Лабораторная работа № 11

Тема: *разработка диаграмм компонентов.*
Цель: *изучить принципы построения диаграмм компонентов языка UML, освоить построение диаграмм компонентов в CASE-средстве Enterprise Architect 8.*

Краткая теория

Диаграмма компонентов разрабатывается для следующих целей:
- визуализация общей структуры исходного кода программной системы;
- спецификация исполнимого варианта программной системы;
- обеспечение многократного использования отдельных фрагментов программного кода;
- представление концептуальной и физической схем баз данных.

Компонент – элемент модели, представляющий некоторую модульную часть системы с инкапсулированным содержимым, спецификация которого является взаимозаменяемой в его окружении. Изображение компонента приведено на рисунке 1.

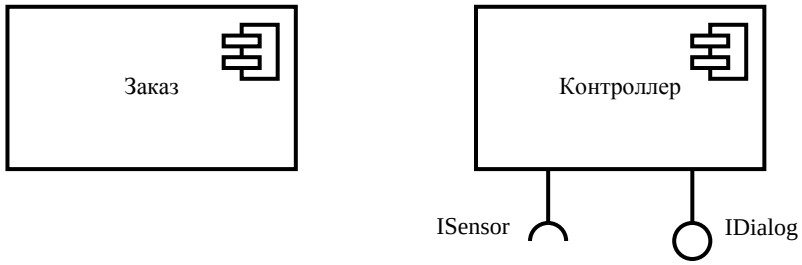

Рисунок 1 – Графическое изображение компонента
а) – простой компонент, б) – компонент с требуемым и предоставляемым интерфейсом

В некоторых CASE-средствах допускается альтернативное обозначение компонента в виде класса со стереотипом «component».

Компонент отображается в двух видах:
- в виде черного ящика – в этом представлении внутренняя структура компонента полностью скрыта от его окружения, а открытыми и общедоступными извне являются только интерфейсы компонента, которые могут быть указаны в форме свойств компонента.
- в виде белого ящика – это представление показывает, как специфицированное и видимое извне поведение компонента реализуется внутри его.

Компонент может рассматриваться либо в качестве типа, либо в качестве экземпляра. Компонент в качестве типа наделяется возможностью иметь атрибуты и операции, а также участвовать в ассоциациях и обобщениях. Компонент в качестве экземпляра может быть инстанцирован от соответствующего компонента в качестве типа. На практике экземпляром компонента может быть некоторый объект времени выполнения или некоторый артефакт, который определяется только во время проектирования.

Интерфейс – вид класса, который представляет собой объявление множества общедоступных характеристик и обязанностей.

Предоставляемый интерфейс – интерфейс, который компонент предлагает для своего окружения.

Требуемый интерфейс – интерфейс, который необходим компоненту от своего окружения для выполнения заявленной функциональности, контракта или поведения.

Предоставляемые интерфейсы иногда называют реализуемыми, реже – обеспеченными интерфейсами.

Пример изображение требуемых и предоставляемых интерфейсов приведен на рисунке 2.

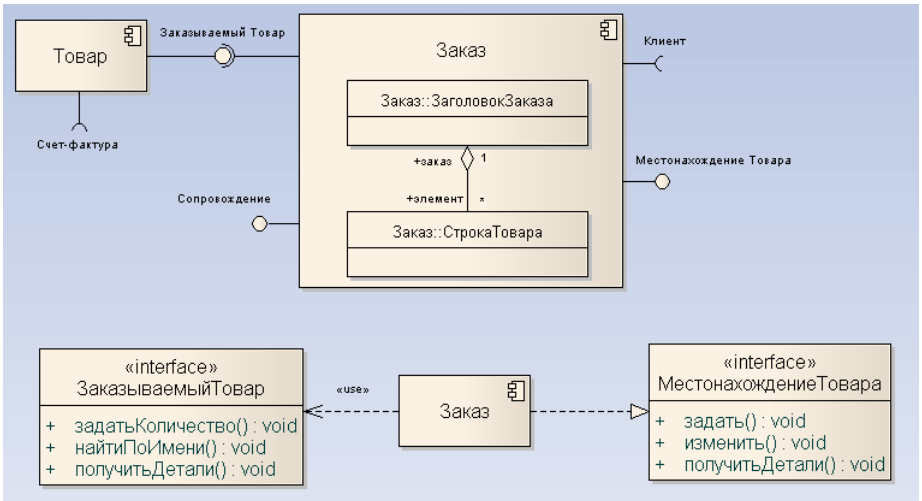

Рисунок 2 – Пример использования интерфейсов на диаграммах компонентов

Порт – свойство классификатора, которое специфицирует отдельную точку взаимодействия между этим классификатором и его окружением или между классификаторами и его внутренними частями. Изображение компонента с портами приведено на рисунке 3.

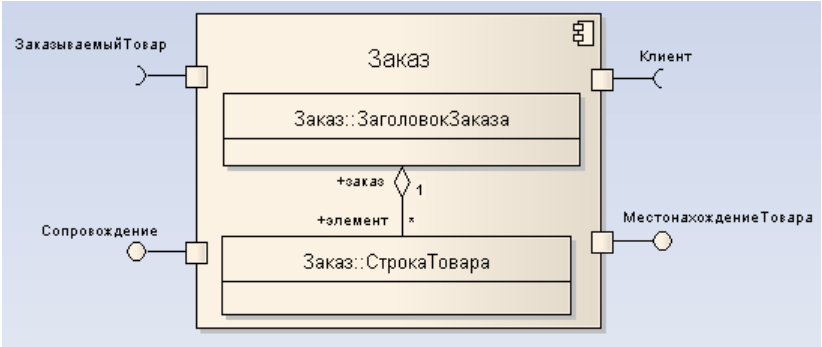

Рисунок 3 – Пример использования портов на диаграммах компонентов

Собирающий соединитель – соединитель, который связывает два компонента в контексте предоставляемых и требуемых сервисов. Пример собирающего соединителя приведен на рисунке 4 справа.

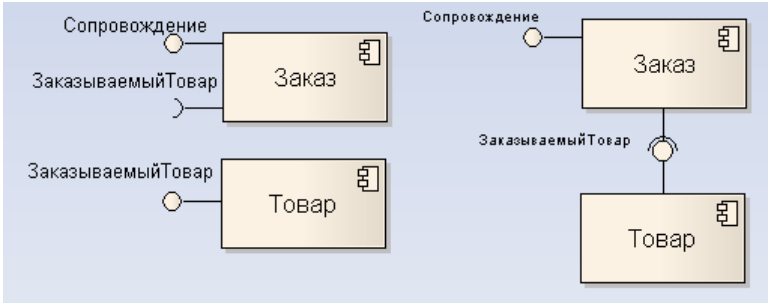

Рисунок 4 – Пример собирающего соединителя

Делегирующий соединитель – соединитель, который связывает внешний контракт компонента с реализацией этого поведения внутренними частями этого компонента (рисунок 5). В общем случае 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.

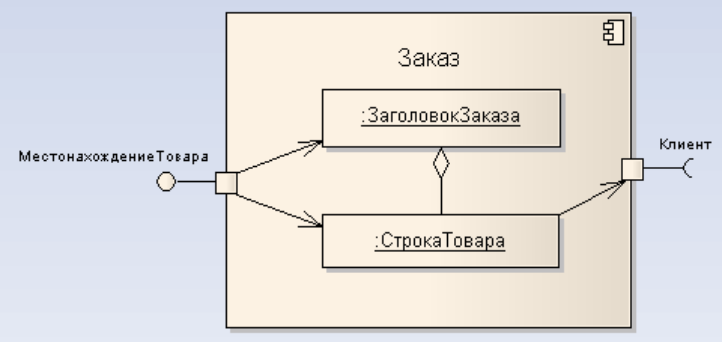

Рисунок 5 – Пример использования делегирующего соединителя

На рисунке 6 приведен пример использования делегирующего соединителя для связи внешних портов компонента с предоставляемыми и требуемыми интерфейсами его внутренних компонентов.

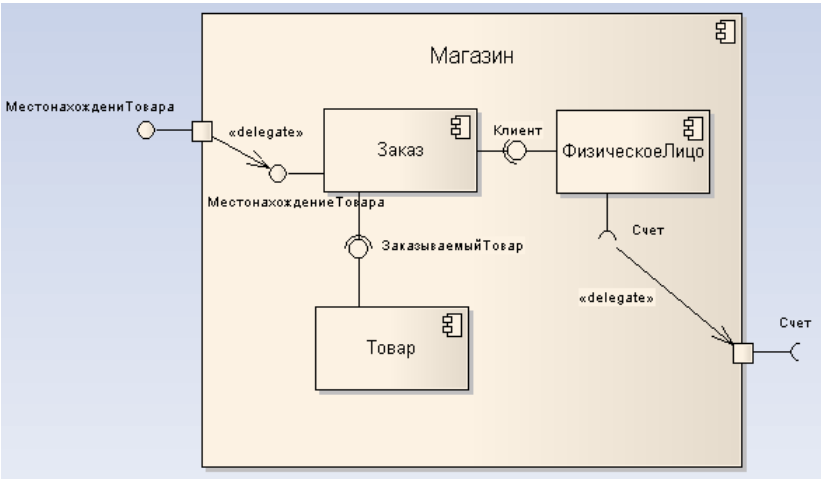

Рисунок 6 – Пример использования делегирующего соединителя для связи с внутренними компонентами

При использовании делегирующего соединителя необходимо учитывать:

- Делегирующий соединитель может быть определен только между интерфейсами или портами одного и того же вида.
- Если делегирующий соединитель определяется между требуемым интерфейсом (портом) и некоторой внутренней частью, то эта внутренняя часть должна иметь отношение реализации с интерфейсом (портом) этого типа.
- Если делегирующий соединитель определяется между интерфейсом (портом) в качестве источника и интерфейсом (портом) в качестве цели, то интерфейс в качестве цели должен поддерживать сигнатуру, совместимую с подмножеством операций интерфейса (порта) источника.

Также на диаграммах компонентов может использоваться отношение зависимости, которое обозначает зависимость одного компонента от другого. Это отношение может быть указано как между двумя компонентами (рисунок 7 слева), так и между соответствующими требуемыми и предоставляемыми интерфейсами этих компонент вместо собирающего соединителя (рисунок 7 справа).

Реализация – специализация отношения зависимости для связи компонентов с классификаторами, которые реализуют функциональность этого компонента. Пример использования реализации приведен на рисунке 8.

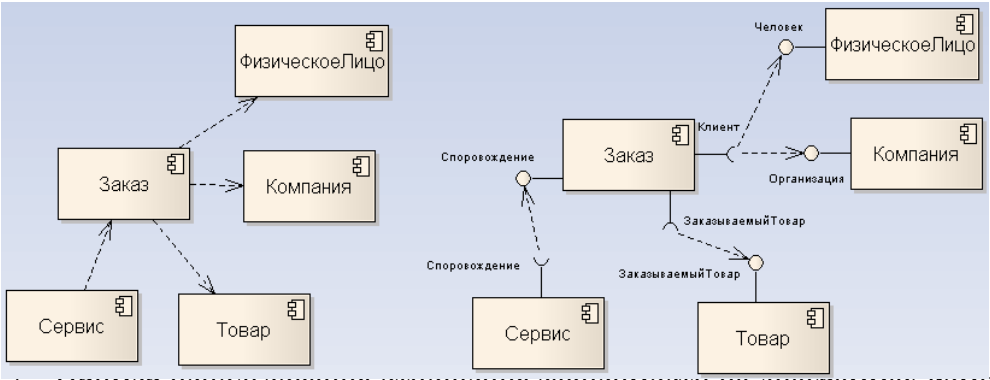

Рисунок 7 – Пример использования отношения зависимости на диаграммах компонентов

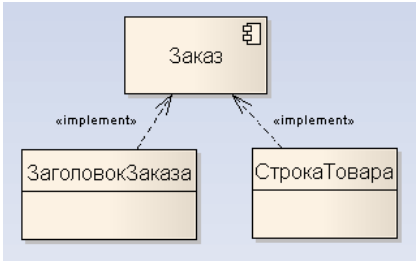

Рисунок 8 – Пример использования отношения реализации на диаграмме компонентов

Для указания некоторой специфики компонентов могут использоваться стереотипы. В языке UML заложен набор готовых стереотипов, перечень которых приведен в таблице.

Таблица – Перечень стандартных стереотипов компонентов

Ключевое слово	Назначение
file	Определяет наиболее общую разновидность компонента, который представляется в виде произвольного физического файла
executable	Определяет разновидность компонента-файла, который является исполнимым файлом и может выполняться на некоторой компьютерной платформе
document	Определяет разновидность компонента-файла, который представляется в форме документа произвольного содержания, не являющегося исполнимым файлом или файлом с исходным текстом программы.
library	Определяет разновидность компонента-файла, который представляется в форме статической или динамической библиотеки.
source	Определяет разновидность компонента-файла, представляющего собой файл с исходным текстом программы, который после компиляции может быть преобразован в исполнимый файл.
table	Определяет разновидность компонента, который представляется в форме таблицы базы данных.

Ход работы

В данной лабораторной работе необходимо построить диаграмму компонентов, которая бы моделировала структуру программного кода разрабатываемого приложения. На диаграмме должны располагаться компоненты, моделирующие отдельные файлы исходного кода, с установленными для них отношениями зависимости. Для выявления компонентов используются классы из диаграмм классов, созданных ранее. Следует руководствоваться следующими вариантами отображений классов в компоненты:

- один класс отображается в один компонент (компонент следует моделировать в виде «черного ящика»),
- несколько классов отображаются в один компонент (компонент следует моделировать в виде «белого ящика»).