Практическое задание №2

Создание диаграммы классов и диаграмм взаимодействия

Целью задания является изучение объектно-ориентированного анализа и моделирования бизнес-процессов в исследуемой предметной области с помощью языка UML.

В ходе выполнения задания для выбранной (или заданной) предметной области изучается процесс построения диаграммы классов и диаграмм взаимодействия с помощью CASE-средства, поддерживающего язык UML.

Для выполнения данного задания используется модель и диаграммы, построенные в ходе выполнения предыдущего практического задания №1.

Общая последовательность выполнения задания №2 включает определенные этапы (разделы).

1. Построение диаграммы классов (Class Diagram).
2. Диаграмма пакетов (Package Diagram).
3. Построение диаграмм взаимодействия (диаграмма последовательности действий Sequence Diagram и диаграмма кооперации Communication/Collaboration Diagram).

В результате выполнения задания и построения диаграмм следует сохранить модель в отдельном файле, а также подготовить и оформить отчет, содержащий последовательность разработки моделей и их элементов, необходимые комментарии, пояснения и краткий вывод в заключении.

В названии файла модели и отчета по заданию необходимо использовать фамилию студента и номер работы.

Далее приводится демонстрационный пример построения моделей основных бизнес-процессов, описывающих деятельность некоторого небольшого интернет-магазина.

1. Построение диаграммы классов (Class Diagram)

Диаграмма классов является частью логической модели системы и представляет статическую картину системы.

Для каждой системы строится не одна, а несколько диаграмм классов: возможно, что для каждого прецедента или сценария своя. На одних показывают подмножества классов, объединенные в пакеты, и отношения между ними, на других – отображают те же подмножества, но с атрибутами и операциями классов. Для представления системы разрабатывается столько диаграмм классов, сколько потребуется.

1.1. Основные элементы диаграмм классов

Дадим некоторые определения и опишем основные элементы нотации диаграмм классов.

Объект – это некоторая сущность реального мира или концептуальная (абстрактная) сущность. Примерами объектов могут служить жилой дом, сотрудник фирмы, компьютер. Или нечто абстрактное: химическая формула, торговый заказ с номером, банковский счет клиента.

Объект имеет четко определенные границы и значение для системы и характеризуется состоянием, поведением и индивидуальностью.

Состояние объекта – это одно из условий, в котором он может находиться. Состояние обычно изменяется со временем и характеризуется набором свойств, которые называются атрибутами.

Пример. Покупатель определяется его именем, адресом, телефоном, датой рождения.

Поведение определяет, как объект реагирует на запросы других объектов и что может делать сам объект. Поведение характеризуется операциями объекта.

Пример. Покупатель может добавить товар в корзину, просматривать каталог, удалять товар из корзины.

Индивидуальность означает, что каждый объект уникален, даже если его состояние идентично состоянию другого объекта.

Пример. Два клиента уникальны, хотя каждый из них является покупателем магазина и имеет одинаковые поведение и состояния.

Как правило, в системе существует множество объектов, имеющих одинаковое поведение, принимающих одинаковые состояния. Например, сотрудники фирмы, которых может быть несколько десятков, и данные о

которых содержатся в базе данных, имеют одинаковые атрибуты – фамилию, имя, отчество, дату рождения, должность и др. – с разными значениями этих атрибутов, а также могут иметь схожее поведение – подать заявление на отпуск или перевод в другое подразделение. Для группировки объектов используются классы.

Класс – это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой. Каждый класс является шаблоном для создания объекта.

Каждый **объект** – это экземпляр класса. Важно помнить, что каждый объект может быть экземпляром только одного класса.

Пример. Применительно к магазину «Style» мы можем сгруппировать сотрудников магазина, описав общий для них класс Сотрудник. Объект этого класса, например, конкретный сотрудник, может включать в себя следующую информацию: имя, адрес, должность, размер заработной платы, кроме того этот объект может выйти в отпуск.

В нотации UML классы и объекты изображаются в виде прямоугольников (рисунок 1.1). Прямоугольник класса всегда делится на три секции (раздела), имя класса помещается в первую секцию, каждое слово в названии класса принято писать с большой буквы. Во второй и третьей секциях могут указываться атрибуты и операции класса соответственно, эти секции могут быть пустыми. Названия классов выбираются в соответствии с понятиями предметной области. Это должно быть существительное или словосочетание в единственном числе, наиболее точно характеризующее предмет. Класс должен описывать только одну сущность.

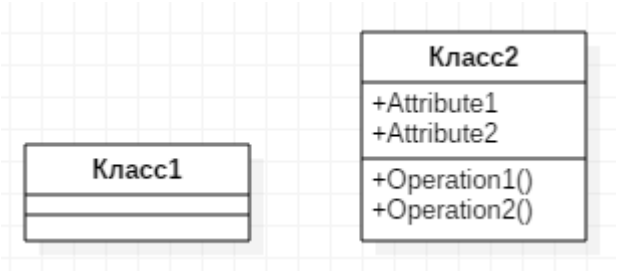

Рисунок 1.1 – Изображение классов

Имя класса может быть простым, как это показано на рисунке 1.1, или составным. **Составное имя класса** состоит из самого имени класса и из имени пакета, которому принадлежит класс, разделенных двоеточием. Имя класса должно быть уникальным внутри пакета.

Составное имя объекта состоит из имени объекта и имени класса, разделенных двоеточием. Объект может быть безымянным (анонимным), если неизвестно его имя. Тогда на диаграмме объект изображается с именем, которое состоит из двоеточия и имени класса, которому принадлежит объект. Если неизвестен класс, экземпляром которого является объект, то изображается только имя объекта. Такой объект называется «сиротой» (рисунок 1.2).

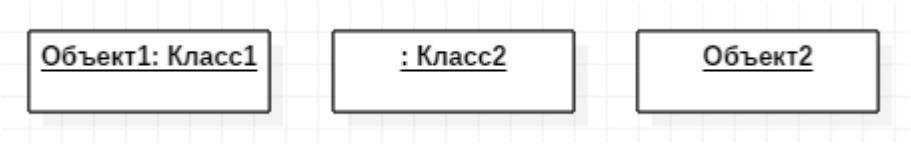

Рисунок 1.2 – Именование объектов (составное имя, безымянный объект и объект-сирота)

Пример. Класс Покупатель и объект этого класса – некоторый покупатель – можно изобразить так, как показано на рисунке 1.3. В данном примере объекту класса Покупатель присвоено имя Покупатель1.

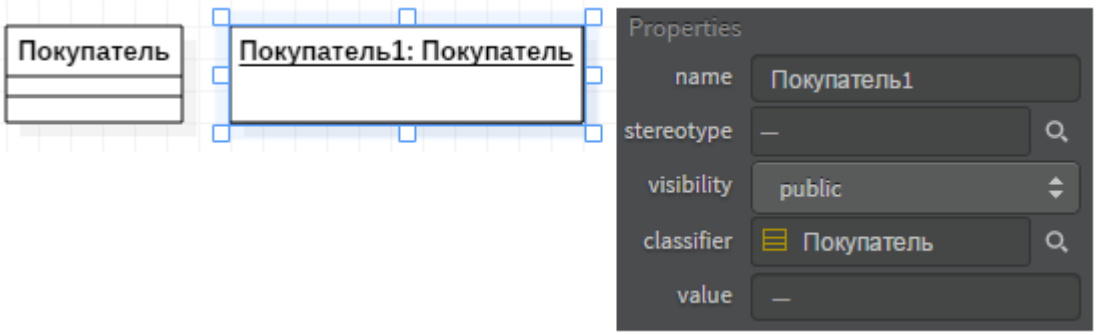

Рисунок 1.3 – Класс и его объект

1.2. Выявление классов

Выявление классов можно начать с изучения потока событий. Имена существительные в описании этого потока дадут понять, что может являться классом. В общем случае существительное может оказаться действующим лицом, классом, атрибутом класса или выражением, не являющимся ни действующим лицом, ни классом, ни атрибутом класса.

Если в ходе проектирования системы уже построены диаграммы взаимодействия, перед тем, как приступать к построению диаграмм классов, следует искать на этих диаграммах похожие объекты. Например, может быть создана диаграмма последовательности, описывающая оформление заказа разными объектами-покупателями. Обратите внимание на эти объекты: они имеют одинаковые свойства: имя, счет в банке и т.п. Значит, в системе должен быть класс с именем Покупатель, который будет шаблоном для разных объектов-покупателей.

Некоторые возможные классы будут выявлены при рассмотрении трех стереотипов: **сущность** (entity), **граница** (boundary) и **управление** (control). Мы уже встречались со стереотипами отношений, когда говорили об отношениях на диаграммах прецедентов. Тот же принцип создания нового типа на основе уже существующего применим и для классов. Стереотип используется для создания нового типа элемента, в данном случае нового типа класса.

Например, Вы хотите выделить все экранные формы в модели. Для этого нужно создать стереотип Form (Форма).

Стереотипы помогают лучше понять ответственности каждого класса в модели, классифицировать выполняемые ими функции. В UML для этого применяют три основных стандартных вида стереотипов классов:

- классы-сущности;
- граничные классы;
- управляющие классы.

Класс-сущность содержит информацию, хранимую постоянно. Используется для моделирования данных и поведения с длинным жизненным циклом. Они могут представлять информацию о предметной области, а могут представлять элементы самой системы. Часто являясь абстракциями предметной области, они имеют наибольшее значение для пользователя, поэтому в их названиях применяются термины предметной области. Если существует проект базы данных, то можно обратиться к изучению названий таблиц, многие из них станут классами-сущностями. Обозначаются классы-сущности стереотипом <<**entity**>> либо специальной пиктограммой (рисунок 1.4).

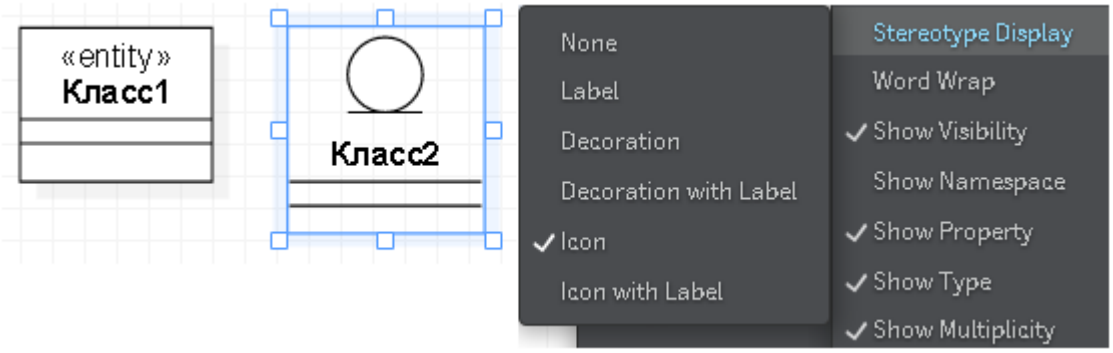

Рисунок 1.4 – Обозначение классов-сущностей

Граничными классами называются классы, расположенные на границе системы со всем остальным миром, и таким образом они обеспечивают взаимодействие между окружающей средой и внутренними элементами системы.

Для вычисления пограничных классов необходимо исследовать диаграммы вариантов использования. Для каждого взаимодействия между актером и прецедентом нужно создать хотя бы один граничный класс. Обратите внимание, что если два действующих лица инициируют один прецедент, то они могут применять один общий пограничный класс для взаимодействия с системой. Обозначаются граничные классы именем стереотипа <<**boundary**>> либо специальной пиктограммой (рисунок 1.5).

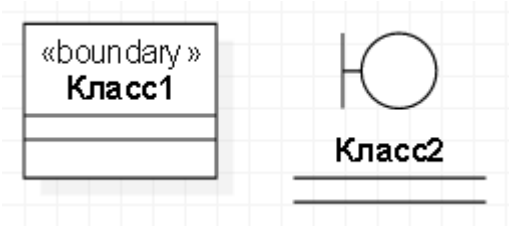

Рисунок 1.5 – Обозначение граничных классов

Управляющий класс отвечает за координацию действий других классов. Они служат для моделирования последовательного поведения одного или нескольких прецедентов и координации событий, реализующих заложенное в них поведение. Обозначаются управляющие классы именем стереотипа <<**control**>> либо специальной пиктограммой (рисунок 1.6).

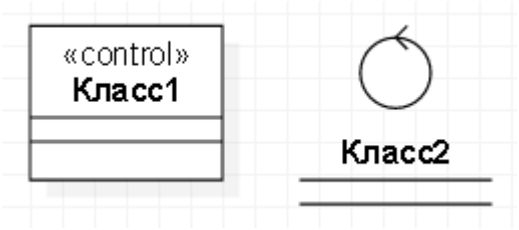

Рисунок 1.6 – Обозначение управляющих классов

Управляющие классы можно представить, как «исполняющие» прецедент, поэтому у каждого варианта использования обычно имеется один управляющий класс, контролирующий последовательность событий этого прецедента. Они обычно зависят от приложения.

Управляющий класс делегирует ответственности другим классам. Сам он может получать мало сообщений, но отсылать множество. Его называют классом-менеджером. Он запускает альтернативные потоки и знает, как поступить в случае ошибки. На начальном этапе проектирования управляющие классы создаются для каждой пары актер/прецедент, в дальнейшем они могут объединяться, разделяться или исключаться.

1.3. Документирование классов

После того, как класс создан, информацию о нем необходимо документировать. Заметим, что документация предназначена для описания предназначения класса, а не его структуры.

Если в нашей модели присутствует класс Сотрудник, то хорошим описанием для него будет: «Сотрудник – это человек, работающий на фирме. Класс содержит информацию, необходимую для исполнения организацией своих обязанностей по отношению к сотруднику (начисление зарплаты, перевод на другую должность, увольнение и т.п.)».

Плохим описанием будет описание структуры класса, которая может быть и так описана с помощью атрибутов. Например, плохое описание класса Сотрудник: «Имя, телефон, адрес, должность, зарплата».

В StarUML документирование классов выполняется также, как и описанное выше документирование прецедентов. Нужно выделить класс, который вы хотите описать, открыть окно документирования Documentation в инспекторе модели и ввести описание класса.

1.4. Построение диаграммы классов

Диаграммы классов относятся к логическому представлению системы Logical View. На диаграмме Main представления Logical View обычно размещают главную диаграмму пакетов, а диаграммы классов помещают на другие листы этого представления.

Для создания новой диаграммы классов выполним следующие шаги: щелкнуть правой кнопкой мыши по папке представления Logical View в навигаторе модели, в контекстном меню выбрать пункт Add Diagram, в списке выбрать диаграмму классов Class Diagram (рисунок 1.7).

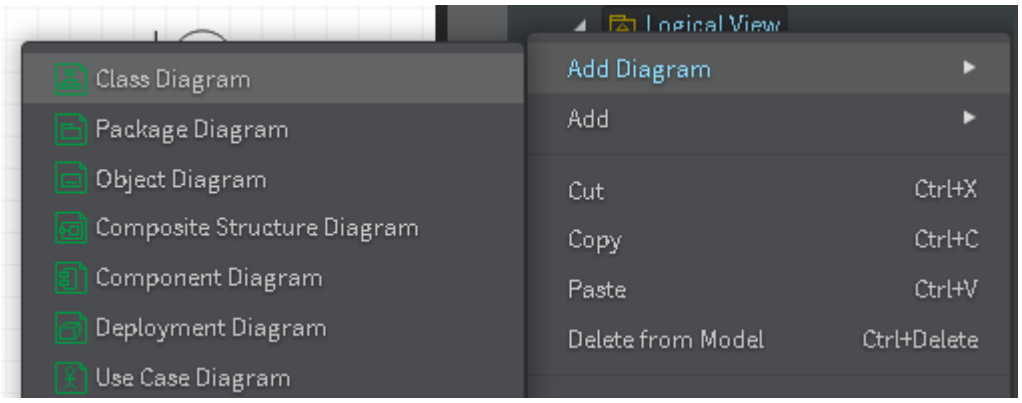

Рисунок 1.7 – Добавление диаграммы классов

Будет создана новая диаграмма классов со стандартным именем ClassDiagram1, которое можно изменить в редакторе свойств диаграммы (рисунок 1.8).

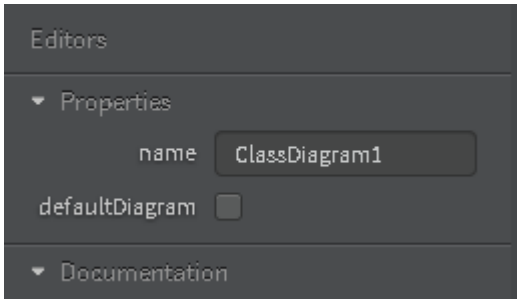

Рисунок 1.8 – Редактор свойств диаграммы классов (свойство Name)

Пример. Рассмотрим сценарий Оформление заказа, который является внутренним потоком для прецедента Заказ товара. Опишем его классы.

Данный сценарий позволяет покупателю оформить заказ из корзины и оплатить его с использованием банковской карты.

Давайте представим, как выполняется этот сценарий, еще раз. Покупатель, находясь в своей покупательской корзине, приняв решение о том, что он готов сделать заказ в магазине «Style», выбирает опцию «Оформить заказ». Как реагирует система на действия покупателя? Запускается сценарий Оформление заказа. Пользователь должен на специальной форме внести свои личные данные, подтвердить заказ или нет, и в зависимости от этого произвести оплату, затем получить подтверждение заказа. В системе появляется новый объект – заказ покупателя.

Построим диаграмму классов для сценария Оформление заказа в StarUML. Создайте в пакете Logical View диаграмму классов описанным выше способом. Переименуйте ClassDiagram1 в **Оформление Заказа (Place Order).**

Выбор граничных классов

По всей видимости, нам нужен будет хотя бы один граничный класс, который осуществляет связь между действующим лицом Покупатель и дополнительным прецедентом Оформить заказ. Назовем его ОформлениеЗаказа (PlaceOrder). Этот класс знает, какие товары и в каком количестве были в корзине покупателя, их нужно перенести в заказ. Также этот класс может знать, пуста корзина покупателя или нет, и, если пуста, то вывести об этом соответствующее сообщение. Для того, чтобы сделать заказ, покупатель должен ввести свои личные данные, электронный адрес, телефон и данные кредитной карты. Для этих целей мы введем еще один класс

ВводЛичныхДанных (EnterPersonalInformation). После того, как выбраны товары и введена личная информация покупателя, остается только проверить детали заказа и согласиться с ними или нет – для этого действия введем класс ПроверкаДеталейЗаказа (ConfirmOrder). Наконец, когда покупатель завершит оформление заказа, то на экране он видит номер заказа и подтверждение заказа отправляется покупателю на электронный адрес, для выполнения этих обязанностей создадим еще один граничный класс ПодтверждениеЗаказа (OrderConfirmation).

Выбор управляющих классов

Создадим один управляющий класс, который будет распределять обязанности других классов и вызывать их операции при выполнении данного сценария. Назовем этот управляющий класс МенеджерОформленияЗаказа (PlaceOrderManager).

Выбор классов-сущностей

В сценарии Оформление заказа речь идет о покупателе, заказе и товарах. Создадим классы-сущности Покупатель (Customer), Заказ (Order), Товар (Item). Возможно, что в ходе дальнейшего проектирования какие-то новые классы будут добавлены для этого сценария, а какие-то, напротив, из этой диаграммы удалены.

Создадим новый класс Покупатель (Customer). Щелкните правой кнопкой мыши по Logical View в навигаторе модели, в контекстном меню выберите пункт Add (Добавить), затем выберите пункт Class (Класс). Новый класс будет создан и отобразится в навигаторе модели. На вкладке Properties (Свойства) измените имя класса на Покупатель (Customer). Заметим, что при таком способе создания класса, StarUML создает новый элемент в навигаторе модели. Теперь перетащите этот класс из навигатора модели на рабочий лист диаграммы Оформление Заказа (Place Order) (рисунок 1.9).

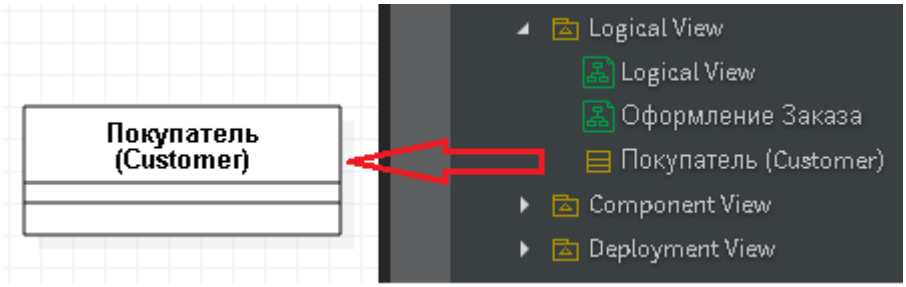

Рисунок 1.9 – Создание класса «Покупатель»

Для того чтобы создать класс ОформлениеЗаказа (PlaceOrder), можно использовать другой метод. Слева на панели инструментов (Toolbox) щелкните изображение класса (Class), затем щелкните на листе диаграммы **Оформление Заказа** в том месте, куда необходимо поместить класс, при этом стандартное имя класса станет активным, давая возможность его изменить. Измените имя класса на ОформлениеЗаказа (PlaceOrder) (рисунок 1.10).

Рисунок 1.10 – Создание класса «ОформлениеЗаказа»

Создайте все остальные классы, в результате итоговая диаграмма классов будет иметь вид, представленный на рисунке 1.11.

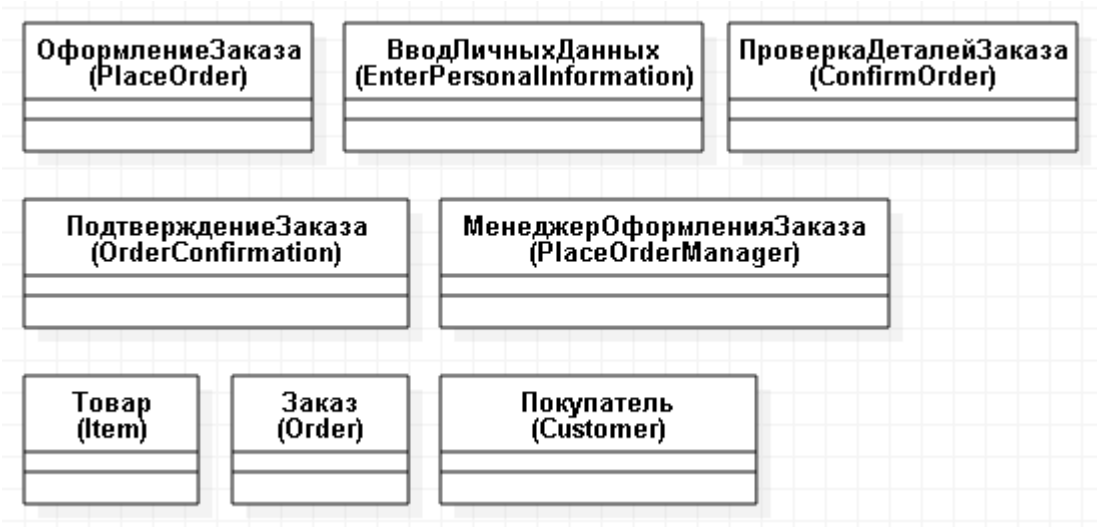

Рисунок 1.11 – Диаграмма классов сценария «Оформление заказа»

Замечание. Создание диаграммы классов и взаимодействия рекомендуется выполнять на английском языке, так как эти диаграммы могут участвовать в генерации программного кода системы. Конечно, если бы мы выполнили их на русском языке, ничего страшного не случилось бы, но сгенерированный код содержал бы имена на кириллице, и для дальнейшего использования программного кода, скорее всего, пришлось бы их заменить. В данном примере для удобства приводятся соответствующие переводы названий.

1.5. Назначение стереотипов

Для текущей модели предварительно необходимо подключить библиотеку стандартных стереотипов «UML Standard Profile» (рисунок 1.12).

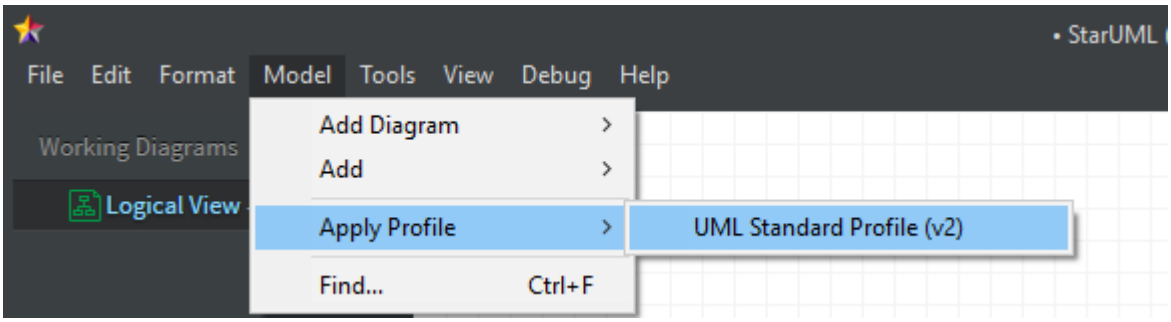

Рисунок 1.12 – Подключение UML Standard Profile

Чтобы присвоить классу один из стереотипов UML, нужно выделить класс щелчком правой кнопки мыши, открыть редактор свойств Properties на инспекторе модели и выбрать раздел **Stereotype** (рисунок 1.13).

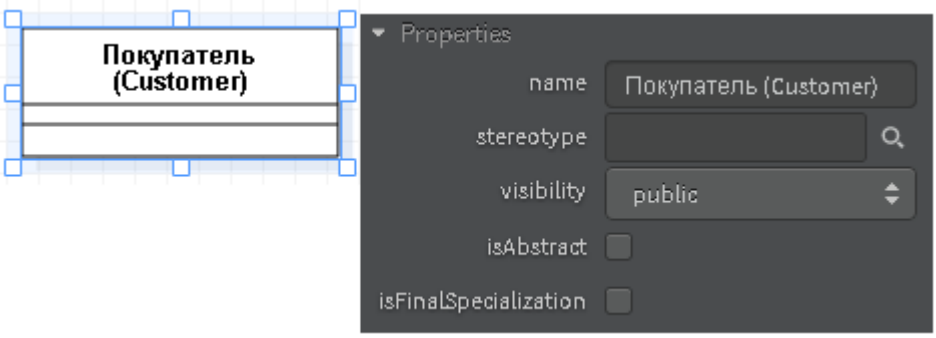

Рисунок 1.13 – Назначение стереотипа классу

При нажатии на значок  в появившемся диалоге будет представлен список доступных стереотипов. Для того чтобы присвоить классу стереотип, нужно выбрать его в предложенном списке или ввести необходимый тип и выбрать из обновленного списка (рисунок 1.14).

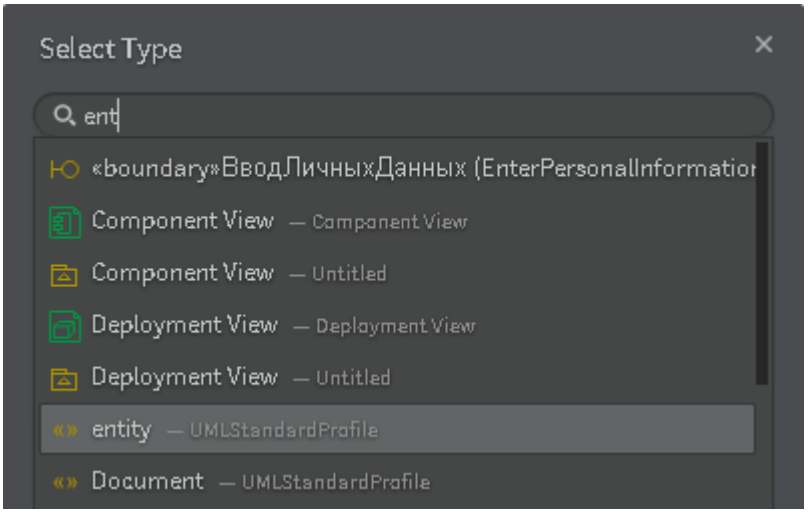

Рисунок 1.14 – Выбор стереотипа

После присвоения классу стереотипа его внешний вид изменится. Рядом с именем класса появится имя стереотипа, заключенное в угловые скобки.

Присвойте классам диаграммы **Оформление заказа** соответствующие стереотипы. Диаграмма классов изменится (рисунок 1.15).

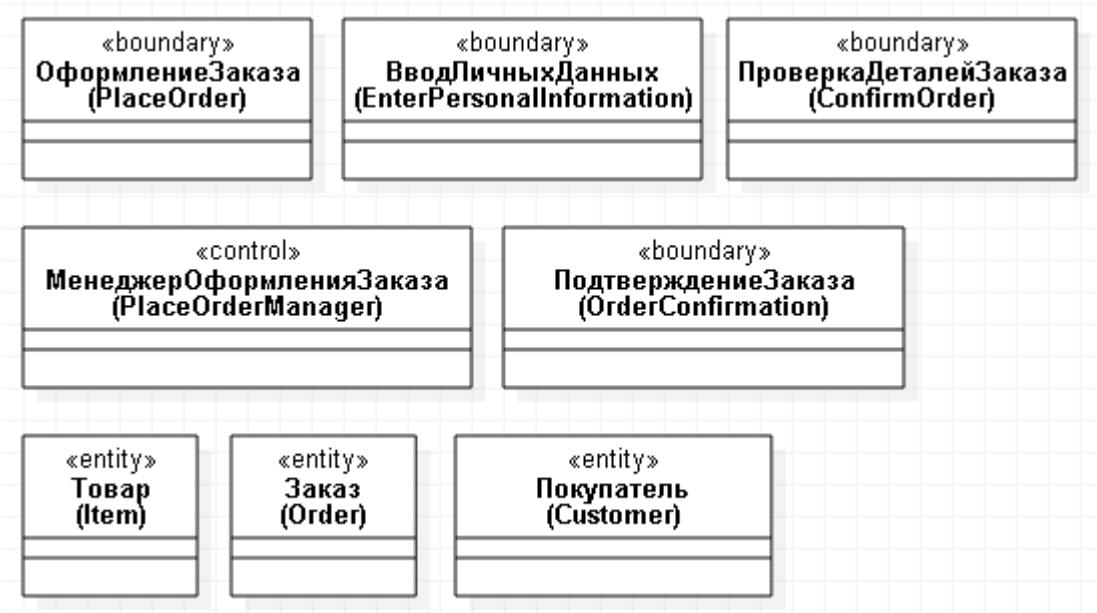

Рисунок 1.15 – Диаграмма классов сценария «Оформление заказа» со стереотипами

Мы можем отобразить стереотипы классов с помощью пиктограмм. Для этого нужно выделить класс, щелкнуть по выделенной области правой кнопкой мыши, в контекстном меню выбрать пункт **Format**, затем пункт **Stereotype Display**, далее в списке выбрать **Icon** (рисунок 1.16). До этого изменения по умолчанию было установлено отображение **Label**.

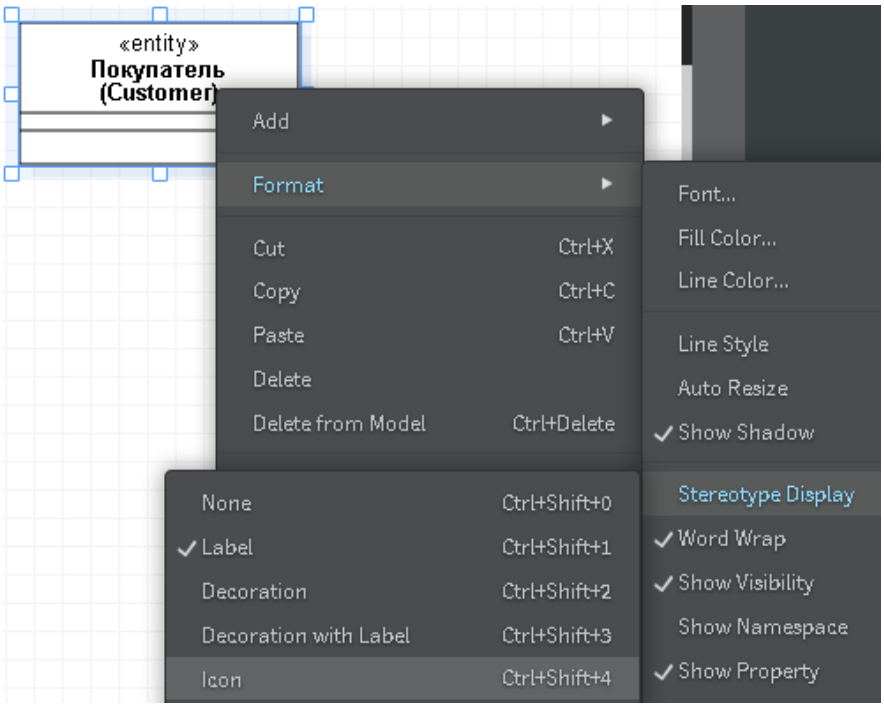

Рисунок 1.16 – Назначение отображения стереотипов классов в виде пиктограмм

В результате классы будут отображаться как пиктограммы (рисунок 1.17).

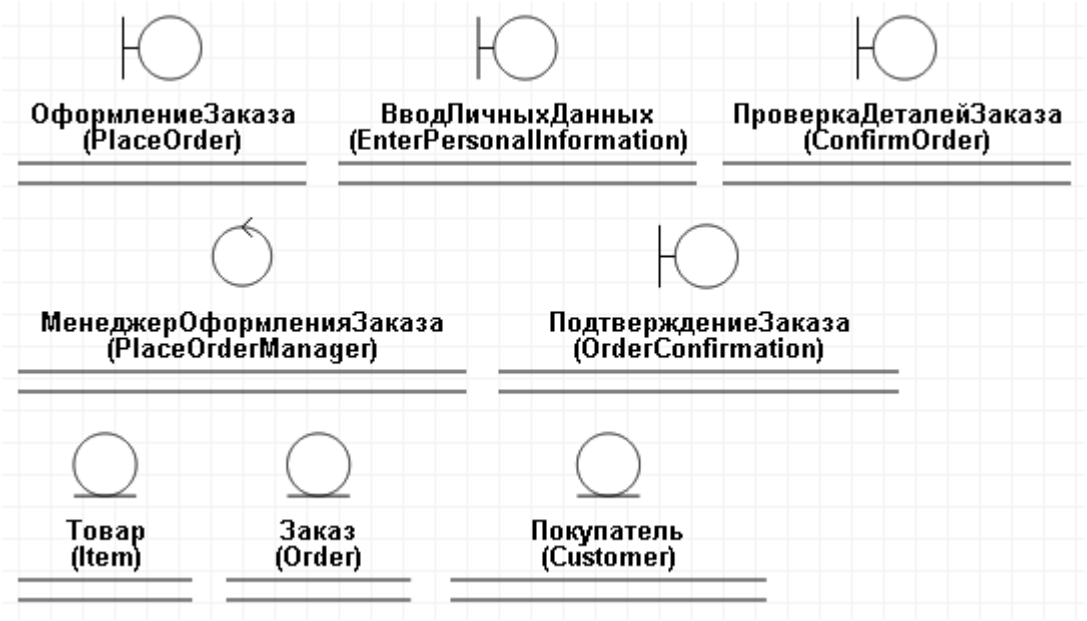

Рисунок 1.16 – Диаграмма классов сценария «Оформление заказа» в пиктограммах

2. Диаграмма пакетов (Package Diagram)

Если в нашей модели немного классов, то нам легко ими управлять, однако многие системы содержат большое количество классов, поэтому необходим механизм, позволяющий классы группировать и облегчающий их повторное использование. Таким механизмом в UML являются пакеты.

Пакет (package) – общецелевой механизм для организации различных элементов модели в группы. **Подпакет (subpackage) –** пакет, который является составной частью другого пакета.

Пакет в логическом представлении модели – это объединение классов или других пакетов. С помощью объединения классов в пакеты мы можем получить представление о системе на более высоком уровне. Напротив, рассматривая пакет, мы получаем более детальное представление модели.

Объединять классы в пакеты можно как угодно, однако, существует несколько наиболее распространенных подходов.

1. Можно группировать классы по стереотипам: классы-сущности, граничные и управляющие классы.
2. Группировка классов по их функциональности: например, пакет классов, отвечающих за безопасность системы или пакет классов Работа с сотрудниками и т.п.
3. Наконец, применяют комбинацию двух указанных методов. В дальнейшем можно вкладывать пакеты друг в друга.

Чаще всего пакет на диаграмме изображается в виде папки с закладкой с именем пакета (рисунок 2.1).

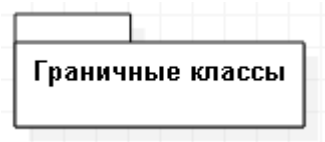

Рисунок 2.1 – Пакет «Граничные классы»

Главную диаграмму пакетов системы обычно помещают в представлении Logical View. Для того чтобы создать пакет на диаграмме, нужно открыть рабочее поле диаграммы, щелкнуть по элементу пакет **Package** на панели элементов слева, затем щелкнуть по рабочему полю диаграммы в том месте, где вы хотите поместить пакет. В окне редактора свойств можно задать новое имя пакета. Чтобы разместить классы по пакетам, используют метод перетаскивания: на навигаторе модели нужно перетащить, удерживая левую кнопку мыши, классы в соответствующие пакеты на навигаторе модели.

Пример. Пакеты можно разместить непосредственно на листе представления Logical View. Создайте пакеты Граничные классы, Классы-сущности, Управляющие классы (рисунок 2.2).

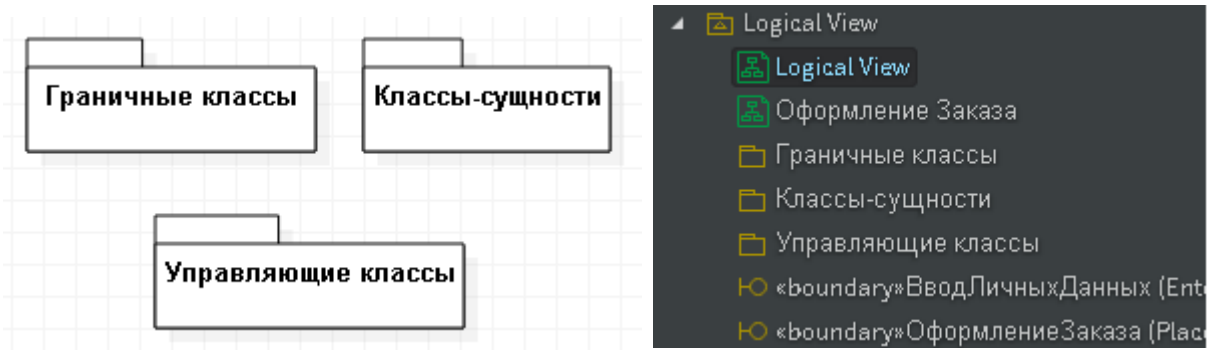

Рисунок 2.2 – Диаграмма пакетов и навигатор модели

Другой способ для размещения пакетов – создание новой диаграммы пакетов. Для представления Logical View в навигаторе модели в контекстном меню следует выбрать пункт Add Diagram, затем в списке выбрать диаграмму **Package Diagram** (рисунок 2.3).

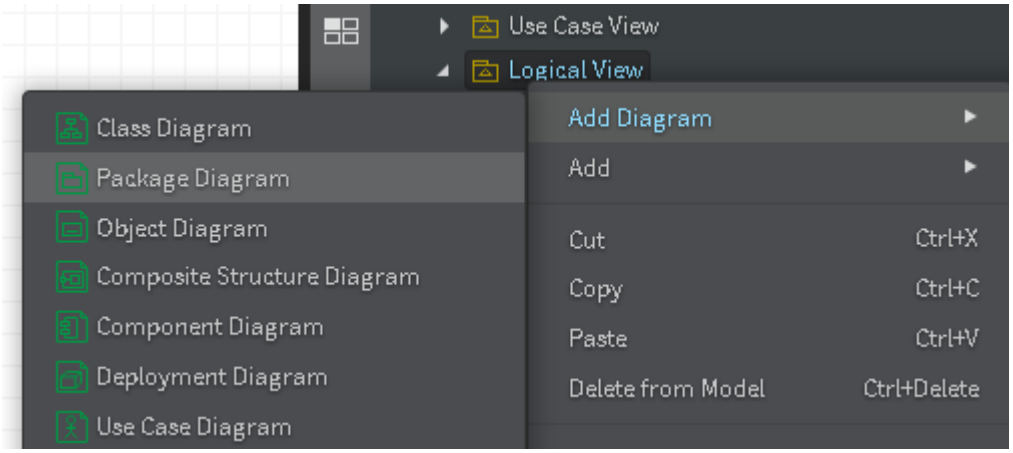

Рисунок 2.3 – Добавление диаграммы пакетов

В таком случае созданные пакеты можно разместить на рабочем листе диаграммы **PackageDiagram1** (рисунок 2.4).

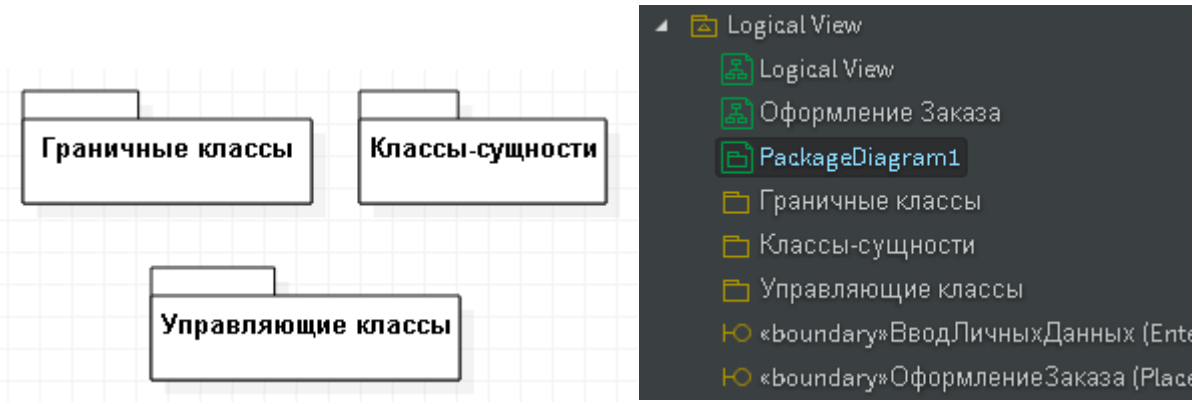

Рисунок 2.4 – Пакеты диаграммы PackageDiagram1

Группировка классов в пакеты

Созданные ранее классы диаграммы **Оформление заказа** сгруппируем по пакетам. Перетащите в навигаторе модели класс Customer (Покупатель) в пакет Классы-сущности. Аналогично классы Item (Товар) и Order (Заказ) разместим также в пакете Классы-сущности.

Классы ОформлениеЗаказа (PlaceOrder), ВводЛичныхДанных (EnterPersonalInformation), ПроверкаДеталейЗаказа (ConfirmOrder) и ПодтверждениеЗаказа (OrderConfirmation) разместим в пакете Граничные классы.

Класс МенеджерОформленияЗаказа (PlaceOrderManager) – в пакете Управляющие классы. Результат группировки классов по пакетам представлен на рисунке 2.5.

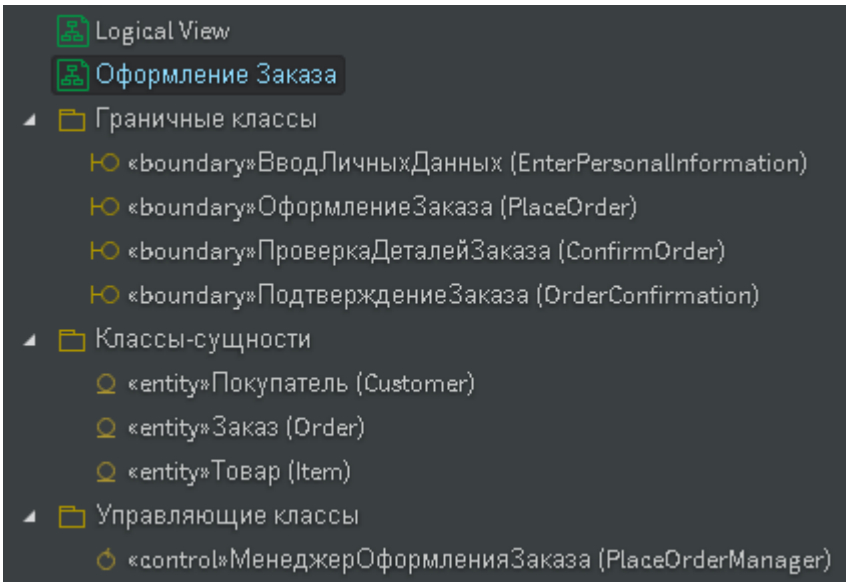

Рисунок 2.5 – Результат перемещения классов в пакеты

Обратите внимание, что при этом внешний вид диаграммы классов **Оформление заказа (Place Order)** и диаграммы пакетов не изменится.

3. Построение диаграмм взаимодействия

Диаграммы взаимодействия отображают один из процессов обработки информации в варианте использования: какие объекты нужны потоку, какими сообщениями обмениваются объекты, какие действующие лица инициируют поток и в какой последовательности отправляются сообщения.

Основной элемент диаграмм взаимодействия – это объект. Объектом описывают нечто содержащее в себе данные и поведение. Это термин, описывающий реальные, конкретные предметы или абстрактные сущности. Объекты изображаются в виде прямоугольников, имена объектов подчеркиваются. Поскольку наша система предназначена для заказа товаров, то, скорее всего нам придется построить диаграмму взаимодействия с участием такого объекта, как Товар (например, диаграмму, описывающую добавление товара в корзину).

Объекты, помещаемые на диаграммы взаимодействия, скорее всего, будут объектами системы и будут относиться к программному обеспечению. При проектировании этих диаграмм можно представлять себе объекты, как экранные формы или части приложений, отвечающие за выполнение определенных действий, или объектом может быть запись в таблице базы данных.

Если перед тем, как строить диаграммы взаимодействия мы построили диаграммы классов, то тогда поиск объектов упрощается. Объекты соответствуют своим классам или их операциям, и мы можем создавать и располагать их на диаграммах взаимодействия. Если мы хотим изучить взаимодействие объектов до того, как переходить к поиску классов, то поиск объектов можно начать с изучения имен существительных в созданных диаграммах. Многие из них станут хорошими кандидатами в объекты. Мы также можем выделить объекты-сущности, граничные объекты и управляющие объекты на основе выбранных классов.

Существует **два типа** диаграмм взаимодействия:

- диаграммы последовательности действий (**Sequence Diagram**);
- диаграммы кооперации (**Communication/Collaboration Diagram**).

Первые отображают обмен сообщениями между объектами во времени, а вторые отображают структуру взаимодействия. На обеих диаграммах отображается одна и та же информация, но разными способами: диаграмма последовательностей изображает поток управления, а кооперативная диаграмма – поток данных.

3.1. Диаграммы последовательности. Элементы нотации

Как правило, поток событий описывает не одну последовательность действий, а несколько возможных, это отражается наличием главного потока событий и альтернативных потоков. Чаще всего невозможно описать прецедент с помощью только одной последовательности действий. Например, для прецедента Заказ товаров возможно оформление заказа без изменения корзины, с изменением состава корзины, или покупатель, просмотрев корзину, захочет вернуться в каталог и что-то в нее добавить, возможно, вернувшись в каталог, покупатель не станет ничего больше добавлять, а снова вернется в корзину и оформит заказ. Каждый такой вариант мы можем описать своей последовательностью действий, своим сценарием. И, таким образом, один прецедент описывает несколько последовательностей – сценариев, каждый из которых описывает один из вариантов возможного потока событий.

Сценарий (Scenario) – это некоторая последовательность действий, иллюстрирующая поведение системы. Сценарий – это экземпляр потока событий. Он представляет собой одиночный проход по потоку событий для прецедента. Для графического отображения сценария используются диаграммы последовательностей. **Диаграмма последовательности действий** отображает взаимодействие объектов, упорядоченное по времени.

На диаграммах последовательности изображаются объекты, классы и последовательность сообщений, которыми обмениваются объекты в ходе выполнения сценария (рисунок 3.1).

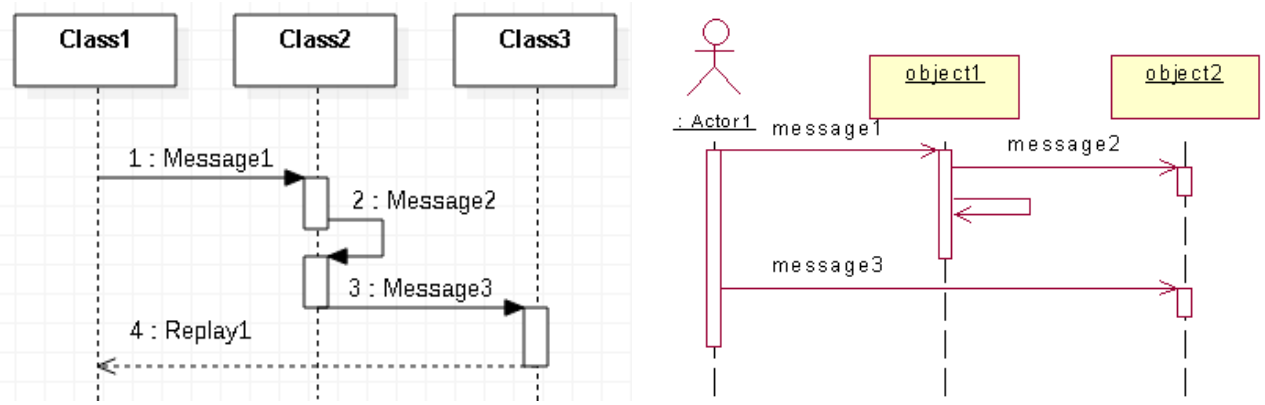

Рисунок 3.1 – Примеры диаграмм последовательности

На диаграмме последовательностей могут также изображаться экземпляры действующих лиц. Действующие лица, присутствующие на диаграммах взаимодействия, выделяются из потока событий как сущности, запускающие процессы. На одной диаграмме их может быть несколько.

Для того чтобы поместить действующее лицо на диаграмму, нужно найти его в навигаторе модели справа и перетащить на поле рабочее диаграммы последовательностей.

Каждый объект или действующее лицо на диаграмме последовательностей имеет свою линию жизни, которая обозначается пунктиром.

Линия жизни объекта (object lifeline) – вертикальная пунктирная линия на диаграмме последовательности, которая представляет существование объекта в течение определенного периода времени.

Фокус управления (активность, focus of control) – специальный символ на диаграмме последовательности, указывающий период времени, в течение которого объект выполняет некоторое действие, находясь в активном состоянии. Фокус управления изображается тонким прямоугольником, расположенным на линии жизни (см. рисунок 3.1).

Объекты и действующие лица на диаграммах последовательности обмениваются сообщениями. Сообщения обозначаются стрелками, идущими от отправителя к получателю.

Сообщение (message) – спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.

Для сообщений на диаграммах последовательностей, как и для других элементов модели, доступен ряд спецификаций.

Во-первых, у каждого сообщения должно быть имя, соответствующее его цели. Во-вторых, сообщения на диаграммах последовательностей можно соотнести с операциями, определенными для классов. Если от одного объекта к другому направлено сообщение, то это означает, что объект-источник вызывает операцию объекта-приемника.

Объект не может вызвать произвольную операцию: она должна быть доступна этому объекту. В особых случаях сообщение не становится операцией: например, ввод логина и пароля подразумевает их печать в соответствующих полях, и сообщение будет реализовано в виде поля ввода в окне программы. Процедура создания операций из сообщений будет описана ниже.

В-третьих, мы можем для каждого сообщения установить тип синхронизации. Каждому типу соответствует его обозначение.

Вызов операции (процедуры) вызывает операцию того объекта, к которому направлено. Объект может вызвать свою операцию. Тогда стрелка начинается и заканчивается на линии жизни одного и того же объекта, такое сообщение называется **рефлексивным**.

Синхронное (Message) сообщение обозначается стрелкой с закрашенным наконечником (рисунок 3.2).

Асинхронное сообщение (Async Message) посылает объекту сигнал. При этом источник не ждет отклика приемника или подтверждения получения, а продолжает свою работу. Обозначается нежирной стрелкой (рисунок 3.2).

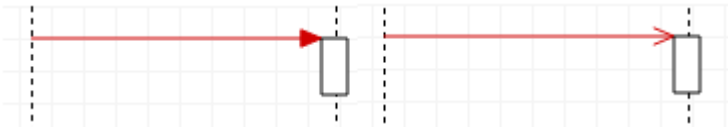

Рисунок 3.2 – Синхронное и асинхронное сообщение

Ответное сообщение (Replay Message) возвращает значение из процедуры тому объекту, к которому направлено. Обозначается пунктирной стрелкой (рисунок 3.3).

Рисунок 3.3 – Ответное сообщение

Создание объекта (Create Message) – создает новый объект. Обозначается стрелкой со стереотипом <<create>> (рисунок 3.4).

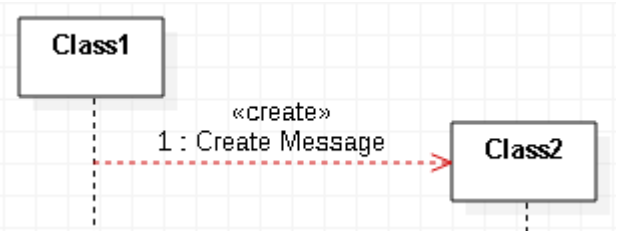

Рисунок 3.4 – Создание объекта

Удаление объекта (Delete Message) – удаляет объект. Объект может уничтожить сам себя. Обозначается стрелкой со стереотипом <<destroy>>. При уничтожении объекта на его линии жизни появляется символ разрушения, который обозначается крестом (рисунок 3.5).

Рисунок 3.5 – Уничтожение объекта

3.2. Добавление диаграммы последовательности

Для создания новой диаграммы последовательности нужно выполнить следующие шаги: щелкнуть правой кнопкой мыши по папке представления Logical View в навигаторе модели, в контекстном меню выбрать пункт Add Diagram, в списке выбрать диаграмму последовательности **Sequence Diagram** (рисунок 3.6).

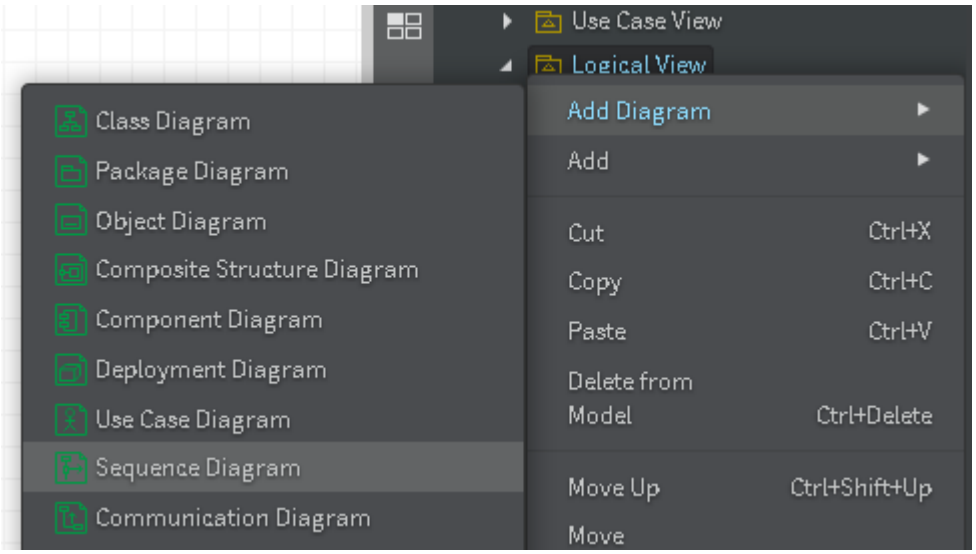

Рисунок 3.6 – Добавление диаграммы последовательности

Мы также можем использовать диаграмму последовательности для детализации прецедента. Для этого нужно связать диаграмму с прецедентом: для создания диаграммы щелкните правой кнопкой мыши по прецеденту, а не по папке Logical View. Однако если строится диаграмма последовательности для анализа системы, то лучше все-таки помещать ее в Logical View.

Пример. Мы уже определили классы сценария Оформление заказа, теперь с помощью диаграммы последовательности покажем, как взаимодействуют объекты этих классов во времени.

Для создания диаграммы последовательностей можно воспользоваться методом перетаскивания. Если перетащить необходимый класс из навигатора модели на диаграмму последовательности, то будет создан анонимный объект этого класса. Можно изменить имя объекта или удалить, оставив только имя класса (рисунок 3.7).

Рисунок 3.7 – Объект класса «Покупатель»

Составим диаграмму последовательности для случая, когда покупатель успешно оформляет заказ (рисунок 3.8).

Действующее лицо Покупатель необходимо перетащить из диаграммы прецедентов на диаграмму последовательности. Класс ОформлениеЗаказа (PlaceOrder) и последующие элементы также следует добавлять на диаграмму последовательности методом перетаскивания классов уже созданных на диаграмме классов.

Покупатель выбирает опцию «Оформить заказ» (Place order), при этом вызывается некоторый объект PlaceOrder (это будет граничный объект, принадлежащий соответствующему граничному классу).

Далее открывается форма ввода личных данных покупателя и его кредитной карты (EnterPersonalInformation), на ней покупатель вводит свое имя, адрес, телефон, адрес электронной почты (Enter personal information) и кредитные данные.

Информация принимается и открывается форма подтверждения заказа (ConfirmOrder), покупатель подтверждает, что согласен с реквизитами заказа (Confirm order), детали заказа сохраняются для дальнейшего использования (Save details). Фокус управления с помощью сообщения Place order передается некоторому управляющему объекту (PlaceOrderManager), который обращается к внешней кредитной системе для проведения платежа (сообщение Validate credit). Элемент Кредитная система следует параллельно создать на диаграмме прецедентов и использовать на диаграмме последовательности.

Если платеж прошел успешно (именно такой сценарий сейчас рассматриваем), то PlaceOrderManager посылает сообщение (Create order) создать объект Заказ (Order). Объект Заказ (Order) обращается к объектам Товар (Item) для того, чтобы получить информацию о товарах и создает заказ. Класс PlaceOrderManage вызывает форму подтверждения заказа (OrderConfirmation) с помощью сообщения Display order. Процесс завершается.

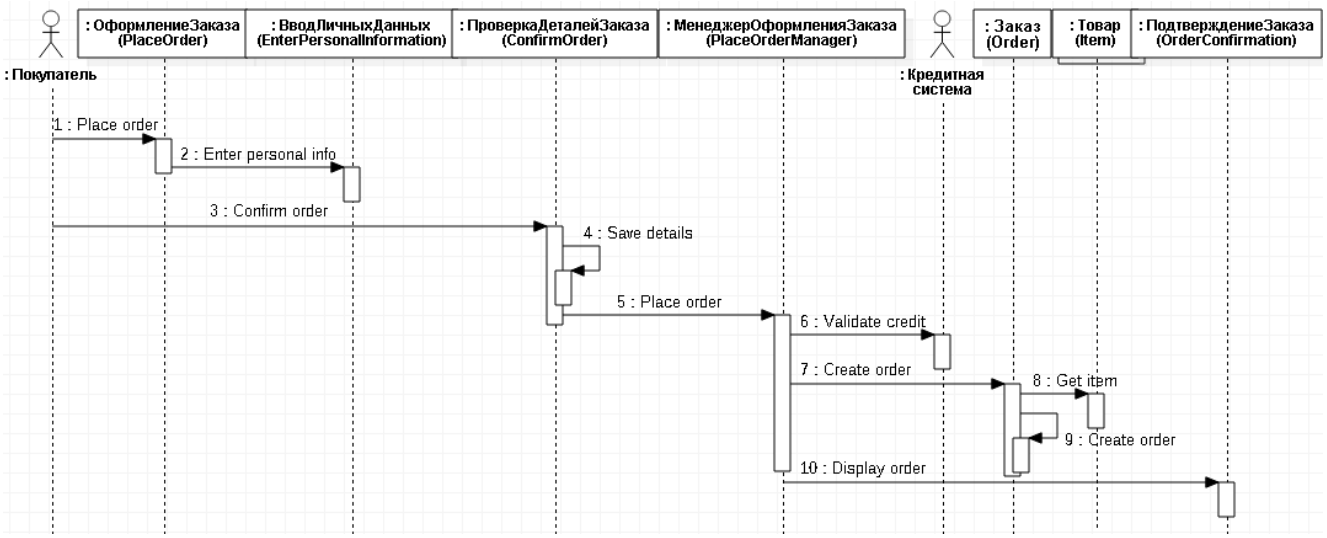

Рисунок 3.8 – Диаграмма последовательности сценария «Оформление заказа»

Замечание. Обратите внимание, что символ объекта Товар (Item) на диаграмме последовательности отличается от символов других объектов. Дело в том, что мы задали множественный экземпляр класса. Действительно, заказ может состоять из нескольких товаров, значит объекту Заказ (Order) требуется получить информацию о нескольких объектах Товар (Item).

Вместо того чтобы представлять каждый товар отдельно, мы используем нотацию UML для множественного экземпляра класса, представляя одним значком несколько объектов.

Чтобы сделать объект множественным в StarUML выделите объект, щелкнув по нему мышью один раз, в открывшемся редакторе свойств поставьте флажок в разделе **IsMultiInstance** (рисунок 3.9).

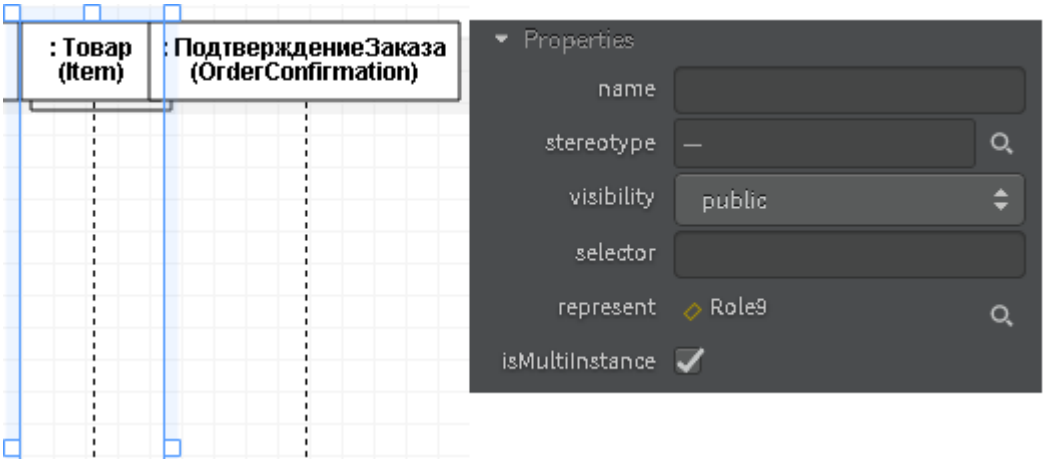

Рисунок 3.9 – Создание множественного объекта

3.3. Взаимосвязь диаграмм классов и последовательности

Процесс построения модели системы является итеративным. Особенно хорошо это можно видеть при создании диаграмм классов и последовательности. Какую диаграмму создавать первой: классов или последовательности? Одни разработчики начинают с диаграмм классов, другие – наоборот, с последовательности. И в том и в другом случае, скорее всего, обе эти диаграммы, построенные для одного сценария, будут в дальнейшем подвергаться изменению. После построения диаграмм последовательности на диаграммах классов могут появиться новые классы, а на диаграммах последовательности – новые объекты, которых раньше там не было, но они придут туда из диаграмм классов. Возможно, что некоторые объекты и классы будут, напротив, удалены.

Замечание. Для создания диаграммы последовательностей, мы могли каждый объект этой диаграммы создавать заново, а не пользоваться методом перетаскивания.

3.4. Добавление кооперативной диаграммы

Диаграмма кооперации – это альтернативный способ изображения сценария варианта использования. Этот тип диаграмм заостряет внимание на связях между объектами, отображая обмен данными в системе. А диаграммы последовательности отображают взаимодействие объектов во времени, поэтому ее следует читать сверху вниз и слева направо.

Диаграммы кооперации содержат все те же элементы, что и диаграммы последовательности: объекты, действующие лица, связи между ними и сообщения, которыми они обмениваются, но они уже не упорядочены во времени.

Для того чтобы добавить диаграмму кооперации в представление Logical View, щелкните правой кнопкой мыши по папке содержащей диаграмму последовательности (если вы ее не переименовывали, то эта папка в навигаторе носит имя **Collaboration1**), в контекстном меню выберите пункт Add Diagram, в списке выберите диаграмму кооперации **Communication Diagram** (рисунок 3.10).

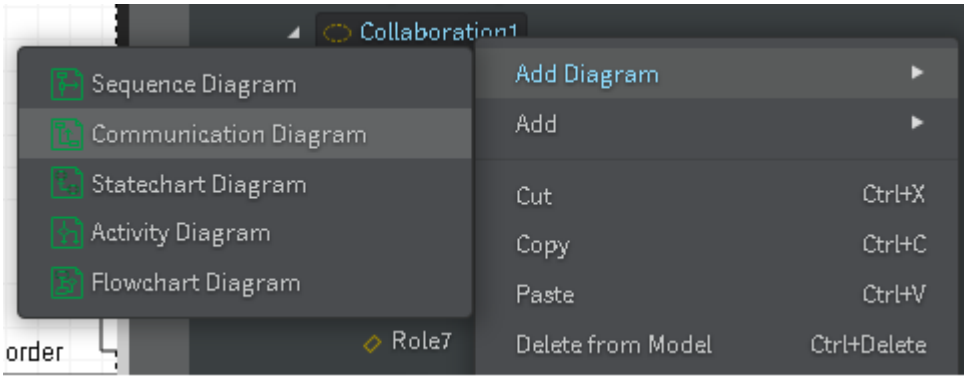

Рисунок 3.10 – Добавление кооперативной диаграммы

Рассмотрим пример для сценария Оформление заказа, для которого уже составлена ранее диаграмма последовательности.

На созданной диаграмме кооперации следует поместить все те же объекты, перетащив их из навигатора модели и назначить советующие сообщения-операции (рисунок 3.11).

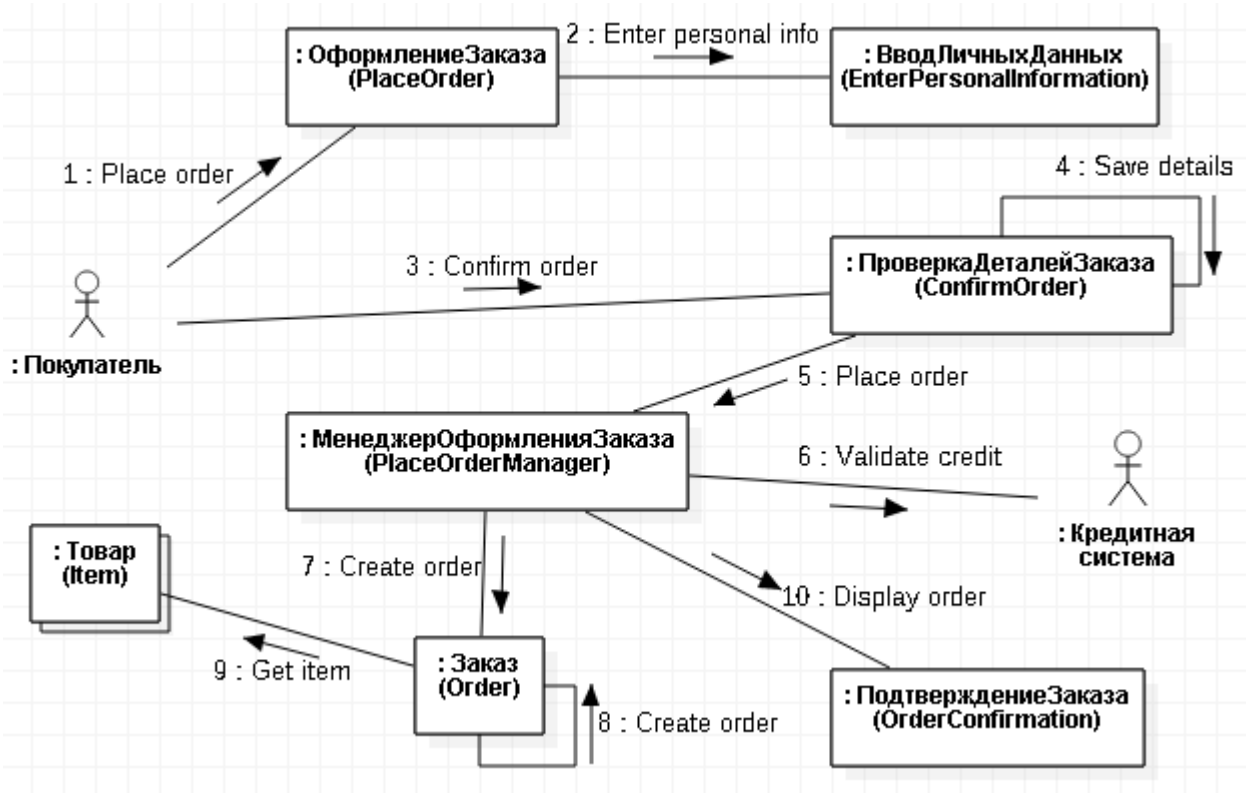

Рисунок 3.11 – Кооперативная диаграмма сценария «Оформление заказа»