Анализ и проектирование на UML

Максим Валерьевич Хлопотов

Темы лекционных занятий

1. Введение в UML
2. Моделирование использования
3. **Моделирование структуры**
4. Моделирование поведения
5. Дисциплина моделирования

Назначение структурного моделирования

Следующие структуры нужно моделировать на UML:

- структура связей между объектами во время выполнения программы;
- структура хранения данных;
- структура программного кода;
- структура компонентов в приложении;
- структура используемых вычислительных ресурсов;
- структура сложных объектов, состоящих из взаимодействующих частей
- структура артефактов в проекте.

Диаграммы классов

- Диаграмма классов является основным средством моделирования структуры UML.
- Диаграммы классов наиболее информационно насыщены по сравнению с другими типами канонических диаграмм UML.
- На диаграммах классов в качестве сущностей применяются прежде всего классы, как в своей наиболее общей форме, так и в форме многочисленных стереотипов и частных случаев: интерфейсы, типы данных, процессы и др.
- Кроме того, в диаграмме классов могут использоваться (как и везде) пакеты и примечания.

Классы и объекты

- Структуру из данных и процедур их обработки, существующую в памяти компьютера во время выполнения программы, чаще всего называют *объектом*, а описание множества однотипных объектов к тексте программы называют *классом*.
- Объект – структура из данных и процедур их обработки, существующая в памяти компьютера во время выполнения программы.
- Класс – описание множества однотипных объектов к тексте программы.

Диаграммы классов

- Сущности на диаграммах классов связываются главным образом отношениями ассоциации (в том числе агрегирования и композиции) и обобщения.
- Отношения зависимости и реализации на диаграммах классов используются для связи классов с интерфейсами.

Диаграммы классов

- Класс — один из самых "богатых" элементов моделирования UML.
- Описание класса может включать множество различных элементов, и чтобы они не путались, в языке предусмотрено группирование элементов описания класса по разделам.

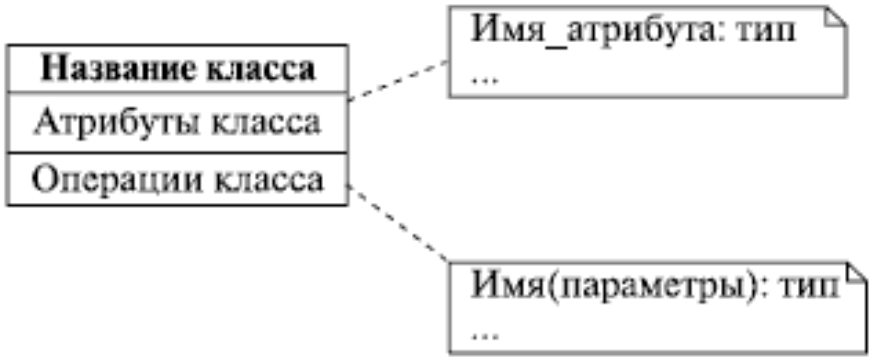
Стандартных разделов три:

- раздел имени — наряду с обязательным именем может содержать также стереотип, кратность и список свойств;
- раздел атрибутов — содержит список описаний атрибутов класса;
- раздел операций — содержит список описаний операций класса.

Класс

Атрибут - это свойство класса, которое может принимать множество значений.

Операция - реализация функции, которую можно запросить у любого объекта класса.



Атрибут

- В общем случае описание атрибута имеет следующий синтаксис:
- видимость ИМЯ кратность : тип = начальное_значение {свойства}

- Видимость, как обычно, обозначается знаками +, –, #. Если видимость не указана, то никакого значения видимости по умолчанию не подразумевается.
- «+» public (открытый доступ)
- «#» protected (только из операций этого же класса и классов, создаваемых на его основе)
- «-» private (только из операций того же класса)

Атрибут

- В обычной ситуации каждый экземпляр класса хранит свое индивидуальное значение атрибута. Если имя атрибута подчеркнуто, то это означает, что областью действия данного атрибута является класс, а не экземпляр класса, как обычно.

- Кратность, если она присутствует, определяет данный атрибут как массив (определенной или неопределенной длины).

- Тип атрибута — это либо примитивный (встроенный) тип, либо тип определенный пользователем

Атрибут

- Начальное значение имеет очевидный смысл: при создании экземпляра данного класса атрибут получает указанное значение. Заметим, что если начальное значение не указано, то никакого значения по умолчанию не подразумевается. Если нужно, чтобы атрибут обязательно имел значение, то об этом должен позаботиться конструктор класса.

- Как и любой другой элемент модели, атрибут может быть наделен дополнительными свойствами в форме ограничений и именованных значений.

Примеры описаний атрибутов

- name
- Минимальное возможное описание — указано только имя атрибута

- +name
- Указаны имя и открытая видимость — предполагается, манипуляции с именем будут производится непосредственно

- -name : String
- Указаны имя, тип и закрытая видимость — манипуляции с именем будут производится с помощью специальных операций

- -name [1..3] : String
- Указана кратность (для хранения трех составляющих; фамилии, имени и отчества)

- -name : String = "Hlopotov"
- Указано начальное значение

Операции

- Выполнение действий, определяемых операцией, инициируется вызовом операции.
- При выполнении операция может, в свою очередь, вызывать операции этого и других классов.
- Описания операций класса перечисляются в разделе операций и имеют следующий синтаксис.
- видимость ИМЯ (параметры) : тип {свойства}
- Здесь слово параметры обозначает последовательность описаний параметров операции. Описания параметров в списке разделяются запятой. Для каждого параметра обязательно указывается имя, а также могут быть указаны направление передачи параметра, его тип и значение аргумента по умолчанию.

Примеры описания операций

- move
- Минимальное возможное описание — указано только имя операции

- +move(in from, in to)
- Указаны видимость операции, направления передачи и имена параметров

- +move(in from : Dpt, in to : Dpt)
- Подробное описание сигнатуры: указаны видимость операции, направления передачи, имена и типы параметров

- +getName() : String {isQuery}
- Функция, возвращающая значение атрибута и не имеющая побочных эффектов

- +setPwd(in pwd : String = "password")
- Процедура, для которой указано значение аргумента по умолчанию

Класс

- Как и все основные сущности UML, класс обязательно имеет имя, а стало быть раздел имени не может быть опущен.
- В разделе имени класса может быть указан стереотип.
- Примеры стереотипов:

enumeration - перечислимый тип данных

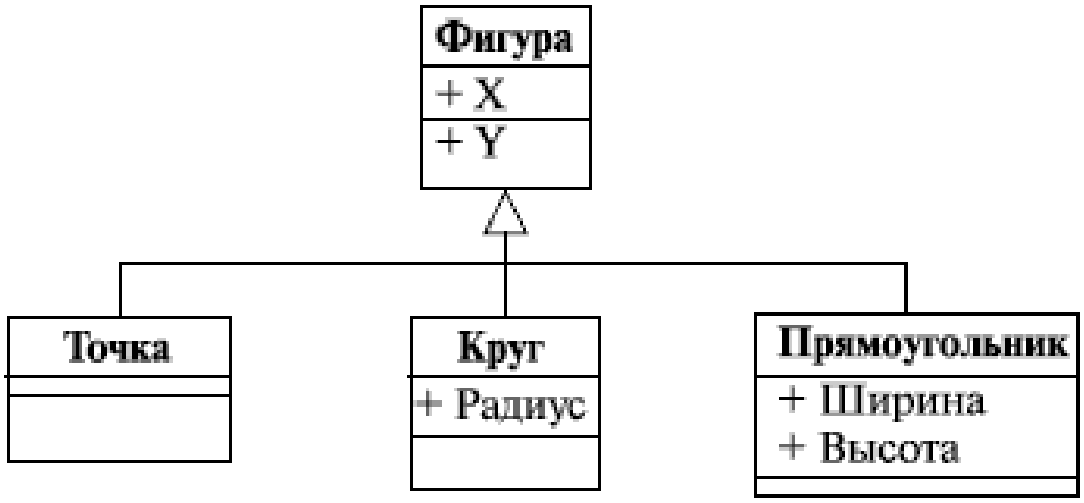
interface – нет атрибутов и все операции абстрактные

type (datatype) – тип данных

entity – хранимая сущность

Отношения между классами

Обобщение - это отношение между более общей сущностью, называемой *суперклассом*, и ее конкретным воплощением, называемым *подклассом*.



Отношения между классами

Ассоциация – связь между объектами, по которой можно между ними перемещаться.

Ассоциация может иметь имя, показывающее природу отношений между объектами, при этом в имени может указываться *направление* чтения связи при помощи треугольного маркера. Однонаправленная ассоциация изображается стрелкой.

Отношения между классами

Роли и кратность

Отношения между классами

Ассоциацией с агрегированием – более сложное отношение между классами, связь типа «часть-целое». Один класс имеет более высокий статус (целое) и состоит из низших по статусу классов (частей).

Выделяют простое и композитное агрегирование (**агрегация** и **композиция)**.

Агрегация предполагает, что части, отделенные от целого, могут продолжать свое существование независимо от него.
Композиция – целое владеет своими частями и их время жизни соответствует времени жизни целого, т. е. независимо от целого части существовать не могут.

Отношения между классами

Класс

- Описание классов и отношений между ними является основным средством моделирования структуры в UML.
- Нет универсального и применимого во всех случаях способа для выделения классов, подлежащих описанию.
- Три приема выделения классов, самых простых, а потому самых действенных и широко применимых:
  - словарь предметной области (набор основных понятий данной предметной области);
  - реализация вариантов использования;
  - образцы проектирования (стандартное решение типичной задачи в конкретном контексте).

Словарь предметной области

- *Словарь (концептуальные классы) предметной области* — это набор основных понятий (сущностей) данной предметной области.
- Для того, чтобы составить словарь, необходимо погрузиться в предметную область. Для анализа предметной области необходимо попытаться выделить объекты следующих типов.

Словарь предметной области

- Транзакции.
- Службы, связанные с транзакциями.
- Места записи транзакций.
- Каталоги и контейнеры других объектов (физических или информационных).
- Системы, внешние по отношению к данной системе.
- Роли людей или организации, связанные с транзакциями. Исполнители прецедентов.
- Места транзакций.
- Важные события, для которых необходимо хранить время и место
- Описания объектов
- Руководства, документы, статьи, книги, на которые ссылаются в процессе работы.

Реализация вариантов использования

- Один полезный и очень простой прием идентификации концептуальных классов на основе анализа текста.
- В реализации варианта использования выделить все существительные, они и будут кандидатами в концептуальные классы (или в атрибуты классов).
- Разумеется, после этой простой операции к полученному списку нужно применить фильтр здравого смысла и опыта, отсекая ненужное.

Идентификация классов

- Рассмотрим пример. Дано описание:

- *Информационная система «Отдел кадров» (сокращенно ОК) предназначена для ввода, хранения и обработки информации о сотрудниках и движении кадров. Система должна обеспечивать выполнение следующих основных функций:*
- *Прием, перевод и увольнение сотрудников.*
- *Создание и ликвидация подразделений.*
- *Создание вакансий и сокращение должностей.*

Идентификация классов

- Выпишем существительные в том порядке, как они встречаются, но без повторений: прием; перевод; увольнение; сотрудник; создание; ликвидация; подразделение; вакансия; сокращение; должность.
- Заметим, что некоторые их этих слов по сути являются названиями действий (и по форме являются отглагольными существительными).
- Отбросим замаскированные глаголы. Таким образом, остается список из четырех элементов: сотрудник; подразделение; вакансия; должность.

Идентификация классов

- При этом вакансия — это должность в особом состоянии.
- Таким образом, это слово в списке лишнее и у нас остались три кандидата, которые мы оставляем в словаре (и заодно присваиваем им английские идентификаторы).
- Сотрудник (Person);
- Подразделение (Department);
- Должность (Position).

Идентификация классов

- *Образец проектирования — это стандартное решение типичной задачи в* конкретном контексте.
- Синтаксически образец в UML является кооперацией, роли объектов в которой рассматриваются как параметры, а аргументами являются конкретные классы модели.

Идентификация классов

- Возможно, наиболее типичной ошибкой при создании модели предметной области является причисление некоторого объекта к атрибутам, в то время как он должен относиться к концептуальным классам.
- Чтобы избежать этой ошибки, следует придерживаться простого правила: если некоторый объект X в реальном мире не является числом или текстом, значит, это скорее концептуальный класс, чем атрибут.

Идентификация классов

- Например, является ли store (магазин) атрибутом объекта Sale (Продажа) или отдельным концептуальным классом Store?
- В реальном мире магазин не является числом или текстом, он представляет реальную сущность, организацию, занимающую некоторое место. Следовательно, Store нужно рассматривать в качестве концептуального класса.

Ассоциация на диаграмме классов

- Отношение ассоциации является самым важным на диаграмме классов.
- В общем случае ассоциация, которая обозначается сплошной линией, соединяющей классы, означает, что экземпляры одного класса связаны с экземплярами другого класса. Поскольку экземпляров может быть много, и каждый может быть связан с несколькими, ясно, что ассоциация является дескриптором, который описывает множество связанных объектов.
- В UML ассоциация является классификатором, экземпляры которого называются связями.

Ассоциация на диаграмме классов

- Связь между объектами в программе может быть организована самыми разными способами. Например, в объекте одного класса может хранится указатель на объект другого класса. Другой вариант: объект одного класса является контейнером для объектов другого класса.
- Связь не обязательно является непосредственно хранимым физическим адресом. Этот адрес может динамически вычисляться во время выполнения программы на основании другой информации. Например, если объекты представлены как записи в таблице базы данных, то связь означает, в записи одного объекта имеется поле, значением которого является первичный ключ записи другого объекта (из другой таблицы).

Ассоциация на диаграмме классов

- При моделировании на UML техника реализации связи между объектами не имеет значения. Ассоциация в UML подразумевает лишь то, что связанные объекты обладают достаточной информацией для организации взаимодействия. Возможность взаимодействия означает, что объект одного класса может послать сообщение объекту другого класса, в частности, вызвать операцию или же прочитать или изменить значение открытого атрибута.
- Поскольку в объектно-ориентированной программе такого рода действия и составляют суть выполнения программы, моделирование структуры взаимосвязей объектов (т. е. выявление ассоциаций) является одной из ключевых задач при разработке.

Ассоциация на диаграмме классов

- Базовая нотация ассоциации (сплошная линия) позволяет указать, что объекты ассоциированных классов могут взаимодействовать во время выполнения. Но это только малая часть того, что можно моделировать с помощью отношения ассоциации. Для ассоциации в UML предусмотрено наибольшее количество различных дополнений, которые мы сначала перечислим, а потом рассмотрим по порядку.
- Дополнения, как обычно, не являются обязательными: их используют при необходимости, в различных ситуациях по разному. Если использовать все дополнения сразу, то диаграмма становится настолько перегруженной, что ее трудно читать.

Ассоциация на диаграмме классов

Для ассоциации определены следующие дополнения:

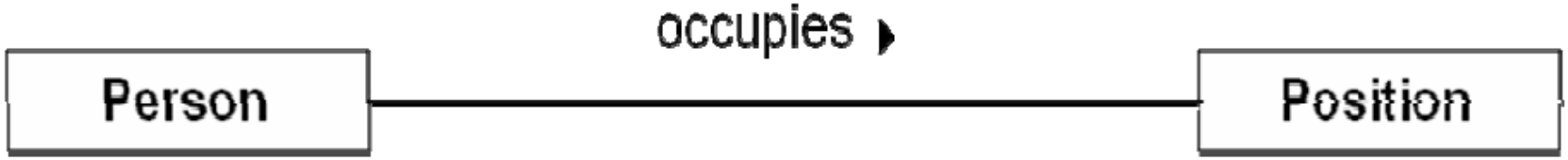
- имя ассоциации (возможно, вместе с направлением чтения);
- кратность полюса ассоциации (полюсом называется конец линии ассоциации. Обычно используются двухполюсные ассоциации, но могут быть и многополюсные);
- вид агрегации полюса ассоциации;
- роль полюса ассоциации;
- направление навигации полюса ассоциации;
- упорядоченность объектов на полюсе ассоциации;
- изменяемость множества объектов на полюсе ассоциации;
- квалификатор полюса ассоциации;
- класс ассоциации;
- видимость полюса ассоциации;
- многополюсные ассоциации.

Ассоциация на диаграмме классов

- *Имя ассоциации* указывается в виде строки текста над (или под, или рядом с) линией ассоциации. Имя не несет дополнительной семантической нагрузки, а просто позволяет различать ассоциации в модели.
- Обычно имя не указывают, за исключением многополюсных ассоциаций или случая, когда одна и та же группа классов связана несколькими различными ассоциациями.
- Например, в информационной системе отдела кадров, если сотрудник занимает должность, то соответствующие объектов классов Person и Position должны быть связаны.
- Дополнительно можно указать направление чтения имени ассоциации.

Ассоциация на диаграмме классов

- Фрагмент графической модели фактически можно прочитать вслух: Person occupies position

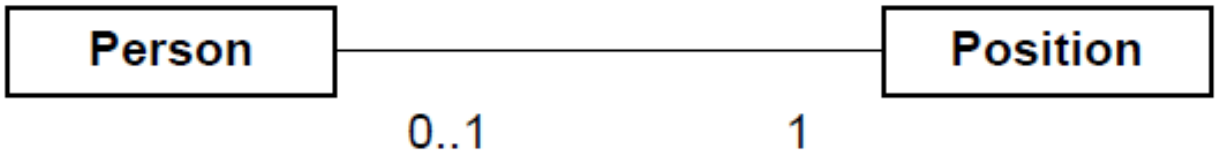


Ассоциация на диаграмме классов

- *Кратность полюса ассоциации указывает, сколько объектов данного класса (со* стороны данного полюса) участвуют в связи. Кратность может быть задана как конкретное число, и тогда в каждой связи со стороны данного полюса участвуют ровно столько объектов, сколько указано. Более распространен случай, когда кратность указывается как диапазон возможных значений, и тогда число объектов, участвующих в связи должно находится в пределах указанного диапазона. При указании кратности можно использовать символ *, который обозначает неопределенное число.

Ассоциация на диаграмме классов

- Например, если в информационной системе отдела кадров не предусматривается дробление ставок и совмещение должностей, то (работающему) сотруднику соответствует одна должность, а должности соответствует один сотрудник или ни одного (должность вакантна).



Кратность задана диапазоном

Кратность задана числом

Ассоциация на диаграмме классов

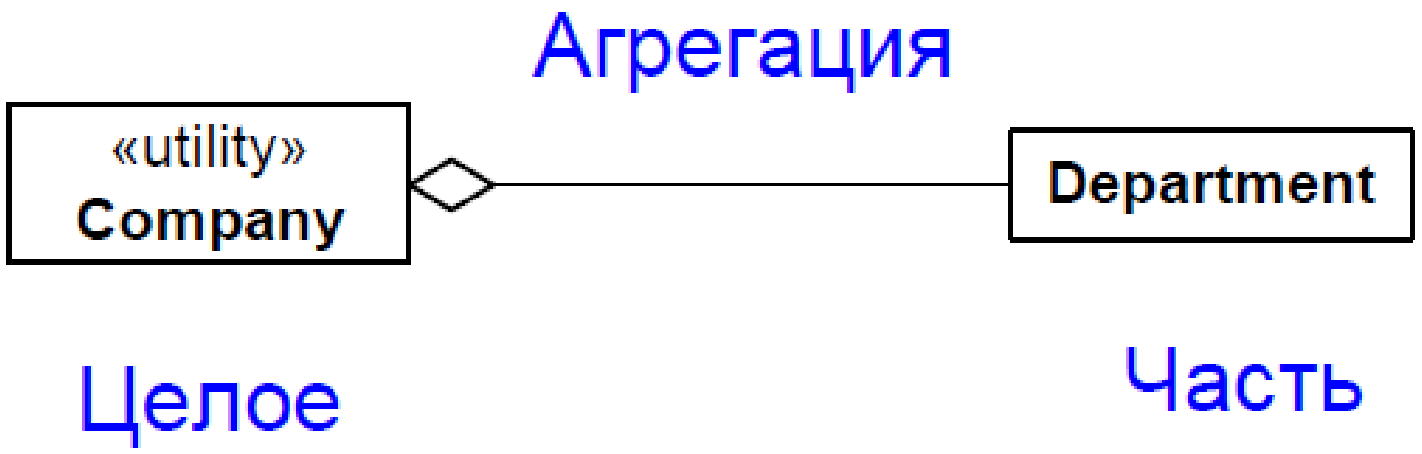
- Более сложные случаи также легко моделируются с помощью кратности полюсов.
- Например, если мы хотим предусмотреть совмещение должностей и хранить информацию даже о неработающих сотрудниках, то диаграмма примет следующий вид (запись * эквивалентна записи 0..*).

Ассоциация на диаграмме классов

- *Агрегация и композиция. В UML используются два частных, но очень важных* случая отношения ассоциации, которые называются агрегацией и композицией. В обоих случаях речь идет о моделировании отношения типа «часть – целое». Ясно, что отношения такого типа следует отнести к отношениям ассоциации, поскольку части и целое обычно взаимодействуют.
- *Агрегация от класса A к классу B означает, что объекты (один или несколько) класса* A входят в состав объекта класса B.

Ассоциация на диаграмме классов

- Это отмечается с помощью специального графического дополнения: на полюсе ассоциации, присоединенному к «целому», т. е., в данном случае, к классу B, изображается ромб.

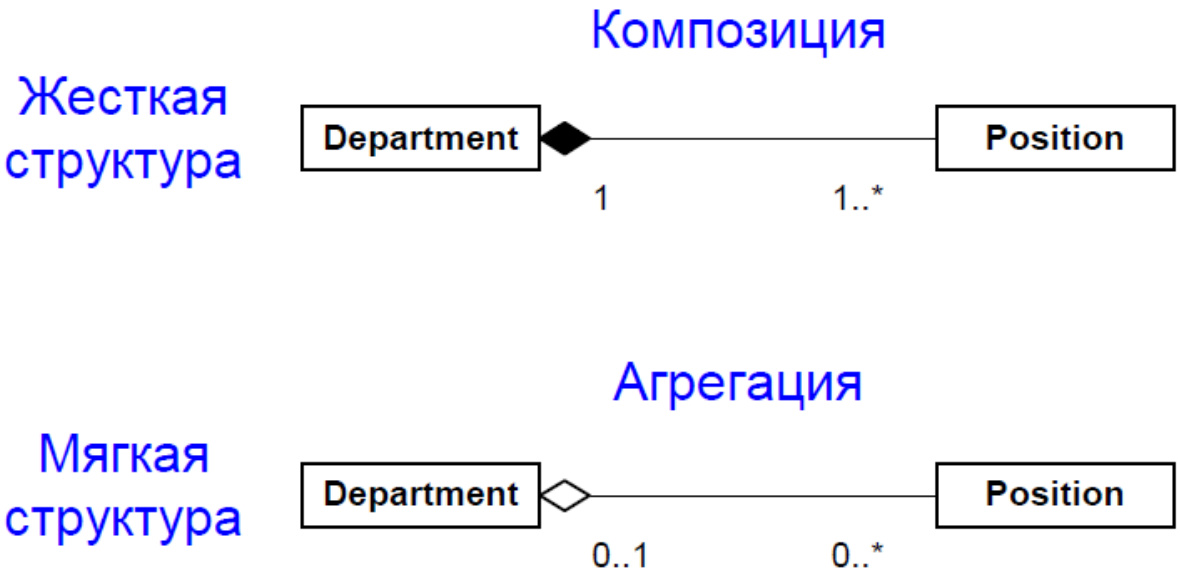


Ассоциация на диаграмме классов

- При этом никаких дополнительных ограничений не накладывается: объект класса A (часть) может быть связан отношениями агрегации с другими объектами (т. Е. участвовать в нескольких агрегациях), создаваться и уничтожаться независимо от объекта класса B (целого).
- *Композиция накладывает более сильные ограничения: композиционно часть может* входить только в одно целое, часть существует только пока существует целое и прекращает свое существование вместе с целым.
- Графически отношение композиции отображается закрашенным ромбом.

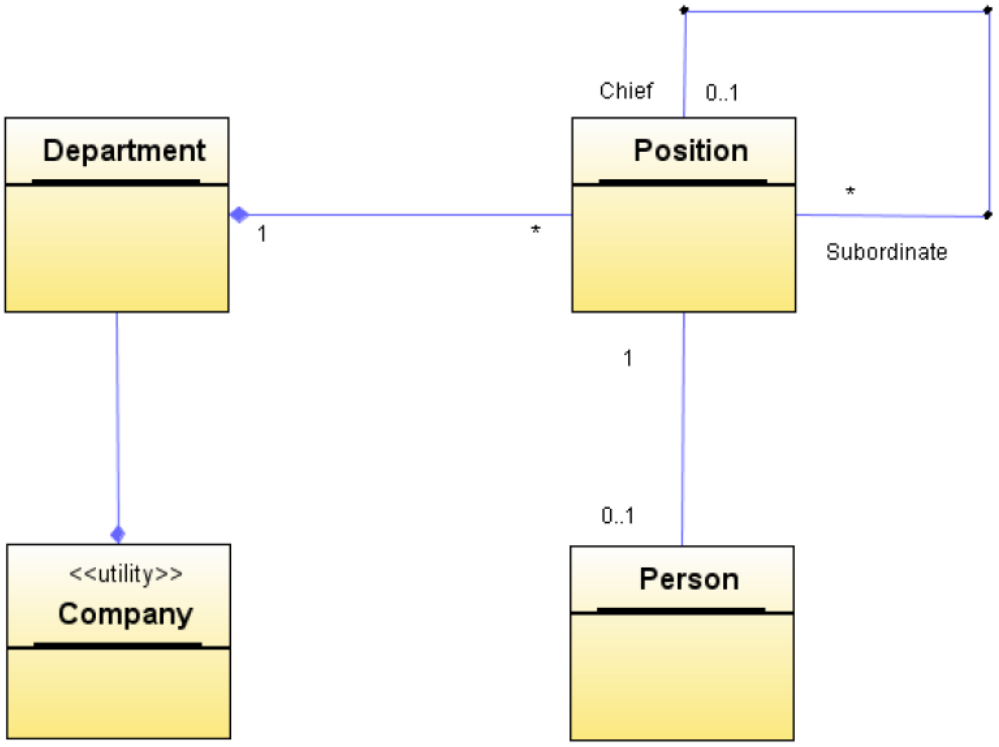
Ассоциация на диаграмме классов

- В первом случае (вверху), мы считаем, что в организации принята жесткая структура: каждая должность входит ровно в одно подразделение, в каждом подразделении есть по меньшей мере одна должность (начальник). Во втором случае (внизу) структура организации более аморфна: возможны «висящие в воздухе» должности, бывают «пустые» подразделения и т. д.



Ассоциация на диаграмме классов

- В комбинации с указанием кратности, отношения ассоциации, агрегации и композиции позволяют лаконично и полно отобразить структуру связей объектов: что из чего состоит и как связано.



Ассоциация на диаграмме классов

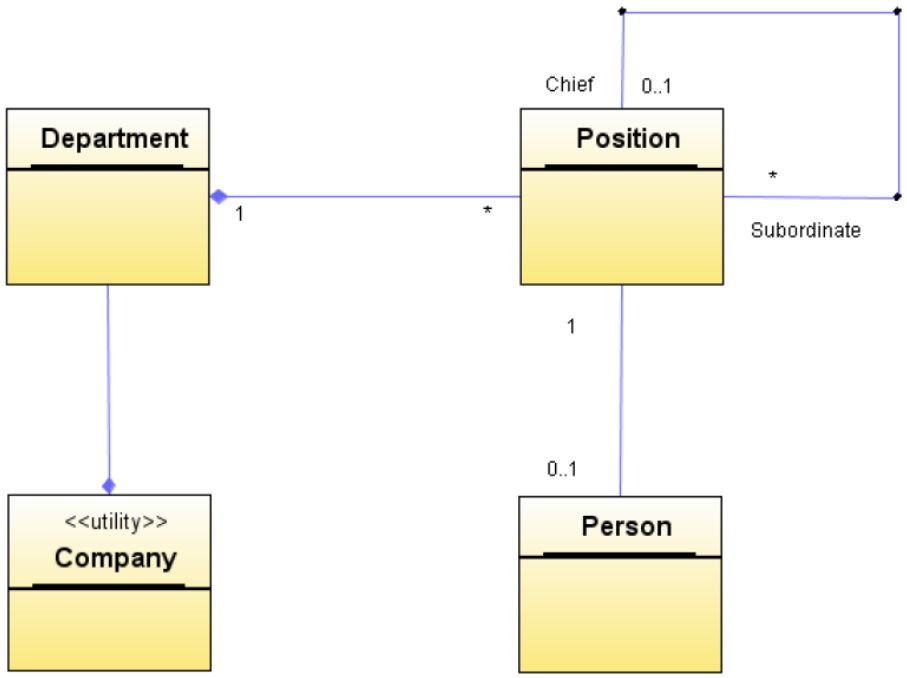
- *Роль полюса ассоциации, называемая также спецификатором интерфейса — это* способ указать, как именно участвует классификатор (присоединенный к данному полюсу ассоциации) в ассоциации. В общем случае данное дополнение имеет следующий синтаксис:
- видимость ИМЯ : тип
- Имя является обязательным, оно называется *именем роли и фактически является* собственным именем полюса ассоциации, позволяющим различать полюса.

Ассоциация на диаграмме классов

- Если рассматривается одна ассоциация, соединяющая два различных класса, то в именах ролей нет нужды: полюса ассоциации легко можно различить по именам классов, к которым они присоединены. Однако, если это не так, т. е. если два класса соединены несколькими ассоциациями, или же если ассоциация соединяет класс с самим собой, то указание роли полюса ассоциации является необходимым.

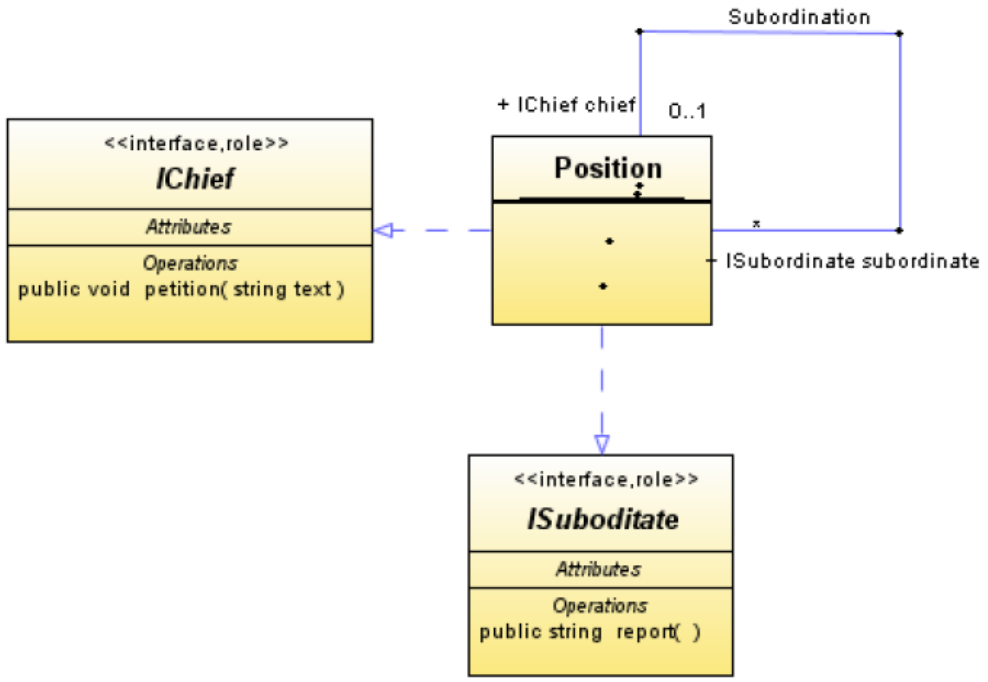
Ассоциация на диаграмме классов

- На этом рисунке изображена ассоциация класса Position с самим собой. Эта ассоциация призвана отразить наличие иерархии подчиненности должностей в организации.
- Видно только, что объекты класса Person образуют некоторую иерархию, но не более того. Используя спецификацию интерфейсов и, заодно, отношения реализации и интерфейсы можно описать субординацию в информационной системе отдела кадров достаточно лаконично и точно.



Ассоциация на диаграмме классов

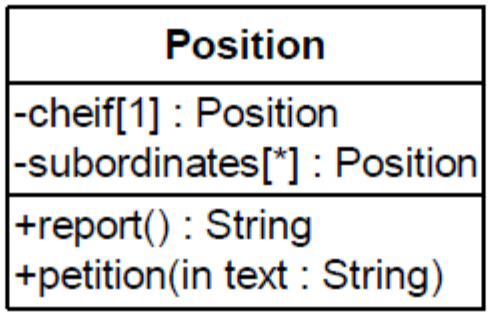
- На рисунке указано, что в иерархии субординации каждая должность может играть две роли. С одной стороны, должность может рассматриваться как начальственная (chief), и в этом случае она предоставляет интерфейс IChief, имеющий операцию petition (начальнику можно подать служебную записку). С другой стороны, должность может рассматриваться как подчиненная (subordinate), и в этом случае она предоставляет интерфейс Isubordinate, имеющий операцию report (от подчиненного можно потребовать отчет).
- У начальника может быть произвольное количество подчиненных, в том числе и 0, у подчиненного может быть не более одного начальника.



Ассоциация на диаграмме классов

- Имея в виду ту же самую цель, можно поступить и по-другому: непосредственно включить в описание класса составляющие, ответственные за обеспечение нужной функциональности. Однако такое решение "не в духе" UML: во-первых, оно слишком привязано к реализации, разработчику не оставлено никакой свободы для творческих поисков эффективного решения; во-вторых оно менее наглядно — неудачный выбор имен атрибутов способен замаскировать семантику отношения для читателя модели, потеряны информативные имена ролей и ассоциации; в-третьих, оно менее надежно (в предыдущей диаграмме модели подчиненный синтаксически не может потребовать отчета от начальника, а в этой модели — может, и нужно предусматривать дополнительные средства для обработки этой ошибки.

Операции, реализующие интерфейсы



Атрибуты, реализующие ассоциацию

Ассоциация на диаграмме классов

- *Направление навигации полюса ассоциации — это свойство полюса, имеющее* значение типа Boolean, и определяющее, можно ли получить с помощью данной ассоциации доступ к объектам класса, присоединенному к данному полюсу ассоциации.
- По умолчанию это свойство имеет значение true, т. е. доступ возможен. Если все полюса ассоциации (обычно их два) обеспечивают доступ, то это никак не отражается на диаграмме (потому, что данный случай наиболее распространенный и предполагается по умолчанию). Если же навигация через некоторые полюса возможна, а через другие нет, то те полюса, через которые навигация возможна, отмечаются стрелками на концах линии ассоциации.

Ассоциация на диаграмме классов

- Таким образом, если никаких стрелок не изображается, то это означает, что подразумеваются стрелки во всех возможных направлениях. Если же некоторые стрелки присутствуют, то это означает, что доступ возможен только в направлениях, указанных стрелками.
- Отсюда следует, что в UML невозможно изобразить случай, когда навигация вдоль ассоциации невозможна ни в каком направлении — но это и не нужно, т. к. такая ассоциация не имеет смысла.

Ассоциация на диаграмме классов

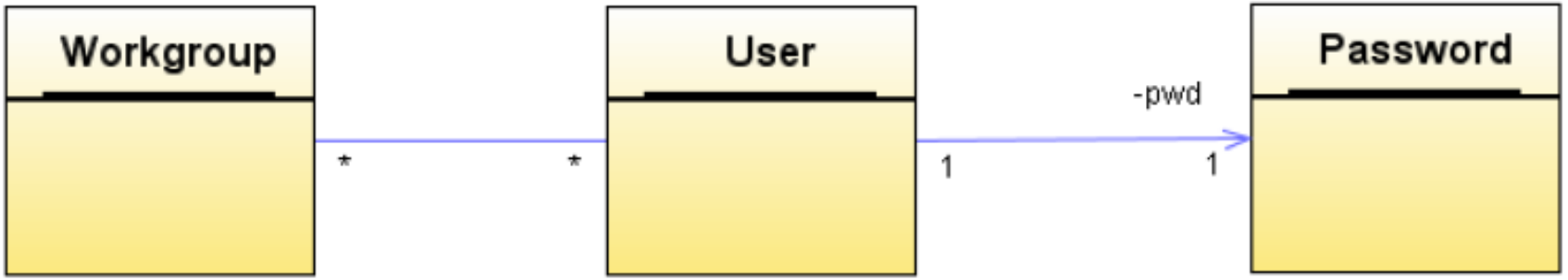
- Допустим, имеются два класса: User (содержит информацию о пользователе) и Password (содержит пароль — информацию, необходимую для аутентификации пользователя). Мы хотим отразить в модели следующую ситуацию: имеется взаимно однозначное соответствие между пользователями и паролями, зная пользователя можно получить доступ к его паролю, но обратное неверно: по имеющемуся паролю нельзя определить, кому он принадлежит.

Ассоциация на диаграмме классов

- *Видимость полюса ассоциации — это указание того, является ли классификатор* присоединенный к данному полюсу ассоциации, видимым для других классификаторов вдоль данной ассоциации, помимо тех классификаторов, которые присоединены к другим полюсам ассоциации.

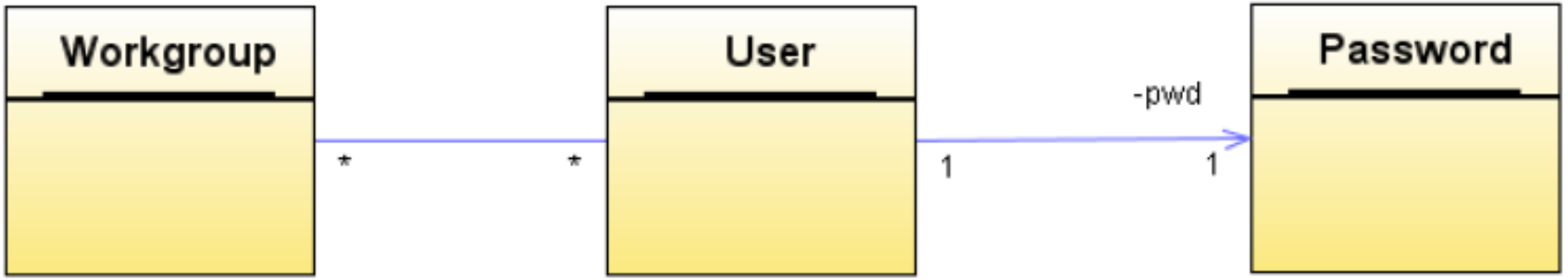
Ассоциация на диаграмме классов

- Допустим, у нас есть третий класс — Workgroup — ответственный за хранение информации о группах пользователей. Тогда очевидно, что зная группу, мы должны иметь возможность узнать, какие пользователи входят в группу, и обратно, для каждого пользователя можно определить в какую группу (или в какие группы) он включен. Но не менее очевидно, что доступ к группе не должен позволять узнать пароли пользователей этой группы. Другими словами, мы хотим ограничить доступ к объектам через полюс ассоциации. В UML для этого нужно явно указать имя роли полюса ассоциации, и перед именем роли поставить соответствующий символ видимости «-».



Ассоциация на диаграмме классов

- Обратите внимание, что, согласно семантическим правилам UML, ограничение видимости полюса pwd распространяется на объекты класса Workgroup, но не на объекты класса User — непосредственно связанные объекты всегда видят друг друга. Объект класса User может использовать интерфейс pwd, предоставляемый классом Password — стрелка навигации разрешает ему это, но объект класса Workgroup не может использовать интерфейс pwd — значение видимости private (знак –) запрещает ему это.



Обобщение на диаграмме классов

- Отношение обобщения часто применяется на диаграмме классов.
- Действительно, трудно представить себе ситуацию, когда между объектами в одной системе нет ничего общего. Как правило, общее есть и это общее целесообразно выделить в отдельный класс.
- При этом общие составляющие, собранные в суперклассе, автоматически наследуются подклассами. Таким образом, сокращается общее количество описаний, а значит, уменьшается вероятность допустить ошибку.

Обобщение на диаграмме классов

- Использование обобщений не ограничивает свободу проектировщика системы, поскольку унаследованные составляющие можно переопределить в подклассе, если нужно. При обобщении выполняется принцип подстановочности. Фактически это означает увеличение гибкости и универсальности программного кода при одновременном сохранении надежности, обеспечиваемой контролем типов.
- Действительно, если, например, в качестве типа параметра некоторой процедуры указать суперкласс, то процедура будет с равным успехом работать в случае, когда в качестве аргумента ей передан объект любого подкласса данного суперкласса.
- Суперкласс может быть конкретным, идентифицированным одним из методов, а может быть абстрактным, введенным именно для построения отношений обобщения.

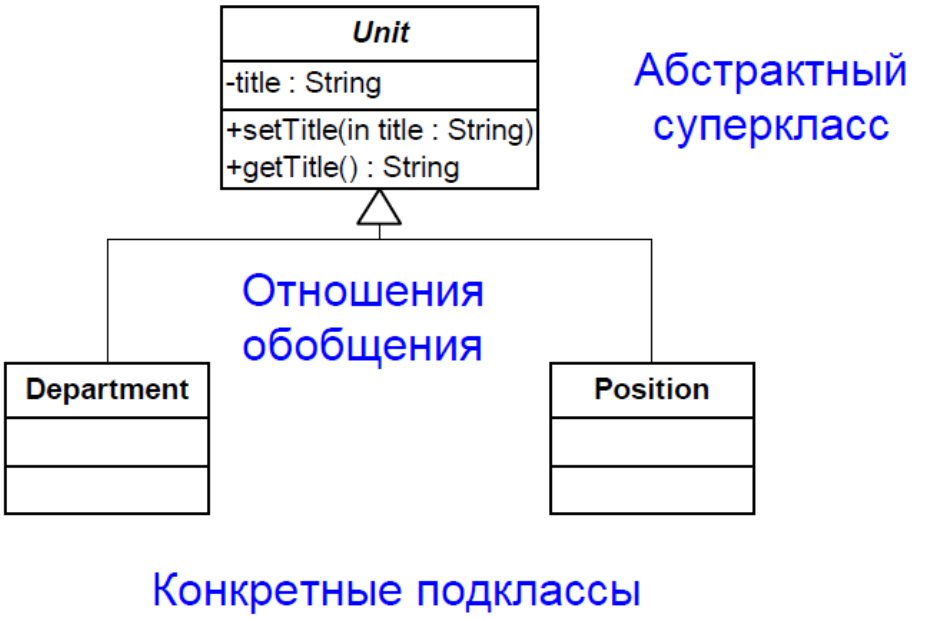
Обобщение на диаграмме классов

- **Рассмотрим пример.** В информационной системе отдела кадров мы выделили классы Position, Department и Person.
- Резонно предположить, что все эти классы имеют атрибут, содержащий собственное имя объекта, выделяющее его в ряду однородных. Для простоты положим, что такой атрибут имеет тип String. В таком случае можно определить суперкласс, ответственный за хранение данного атрибута и работу с ним, а прочие классы связать с суперклассом отношением обобщения.
- Однако более пристальный анализ предметной области наводит на мысль, что работа с собственным именем для выделенных классов производится не совсем одинаково.

Обобщение на диаграмме классов

- Назначение и изменение собственных имен подразделениям и должностям находится в пределах ответственности информационной системы отдела кадров, но назначение собственного имени сотрудника явно выходит за эти пределы.
- Исходя из этих соображений, мы приходим к следующей структуре обобщений.

Обобщение на диаграмме классов



Обратите внимание, что суперкласс *Unit мы определили как абстрактный,* т. е. не могущий иметь непосредственных экземпляров, поскольку не предполагаем иметь в системе объекты данного класса. Класс *Unit в данном нужен только для* того, чтобы свести описания одного атрибута и двух операций в одно место и не повторять их дважды.

Обобщение на диаграмме классов

- В UML допускается, чтобы класс был подклассом нескольких суперклассов (множественное наследование), не требуется, чтобы у базовых классов был общий суперкласс (несколько иерархий обобщения) и вообще не накладывается никаких ограничений, кроме частичной упорядоченности (т. е. отсутствия циклов в цепочках обобщений). Нарушение данного условия является синтаксической ошибкой. При множественном обобщении возможны конфликты: суперклассы содержат составляющие, которые невозможно включить в один подкласс, например, атрибуты с одинаковыми именами, но разными типами.
- В UML конфликты при множественном обобщении считаются нарушением правил непротиворечивости модели

Выводы

- Диаграммы классов моделируют структуру объектов и связей между ними.
- Основные сущности на диаграмме классов – это классы, которые связываются между собой отношениями разных типов.
- Классы идентифицируются на основе анализа предметной области, взаимного согласования элементов модели и общих теоретических соображений.
- Существует несколько приёмов для идентификации классов, но универсального нет.
- Ассоциация между классами – это наиболее общий вид отношений. На диаграммах классов имеет множество дополнений.