Лабораторная работа №4

Разработка диаграмм классов с помощью Visual Paradigm for UML

Цель работы

Целью работы является изучение основ создания диаграмм классов на языке UML с помощью case-средства Visual Paradigm for UML CE, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

Задачи

Основными задачами практической работы являются:

– получить навыки создания диаграмм классов с помощью case-средства Visual Paradigm for UML CE.

Краткие теоретические сведения

Диаграмма классов отражает различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений.

Диаграмма классов (class diagram) – диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

Диаграмма классов предназначена для представления статической структуры модели системы в терминологииклассов объектно-ориентированного программирования.

Диаграмма классов содержит интерфейсы, пакеты, отношения и даже отдельные экземпляры классификаторов, такие как объекты и связи.

Класс (class) – абстрактное описание множества однородных объектов, имеющих одинаковые атрибуты, операции и отношения с объектами других классов.

Графически изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 1). В этих секциях могут указываться имя класса, атрибутыи операции класса. Даже если секции атрибутов и операций пусты, в обозначении класса они должны быть выделены горизонтальной линией.

Рисунок 1 – Варианты графического изображения класса на диаграмме классов

В первом случае для класса Окружность (рис. 2, а) указаны только его атрибуты – точка на координатной плоскости, которая определяет расположение ее центра. Для класса Окно (рис. 2, б) указаны только его операции,при этом секция его атрибутов оставлена пустой. Для класса Счет (рис. 2, в) дополнительно изображена четвертаясекция, в которой указано требование – реализовать резервное копирование объектов этого класса.

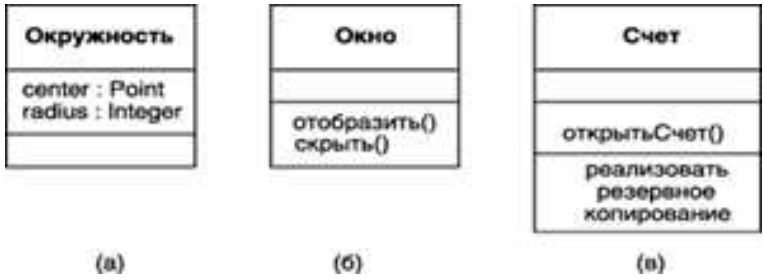

Рисунок 2 – Примеры графического изображения конкретных классов

Имя класса должно быть уникальным в пределах пакета, который может содержать одну или несколько диаграмм классов. Имя класса записывается по центру секции имени полужирным шрифтом и должно начинатьсяс заглавной буквы.

В секции имени класса могут также находиться стереотипы или ссылки на стандартные шаблоны, от которыхобразован данный класс и, соответственно, от которых он наследует атрибуты и операции. Здесь также могут записываться и другие общие свойства этого класса.

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в **языке UML различают**

конкретные и абстрактные классы.

Конкретный класс (concrete class) – класс, на основе которого могут быть непосредственно созданыэкземпляры или объекты.

Рассмотренные выше обозначения относятся к конкретным классам.

Абстрактный класс (abstract class) – класс, который не имеет экземпляров или объектов. Для обозначения имени абстрактного класса используется курсив.

Атрибут (attribute) – содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса.

Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, егоисходного значения.

Общий формат записи отдельного атрибута класса следующий:

<квантор видимости> <имя атрибута> [кратность] : <тип атрибута> = <исходное значение> {строка-свойство}

Видимость (visibility) – качественная характеристика описания элементов класса, характеризующаяпотенциальную возможность других объектов модели оказывать влияние на отдельные аспекты поведения данногокласса.

Видимость в языке UML специфицируется с помощью квантора видимости (может быть опущен), который может принимать одно из 4-х возможных значений и отображаться при помощи специальных символов.

– "+" – область видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.

– "#" – область видимости типа защищенный (protected). Атрибут недоступен или невиден для всех классов,за исключением подклассов данного класса.

– "-" – область видимости типа закрытый (private). Атрибут недоступен или невиден для всех классов без исключения.

– "~" – область видимости типа пакетный (package). Атрибут недоступен или невиден

для всех классов за пределами пакета, в котором определен класс-владелец данного атрибута.

Вместо условных графических обозначений можно записывать соответствующее ключевое слово: public, protected, private, package.

Имя атрибута используется в качестве идентификатора соответствующего атрибута и поэтому должно быть уникальным в пределах данного класса. Имя атрибута - обязательный элемент, должно начинаться со строчной (малой) буквы и не должно содержать пробелов.

Кратность (multiplicity) – спецификация области значений допустимой мощности, которой могут обладать соответствующие множества.

Тип атрибута представляет собой выражение, семантика которого определяется некоторым типом данных, определенным в пакете Типы данных языка UML или самим разработчиком, или в зависимости от языка программирования, который предполагается использовать для реализации данной модели.

Исходное значение служит для задания начального значения соответствующего атрибута в момент создания отдельного экземпляра класса. Если оно не указано, то значение соответствующего атрибута не определено на момент создания нового экземпляра класса.

Пояснительный текст в фигурных скобках может означать две различные конструкции. Если в этой строке имеется знак равенства, то вся запись Строка-свойство служит для указания дополнительных свойств атрибута, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения программы. Фигурные скобки как раз и обозначают фиксированное значение соответствующего атрибута для класса в целом, которое должны принимать все вновь создаваемые экземпляры класса без исключения. Это значение принимаетсяза исходное значение атрибута, которое не может быть переопределено в последующем.

Отсутствие строки-свойства по умолчанию трактуется так, что значение соответствующего атрибута может быть изменено в программе.

Знак "/" перед именем атрибута указывает на то, что данный атрибут является производным от некоторого другого атрибута этого же класса.

Операция (operation) – это сервис, предоставляемый каждым экземпляром или объектом класса по требованию своих клиентов, в качестве которых могут выступать другие объекты, в том числе и экземпляры данного класса.

Общий формат записи отдельной операции класса следующий:

<квантор видимости> <имя операции>(список параметров):

<выражение типа возвращаемого значения> {строка-свойство}

Квантор видимости операции аналогичен квантору видимости атрибутов.

Имя операции – обязательный элемент, должно начинаться со строчной (малой) буквы, и записываться без пробелов.

Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:

<направление параметра> <имя параметра> : <выражение типа> = <значение параметра по умолчанию>

Параметр (parameter) – спецификация переменной операции, которая может быть изменена, передана или возвращена.

Выражение типа возвращаемого значения указывает на тип данных значения, которое возвращается объектом после выполнения соответствующей операции. Две точки и выражение типа возвращаемого значения могут быть опущены, если операция не возвращает никакого значения. Для указания нескольких возвращаемых значений данный элемент спецификации операции может быть записан в виде списка отдельных выражений.

Кроме внутреннего устройства классов важную роль при разработке проектируемой системы имеют различные отношения между классами, которые также могут быть изображены на диаграмме классов. Базовые отношения, изображаемые на диаграммах классов:

– отношение ассоциации (association relationship);
– отношение обобщения (generalization relationship);
– отношение агрегации (aggregation relationship);
– отношение композиции (composition relationship).

Отношение ассоциации соответствует наличию произвольного отношения или взаимосвязи между классами.Оно обозначается сплошной линией со стрелкой или без нее и с дополнительными символами, которые характеризуют специальные свойства ассоциации.

В качестве простого примера **ненаправленной бинарной ассоциации** можно рассмотреть отношение между двумя классами – классом «Компания» и классом «Сотрудник» (рис. 3). Они связаны между собой бинарной ассоциацией «Работает». Для данного отношения определен следующий порядок чтения следования классов - сотрудник работает в компании.

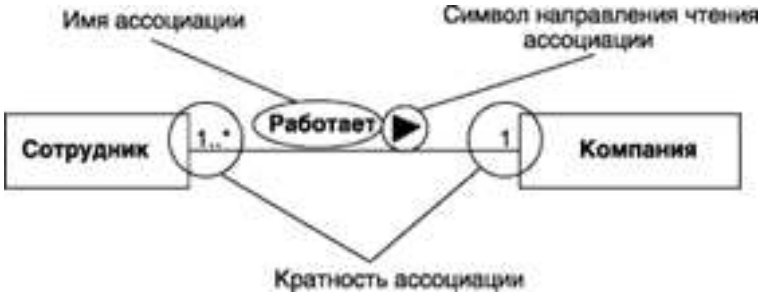

Рисунок 3 – Графическое изображение ненаправленной бинарной ассоциации между классами

Направленная бинарная ассоциация изображается сплошной линией с простой стрелкой на одной из ееконцевых точек. Направление этой стрелки указывает на то, какой класс является первым, а какой – вторым.

В качестве простого примера направленной бинарной ассоциации можно рассмотреть отношение между двумяклассами – классом «Клиент» и классом «Счет» (рис. 4). Они связаны между собой бинарной ассоциацией с именем

«Имеет», для которой определен порядок следования классов. Это означает, что конкретный объект класса

«Клиент» всегда должен указываться первым при рассмотрении взаимосвязи с объектом класса «Счет», например,

<клиент, счет_1, счет_2,…, счет_n>.

Рисунок 4 – Графическое изображение направленной бинарной ассоциации между классами

Частный случай отношения ассоциации – так называемая исключающая ассоциация (Xor-association).

Семантика данной ассоциации указывает на то, что из нескольких потенциально возможных вариантов даннойассоциации в каждый момент времени может использоваться только один.

На диаграмме классов исключающая ассоциация изображается пунктирной линией, соединяющей две и болееассоциации (рис. 5), рядом с которой записывается ограничение в

форме строки текста в фигурных скобках: {xor}.

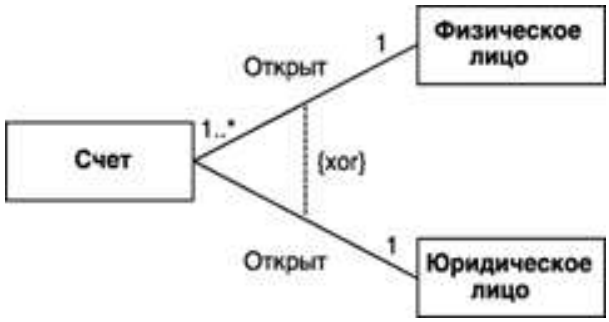

Рисунок 5 – Графическое изображение исключающей ассоциации между тремя классами

Отношение обобщения является отношением классификации между более общим элементом (родителем илипредком) и более частным или специальным элементом (дочерним или потомком).

Менее общий элемент модели должен быть согласован с более общим элементом и может содержать дополнительную информацию. Отношение обобщения описывает иерархическое строение классов и наследованиеих свойств и поведения.

Согласно одному из главных принципов методологии ООП – наследованию, класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет собственные свойства и поведение, которые могут отсутствовать у класса-предка.

Отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов (рис. 6). Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее начало – на более специальный класс(класс-потомок или подкласс).

Рисунок 6 – Графическое изображение отношения обобщения в языке UML

От одного класса-предка одновременно могут наследовать несколько классов-потомков. т.е. в класс-предоквходит несколько линий со стрелками.

Пример: класс «Транспортное средство» (курсив обозначает абстрактный класс) может выступать в качествесуперкласса для подклассов, соответствующих конкретным транспортным средствам, таким как: «Автомобиль», «Автобус», «Трактор» и другим (рис 7).

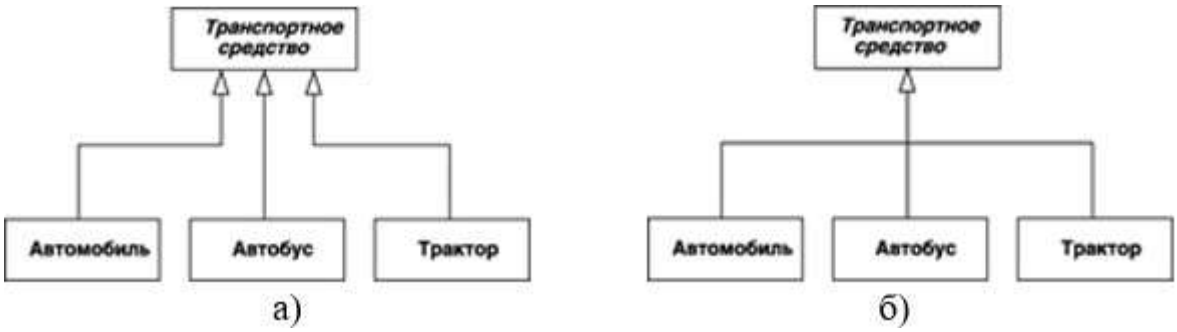

Рисунок 7 – а) Пример графического изображения отношения обобщения для нескольких классов- потомков; б) Альтернативный вариант графического изображения отношения обобщения классов для случая объединения отдельных линий

Отношение агрегации имеет место между несколькими классами в том случае, если один из классовпредставляет собой сущность, которая включает в себя в качестве составных частей другие сущности. Данное отношение применяется для представления системных взаимосвязей типа «часть-целое» и показывает, из каких элементов состоит система, и как они связаны между собой.

Очевидно, что рассматриваемое в таком аспекте деление системы на составные части представляет собой иерархию, но принципиально отличную от той, которая порождается отношением обобщения. Отличие заключается в том, что части системы никак не обязаны наследовать ее свойства и поведение, поскольку являютсясамостоятельными сущностями. Более того, части целого обладают собственными атрибутами и операциями, которые существенно отличаются от атрибутов и операций целого. Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой «целое» или класс-контейнер. Остальные классы являются его «частями» (рис. 8).

Рисунок 8 – Графическое изображение отношения агрегации в языке UML

В качестве примера отношения агрегации можно рассмотреть взаимосвязь типа «часть-целое», которая имеетместо между классом «Системный блок» персонального компьютера и его составными частями: «Процессор»,

«Материнская плата», «Оперативная память», «Жесткий диск» и «Дисковод гибких дисков» (рис. 9).

Рисунок 9 – Диаграмма классов для иллюстрации отношения агрегации на примере системного блока ПК

Отношение композиции – частный случай отношения агрегации, и служит для спецификации более сильнойформы отношения «часть-целое», при которой составляющие части тесно взаимосвязаны с целым. Особенность: суничтожением целого уничтожаются и все его составные части.

Пример (рис. 10): Программное средство представляет собой базу данных «Автоматизация процесса составления расписания в учебном заведении». Программное средство обеспечивает корректировку данных, а именно в БД «Группы» может изменяться перечень предметов в соответствие с курсом группы и отделением; в БД

«Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретномупредмету, а также отчет о загрузке аудиторий на каждый день.

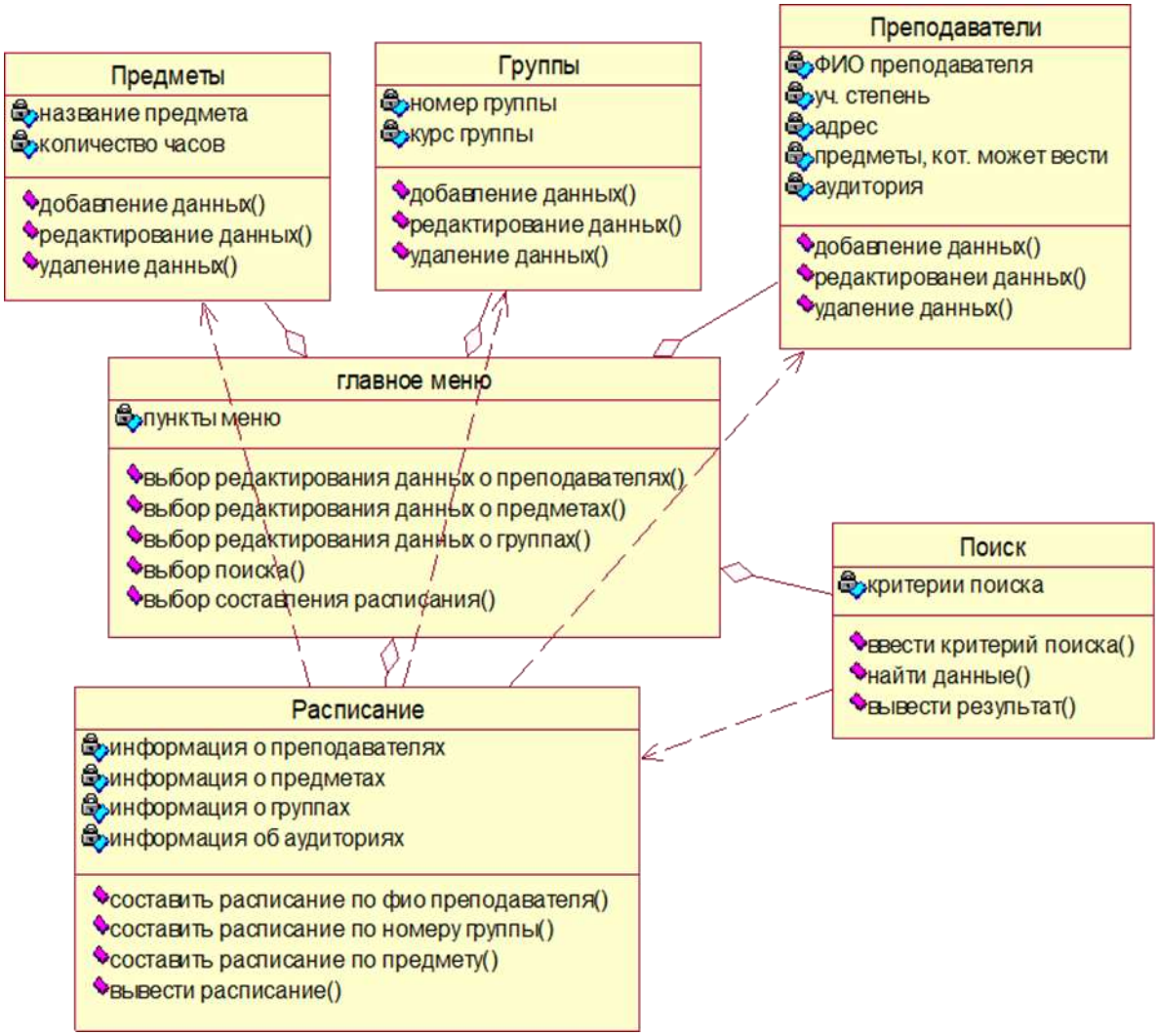

Рисунок 10 – Пример диаграммы классов

1. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите Visual Paradigm for UML CE.
2. Откройте проект, содержащий диаграмму вариантов использования, построенную в практическом занятии №1.
3. Для добавления новой диаграммы достаточно в меню Diagram > New Diagram выбрать тип диаграммы ClassDiagram и нажмите кнопку Next.
4. Задайте имя диаграммы SaleSystemClassDiagram. Нажмите OK.
5. В результате отобразится пустая рабочая область с элементами для построения диаграммы классов (рис.11).

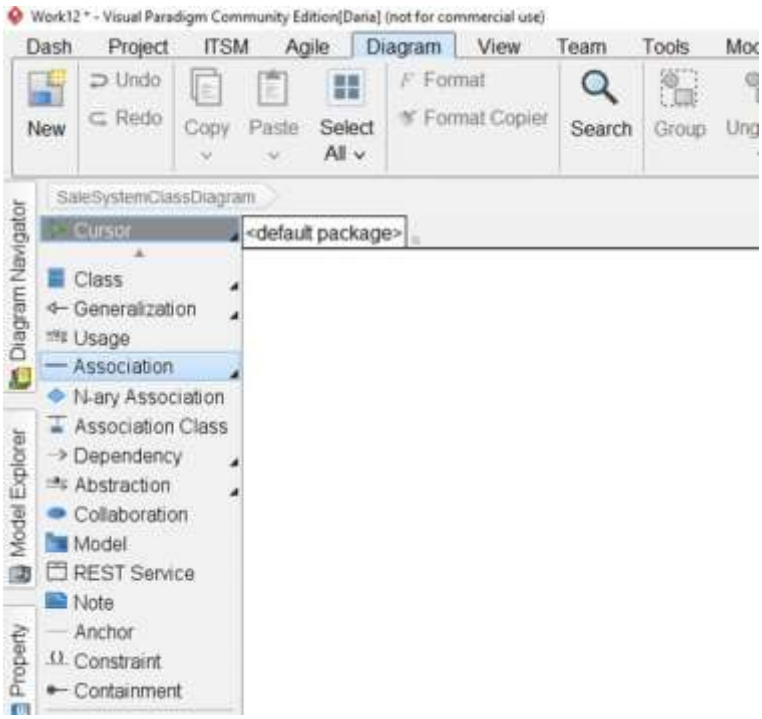

Рисунок 11 – Рабочая область для построения диаграммы классовДалее будут описаны принципы построения диаграммы классов.

Для создания класса необходима на панели инструментов выбрать Class и затем курсором мыши выбратьместо расположения на диаграмме (Рисунок 12). После создания класса необходимо задать его имя (Рисунок 13).

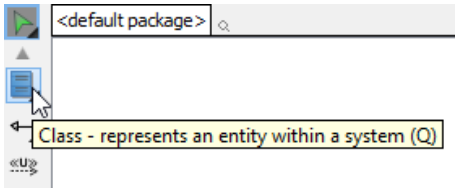

Рисунок 12 – Создание класса

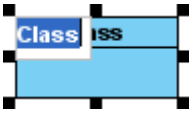

Рисунок 13 – Задание имени в созданном классе

Для создания ассоциативной связи между классами выберите **Association** > **Class (**Рисунок 14**).**

Перетащите курсор мыши на пустое место на диаграмме для создания нового или на уже существующий классдля соединения. Отпустите кнопку мыши (Рисунок 15)**.**

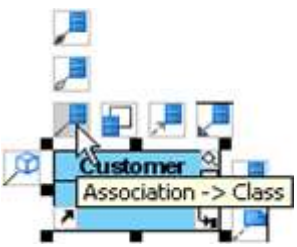

Рисунок 14 – Выбор типа связи

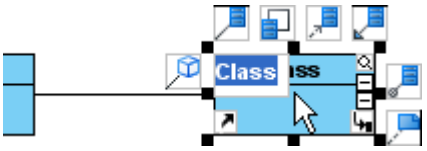

Рисунок 15 – Создание ассоциативной связи

Для создания связи агрегации между классами выберите **Aggregation** -> **Class (**Рисунок 16**).**

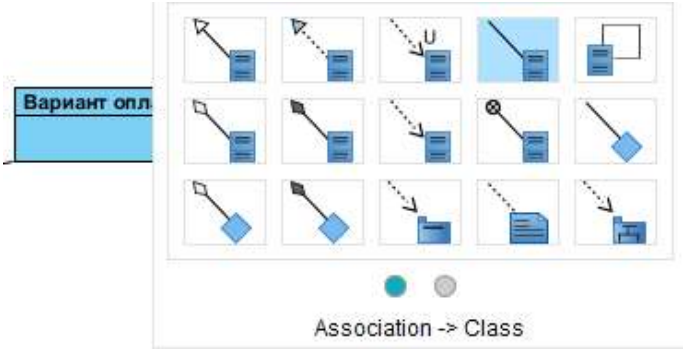

Рисунок 16 – Создание связи агрегации

Чтобы изменить кратность ассоциативной связи, щелкните правой кнопкой мыши на конце ассоциации, выберите **Multiplicity** из всплывающего меню, а затем выберите кратность (Рисунок 17). Либо выделите связи и нажмите **Enter** (рисунок 18).

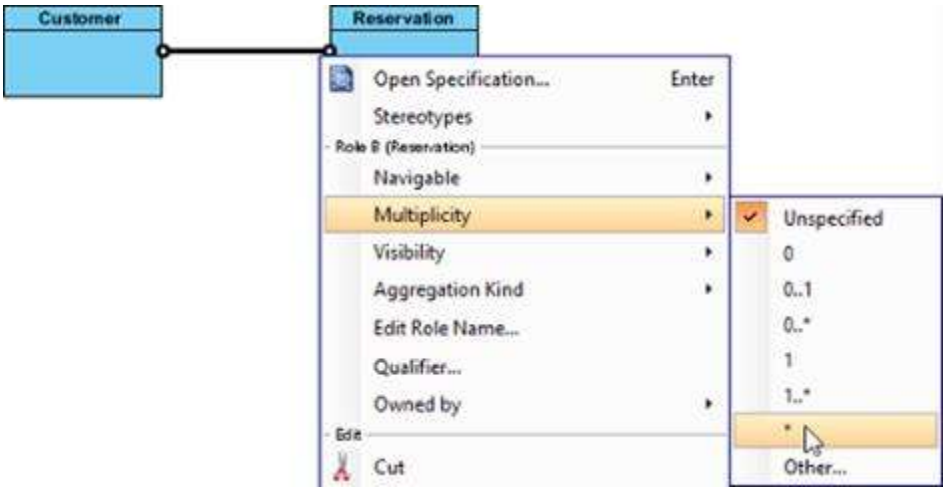

Рисунок 17 – Изменение кратности связи

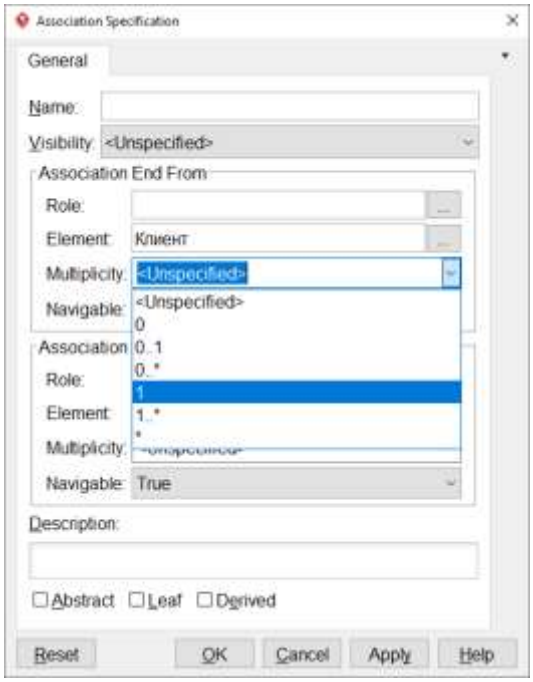

Рисунок 18 – Изменение кратности связи

Чтобы показать направление ассоциации, щелкните правой кнопкой мыши на связи и выберите **PresentationOptions** > **Show Direction** из всплывающего меню (Рисунок 19).

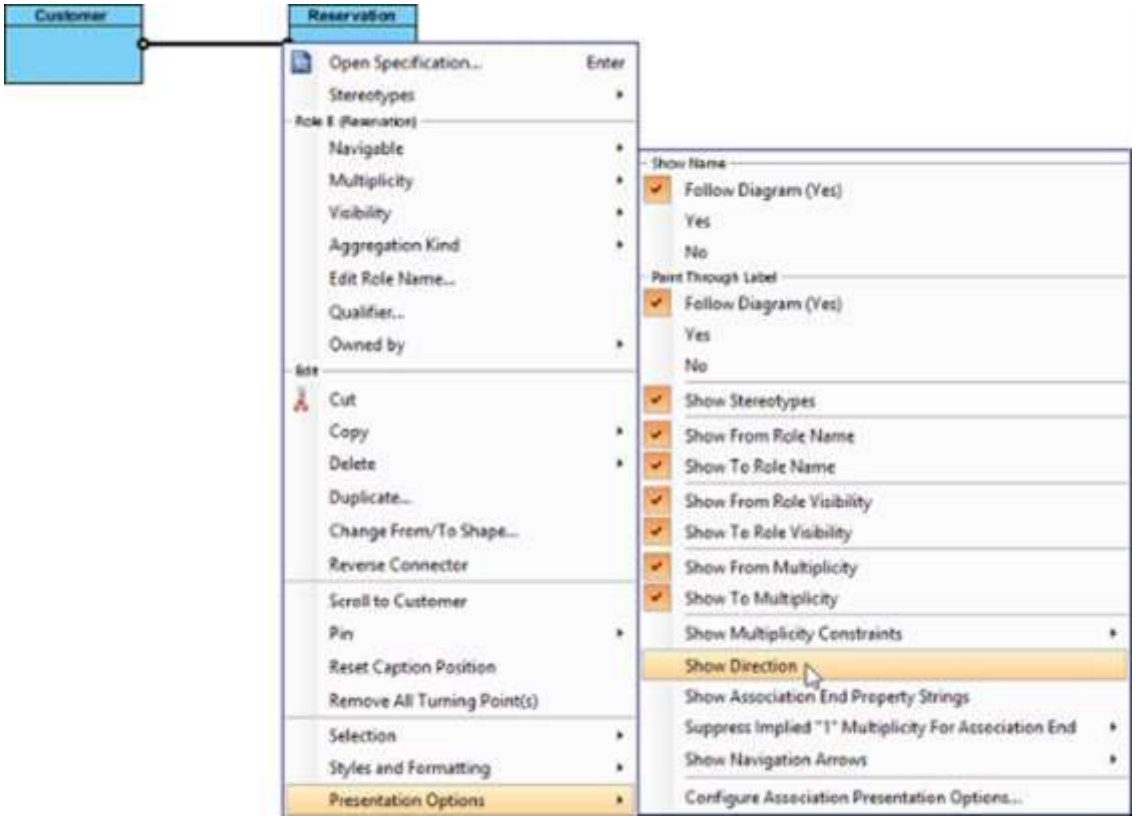

Рисунок 19 – Определение направления связи

Направление стрелки покажет связанный элемент диаграммы (Рисунок 20).

Рисунок 20 – Результат определения ассоциации

Для создания связи обобщения между классами выберите **Generalization -> Class (**Рисунок 21**).**

Перетащите курсор мыши на пустое место на диаграмме для создания нового или на уже существующий классдля соединения. Отпустите кнопку мыши (Рисунок 22)**.**

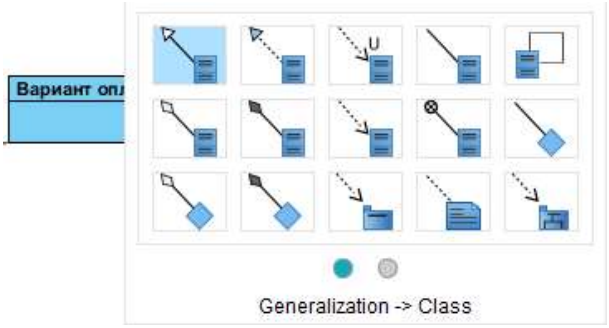

Рисунок 21 – Выбор связи

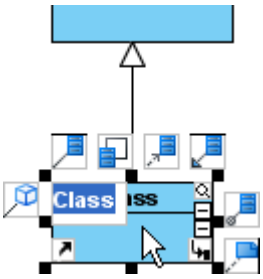

Рисунок 22 – Создание связи обобщения

Для создания атрибута щелкните правой кнопкой мыши на классе и выберете **Add** > **Attribute** в выпадающемменю (Рисунки 23-24).

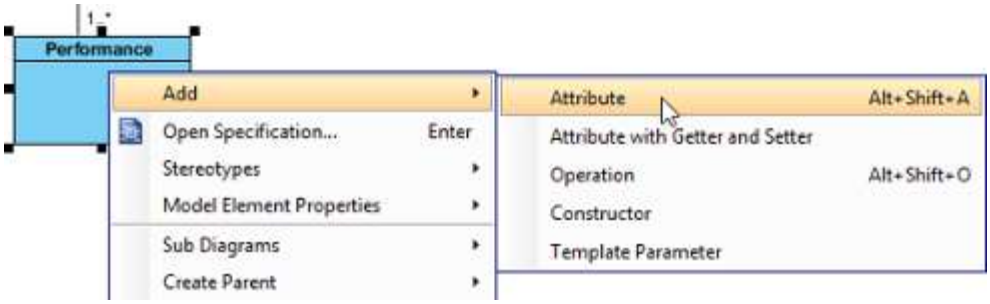

Рисунок 23 – Создание атрибута

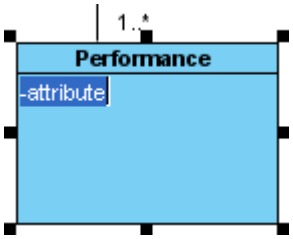

Рисунок 24 – Класс с созданным атрибутом

Для ускорения процесса создания атрибутов нужно сразу же после созданного атрибута на жать на клавиатуреклавишу **Enter**.

Для добавления типа данных атрибута выделите его и нажмите **Enter** либо щелкните правой кнопкой мыши ивыберите **Open Specification…** в окне свойств выберите тип данных (рис. 25).

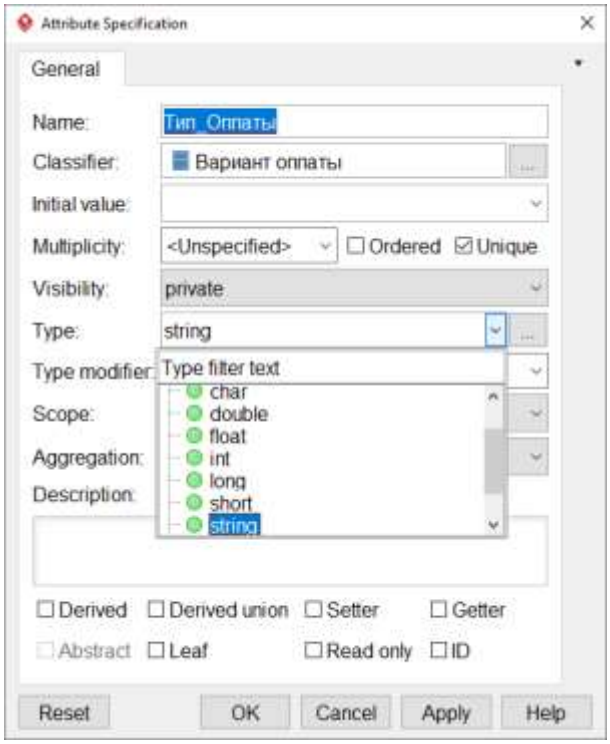

Рисунок 25 – Задание типа данных атрибута

Для создания операции щелкните правой кнопкой мыши на классе и выберете **Add** > **Operation** в выпадающемменю (Рисунки 26-27).

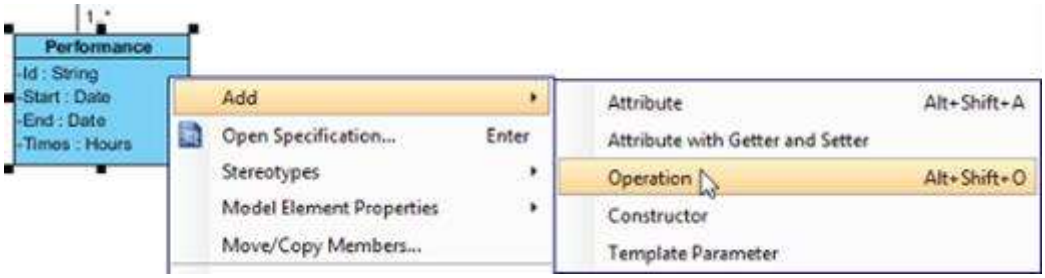

Рисунок 26 – Создание операции

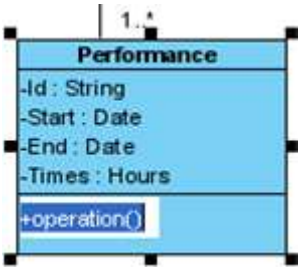

Рисунок 27 – Созданная операция

Для того чтобы переупорядочить члены класса необходимо зацепить курсором мышки один из членов иперетащить (Рисунки 28-29)

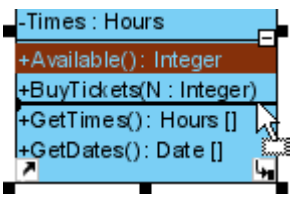

Рисунок 28 – Перетаскивание члена класса

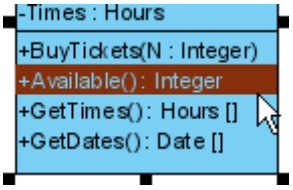

Рисунок 29 – Результат перетаскивания

Для копирования члена из одного класса в другой необходимо выбрать курсором мыши этот член класс сзажатой клавишей **Ctrl** и указать другой класс (Рисунки 30-31).

Рисунок 30 – Копирование члена класса

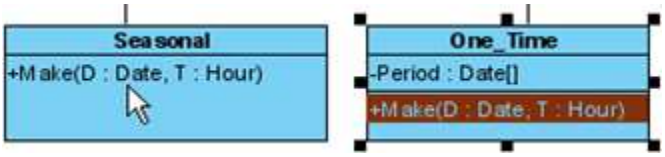

Рисунок 31 – Результат копирования члена класса

Перетаскивание члена класса из одного класса в другой производится подобным образом что и копирование,за исключением того, что не нужно зажимать клавишу **Ctrl**.

Для того чтобы выбрать все члены класса необходимо выбрать один член класса и нажать сочетание клавиш Ctrl+A**.**

Связывание классов производится по следующему алгоритму.

1. Выбрать тип связи из панели инструментов (Рисунок 32).

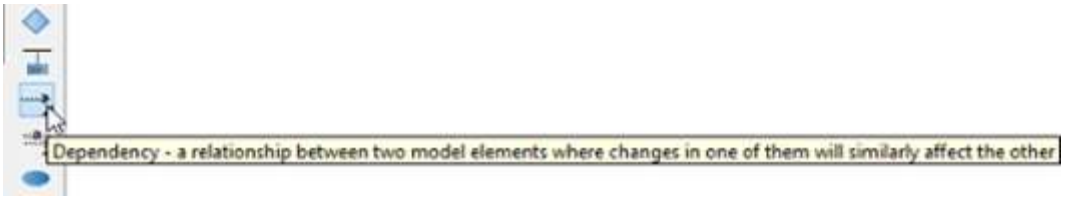

Рисунок 32 – Выбор зависимости

2. Переместить курсор мыши на атрибут-источник исходного класса (Рисунок 33).

Рисунок 33 – Определение связи по члену класса

3. Зажать клавишу мыши и не отпускать ее.
4. Переместить курсор мыши на член другого класса для связи (Рисунок 34).

Рисунок 34 – Определение члена класса в целевом классе

5. Отпустить клавишу мыши для определения соединения. В спецификации можно убедиться, что связьопределена именно по указанным атрибутам (Рисунок 35).

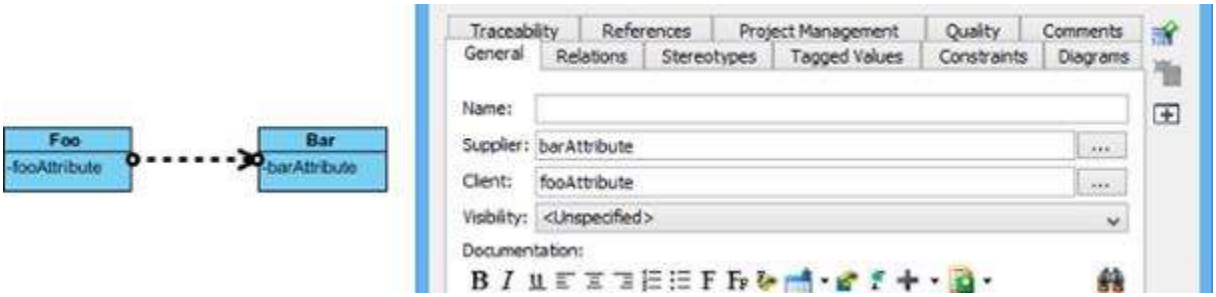

Рисунок 35 – Созданная связь

Основываясь, на описанных выше принципах построения диаграммы классов, постройте диаграмму классовдля системы продажи товаров по каталогу в соответствии с рисунком 36.

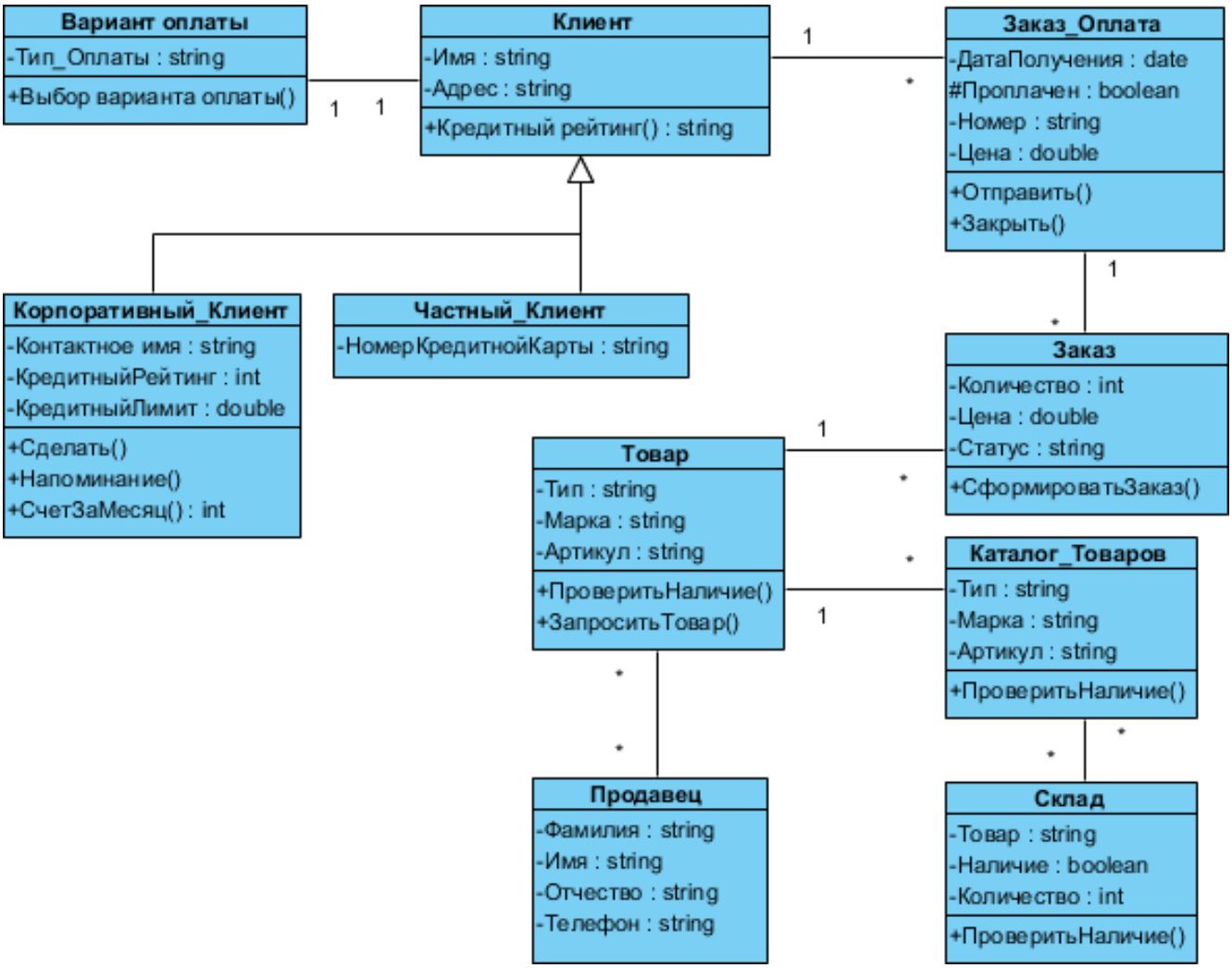

Рисунок 36 – Фрагмент диаграммы классов, описывающей реализацию систем продаж товаров по каталогу

Задание

Построить диаграмму классов в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит

пошаговое описаниепроцесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. 10.«Таксопарк».

Контрольные вопросы

1. Дайте определение понятию «диаграмма классов» и ее назначение.
2. Дайте определение следующим понятиям «класс», «атрибут», «операция», «отношение».
3. Опишите отношение ассоциаций между экземплярами классов.
4. Опишите отношение обобщения между экземплярами классов.
5. Опишите отношение агрегации между экземплярами классов.
6. Опишите отношение композиции между экземплярами классов.
7. Приведите пример графического представления основных компонентов диаграммы классов.
8. Дайте определение «квантор видимости» и его классификацию.