ООП. Лекция *4

1. UML

UML

- UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

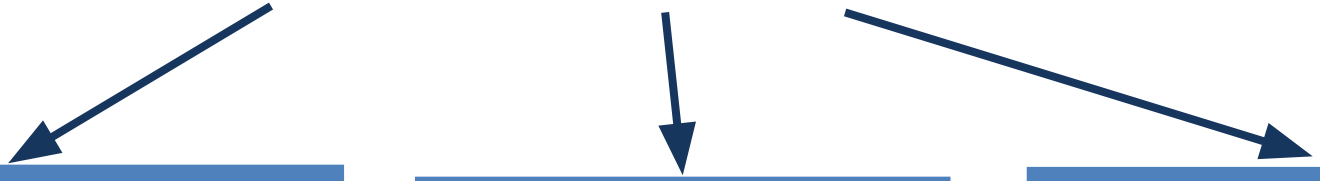
Минусы:
- трата времени;
- необходимость знания различных диаграмм и их нотаций.

Плюсы:
+ возможность посмотреть на задачу с разных точек зрения;
+другим программистам легче понять суть задачи и способ ее реализации;
+диаграммы сравнительно просты для чтения после достаточно быстрого ознакомления с их синтаксисом.

UML

1. Позволяет моделировать как ПО сложных систем, так и широкие классы самих систем и бизнес-приложений с использованием объектно-ориентированных понятий и методов;
2. Позволяет обеспечивать взаимосвязь между базовыми понятиями моделей концептуального, программируемого и физического уровней;
3. понятен системным аналитикам и программистам;
4. реализуется на различных платформах.

Диаграммы UML



Структурные диаграммы

- классов
- компонентов
- композитной/составной структуры
- объектов
- пакетов

Диаграммы поведения

- деятельности
- состояний
- вариантов использования

Диаграммы взаимодействия

- коммуникации/кооперации.
- обзора взаимодействия
- последовательности
- синхронизации

Диаграммы UML

Структурные диаграммы

- **классов** – описывает структуру системы, показывая её классы, их атрибуты и операторы, и взаимосвязи этих классов.
- **компонентов** - описывает особенности физического представления системы.
- **композитной/составной структуры** - демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.
- **кооперации** - показывает роли, которые играют участвующие во взаимодействии элементы.
- **развёртывания** - предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения.
- **объектов** - позволяет моделировать экземпляры сущностей, которые содержатся в диаграммах классов.
- **пакетов** - содержит пакеты классов и зависимости между ними

Диаграммы поведения

Диаграммы взаимодействия

Диаграммы UML

Структурные диаграммы

Диаграммы поведения

- **деятельности** - используется для моделирования процесса выполнения операций.
- **состояний** - предназначена для отображения состояний объектов системы, имеющих сложную модель поведения.
- **вариантов использования** - используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе.

Диаграммы взаимодействия

Диаграммы UML

Структурные диаграммы

Диаграммы поведения

Диаграммы взаимодействия

- **коммуникации/кооперации** - диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации.
- **обзора взаимодействия** - разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления.
- **последовательности** - диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.
- **синхронизации** - альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени. Может быть полезна в приложениях реального времени

Диаграммы UML

Структурные диаграммы

- ***Диаграмма классов*** *– описывает структуру системы, показывая её классы, их атрибуты и операторы, и взаимосвязи этих классов.*

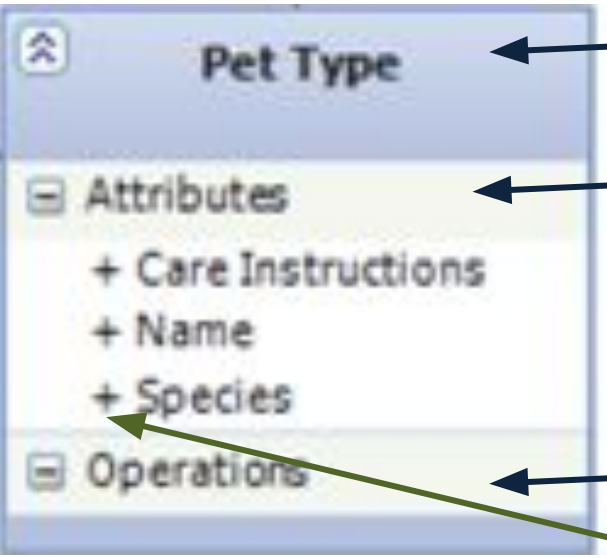
- Это один из наиболее часто используемых видов диаграмм UML.
- Обычно создание диаграммы классов – это показатель окончания процесса анализа и начала процесса проектирования.

Диаграммы поведения

Диаграммы взаимодействия

Отображение класса в UML

Класс на диаграмме изображается в виде прямоугольника, разделенного горизонтальными линиями на три части.

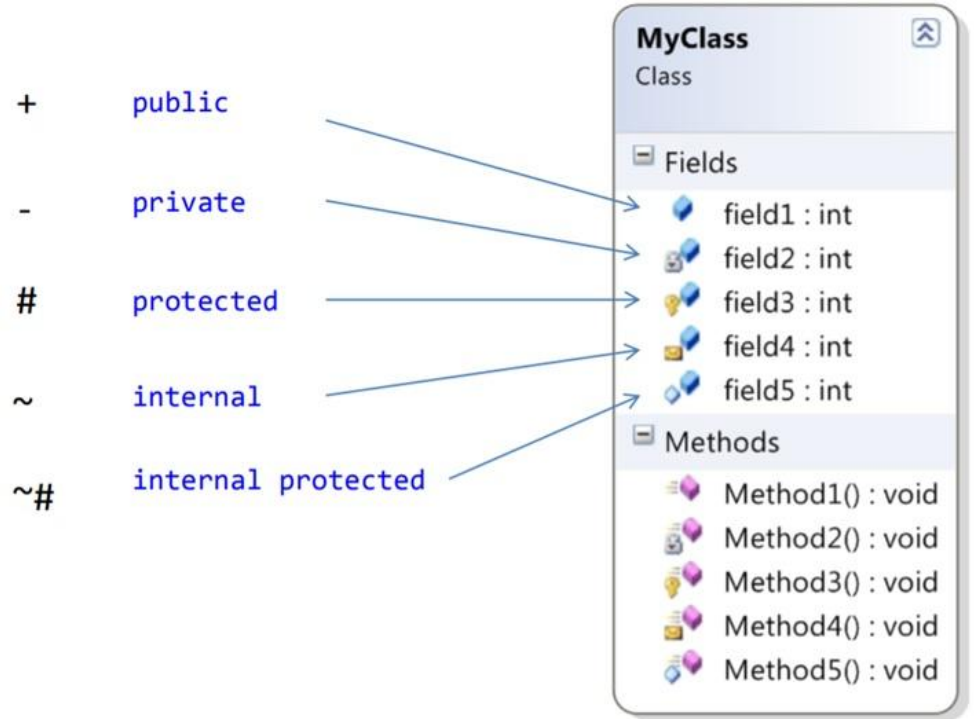


В первой части указывается название класса.

Вторая часть содержит перечень атрибутов класса, которые характеризуют тот или иной объект этого класса в модели предметной области.

Третья часть содержит перечень операций, отражающих его поведение в модели предметной области.

Спецификаторы доступа обозначаются специальными значками.



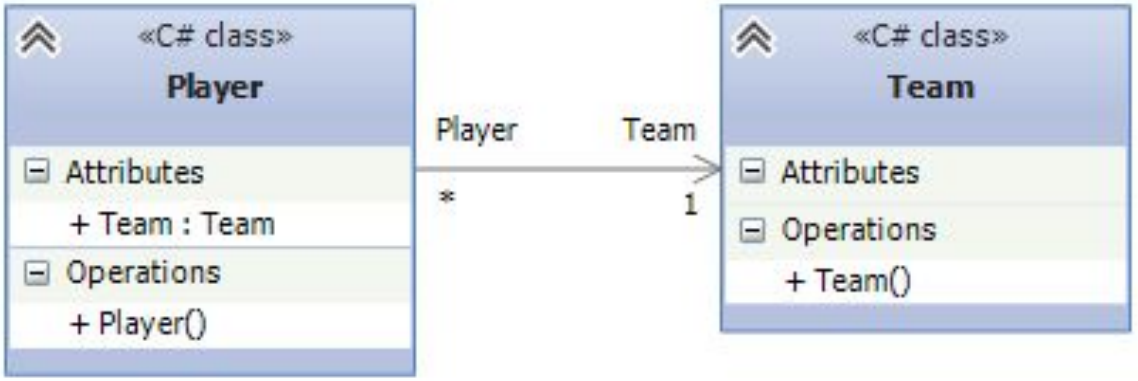
Отношения в UML

Отношение ассоциации

Отношение ассоциации - наличие некоторого отношения между классами.

- Данное отношение обозначается сплошной линией (либо в виде обычно стрелки) с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации. В качестве дополнительных специальных символов могут использоваться имя ассоциации, а также имена и кратность классов-ролей ассоциации. Имя ассоциации является необязательным элементом ее обозначения. Если оно задано, то записывается с заглавной (большой) буквы рядом с линией соответствующей ассоциации.
- Агрегация и композиция являются частными случаями ассоциации.

При отношении ассоциации указывается **кратность** связей. В данном случае единица у Team и звездочка у Player на диаграмме отражает связь 1 ко многим. То есть одна команда будет соответствовать многим игрокам.

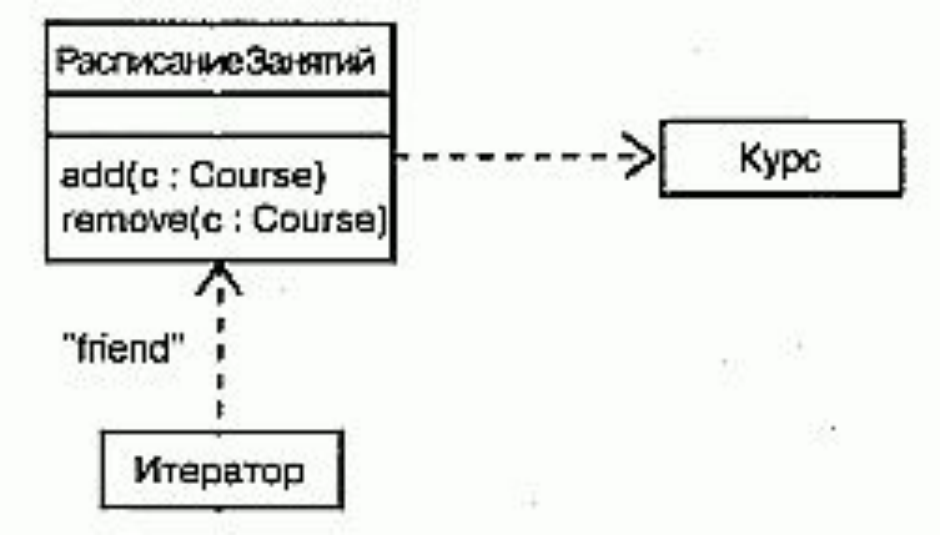



```
class Team
{

}
class Player
{
    public Team Team { get; set; }
}
```

Отношение зависимости

Отношение зависимости в общем случае указывает некоторое отношение между двумя элементами модели или двумя множествами таких элементов, которое не является отношением ассоциации, обобщения или реализации.

- Отношение зависимости используется в такой ситуации, когда некоторое изменение одного (исходного) элемента модели может вызвать изменения другого зависимого (целевого) от него элемента модели.



Отношение обобщения

Отношение обобщения - обычное отношение между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).

- Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML.
- Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.
- На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее отсутствие - на более специальный класс (класс-потомок или подкласс).

```
class User
{
    public int Id { get; set; }
    public string Name { get; set; }
}

class Manager : User
{
    public string Company{ get; set; }
}
```

Отношение композиции

Отношение композиции служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого.

- Композиция определяет отношение **HAS A**, то есть отношение «имеет».
- Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.
- На диаграммах UML отношение композиции проявляется в обычной стрелке от главной сущности к зависимой, при этом со стороны главной сущности, которая содержит, объект второй сущности, располагается закрашенный ромбик

Класс автомобиля полностью управляет жизненным циклом объекта двигателя. При уничтожении объекта автомобиля в области памяти вместе с ним будет уничтожен и объект двигателя. И в этом плане объект автомобиля является главным, а объект двигателя - зависимым.

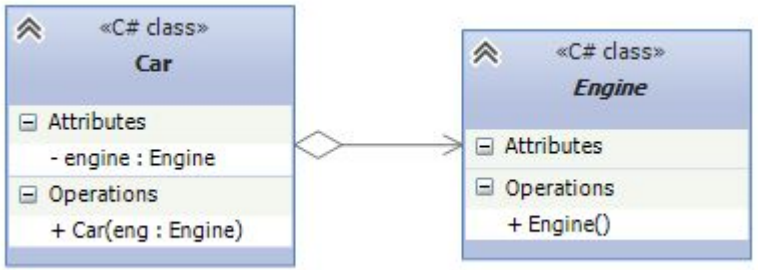

```
public class ElectricEngine
{ }

public class Car
{
    ElectricEngine engine;
    public Car()
    {
        engine = new ElectricEngine();
    }
}
```

Отношение агрегации

Отношение агрегации также предполагает отношение **HAS A**, но реализуется она иначе. Ее следует отличать от композиции.

- Отношение агрегации на диаграммах UML отображается также, как и отношение композиции, только теперь ромбик будет незакрашенным.

При агрегации реализуется слабая связь, то есть в данном случае объекты Car и Engine будут равноправны. В конструктор Car передается ссылка на уже имеющийся объект Engine. И, как правило, определяется ссылка не на конкретный класс, а на абстрактный класс или интерфейс, что увеличивает гибкость программы.




```
public abstract class Engine
{ }

public class Car
{
    Engine engine;
    public Car(Engine eng)
    {
        engine = eng;
    }
}
```

Диаграммы UML

Структурные диаграммы

Диаграммы поведения

- Аналог схемы алгоритмов.
- **Диаграмма состояний** – известное средство описания поведения систем.
- Она определяют все возможные состояния, в кото рых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий

Диаграммы взаимодействия

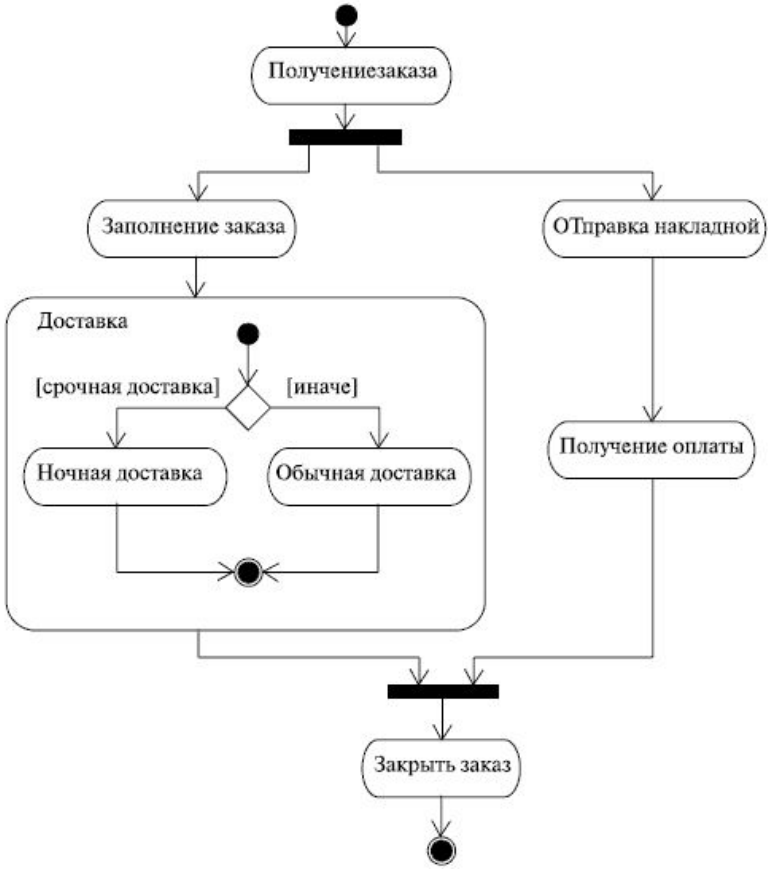
Начальное состояние

Переход

Действие

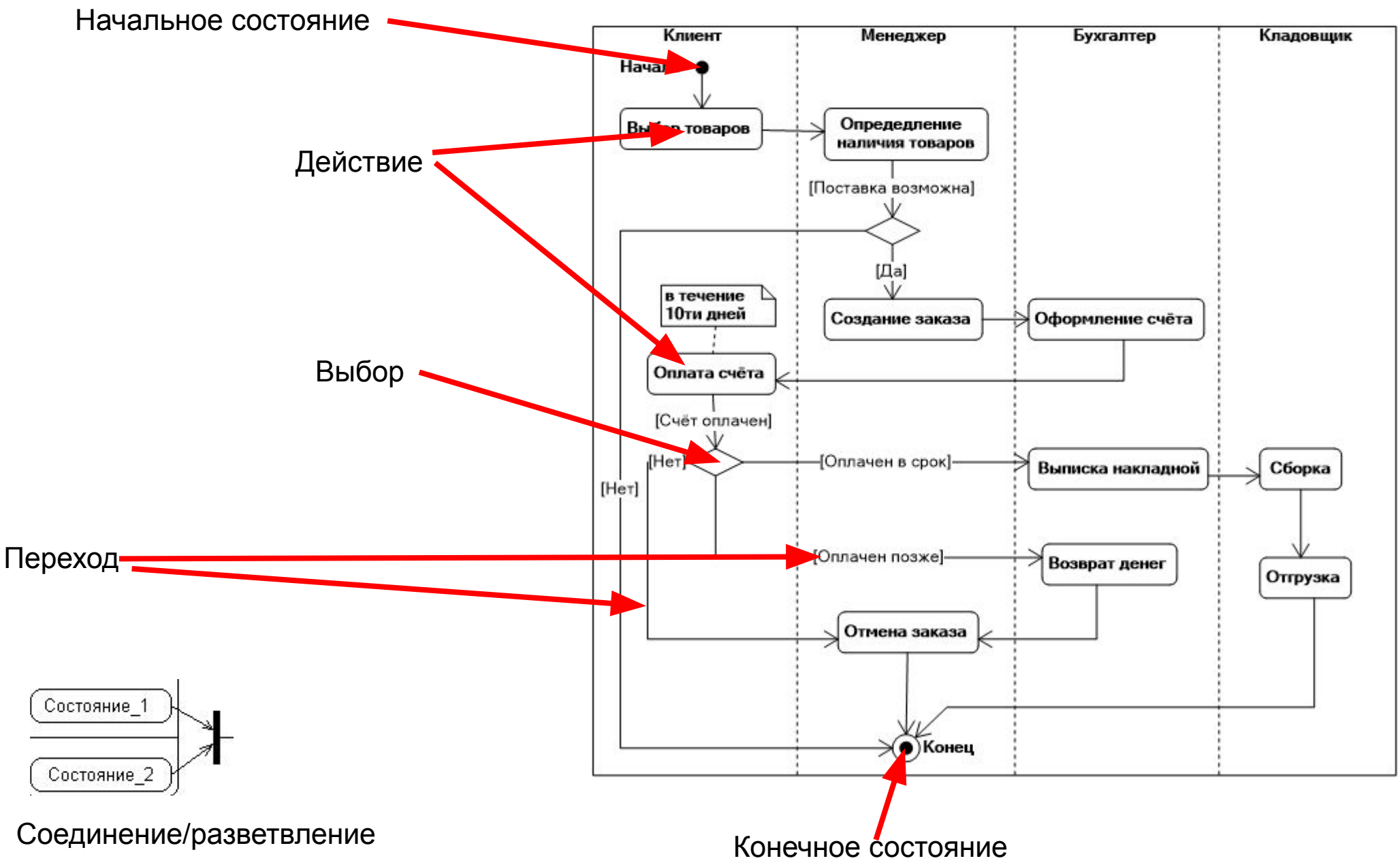
Выбор

Линии (



Конечное состояние

Диаграмма состояний



Диаграммы UML

Структурные диаграммы

Диаграммы поведения

Диаграммы взаимодействия

Диаграмма последовательности - диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

- Диаграмма последовательностей используется для точного определения логики сценария выполнения прецедента.
- Отображает типы объектов, взаимодействующих при исполнении прецедентов, сообщения, которые они посылают друг другу, и любые возвращаемые значения, ассоциированные с этими сообщениями

На диаграмму может быть добавлена управляющая информация:

- описание условий, при которых посылается сообщение;
- признак многократной отправки сообщения (маркер итерации);
- признак возврата сообщения.

Диаграмма последовательности

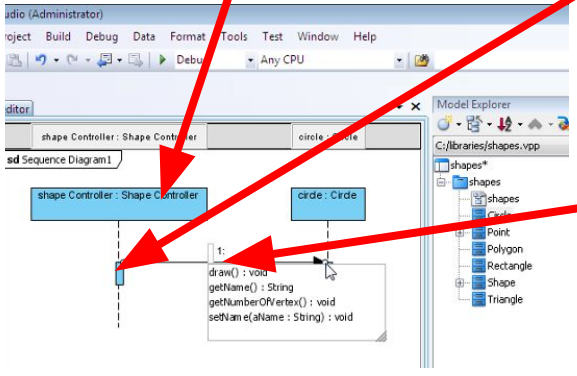
На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами.
Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

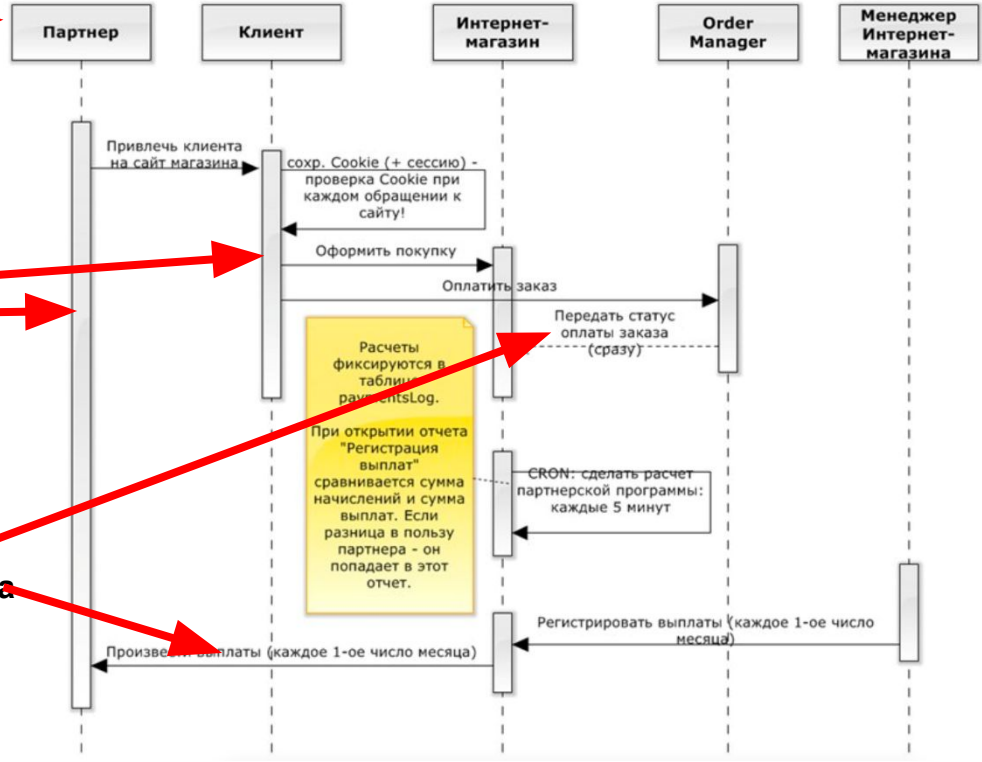
Каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни.

Внутри прямоугольника записываются имя объекта и имя класса, разделенные двоеточием.

"время жизни" объекта

вызов метода у объекта





- Диаграммы классов в Visual Studio https://professorweb.ru/my/programs/visual-studio/level3/3_6.php
- Отношения классов — от UML к коду https://habr.com/ru/post/150041/
- UML (Unified Modeling Language) Диаграммы классов и состояний https://intellect.icu/uml-unified-modeling-language-diagrammy-klassov-i-sostoyanij-4300
- Создание схем классов UML из кода https://docs.microsoft.com/ru-ru/previous-versions/ff657806(v=vs.120)?redirectedfrom=MSDN