
Тема лекции №2.

# Построение моделей транспортного процесса с помощью унифицированного языка моделирования UML.

**Цель лекции:** изучить основы использования унифицированного языка моделирования UML.

**План лекции.**

1. Возможности построения моделей в UML.
2. Элементы UML.
3. Виды диаграмм UML.
4. Диаграмма классов.
5. Реализация диаграмм UML при создании моделей транспортных процессов.

# 1. Возможности построения моделей в UML.

UML (англ.UML (англ. Unified Modeling Language — унифицированный язык моделирования) — языкUML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графическогоUML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделированияUML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания

**UML** является языком широкого профиля, это — открытый стандарт является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML-моделью*.

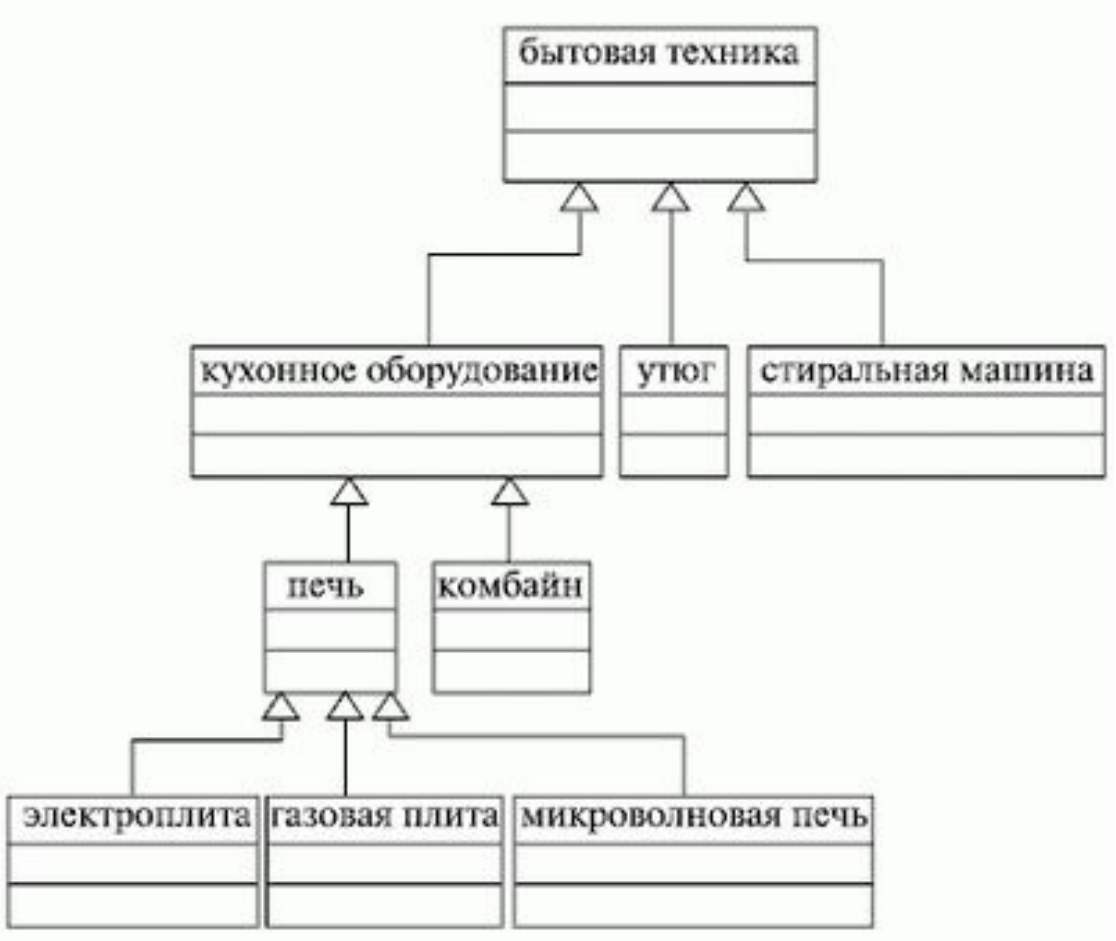
**UML** был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. **UML** не является языком программирования, но на основании *UML-моделей* возможна генерация кода.

**UML** позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких

# Задачи языка UML:

1. Предоставить в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения.
2. Снабдить исходные понятия языка UML возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области.
3. Описание языка UML должно поддерживать такую спецификацию моделей, которая не зависит от конкретных языков программирования и инструментальных средств проектирования программных систем.
4. Поощрять развитие рынка объектных инструментальных средств. Способствовать распространению объектных технологий и соответствующих понятий *объектно-ориентированного анализа и проектирования* (ООАП).
5. Интегрировать в себя новейшие и наилучшие достижения практики ООАП .

# Пример простейшего вида UML модели

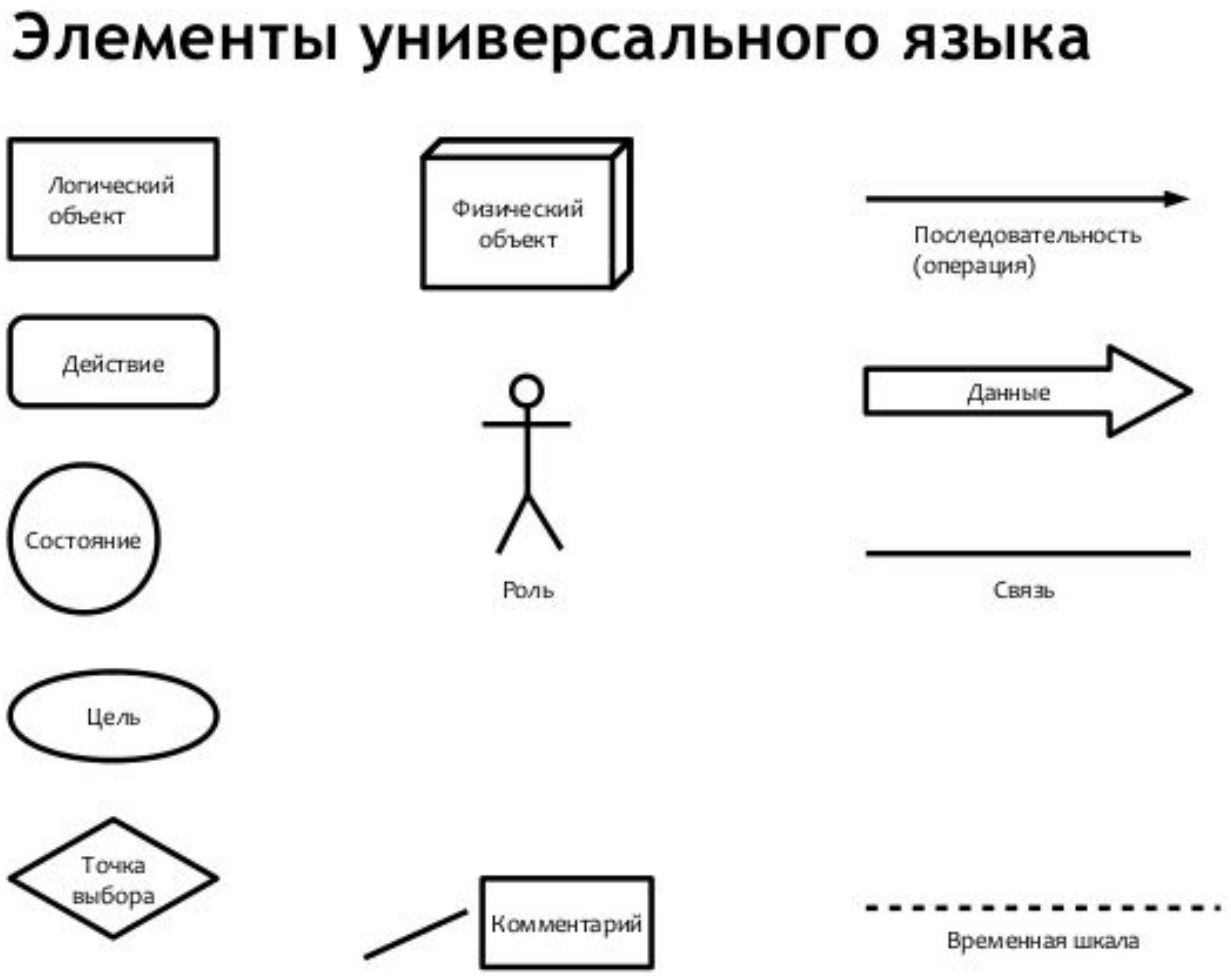


# 2. Элементы UML. Основные понятия визуального моделирования

- **Нотация** – система условных обозначений для графического представления визуальных моделей
- **Семантика** – система правил и соглашений, определяющая смысл и интерпретацию конструкций некоторого языка
- **Методология** – совокупность принципов моделирования и подходов к логической организации методов и средств разработки моделей
- **CASE (Computer Aided Software Engineering) – м**етодология разработка программного обеспечения, основанная на комплексном использовании компьютеров не только для написания исходного кода, но и для анализа и моделирования соответствующей предметной области

# Графические элементы UML

- фигура (shape);
- линия (line);
- значок (icon);
- текст (text);
- рамка (frame).



# Структурные сущности

*Объект* (object) (1) – сущность, обладающая уникальностью и инкапсулирующая в себе состояние и поведение.

*Класс* (class) (2) – описание множества объектов с общими атрибутами, определяющими состояние, и операциями, определяющими поведение.

*Интерфейс* (interface) (3) – именованное множество операций, определяющее набор услуг, которые могут быть запрошены потребителем и предоставлены поставщиком услуг.

*Кооперация* (collaboration) (4) – совокупность объектов, которые взаимодействуют для достижения некоторой цели.

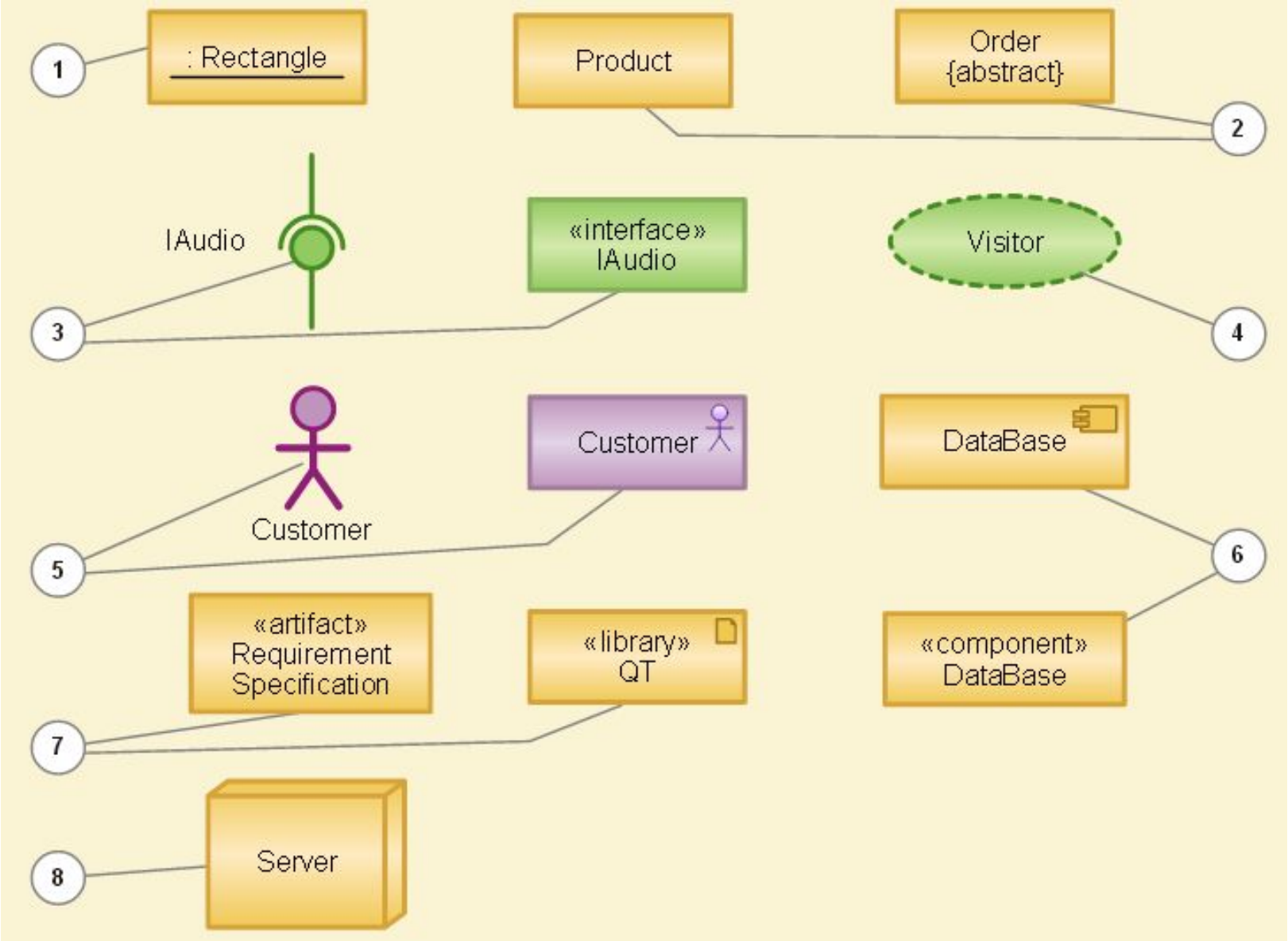
*Действующее лицо* (actor) (5) – сущность, находящаяся вне моделируемой системы и непосредственно взаимодействующая с ней.

*Компонент* (component) (6) – модульная часть системы с четко определенным набором требуемых и предоставляемых интерфейсов.

*Артефакт* (artifact) (7) – элемент информации, который используется или порождается в процессе разработки программного обеспечения. Другими словами, артефакт – это физическая единица реализации, получаемая из элемента модели (например, класса или компонента).

*Узел* (node) (8) – вычислительный ресурс, на котором размещаются и при необходимости выполняются артефакты.

# Нотации структурных сущностей



В UML используются четыре основных типа отношений:

- зависимость (dependency);
- ассоциация (association);
- обобщение (generalization);
- реализация (realization).

*Отношение* ***зависимости*** указывает на то, что изменение независимой сущности каким-то образом влияет на зависимую сущность.

Графически отношение зависимости изображается в виде пунктирной линии со стрелкой 1, направленной от зависимой сущности 2 к независимой 3, как показано на следующем рисунке. Как правило, семантика конкретной зависимости уточняется в модели с помощью дополнительной информации. Например, зависимость со стереотипом «use» означает, что зависимая сущность использует (скажем, вызывает операцию) независимую сущность.

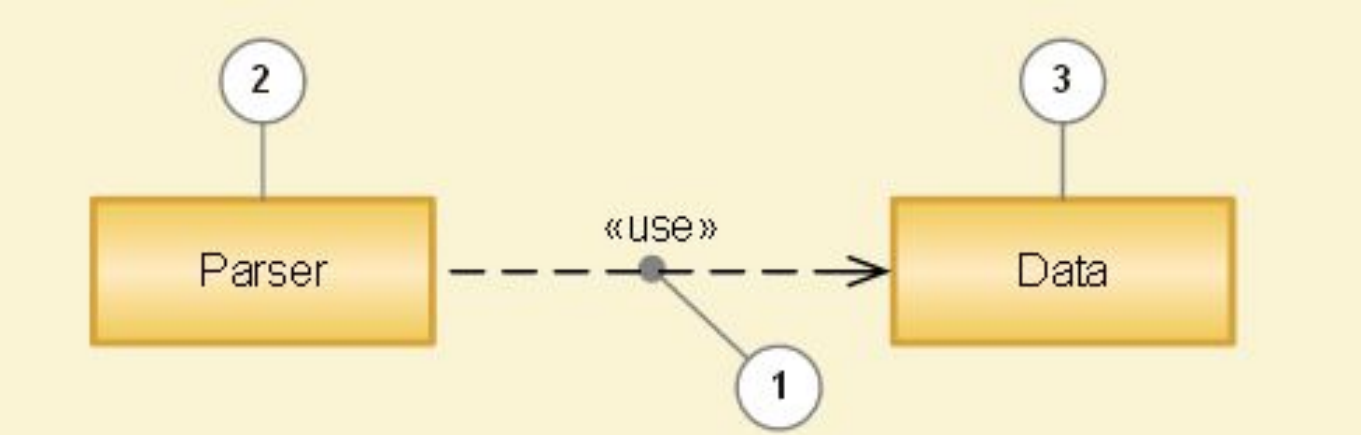



Рис. **Отношение зависимости**

*Отношение* ***ассоциации*** имеет место, если одна сущность непосредственно связана с другой.

Графически ассоциация изображается в виде сплошной линии 1 с различными дополнениями, соединяющей связанные сущности, как показано на следующем рисунке. На программном уровне непосредственная связь может быть реализована различным образом, главное, что ассоциированные сущности знают друг о друге. Например, отношение часть-целое является частным случаем ассоциации и называется отношением агрегации.

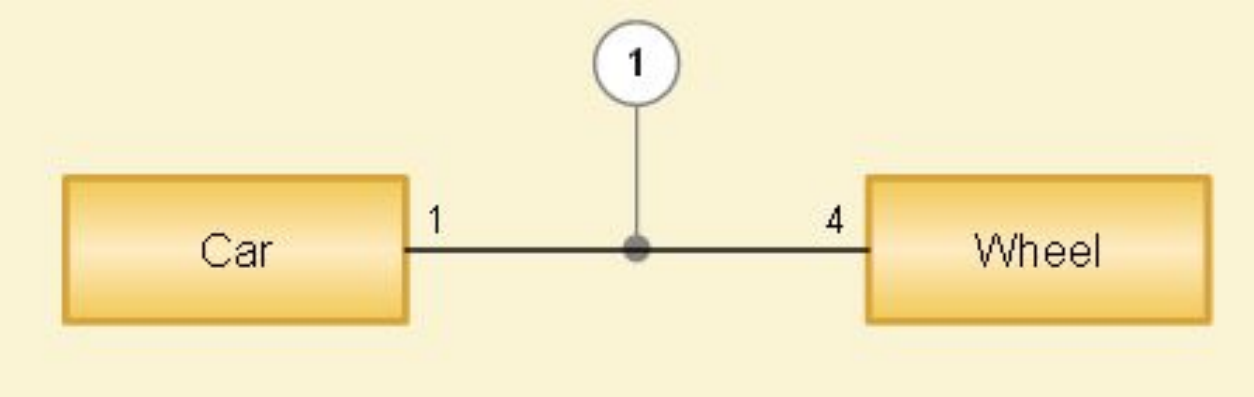



Рис. **Отношение ассоциации**

***Обобщение*** – это отношение между двумя сущностями, одна их которых является частным (специализированным) случаем другой.

Графически обобщение изображается в виде линии с треугольной незакрашенной стрелкой на конце 1, направленной от частного 2 (подкласса) к общему 3 (суперклассу), как показано на следующем рисунке.

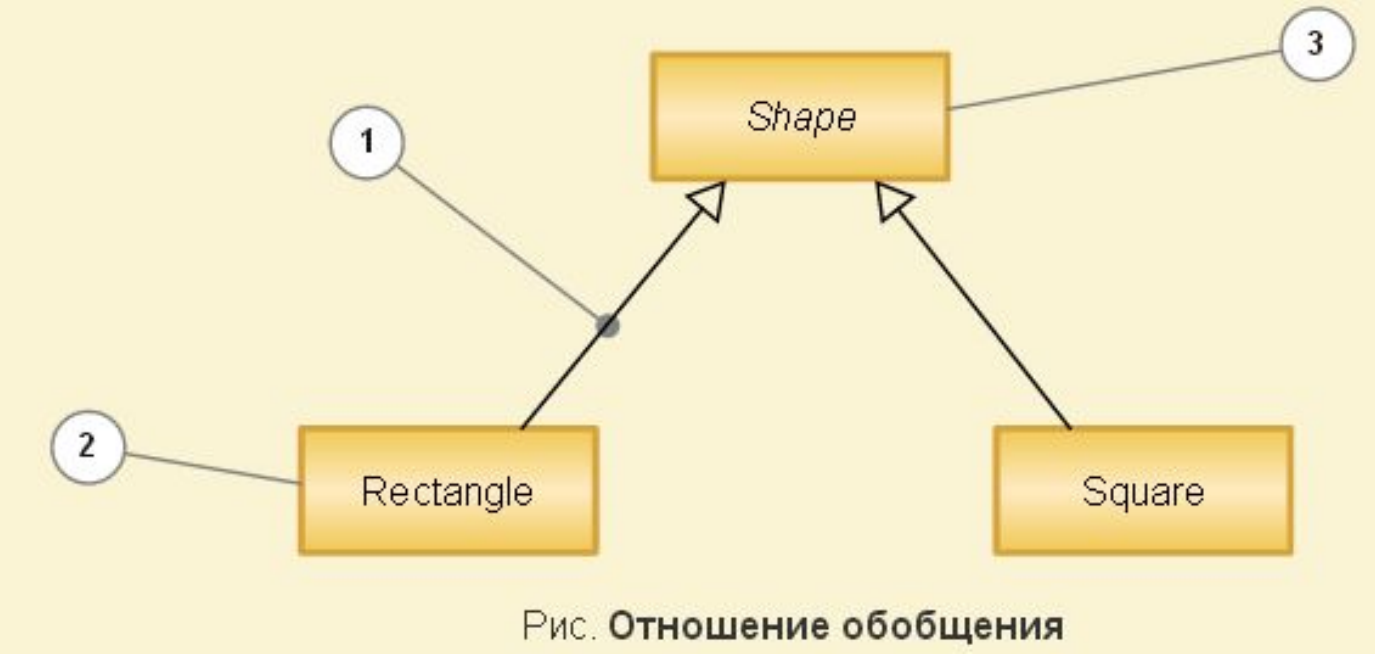
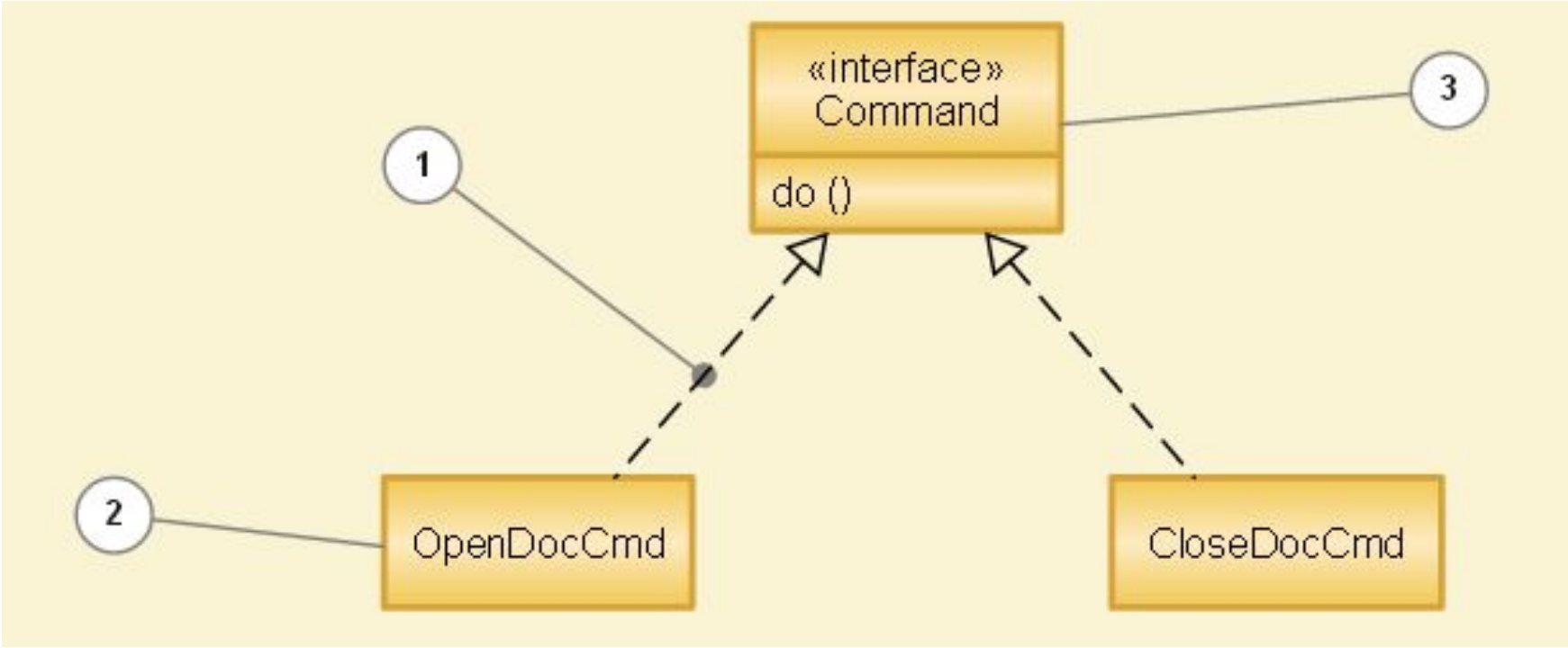



Рис. **Отношение обобщения**

# *Отношение **реализации*** указывает, что одна сущность является реализацией другой.

Графически реализация изображается в виде пунктирной линии с треугольной незакрашенной стрелкой на конце 1, направленной от реализующей сущности 2 к реализуемой 3, как показано на следующем рисунке.



# 3. Виды диаграмм UML.

***Диаграмма*** **(diagram) – это графическое представление некоторой части графа модели.**

Виды диаграмм:

- Диаграмма использования (Use Case diagram)
- Диаграмма классов (Class diagram)
- Диаграмма объектов (Object diagram)
- Диаграмма состояний (State chart diagram)
- Диаграмма деятельности (Activity diagram)
- Диаграмма последовательности (Sequence diagram)
- Диаграмма кооперации (Collaboration diagram)
- Диаграмма компонентов (Component diagram)
- Диаграмма размещения (Deployment diagram)

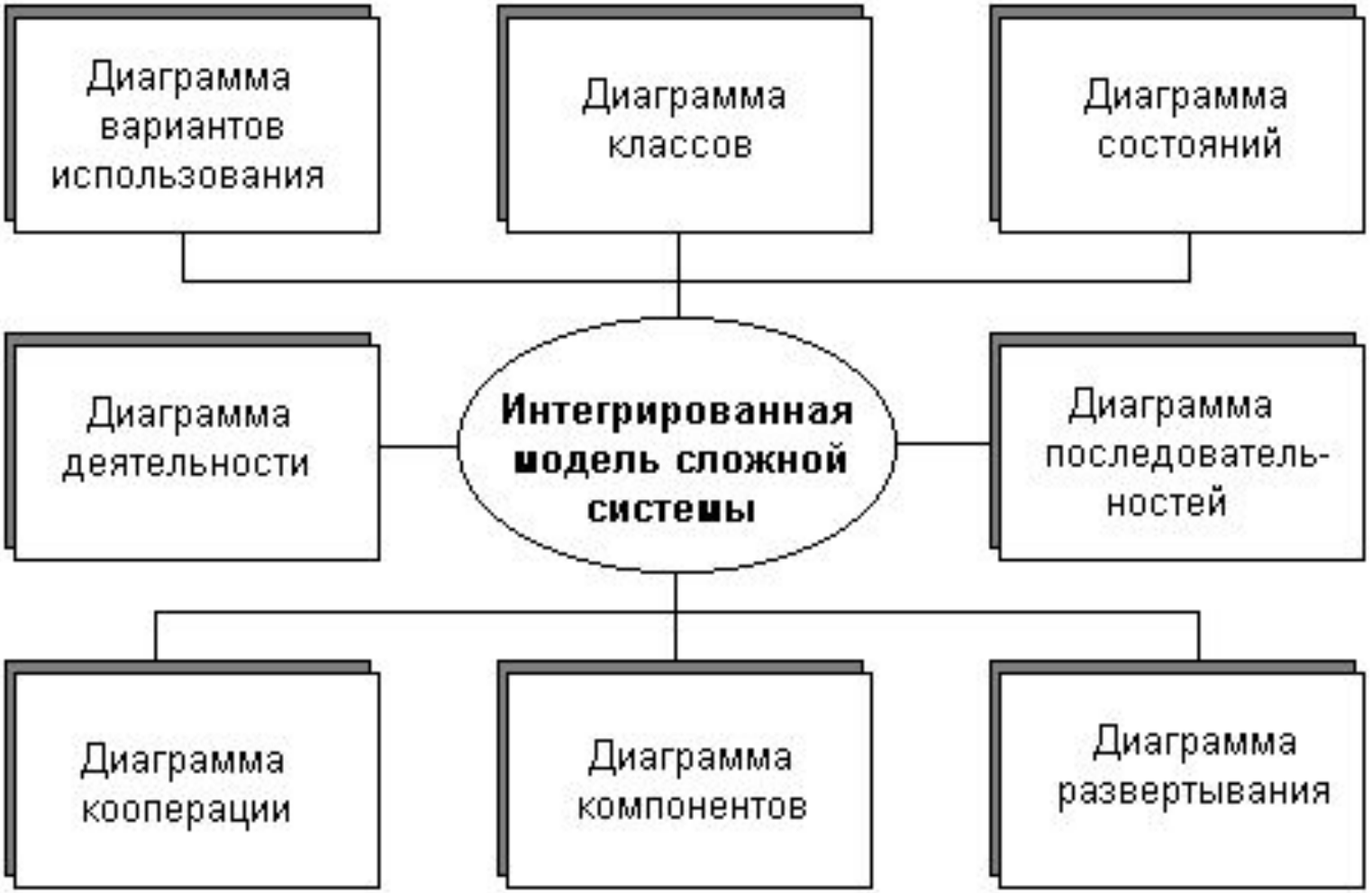
# Общий шаблон представления диаграммы


Рисунок 3.1 - **Нотация для диаграмм**

# Таблица 3.1 - Типы и теги диаграмм

| Название диаграммы | Тег (стандартный) | Тег (предлагаемый) |
|---|---|---|
| Диаграмма использования | **use case** или **uc** | **use case** |
| Диаграмма классов | **class** | **class** |
| Диаграмма автомата | **state machine** или **stm** | **state machine** |
| Диаграмма деятельности | **activity** или **act** | **activity** |
| Диаграмма последовательности | **interaction** или **sd** | **sd** |
| Диаграмма коммуникации | **interaction** или **sd** | **comm** |
| Диаграмма компонентов | **component** или **cmp** | **component** |
| Диаграмма размещения | не определен | **deployment** |
| Диаграмма объектов | не определен | **object** |
| Диаграмма внутренней структуры | **class** | **class** или **component** |
| Обзорная диаграмма взаимодействия | **interaction** или **sd** | **interaction** |
| Диаграмма синхронизации | **interaction** или **sd** | **timing** |
| Диаграмма пакетов | **package** или **pkg** | **package** |

# Интегрированная модель сложной системы в нотации UML



# Диаграмма использования

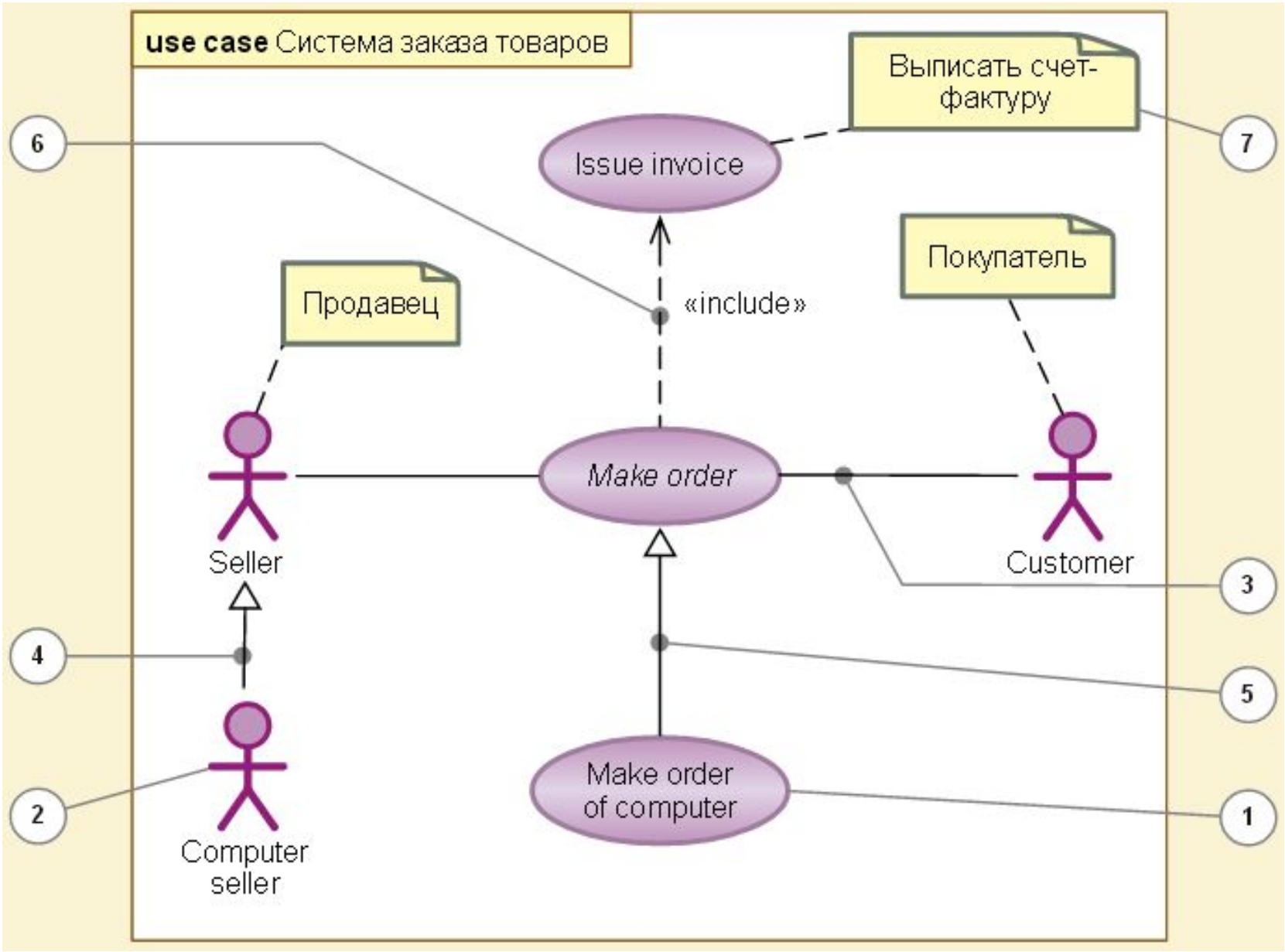
***Диаграмма использования*** **(use case diagram) – это наиболее общее представление функционального назначения системы.**

На диаграмме использования применяются два типа основных сущностей: варианты использования 1 и действующие лица 2, между которыми устанавливаются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования 3;
- обобщение между действующими лицами 4;
- обобщение между вариантами использования 5;
- зависимости (различных типов) между вариантами использования 6.

На диаграмме использования, как и на любой другой, могут присутствовать комментарии 7.

# Нотация диаграммы использования



# Диаграмма объектов

***Диаграмма объектов* (object diagram) – является экземпляром диаграммы классов.**

На диаграмме объектов применяют один основной тип сущностей: объекты 1 (экземпляры классов), между которыми указываются конкретные связи 2 (чаще всего экземпляры ассоциаций).

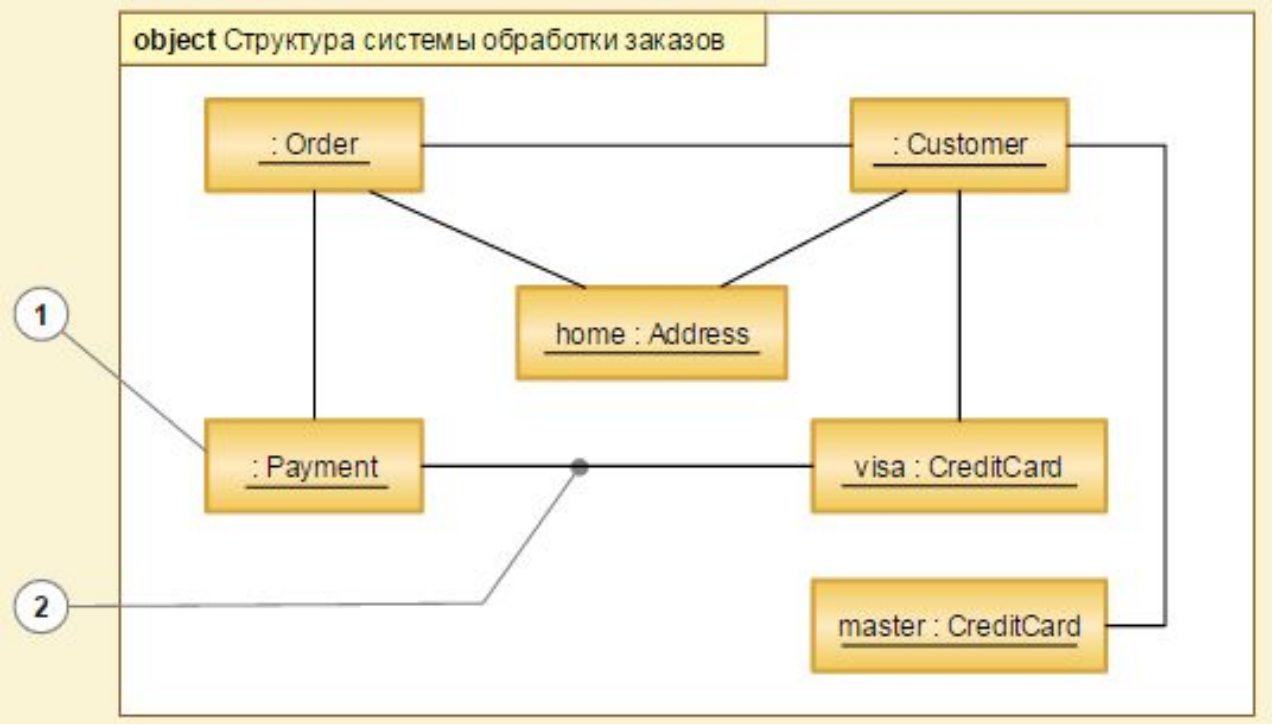


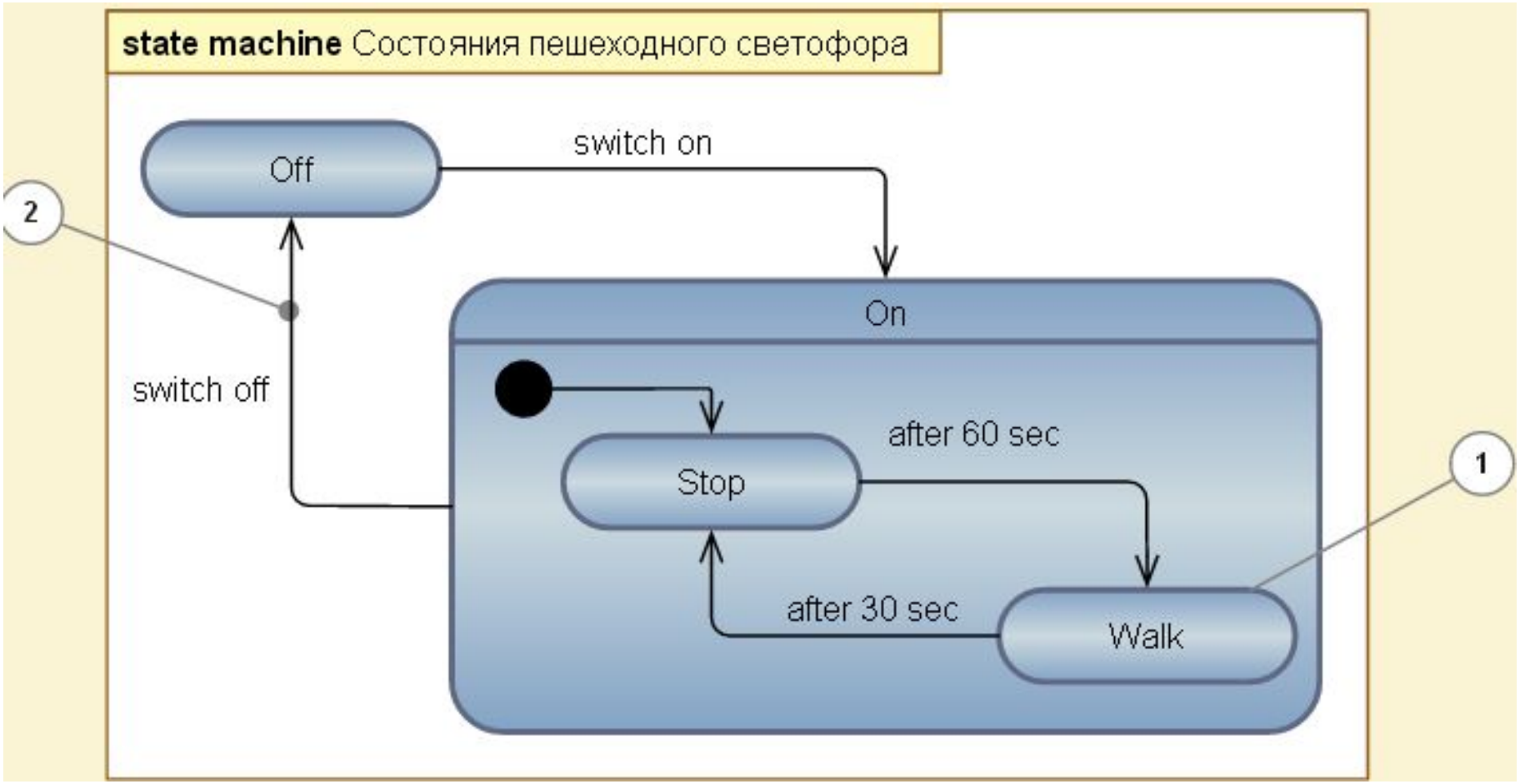

**Рисунок - Нотация диаграммы объектов**

# *Диаграмма автомата*

***Диаграмма автомата*** **(state machine diagram) – это один из способов детального описания поведения в UML на основе явного выделения состояний и описания переходов между состояниями.**

На диаграмме автомата применяют один основной тип сущностей – состояния 1, и один тип отношений – переходы 2, но и для тех и для других определено множество разновидностей, специальных случаев и дополнительных обозначений.

# Нотация диаграммы автомата



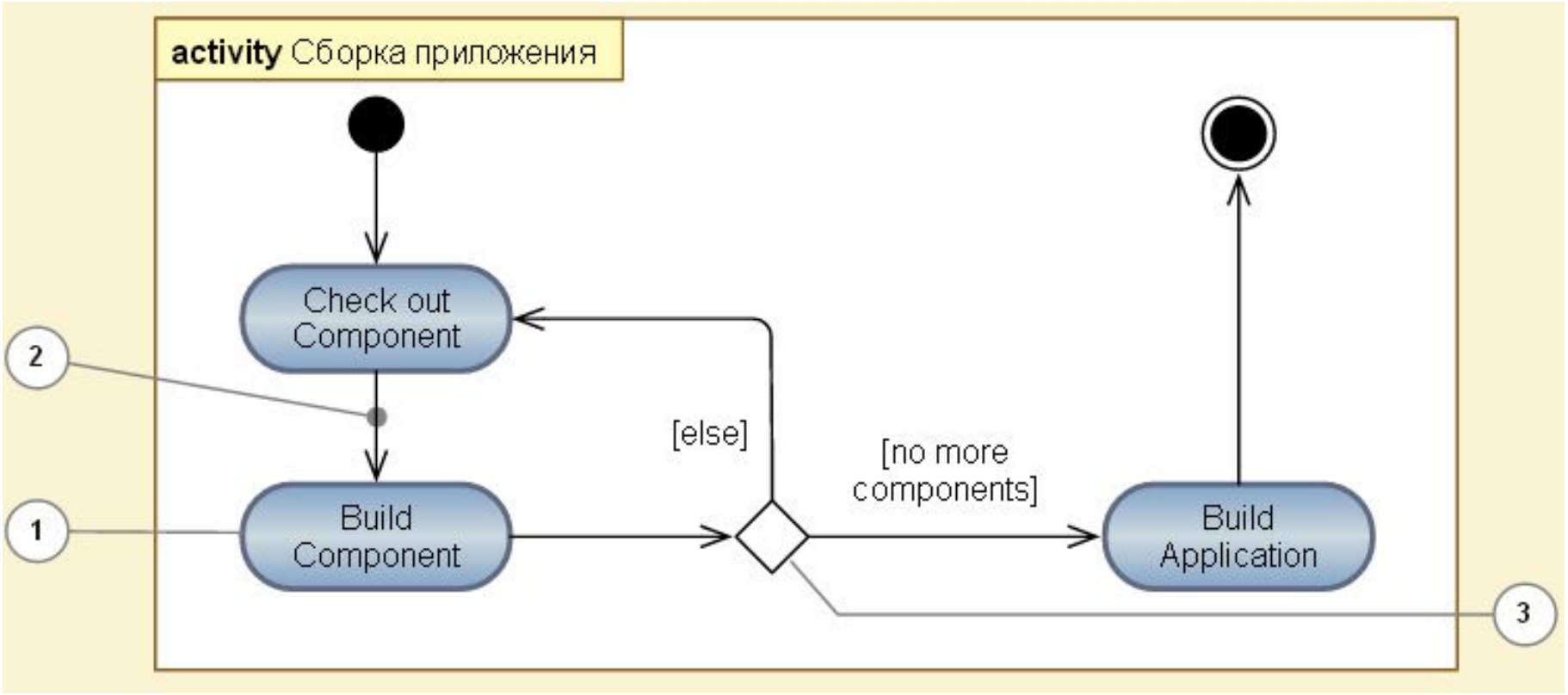
# Диаграмма деятельности

***Диаграмма деятельности*** **(activity diagram) – способ описания поведения на основе указания потоков управления и потоков данных.**

**Диаграмма деятельности** – еще один способ описания поведения, который визуально напоминает старую добрую блок-схему алгоритма. Однако за счет модернизированных обозначений, согласованных с объектно-ориентированным подходом, а главное, за счет новой семантической составляющей (свободная интерпретация сетей Петри), диаграмма деятельности UML является мощным средством для описания поведения системы.

На диаграмме деятельности применяют один основной тип сущностей – действие 1, и один тип отношений – переходы 2 (передачи управления и данных). Также используются такие конструкции как развилки, слияния, соединения, ветвления 3, которые похожи на сущности, но таковыми на самом деле не являются, а представляют собой графический способ изображения некоторых частных случаев многоместных отношений.

# Нотация диаграммы деятельности

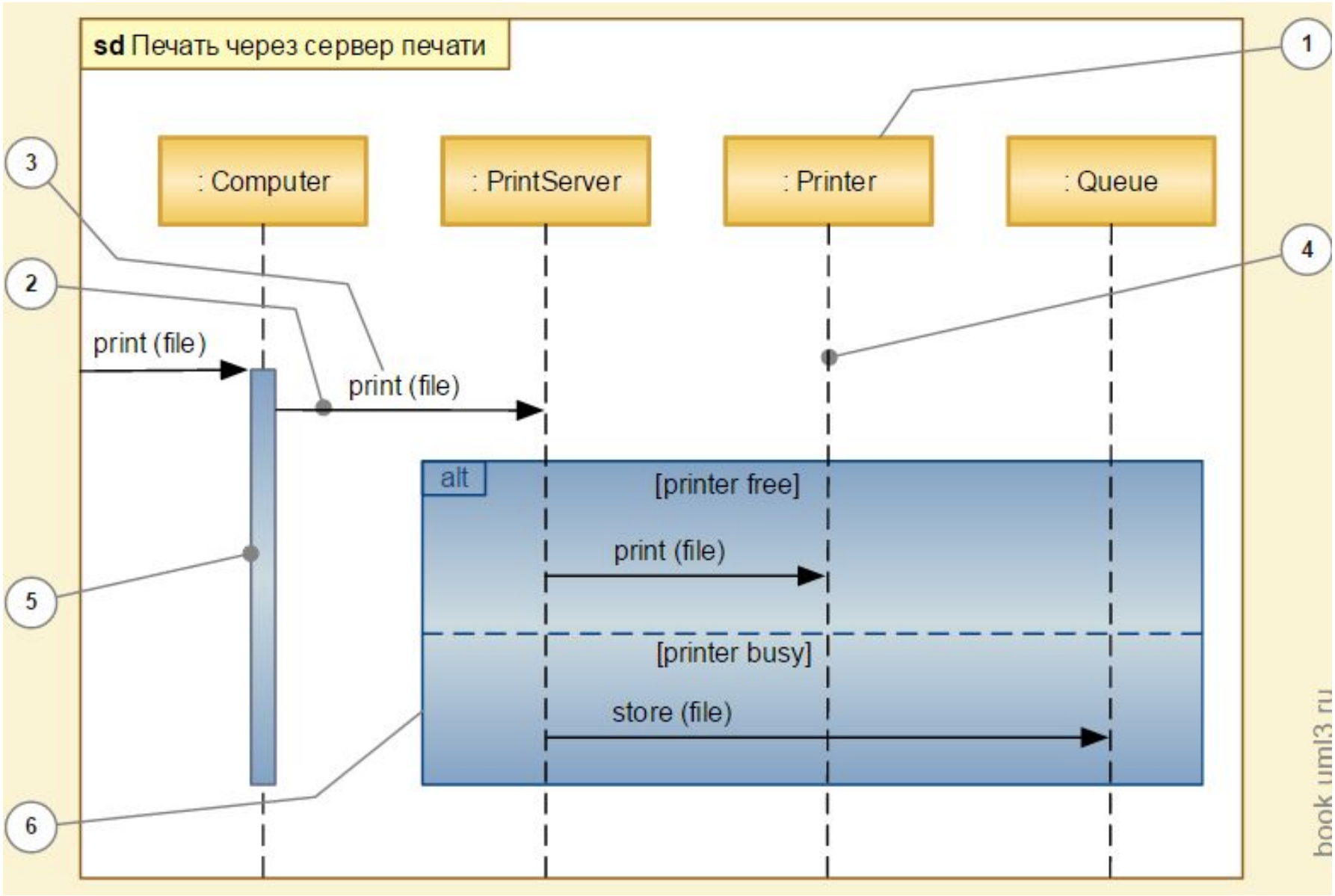


# Диаграмма последовательности

***Диаграмма последовательности*** **(sequence diagram) – это способ описания поведения системы на основе указания последовательности передаваемых сообщений.**

На диаграмме последовательности применяют один основной тип сущностей – экземпляры взаимодействующих классификаторов 1 (в основном классов, компонентов и действующих лиц), и один тип отношений – связи 2, по которым происходит обмен сообщениями 3. Для обозначения самих взаимодействующих объектов применяется стандартная нотация – прямоугольник с именем экземпляра классификатора. Пунктирная линия, выходящая из него, называется линией жизни (lifeline) 4. Графический комментарий, показывающий отрезки времени, в течении которых объект владеет потоком управления (execution occurrence) 5 или другими словами имеет место активация (activation) объекта. Составные шаги взаимодействия(combined fragment) 6 позволяют на диаграмме последовательности, отражать и алгоритмические аспекты протокола взаимодействия.

# Нотация диаграммы последовательности

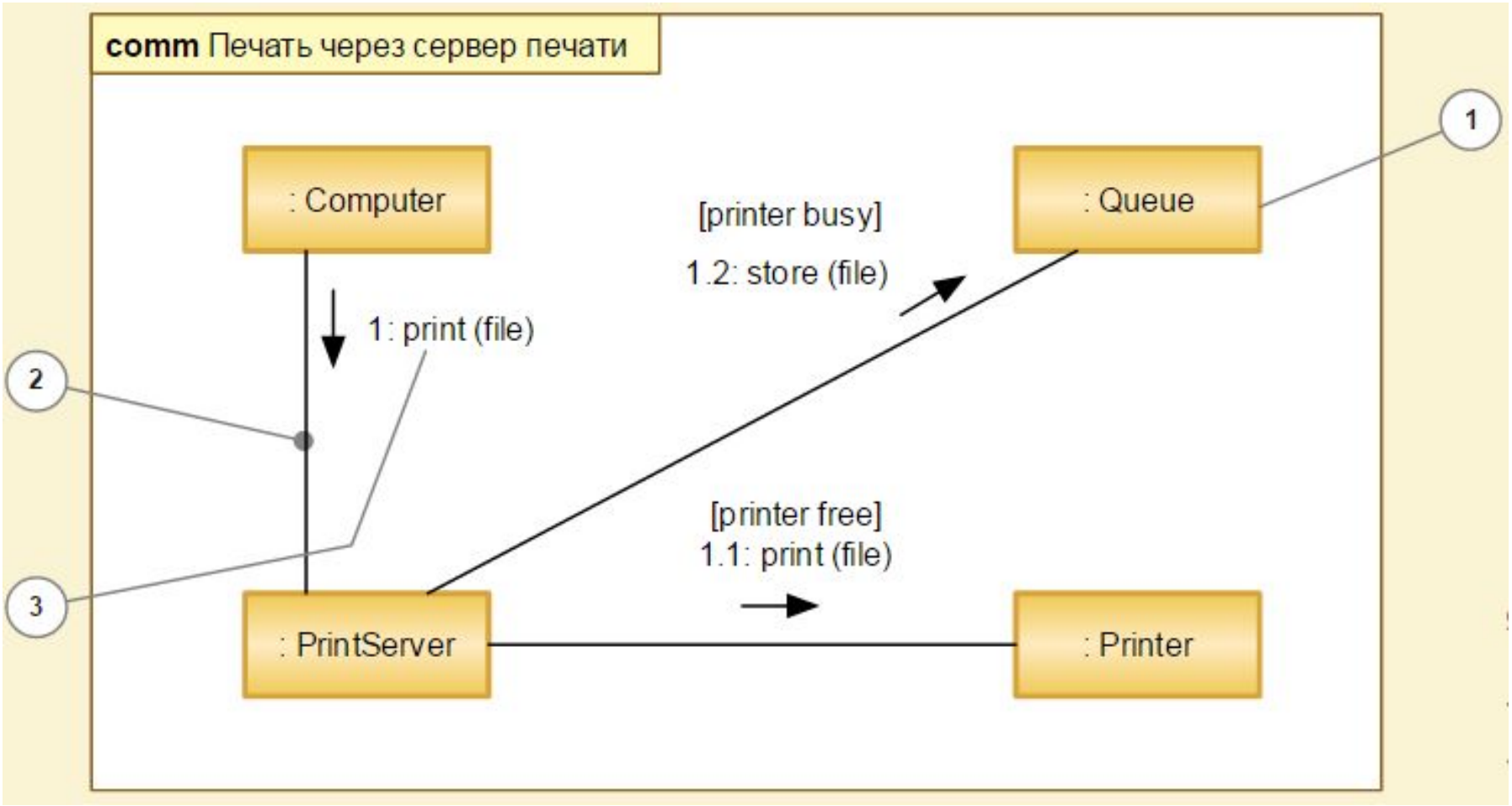


# Диаграмма коммуникации

***Диаграмма коммуникации*** **(communication diagram) – способ описания поведения, семантически эквивалентный диаграмме последовательности.**

На диаграмме коммуникации также как и на диаграмме последовательности применяют один основной тип сущностей – экземпляры взаимодействующих классификаторов 1 и один тип отношений – связи 2. Однако здесь акцент делается не на времени, а на структуре связей между конкретными экземплярами. Для обозначения самих взаимодействующих объектов применяется стандартная нотация – прямоугольник с именем экземпляра классификатора. Взаимное положение элементов на диаграмме кооперации не имеет значения – важны только связи (чаще всего экземпляры ассоциаций), вдоль которых передаются сообщения 3.

# Нотация диаграммы коммуникации



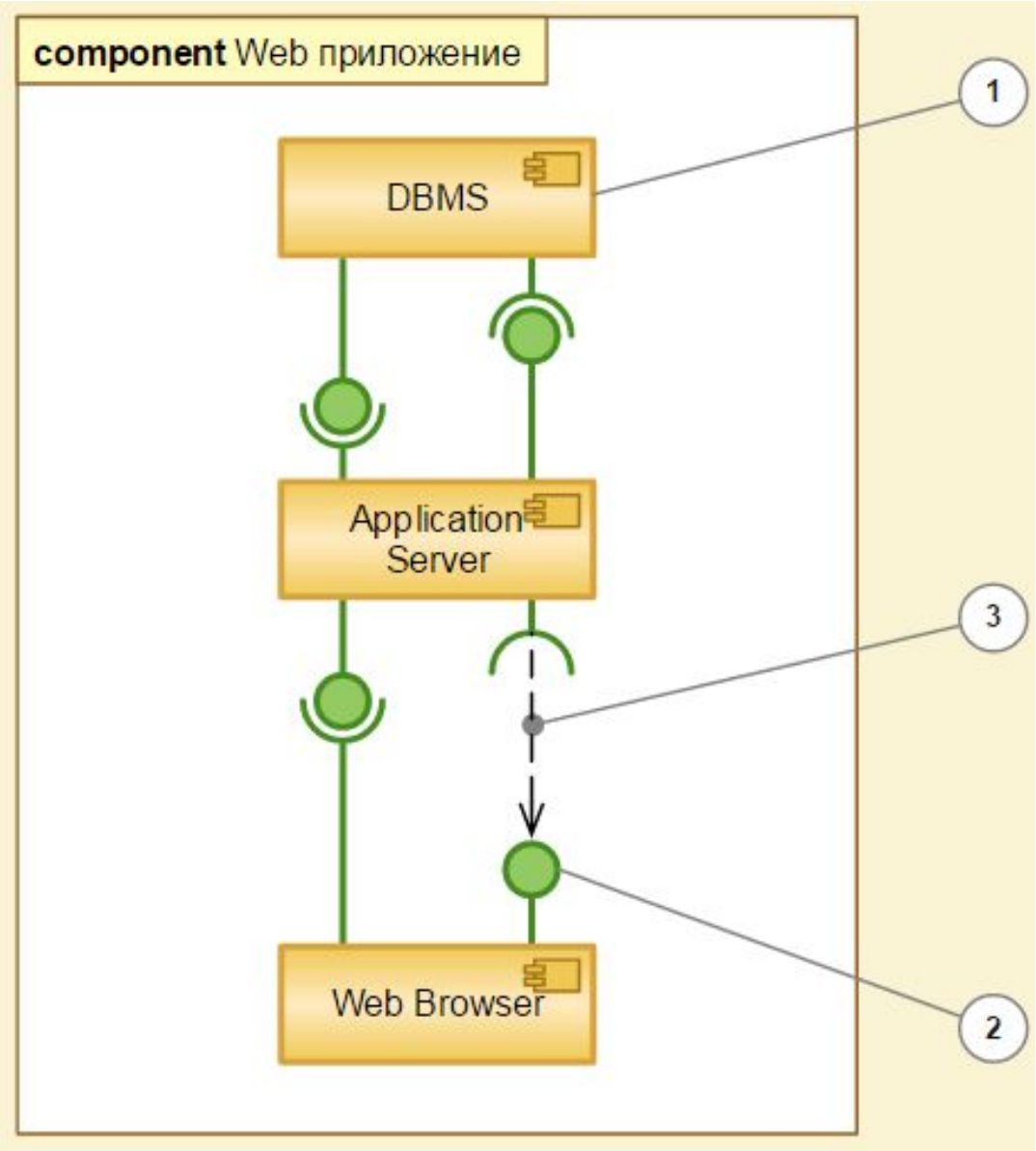
# Диаграмма компонентов

***Диаграмма компонентов*** **(component diagram) – показывает взаимосвязи между модулями (логическими или физическими), из которых состоит моделируемая система.**

Основной тип сущностей на диаграмме компонентов – это сами компоненты 1, а также интерфейсы 2, посредством которых указывается взаимосвязь между компонентами. На диаграмме компонентов применяются следующие отношения:

- реализации между компонентами и интерфейсами (компонент реализует интерфейс);
- зависимости между компонентами и интерфейсами (компонент использует интерфейс) 3.

# Нотация диаграммы компонентов

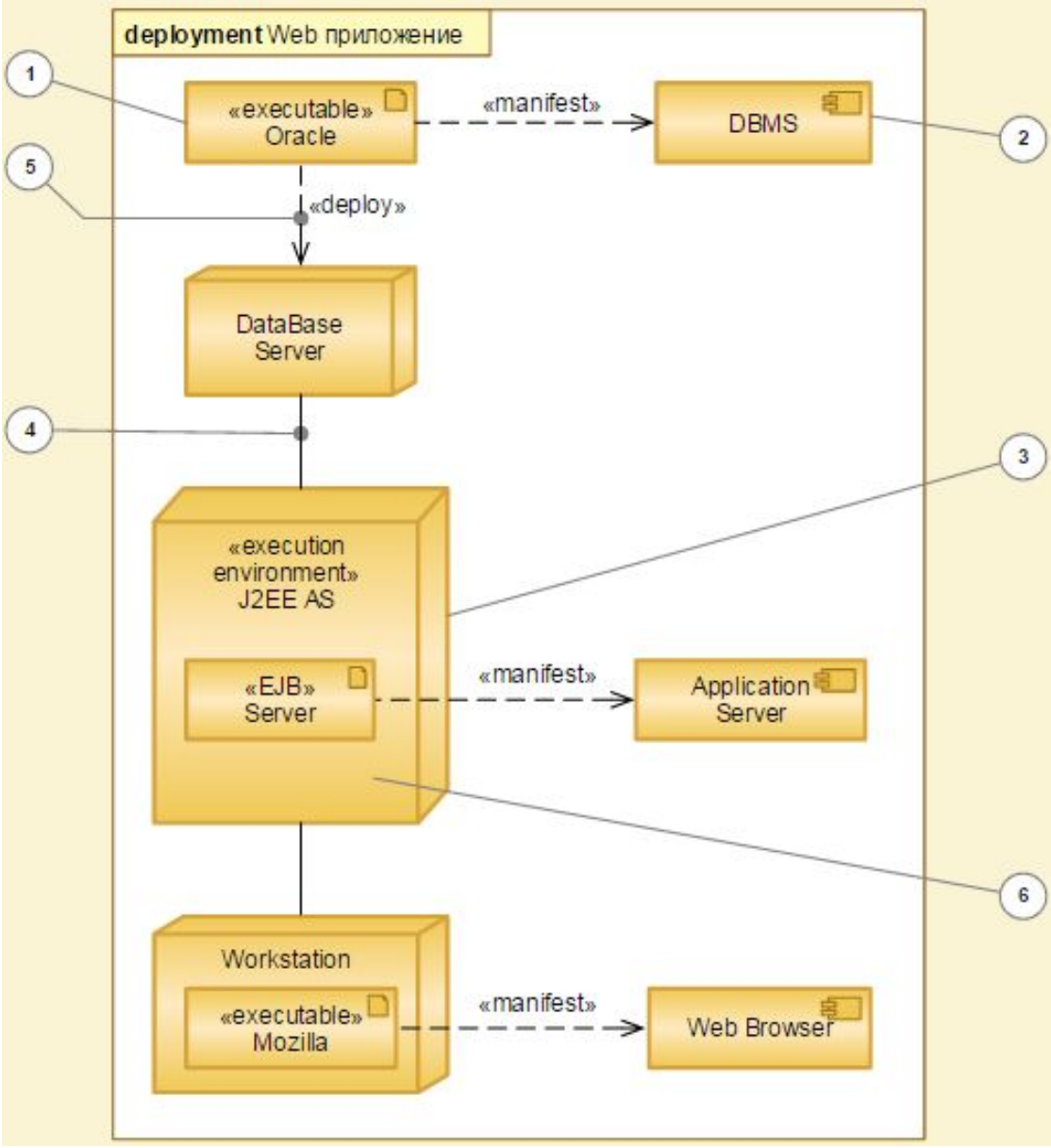


# Диаграмма размещения

***Диаграмма размещения*** **(deployment diagram) наряду с отображением состава и связей элементов системы показывает, как они физически размещены на вычислительных ресурсах во время выполнения.**

На диаграмме размещения, по сравнению с диаграммой компонентов, добавляется два типа сущностей: артефакт 1, который является реализацией компонента 2 и узел 3 (может быть как классификатор, описывающий тип узла, так и конкретный экземпляр), а также отношение ассоциации между узлами 4, показывающее, что узлы физически связаны во время выполнения. Для того чтобы показать, что одна сущность является частью другой, применяется либо отношение зависимости «deploy» 5, либо фигура одной сущности помещается внутрь фигуры другой сущности 6.

# Нотация диаграммы размещения



# 4. Диаграмма классов

***Диаграмма классов* (class diagram) – основной способ описания структуры системы.**

*Особенности:*

1. *Диаграммы классов* используются при моделировании систем, наиболее часто, являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру.
2. *Диаграмма классов* не отображает динамическое поведение объектов изображенных на ней классов.
3. На *диаграммах классов* показываются классы, интерфейсы и отношения между ними.

На диаграмме классов применяется один основной тип сущностей:
классы 1 (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений:

- ассоциация между классами 2 (с множеством дополнительных подробностей);
- обобщение между классами 3;
- зависимости (различных типов) между классами 4 и между классами и интерфейсами.

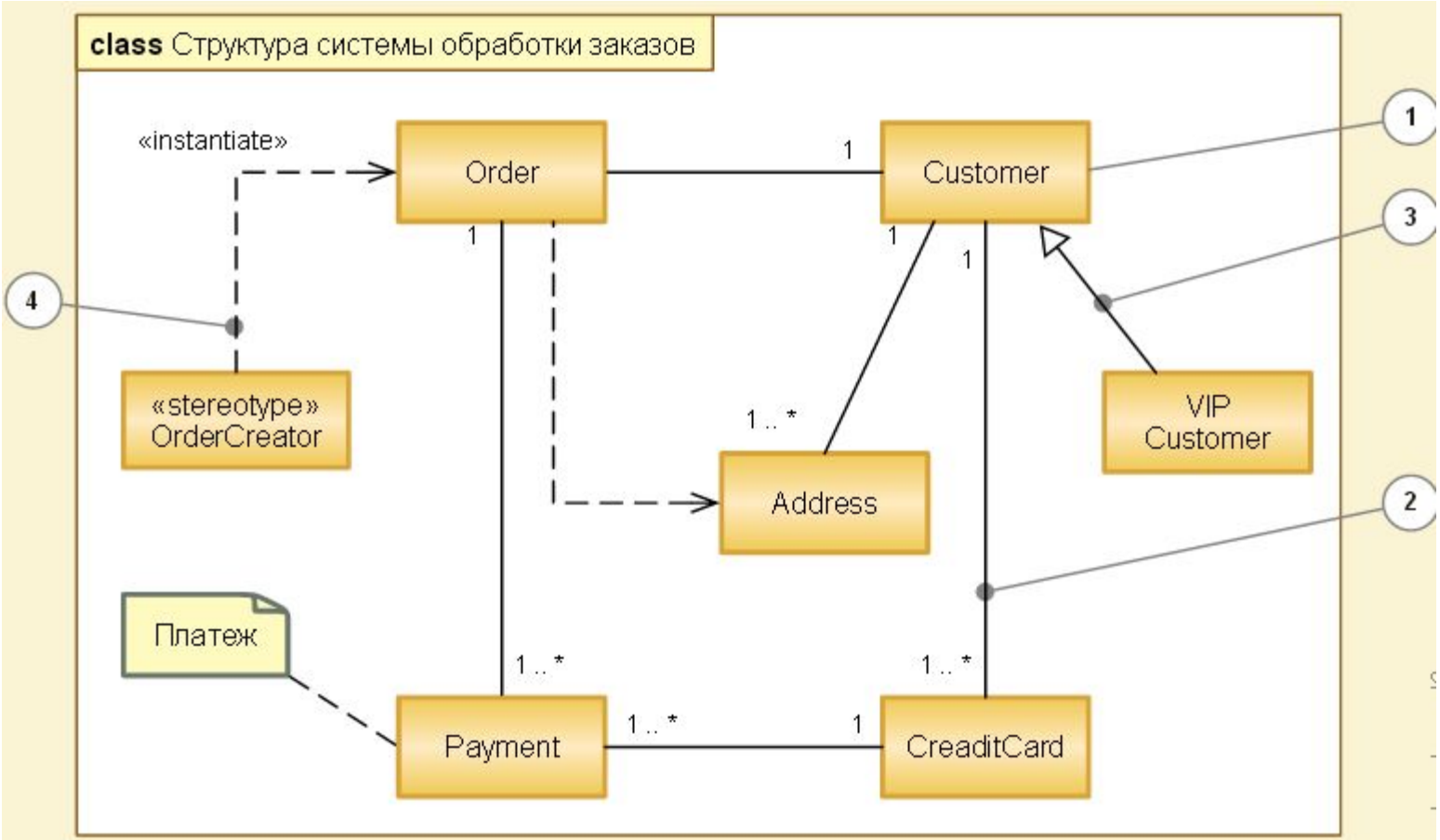
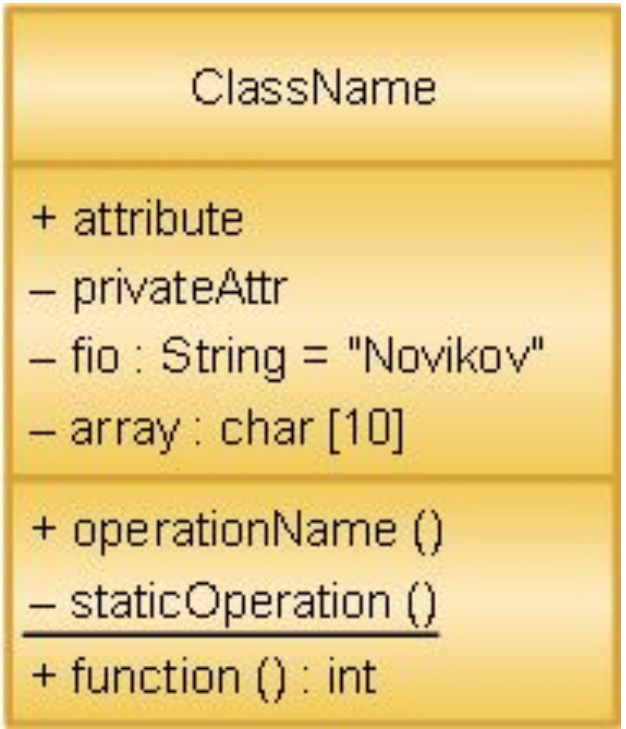



Рис. **Нотация диаграммы классов**

Описание класса может включать множество различных элементов, и чтобы они не путались, в языке предусмотрено группирование элементов описания класса по *секциям* (compartment). Стандартных секций три:

- ***секция имени*** – наряду с обязательным именем может содержать также стереотип, кратность и список именованных значений;
- ***секция атрибутов*** – содержит список описаний атрибутов класса;
- ***секция операций*** – содержит список описаний операций класса.



При формировании имен классов в UML допускается использование произвольной комбинации букв, цифр и даже знаков препинания.
Рекомендуется использовать в качестве имен классов короткие и осмысленные прилагательные и существительные, каждое из которых начинается с заглавной буквы.

**Стандартные стереотипы классов**

| Стереотип | Описание |
|---|---|
| «actor» | Действующее лицо |
| «auxiliary» | Вспомогательный класс |
| «enumeration» | Перечислимый тип данных |
| «exception» | Исключение (только в UML 1) |
| «focus» | Основной класс |
| «implementationClass» | Реализация класса |
| «interface» | Все составляющие абстрактные |
| «metaclass» | Экземпляры являются классами |
| «powertype» | Метакласс, экземплярами которого являются все наследники данного класса (только в UML 1) |
| «process» | Активный класс |
| «thread» | Активный класс (только в UML 1) |
| «signal» | Класс, экземплярами которого являются сигналы |
| «stereotype» | Новый элемент на основе существующего |
| «type» | Тип данных |
| «dataType» | Тип данных |
| «utility» | Нет экземпляров, служба |

# *Атрибут* — это именованное место (или, как говорят, *слот*), в котором может храниться значение.

Атрибуты класса перечисляются в секции атрибутов. В общем случае описание атрибута имеет следующий синтаксис.

видимость ИМЯ кратность : тип = начальное_значение {свойства}

## Примеры описаний атрибутов

| Пример | Пояснение |
|---|---|
| `name` | Минимальное возможное описание – указано только имя атрибута |
| `+name` | Указаны имя и открытая видимость – предполагается, что манипуляции с именем будут производиться непосредственно |
| `-name : String` | Указаны имя, тип и закрытая видимость – манипуляции с именем будут производиться с помощью специальных операций |
| `-name[1..3] : String` | В дополнение к предыдущему указана кратность (для хранения трех составляющих: фамилии, имени и отчества) |
| `-name : String="Novikov"` | Дополнительно указано начальное значение |
| `+name : String{readOnly}` | Атрибут объявлен не меняющим своего значения после начального присваивания и открытым |

***Операция*** **– это спецификация действия с объектом: изменение значения его атрибутов, вычисление нового значения по информации, хранящейся в объекте и т.д.**

Описания операций класса перечисляются в секции операций и имеют следующий синтаксис:

**видимость ИМЯ (параметры) : тип {свойства}**

## Примеры описания операций

| Пример | Пояснение |
|---|---|
| `move()` | Минимальное возможное описание – указано только имя операции |
| `+move(in from, in to)` | Указаны видимость операции, направления передачи и имена параметров |
| `+move(in from:Department, in to:Department)` | Подробное описание сигнатуры: указаны видимость операции, направления передачи, имена и типы параметров |
| `+getName():String{isQuery}` | Функция, возвращающая значение атрибута и не имеющая побочных эффектов |

# Категории связей в диаграммах классов

В диаграмме классов могут участвовать связи трех разных категорий:

- зависимость (dependency),
- обобщение (generalization),
- ассоциация (association).

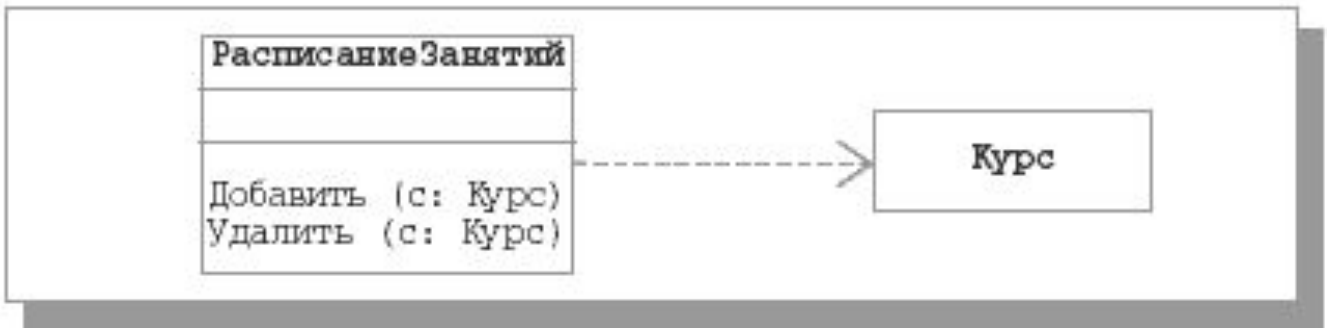



Рисунок - Диаграмма классов со связью-зависимостью (пример)

# 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.

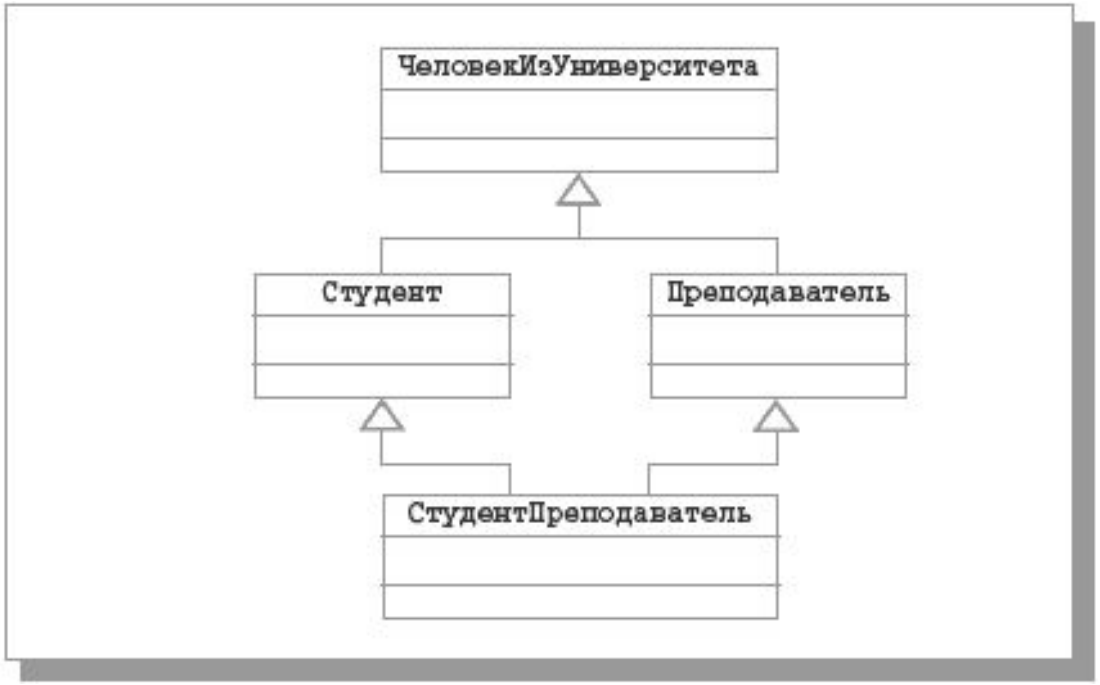



Рисунок - Пример множественного наследования классов

# 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.

С понятием ассоциации связаны четыре важных дополнительных понятия: имя, роль, кратность и агрегация. Во-первых, 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 имя связи.


Рисунок - Пример именованной ассоциации

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.

На рисунке показаны две ассоциации между классами Человек и Университет, в которых эти классы играют разные роли.


Рисунок - Две ассоциации с разными ролями классов

*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. 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.

На диаграмме классов на рисунке показано, что произвольное (может быть, нулевое) число людей являются служащими произвольного числа университетов. Каждый университет обучает произвольное (может быть, нулевое) число студентов, но каждый студент может быть студентом только одного университета.


Рисунок - Ассоциации с указанными кратностями ролей

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».


Рисунок - Пример агрегатной ассоциации

# Советы по проектированию структуры диаграмм классов

1. Описывать структуру удобнее параллельно с описанием поведения. Каждая итерация должна быть небольшим уточнением, как структуры, так и поведения.
2. Не обязательно включать в модель все классы сразу. На первых итерациях достаточно идентифицировать очень небольшую (10%) долю всех классов системы.
3. Не обязательно определять все составляющие класса сразу. Начните с имени класса – операции и атрибуты постепенно выявятся в процессе моделирования поведения.
4. Не обязательно показывать на диаграмме все составляющие класса и их свойства. В процессе работы диаграмма должна легко охватываться одним взглядом.
5. Не обязательно определять все отношения между классами сразу. Пусть класс на диаграмме "висит в воздухе" – ничего с ним не случится.

# 5. Реализация диаграмм UML при создании моделей транспортных процессов.

## Ресурсы для построения UML диаграмм.

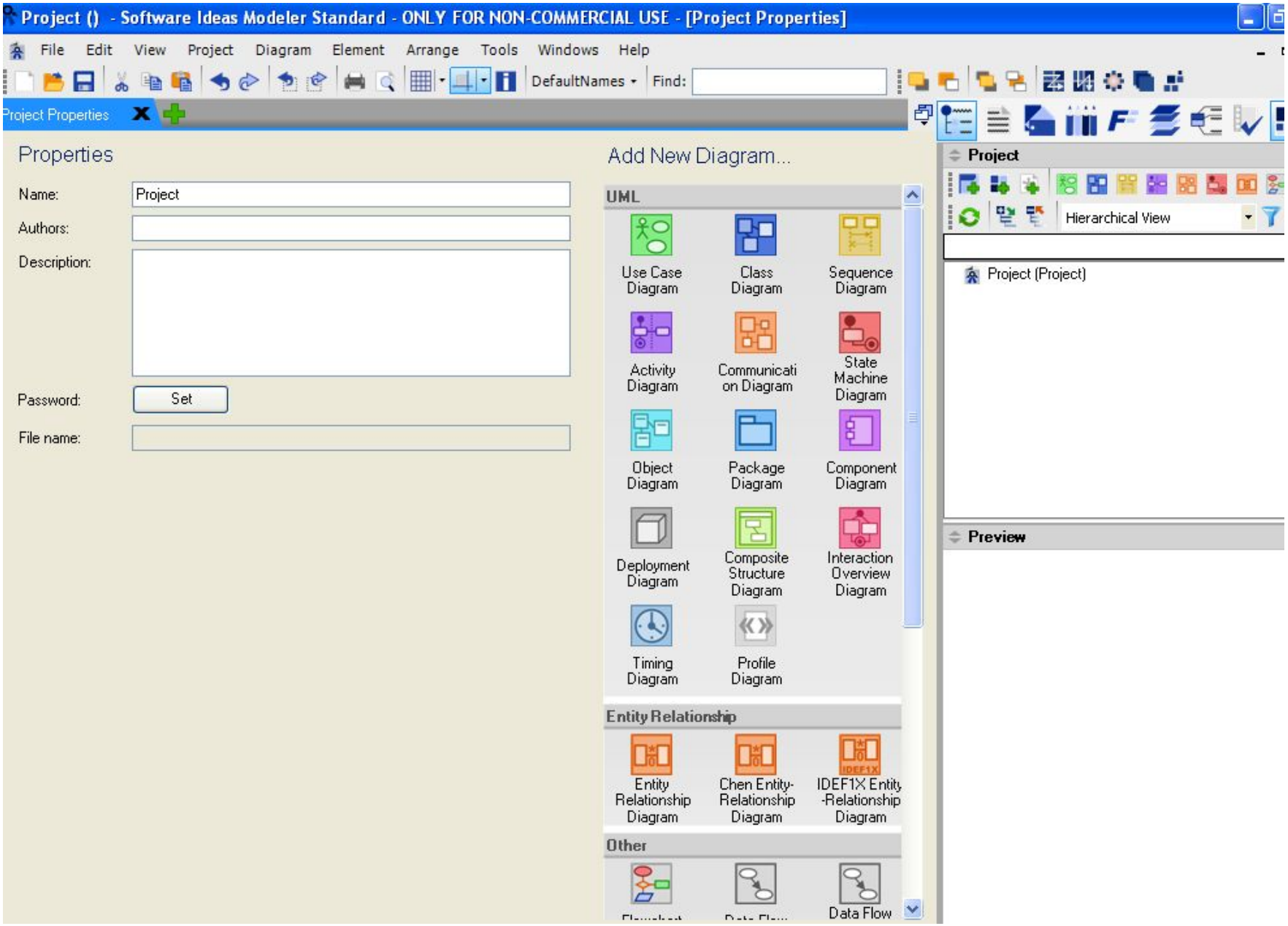
### Программные продукты:

1. MS Visio
2. Software Ideas Modeler
3. Plant UML
4. Dia
5. StartUML и др.

### Онлайн-сервисы:

1. creately.coma
2. Uml.diagrams.org
3. Omg.org и др.

# Software Ideas Modeler



# Dia (http://soft.mydiv.net/win/download-Dia.html)

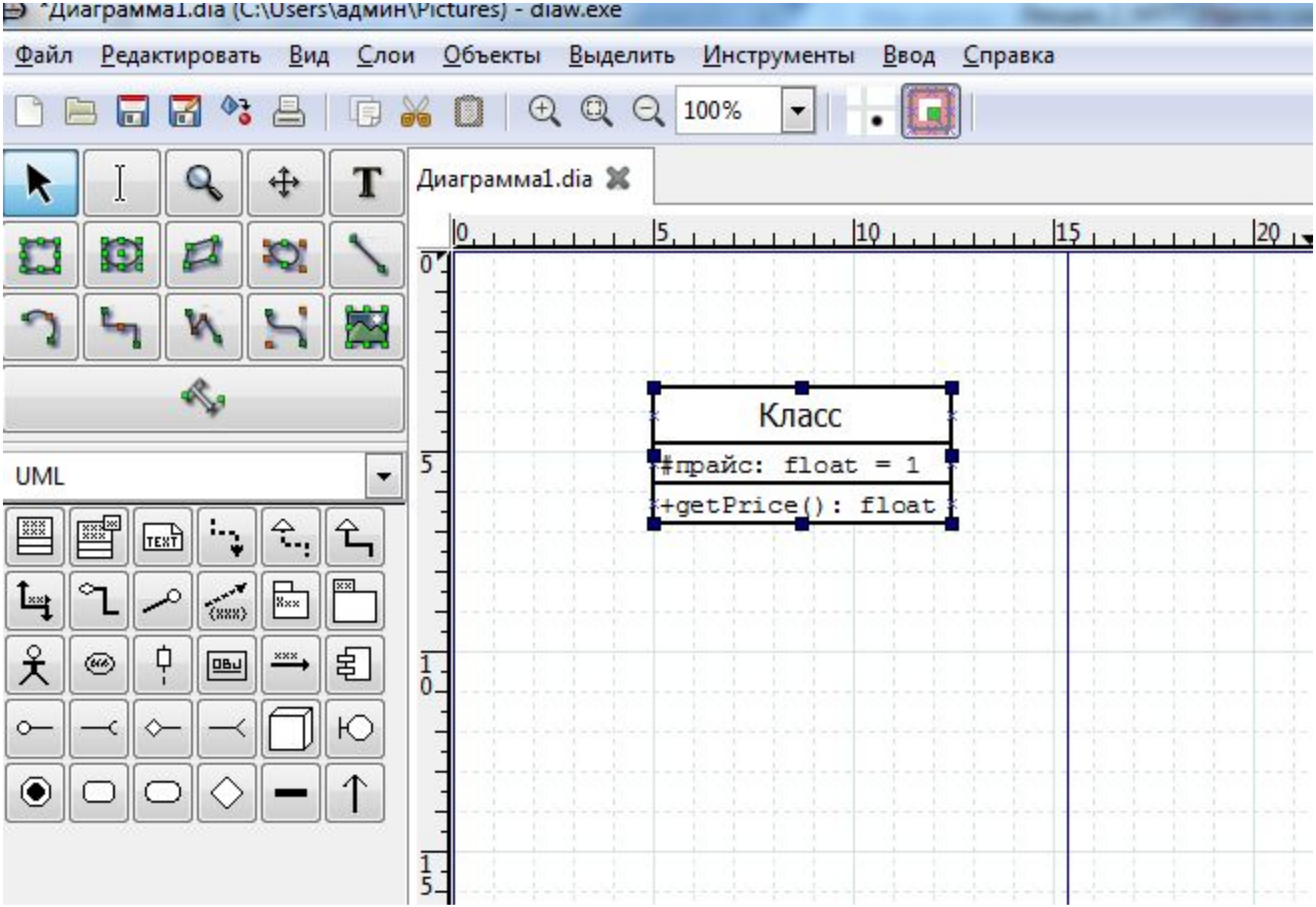



Рис.5.1. Изображение класса в нотации UML

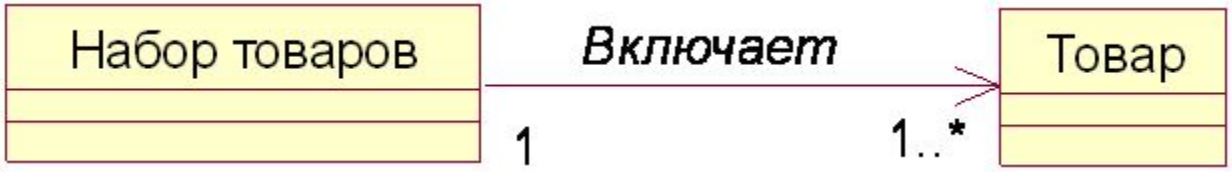



Рис.5.2. Применение ассоциаций


Рис.5.3. Наследуются атрибуты и операции

# Пример построения UML диаграммы классов