

# Диаграмма компонентов

# UML-диаграммы

# Диаграмма компонентов

**Диаграмма компонентов** (*Component Diagram*) – это диаграмма, которая служит для представления программных компонентов и зависимостей между ними.

**Диаграмма компонентов** разрабатывается для следующих целей:

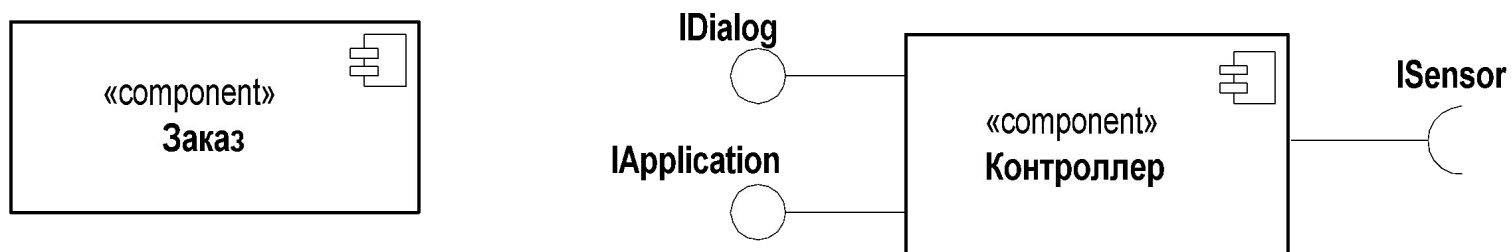
- Визуализация общей структуры ПО.
- Спецификация исполняемого кода (единиц развертывания) ПО.
- Обеспечение многократного использования отдельных фрагментов кода ПО.
- Представление концептуальной и физической схем баз данных.

Имя компонента записывается аналогично имени линии жизни на диаграмме последовательности в следующем формате (БНФ):

*<имя-компонента> ::= [*собственное-имя-компонента*] [ ':' *имя-типа* ],*

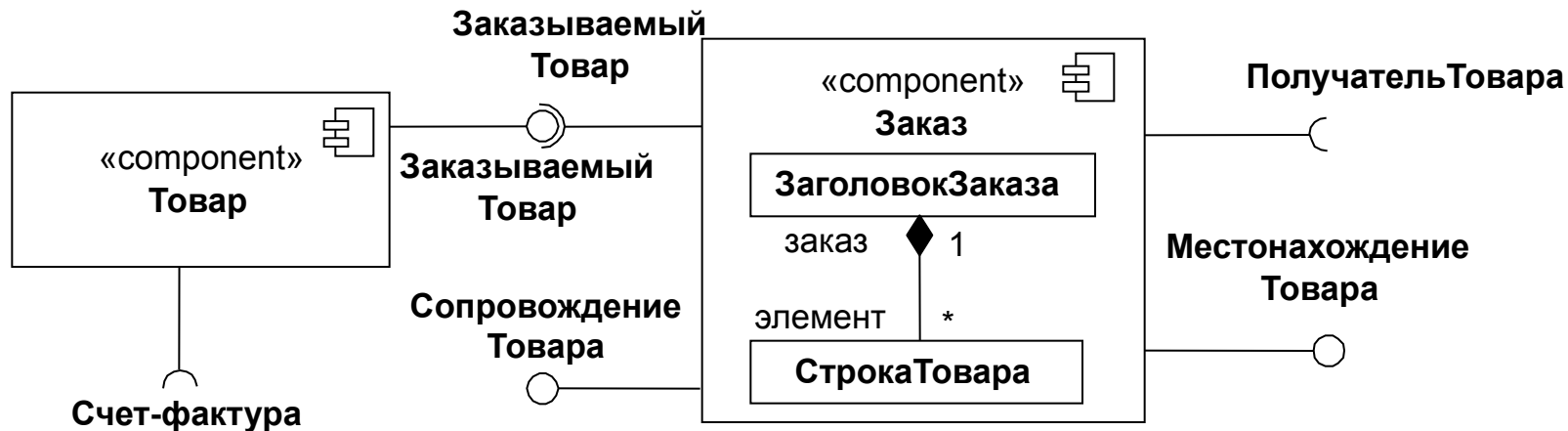
при этом собственное имя компонента записывается со строчной буквы и в имени компонента должен присутствовать хотя бы один терм.

Пример изображения простого компонента и компонента с интерфейсами:

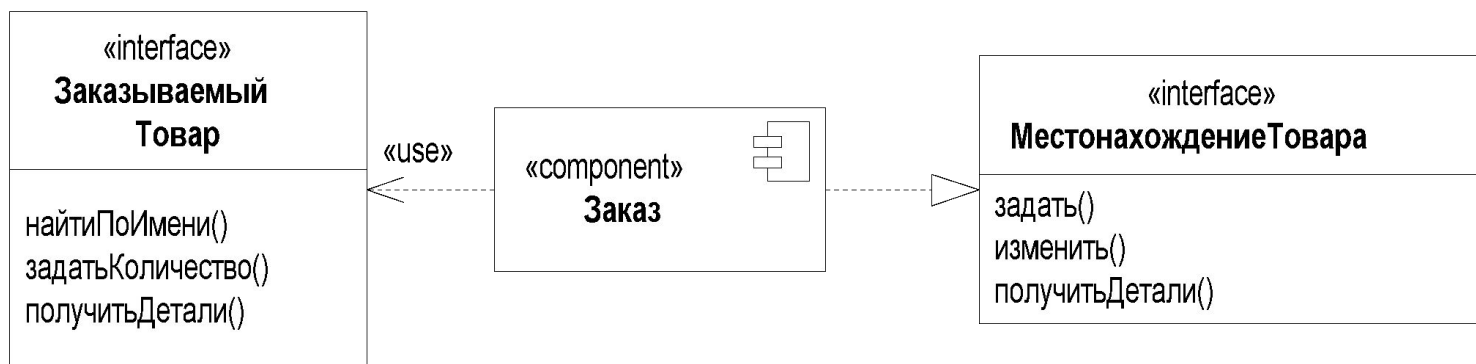


# Диаграмма компонентов

Пример представления компонентов и интерфейсов:

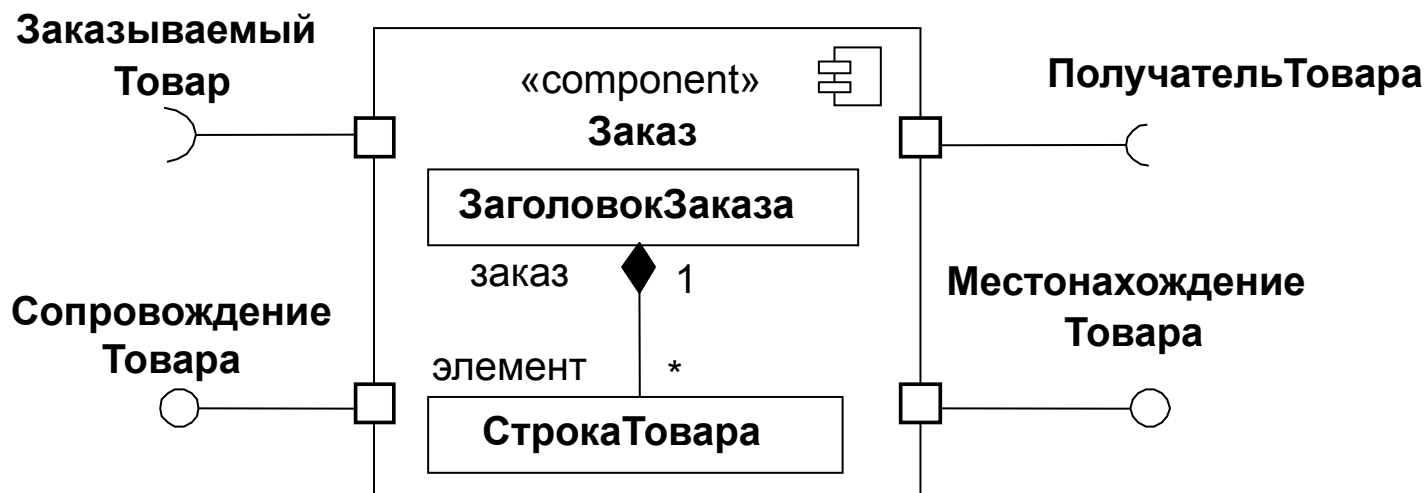


Представление интерфейсов в форме классификаторов с отношениями зависимости и реализации:



# Диаграмма компонентов

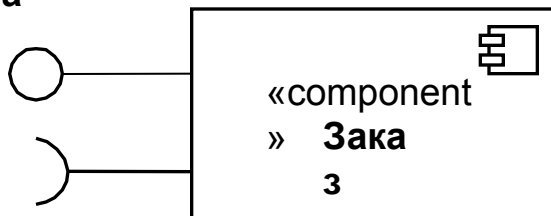
**Порт** определяет различимую точку взаимодействия между компонентом и окружающей средой или между компонентом и его внутренними частями. Наличие имени у порта не является обязательным. При отсутствии имени порта его тип ассоциируется с типом интерфейса, с которым связан порт.



# Диаграмма компонентов

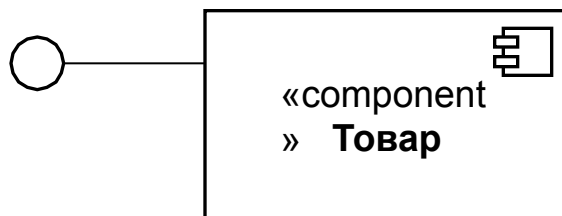
**Собирающий соединитель** связывает два компонента в контексте требуемых и предоставляемых интерфейсов:

Сопровождение  
Товара

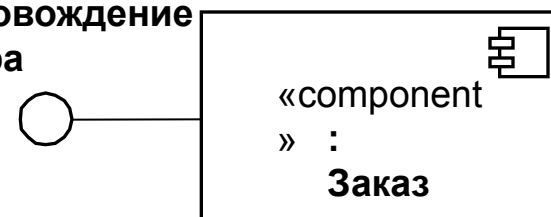


ЗаказываемыйТовар

ЗаказываемыйТовар

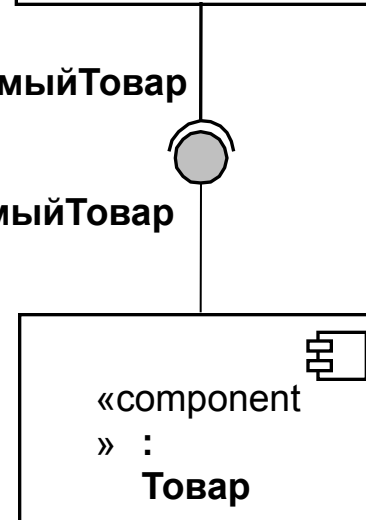


Сопровождение  
Товара



ЗаказываемыйТовар

ЗаказываемыйТовар



# Диаграмма компонентов

Пример диаграммы компонентов с собирающими соединителями для одинаковых интерфейсов:

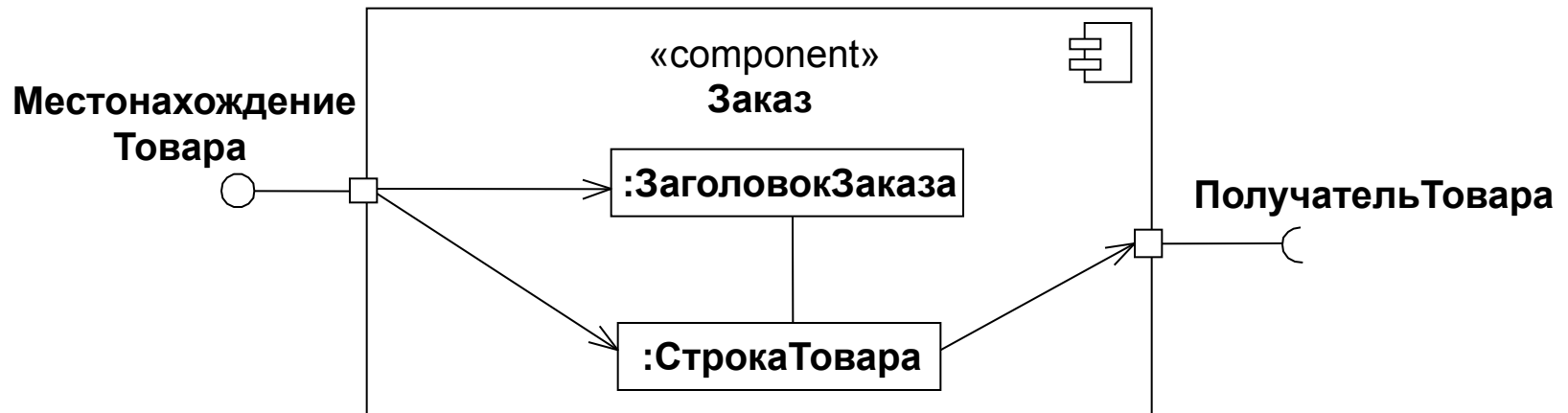


# Диаграмма компонентов

**Делегирующий соединитель** связывает контракт компонента (интерфейсы) с внутренними частями этого компонента.

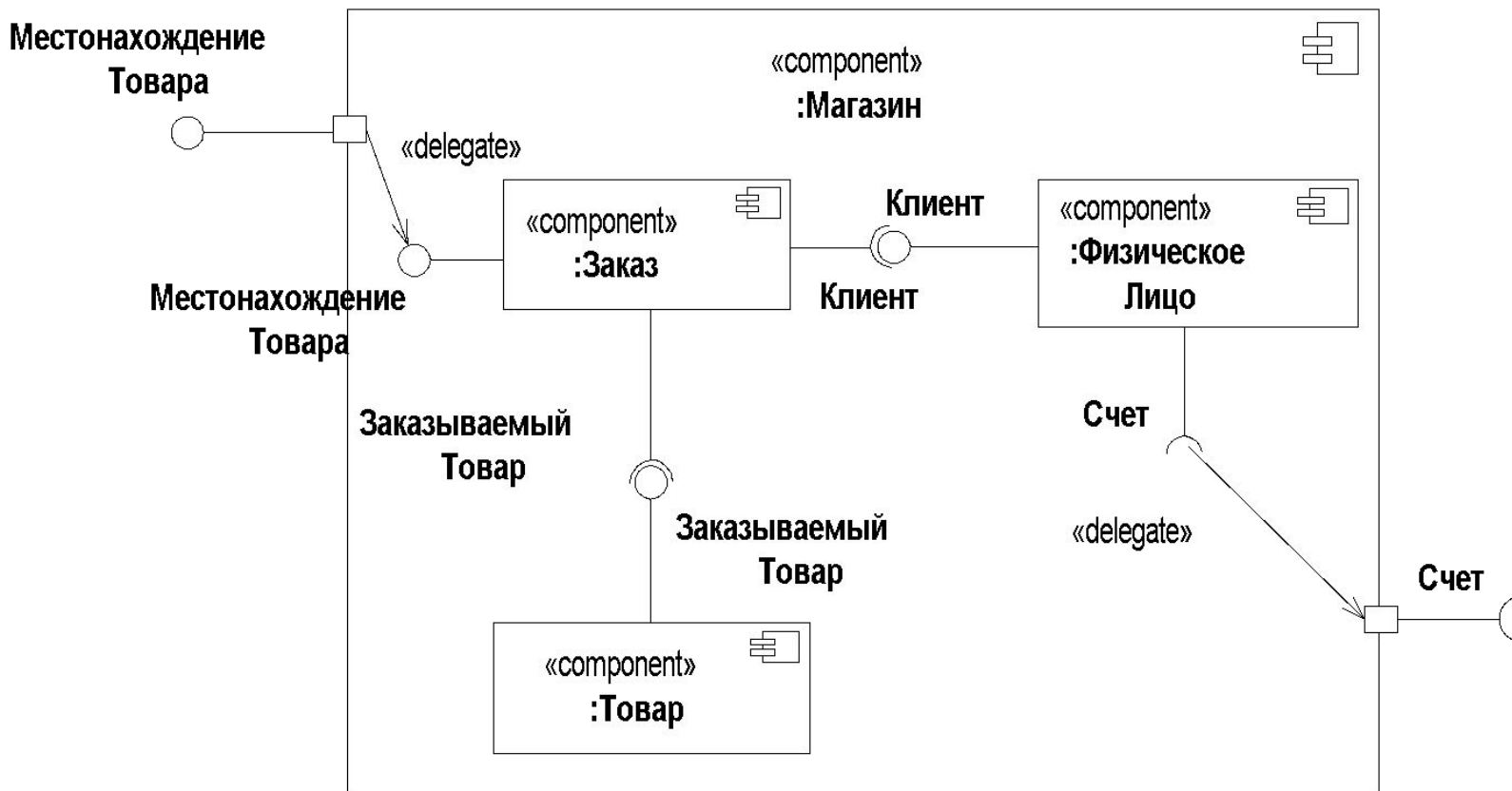
Делегирующий соединитель выполняет одну из следующих задач:

- Передача сообщений или сигналов, поступающих в порт компонента извне, для обработки в некоторую внутреннюю часть компонента или другой порт.
- Передача сообщений или сигналов, поступающих из некоторой внутренней части компонента, для обработки во внешний порт компонента.



# Диаграмма компонентов

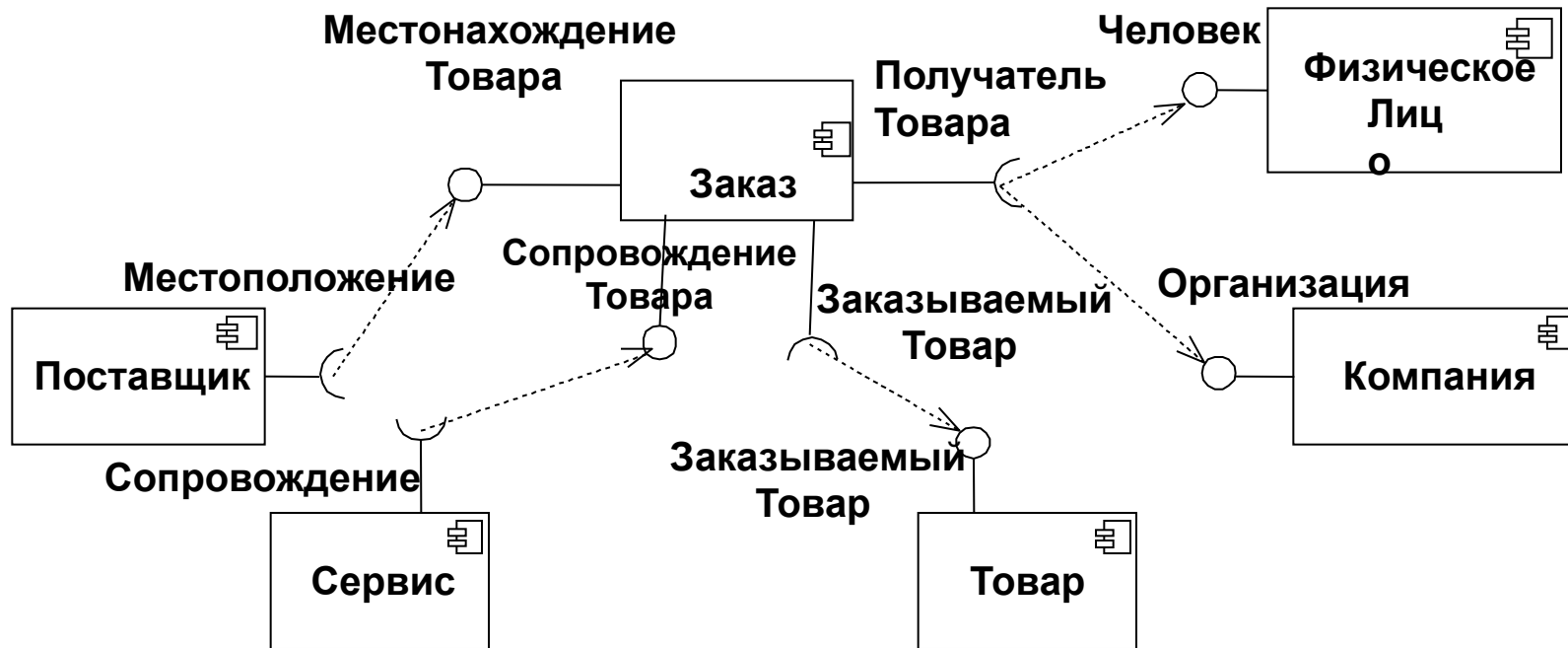
Пример представления внутренней структуры компонента:





# Диаграмма компонентов

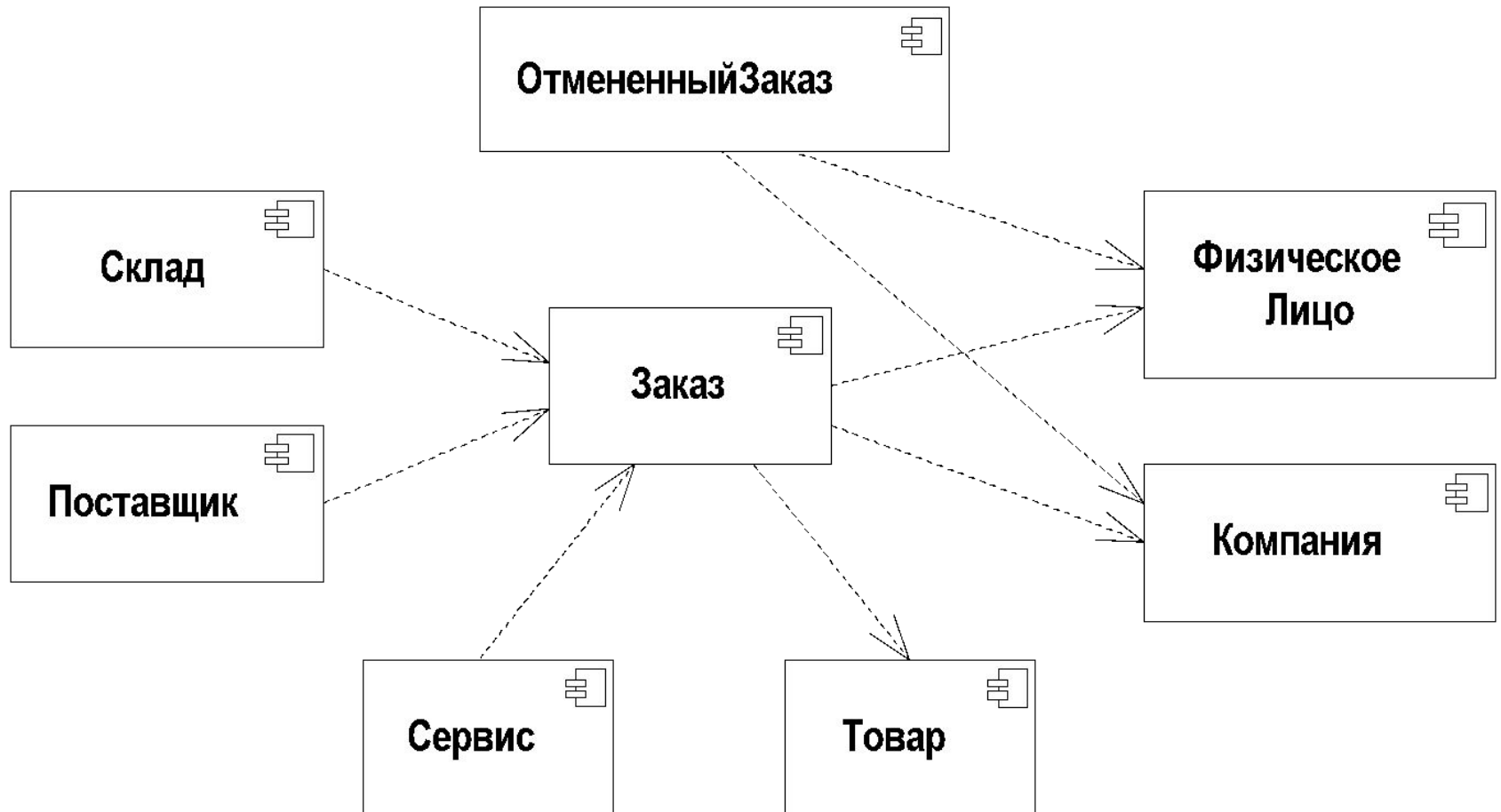
Пример отношений зависимости между компонентами с интерфейсами:



Обратите внимание: стрелка «отношения зависимости» направлена от «требуемого интерфейса» к «предоставляемому интерфейсу»

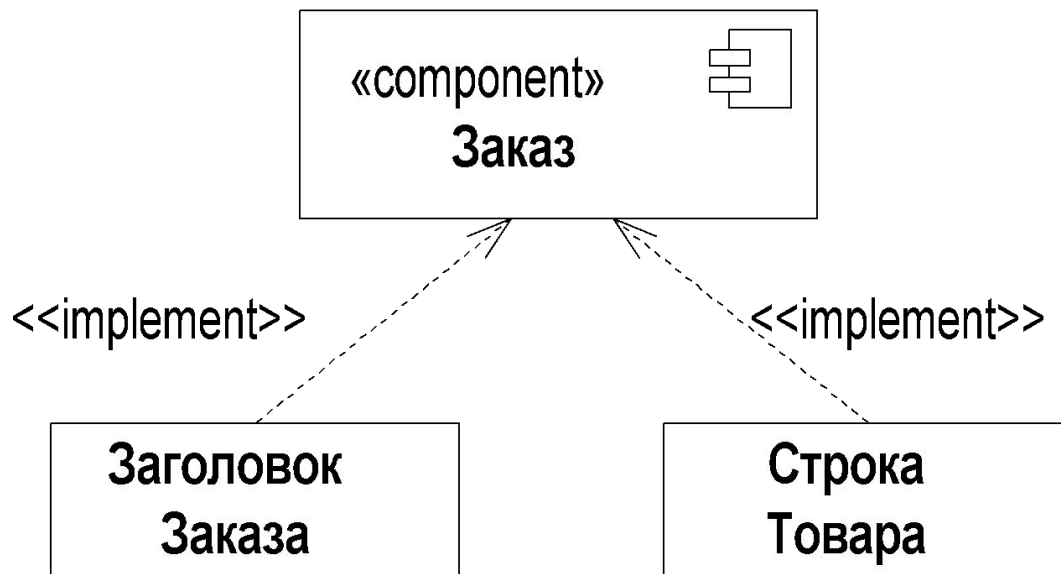
# Диаграмма компонентов

Пример отношений зависимости между компонентами:



# Диаграмма компонентов

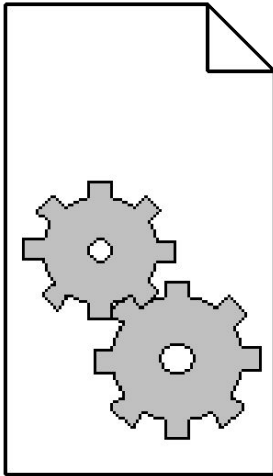
**Отношение реализации** есть отношение зависимости для связи компонентов с теми классификаторами, которые реализуют функциональность данного компонента. Реализация компонента может быть дополнительно помечена стереотипом *«implement»*.



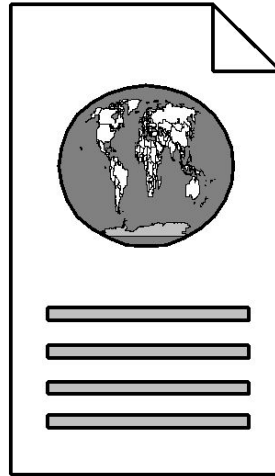
# Диаграмма компонентов

*Примеры нотаций для компонентов специального вида*

Dialog.dll



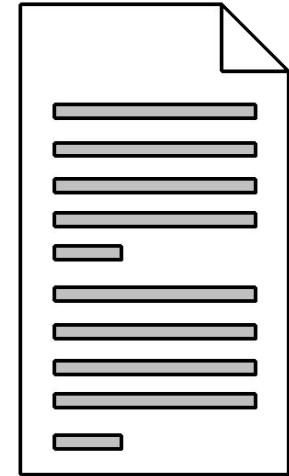
Index.html



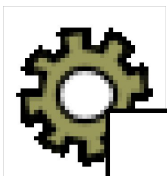
Context .hlp



Main.cpp



# Диаграмма компонентов



<<server page>>



<<client page>>

Компонент **Серверная страница** представляет *Web-страницу* на стороне сервера, содержащую выполняемые сервером сценарии.

Эти сценарии могут взаимодействовать с серверными ресурсами, такими как базы данных, бизнес-логика и внешние системы.

Операции классов, которые реализуют данный компонент – это функции сценария, а их атрибуты – переменные, видимые в пределах этой страницы.

Компонент **Клиентская страница** представляет *Web-страницу* в формате *HTML*, а также данные, элементы интерфейса и даже бизнес-логику.

Клиентские страницы отображаются браузерами и могут содержать сценарии, которые интерпретируются браузером.

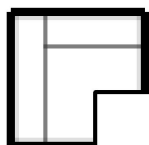
Операции клиентской страницы могут соответствовать функциям, содержащимся в дескрипторах сценария страницы.

Атрибутам клиентской страницы соответствуют объявленные в дескрипторах сценария переменные, которые доступны любой функции в пределах данной страницы.

# Диаграмма компонентов



<<form>>



<<frame set>>

Компонент **Форма** есть набор полей ввода и представляет собой часть клиентской страницы. Форма преобразуется непосредственно в дескриптор HTML `<form>`.

Атрибуты формы могут представлять поля ввода, текстовые поля, переключатели, флажки, скрытые поля формы HTML.

С формой не связано никаких операций, поскольку их нельзя в ней инкапсулировать.

Любые операции взаимодействия с формой являются свойствами содержащей ее страницы.

Компонент **Набор фреймов** - это контейнер, состоящий из нескольких *Web-страниц*.

Любую прямоугольную область web-страницы можно разделить на несколько фреймов.

Каждый фрейм может быть связан с компонентом «**Цель**», однако это необязательно.

Содержимым фрейма может быть *Web-страница* или другой фрейм. Набор фреймов преобразуется непосредственно в набор фреймов *Web-страницы* и дескриптор HTML `<frame>`.

# Диаграмма компонентов

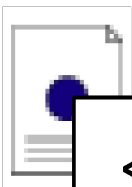


`<<target>>`

Компонент **Цель** представляет собой именованную область окна браузера, в которой могут отображаться *Web-страницы*.

Имя цели соответствует имени целевого объекта. Обычно целью является один из фреймов набора. Однако целью может быть и новое окно браузера. Для цели может быть задано место назначения, где будет отображена новая *Web-страница*.

Имя цели должно быть уникальным для каждого клиента системы.



`<<web-page>>`

Компонент **Web-страница** представляет такую *Web-страницу*, которую браузер может запрашивать по её имени. Этот компонент при необходимости может содержать клиентские или серверные сценарии. Обычно *web-страницы* являются текстовыми файлами, доступ к которым можно получить через *Web-сервер*. Однако они могут быть также компилируемыми модулями, загружаемыми и запускаемыми *Web-сервером*. В любом случае, при доступе к такой странице, она генерирует документ в формате *HTML*, который отправляется в ответ на запрос браузера.

# Диаграмма компонентов



```
<<java server  
page>>
```

Компонент **JSP (Java Server Page)** представляет *Web-страницы*, реализующие код *JSP* серверной части приложения. Этот стереотип применим лишь к приложениям, в которых используется технология *Java Server Pages*.

Этот компонент представляет сервлет Java. Стереотип применим лишь к приложениям, поддерживающим сервлеты компании Sun.



```
<<servlet>>
```

Компонент **Servlet** представляет сервлет *Java*. Стереотип применим лишь к приложениям, поддерживающим сервлеты компании *Sun*.