Практическая работа № 2
Создание диаграммы классов.

Цель работы: получить навыки построения диаграмм классов, создания пакетов и группировки классов в пакеты.

Задание:

создать диаграмму классов * для одного из сценарией диаграммы прецедентов, созданной в предыдущей лабораторной работе. Для каждого класса необходимо задать атрибуты и операции. Каждый класс должен быть подробно задокументирован - необходимо задать текстовое описание самого класса, описания его атрибутов и операций;

** Примечание: рассматривать диаграмму классов рекомендуется с концептуальной точки зрения, которая используется на начальных этапах моделирования и разработки.*

Содержание отчета:

1. созданная диаграмма классов;
2. краткое описание каждого созданного класса и отношений между классами.

Краткие теоретические сведения

Диаграммы классов (class diagram) используются для моделирования статического вида системы с точки зрения проектирования. *Диаграмма классов* - диаграмма, на которой показано множество классов, интерфейсов, кооперацией и отношений между ними. Используется в следующих целях:

- для *моделирования словаря* системы: предполагает принятие решения о том, какие абстракции являются частью системы, а какие - нет. С помощью диаграмм классов можно определить эти абстракции и их обязанности;
- для *моделирования простых кооперацией*. Кооперация - это сообщество классов, интерфейсов и других элементов, работающих совместно для обеспечения некоторого кооперативного поведения;
- для *моделирования логической схемы базы данных*.

Согласно Мартину Фаулеру существуют три различные точки зрения на построение диаграмм классов или любой другой модели:

- *концептуальная точка зрения* - диаграммы классов служат для представления понятий изучаемой предметной области. Эти понятия будут соответствовать реализующим их классам, но прямое соответствие может отсутствовать. Концептуальная модель может иметь слабое отношение или вообще не иметь никакого отношения к реализующему ее программному обеспечению, поэтому ее можно рассматривать без привязки к какому-то языку программирования;
- *точка зрения спецификации* - рассматривается программная система, при этом рассматривается только ее интерфейсы, но не реализация;
- *точка зрения реализации* - классы диаграммы соответствуют реальным классам программной системы.

Пример выполнения работы.

1. Создание диаграммы классов для сценария "Добавить новый заказ" прецедента "Работа с заказом"

Диаграммы классов будем рассматривать с концептуальной точки зрения. Для упрощения задачи и чтобы не загромождать диаграммы несущественными деталями методы setX, getX для каждого атрибута X классов задавать не будем.

Создадим в Логическом представлении браузера новую диаграмму классов и назовем ее "*Add New Order*". В поле документации запишем для нее следующий текст: "*Диаграмма классов для сценария "Добавить новый заказ" прецедента "Работа с заказом"*".

Заполнение диаграммы начнем с определения классов-сущностей. Рассматриваемый сценарий состоит из:

- самого *заказа*;
- *клиента*, который делает заказ;
- *комплектующих изделий*, которые входят в заказ.

Создадим классы-сущности *Order (Заказ)*, *Client (Клиент)* и *ComponentPart (Комплектующее изделие)*. Поскольку в один заказ может входить много разных комплектующих изделий, и одно комплектующее изделие может входить во много заказов, то введем еще один класс-сущность *OrderItem (Состав заказа)*. Опишем каждый класс.

Класс *Client:*

Параметр	**Значение**
Комментарий	Класс, представляющий собой клиента фирмы
Атрибуты	name : String - наименование клиента address : String - адрес клиента phone : String - телефон клиента Все атрибуты имеют модификатор доступа - private
Операции	AddClient() - добавление нового клиента RemoveClient() - удаление существующего клиента GetInfo() - получить информацию о клиенте Все операции имеют модификатор доступа - public

Класс *Order:*

Параметр	**Значение**
Комментарий	Класс, представляющий собой заказ, который делает клиент
Атрибуты	orderNumber : Integer - номер заказа orderDate : Date - дата оформления заказа orderComplete : Date - дата выполнения заказа Все атрибуты имеют модификатор доступа - private
Операции	Create() - создание нового заказа SetInfo() - занести информацию о заказе GetInfo() - получить информацию о заказе Все операции имеют модификатор доступа - public

Класс *OrderItem:*

Параметр	**Значение**
Комментарий	Класс, представляющий собой пункт заказа, который делает клиент
Атрибуты	itemNumber : Integer - номер пункта заказа quantity : Integer - количество комплектующих изделий price : Double - цена за единицу Все атрибуты имеют модификатор доступа - private
Операции	Create() - создание новой строки заказа SetInfo() - занести информацию о строке заказа GetInfo() - получить информацию о строке заказа Все операции имеют модификатор доступа - public

Класс *ComponentPart:*

Параметр	**Значение**
Комментарий	Класс, представляющий собой комплектующие изделия
Атрибуты	name : String - наименование manufacturer : String - производитель price : Double - цена за единицу description - описание Все атрибуты имеют модификатор доступа - private
Операции	AddComponent() - добавление нового комплектующего изделия RemoveComponent() - удаление комплектующего изделия GetInfo() - получить информацию о комплектующем изделии Все операции имеют модификатор доступа - public

Результат создания классов-сущностей показан на рис. 1:

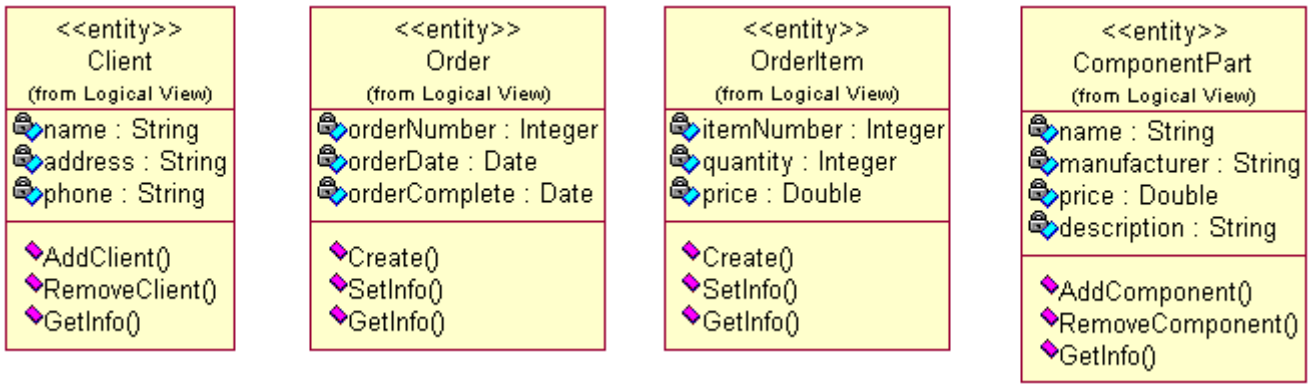

Рисунок 1. Созданные классы-сущности

Добавим отношения между классами (рис. 2):

- класс *Client* и *Order* - отношение ассоциации, поскольку данные два класса просто связаны друг с другом и никакие другие типы связей здесь применить нельзя. Один клиент может сделать несколько заказов, каждый заказ поступает только от одного клиента, поэтому кратность связи со стороны класса *Client* - 1, со стороны *Order* - 1..n;
- класс *Order* и *OrderItem* - отношение композиции, поскольку строка заказа является частью заказа, и без него существовать не может. В один заказ может входить несколько строк заказа, строка заказа относится только к одному заказу, поэтому кратность связи со стороны *Order* - 1, со стороны *OrderItem* - 1..n;
- класс *OrderItem* и *ComponentPart* - отношение агрегации, поскольку комплектующие изделия являются частями строки заказа, но и те, и другие, являются самостоятельными классами. Одно комплектующее изделие может входить во много строк заказа, в одну строку заказа входит только одно комплектующее изделие, поэтому кратность связи со стороны *ComponentPart* - 1, со стороны *OrderItem* - 1..n.

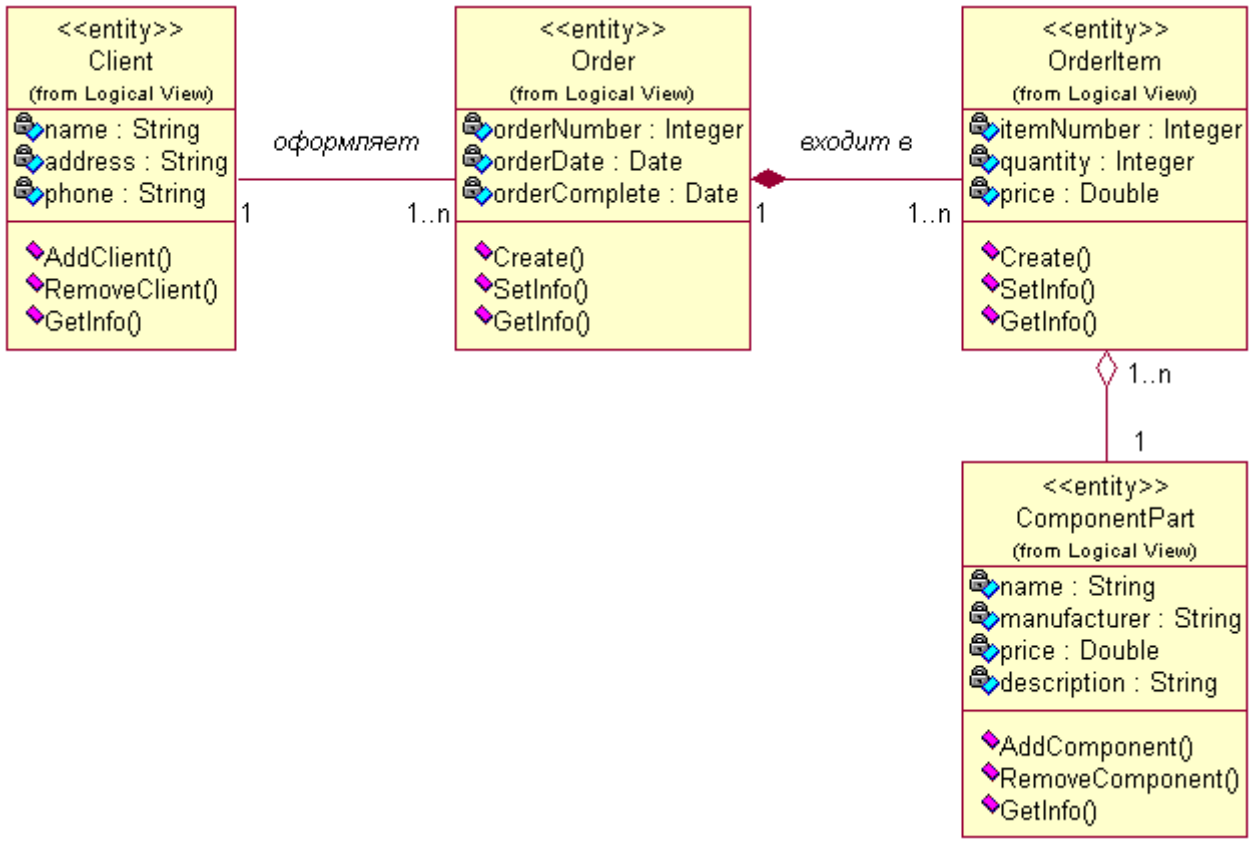

Рисунок 2. Классы-сущности и отношения между ними

Добавим теперь на диаграмму граничные и управляющие классы (рис. 3). Рассматриваемый сценарий - это только одно из действий, которые обеспечивает прецедент "Работа с заказом". Прецедент также позволяет просмотреть, отредактировать или удалить заказ. Это означает, что необходимо предусмотреть механизм, который позволяет выбирать необходимое действие. Создадим для этого граничный класс *OrderOptions* (*Параметры работы с заказом*) с комментарием "*Класс, обеспечивающий механизм работы с заказами*". Также создадим граничный класс *AddNewOrder*

(Добавление нового заказа), который будет служить для добавления новых заказов (комментарий - "*Класс служит для добавления новых заказов*". Отношение между этими классами - агрегация, поскольку в данном случае класс *AddNewOrder* рассматривается как часть класса *OrderOptions*, частями которого также будут классы для просмотра, редактирования и удаления заказов. Кратность связи 1 к 1, поскольку в состав класса *OrderOptions* входит только один класс *AddNewOrder*.

Перейдем теперь к управляющим классам. Добавим управляющий класс *OrderManager* (*Менеджер по работе с заказами*) с комментарием "*Управляющий класс для обработки потока событий прецедента "Работа с заказами"*", который будет обеспечивать обработку потока событий для рассматриваемого прецедента. Данный класс будет связан с классами *AddNewOrder* и *Order*. Отношение между классами *AddNewOrder* и *OrderManager* - однонаправленная ассоциация с кратностью связи 1 к 1, поскольку один экземпляр класса *AddNewOrder* взаимодействует только с одним экземпляром класса *OrderManager*. Отношение между классами *OrderManager* и *Order* - однонаправленная ассоциация с кратностью связи 1 к 1..n, поскольку один класс *OrderManager* может взаимодействовать с несколькими классами *Order*.

Окончательный вариант диаграммы классов показан на рис. 3:

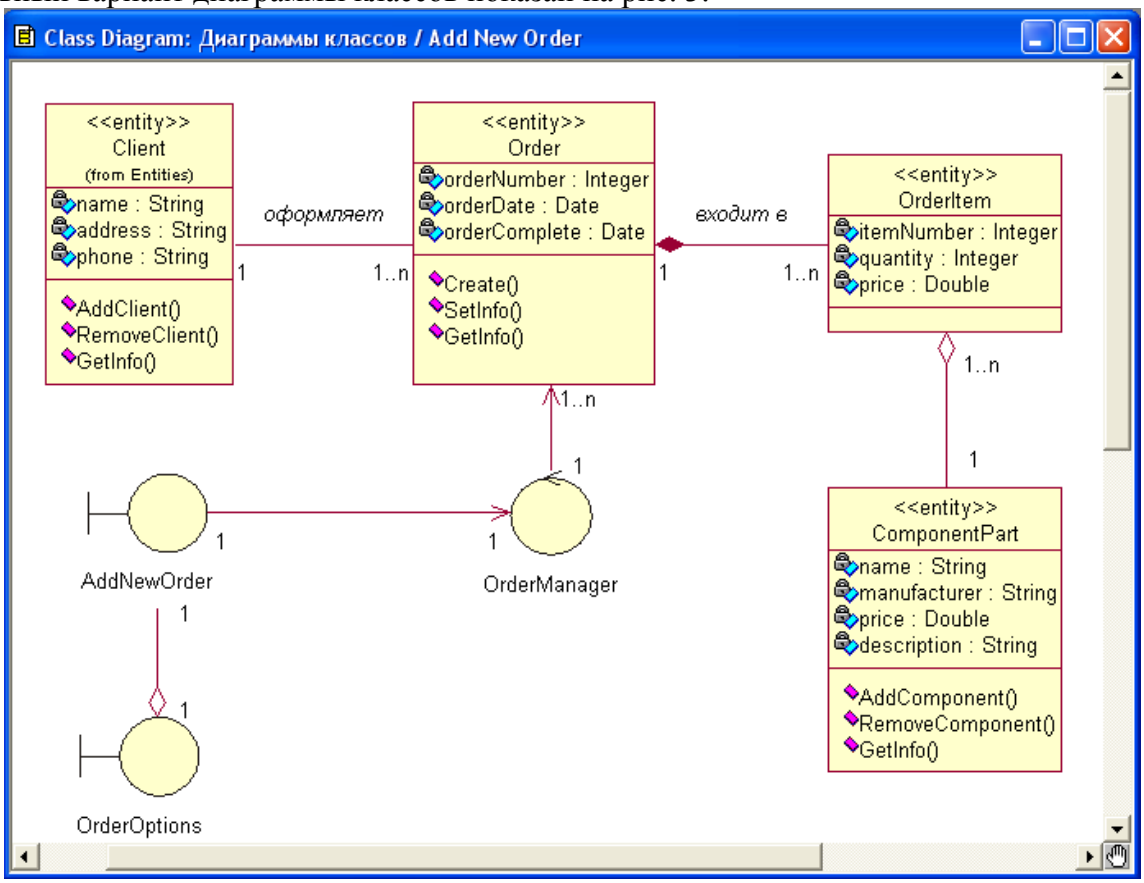

Рисунок 3. Итоговая диаграмма классов