

ЛЕКЦИЯ 6. НОТАЦИЯ UML

НА ЭТОЙ ЛЕКЦИИ МЫ:

1. Изучим нотацию UML;
2. Узнаем, какие основные виды нотации UML существуют и где их применять;
3. Разберем ключевые элементы нотации UML.

ОСНОВНЫЕ ТЕРМИНЫ:

Нотация – Совокупность графических элементов, которые используются для описания бизнес-процессов компании.

ПЛАН ЛЕКЦИИ

Введение

UML

Типы диаграмм UML

Диаграмма прецедентов

Диаграмма классов

Диаграмма активностей

Плюсы и Минусы UML

Домашнее задание

ВВЕДЕНИЕ

Здравствуйте! Добро пожаловать на курс по «Бизнес-процессам»! Меня зовут Алина Загидуллина, я автор этого курса в компании GeekBrains. Я

более 4 лет работала в операционном консалтинге в большой четверке (Deloitte, KPMG) с фокусом на проекты по оптимизации бизнес-процессов и разработке программ диджитализации. Делала проекты для различных индустрий, среди которых - ритейл, нефтяная промышленность, телеком, банки и транспорт. Также работала в Mail.ru Group, в отделе аналитики и эффективности, где разрабатывала сценарии развития для таких продуктов как ВКонтакте, GeekBrains, Юла, Delivery Club, Одноклассники и многих других.

Сегодня вас ждет урок, на котором мы разберем основные нотации для моделирования бизнес-процессов:

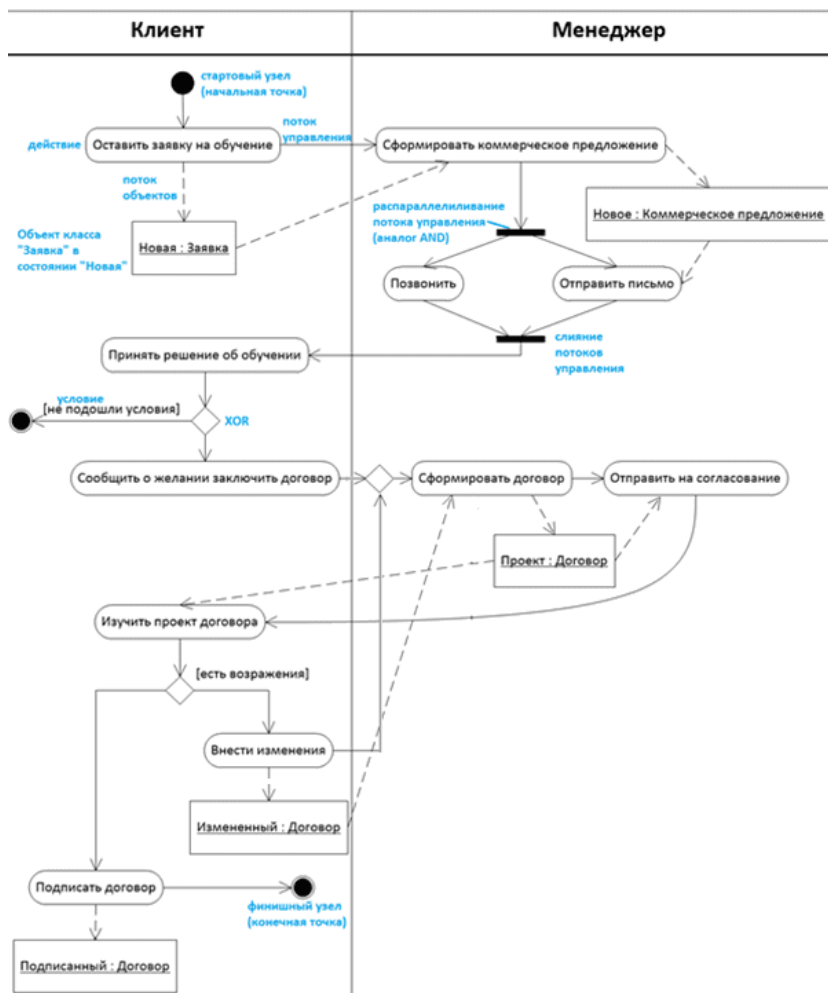
- Познакомимся с нотацией UML;
- Узнаем, как применять данную нотацию, какие у нее ключевые элементы и важные правила использования.

UML

Unified Modeling Language (UML) - унифицированный язык моделирования.

Расшифруем: modeling подразумевает создание модели, описывающей объект. Unified (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов, а в нашем случае и для бизнес-процессов. Нотация использует объектно-ориентированные методы (методология, основанная на представлении программы/процесса в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования).

Пример: Процесс «Подписание договора на обучение»



ТИПЫ ДИАГРАММ UML

В языке UML есть 12 типов диаграмм:

- Диаграмма прецедентов (**Use-case diagram**) - В основе — Actor (исполнитель), который устанавливает логические связи между ролями и прецедентами (вариант использования);
- Диаграмма классов (**Class diagram**) - представляет собой набор статических и декларативных элементов модели, имеющие общие атрибуты и операции. Диаграмма имеет наиболее полное и развернутое описание связей в программном коде, функциональности и информации об отдельных классах;

- Диаграмма активностей (**Activity diagram**) отображает динамические аспекты поведения и общее представление о работе системы в формате блок-схемы. Диаграмма необходима для описания бизнес-процессов, взаимодействия нескольких систем, логики процедур и потоков работ, особенно при переходе от одной деятельности к другой;
- Диаграмма последовательности (**Sequence diagram**) описывает поведенческие аспекты системы, вид сообщений и уточняет прецедентов. Необходима для отображения взаимодействия объектов в динамике и во времени, подразумевает обмен сообщениями в рамках конкретного сценария;
- Диаграмма развёртывания (**Deployment diagram**) отображает графическое представление инфраструктуры, а именно распределение компонентов системы по узлам и маршруты их соединений. Диаграмма организует компоненты и решает второстепенные задачи, связанные с определенным аспектом бизнес-процесса;
- Диаграмма сотрудничества (**Collaboration diagram**) - диаграмма взаимодействия, которая подчеркивает организационную структуру между объектами, которые отправляют и получают сообщения;
- Диаграмма объектов (**Object diagram**) предназначена для демонстрации совокупности моделируемых объектов и связей между ними в фиксированный момент времени;
- Диаграмма состояний (**Statechart diagram**) позволяет описывать поведение системы (демонстрирует поведение одного объекта в течение его жизненного цикла).

Мы сконцентрируемся на первых трех, так как именно они наиболее пригодны для описания бизнес-процессов, а остальные больше подходят для описания работы систем и интеграций.

ДИАГРАММА ПРЕЦЕДЕНТОВ

Диаграмма прецедентов — Use-case diagram - это графическое представление всех или части акторов, прецедентов и их взаимодействий в системе или бизнес-процессе.

Назначение

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему или бизнес-процесс.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- чётко отделить систему от её окружения;
- определить действующих лиц (акторов), их взаимодействие с системой и ожидаемую функциональность системы;
- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ).

Элементы

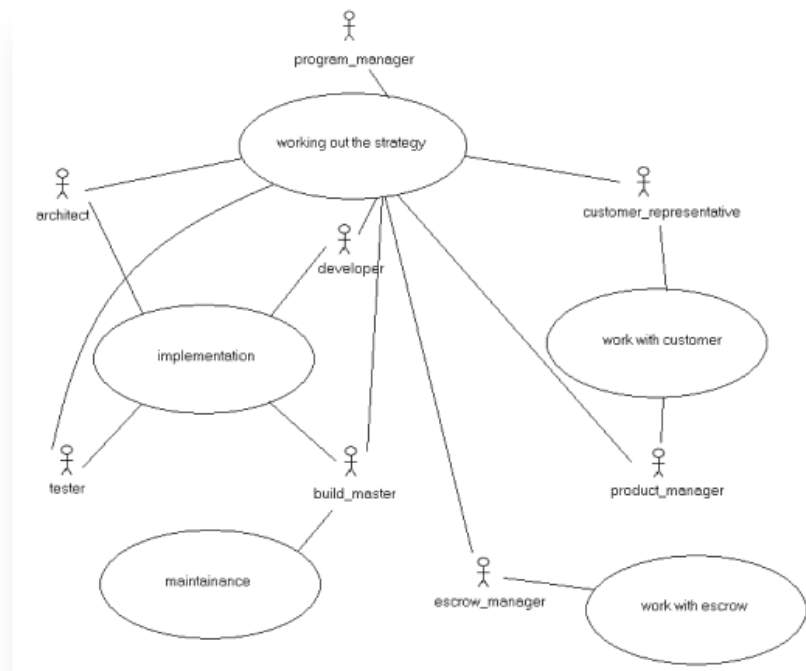
Диаграмма прецедентов использует 3 основных элемента:

- 1) Actor (участник) — множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек, роль человека в системе или другая система, подсистема или класс, которые представляют нечто вне сущности.
- 2) Use case (прецедент) — описание отдельного аспекта поведения системы с точки зрения пользователя. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.
- 3) System boundary (рамки системы) — прямоугольник с названием в верхней части и эллипсами (прецедентами) внутри. Часто может быть опущен без потери полезной информации,



Рассмотрим классический студенческий пример, в котором есть 2 участника: студент и библиотекарь. Прецеденты для студента: ищет в

каталоге, заказывает, работает в читальном зале. Роль библиотекаря: выдача заказа, консультации (рекомендации книг по теме, обучение использованию поисковой системы и заполнению бланков заказа).



Второй пример немного сложнее. Видим, что одно и то же лицо может выступать в нескольких ролях. Например, product manager у нас работает над стратегией и больше ничем не занимается, архитектор работает над стратегией и занимается внедрением, build master занимается тремя вещами одновременно, и так далее. По такой схеме мы можем проследить, какая из ролей связана с какими прецедентами.

Правила

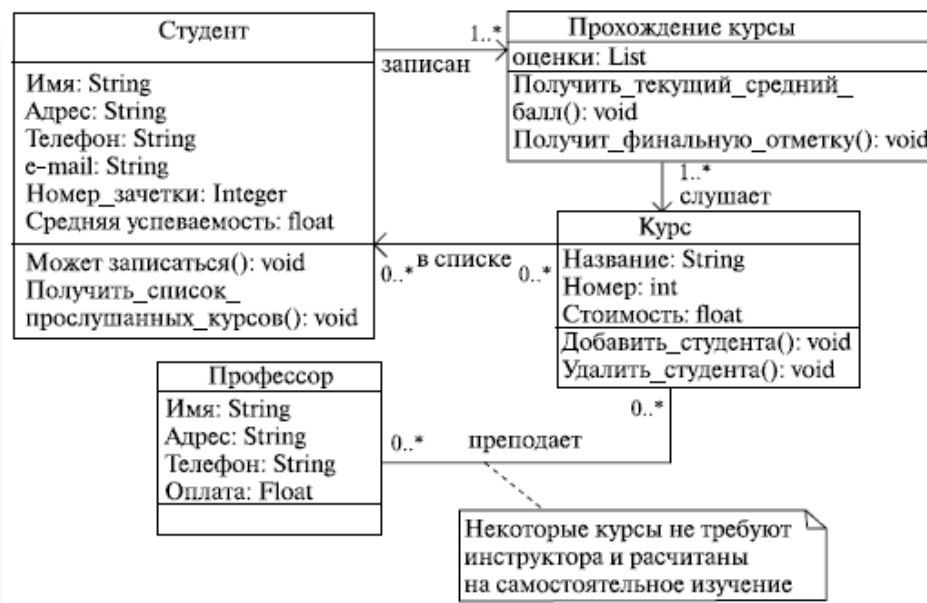
При работе с вариантами использования важно помнить несколько простых правил:

- каждый прецедент относится как минимум к одному действующему лицу;
- каждый прецедент имеет инициатора;
- каждый прецедент приводит к соответствующему результату.

ДИАГРАММА КЛАССОВ

Диаграмма классов — Class diagram — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Класс (class) — категория вещей, которые имеют общие атрибуты и операции. Сама диаграмма классов являет собой набор статических, декларативных элементов модели. Она дает нам наиболее полное и развернутое представление о связях в программном коде, функциональности и информации об отдельных классах. Приложения генерируются зачастую именно с диаграммы классов. Рассмотрим на простом примере ниже:



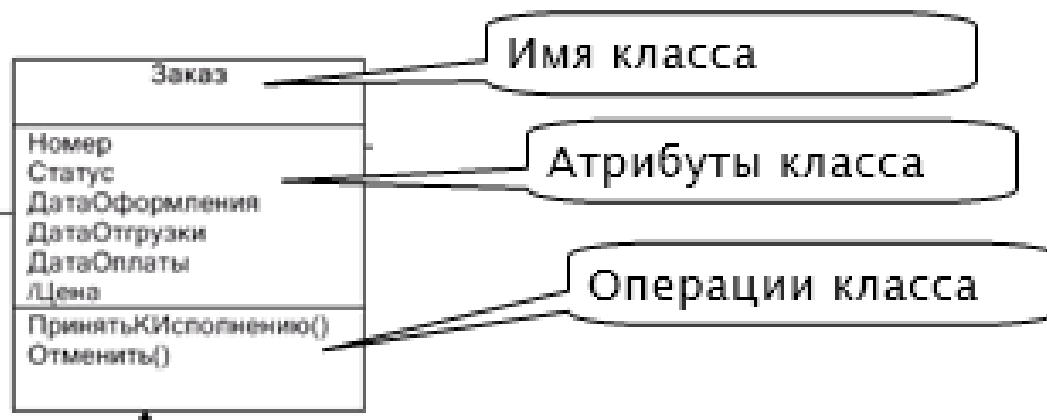
Для класса "студент" есть таблица, содержащая атрибуты: имя, адрес, телефон, e-mail, номер зачетки, средняя успеваемость. И также показаны связи данной сущности с другими: прохождением курса, какой курс слушает, кто профессор. В этом примере также добавляются функции, которые могут быть применены к сущности "студент".

Элементы

Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента:

1. В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный — то его имя пишется полужирным курсивом.

2. Посередине располагаются поля (атрибуты) класса, они выровнены по левому краю. Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными значениями атрибутов и взаимосвязей. Т.е. индивидуальность объектов определяется самим фактом их существования, а не различиями в их свойствах. Имя атрибута должно быть уникально в пределах класса. За именем атрибута может следовать его тип и значение по умолчанию.
3. Нижняя часть содержит методы (или операции) класса. Они также выровнены по левому краю. Операция – функция (действие) или преобразование. Операция может иметь параметры и возвращать значения.

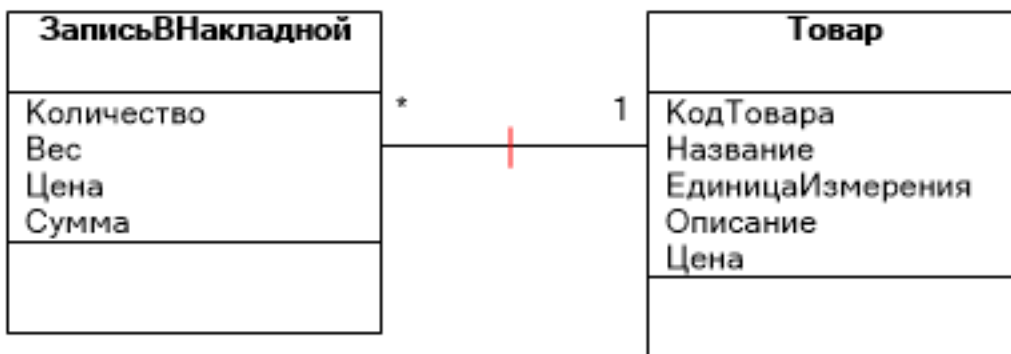


Виды связей

1. Ассоциация (association) – представляет собой отношения между экземплярами классов.

Каждый конец ассоциации обладает кратностью (синоним – мощностью, ориг. – multiplicity), которая показывает, сколько объектов, расположенных с соответствующего конца ассоциации, может участвовать в данном отношении. В примере на рисунке каждый Товар имеет сколь угодно Записей в накладной, но каждая Запись в накладной обязательно один Товар. В общем случае кратность может быть задана любым множеством.

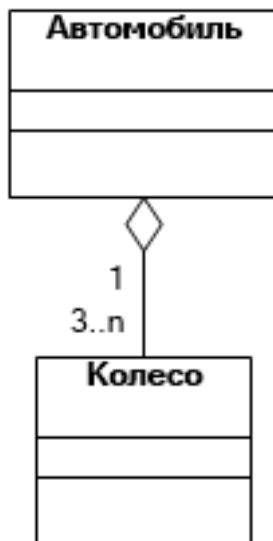
Ассоциации может быть присвоено имя. В качестве имени обычно выбирается глагол или глагольное словосочетание, сообщаящие смысл и назначение связи. Также на концах ассоциации под кратностью может указываться имя роли, т.е. какую роль выполняют объекты, находящиеся с данного конца ассоциации.



2. Агрегация (aggregation) – это ассоциация типа «целое-часть».

Агрегация в UML представляется в виде прямой с ромбом на конце.

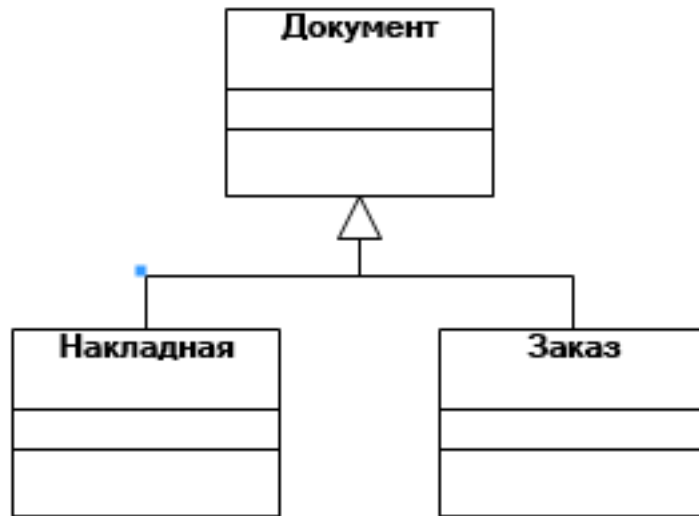
Ромб на связи указывает, какой класс является агрегирующим (т.е. «состоящим из»); класс с противоположного конца – агрегированным (т.е. те самые «части»).



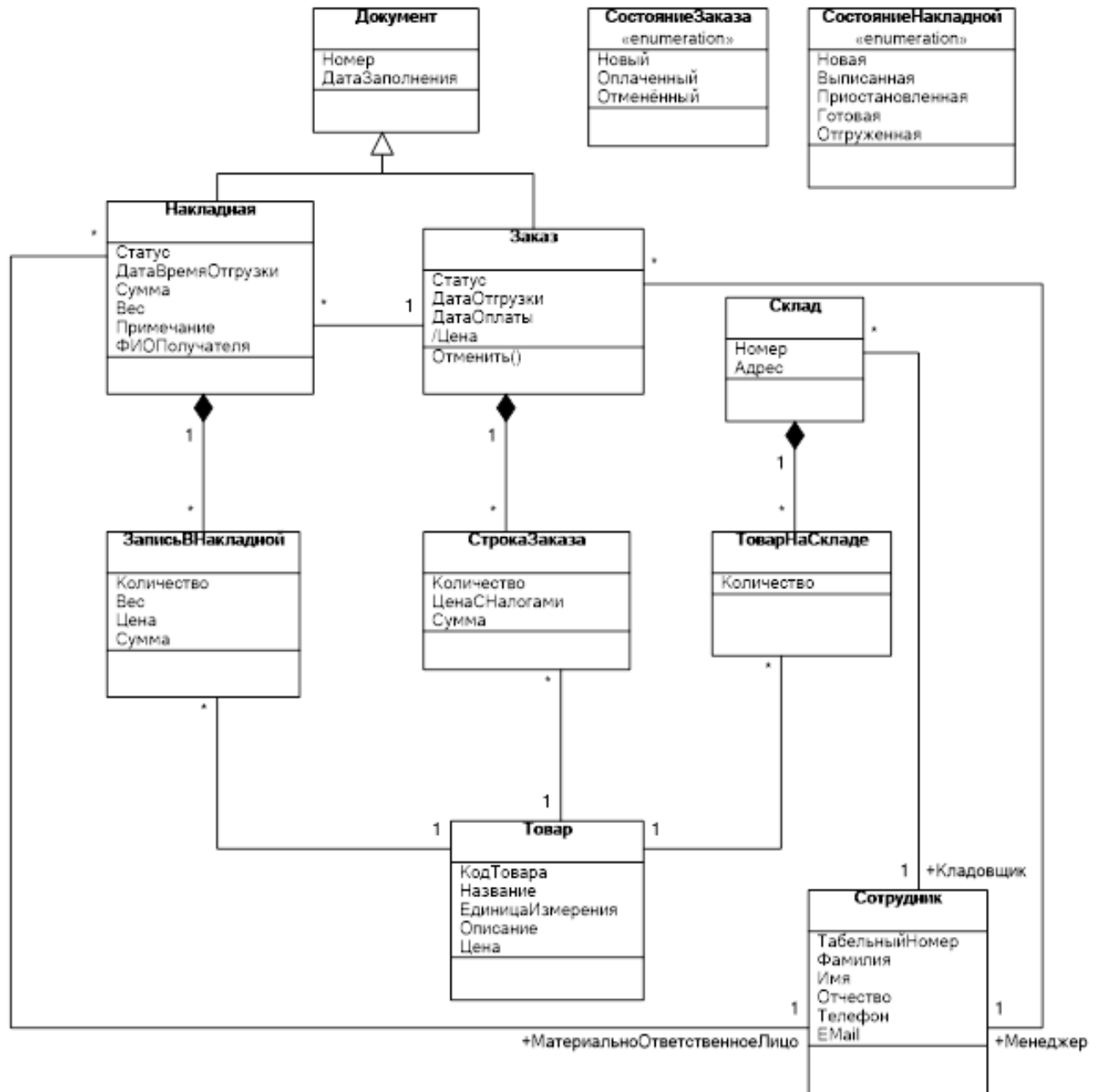
3. Композиция (composition) – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен. Важно понимать разницу между агрегацией и композицией: при агрегации объекты-части могут существовать сами по себе, а при композиции — нет. Пример агрегации: автомобиль—колесо, пример композиции: дом—комната.



4. Наследование (inheritance) – это отношение типа «общее-частное».
- Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов. При создании производного класса на основе базового (одного или нескольких) возникает иерархия наследования.
- Реализация принципов наследования является ключевой предпосылкой возможности повторного использования кода, поскольку это основной инструмент достижения полиморфизма.



Пример построения



Основной сущностью в системе будет являться товар. Как известно из задания на проектирование, товар хранится на складе. Но понятия товара как некоего описания и товара, лежащего непосредственно на складе, отличаются друг от друга. Товар, лежащий на складе, кроме того, что связан со складом отношением композиции (агрегация не совсем подходит, поскольку в данной системе товар является товаром, пока он не покинет склад), ещё характеризуется количеством.

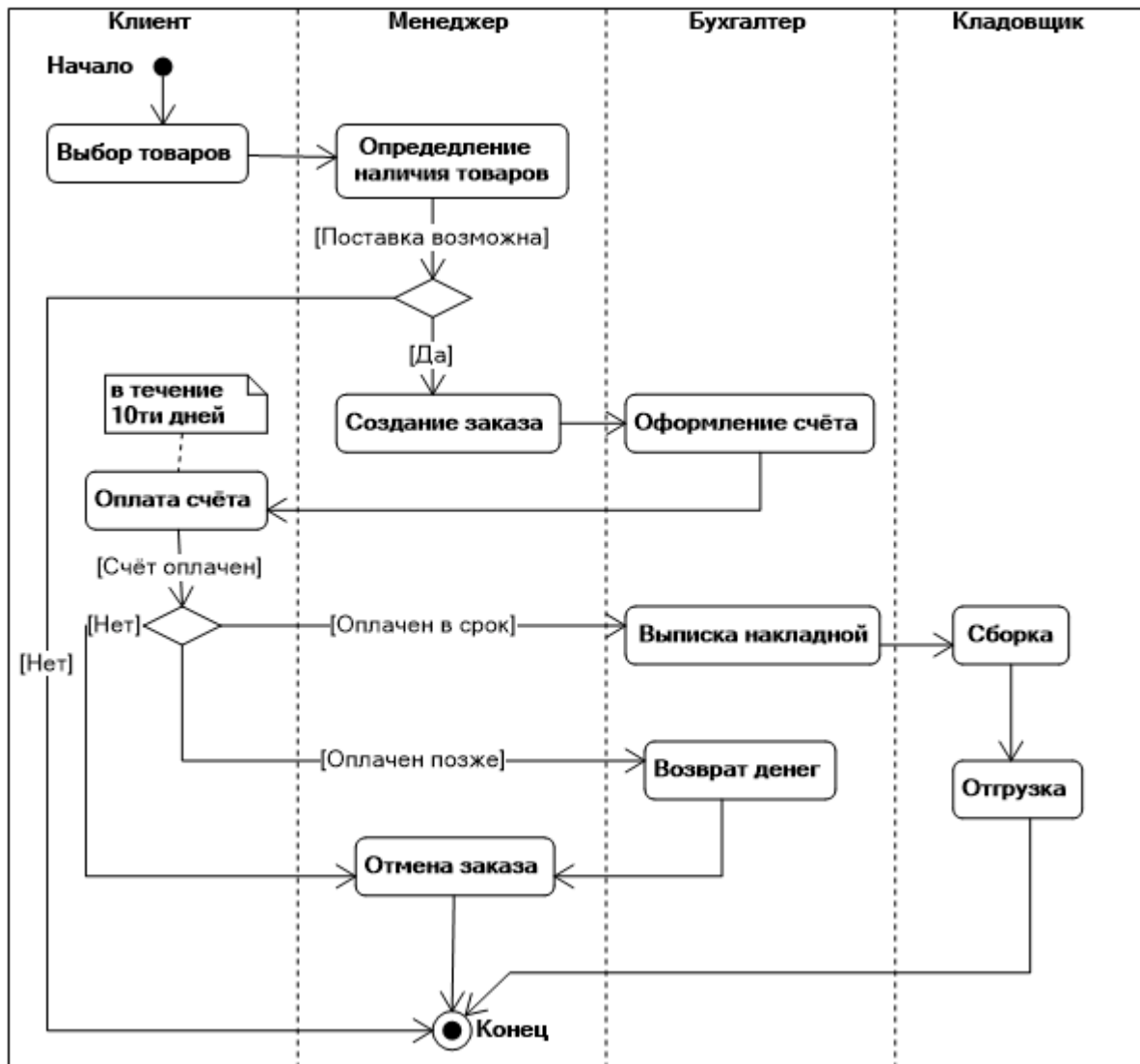
Аналогично следует рассуждать и при рассмотрении отношения Товара и Заказа, Товара и Накладной. В связи с тем, что Заказ и Накладная в сущности являются документами и имеют сходные атрибуты, они были объединены с помощью общего класса-предка Документ.

Примечательно, что на диаграмме представлены два класса со стереотипом Enumeration (перечисление). Стереотип можно установить из контекстного меню для класса.








ДИАГРАММА АКТИВНОСТЕЙ







Диаграмма активностей — Activity diagram позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

Тоже крутая штука, которая очень часто используется на практике. Диаграмма активностей описывает динамические аспекты поведения системы в виде блок-схемы, которая отражает бизнес-процессы, логику процедур и потоки работ — переходы от одной деятельности к другой. По сути, мы рисуем алгоритм действий (логику поведения) системы или взаимодействия нескольких систем.

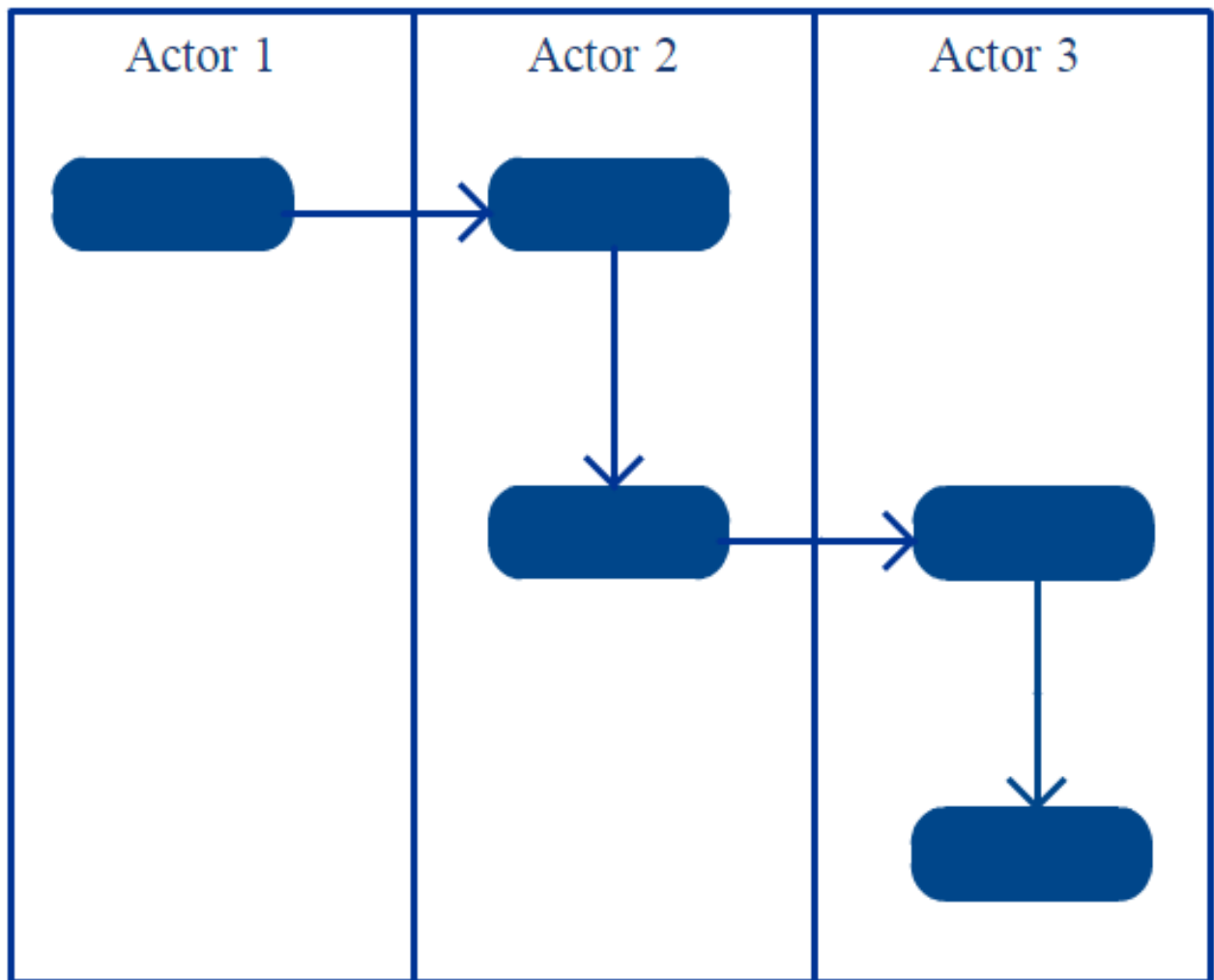


Элементы

Символ	Имя	Использовать
	Пуск/ начальный узел	Используется для представления отправной точки или начального состояния деятельности
	Действие / Состояние действия	Используется для представления деятельности процесса
	Действие	Используется для представления исполняемых подрайонов деятельности
	Поток управления / Край	Используется для представления потока управления от одного действия к другому
	Поток объекта / края управления	Используется для отображения пути движения объектов по активности
	Конечный узел активности	Используется для обозначения конца всех контрольных потоков в рамках деятельности
	Поток конечный узел	Используется для обозначения конца одного потока управления
	Узел принятия решений	Используется для представления условной точки ответвления с одним входом и несколькими выходами

	Узел слияния	Используется для представления слияния потоков. Он имеет несколько входов, но один выход.
	Вилка	Используется для представления потока, который может разветвляться на два и более параллельных потока
	Слияние	Используется для представления двух входов, которые объединяются в один выход
	Отправка сигнала	Используется для представления действия по отправке сигнала на приемную деятельность
	Получение сигнала	Используется для обозначения того, что сигнал получен
	Примечание/ комментарий	Используется для добавления соответствующих комментариев к элементам

Как и в BPMN в диаграмме активностей используется Swimlane.



В Диаграммы деятельности Swimlanes – также известные как разделы – используются для представления или группирования действий, выполняемых различными действующими лицами в одном потоке. Вот несколько советов, вы можете следовать при использовании Swimlanes.

- Добавить Swimlanes линейных процессов. Это позволяет легко читать.
- Не добавляйте более 5 Swimlanes.
- Расположить Swimlanes в логическом порядке.

Ниже – пример подобной диаграммы для интернет-магазина.

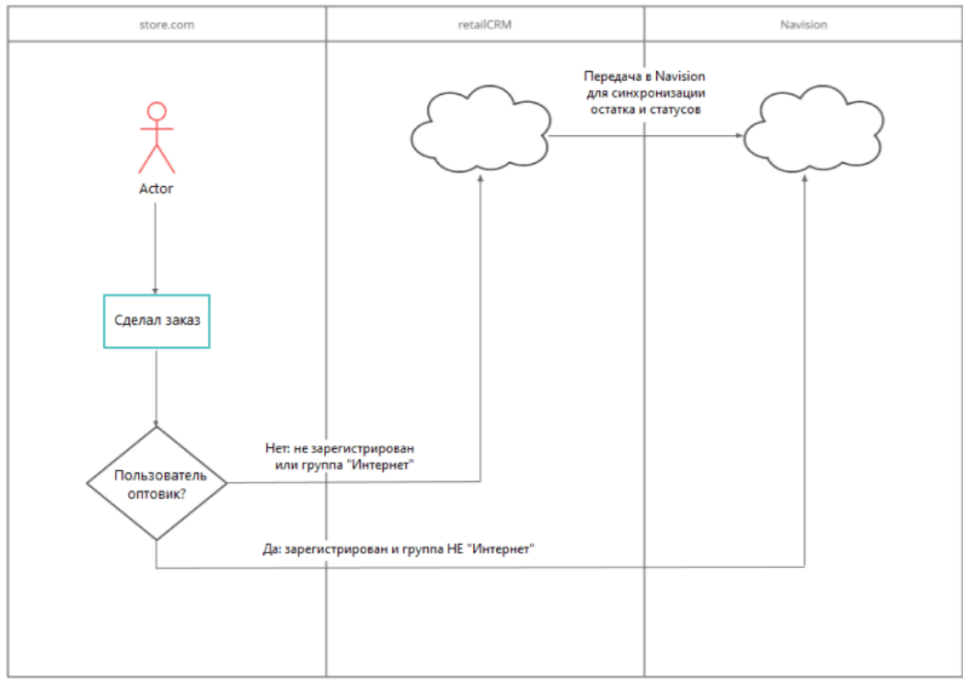


Диаграмма активностей для сайта магазина максимально доступно объясняет, какие есть интеграции в системе. Актор (в нашем случае — покупатель), зашедший на сайт, делает заказ. Далее у нас происходит разветвление: проверяем, является ли пользователь оптовиком (Да/Нет). Если он не зарегистрирован в системе и не оптовик, заказ отправляется в retailCRM. Если пользователь зарегистрирован, его заказ попадает в Navision. При этом между retailCRM и Navision происходит синхронизация остатка и статусов.

Эту базовую диаграмму мы можем дополнить, расширить, она может выступить частью документации и дает общее представление о работе системы.

ПЛЮСЫ И МИНУСЫ UML

Среди ключевых преимуществ модели можно выделить следующие характеристики:

- + UML — это объектно-ориентированный язык, поэтому методы, с помощью которых описываются результаты анализа и проектирования являются семантически идентичными к методам программирования на ключевых ОО-языках, используемых в современных технологиях.
- + С помощью UML можно описать ситуацию или ключевую задачу с различных точек зрения и аспектов поведения системы.
- + UML диаграммы простые в восприятии: прочитав схему и ознакомившись с ее синтаксисом сможет даже работник, не имеющих специальных знаний в области программирования и постройки бизнес-моделей (речь идет о стандартных схемах с 20-40 условными обозначениями).
- + UML минимизирует процент возможных ошибок при создании бизнес-процесса. Например, она исключает несогласованность параметров дополнительных программ или изменение основных атрибутов.
- + Любой этап бизнес-процесса может быть использован повторно в уже существующем или новом проекте организации.
- + UML можно применять не только для решения вопросов программной инженерии, но и для введения собственных текстовых и графических стереотипов.
- + В отличие от аналогичных нотаций, UML стремительно развивается и получает широкое распространение в построении моделей различных сферах бизнеса.

Как и другие нотации, UML имеет недостатки:

- Многие программисты используют современную версию нотации UML 2.0. Она характеризуется высокой избыточностью языка, содержит множество диаграмм и конструкций, которые не всегда важны при создании модели бизнес-процесса.

- Применение объектно-ориентированного подхода требует наличия знаний о предметной области и методах анализа на языке программирования. Крупные проекты могут включать свыше 100 условных обозначений. В таком случае в команде должны быть специалисты, владеющие определенным уровнем квалификации и умеющие отходить от традиционных подходов к работе.

ДОМАШНЕЕ ЗАДАНИЕ

Необходимо описать бизнес-процесс используя сначала нотацию UML. Описать 1 процесс на выбор:

- Отправка посылки Почтой России
- Поиск работы
- *Свой вариант*

В СЛЕДУЮЩЕЙ ЛЕКЦИИ:

1. Узнаем, как находить зоны для развития в процессе;
2. Поговорим про то, какие у процесса есть критерии эффективности.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА:

1. [Официальный сайт UML с оригинальными спецификациями и правилами на английском языке](#)

2. [Miro](#)

Платформа для моделирования бизнес-процессов

3. [Use case diagram](#)

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА И ИСТОЧНИКИ:

1. [UML](#)