Тема №5 Модели данных в нотации UML

1. Понятие UML
2. Классы, атрибуты и операции
3. Категории связей. Связь-зависимость
4. Связи-обобщения и механизм наследования классов в UML
5. Связи-ассоциации: роли, кратность, агрегация

Понятие UML

Язык объектно-ориентированного моделирования UML (Unified Modeling Language) разработан и развивается консорциумом OMG (Object Management Group) и имеет много общего с объектными моделями, на которых основана технология распределенных объектных систем CORBA, и объектной моделью ODMG (Object Data Management Group).

UML позволяет моделировать разные виды систем: программные, аппаратные, программно-аппаратные, смешанные, явно включающие деятельность людей и т. д.

UML активно применяется для проектирования реляционных БД. Для этого используется небольшая часть языка (*диаграммы классов*)

Классы, атрибуты и операции

Диаграммой классов в терминологии *UML* называется диаграмма, на которой показан набор *классов*, а также связей между этими *классами*.

Кроме того, *диаграмма классов* может включать комментарии и ограничения.

Ограничения могут неформально задаваться на естественном языке или же могут формулироваться на *языке объектных ограничений OCL (Object Constraints Language)*.

Классы, атрибуты и операции

Классом называется именованное описание совокупности объектов с общими *атрибутами*, *операциями*, связями и семантикой.

Графически *класс* изображается в виде прямоугольника. У каждого *класса* должно быть имя (текстовая строка), уникально отличающее его от всех других *классов*.

При формировании имен *классов* в *UML* на практике рекомендуется использовать короткие и осмысленные прилагательные и существительные, каждое из которых начинается с заглавной буквы.

Классы, атрибуты и операции

Примеры описания *классов*:

Атрибутом класса называется именованное *свойство класса*, описывающее множество значений, которые могут принимать экземпляры этого *свойства*.

Класс может иметь любое число *атрибутов* (в частности, не иметь ни одного *атрибута*).

Классы, атрибуты и операции

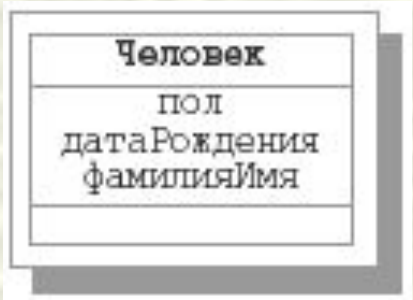
Свойство, выражаемое *атрибутом*, является свойством моделируемой сущности, общим для всех объектов данного *класса*.

Любой *атрибут* любого объекта *класса* должен иметь некоторое значение. Имена *атрибутов* представляются в разделе *класса*, расположенном под именем *класса*.

Для имен атрибутов рекомендуется использовать короткие прилагательные и существительные, отражающие смысл соответствующего *свойства класса*. Первое слово в имени *атрибута* рекомендуется писать с прописной буквы, а все остальные слова – с заглавной.

Классы, атрибуты и операции

Пример описания *класса* с указанными *атрибутами*:



Операцией класса называется именованная услуга, которую можно запросить у любого объекта этого *класса*.

Операция – это абстракция того, что можно делать с объектом. *Класс* может содержать любое число *операций* (в частности, не содержать ни одной *операции*). Набор *операций класса* является общим для всех объектов данного *класса*.

Классы, атрибуты и операции

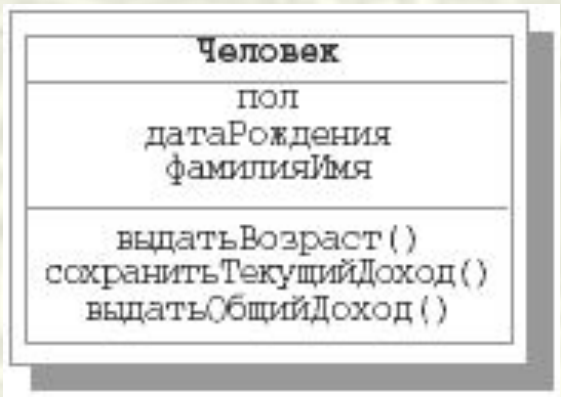
Операции класса определяются в разделе, расположенном ниже раздела с *атрибутами*. При этом можно ограничиться только указанием имен *операций*, оставив детальную спецификацию выполнения *операций* на более поздние этапы моделирования.

Для именования *операций* рекомендуется использовать глагольные формы, соответствующие ожидаемому поведению объектов данного *класса*.

Описание *операции* может также содержать ее ***сигнатуру***, т.е. имена и типы всех параметров, а если *операция* является функцией, то и тип ее значения.

Классы, атрибуты и операции

Класс Человек с определенными *операциями*:



Для *класса* Человек мы определили три *операции*: выдатьВозраст, сохранитьТекущийДоход, выдатьОбщийДоход.

Состояние объекта меняется при выполнении только второй *операции*. Результаты первой и третьей *операций* формируются на основе текущего состояния объекта.

Категории связей. Связь-зависимость

В *диаграмме классов* могут участвовать связи трех разных категорий: *зависимость (dependency)*, *обобщение (generalization)* и *ассоциация (association)*.

При проектировании реляционных БД наиболее важны вторая и третья категории связей, поэтому *связи-зависимости* будем рассматривать в общем виде.

Категории связей. Связь-зависимость

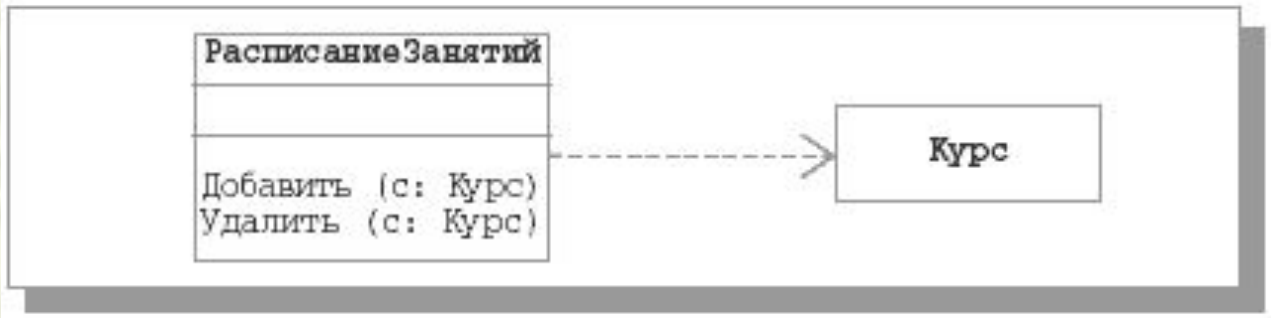
Зависимостью называют *связь* по применению, когда изменение в спецификации одного *класса* может повлиять на поведение другого *класса*, использующего первый *класс*.

Чаще всего *зависимости* применяют в *диаграммах классов*, чтобы отразить в *сигнатуре операции* одного *класса* тот факт, что параметром этой *операции* могут быть объекты другого *класса*.

Очевидно, что если интерфейс второго *класса* изменяется, это влияет на поведение объектов первого *класса*.

Категории связей. Связь-зависимость

Пример *диаграммы классов* со *связью-зависимостью:*



Зависимость показывается прерывистой линией со стрелкой, направленной к *классу*, от которого имеется зависимость.

Связи-зависимости существенны для объектно-ориентированных систем (в том числе и для ООБД). При проектировании реляционных БД непонятно, что делать с зависимостями (как воспользоваться этой информацией в реляционной БД?).

Связи-обобщения и механизм наследования классов в UML

Связью-обобщением называется *связь* между общей сущностью, называемой ***суперклассом***, или родителем, и более специализированной разновидностью этой сущности, называемой ***подклассом***, или потомком.

Обобщения иногда называют ***связями «is a»***, имея в виду, что *класс*-потомок является частным случаем *класса*-предка.

Класс-потомок наследует все *атрибуты* и *операции класса*-предка, но в нем могут быть определены дополнительные *атрибуты* и *операции*.

Связи-обобщения и механизм наследования классов в UML

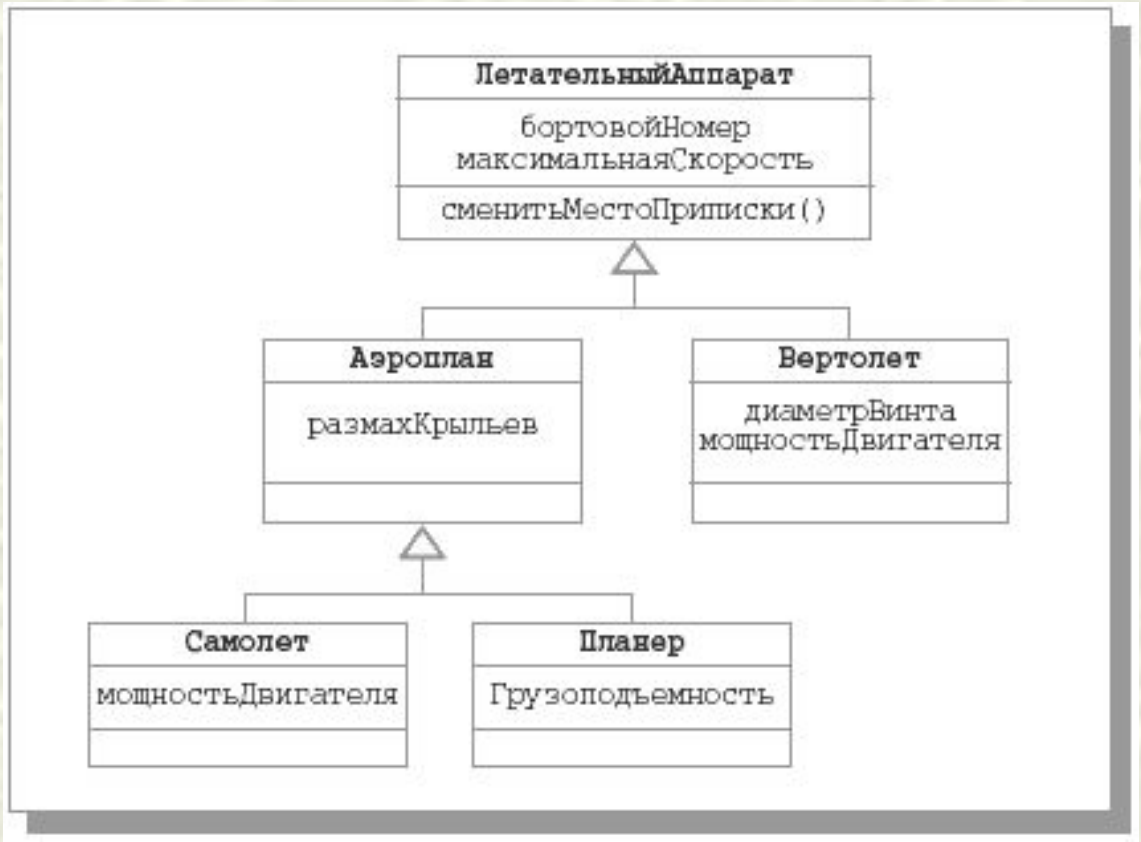
Объекты *класса*-потомка могут использоваться везде, где могут использоваться объекты *класса*-предка.

Это свойство называют ***полиморфизмом по включению***, имея в виду, что объекты потомка можно считать включаемыми во множество объектов *класса*-предка.

Графически *обобщения* изображаются в виде сплошной линии с большой незакрашенной стрелкой, направленной к *суперклассу*.

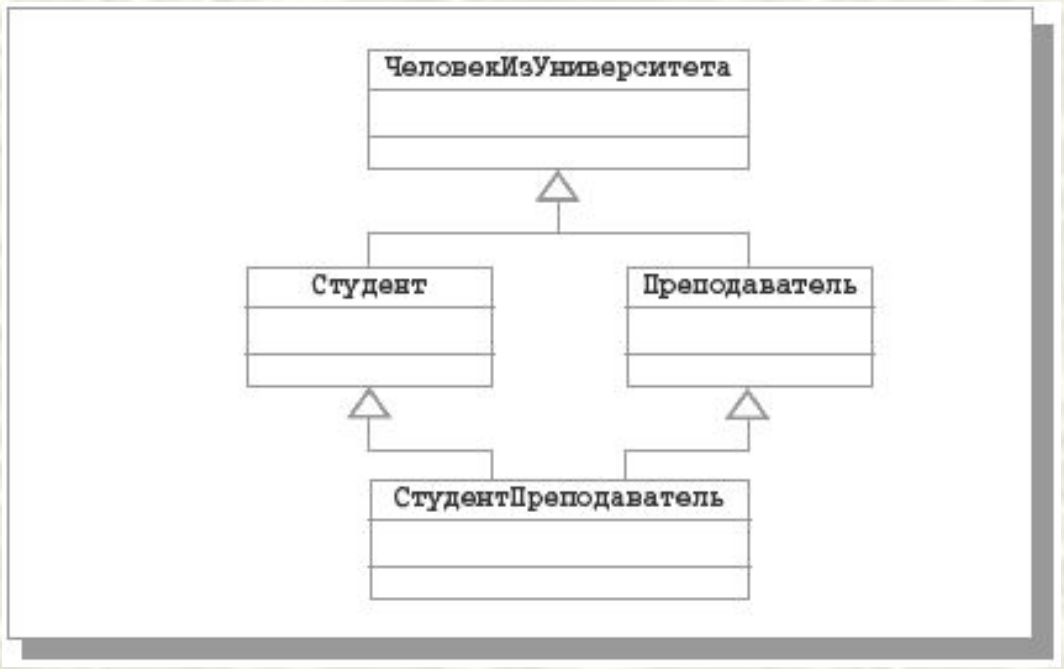
Связи-обобщения и механизм наследования классов в UML

Пример иерархии одиночного наследования классов:



Связи-обобщения и механизм наследования классов в UML

Одиночное наследование является достаточным в большинстве случаев применения *связи-обобщения*. Однако в *UML* допускается и *множественное наследование*, когда один *подкласс* определяется на основе нескольких *суперклассов*:



Связи-обобщения и механизм наследования классов в UML

Множественное наследование порождает ряд проблем, из которых одной из наиболее известных является проблема именования *атрибутов* и *операций* в *подклассе*, полученном путем *множественного наследования*.

Пример: при образовании *подклассов* Студент и Преподаватель в них обоих был определен *атрибут* с именем номерКомнаты. Для объектов *класса* Студент значениями этого *атрибута* будут номера комнат в студенческом общежитии, а для объектов *класса* Преподаватель – номера служебных кабинетов.

Как быть с объектами *класса* СтудентПреподаватель, для которых существенны оба одноименных *атрибута* (у студента-преподавателя могут иметься и комната в общежитии, и служебный кабинет)?

Связи-обобщения и механизм наследования классов в UML

На практике применяется одно из следующих решений:

- запретить образование *подкласса* СтудентПреподаватель, пока в одном из *суперклассов* не будет произведено переименование *атрибута* номерКомнаты;

- наследовать это *свойство* только от одного из *суперклассов*, так что, например, значением *атрибута* номерКомнаты у объектов *класса* СтудентПреподаватель всегда будут номера служебных кабинетов;

- унаследовать в *подклассе* оба *свойства*, но автоматически переименовать оба *атрибута*, чтобы прояснить их смысл; назвать их, например, номерКомнатыСтудента и номерКомнатыПреподавателя.

Связи-обобщения и механизм наследования классов в UML

Ни одно из решений не является полностью удовлетворительным.

Первое решение требует возврата к ранее определенному *классу*, имена *атрибутов* и *операций* которого, возможно, уже используются в приложениях.

Второе решение нарушает логику наследования, не давая возможности на уровне *подкласса* использовать все *свойства суперклассов*.

Третье решение заставляет использовать длинные имена *атрибутов* и *операций*, которые могут стать недопустимо длинными, если процесс *множественного наследования* будет продолжаться от полученного *подкласса*.

Связи-обобщения и механизм наследования классов в UML

Сложность проблемы именования *атрибутов* и *операций* несопоставимо меньше сложности реализации *множественного наследования* в реляционных БД.

При использовании *UML* для проектирования реляционных БД нужно очень осторожно использовать наследование *классов* вообще и стараться избегать *множественного наследования*.

Связи-ассоциации: роли, кратность, агрегация

Ассоциацией называется структурная *связь*, показывающая, что объекты одного *класса* некоторым образом связаны с объектами другого или того же самого *класса*.

Допускается, чтобы оба конца *ассоциации* относились к одному *классу*. В *ассоциации* могут связываться два *класса*, и тогда она называется бинарной. Допускается создание *ассоциаций*, связывающих сразу n *классов* (они называются ***n-арными ассоциациями***).

Графически *ассоциация* изображается в виде линии, соединяющей *класс* сам с собой или с другими *классами*.

Связи-ассоциации: роли, кратность, агрегация

С понятием *ассоциации* связаны четыре важных дополнительных понятия: *имя*, роль, *кратность* и агрегация.

Ассоциации может быть присвоено *имя*, характеризующее природу связи.

Смысл *имени* уточняется с помощью черного треугольника, который располагается над линией связи справа или слева от *имени ассоциации*. Этот треугольник указывает направление чтения имени связи.

Связи-ассоциации: роли, кратность, агрегация

Пример именованной ассоциации:

Треугольник показывает, что именованная *ассоциация* должна читаться как «Студент учится в Университете».

Связи-ассоциации: роли, кратность, агрегация

Другим способом именования *ассоциации* является указание *роли* каждого *класса*, участвующего в этой *ассоциации*.

Роль класса, как и имя конца связи в ER-модели, задается именем, помещаемым под линией *ассоциации* ближе к данному *классу*.

Связи-ассоциации: роли, кратность, агрегация

В примере показаны две *ассоциации* между *классами* Человек и Университет, в которых эти *классы* играют разные *роли*.

Объекты *класса* Человек могут выступать в *роли* РАБОТНИКОВ при участии в *ассоциации*, в которой объекты *класса* Университет играют *роль* НАНИМАТЕЛЯ.

В другой *ассоциации* объекты *класса* Человек играют *роль* СТУДЕНТА, а объекты *класса* УНИВЕРСИТЕТ – *роль* ОБУЧАЮЩЕГО.

Связи-ассоциации: роли, кратность, агрегация

В общем случае, для *ассоциации* могут задаваться и ее собственное *имя*, и имена *ролей классов*.

Это связано с тем, что *класс* может играть одну и ту же *роль* в разных *ассоциациях*, так что в общем случае пара имен *ролей классов* не идентифицирует *ассоциацию*.

С другой стороны, в простых случаях, когда между двумя *классами* определяется только одна *ассоциация*, можно вообще не связывать с ней дополнительные имена.

Связи-ассоциации: роли, кратность, агрегация

Кратностью (multiplicity) роли ассоциации называется характеристика, указывающая, сколько объектов *класса* с данной *ролью* может или должно участвовать в каждом *экземпляре ассоциации*.

Наиболее распространенным способом задания *кратности роли ассоциации* является указание конкретного числа или диапазона.

Связи-ассоциации: роли, кратность, агрегация

Например, указание «1» говорит о том, что каждый объект *класса* с данной *ролью* должен участвовать в некотором *экземпляре* данной *ассоциации*, причем в каждом *экземпляре ассоциации* может участвовать ровно один объект *класса* с данной *ролью*.

Указание диапазона «0..1» говорит о том, что не все объекты *класса* с данной *ролью* обязаны участвовать в каком-либо *экземпляре* данной *ассоциации*, но в каждом *экземпляре ассоциации* может участвовать только один объект.

Связи-ассоциации: роли, кратность, агрегация

Указание диапазона «1..*» говорит о том, что все объекты *класса* с данной *ролью* должны участвовать в некотором *экземпляре* данной *ассоциации*, и в каждом *экземпляре ассоциации* должен участвовать хотя бы один объект (верхняя граница не задана).

Толкование диапазона «0..*» является очевидным расширением случая «0..1».

Связи-ассоциации: роли, кратность, агрегация

В более сложных (но крайне редко встречающихся на практике) случаях определения *кратности* можно использовать списки диапазонов.

Например, список «2, 4..6, 8..*» говорит о том, что все объекты *класса* с указанной *ролью* должны участвовать в некотором *экземпляре* данной *ассоциации*, и в каждом *экземпляре ассоциации* должны участвовать два, от четырех до шести или более семи объектов *класса* с данной *ролью*.

Связи-ассоциации: роли, кратность, агрегация

На *диаграмме классов* показано, что произвольное (может быть, нулевое) число людей являются сотрудниками произвольного числа университетов:

Каждый университет обучает произвольное (может быть, нулевое) число студентов, но каждый студент может быть студентом только одного университета.

Связи-ассоциации: роли, кратность, агрегация

Обычная *ассоциация* между двумя *классами* характеризует *связь* между равноправными сущностями: оба *класса* находятся на одном концептуальном уровне.

Иногда в *диаграмме классов* требуется отразить тот факт, что *ассоциация* между двумя *классами* имеет специальный вид «часть-целое».

В этом случае *класс* «целое» имеет более высокий концептуальный уровень, чем *класс* «часть». *Ассоциация* такого рода называется ***агрегатной***. Графически *агрегатные ассоциации* изображаются в виде простой *ассоциации* с незакрашенным ромбом на стороне *класса*-«целого».

Связи-ассоциации: роли, кратность, агрегация

Простой пример *агрегатной ассоциации*:

Объектами *класса* Аудитория являются студенческие аудитории, в которых проходят занятия. В каждой аудитории должны быть установлены парты. Поэтому *класс* Парта является «частью» *класса* Аудитория.

Роль класса Парта является необязательной.

Связи-ассоциации: роли, кратность, агрегация

Бывают случаи, когда связь «части» и «целого» настолько сильна, что уничтожение «целого» приводит к уничтожению всех его «частей».

Агрегатные ассоциации, обладающие таким свойством, называются ***композитными***, или просто *композициями*. При наличии *композиции* объект-часть может быть частью только одного объекта-целого (композита).

При обычной *агрегатной ассоциации* «часть» может одновременно принадлежать нескольким «целым». Графически *композиция* изображается в виде простой *ассоциации*, дополненной закрашенным ромбом со стороны «целого».

Связи-ассоциации: роли, кратность, агрегация

Пример композитной агрегатной ассоциации:

Любой факультет является частью одного университета, и ликвидация университета приводит к ликвидации всех существующих в нем факультетов (хотя во время существования университета отдельные факультеты могут ликвидироваться и создаваться).

Связи-ассоциации: роли, кратность, агрегация

При наличии простой *ассоциации* между двумя *классами* (например, *ассоциации* между *классами* Студент и Университет предполагается возможность *навигации* между объектами, входящими в один *экземпляр ассоциации*.

Если известен конкретный объект-студент, то должна обеспечиваться возможность узнать соответствующий объект-университет. Если известен конкретный объект-университет, то должна обеспечиваться возможность узнать все соответствующие объекты-студенты.

Если не оговорено иное, то *навигация по ассоциации* может проводиться в обоих направлениях

Связи-ассоциации: роли, кратность, агрегация

Иногда желательно ограничить направление *навигации* для некоторых *ассоциаций*. В этом случае на линии *ассоциации* ставится стрелка, указывающая направление *навигации*:

Связи-ассоциации: роли, кратность, агрегация

В библиотеке должно содержаться некоторое количество книг, и каждая книга должна принадлежать некоторой библиотеке.

С точки зрения библиотечного хозяйства разумно иметь возможность найти книгу в библиотеке, т. е. произвести *навигацию* от объекта-библиотеки к связанным с ним объектам-книгам.

Однако вряд ли потребуется по данному экземпляру книги узнать, в какой библиотеке она находится.