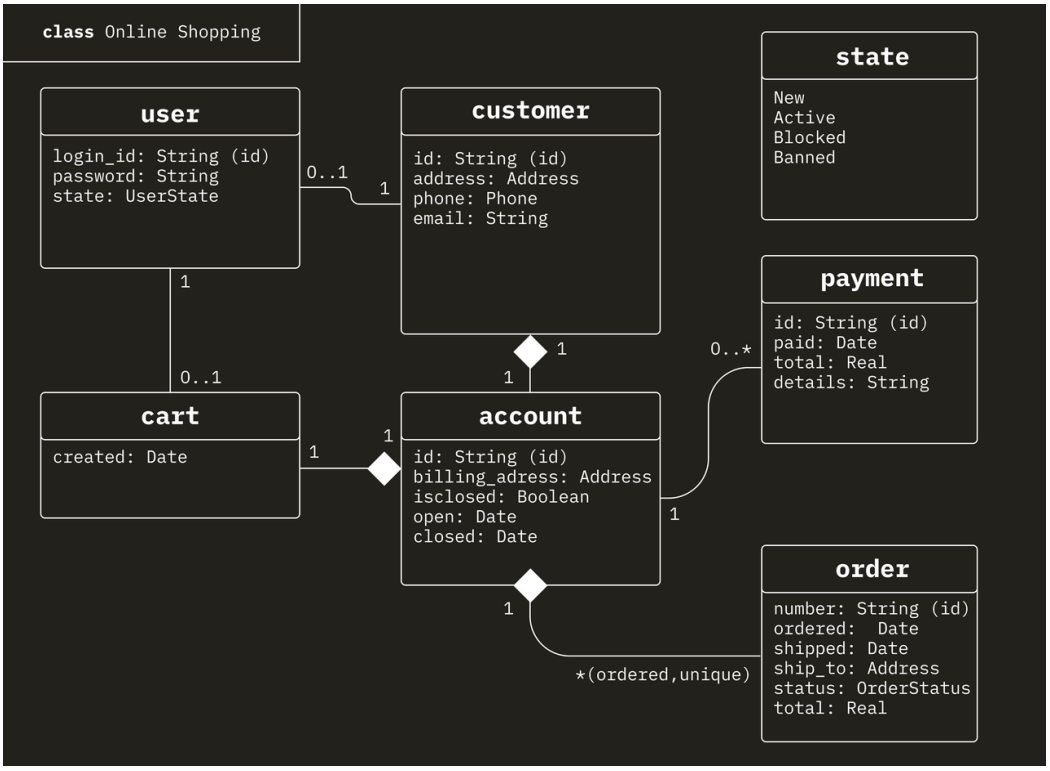
Типы диаграмм UML

Все диаграммы UML можно поделить на структурные и поведенческие. Первые описывают структуру сложных объектов и систем, вторые иллюстрируют взаимодействие с системой и процесс её работы. Внутри эти типы делятся на виды UML-диаграмм. Разберём наиболее популярные.

Структурные диаграммы

Диаграмма классов. Отображает структуру системы, содержащей различные объекты и классы. Чаще всего используется, чтобы продемонстрировать иерархию классов



внутри программы.

Диаграмма классов, описывающая существующие классы в коде интернет-магазина и связи между ними

Диаграмма компонентов. Описывает компоненты ПО и их связи между собой. Например, как микросервисы взаимодействуют друг с другом.

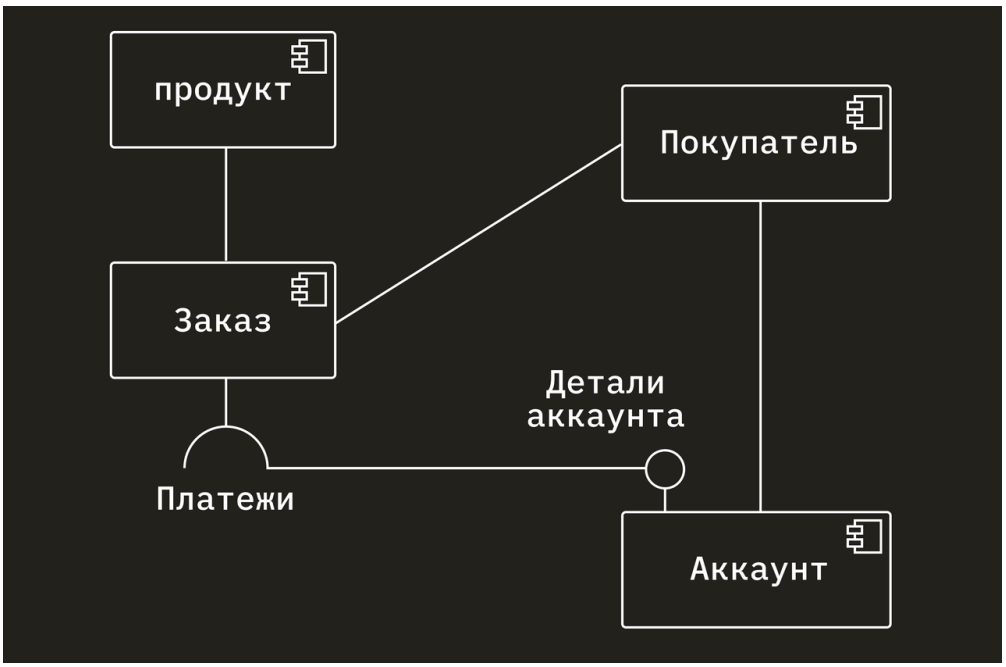
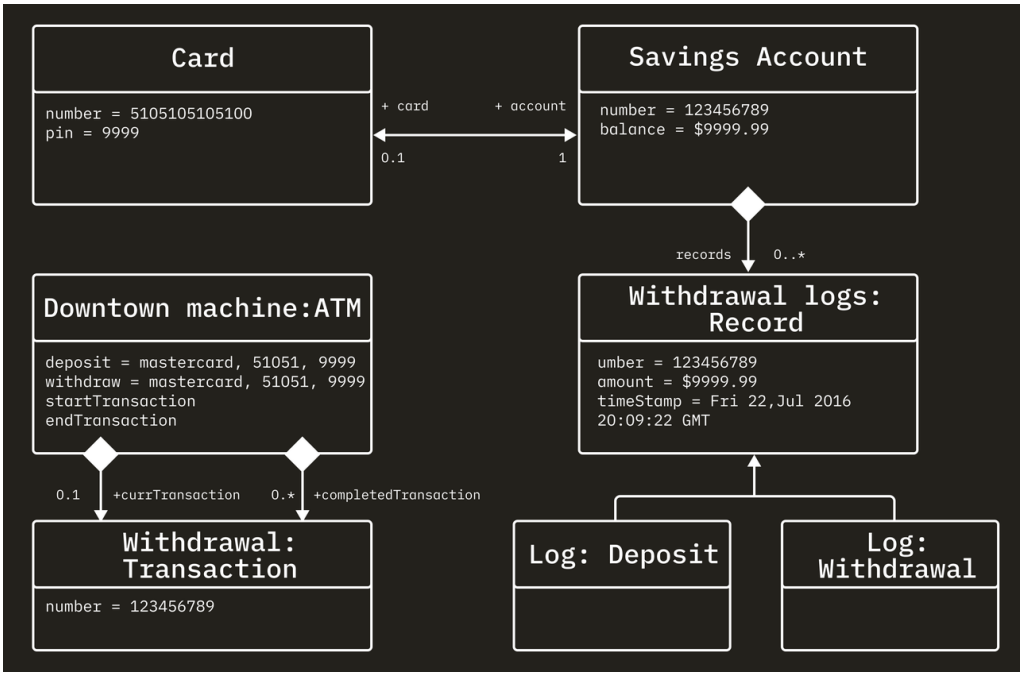


Диаграмма компонентов интернет-магазина. Внутри каждого элемента — большой объём информации, который можно представить диаграммой компонентов или классов

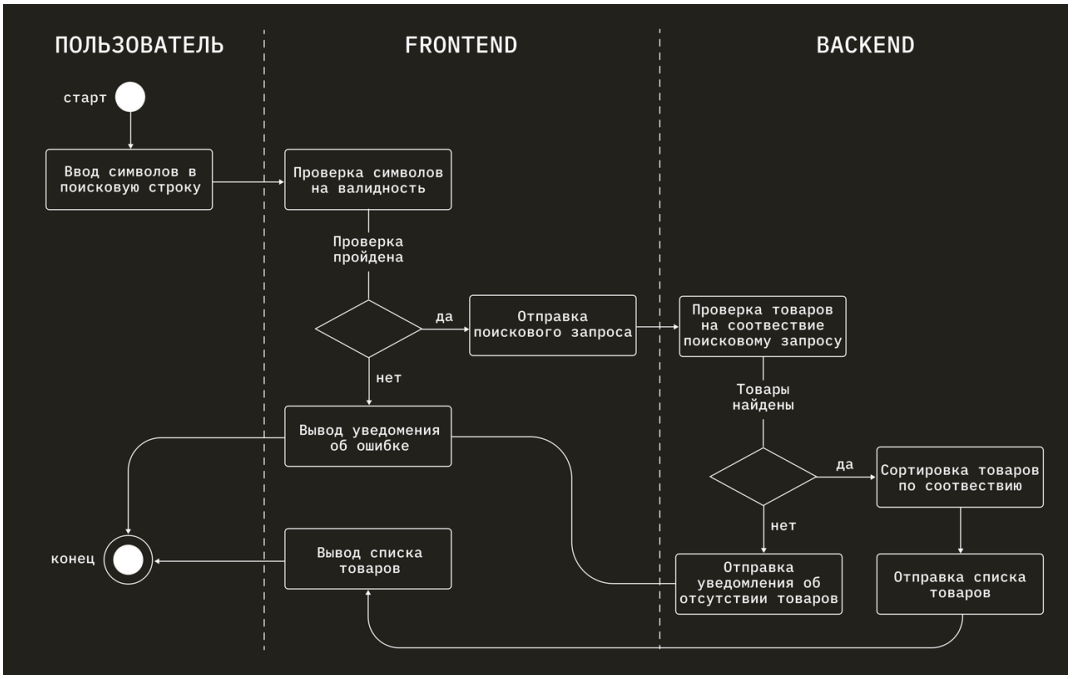
Диаграмма объектов. Показывает, как компоненты системы выглядят в определённый момент времени. Позволяет смоделировать объекты системы и связи между ними.



На диаграмме объектов видны текущие состояния элементов, в том числе их конкретные значения

Поведенческие диаграммы

Диаграмма действий, или диаграмма активностей, активити-диаграмма. Показывает последовательность действий, варианты решений и их результаты.

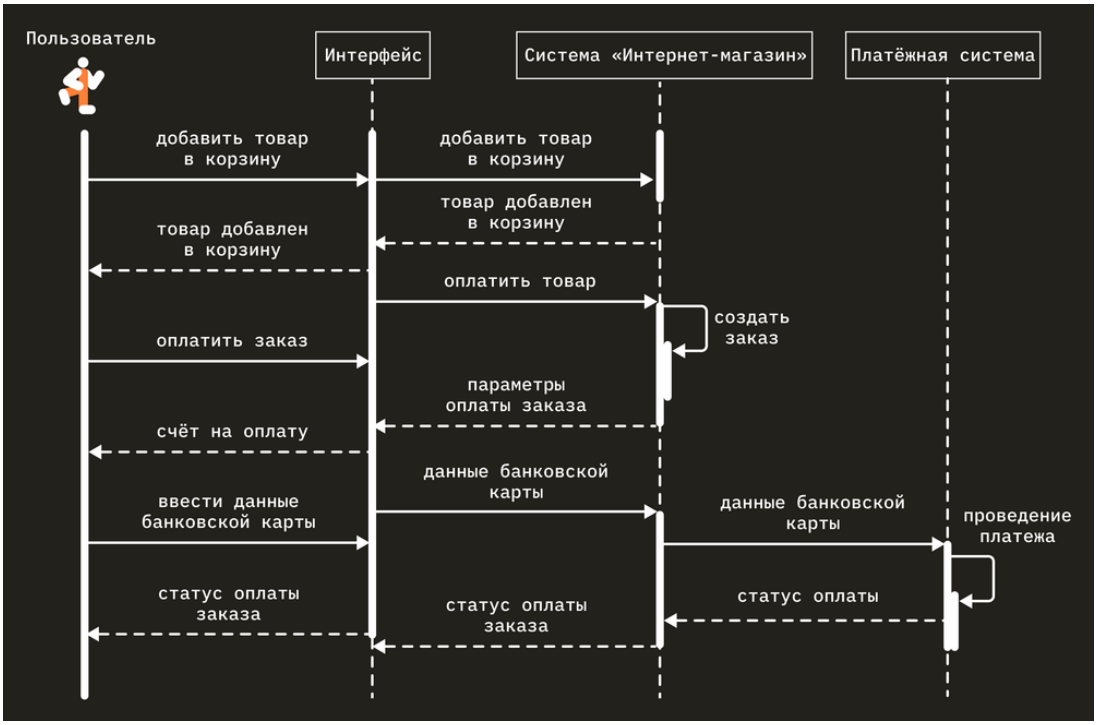


Процесс поиска товаров в интернет-магазине

Диаграмма сценариев использования. В ней обычно изображают пользователей, «агентов», которые взаимодействуют с системой. Эту диаграмму используют для определения функций ПО и связи сценариев использования, то есть юзкейсов, друг с другом. По ней определяют, какие возможности есть у разных групп пользователей и как системы участвуют в выполнении юзкейса.

Упрощённая схема онлайн-покупок, где перечислены сценарии и показано, кто именно в них участвует

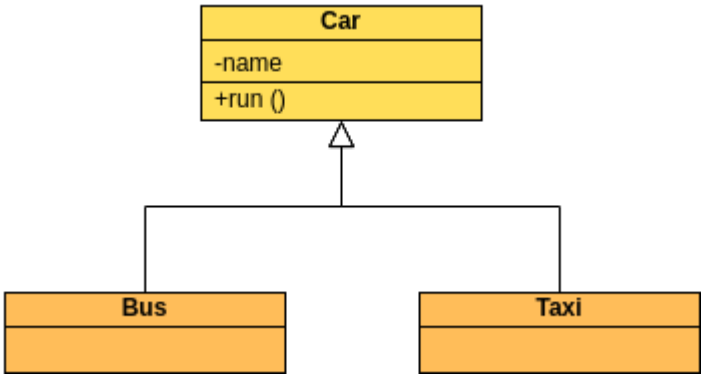
Диаграмма последовательностей. Изображает последовательные действия во времени, которые иногда называют сценариями.



Типы отношений между классами

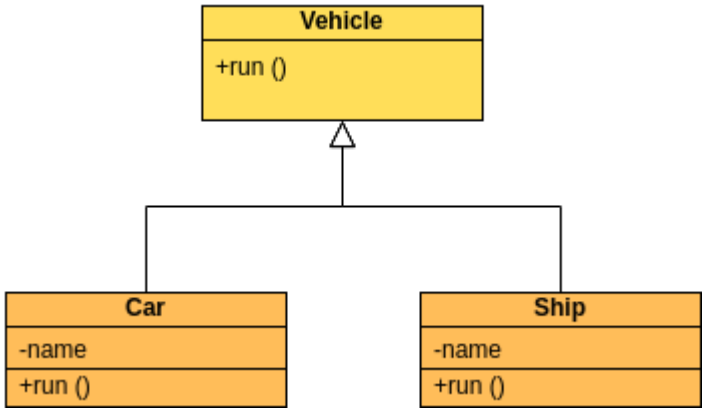
Наследование

Наследование также называется обобщением и используется для описания отношений между родительским и дочерним классами. Родительский класс также называется базовым классом, а подкласс также называется производным классом.
В отношениях наследования подкласс наследует все функции родительского класса, а родительский класс имеет все атрибуты, методы и подклассы. Подклассы содержат дополнительную информацию в дополнение к той же информации, что и родительский класс.
Например: автобусы, такси и автомобили — это автомобили, у них у всех есть имена, и все они могут находиться в дороге.



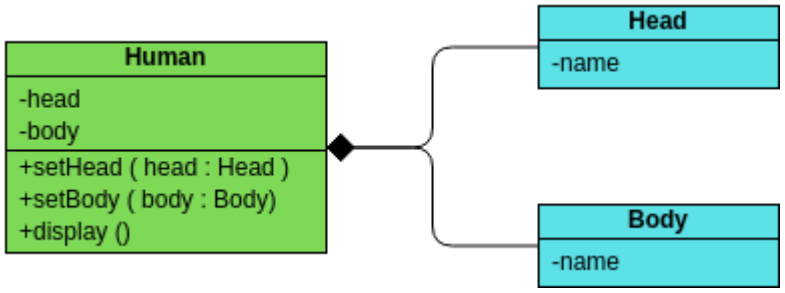
Реализация / Внедрение

Реализация (Implementation) в основном используется для указания связи между интерфейсами и классами реализации .
Интерфейс (включая абстрактный класс) — это набор методов. В отношениях реализации класс реализует интерфейс, а методы в классе реализуют все методы объявления интерфейса.
Например: автомобили и корабли — это транспортные средства, а транспортное средство — всего лишь абстрактное понятие мобильного средства, а корабль и транспортное средство реализуют конкретные мобильные функции.



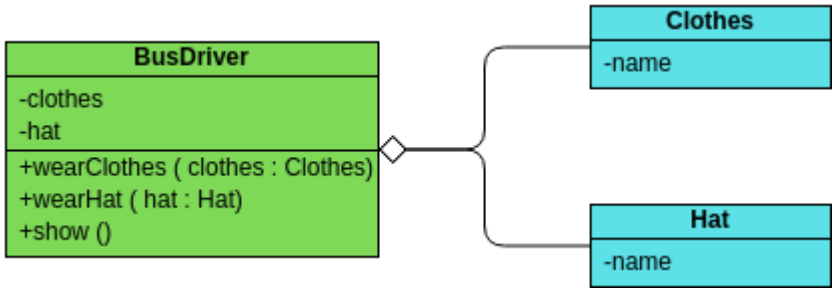
Связь композиции

Композиция: Отношения между целым и частью, но целое и часть не могут быть разделены .
Комбинированное отношение представляет собой отношение между целым и частью класса, а общее и часть имеют согласованное время жизни. Как только объект в целом перестанет существовать, некоторые объекты не будут существовать, и все они умрут в одной и той же жизни. Например, человек состоит из головы и тела. Они неразделимы и сосуществуют.



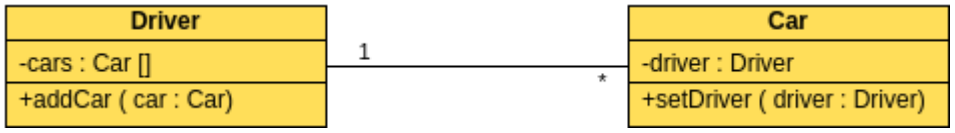
Отношения агрегации

Агрегация: отношения между целым и частью, а также целым и частью могут быть разделены.
Агрегатные отношения также представляют отношения между целым и частью класса, объекты-члены являются частью общего объекта, но объект-член может существовать независимо от общего объекта.
Например, водители автобусов, рабочая одежда и головные уборы являются частью общих отношений, но их можно разделить. Рабочую одежду и головные уборы можно надевать на других водителей. Водители автобусов также могут носить другую рабочую одежду и головные уборы.



Отношения ассоциации

Ассоциация: указывает, что свойство класса содержит ссылку на экземпляр (или экземпляры) другого класса .
Ассоциация — это наиболее часто используемая связь между классом и классом, что означает наличие связи между одним типом объекта и другим типом объекта. Комбинации и агрегации также относятся к ассоциативным отношениям , но отношения между классами принадлежности слабее двух других.
Существует четыре вида ассоциаций : двусторонние ассоциации , односторонние ассоциации , самоассоциация и многозначные ассоциации .
Например: машины и водители, одна машина соответствует конкретному водителю, и один водитель может управлять несколькими машинами.



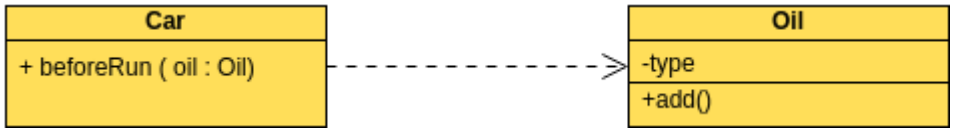
На диаграммах UML двунаправленные ассоциации могут иметь две стрелки или не иметь стрелок , а односторонние ассоциации или самоассоциации имеют стрелку .
В отношении множественности вы можете добавить число непосредственно к связанной строке, чтобы указать количество объектов в соответствующем классе.

- 1..1: Единственный
- 0..*: ноль или больше
- 1..*:один или больше
- 0..1: Нет или только один
- m..n: не менее m, не более n (m<=n)

Зависимости

Зависимость: предположим, что изменение в классе A вызывает изменение в классе B, тогда скажем, что класс B зависит от класса A.
В большинстве случаев зависимости отражаются в методах класса, использующих в качестве параметра объект другого класса .

Отношение зависимости — это отношение «использования». Изменение в конкретной вещи может повлиять на другие вещи, которые ее используют, и использовать зависимость, когда необходимо указать, что одна вещь использует другую. Например: Автомобиль работает на бензине. Если нет бензина, машина не сможет ехать.



Стереотипы в UML

В UML (Unified Modeling Language) стереотип представляет собой механизм расширения базовых элементов языка. Стереотипы позволяют добавлять дополнительные свойства и ограничения к элементам модели, что позволяет уточнить их семантику и использовать для специфических целей.

Стереотипы в UML выражаются в виде символа «< >», который добавляется к имени элемента модели. Например, стереотип «<<interface>>» может быть добавлен к классу, чтобы указать, что он является интерфейсом. Также можно создавать свои собственные стереотипы и использовать их для определения специфических свойств элементов модели.

Одним из преимуществ использования стереотипов является возможность уточнения семантики элементов модели. Например, класс может иметь стереотип «<<singleton>>», который указывает, что он является синглтоном. Это позволяет уточнить, что класс может иметь только один экземпляр, что может быть полезно при проектировании системы.

Кроме того, стереотипы позволяют использовать элементы модели для специфических целей. Например, класс может иметь стереотип «<<entity>>», который указывает, что он является сущностью в базе данных. Это позволяет использовать класс для генерации SQL-кода для создания таблицы в базе данных.

В целом, стереотипы в UML являются мощным механизмом для уточнения семантики элементов модели и использования их для специфических целей. Они позволяют создавать более точные и полезные модели, что может существенно улучшить процесс проектирования системы.