Савва Юрий Болеславович

Проектирование распределенных информационных систем

Лекция 3+.
Введение в язык UML
(Продолжение)

Моделирование системы предполагает идентификацию сущностей, важных с той или иной точки зрения. Эти сущности составляют словарь моделируемой системы.

В языке UML все сущности подобного рода моделируются как классы. Класс - это абстракция сущностей, являющихся частью вашего словаря. Класс представляет не индивидуальный объект, а целую их совокупность.

Многие языки программирования непосредственно поддерживают концепцию классов. В этом случае создаваемые абстракции могут быть непосредственно отображены в конструкциях языка программирования, даже если речь идет об абстракциях не программных сущностей типа "покупатель", "торговля" или "разговор".

Графическое изображение класса в UML показано на рис. 4.1. Такое обозначение позволяет визуализировать абстракцию независимо от конкретного языка программирования и подчеркнуть ее наиболее важные характеристики: имя, атрибуты и операции.

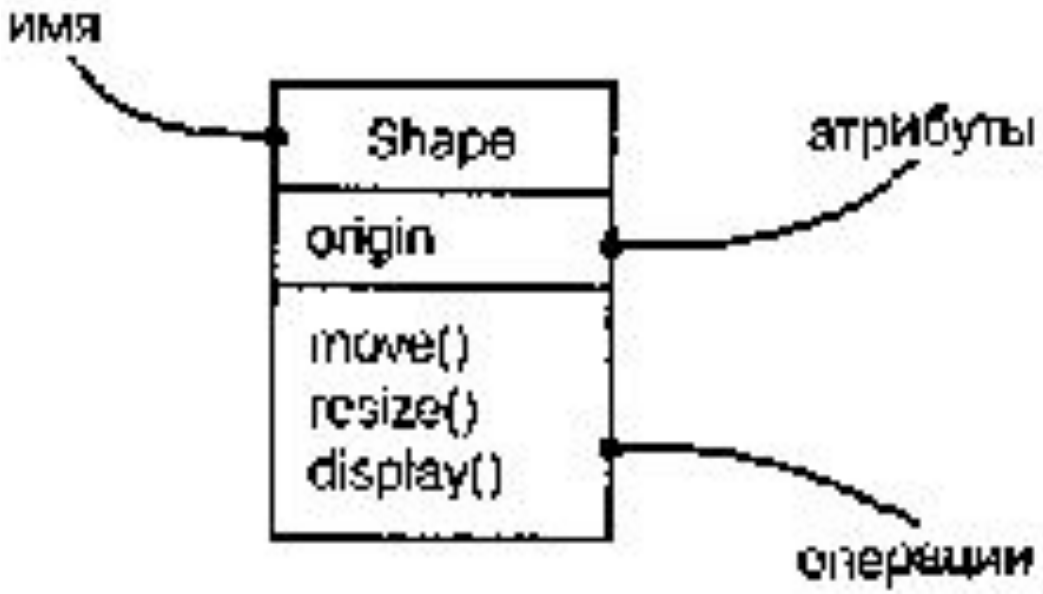


Рис. 4.1. Классы

Имена классов

- Укаждого класса должно быть имя, отличающее его от других классов. *Имя класса* - это текстовая строка. Взятое само по себе, оно называется *простым именем*; к *составному имени* спереди добавлено имя пакета, куда входит класс. Имя класса в объемлющем пакете должно быть уникальным.
- При графическом изображении класса показывается только его имя, как на рис. 4.2.

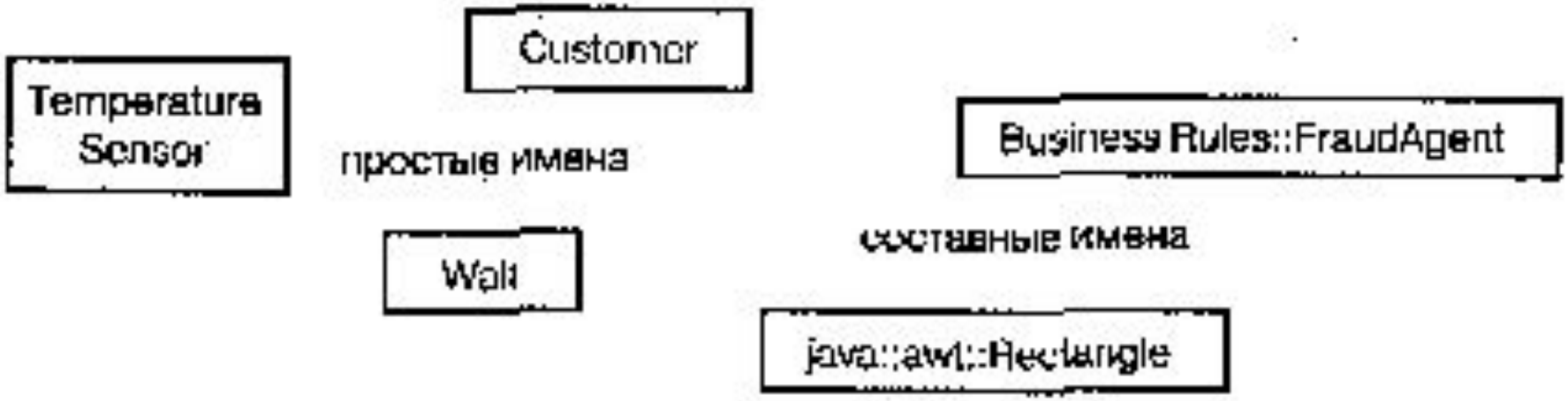


Рис. 4.2. Простые и составные имена

Имя класса может состоять из любого числа букв, цифр и ряда знаков препинания (за исключением таких, например, как двоеточие, которое применяется для отделения имени класса от имени объемлющего пакета). Имя может занимать несколько строк.

На практике для именования класса используют одно или несколько коротких существительных, взятых из словаря моделируемой системы. Обычно каждое слово в имени класса пишется с заглавной буквы, например: Customer(Клиент), Wall(Стена), TemperatureSensor(ДатчикТемпературы).

Атрибуты

- ***Атрибут*** **- это именованное свойство класса, включающее описание множества значений, которые могут принимать экземпляры этого свойства. Класс может иметь любое число атрибутов или не иметь их вовсе.**
- **Атрибут представляет некоторое свойство моделируемой сущности, общее для всех объектов данного класса. Например, у любой стены есть высота, ширина и толщина; при моделировании клиентов можно задавать фамилию, адрес, номер телефона и дату рождения. Таким образом, 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 этом указываются только их имена (рис. 4.3).**
- ***Имя атрибута, как и имя класса, может быть произвольной текстовой строкой. На практике для именования атрибута используют одно или несколько коротких существительных, соответствующих некоторому свойству объемлющего класса. Каждое слово в имени атрибута, кроме самого первого, обычно пишется с заглавной буквы, например*** *пате или loadBearing.*

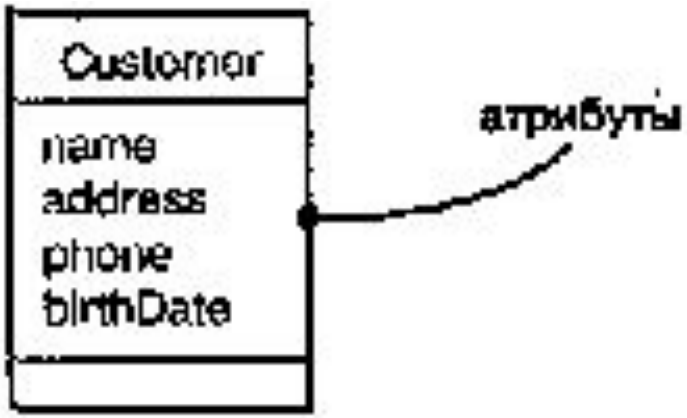


Рис. 4.3. Атрибуты

При описании атрибута можно явным образом указывать его класс и начальное значение, принимаемое по умолчанию, как продемонстрировано на рис. 4.4

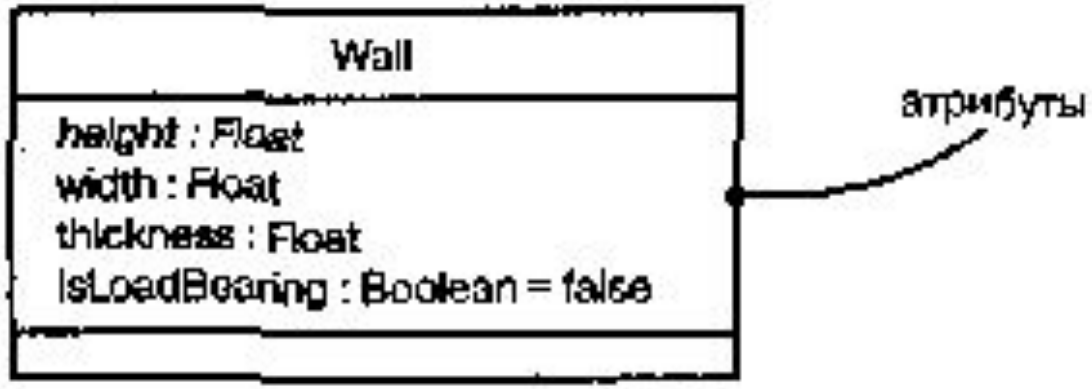


Рис. 4.4. Атрибуты и их класс

Операции

- *Операцией* называется реализация услуги, которую можно запросить у любого объекта класса для воздействия на поведение. Иными словами, операция - это абстракция того, что позволено делать с объектом. У всех объектов класса имеется общий набор операций. Класс может содержать любое число операций или не содержать их вовсе. Например, для всех объектов класса Rectangle(Прямоугольник) из библиотеки для работы с окнами, содержащейся в пакете awtязыка Java, определены операции перемещения, изменения размера и опроса значений свойств.
- Часто (хотя не всегда) обращение к операции объекта изменяет его состояние или его данные.
- Операции класса изображаются в разделе, расположенном ниже раздела с атрибутами. При этом можно ограничиться только именами, как показано на рис. 4.5.

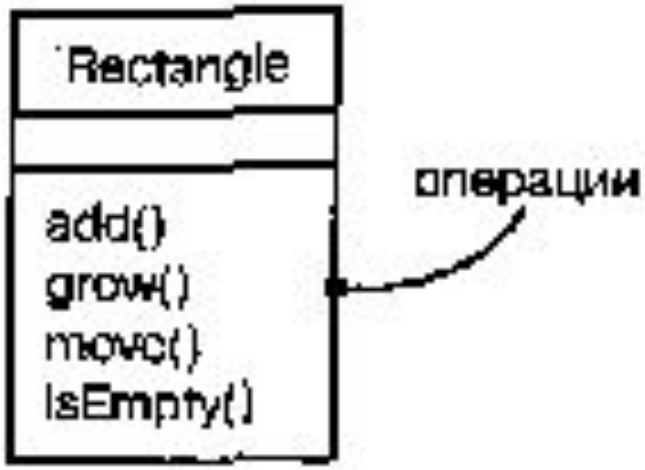


Рис. 4.5. Операции

Имя операции, как и имя класса, может быть произвольной текстовой строкой. На практике для именования операций используют короткий глагол или глагольный оборот, соответствующий определенному поведению объемлющего класса. Каждое слово в имени операции, кроме самого первого, обычно пишут с заглавной буквы, например move или isEmpty.

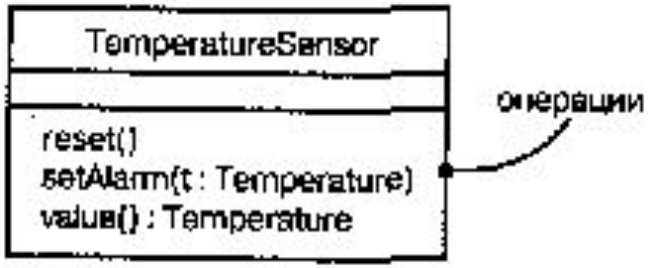


Рис. 4.6. Операции и их сигнатуры

Операцию можно описать более подробно, указав ее сигнатуру, в которую входят имена и типы всех параметров, их значения, принятые по умолчанию, а применительно к функциям - тип возвращаемого значения, как показано на рис. 4.6.

Организация атрибутов и операций

- При изображении класса необязательно сразу показывать все его атрибуты и операции. Как правило, это попросту невозможно - их чересчур много для одного рисунка, - да и не требуется (поскольку для данного представления системы лишь небольшое подмножество атрибутов и операций имеет значение).
- По этим причинам класс обычно сворачивают, то есть изображают лишь некоторые из имеющихся атрибутов и операций, а то и вовсе опускают их.
- Таким образом, пустой раздел в соответствующем месте прямоугольника может означать не отсутствие атрибутов или операций, а только то, что их не сочли нужным изобразить.
- Явным образом наличие дополнительных атрибутов или операций можно обозначить, поставив в конце списка многоточие.

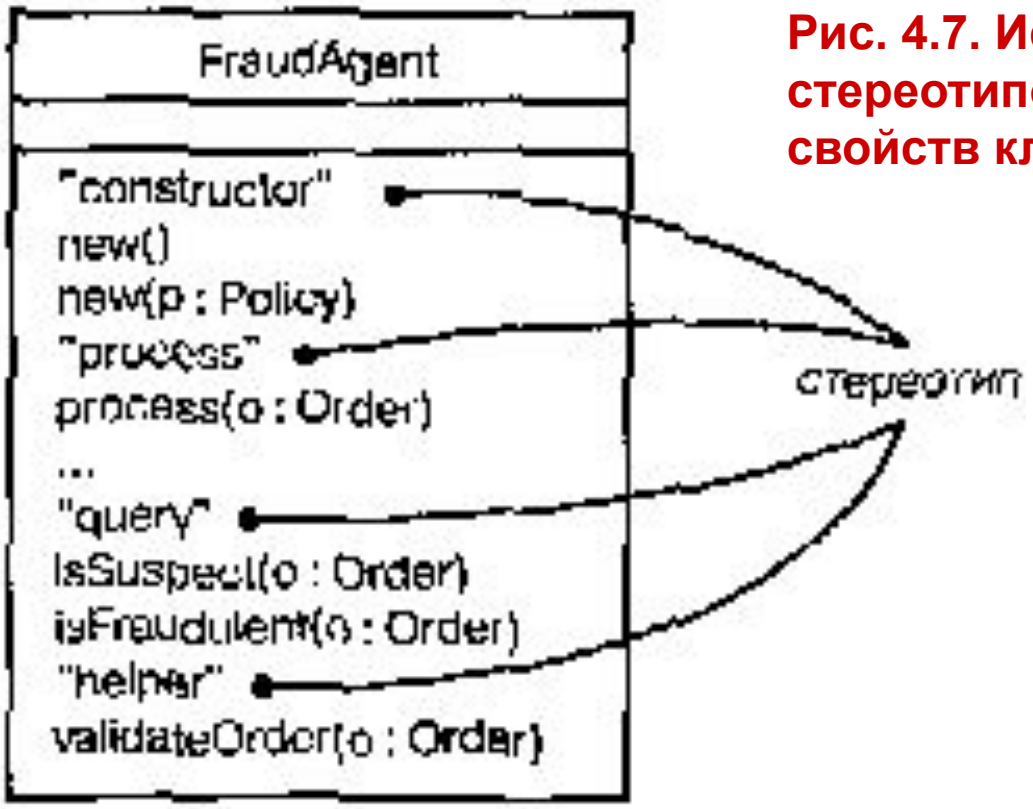


Рис. 4.7. Использование стереотипов для описания свойств класса

Для лучшей организации списков атрибутов и операций можно снабдить каждую группу дополнительным описанием, воспользовавшись стереотипами, как показано на рис. 4.7.

Обязанности

- *Обязанности* (Responsibilities) класса - это своего рода контракт, которому он должен подчиняться. Определяя класс, вы постулируете, что все его объекты имеют однотипное состояние и ведут себя одинаково. Выражаясь абстрактно, соответствующие атрибуты и операции как раз и являются теми свойствами, посредством которых выполняются обязанности класса.
- Например, класс TemperatureSensor(ДатчикТемпературы) отвечает за измерение температуры и подачу сигнала тревоги в случае превышения заданного уровня.

Графически обязанности изображают в особом разделе в нижней части пиктограммы класса (см. рис. 4.8).

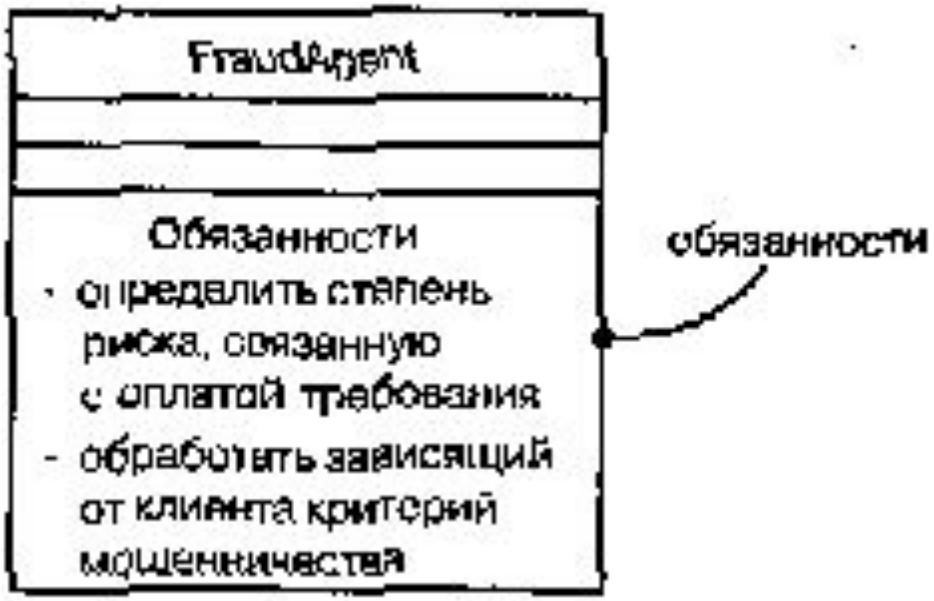


Рис. 4.8. Обязанности

Отношения

Существует три вида отношений, особенно важных для объектно-ориентированного моделирования:

1. *зависимости*, которые описывают существующие между классами отношения использования (включая отношения уточнения, трассировки и связывания);
2. *обобщения*, связывающие обобщенные классы со специализированными;
3. *ассоциации*, представляющие структурные отношения между объектами.

Каждый из этих типов позволяет по-разному комбинировать абстракции

- Отношения *зависимости* - это отношения использования. Например, трубы отопления зависят от нагревателя, подогревающего воду, которая в них течет.
- Отношения *обобщения* связывают общие классы со специализированными; здесь также применяются термины "субкласс/суперкласс" или "потомок/родитель". Например, *"фонарь"* - это окно с большой нераздвижной рамой, а патио - окно, раздвигающееся в стороны.
- Отношения *ассоциации* - структурные взаимосвязи между объектами: комнаты состоят из стен, в которые могут быть встроены двери и окна; иногда через стены проходят трубы отопления.

Эти три типа отношений охватывают большую часть способов взаимодействия элементов и, что неудивительно, хорошо отображаются на способы связи объектов, принятые в объектно-ориентированных языках программирования.

Для каждого из названных типов отношений язык UML предоставляет графическое изображение, как показано на рис. 5.1.

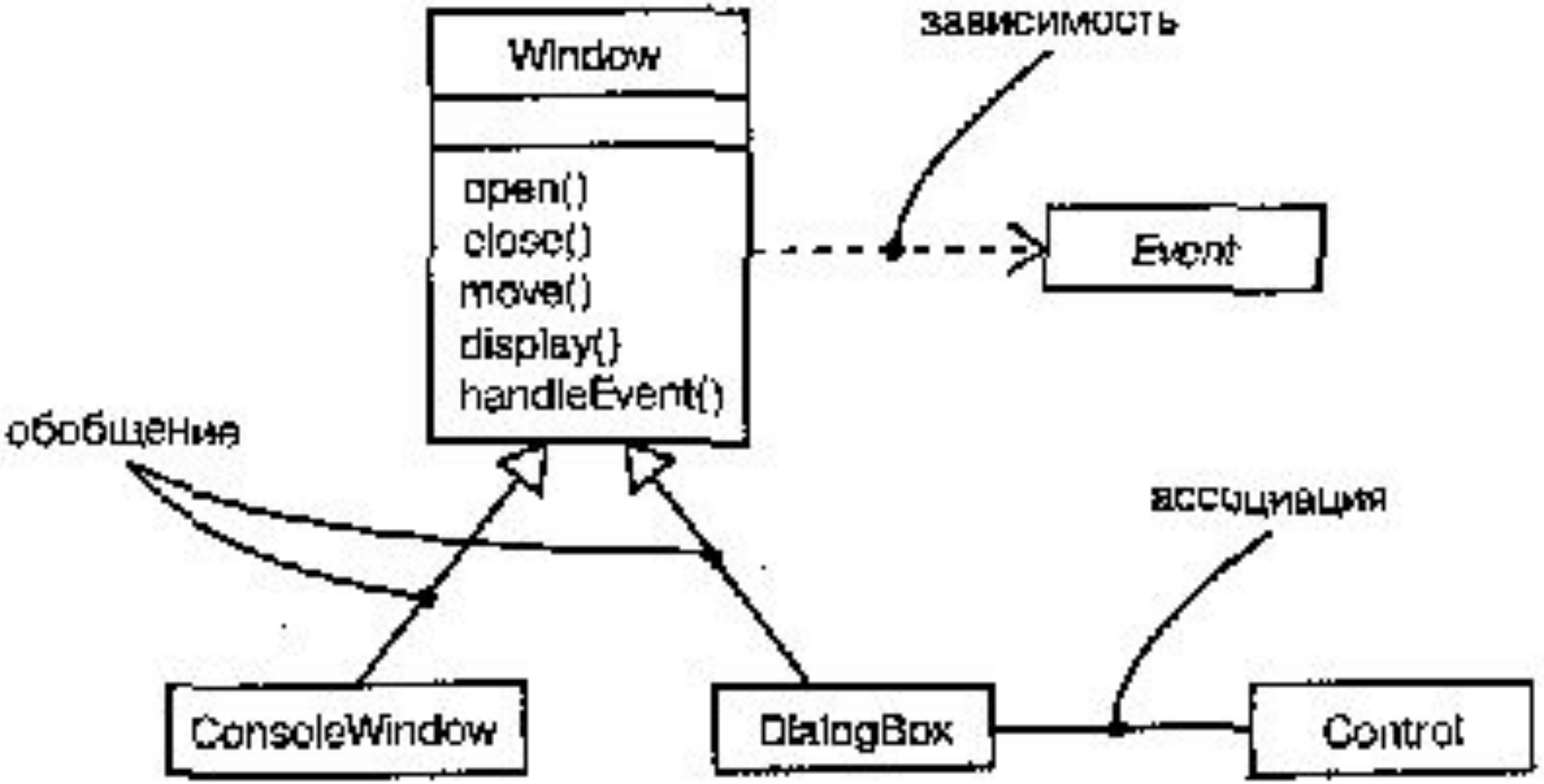


Рис. 5.1. Отношения

- *Отношением* (Relationship) называется связь между элементами.
- В объектно- ориентированном моделировании тремя самыми важными отношениями являются зависимости, обобщения и ассоциации.
- Графически отношение представлено линией, тип которой зависит от вида отношения.

- *Зависимостью* (Dependency) называют отношение использования, согласно которому изменение в спецификации одного элемента (например, класса Event) может повлиять на другой элемент, его использующий (в данном случае - класс Window), причем обратное не обязательно.
- Графически зависимость изображается пунктирной линией со стрелкой, направленной от данного элемента на тот, от которого он зависит. Используются зависимости, когда необходимо показать, что один элемент использует другой.

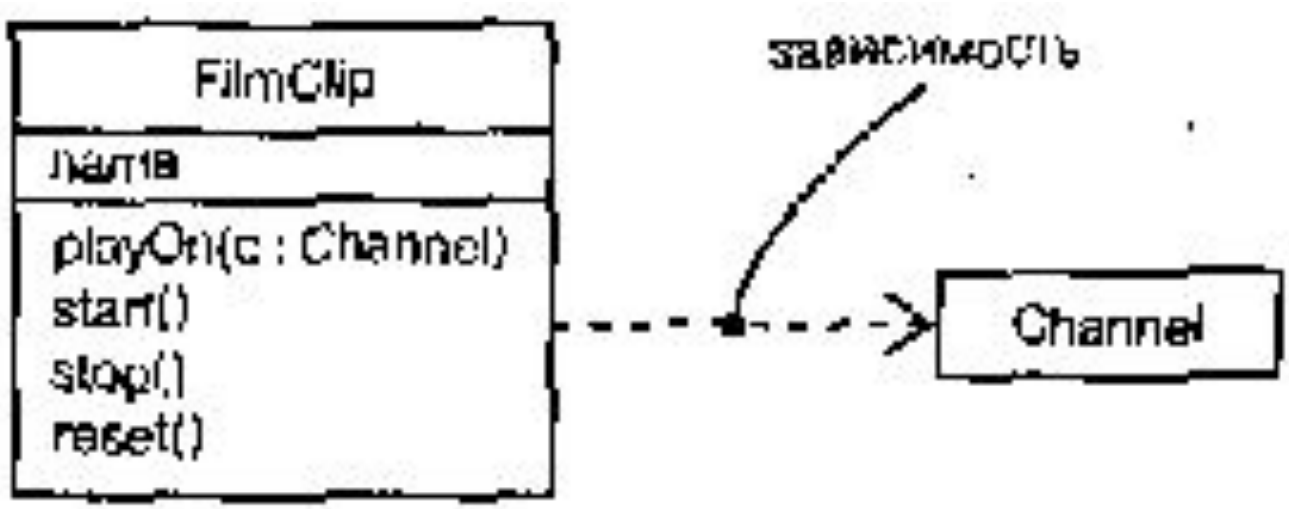


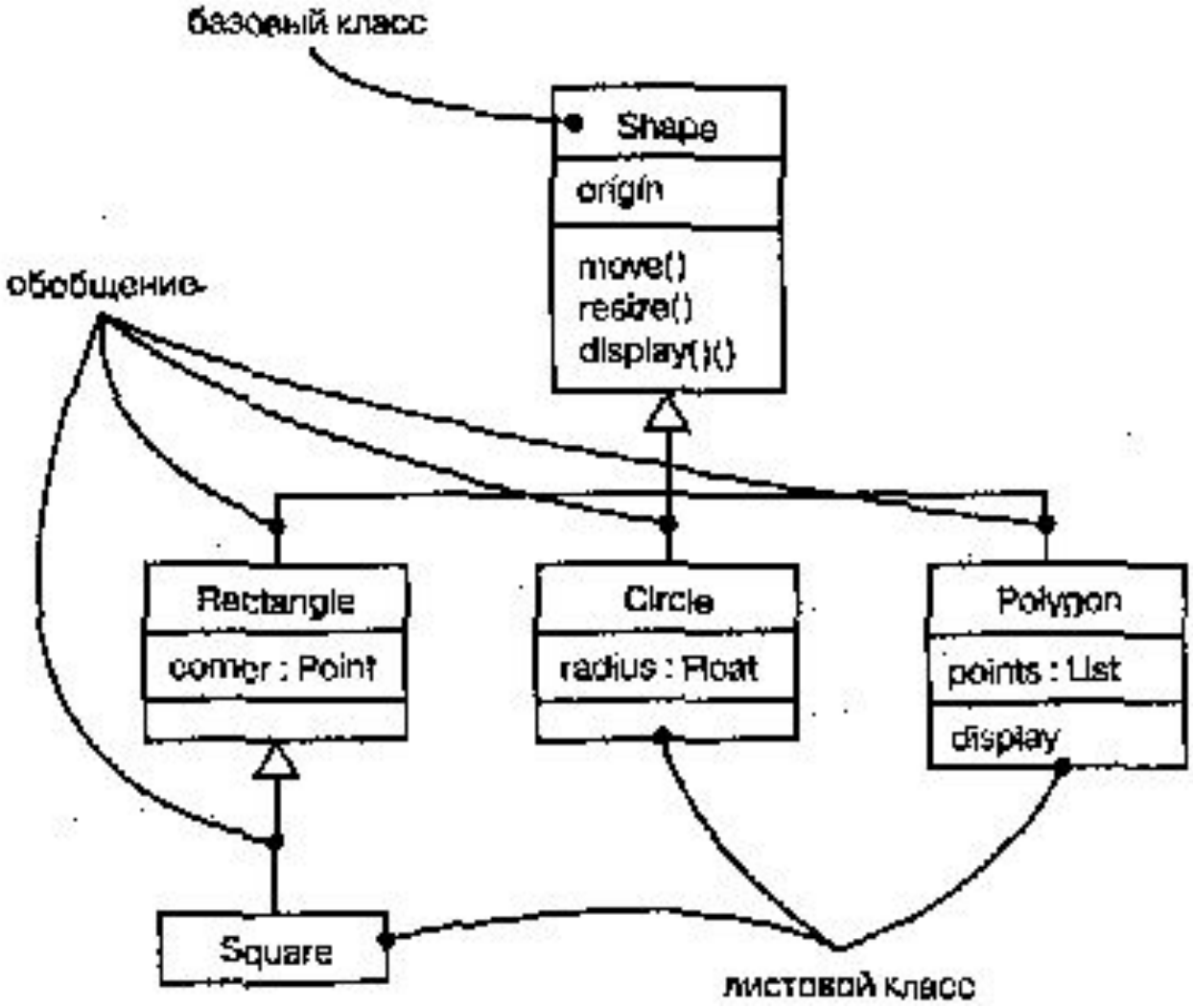
Рис. 5.2. Зависимости

У зависимости может быть собственное имя, хотя оно редко требуется - разве что в случае, когда модель содержит много зависимостей и возникает необходимость ссылаться на них или отличать их друг от друга. Чаще, однако, для различения зависимостей используют стереотипы

- *Обобщение* (Generalization) - это отношение между общей сущностью (суперклассом, или родителем) и ее конкретным воплощением (субклассом, или потомком).
- Обобщения иногда называют отношениями типа "является", имея в виду, что одна сущность (например, класс BayWindow) *является* частным выражением другой, более общей (скажем, класса Window).
- Обобщение означает, что объекты класса-потомка могут использоваться всюду, где встречаются объекты класса-родителя, но не наоборот. Другими словами, потомок может быть подставлен вместо родителя. При этом он наследует свойства родителя, в частности его атрибуты и операции.
- Часто, хотя и не всегда, у потомков есть и свои собственные атрибуты и операции, помимо тех, что существуют у родителя. Операция потомка с той же сигнатурой, что и у родителя, замещает операцию родителя; это свойство называют *полиморфизмом* (Polymorphism).
- Графически отношение обобщения изображается в виде линии с большой незакрашенной стрелкой, направленной на родителя, как показано на рис. 5.3. Применяют обобщения, когда необходимо показать отношения типа "родитель/потомок".

Рис. 5.3. Обобщение

Обобщение может обладать именем, хотя это требуется редко - лишь тогда, когда в модели много обобщений и вам нужно ссылаться на них или отличать друг от друга.

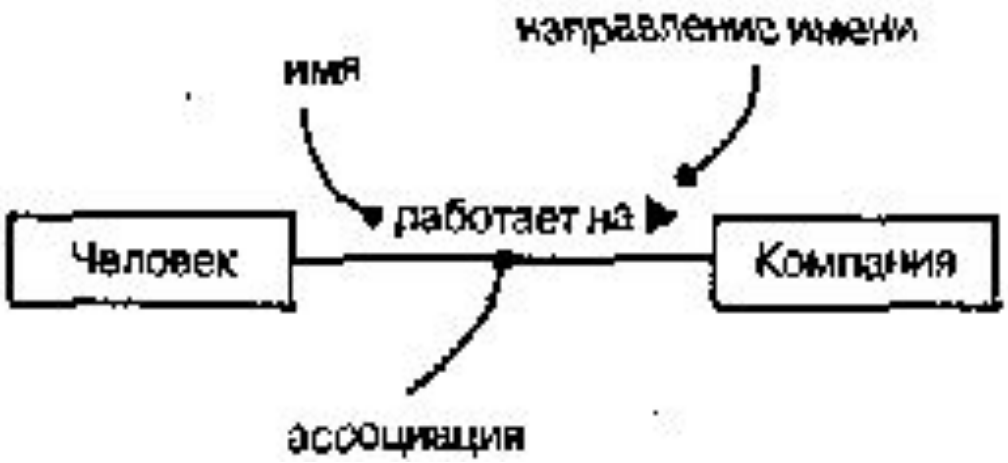


Обобщение чаще всего используют между классами и интерфейсами, чтобы показать отношения наследования. В UML можно создавать отношения обобщения и между другими элементами, в частности пакетами

- Класс может иметь одного или нескольких родителей или не иметь их вовсе.
- Класс, у которого нет родителей, но есть потомки, называется базовым (base) или корневым (root), а тот, у которого нет потомков, - листовым (leaf).
- О классе, у которого есть только один родитель, говорят, что он использует одиночное наследование (Single inheritance); если родителей несколько, речь идет о множественном наследовании (Multiple inheritance).

- *Ассоциацией* (Association) называется структурное отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа.
- Если между двумя классами определена ассоциация, то можно перемещаться от объектов одного класса к объектам другого.
- Вполне допустимы случаи, когда оба конца ассоциации относятся к одному и тому же классу. Это означает, что с объектом некоторого класса позволительно связать другие объекты из того же класса.
- Ассоциация, связывающая два класса, называется бинарной. Можно, хотя это редко бывает необходимым, создавать ассоциации, связывающие сразу несколько классов; они называются n-арными.
- Графически ассоциация изображается в виде линии, соединяющей класс сам с собой или с другими классами. Используют ассоциации, когда необходимо показать структурные отношения. (Ассоциации и зависимости, в отличие от отношений обобщения, могут быть рефлексивными) Помимо описанной базовой формы существует четыре дополнения, применимых к ассоциациям.

Рис. 5.4. Имена ассоциаций



Имя. Ассоциации может быть присвоено имя, описывающее природу отношения. Чтобы избежать возможных двусмысленностей в понимании имени, достаточно с помощью черного треугольника указать направление, в котором оно должно читаться, как показано на рис. 5.4. Не путайте направление чтения имени с навигацией по ассоц

Обычно имя ассоциации не указывается, если только вы не хотите явно задать для нее ролевые имена или в вашей модели настолько много ассоциаций, что возникает необходимость ссылаться на них и отличать друг от друга. Имя будет особенно полезным, если между одними и теми же классами существует несколько различных ассоциаций.

Роль. Класс, участвующий в ассоциации, играет в ней некоторую роль. По существу, это "лицо", которым класс, находящийся на одной стороне ассоциации, обращен к классу с другой ее стороны. Вы можете явно обозначить роль, которую класс играет в ассоциации. На рис. 5.5 показано, что класс Человек, играющий роль работника, ассоциирован с классом Компания, играющим роль работодателя. Роли тесно связаны с семантикой интерфейсов

Рис. 5.5. Роли

Один класс может играть в разных ассоциациях как одну и ту же роль, так и различные

Кратность. Ассоциации отражают структурные отношения между объектами.

Часто при моделировании бывает важно указать, сколько объектов может быть связано посредством одного экземпляра ассоциации. Это число называется кратностью (Multiplicity) роли ассоциации и записывается либо как выражение, значением которого является диапазон значений, либо в явном виде, как показано на рис. 5.6.

Указывая кратность на одном конце ассоциации, вы тем самым говорите, что на этом конце именно столько объектов должно соответствовать каждому объекту на противоположном конце. Кратность можно задать равной единице (1), можно указать диапазон: "ноль или единица" (0..1), "много" (0..*), "единица или больше" (1..*). Разрешается также указывать определенное число (например, 3).

Рис. 5.6. Кратность

С помощью списка можно задать и более сложные кратности, например 0..1, 3..4, 6.., что означает "любое число объектов, кроме 2 и 5".*

Агрегирование. Простая ассоциация между двумя классами отражает структурное отношение между равноправными сущностями, когда оба класса находятся на одном концептуальном уровне и ни один не является более важным, чем другой.

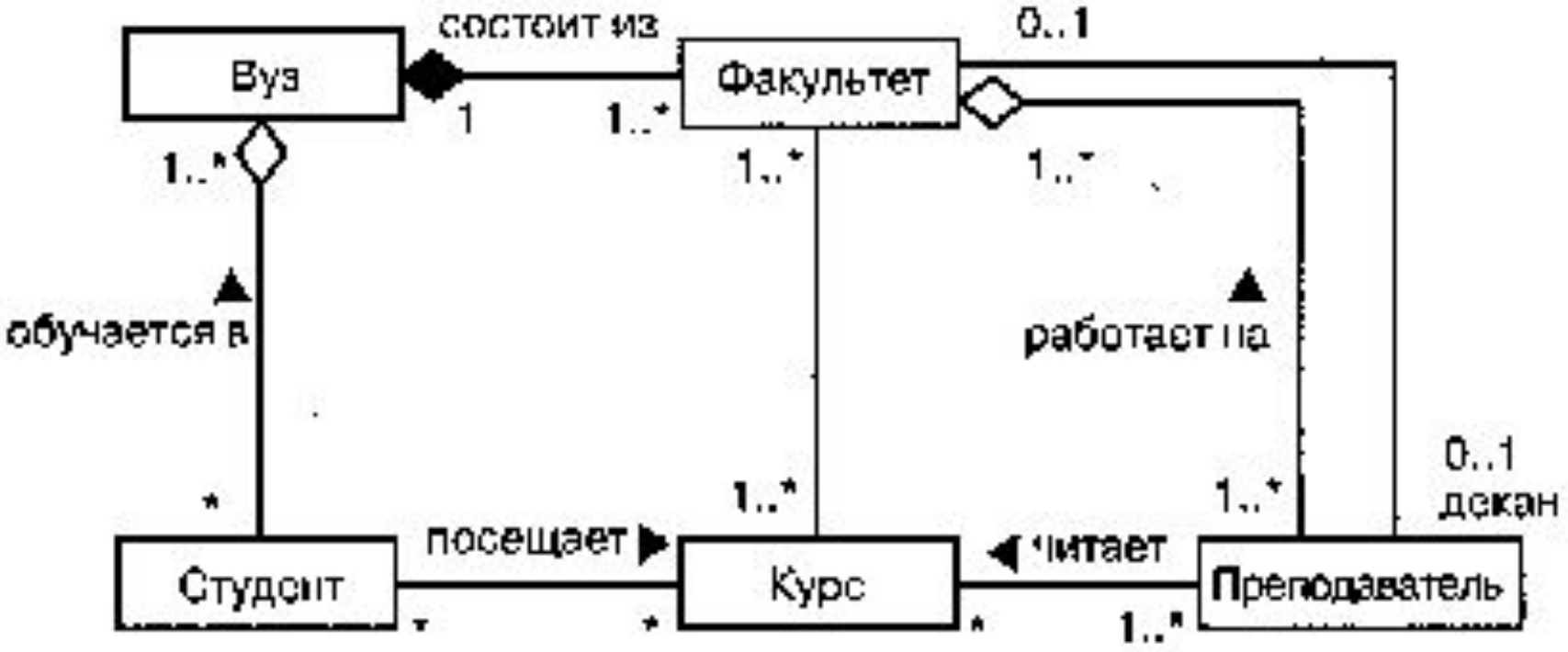
Но иногда приходится моделировать отношение типа "часть/целое", в котором один из классов имеет более высокий ранг (целое) и состоит из нескольких меньших по рангу (частей). Отношение такого типа называют агрегированием; оно причислено к отношениям типа "имеет" (с учетом того, что объект-целое *имеет* несколько объектов-частей).

Агрегирование является частным случаем ассоциации и изображается в виде простой ассоциации с незакрашенным ромбом со стороны "целого", как показано на рис. 5.7.

Рис. 5.7. Агрегирование отношения

Незакрашенный ромб отличает "целое" от "части" - и только.

Эта простая форма агрегирования является чисто концептуальной; она не влияет на результат навигации по ассоциации между целым и его частями и не подразумевает наличия между ними какой-либо зависимости по времени жизни.


Вуз
состоит из
1
1..*
Факультет
0..1
1..*
1..*
1..*
обучается в
работает на
*
1..*
1..*
0..1
декан
посещает
читает
Студент
Курс
Преподаватель
*
*
*
1..*

Советы

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";
- множественное наследование часто можно заменить агрегированием;
- остерегайтесь циклических отношений обобщения;
- 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.

При изображении отношений в UML руководствуйтесь нижеследующими рекомендациями:

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