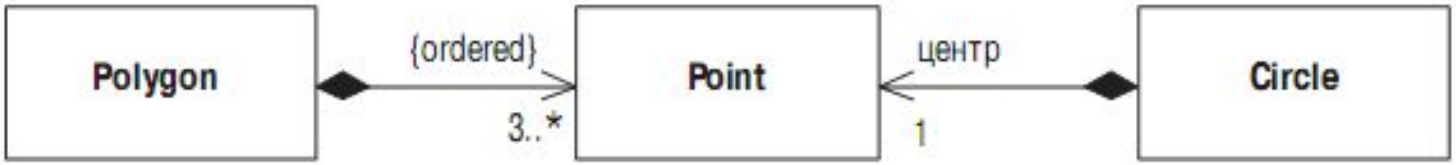


Отношение агрегации

- Агрегирование подразумевает, что один объект владеет другим или несет за него ответственность.

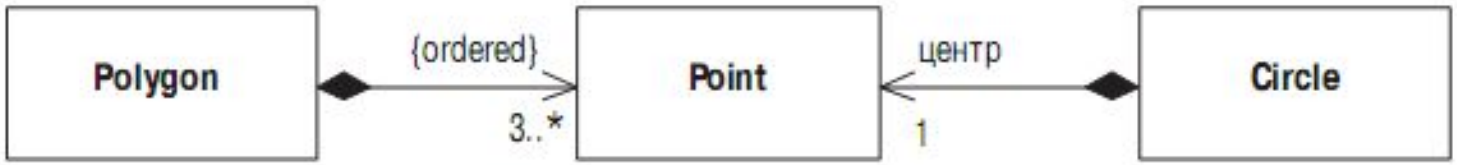
Отношение агрегации

- Композиция подразумевает, что один объект владеет другим или несет за него ответственность (создает и уничтожает).



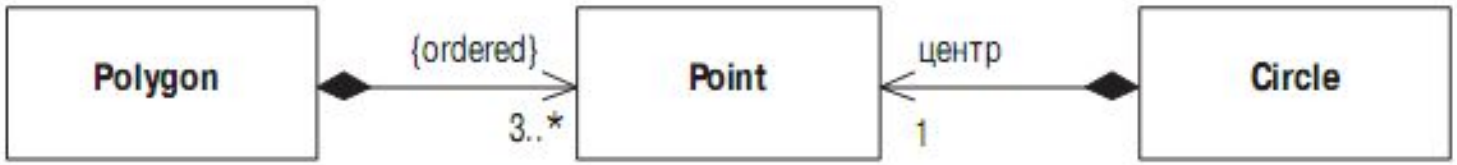
Отношение агрегации

На диаграмме классов можно показать несколько классов потенциальных владельцев, но у любого экземпляра класса есть только один объект владелец



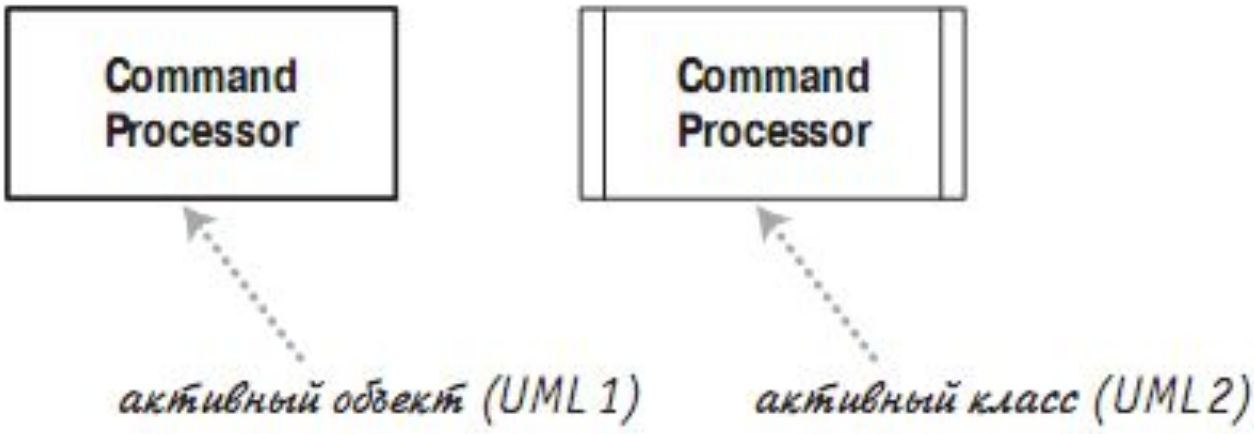
Отношение агрегации

Правило «нет совместного владения» является ключевым в композиции

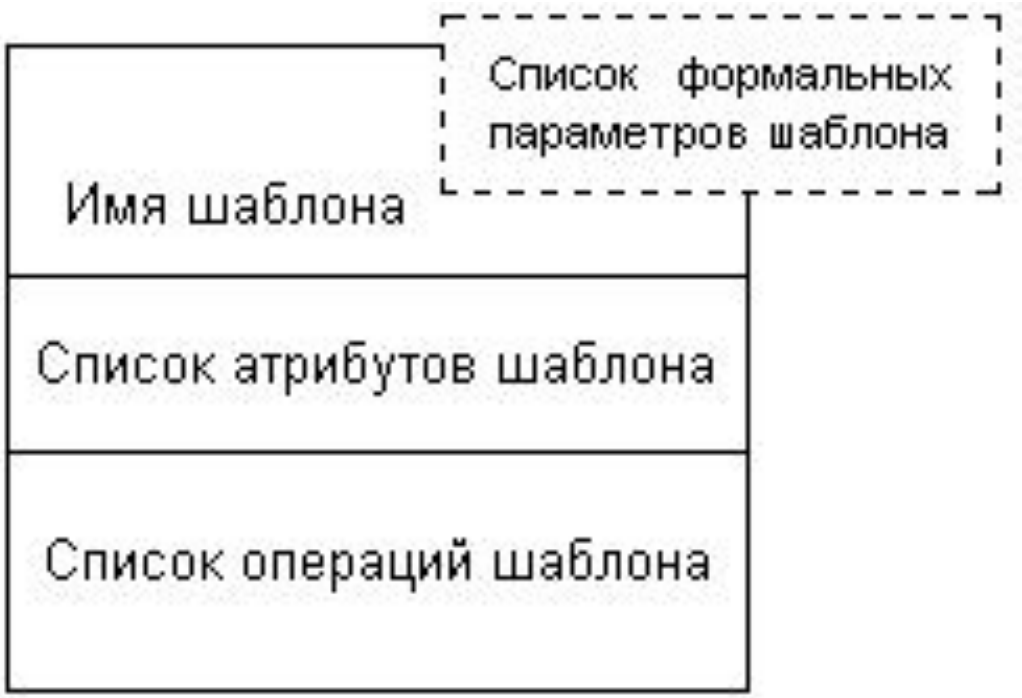


Производные свойства

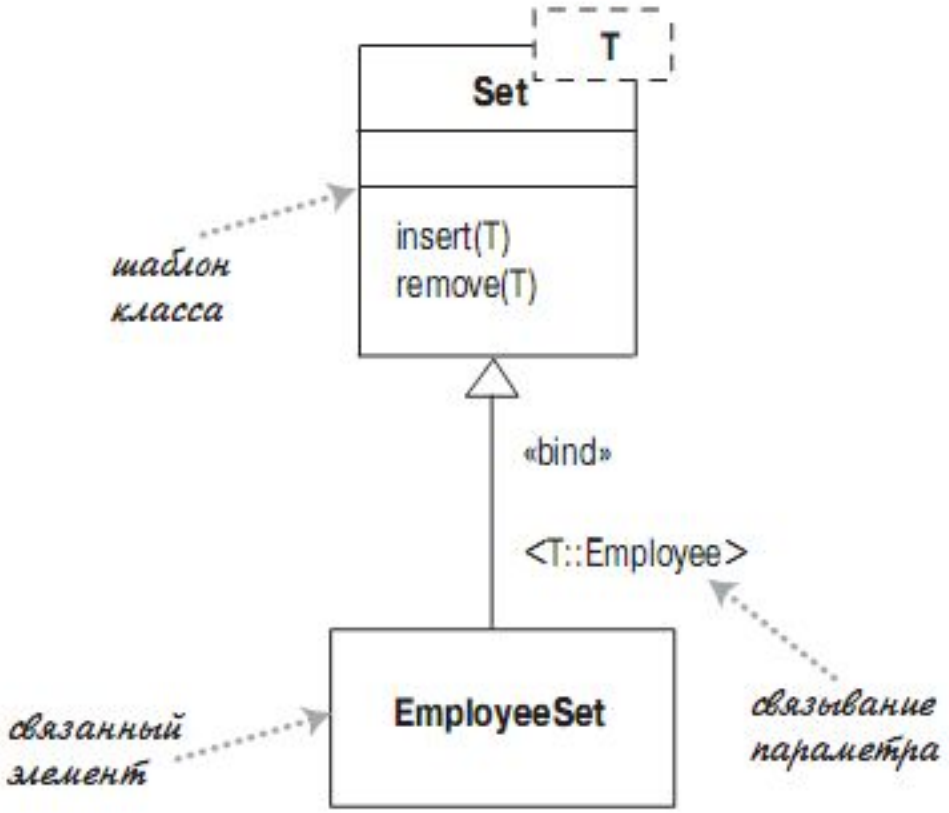
Активный класс



Шаблоны или параметризованные классы

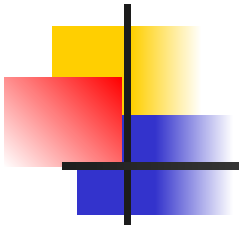


Шаблоны или параметризованные классы



Объекты


квадрат: Прямоугольник
(а)
квадрат
(б)
квадрат: Прямоугольник
вершина = (1 , 10)
сторона = 15
цвет_границы = черный
цвет_заливки = белый
(в)
Прямоугольник
(г)



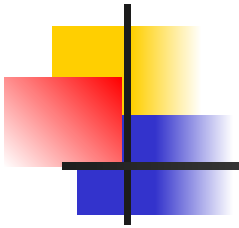
Объекты

Диаграммы объектов удобны для показа примеров связанных друг с другом объектов. Во многих ситуациях точную структуру можно определить с помощью диаграммы классов, но при этом структура оста ется трудной для понимания. В таких случаях пара примеров диаграммы объектов может прояснить ситуацию.

Объекты

<собственное имя объекта >'/'<Имя роли класса>:<Имя класса >.

- o : C– *объект* с собственным именем o, экземпляр класса C.
- : C– анонимный *объект*, экземпляр класса C.
- o :(или просто o) — *объект*-сирота с собственным именем o.
- o / R : C— *объект* с собственным именем o, экземпляр класса C, играющий роль R.
- / R : C— анонимный *объект*, экземпляр класса C, играющий роль R.
- o / R— *объект*-сирота с собственным именем o, играющий роль R.
- / R— анонимный *объект* и одновременно *объект*-сирота, играющий роль R.



Рекомендации

Как разбить на классы?

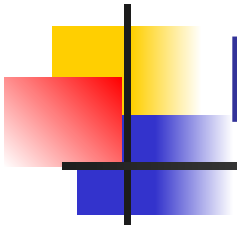
Русский язык или английский?

Рекомендации

1. Использование классов, ассоциаций, атрибутов, отношений и ограничений решает 90% всех задач моделирования.
2. Сконцентрировать внимание только на важных аспектах проблемы
3. Выбор точки зрения должен соответствовать определенному этапу работы: концептуальный, спецификации, реализации)

Навигация в ассоциации

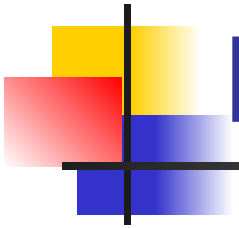
Это возможность легко находить те классы, на которые указывает данная ассоциация



Навигация в ассоциации

Если у ассоциации нет стрелок, то это трактуется:

- Двунаправленная ассоциация
- Направление не известно

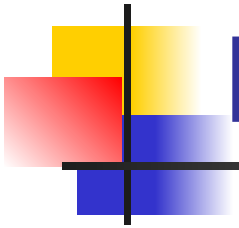


Методы

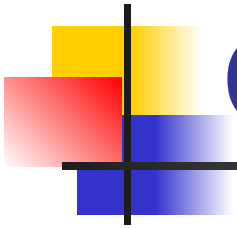
Как называть, метод или операция?

- Операция – функция класса
- Метод – экземпляр операции

Виды методов

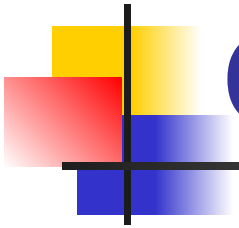


- Операция-запрос (не меняет состояние класса)
- Операция модификатор (меняет состояние класса)
  - Итераторы (Г.Буч)



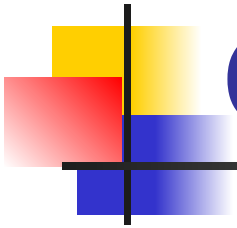
Советы использования

- Не пытайтесь использовать все доступные понятия. Начните с классов, ассоциаций, обобщений и ограничений
- Соответствуйте точке зрения модели (концептуальная, спецификации, реализации)
- Стоит сконцентрироваться на главных аспектах



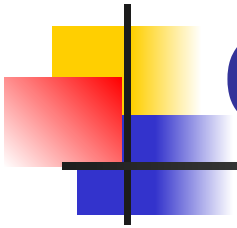
Ограничения

- Используются в языке Eiffel (Design by contract, Бертран Мейер)
- В основе лежит понятие утверждения: булевское высказывание, которое всегда истинно
- Предусловия, условия и инвариант



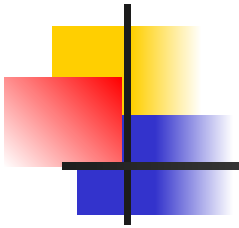
Ограничения

- Пусть A – это некоторая операция, тогда формула корректности (correctness formula)
- {P} A {Q} (Триада Хоара)
- {x = 5} x = x ^ 2 {x > 0}

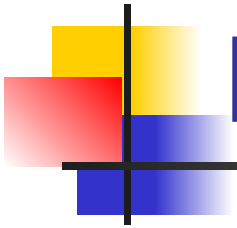


Ограничения

- Кто ответственен за выполнение проверки?
- Для предусловия ответственен вызывающий класс



- Самая трудная задача в объектно-ориентированном проектировании – разложить систему на объекты
- Можно сформулировать задачу письменно, выделить из получившейся фразы существительные и глаголы, после чего создать соответствующие классы и операции.



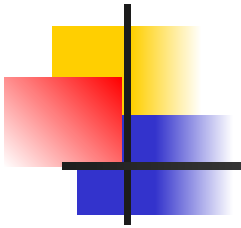
Моделирование

- Другой путь – сосредоточиться на отношениях и разделении обязанностей в системе.
- Согласие по поводу того, какой подход самый лучший, никогда не будет достигнуто. (GoF)

CRC – карточки

- Уорд Каннингхем и Кент Бек (разработчики Smalltalk) в конце 80-х, Удобны при построении диаграмм взаимодействия
- CRC: Class-Responsibility-Collaboration (Класс- Ответсвенность- Кооперация)
- Технология использовалась для проектирования модели классов

CRC карточки



ИМЯ КЛАССА	
ОТВЕТСТВЕННОСТЬ	КООПЕРАЦИЯ

CRC – карточки

- Небольшие карточки, размером 4 x 6см

Заказ	
Проверить наличие товара	Строка заказа
Определить цену	
Проверить факт оплаты	Клиент
Отправить по адресу доставки	

CRC карточки

Диаграммы классов используются:

диаграммы классов используются в следующих целях:

- для *моделирования словаря* системы
- для *моделирования простых кооперацій*
- для *моделирования логической схемы базы данных*.

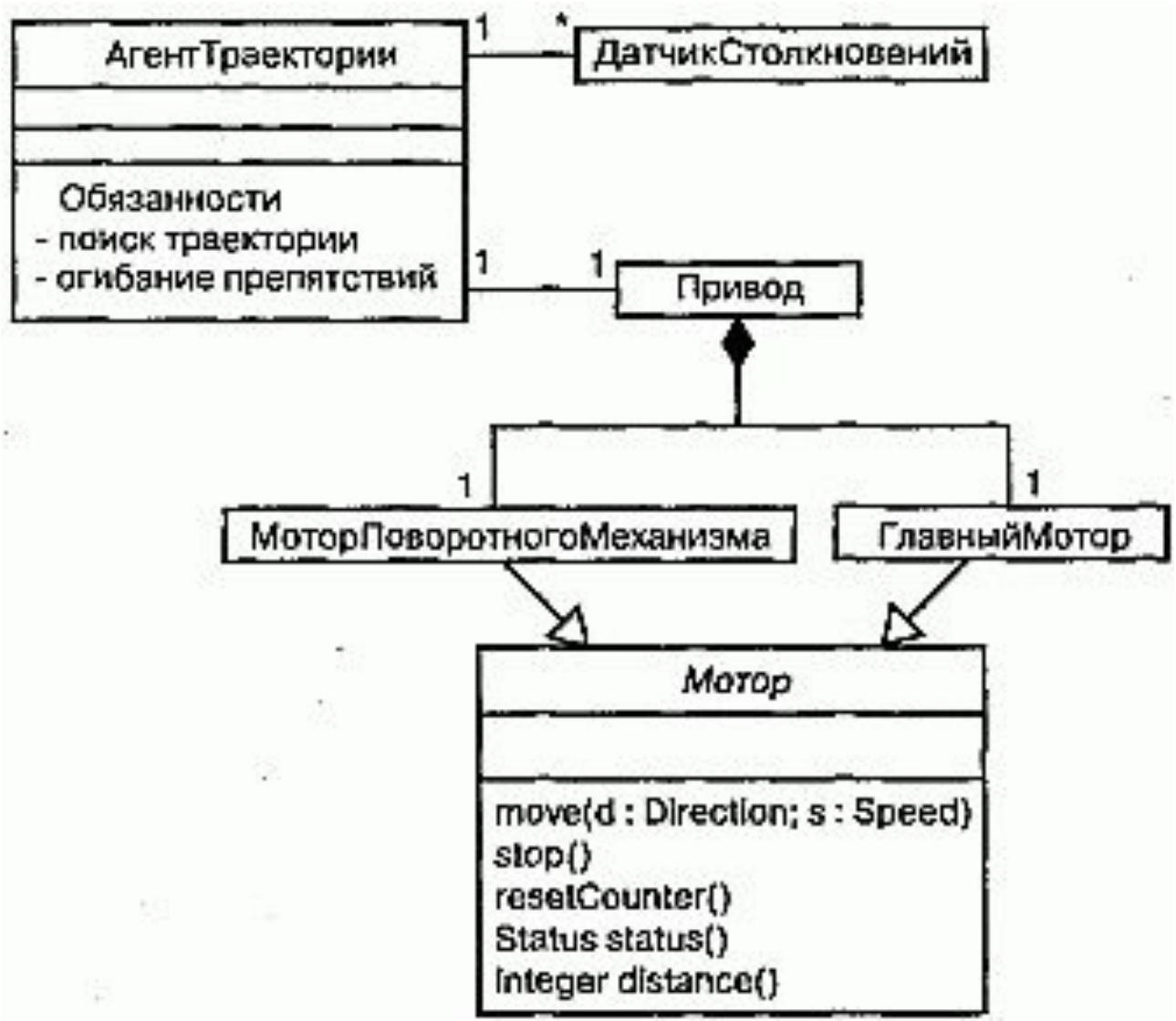
Диаграммы классов используются:

Моделирование словаря системы:

- Определите, какие элементы пользователи и разработчики применяют для описания задачи или ее решения. Используйте CRC-карточки.
- Выявите для каждой абстракции соответствующее ей множество обязанностей.
- Разработайте атрибуты и операции, необходимые для выполнения класса ими своих обязанностей.

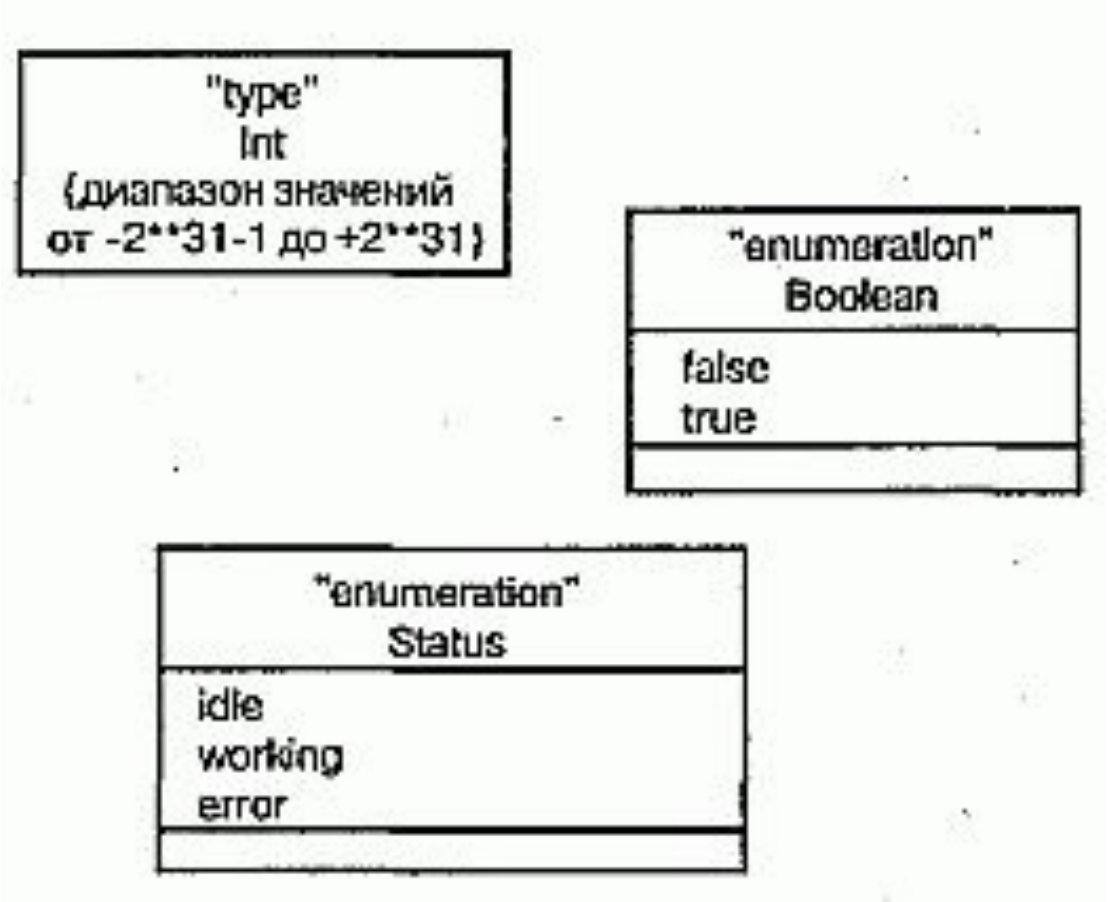
Диаграммы классов используются:

Пример:
робот



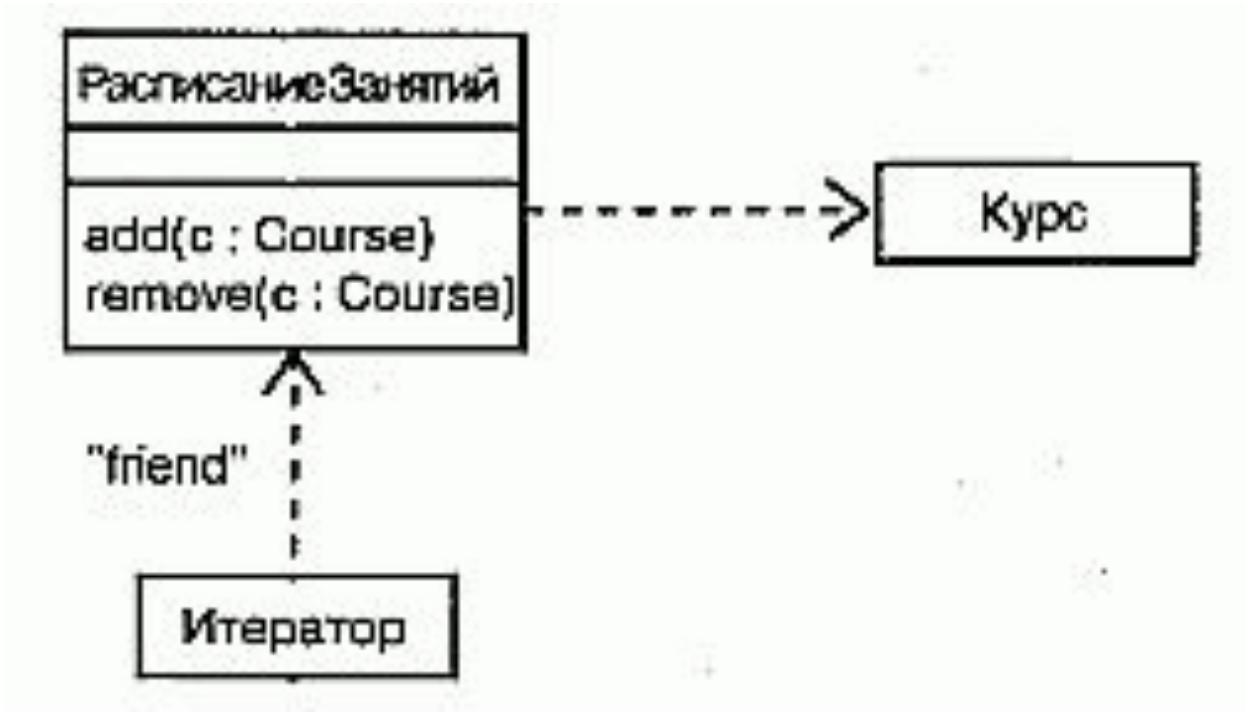
Диаграммы классов используются:

- Примитивные типы

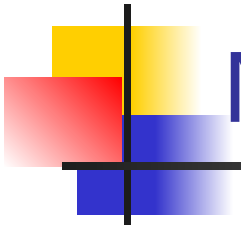


Диаграммы классов используются:

- соединение между классами, когда один класс использует другой в качестве параметра операции

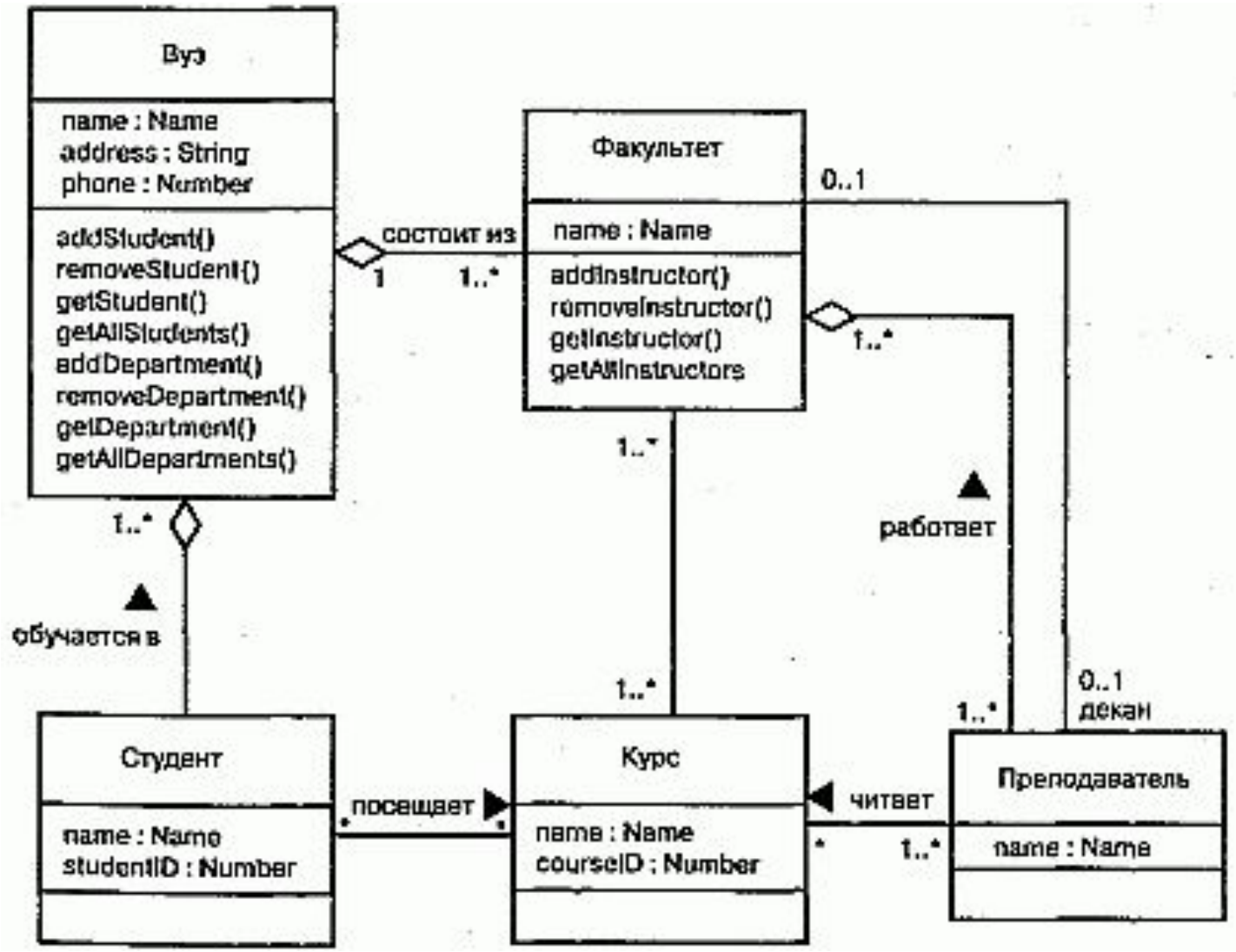


Моделирование схемы БД



- Идентифицируйте классы вашей модели
- Создайте содержащую эти классы диаграмму классов
- Раскройте структурные особенности классов (акрибуты, ассоциации, кратности)
- Поищите образцы, усложняющие проектирование физической базы данных, например циклические ассоциации, и др.
- Рассмотрите поведение этих классов

Моделирование схемы БД



Моделирование схемы БД

- используйте зависимость, только если моделируемое отношение не является структурным;
- используйте обобщение, только если имеет место отношение типа "является";
- множественное наследование часто можно заменить агрегированием;
- остерегайтесь циклических отношений обобщения;

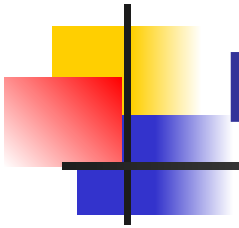
Моделирование схемы БД

- поддерживайте баланс в отношениях обобщения: иерархия наследования не должна быть ни слишком глубокой (желательно не более пяти уровней), ни слишком широкой (лучше прибегнуть к промежуточным абстрактным классам);
- применяйте ассоциации прежде всего там, где между объектами существуют структурные отношения.

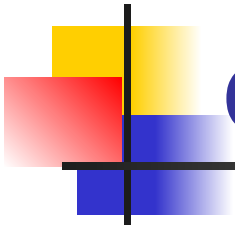
Моделирование схемы БД

- выбрав один из стилей оформления линий (прямые или наклонные), в дальнейшем старайтесь его придерживаться. Прямые линии подчеркивают, что соединения идут от родственных сущностей к одному общему родителю. Наклонные линии позволяют существенно сэкономить пространство в сложных диаграммах. Если вы хотите привлечь внимание к разным группам отношений, применяйте одновременно оба типа линий;

Моделирование схемы БД



- избегайте пересечения линий;
- показывайте только такие отношения, которые необходимы для понимания особенностей группирования элементов модели; скрывайте несущественные (особенно избыточные) ассоциации.



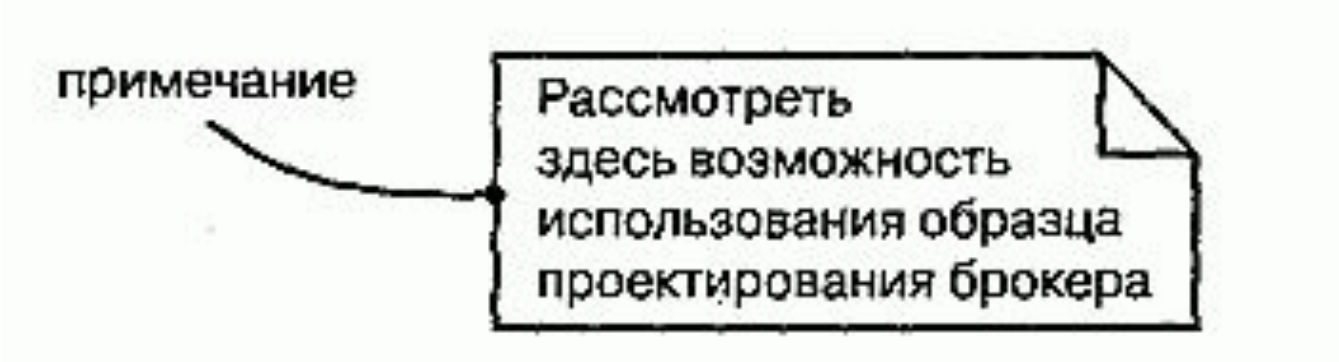
Основы структурного моделирования

Общие механизмы UML:

- *Примечание* (Note)
- *Стереотипом* (Stereotype)
- *Помеченное значение* (Tagged value)
- *Ограничение* (Constraint)

Основы структурного моделирования

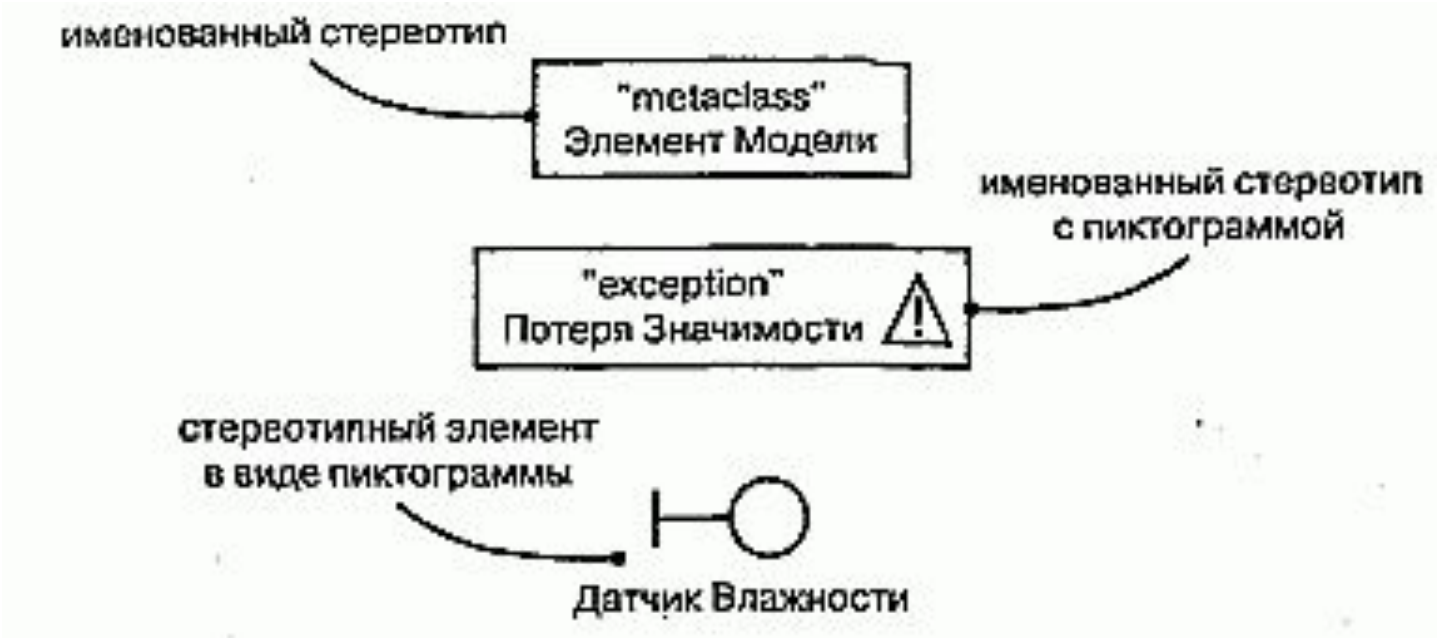
- *Примечание* (Note) - это графический символ, используемый для изображения ограничений или комментариев, присоединенных к элементу модели или их совокупности. Примечание выглядит как прямоугольник с загнутым углом, содержащий текстовый или графический комментарий



Основы структурного моделирования

- *Стереотипом* (Stereotype) называют расширение словаря UML, позволяющее создавать новые виды строительных блоков, аналогичные существующим, но специфичные для данной задачи. Стереотип представлен в виде имени, заключенного в кавычки и расположенного над именем другого элемента. Стереотипный элемент можно изображать также с помощью новой связанной с ним пиктограммы.

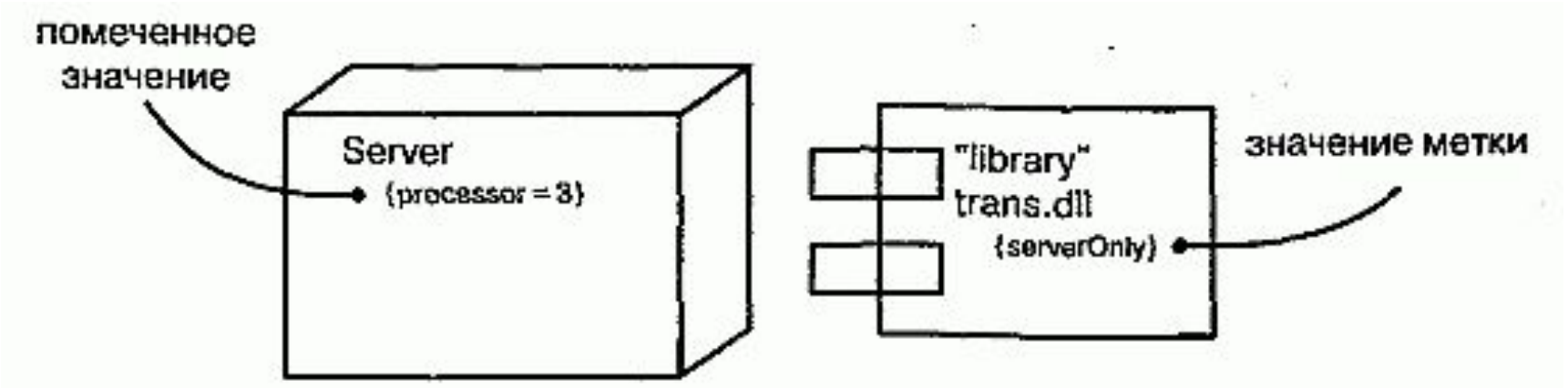
Основы структурного моделирования



Основы структурного моделирования

- *Помеченное значение* (Tagged value) - это расширение свойств элемента UML, позволяющее вводить новую информацию в его спецификацию. Помеченные значения изображаются в виде строки в скобках, расположенной под именем другого элемента.

Основы структурного моделирования



Основы структурного моделирования

- *Ограничение* (Constraint) - это расширение семантики элемента UML, позволяющее создавать новые или изменять существующие правила. Изображаются ограничения в виде строки в скобках, которая расположена возле ассоциированного элемента или связана с ним отношениями зависимости. Можно также представить ограничение в виде примечания.

Основы структурного моделирования