МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра Сетевых информационных технологий и сервисов (СИТиС)

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «Методы и средства проектирования

информационных систем и технологий»

Москва 2024

Оглавление

[Лабораторная работа №1 5](#_heading=h.gjdgxs)

[Выполнение лабораторной работы 5](#_heading=h.30j0zll)

[Контрольные вопросы 8](#_heading=h.1fob9te)

[Лабораторная работа №2 9](#_heading=h.3znysh7)

[Введение 9](#_heading=h.1t3h5sf)

[*Вариант использования* 10](#_heading=h.2s8eyo1)

[*Актеры* 10](#_heading=h.3rdcrjn)

[*Интерфейсы* 11](#_heading=h.lnxbz9)

[*Примечания* 12](#_heading=h.1ksv4uv)

[*Отношения* 12](#_heading=h.z337ya)

[*Отношение расширения* 13](#_heading=h.1ci93xb)

[*Отношение обобщения* 14](#_heading=h.2bn6wsx)

[*Отношение включения* 15](#_heading=h.3as4poj)

[*Типичные примеры применения* 15](#_heading=h.2p2csry)

[*Контекст системы* 16](#_heading=h.3o7alnk)

[*Требования к системе* 17](#_heading=h.ihv636)

[Ход работы 19](#_heading=h.2grqrue)

[*Обозначение системы* 20](#_heading=h.1v1yuxt)

[*Обозначение актёра* 21](#_heading=h.2u6wntf)

[*Обозначение варианта использования* 21](#_heading=h.3tbugp1)

[*Обозначение отношения «расширение»* 21](#_heading=h.nmf14n)

[*Обозначение отношения «включает»* 22](#_heading=h.1mrcu09)

[*Пример построения диаграммы прецедентов* 22](#_heading=h.2lwamvv)

[*Выделение прецедентов* 23](#_heading=h.4k668n3)

[*Описание прецедентов* 25](#_heading=h.2dlolyb)

[*Расширения (или альтернативные потоки)* 26](#_heading=h.4bvk7pj)

[*Построение диаграммы прецедентов* 26](#_heading=h.1664s55)

[Контрольные вопросы 27](#_heading=h.3q5sasy)

[Содержание отчета 29](#_heading=h.25b2l0r)

[Лабораторная работа №3 30](#_heading=h.1jlao46)

[Создание диаграммы классов 30](#_heading=h.xvir7l)

[*Создание класса* 30](#_heading=h.1x0gk37)

[*Создание атрибутов* 31](#_heading=h.4h042r0)

[*Создание отношения* 31](#_heading=h.1baon6m)

[*Создание отношения агрегация* 32](#_heading=h.pkwqa1)

[*Создание отношения обобщение* 33](#_heading=h.1opuj5n)

[*Создание отношения реализация* 35](#_heading=h.1302m92)

[Контрольные вопросы 36](#_heading=h.3mzq4wv)

[Лабораторная работа №4-5 37](#_heading=h.haapch)

[Создание диаграммы последовательности 37](#_heading=h.40ew0vw)

[*Создание актёра* 39](#_heading=h.upglbi)

[*Создание сообщений* 40](#_heading=h.1tuee74)

[*Создание фреймов* 42](#_heading=h.2szc72q)

[Контрольные вопросы 46](#_heading=h.3s49zyc)

[Содержание отчета 48](#_heading=h.279ka65)

[Лабораторная работа №7 49](#_heading=h.meukdy)

[Теоретическая часть 49](#_heading=h.36ei31r)

[Агрегация 50](#_heading=h.1ljsd9k)

[Ассоциация 51](#_heading=h.45jfvxd)

[Компонент 52](#_heading=h.2koq656)

[Композиция 54](#_heading=h.zu0gcz)

[Ограничение 55](#_heading=h.3jtnz0s)

[Зависимость 56](#_heading=h.1yyy98l)

[Обобщение 57](#_heading=h.4iylrwe)

[Интерфейс 58](#_heading=h.2y3w247)

[Связь 58](#_heading=h.1d96cc0)

[Примечание 59](#_heading=h.3x8tuzt)

[Реализация 59](#_heading=h.2ce457m)

[Практическая часть 60](#_heading=h.rjefff)

[Содержание отчета 63](#_heading=h.3bj1y38)

[Лабораторная работа №8 64](#_heading=h.1qoc8b1)

[Теоретическая часть 67](#_heading=h.2pta16n)

[*Доступ* 67](#_heading=h.3oy7u29)

[*Импорт* 67](#_heading=h.j8sehv)

[*Объединение* 67](#_heading=h.1idq7dh)

[*Пакет* 68](#_heading=h.2hio093)

[*Подсистема* 68](#_heading=h.3gnlt4p)

[Практическая часть 69](#_heading=h.4fsjm0b)

[Содержание отчета 72](#_heading=h.1a346fx)

[Приложение 1. Титульный лист 73](#_heading=h.3u2rp3q)

# Лабораторная работа №1

Цель работы: знакомство с возможностями среды моделирования Visual Paradigm. Приобретение практических знаний в процессе моделирования.

## Выполнение лабораторной работы

Для знакомства со средой Visual Paradigm, надо её установить на персональный компьютер (ПК). Для этого необходимо посетить официальный сайт компании разработчика <https://www.visual-paradigm.com>.

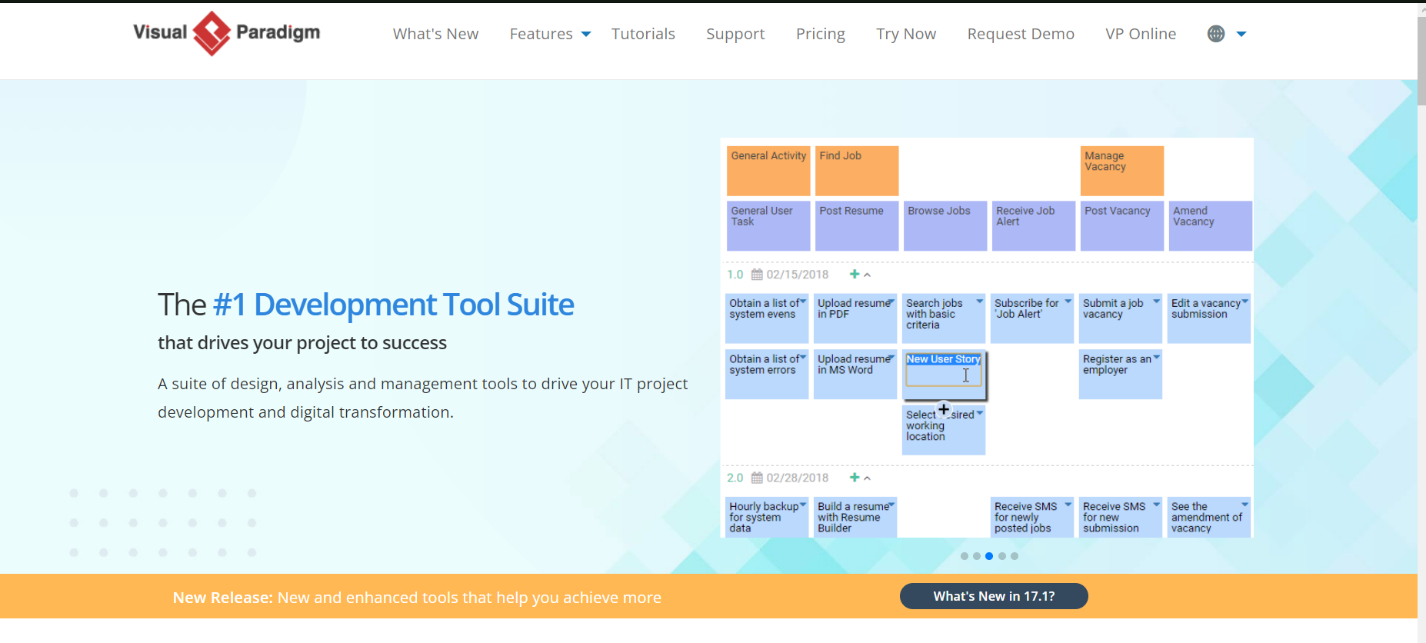


Рисунок 1 - Главная страница официального сайта разработчика

Перейдя на страницу загрузок «Try now» (ссылка находится вверху сайта), вам надо спуститься ниже и нажать кнопку «Get Community Edition (FREE for non-commercial use)». После этого нажать на кнопку «Download Visual Paradigm Community Edition». Если у вас другая операционная система, то надо выбрать пункт «More Options», и вам будет предложен полный список поддерживаемых операционных систем, в котором можно выбрать нужную.

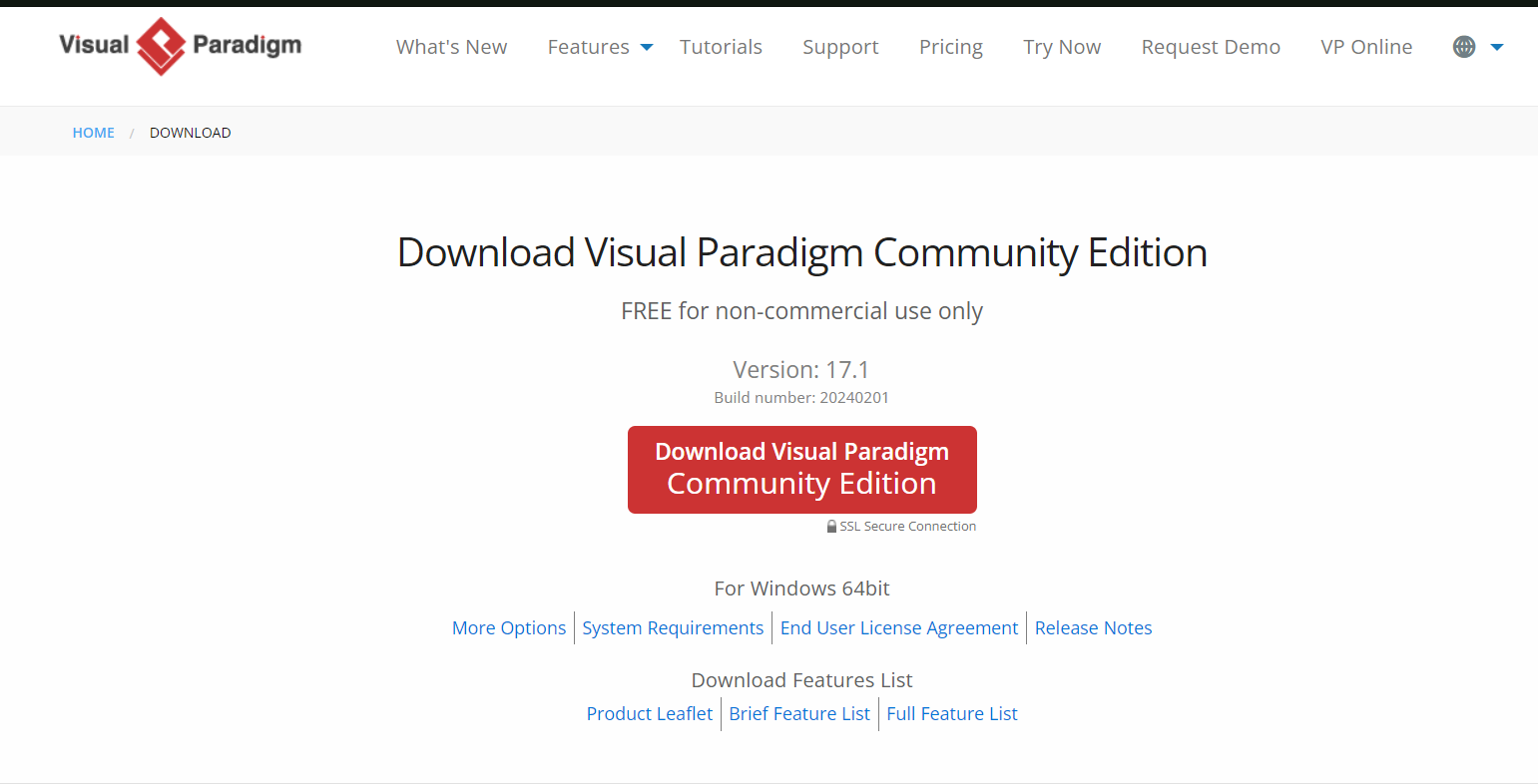


Рисунок 2 - Страница загрузок

После того, как дистрибутив был скачан, его требуется установить. Если же был выбран вариант, который не требует установки, то необходимо просто распаковать файлы в любую удобную для пользователя папку. Далее пройдите по следующему пути «\Visual Paradigm for UML 13.0\bin» и запустите файл «Visual Paradigm for UML.exe». Для удобства можно создать ярлык на этот файл. Ярлык будет расположен на «Рабочем столе». После запуска появится главное окно программы.

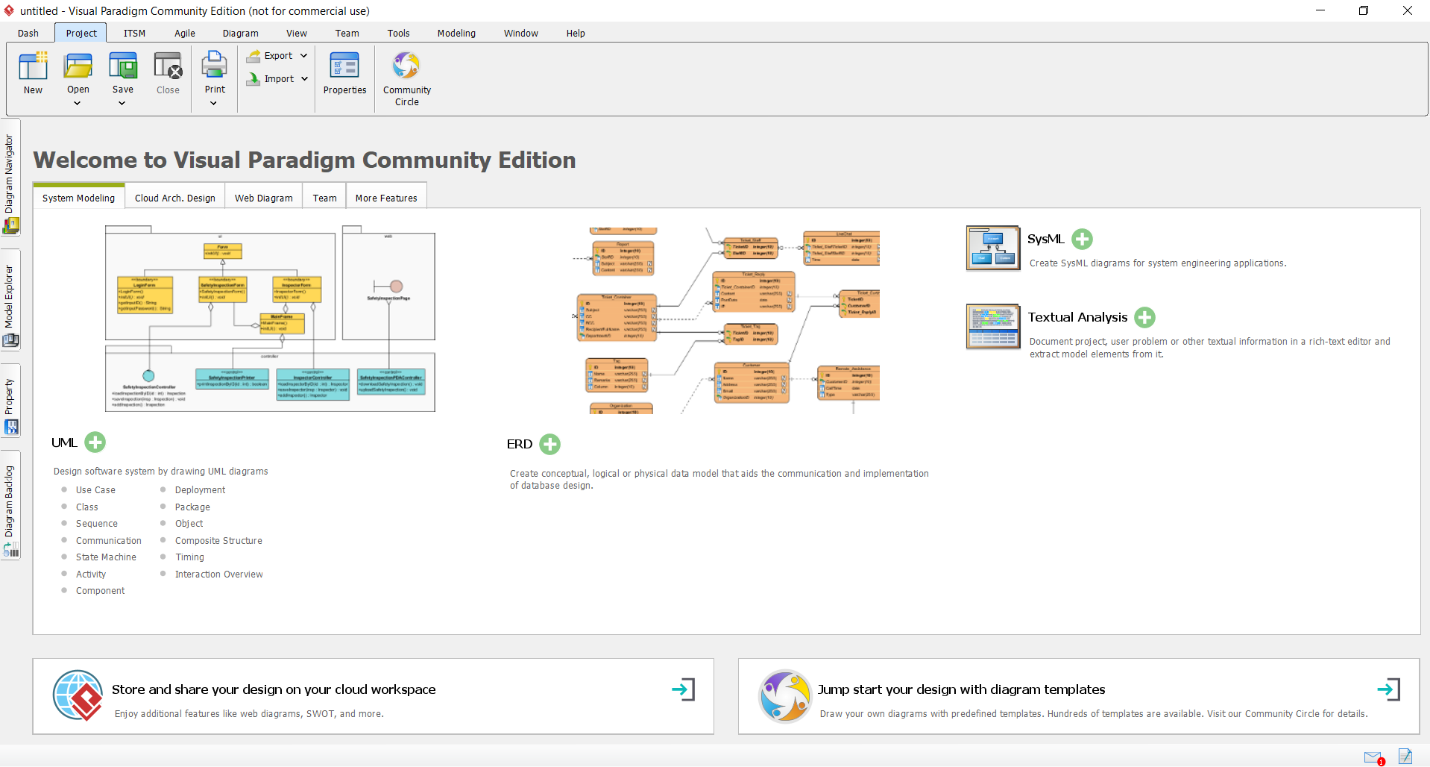


Рисунок 4 - Главное окно средства моделирования Visual Paradigm

Среда моделирования имеет дружественный интерфейс. Все элементы оформленный в виде панелей, которые можно переносить в удобное для пользователя место на экране, а также прикреплять к той или иной части области экрана.

Слева находится основная панель, которая называется «Навигатор по диаграммам». Все диаграммы делятся на 5 категорий:

1. UML – диаграммы;

2. Моделирование требований;

3. Моделирование баз данных;

4. Моделирование бизнес-процессов;

5. Остальные.

## Контрольные вопросы

1. Перечислить основные элементы управления среды разработки Visual Paradigm

2. Перечислить основные виды диаграмм UML;

3. Обозначить возможные области использования языка UML.

Содержание отчета Отчёт должен содержать:

1. Титульный лист (см. Приложение 1);

2. Цель работы;

3. Ход выполнения работы – скриншоты установки программы;

4. Ответы на контрольные вопросы.

# Лабораторная работа №2

Цель работы: знакомство с основными составляющими диаграммы прецедентов, методами построения и обозначениями соответствующих элементов диаграммы.

Введение

Диаграмма прецедентов играет основную роль в моделировании поведения системы, подсистемы или моделировании класса. Каждая диаграмма показывает множество прецедентов, актёров и отношений между ними.

Диаграммы прецедентов применяются для моделирования вида системы с точки зрения прецедентов (или вариантов использования). Чаще всего это предполагает моделирование контекста системы, подсистемы или класса либо моделирование требований, предъявляемых к поведению указанных элементов.

В языке UML диаграммы прецедентов позволяют визуализировать поведение системы, подсистемы или класса, чтобы пользователи могли понять, как их использовать, а разработчики – реализовать соответствующий элемент. На рис.1 приводится диаграмма, описывающая использование устройства – сотового телефона.

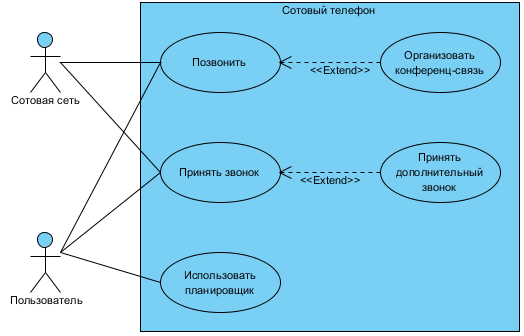


Рисунок 5 – Диаграмма прецедентов Диаграмма прецедентов обычно включает в себя:

* 1. Прецеденты – описание отдельного аспекта поведения системы;
  2. Актёры – сущности, взаимодействующие с системой в процессе её работы;
  3. Отношения – зависимости, обобщения и ассоциации.

## *Вариант использования*

Отдельный вариант использования (use-case) обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия её внутренней структуры. В качестве такой сущности может выступать система или любой элемент модели, который обладает собственным поведением.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемая сущность по запросу актера, то есть определяет способ применения этой сущности. Сервис, который инициализируется по запросу актера, представляет собой законченную неделимую последовательность действий. Это означает, что после того как система закончит обработку запроса, она должна возвратиться в исходное состояние, чтобы быть готовой к выполнению следующих запросов.

Варианты использования могут применяться как для спецификации внешних требований к проектируемой системе, так и для спецификации функционального поведения уже существующей системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы. Кроме этого, варианты использования неявно устанавливают требования, определяющие, как актеры должны взаимодействовать с системой, чтобы иметь возможность корректно работать с предоставляемыми сервисами. Для удобства множество вариантов использования может рассматриваться как отдельный пакет.

Примерами вариантов использования могут являться следующие действия: проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение дополнительной информации о кредитоспособности клиента, отображение графической формы на экране монитора и другие действия.

## *Актеры*

Актер (actor) представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка человечка, под которой записывается имя актера.

В некоторых случаях актер может обозначаться в виде прямоугольника класса с ключевым словом «актер» и обычными составляющими элементами

класса. Имена актеров должны записываться заглавными буквами и следовать рекомендациям использования имен для типов и классов модели.

Примерами актеров могут быть: клиент банка, банковский служащий, продавец магазина, менеджер отдела продаж, пассажир авиарейса, водитель автомобиля, администратор гостиницы, сотовый телефон и другие сущности, имеющие отношение к концептуальной модели соответствующей предметной области.

Так как в общем случае актер всегда находится вне системы, его внутренняя структура никак не определяется. Для актера имеет значение только то, как он воспринимается со стороны системы.

Актеры взаимодействуют с системой посредством обмена сообщениями с вариантами использования. Сообщение представляет собой запрос актером определенного сервиса системы и получение этого сервиса. Это взаимодействие может быть выражено посредством ассоциаций между отдельными актерами и вариантами использования или классами. Кроме этого, с актерами могут быть связаны интерфейсы, которые определяют, каким образом другие элементы модели взаимодействуют с этими актерами.

Два и более актера могут иметь общие свойства, то есть взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.

## *Интерфейсы*

Интерфейс (interface) служит для спецификации параметров модели, которые видимы извне, без указания их внутренней структуры. В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов для актеров.

На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя. В качестве имени может быть существительное или строка текста. Если имя записывается на английском языке, то оно должно начинаться с заглавной буквы I.

Графический символ отдельного интерфейса соединяется на диаграмме сплошной линией или пунктирной линией со стрелкой с тем вариантом использования, который его поддерживает. Сплошная линия указывает, что связанный с интерфейсом вариант использования должен реализовывать все необходимые для него сервисы. Пунктирная линия со стрелкой означает, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Таким образом, интерфейс отделяет спецификацию операций системы от их реализации и определяет общие границы проектируемой системы.

## *Примечания*

Примечания (notes) в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Графически примечания обозначаются прямоугольником с загнутым верхним правым углом. Внутри прямоугольника содержится текст примечания.Если в примечании указывается ключевое слово «constraint», то оно является ограничением, налагаемым на соответствующий элемент модели.

## *Отношения*

Между элементами диаграммы вариантов использования могут существовать различные отношения (relationship), которые описывают взаимодействие экземпляров актеров и вариантов использования.

В языке UML существует несколько стандартных видов отношений между актерами и вариантами использования:

* ассоциации (association relationship);
* расширения (extend relationship);
* обобщения (generalization relationship);
* включения (include relationship).

*Отношение ассоциации*

Применительно к диаграммам вариантов использования ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы, то есть, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь условные обозначения, такие как имя и кратность.

Кратность (multiplicity) ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации, и характеризует количество экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и символа звездочка.

Для диаграмм вариантов использования наиболее распространенными являются четыре основные формы записи кратности отношения ассоциации:

* целое неотрицательное число (включая 0). Предназначено для указания кратности, которая является строго фиксированной для элемента соответствующей ассоциации. В этом случае количество экземпляров актеров или вариантов использования, которые могут выступать качестве элементов отношения ассоциации, в точности равно указанному числу;
* два целых неотрицательных числа, разделенные двумя точками. Данная запись соответствует нотации для множества или интервала целых чисел, который применяется в некоторых языках программирования для обозначения границ массива элементов. Эту запись следует понимать как множество целых неотрицательных чисел, следующих в последовательно возрастающем порядке;
* два символа, разделенные двумя точками. При этом первый из них является целым неотрицательным числом или 0, а второй - специальным символом «\*», который обозначает произвольное конечное целое неотрицательное число, значение которого неизвестно на момент задания соответствующего отношения ассоциации;
* единственный символ «\*», который является сокращением записи интервала «0..\*».

Если кратность отношения ассоциации не указана, то, по умолчанию, принимается значение равное 1.

## *Отношение расширения*

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров. В метамодели отношение расширения является направленным и указывает, что применительно к отдельным примерам некоторого варианта использования должны быть выполнены конкретные условия, определенные для расширения данного варианта использования. Так, если имеет место отношение расширения от варианта использования А к варианту использования В, то это означает, что свойства экземпляра варианта использования В могут быть дополнены благодаря наличию свойств у расширенного варианта использования А.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом «extend» (расширяет).

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования. Данное отношение включает в себя некоторое условие и ссылки на точки расширения в базовом варианте использования. Чтобы расширение имело место, должно быть выполнено определенное условие данного отношения. Ссылки на точки расширения определяют те места в базовом варианте использования, в которые должно быть помещено соответствующее расширение при выполнении условия.

Один вариант использования может быть расширением для нескольких базовых вариантов, а также иметь в качестве собственных расширений несколько других вариантов. Базовый вариант использования может дополнительно никак не зависеть от своих расширений.

Семантика отношения расширения определяется следующим образом. Если экземпляр варианта использования выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляр другого варианта использования, которая является первой из всех точек расширения исходного варианта, то проверяется условие данного отношения. Если условие выполняется, исходная последовательность действий расширяется посредством включения действий экземпляра другого варианта использования. Следует заметить, что условие отношения расширения проверяется лишь один раз при первой ссылке на точку расширения, и если оно выполняется, то все расширяющие варианты использования вставляются в базовый вариант.

## *Отношение обобщения*

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В. В этом случае вариант А будет являться специализацией варианта В. При этом, В называется предком или родителем по отношению А, а вариант А - потомком по отношению к варианту использования В. Потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования.

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом, дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Применительно к данному отношению, один вариант использования может иметь несколько родительских вариантов. В этом случае реализуется множественное наследование свойств и поведения отношения предков. С другой стороны, один вариант использования может быть предком для нескольких дочерних вариантов, что соответствует таксономическому характеру отношения обобщения.

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Например, отношение обобщения от актера А к актеру В отмечает тот факт, что каждый экземпляр актера А является одновременно экземпляром актера В и обладает всеми его свойствами. В этом случае актер В является родителем по отношению к актеру А, а актер А потомком актера В. При этом актер А обладает способностью играть такое же множество ролей, что и актер В. Графически данное отношение также обозначается стрелкой обобщения.

## *Отношение включения*

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования.

Семантика этого отношения определяется следующим образом. Когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, экземпляр первого варианта использования выполняет последовательность действий, определяющую поведение экземпляра второго варианта использования, после чего продолжает выполнение действий своего поведения. При этом предполагается, что даже если экземпляр первого варианта использования может иметь несколько включаемых в себя экземпляров других вариантов, выполняемые ими действия должны закончиться к некоторому моменту, после которого должно быть продолжено выполнение прерванных действий экземпляра первого варианта использования в соответствии с заданным для него поведением.

Один вариант использования может быть включен в несколько других вариантов, а также включать в себя другие варианты. Включаемый вариант использования может быть независимым от базового варианта в том смысле, что он предоставляет ему некоторое инкапсулированное поведение, детали реализации которого скрыты и могут быть перераспределены между несколькими включаемыми вариантами использования. Более того, базовый вариант может зависеть только от результатов выполнения включаемого в него поведения, но не от структуры включаемых в него вариантов.

Отношение включения, направленное от варианта использования А к варианту использования В, указывает, что каждый экземпляр варианта А включает в себя функциональные свойства, заданные для варианта В. Эти свойства специализируют поведение соответствующего варианта А на данной диаграмме. Графически данное отношение обозначается пунктирной линией со стрелкой, которая помечается ключевым словом «include» (включает).

## *Типичные примеры применения*

Диаграммы прецедентов, или использования, применяют для моделирования статического вида системы с точки зрения прецедентов. Этот вид охватывает главным образом поведение системы, то есть видимые извне сервисы, предоставляемые системой в контексте ее окружения.

При моделировании статического вида системы с точки зрения прецедентов диаграммы использования обычно применяются два способа:

1. для моделирования контекста системы. Моделирование контекста подразумевает, что мы обводим систему воображаемой линией и выявим актеров, которые находятся за этой линией и взаимодействуют с системой. Диаграммы прецедентов нужны на этом этапе для идентификации актеров и семантики их ролей;
2. для моделирования требований к системе. Моделирование требований к системе предполагает указание на то, что система должна делать (с точки зрения внешнего наблюдателя), независимо от того, как она должна это делать. Диаграммы прецедентов необходимы здесь для специфицирования желаемого поведения системы. Они позволяют рассматривать всю систему как "черный ящик": вы видите все, что находится вне ее, наблюдаете за ее реакцией на события, но ничего не знаете о ее внутреннем устройстве.

## *Контекст системы*

Любая система содержит внутри себя какие-либо сущности, в то время как другие сущности остаются за ее пределами. Например, в системе проверки кредитных карточек имеются счета, транзакции и механизмы проверки подлинности. В то же время обладатели кредитных карточек и торговые предприятия находятся вне системы. Сущности внутри системы отвечают за реализацию поведения, которого ожидают сущности, находящиеся снаружи. Сущности, находящиеся вне системы и взаимодействующие с ней, составляют ее контекст. Таким образом, контекстом называется окружение системы.

UML позволяет моделировать контекст с помощью диаграмм прецедентов, в которых внимание акцентируется на окружающих систему актерах. Важно правильно определить актеров, поскольку это позволяет описать класс сущностей, взаимодействующих с системой. Еще важнее определить, что не является актером, так как при этом ограничивается окружение системы: в нем остаются только те элементы, которые участвуют в ее работе.

Моделирование контекста системы состоит из следующих шагов:

1. Идентифицируйте окружающих систему актеров. Для этого нужно найти группы, которым требуется участие системы для выполнения их задач; группы, которые необходимы для осуществления системой своих функций; группы, взаимодействующие с внешними программными и аппаратными средствами, а также группы, выполняющие вспомогательные функции администрирования и поддержки;
2. Организуйте похожих актеров с помощью отношений обобщения/специализации;
3. Введите стереотипы для каждого актера, если это облегчает понимание;
4. Поместите актеров на диаграмму прецедентов и определите способы их связи с прецедентами системы.

Например, на рис.2 показан контекст системы, работающей с кредитными карточками, где основное внимание уделяется окружающим ее актерам. В первую очередь это Клиенты двух типов (Индивидуальный клиент и Корпоративный клиент), соответствующие ролям, которые играют люди при взаимодействии с системой. В этом контексте показаны и актеры, представляющие другие организации, такие как Торговые предприятия (с ними покупатели совершают карточные транзакции, приобретая вещи или услуги) и Субсидирующие финансовые институты (выполняющие роль клиринговой палаты для карточных счетов). В реальном мире последние два актера, скорее всего, сами будут программными системами.



Рисунок 6 – Моделирование контекста системы

Тот же метод позволяет моделировать и контекст подсистемы. Вообще, элемент, который на одном уровне абстракции выглядит как система, часто становится подсистемой на другом, более высоком уровне абстракции. Моделирование контекста подсистемы может пригодиться при построении системы из нескольких взаимосвязанных частей.

## *Требования к системе*

Требование (requirement) – это особенность проекта, свойство или поведение системы. Приступая к сбору требований, описываются условия контракта, заключаемого между системой и сущностями вне ее, в котором декларируется, что система должна делать. При этом, как правило, пользователя заботит не то, как именно система будет выполнять поставленные задачи, а только то, что она будет делать. Хорошо спроектированная система должна полностью выполнять все требования, причем делать это предсказуемо и надежно. Ее создание начинается с соглашения о том, каково ее назначение, хотя в ходе разработки понимание требований будет постепенно изменяться. Аналогично при работе с готовой системой понимание того, как она себя ведет, имеет принципиальное значение для ее правильного использования.

Требования можно выразить по-разному, от неструктурированного текста до выражений на формальном языке (или, например, с помощью примечаний). Большая часть функциональных требований к системе или даже все они могут быть выражены в виде прецедентов использования, в чем помогают диаграммы прецедентов UML.

Моделирование требований к системе производится следующим образом:

1. Установите контекст системы, идентифицировав окружающих ее актеров;
2. Для каждого актера рассмотрите поведение, которого он ожидает или требует от системы;
3. Поименуйте эти общие варианты поведения как прецеденты;
4. Выделите общее поведение в новые прецеденты, которые будут использоваться другими; выделите вариации поведения в новые прецеденты, расширяющие основные потоки событий;
5. Смоделируйте этих прецедентов, актеров и отношения между ними на диаграмме прецедентов;
6. Дополните прецеденты примечаниями, описывающими нефункциональные требования; некоторые из таких примечаний можно присоединить к системе в целом.

Рисунок 7 расширяет предыдущую диаграмму. Хотя отношения между актерами и прецедентами на ней опущены, но присутствуют дополнительные прецеденты, которые описывают важные, пусть и невидимые для обычного пользователя, элементы поведения системы. Эта диаграмма удобна, поскольку дает возможность пользователям, экспертам и разработчикам совместно визуализировать, специфицировать, конструировать и документировать свои решения относительно функциональных требований к системе. Например, прецедент Обнаружение фальшивых карточек описывает поведение, важное как для Торговых предприятий, так и для Субсидирующих финансовых институтов. Другой прецедент - Отчет о состоянии счета - также описывает поведение, требуемое от системы различными организациями в своих контекстах.

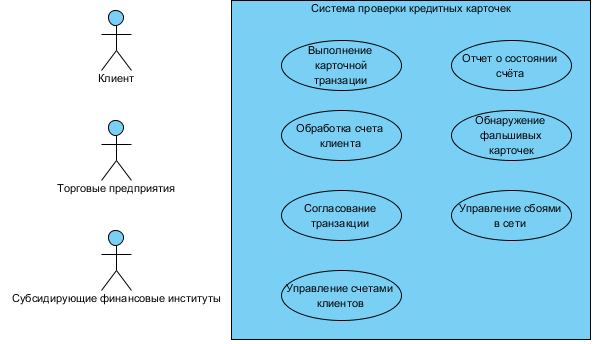


Рисунок 7 – Моделирование требований к системе

Создавая диаграммы прецедентов в UML, помните, что каждая из них является всего лишь графическим представлением статического вида системы с точки зрения прецедентов. Это означает, что ни одна диаграмма прецедентов, взятая в отдельности, не может, да и не должна охватывать весь этот вид целиком. В совокупности диаграммы прецедентов дают полное представление о виде системы с точки зрения прецедентов, а каждая из них в отдельности - только об одном из его аспектов.

Ход работы

*Создание диаграммы прецедентов в Visual Paradigm*

Для построения диаграммы прецедентов выполните следующие шаги:

* нажмите на UML на панели инструментов и выберите диаграммы прецедентов из выпадающего меню;
* Щелкните правой кнопкой мыши на диаграмму прецедентов в пункте «Навигатор диаграмм» и выберите новая диаграмма прецедентов во всплывающем меню;
* Выберите File → New Diagram → UML Diagrams → Use Case Diagram из главного меню.

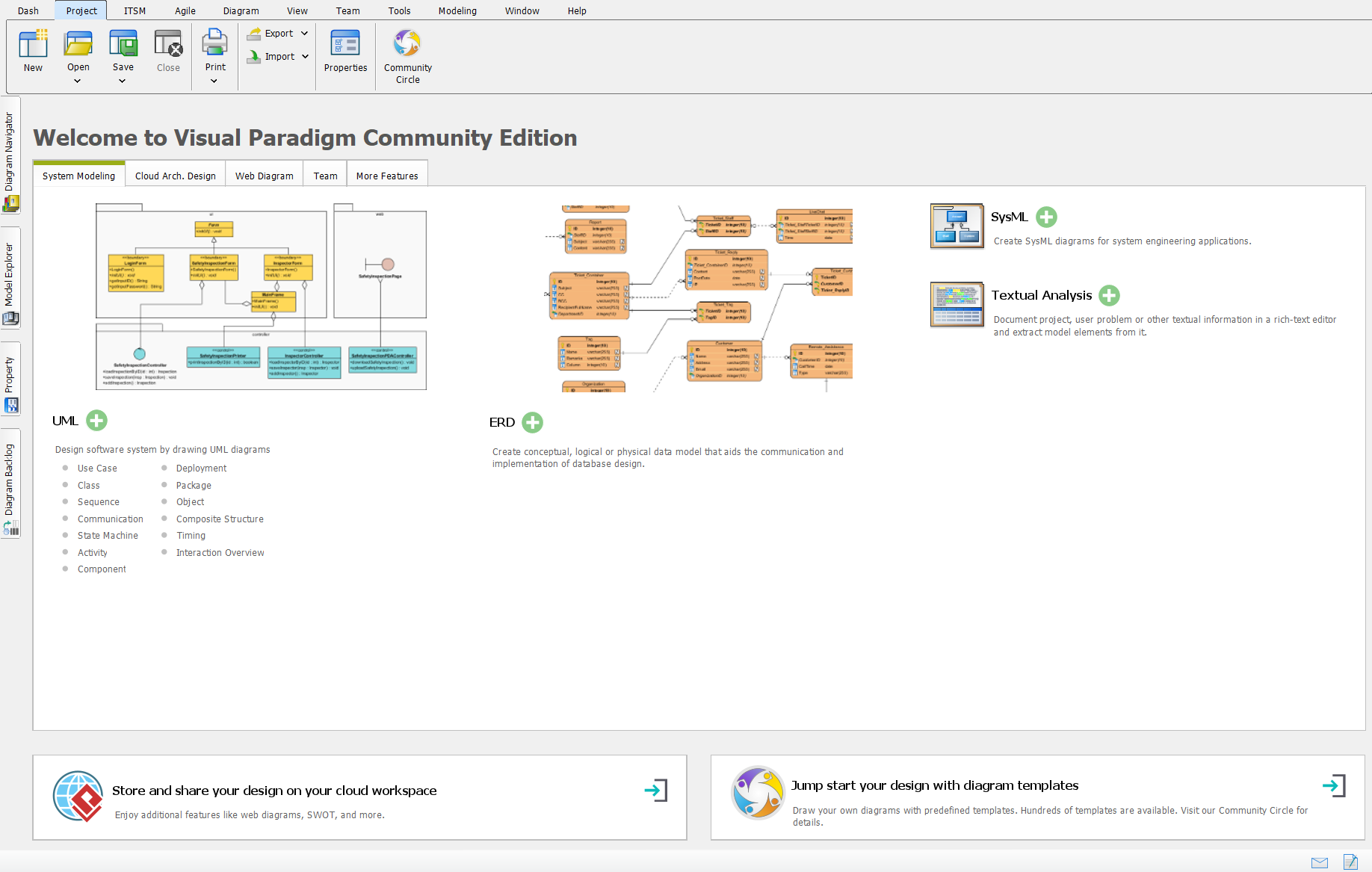


Рисунок 8 – Создание новой диаграммы прецедентов

В левом верхнем углу, в текстовом поле всплывающего окна, ведите имя для только что созданной диаграммы прецедентов.



Рисунок 9 – Поле для ввода названия диаграммы

## *Обозначение системы*

Для создания системы, выберите на панели инструментов Система, а затем щелкните в области построения диаграммы. Затем, дайте имя созданной системе.

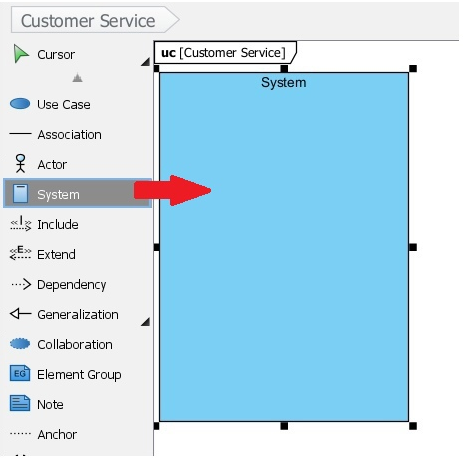


Рисунок 10 – Создание системы

*Обозначение актёра*

Чтобы обозначить актера, выберите Актер на панели инструментов диаграммы, а затем щелкните в области построения диаграммы. Затем, задайте имя актёру.

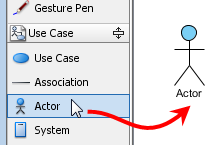


Рисунок 11 – Создание актёра

## *Обозначение варианта использования*

Помимо создания прецедента через панель инструментов диаграммы, также можно создавать его через значок ресурсов.

Наведите указатель мыши на фигуру и нажмите на значок ресурса, который может создать прецедент. Перетащите его, а затем отпустите кнопку мыши. Исходная фигура и вновь созданный прецедент станут связанными. Затем задайте имя только что созданному прецеденту.

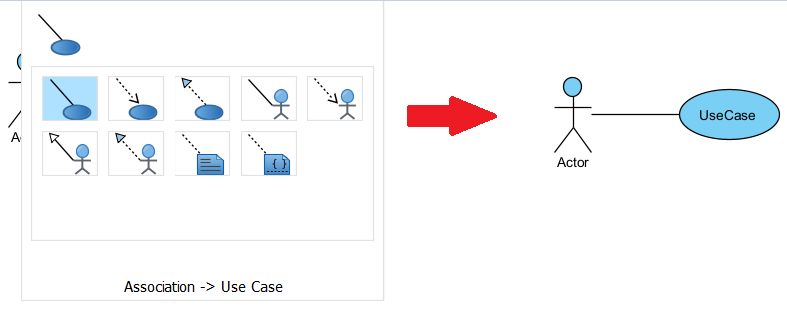


Рисунок 12 – Добавление прецедента с помощью иконки «ресурсы»

## *Обозначение отношения «расширение»*

Чтобы создать отношение «расширение», переместите указатель мыши на прецеденте и нажмите его значок ресурса Extend → Use Case. Перетащите его в нужное место, а затем отпустите кнопку мыши. Прецедент расширение и вновь созданный прецедент станут связаны. После того как будет назван новый прецедент, появится всплывающее диалоговое окно, которое спросит, хотите ли вы чтобы в новом прецеденте, отображалось имя прецедента расширения. Нажмите кнопку Да, если вы хотите, чтобы имя отображалось, нажмите НЕТ, если вы хотите ввести другое имя.

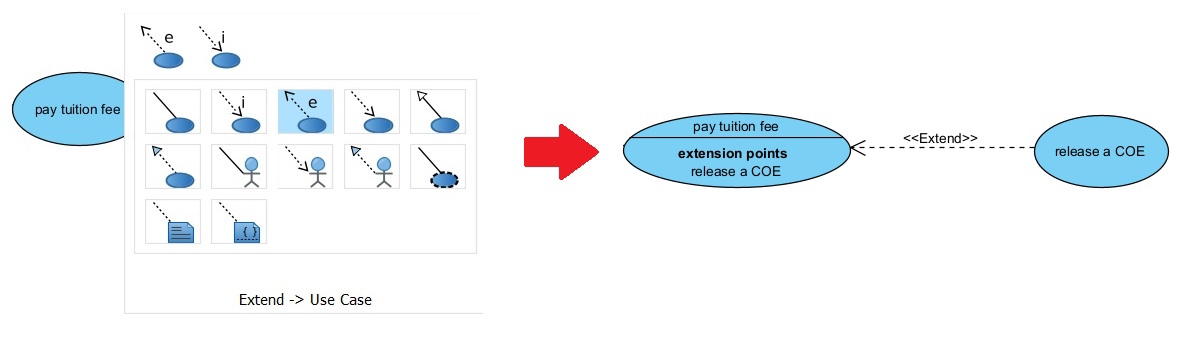


Рисунок 13 – Создание отношения расширения

*Обозначение отношения «включает»*

Чтобы создать отношение включения, наведите курсор на прецедента и нажмите его значок ресурса Include → Use Case. Перетащите его в нужное место, а затем отпустите кнопку мыши. Новый вариант использования вместе с отношением включения успешно создан. Далее, назовите созданный прецедент.

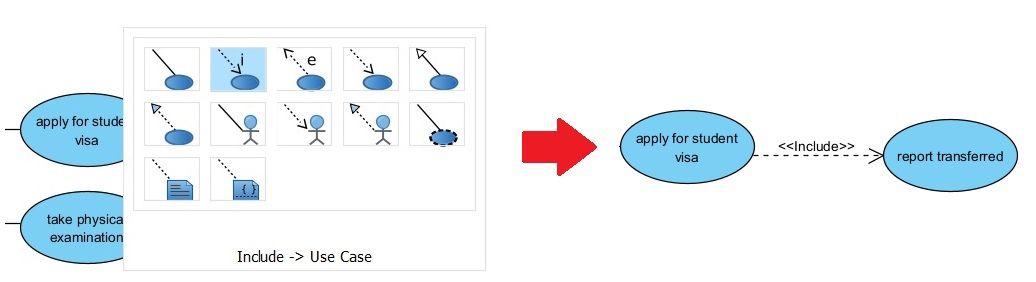


Рисунок 14 – Создание отношения включения

## *Пример построения диаграммы прецедентов*

*Предварительное описание*

Система обеспечивает автоматическую выдачу билетов с

использованием точки доступа и мобильных устройств, оснащенных модулем беспроводной связи. Система должна осуществлять проверку доступа студента к билетам только с использованием одного мобильного устройства. Для составления пар студент-устройство используются ФИО студента и MAC-адрес устройства. Преподаватель должен иметь информацию о том, какой студент вытянул какой вопрос, и время, в которое было произведено это действие. Доступ к экзаменационным билетам должен быть только у студентов группы, указанной преподавателем и допущенных к экзамену. Преподаватель должен иметь возможность допускать студентов до экзамена и разрешать сдавать экзамен студентам из другой группы в виде исключения. После регистрации студента и его мобильного устройства система выдает случайный, еще не занятый, билет, и при последующих обращениях с мобильного устройства выдает тот же самый билет.

В процессе проведения экзамена студент может вытянуть билет позволяющий получить оценку автоматически (без ответа на вопросы). Для этого в течение семестра использовалась система промежуточной оценки остаточных знаний, результатом работы которой являются 4 пары тема - оценка. Весь курс разбит на 10 тем, таким образом, чтобы ответив на любой вопрос из темы в течение семестра, можно было получить оценку за всю тему. Для вычисления автоматической оценки на основе выбранного билета система должна проверить в какие темы попадают вопросы, выбранного билета, и сопоставить их с оценками, полученными студентом, вытянувшим билет.

## *Выделение прецедентов*

Определение рамок системы

Для, того чтобы яснее очертить рамки проектируемой системы определим те функции, которые она не должна выполнять, т.е. определим внешних вспомогательных исполнителей:

1. Система не отвечает за процессы подключения и аутентификации мобильных устройств студентов, за это отвечают протоколы безопасности беспроводных сетей;
2. Система не отвечает за сопоставление конкретных МАС адресов и адресов мобильных устройств в сети IP, за это отвечают протоколы DHCP и ARP, реализованные в рамках операционной системы или роутера (точки доступа);
3. Система не отвечает за визуализацию содержимого экзаменационного билета, это выполняет браузер на мобильном устройстве.

Таким образом, внешними вспомогательными исполнителями являются: операционная система, беспроводной роутер, браузер мобильного устройства.

*Определение основных исполнителей и задач*

Для того чтобы не упустить некоторые не очевидные моменты в процессе определения основных исполнителей и задач полезно ответить на следующие вопросы:

1. Кто запускает и выключает систему?
2. Кто является системным администратором?
3. Кто осуществляет управление пользователями и безопасностью?
4. Относится ли время к числу исполнителей, другими словами, должна ли система выполнять какие-либо действия в ответ на события времени?
5. Существует ли процесс мониторинга, благодаря которому система перезапускается в случае сбоя?
6. Кто контролирует деятельность и производительность системы?
7. Как выполняется обновление программного обеспечения?
8. Кто анализирует журналы регистрации? Можно ли обеспечить удаленный доступ к ним?
9. Могут ли в качестве исполнителей выступать внешние программы или автоматические системы?
10. Кого следует уведомлять при ошибках или сбоях системы?

Составим перечень исполнителей и задач в виде таблицы:

|  |  |
| --- | --- |
| **Исполнитель** | **Задачи** |
| Студент | Регистрируется на экзамене Получает билет |
| Преподаватель | Включает и выключает систему  Уточняет участие студента в сдаче экзамена Анализирует информацию о вытянутых билетах Анализирует информацию о времени получения билетов |
| Ассистент (деканат) | Формирует списки студентов |
| Система промежуточной оценки знаний | Предоставляет информацию для выставления автоматической оценки за экзамен |

Стоит пояснить таблицу «Исполнитель-Задачи». Данная таблица представляет собой сопоставление исполнителей и их задач в рамках разрабатываемой системы. Исполнители включают студентов, преподавателей, ассистентов (деканат) и систему промежуточной оценки знаний. Каждому исполнителю соответствуют определенные задачи, которые они выполняют в контексте системы.

* **«Студент» -** основные задачи студента включают регистрацию на экзамен и получение билета для сдачи.
* **«Преподаватель»** - задачи преподавателя включают управление системой (включение и выключение), уточнение участия студентов в экзамене, а также анализ информации о вытянутых билетах и времени их получения.
* **«Ассистент (деканат)»** - основная задача ассистента заключается в формировании списков студентов, возможно, связанных с процессом регистрации на экзамен.
* **«Система промежуточной оценки знаний»** - задача исполнителя заключается в предоставлении информации, необходимой для выставления автоматической оценки за экзамен.

Как правило, каждой задаче пользователя соответствует один прецедент. Его имя должно начинаться с существительного, описывающего действие. Из таблицы выше, можно сделать вывод, что в разрабатываемой системе присутствует два основных исполнителя: Студент и Преподаватель. Поэтому в качестве прецедентов определим те, которые соответствуют задачам основных исполнителей. (Регистрация на экзамене, Получение билета, Допуск на экзамен, Вызов на собеседование, Собеседование на экзамене ).

Далее в таблице «Прецедент-Задачи» опишем для каждого прецедента задачи, которые он должен обрабатывать.

|  |  |
| --- | --- |
| **Прецедент** | **Задачи** |
| Регистрация на экзамене | Регистрируется на экзамен |
| Получение билета | Получает билет |
| Допуск на экзамен | Уточняет участие студента на экзамене |
| Вызов на собеседование | Вызов на собеседование |
| Собеседование на экзамене | Анализирует ответ на экзамене |

Данная таблица представляет собой соответствие между прецедентами, описывающими действия или события в рамках системы, и задачами, которые выполняются в контексте каждого прецедента. Каждый прецедент является своего рода шаблоном для определенного вида действий или задач, которые могут возникнуть в системе.

* **«Регистрация на экзамене»** - этот прецедент описывает процесс регистрации студентов на экзамен.
* **«Получение билета»** - в этом прецеденте студент получает билет для сдачи экзамена.
* **«Допуск на экзамен**» - этот прецедент описывает процесс уточнения участия студента в экзамене.
* **«Вызов на собеседование»** - в данном прецеденте преподаватель вызывает студента для сдачи экзамена.
* **«Собеседование на экзамене»** - этот прецедент описывает процесс анализа ответа студента на экзамене.

Каждый прецедент представляет определенный этап или событие в рамках работы системы, а соответствующие задачи являются конкретными шагами или действиями, которые выполняются для завершения каждого прецедента. Эта таблица помогает структурировать и организовать процессы в системе, обеспечивая понятное понимание последовательности действий и их связь с прецедентами.

Далее составим ранжировку прецедентов в виде таблицы.

В таблице “Ранжирования прецедентов” мы должны определить, какой прецедент имеет более высокий ранг. Для ранжирования прецедентов, сравним их по 3 параметрам: важность, объем и сложность.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Прецеденты** | **Важность** | **Объем** | **Сложность** | **Ранг** |
| Получение билета | 5 | 5 | 4 | 4,7 |
| Регистрация на экзамене | 3 | 4 | 5 | 4 |
| Допуск на экзамене | 4 | 3 | 3 | 3,3 |
| Вызов на собеседование | 4 | 2 | 1 | 2,3 |
| Собеседование на экзамене | 2 | 1 | 2 | 1,7 |

Данная таблица представляет собой систему оценки и ранжирования прецедентов, которая помогает определить приоритетность выполнения различных задач или действий в контексте заданной системы. Оценки прецедентов основаны на трех ключевых параметрах: важность, объем и сложность.

* **«Важность»** отражает степень значимости прецедента для системы или организации. Прецеденты, имеющие более высокую важность, считаются более критическими для успешного функционирования системы или достижения целей.
* **«Объем»** указывает на размер или масштаб прецедента, включая количество ресурсов, времени или участников, необходимых для его выполнения. Прецеденты с большим объемом могут требовать больших затрат ресурсов или времени для выполнения.
* **«Сложность»** отражает уровень сложности или трудоемкости выполнения прецедента. Прецеденты с более высокой сложностью могут требовать дополнительных усилий, навыков или ресурсов для успешного завершения.

Каждый прецедент оценивается по каждому из этих параметров с использованием шкалы оценок. Оценки важности, объема и сложности вычисляются как среднее арифметическое всех оценок, полученных от участников. Затем для каждого прецедента вычисляется *ранг*, который представляет собой среднее значение его оценок по указанным параметрам. Чем выше ранг, тем более приоритетным считается прецедент.

Таким образом, таблица "Ранжирования прецедентов" предоставляет систематизированный метод оценки и приоритизации прецедентов на основе их важности, объема и сложности, что облегчает принятие решений о порядке выполнения задач или действий в рамках данной системы или организации.

## *Описание прецедентов*

В качестве примера, рассмотрим развернутое описание прецедента

*Получение билета*.

Прецедент П1. Получение билета

*Рамки*. Система поддержки проведения экзамена.

*Уровень*. Задача, определенная пользователем.

*Основной исполнитель*. Студент. Заинтересованные лица и их требования.

* Студент. Хочет получить билет и узнать о возможности выставления автоматической оценки. Все это он хочет проделать без лишних волнений и не отвлекая остальных участников экзамена;
* Преподаватель. Хочет быстро определить, кому и какую оценку можно поставить автоматически;
* Деканат. Хочет получить аккуратно заполненные ведомости о проведении экзамена. *Предусловия*. Студент зарегистрировался на экзамене и имеет допуск.

*Результаты (Постусловия)*. Студенту предоставлен случайный и еще не занятый билет. Зафиксировано время получения билета. Определены автоматические оценки за каждый вопрос в полученном билете.

*Основной успешный сценарий (или основной процесс)*

* 1. Студент сообщает системе о своем желании получить билет;
  2. Система проверяет факт выдачи билета студенту во время его предыдущих обращений;
  3. Система случайным образом выбирает билет, который до этого ни разу не был выбран и делает пометку о том, что билет занят конкретным студентом;
  4. Система запоминает время начала подготовки студента;
  5. Система определяет номер темы, к которой относится вопрос, выбранного билета;
  6. Система определяет оценку, которую можно поставить автоматически за данный вопрос на основании информации полученной от системы промежуточной оценки знаний;
  7. Система повторяет пункты 5 и 6 для всех вопросов, выбранного билета;
  8. Система формирует билет в виде возможном для отображения и передает его на мобильное устройство;
  9. Студент получает на экране мобильного устройства все вопросы и автоматические оценки и начинает готовиться к ответу.

## *Расширения (или альтернативные потоки)*

2-4а. При повторном обращении студента к системе для получения билета:

1. Система определяет, какой билет был выдан студенту при его первом обращении.

4а. Если в системе не осталось ни одного билета, который еще ни разу не был выдан:

1. Система сообщает студенту о том, что необходимо подождать, пока билеты не освободятся;
2. Система сообщает преподавателю о том, что свободных билетов нет и конкретный студент не может начать подготовку к ответу;
3. Система завершает обслуживание студента.

*Специальные требования*

На 3 шаге основного сценария необходимо обеспечить, чтобы у разных студентов выполнивших одновременное обращение к системы были разные билеты

*Список технологий и типов данных*

Для возможности работы с более широким кругом различных устройств содержание билетов должно формироваться в виде html разметки.

## *Построение диаграммы прецедентов*

В качестве CASE-средства в данном описании будет использоваться Visual Paradigm. Диаграмма прецедентов может выглядеть так, как показано на рисунке 15.

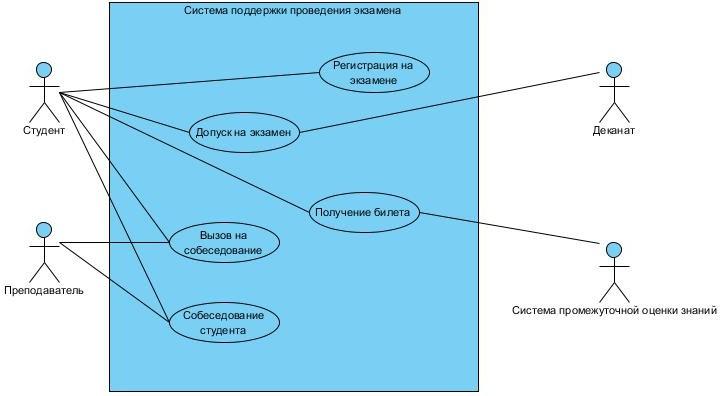
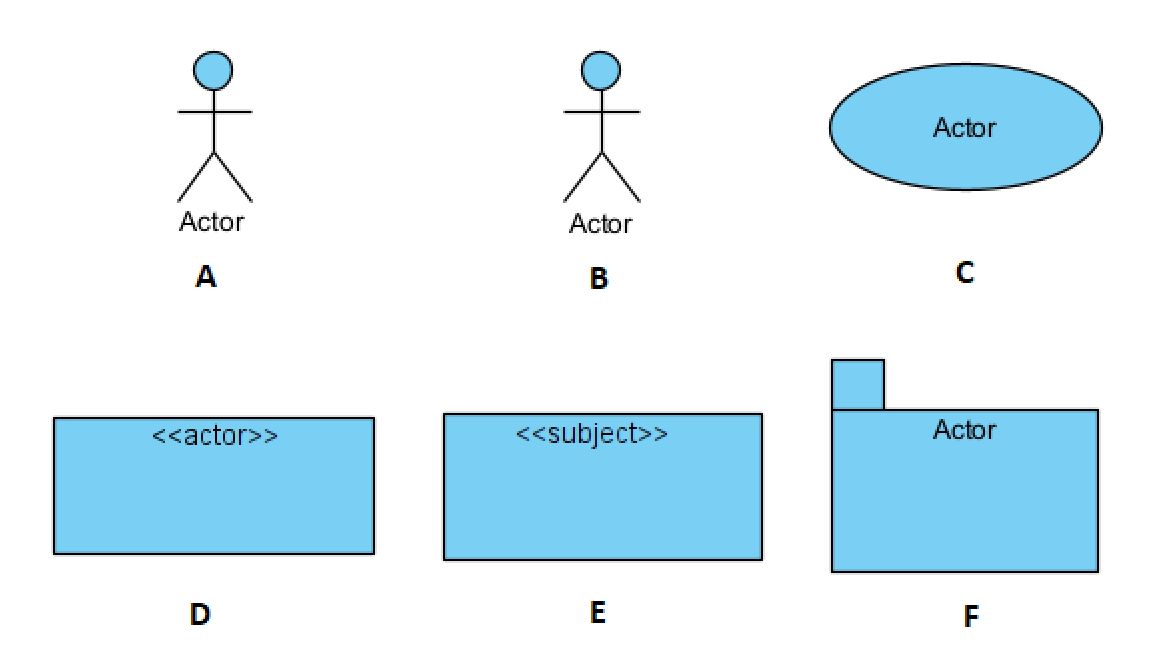


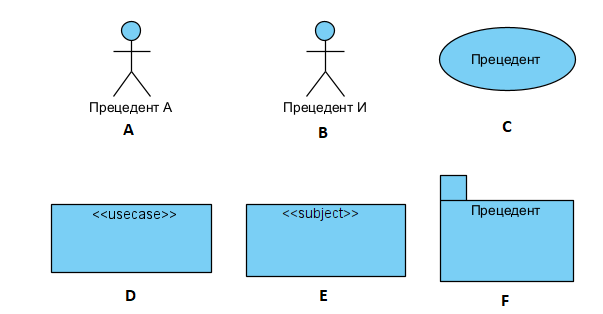
Рисунок 15 – Диаграмма прецедентов

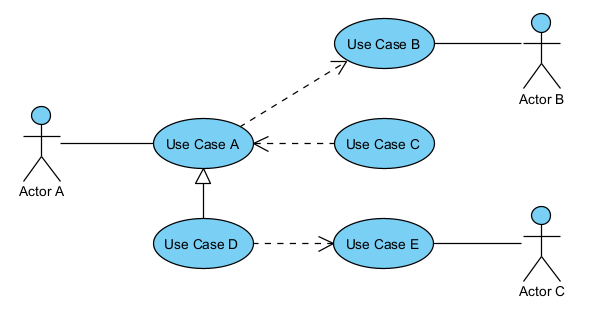
На этой диаграмме основные исполнители показаны слева, а вспомогательные справа. Связь взаимодействия может соединять так же и двух исполнителей, и два прецедента. Однако для этого вводятся специальные виды взаимодействия: обобщение, расширение и включение (использование этих типов связи в рамках анализа начальной фазы не является желательным).

Контрольные вопросы

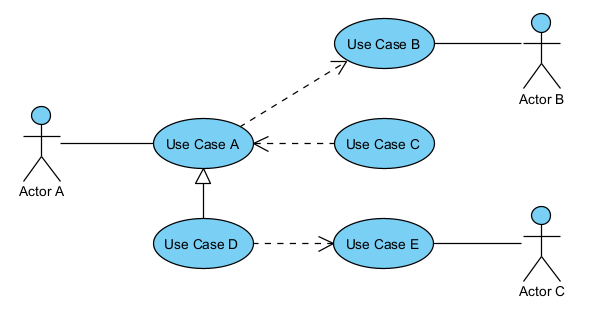
1. Какие символы являются стандартными представлениями актёра?

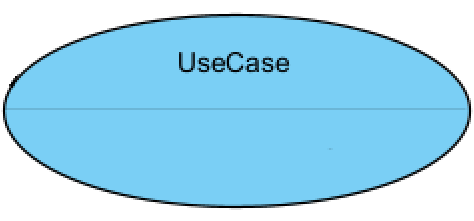


1. Каким символом изображается прецедент?
2. Какое значение имеет стрелка, изображенная на линии, связывающей актёра и прецедент?
3. Какие актёры вовлечены в выполнение прецедента Use Case E?



1. В каком количественном отношении находятся сценарии и прецеденты?
2. В каком отношении находятся понятия прецедента и кооперации?
3. Что такое прецедент?
4. Какой смысл вкладывают в понятие актёра?
5. В каких отношениях могут состоять прецеденты?
6. Какие актёры вовлечены в выполнение прецедента Use Case E?



1. В чем состоит смысл операции расширения прецедента?
2. Что описывается в дополнительном разделе прецедента, отделенном от его названия горизонтальной линией?
3. Как на диаграммах прецедентов изображается включение прецедентов?
   1. в виде зависимости со стереотипом <<include>>;
   2. в виде зависимости со стереотипом <<inside>>;
   3. в виде зависимости со стереотипом <<within>>;
   4. в виде зависимости со стереотипом <<switch on>>;
   5. в виде зависимости со стереотипом <<contain>>.

Содержание отчета

1. Титульный лист;
2. Цель работы;
3. Выполненное индивидуальное задание в соответствии с приведенным примером построения диаграммы прецедентов. Список индивидуальных заданий приводится в конце методического пособия;
4. Ответы на 5 контрольных вопросов;
5. Вывод

# Лабораторная работа №3

Цель: познакомиться с методами построения диаграммы классов, освоить элементы, используемые в процессе построения диаграммы, и научиться понимать назначение каждого из элементов.

Создание диаграммы классов

Диаграмма классов показывает объекты и отношения между ними. Она предоставляет подробную информацию о свойствах и интерфейсах классов, её можно рассматривать в качестве основной модели, а также рассматривать другие схемы в качестве дополнительных моделей.

Чтобы создать диаграмму классов, надо:

* Щелкнуть по панели «Diagram Navigator» и в выпадающем меню выбрать пункт «Class Diagram».
* Щелкнуть правой кнопкой мыши по «Class Diagram» в навигаторе диаграмм, и в контекстном меню выбрать «New Class Diagram».

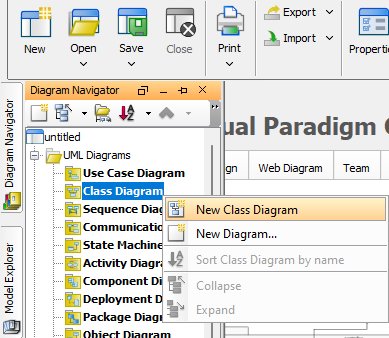


Рисунок 16 – Создание диаграммы классов

## *Создание класса*

Для создания класса необходимо щелкнуть на элементе «Class», который находится на панели инструментов диаграммы. После этого щелкнуть по пустому месту на полотне построения диаграммы.



Рисунок 17 – Создание и размещение нового класса

## *Создание атрибутов*

Чтобы создать новый атрибут класса, надо щелкнуть правой кнопкой мыши по классу и в контекстном меню выбрать «Add -> Attribute».

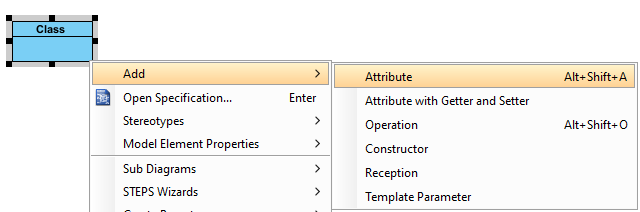


Рисунок 18 – Диалог добавления нового атрибута

После того, как атрибут добавится, можно нажать по клавише «Enter», после чего добавится ещё один новый атрибут. Этот метод позволяет быстро добавить несколько атрибутов.

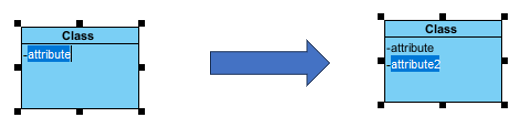


Рисунок 19 – Добавление нового атрибута

## *Создание отношения*

Для создания новой связи, необходимо щелкнуть по классу,

открыть Resource Catalog, выбрать «Association» в меню ресурсов и выбрать существующий класс для установления связи с ним, либо создать новый.

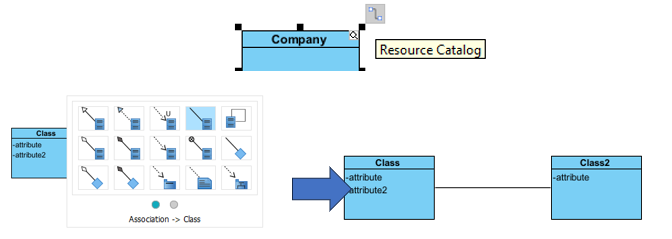


Рисунок 20 – Создание новой связи между классами

## *Создание отношения агрегация*

Для создания связи типа агрегация, необходимо выбрать «Aggregation Class» в меню ресурсов. Чтобы установить множественность для ассоциации, необходимо щелкнуть правой кнопкой мыши у того конца связи, который требуется модифицировать. В контекстном меню кликнуть по пункту Role A (Comapany), далее «Multiplicity», а затем выбрать требуемый вариант множественности.

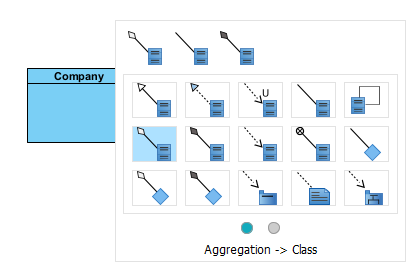
**

Рисунок 21 – Создание связи типа агрегация

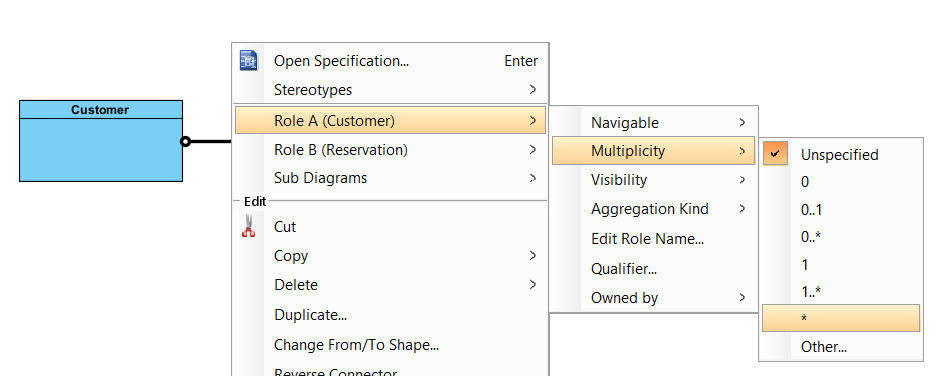
**

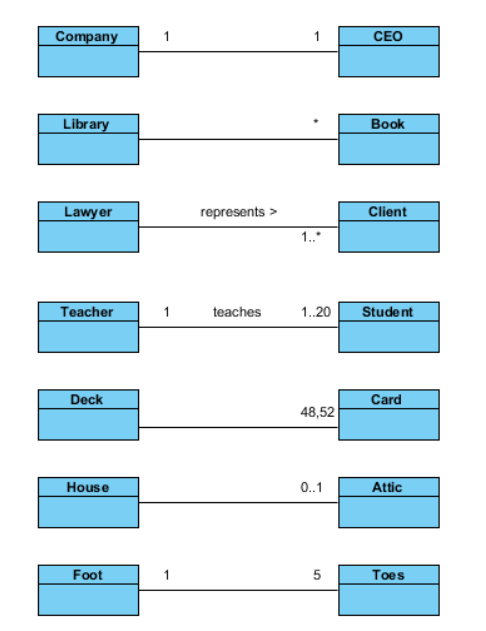
Рисунок 22 – Диалог выбора множественности…

Рисунок 23 – Примеры возможных множественностей

Для отображения направления связи, кликните правой кнопкой мыши по связи и выберете «Presentation Options», а затем выберите пункт «Show direction».

Стрелка направления связи располагается рядом со связью.

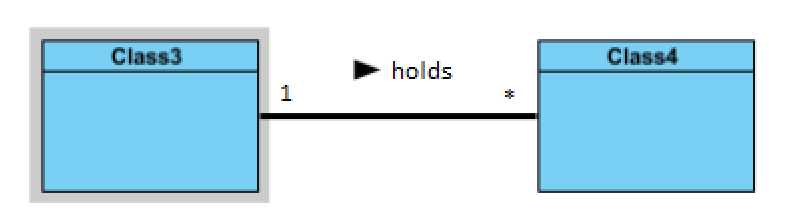


Рисунок 24 – Обозначение направления связи

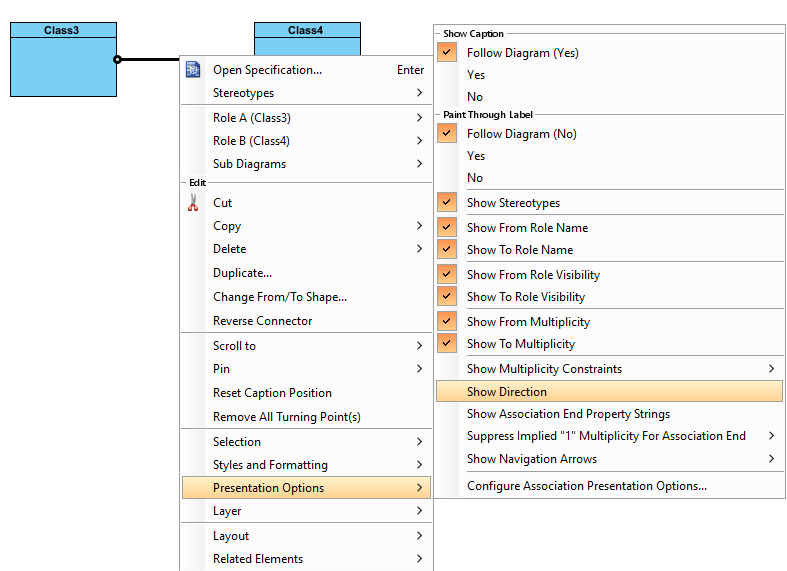


Рисунок 25 – Диалог включения свойства «Показать направление»

## *Создание отношения обобщение*

Для создания отношения типа обобщение, необходимо щелкнуть по пункту «Generalization» в меню ресурсов, расположенном рядом с классом. Затем вновь выбрать существующий класс, либо создать новый, для соединения с ним.

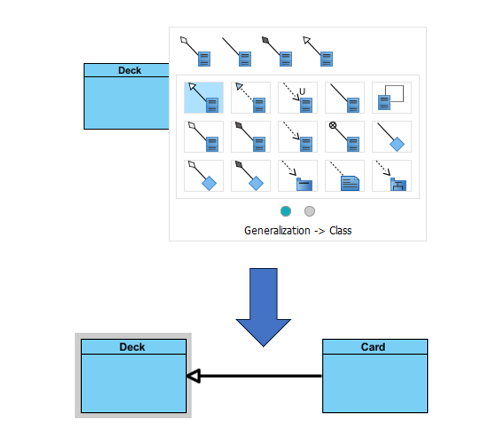


Рисунок 26 – Создание отношения обобщение

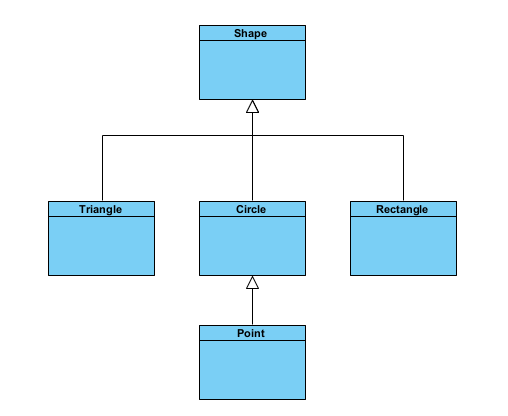


Рисунок 27 – Пример обобщения

## *Создание отношения реализация*

Для создания отношения типа реализация, необходимо кликнуть по пункту «Realization» в меню ресурсов, расположенном рядом с классом. Затем вновь выбрать существующий класс, либо создать новый, для соединения с ним.

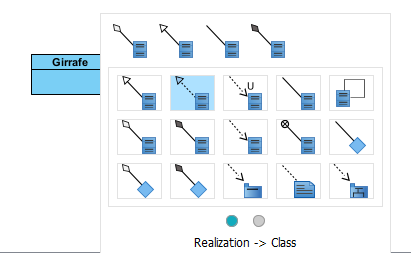


Рисунок 28 – Создание отношения реализация

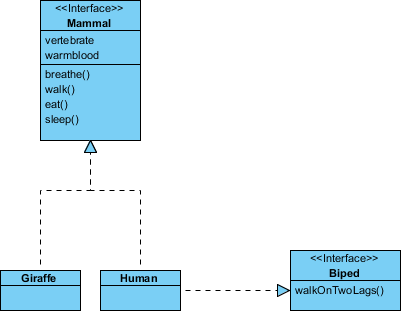


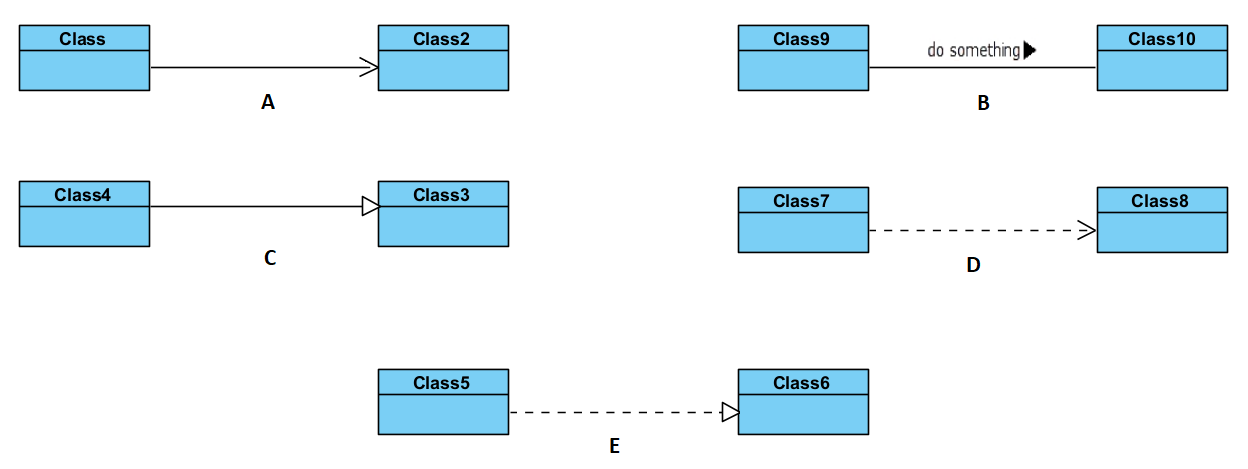
Рисунок 29 – Пример отношения реализация

Контрольные вопросы

1. В чем разница между модификаторами видимости public и

protected?

1. Что такое интерфейс?
2. Что является основой для реализации механизма интерфейсов в языках программирования?
3. Какими способами может изображаться однонаправленная ассоциация на диаграммах UML?



1. Что такое класс ассоциации?
2. Перечислите основные элементы диаграммы классов.
3. Как обозначается рекурсивная связь? Приведите пример.
4. Как обозначаются отношения агрегация и композиция. В чем разница между ними?

# Лабораторная работа №4-5

Цель: приобрести навыки построения диаграммы последовательностей, познакомиться с основными элементами диаграммы, изучить их назначение и обозначение.

Создание диаграммы последовательности

Диаграмма последовательности разрабатывается для представления взаимодействия или передачи сообщений между пользователями, системой, подсистемами и объектами.

Для создания диаграммы последовательности необходимо выбрать

«Sequence Diagram» в контекстном меню, которое появляется при клике правой кнопкой мыши по пункту «New» в навигаторе диаграмм.

A screenshot of a computer

Description automatically generated

Рисунок 30 – Создание диаграммы последовательности

Панель инструментов находится слева от полотна, на котором изображается диаграмма последовательности.

A screenshot of a computer

Description automatically generated

Рисунок 31 – Меню добавления новых элементов на диаграмму

*Создание актёра*

Чтобы создать нового актёра, нажмите по иконке «Actor», расположенной на панели инструментов диаграммы и затем на полотне рисования диаграммы.

A white background with black text

Description automatically generated

Рисунок 32 – Создание актёра

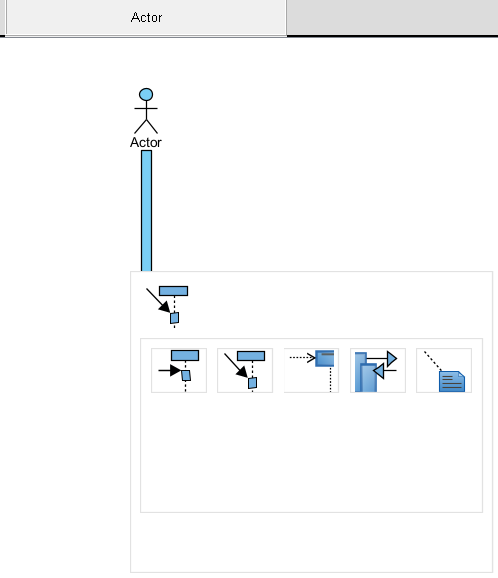


Рисунок 33 – Добавление актёра

Как видно из рис.33 с помощью меню ресурсов, можно обозначить простое новое сообщение или, например, сообщение, которое создает новый объект.

Также, на диаграмме последовательности, могут быть расположены

«Lifeline» или объекты определенного класса. Для добавления нового объекта на диаграмму, выберите мышкой элемент Lifeline на панели инструментов диаграммы и перенесите его на полотно рисования диаграммы.

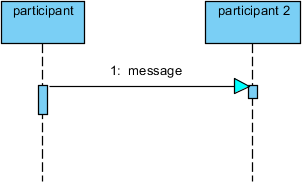


Рисунок 34 – Простая диаграмма последовательности

## *Создание сообщений*

Для создания нового сообщения необходимо выбрать один из возможных типов сообщений и провести линию между двумя объектами на диаграмме. На рисунке 35 показаны возможные типы сообщений, которые могут присутствовать на диаграмме последовательности.

Для обозначения простого сообщения используется линия с закрашенной стрелкой на конце. Такой тип сообщений называется синхронные сообщения. Это говорит о том, что после этого сообщения система ожидает результата, либо завершения операции.

Так называемые возвращаемые сообщения (return message)

обозначаются пунктирной линией с незакрашенной стрелкой на конце.

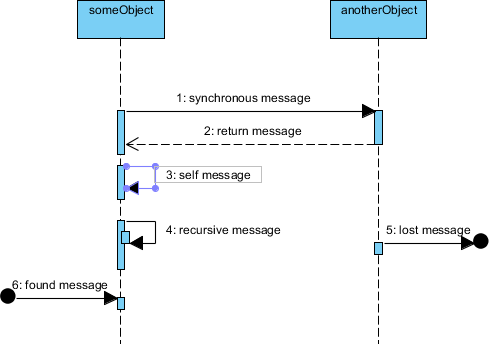


Рисунок 35 – Возможные типы сообщений

Существует также тип сообщений, которые посылаются самому себе (self-message). Это могут быть вызовы операций внутри объекта и т.д. Похожим способом отображаются рекурсивные (recursive message) сообщения.

Также существует 2 типа сообщений: потерянные (lost) и найденные (found). При обозначении потерянных сообщений, подразумевается, что объект, который должен принять отправленное сообщение не изображен на данной диаграмме последовательности, и, наоборот; при обозначении найденных сообщений, подразумевается, что объект-отправитель не обозначен на текущей диаграмме, а объект-приёмник обозначен.

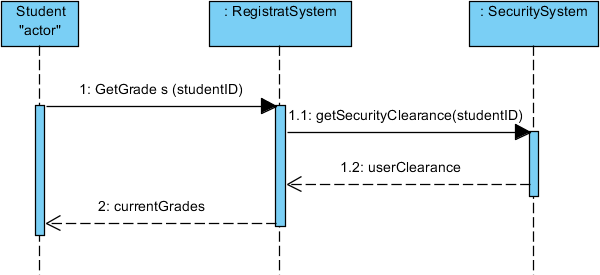


Рисунок 36 – Пример диаграммы последовательности с разными типами сообщений

Иногда появляется необходимость обозначить объект, который создает другой объект, т.е. изначально объект-2 не существует. Для реализации такого сообщения существует два типа: сообщение создание объекта (create message) и сообщение уничтожение объекта (destroy message). Оба типа сообщения введены для определения времени жизни объекта. С их помощью можно создавать и удалять какой-либо объект.

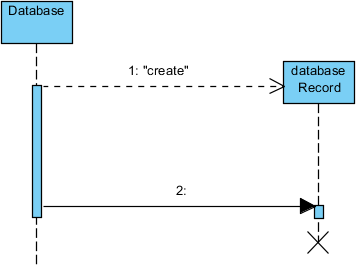


Рисунок 37 – Сообщения создания и удаления записи БД

## *Создание фреймов*

Предположим, что существуют сообщения, которые объект должен инициировать по какому-то условию. В таких ситуациях нам могут помочь фреймы. Однако, если сообщение единично, то можно прямо в его названии указать условие, по которому оно должно быть инициировано.

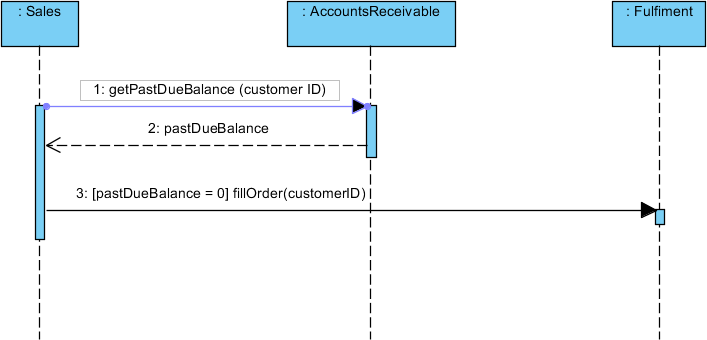


Рисунок 38 – Описания условия инициирования единичного сообщения

Но когда возникает необходимость инициировать несколько сообщений, такой подход не подойдет. Для этого существуют фреймы.

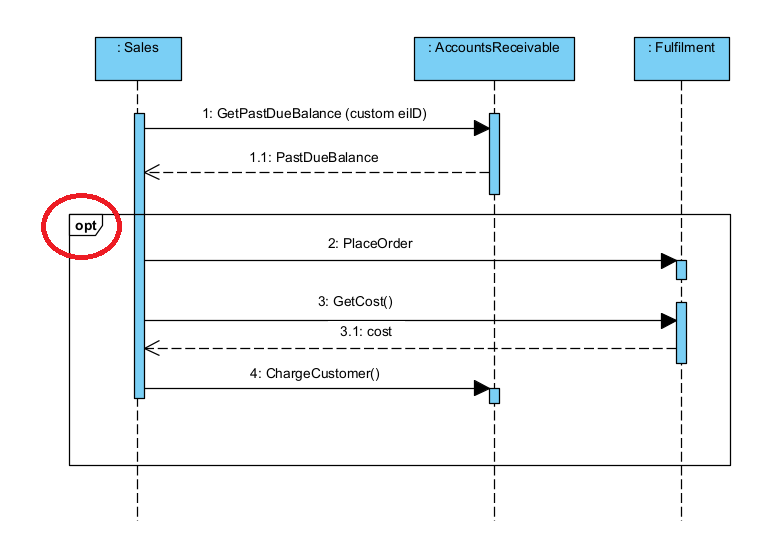


Рисунок 39 – Пример использования опционального фрейма Выделенный элемент на рисунке 39 показывает, что этот фрейм имеет

тип опционально (opt), т.е. сообщения внутри него будут выполняться после определенного условия, которое указывается рядом с типом фрейма.

Фреймы бывают следующих типов:

* alt – альтернативный фрейм. Можно описать как последовательность операторов if … else в алгоритмах;
* loop – циклический фрейм. Означает, что все сообщения, находящиеся внутри него, будут выполняться несколько раз;
* opt – опциональный фрейм. Можно представить как последовательность оператора if без блока else;
* par – параллельное выполнение. Этот тип фрейма означает, что сообщения внутри него посылаются параллельно друг другу;

Для выбора соответствующего типа среда Visual Paradigm

предоставляет следующее меню.

A screenshot of a computer

Description automatically generated

Рисунок 40 – Добавление нового фрейма

Для вызова этого контекстного меню, требуется выбрать те сообщения, которые предполагается заключить внутрь фрейма, затем щелкнуть правой кнопкой мыши и выбрать требуемый тип фрейма.

На рисунке 41 показан вариант диаграммы с альтернативным фреймом. У него есть пунктирная линия, которая разделяет те сообщения, которые будут выполняться при срабатывании условия, от тех, которые будут выполнены при возврате отрицательного результата при проверке условия.

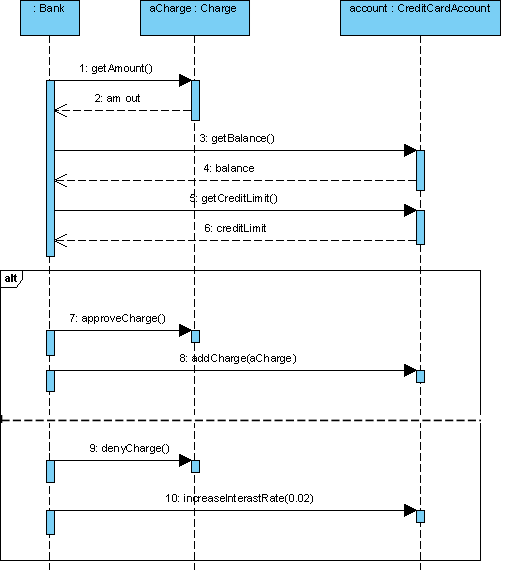


Рисунок 41 – Пример диаграммы с альтернативным фреймом

На рисунке 42 показан пример с параллельными сообщениями.

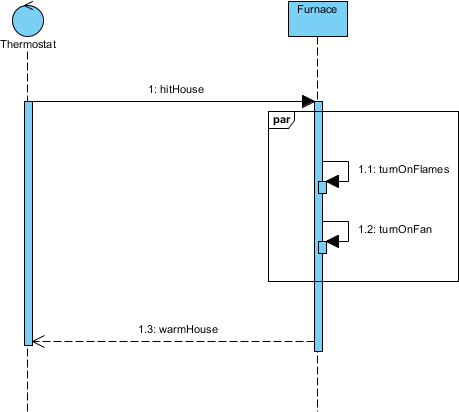
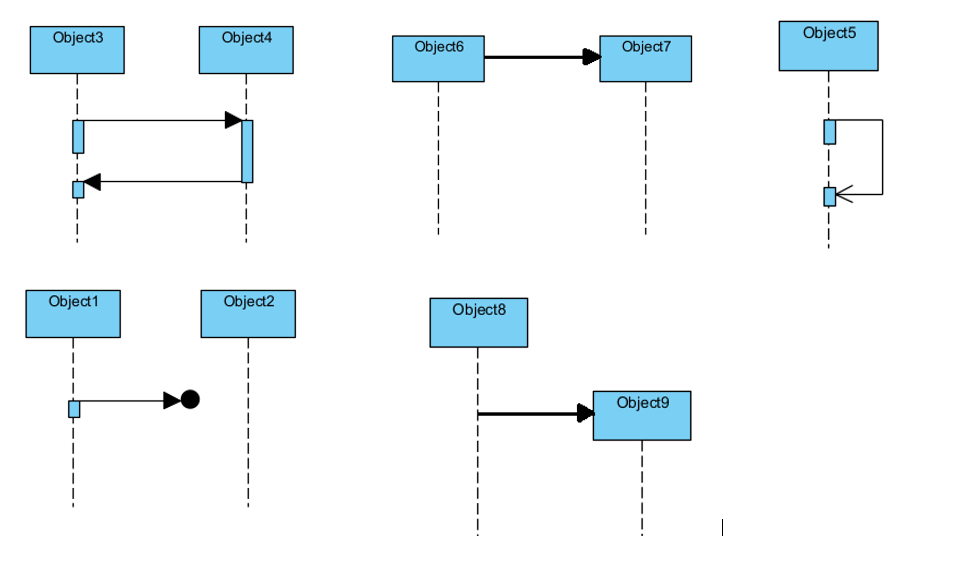


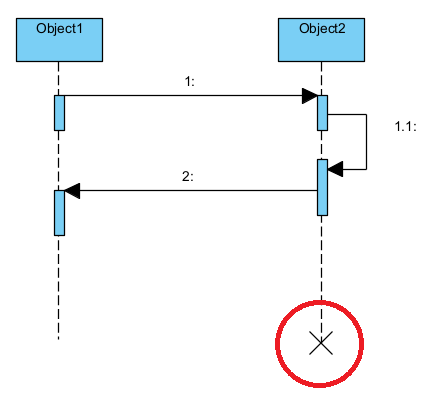
Рисунок 42 – Диаграмма с параллельным фреймом

Контрольные вопросы

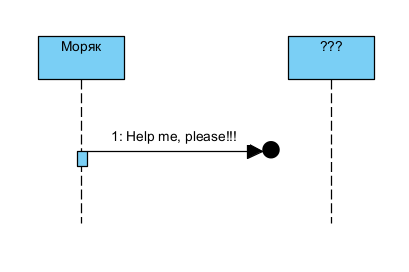
1. Использование каких элементов UML, кроме объектов, допускается на диаграмме последовательностей?
2. Что такое асинхронное сообщение?
3. Что такое найденные сообщения?
4. Каким образом отображается на диаграмме последовательностей порядок отправки и приема сообщений?
5. Почему некоторые сообщения на диаграммах последовательностей иногда обозначают пунктирной линией?
6. Что такое рефлексивное сообщение?
7. Что такое синхронное сообщение?
8. Какие из фрагментов диаграмм последовательностей НЕ противоречат нотации UML?



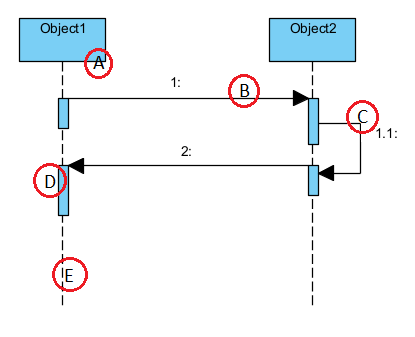
1. Что означает символ, выделенный на рисунке?



1. Как называется тип сообщения, пример которого изображен на рисунке?



1. Какой буквой на рисунке обозначена линия жизни объекта?



Содержание отчета

1. Титульный лист;
2. Цель работы;
3. Выполненное индивидуальное задание. Список индивидуальных заданий приводится в конце методического пособия. Необходимо привести пример диаграммы последовательности, которая отразит взаимодействие актёра с системой, а также взаимодействие объектов внутри самой системы;
4. Ответы на 5 контрольных вопросов;
5. Выводы

# Лабораторная работа №7

Цель: освоить навыки построения диаграммы компонентов, изучить основные элементы, их назначение и обозначение.

## **Теоретическая часть**

Диаграмма компонентов используется для моделирования физического аспекта объектно-ориентированного программного обеспечения, иллюстрируя архитектуру программных компонентов и связи между ними. Эти программные компоненты включают в себя: компоненты времени выполнения, исполняемые компоненты и исходный код компонентов.

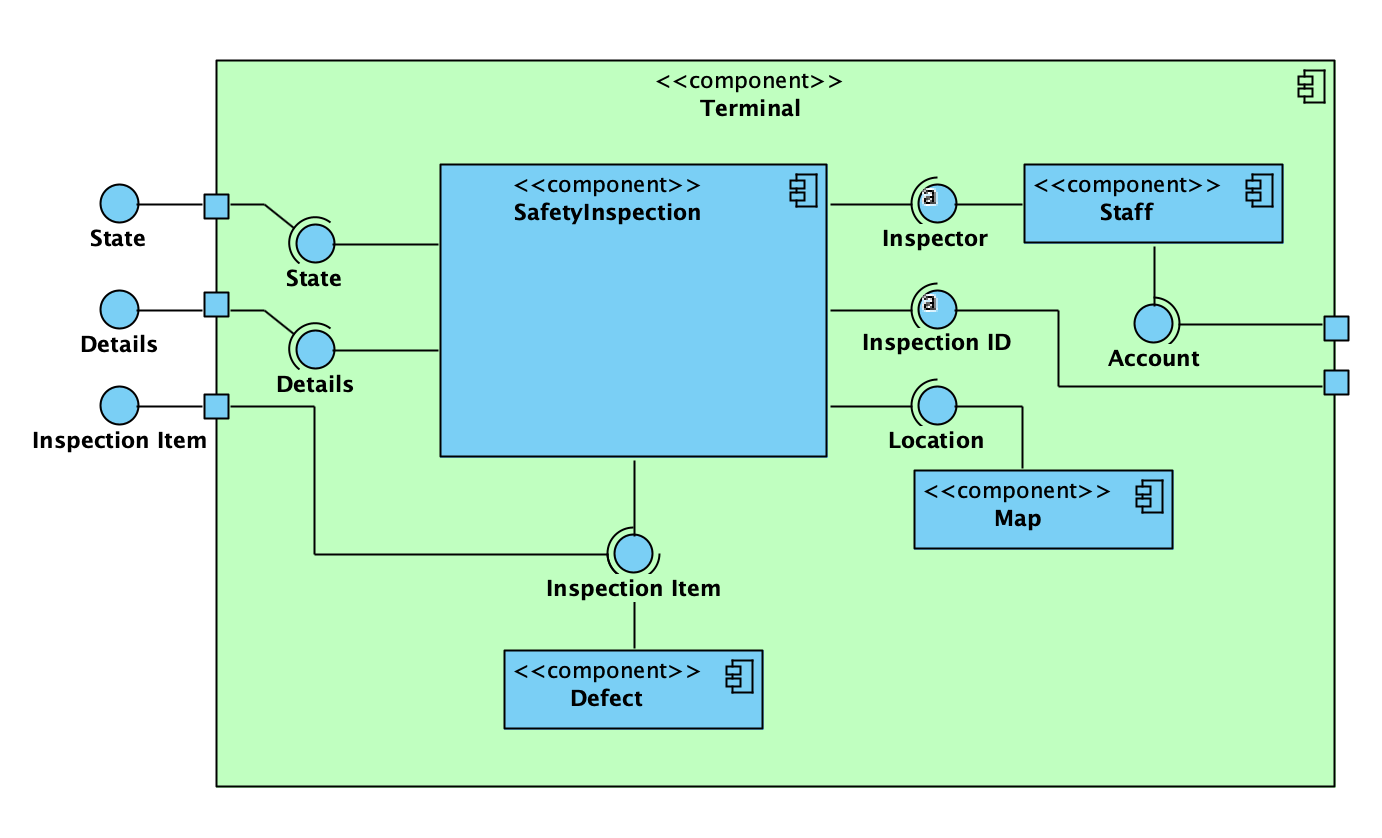


Рисунок 44 – Пример диаграммы компонентов

Для построения диаграммы компонентов используется следующая нотация:

|  |  |
| --- | --- |
| *Обозначение* | *Название* |
|  | Агрегация |
|  | Компонент |
|  | Ограничение |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Component/Generalization_ico.png | Обобщение |
|  | Интерфейс |
|  | Примечание |
|  | Реализация |
|  | Ассоциация |
|  | Композиция |
|  | Спецификация экземпляра компонента |
|  | Связь |
|  | Port |
|  | Usage |

## Агрегация

Агрегация – тип ассоциации, который означает, что элемент агрегирует в себе другой элемент, указанный на другом конце связи.

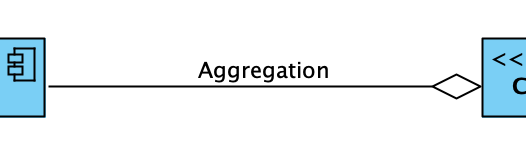


Рисунок 45 – Обозначение агрегации

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя агрегации |
| Видимость | Определяет, где появляется агрегация в других пространствах имен в рамках общей модели, и её доступность |
| Начало ассоциации | Источник агрегации |
| Конец ассоциации | Объект агрегации |
| Документация | Описание агрегации |
| Абстрактная | Если истинно, значит, агрегация не обеспечивает полное объявление и не может быть создана.  Абстрактная агрегация предназначена для использования другими агрегациями |
| Специализация | Указывает, возможно ли в дальнейшем специализировать агрегацию. Если это значение истинно, значит, агрегацию специализировать в дальнейшем нельзя |
| Производная | Указывает, является ли агрегация производной от других элементов модели, например, агрегаций или ограничений |

## Ассоциация

Агрегация обозначает семантическую связь между типизированными экземплярами, имеющую по крайней мере два конца, представленные свойствами, каждый из которых связан с типом. Оба конца ассоциации могут иметь один и тот же тип.

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя ассоциации |
| Видимость | Определяет, где появляется ассоциация в других пространствах имен в рамках общей модели, и её доступность |
| Начало ассоциации | Источник ассоциации |
| Конец ассоциации | Объект ассоциации |
| Документация | Описание ассоциации |
| Абстрактная | Если истинно, значит, ассоциация не обеспечивает полное объявление и не может быть создана.  Абстрактная ассоциация предназначена для использования другими ассоциациями |
| Специализация | Указывает, возможно ли в дальнейшем специализировать ассоциацию.  Если это значение истинно, значит, ассоциацию специализировать в дальнейшем нельзя |
| Производная | Указывает, является ли ассоциация производной от других элементов модели, например, ассоциаций или ограничений. |

## Компонент

Компонент представляет собой модульную часть системы, которая инкапсулирует своё содержимое и способна изменять своё описание в пределах своей среды.

Он определяет поведение через предоставляемые и требуемые интерфейсы, служа типом соответствия для этих интерфейсов. Замена одного компонента другим возможна только при совместимости их типов. Большие функциональные блоки системы могут быть собраны за счёт повторного использования компонентов или сборки из них, связанных соответствующими интерфейсами.

Моделирование компонента происходит на протяжении всего жизненного цикла разработки и детализируется в процессе развёртывания и выполнения. Спецификация развертывания может определять значения, параметризующие выполнение компонентов.

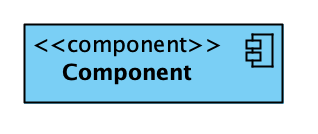


Рисунок 46 – Обозначение компонента

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя компонента |
| Видимость | Определяет, где появляется компонент в других пространствах имен в рамках общей модели, и его доступность |
| Документация | Описание компонента |
| Абстрактный | Если истинно, значит, компонент не обеспечивает полное объявление и не может быть создан.  Абстрактный компонент предназначен для использования другими компонентами |
| Специализация | Указывает, возможно ли в дальнейшем специализировать компонент. Если это значение истинно, значит, компонент специализировать в дальнейшем нельзя |
| Родительский | Указывает, имеет ли компонент родителя |
| Активный | Определяет, может ли объект являться активным |
| Тип создания экземпляра | Тип создания экземпляра, который относится к компоненту.  Если ложь, значит, компонент создается как адресуемый объект. Если истина, то компонент определяется во время  разработки, но во время выполнения объект, имеющий тип этого компонента не существует, т.е. объект создается косвенно, в результате конкретизации его реализующих классификаторов |

## 

## Композиция

Ассоциация может быть представлена как композиция, являющаяся частным случаем агрегации. Только бинарная ассоциация может быть композицией. Композиция представляет собой более строгую форму агрегации, где каждый экземпляр может быть включен только в одну композицию одновременно. При удалении композиции все её части также будут удалены. Важно отметить, что часть композиции может быть удалена до самой композиции.

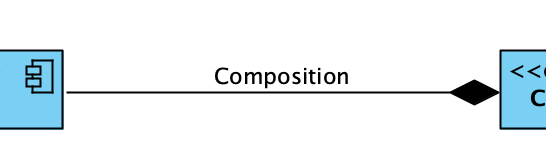


Рисунок 47 – Обозначение композиции

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя композиции |
| Видимость | Определяет, где появляется композиция в других пространствах имен в рамках общей модели, и её доступность |
| Начало ассоциации | Источник композиции |
| Конец ассоциации | Объект композиции |
| Документация | Описание композиции |
| Абстрактная | Если истинно, значит, композиция не обеспечивает полное объявление и не может быть создана. Абстрактная композиция предназначена для использования другими композициями |
| Специализация | Указывает, возможно ли в дальнейшем специализировать композицию. Если это значение истинно, значит, композицию специализировать в дальнейшем нельзя |
| Производная | Указывает, является ли композиция производной от других элементов модели, например, композиций или ограничений |

## 

## Ограничение

Условие или ограничение, выраженное в тексте на естественном или машиночитаемом языке, с целью определения некоторой семантики элемента.

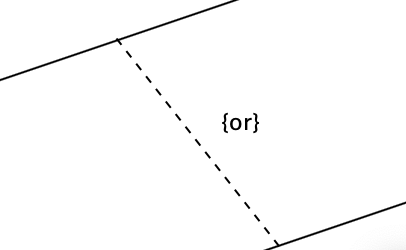


Рисунок 48 – Обозначение ограничения

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя ограничения, обычно опускается |
| Выражение | Условие, по которому происходит сравнение чего-либо |
| Документация | Описание ограничения |

## 

## Зависимость

Зависимость - это отношение, которое подразумевает, что один или несколько элементов модели нуждаются в других элементах модели для своего описания или реализации. То есть полная семантика элемента зависит либо семантически, либо структурно от элементов-поставщиков.

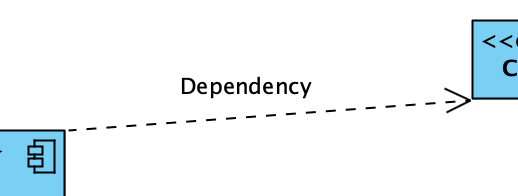


Рисунок 49 – Обозначение зависимости

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя зависимости |
| Поставщик | Элемент, от которого зависит какойлибо другой элемент. |
| Клиент | Элемент, который зависит от элемента или множества элементов поставщиков. |
| Видимость | Определяет, где появляется зависимость в других пространствах имен в рамках общей модели, и её доступность. |
| Документация | Описание зависимости |

## Обобщение

Обобщение – это отношение между более общим классификатором и более конкретным классификатором, при котором каждый экземпляр конкретного классификатора косвенно является общим классификатором. Таким образом, более конкретный классификатор наследует черты общего классификатора.

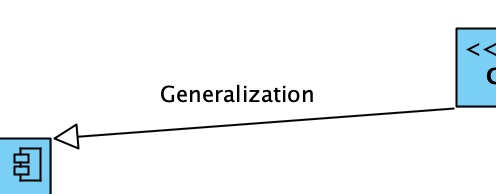


Рисунок 50 – Обозначение обобщения

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя обобщения |
| Общий | Определяет общий классификатор в отношении ‘обобщение’ |
| Конкретный | Определяет конкретный классификатор в отношении ‘обобщение’ |
| Видимость | Определяет, где появляется отношение обобщение в других  пространствах имен в рамках общей модели и его доступность |
| Документация | Описание отношения обобщение |
| Заменяемый | Определяет, можно ли использовать конкретный классификатор там, где используется общий. |

## Интерфейс

Интерфейс - это тип классификатора, который представляет собой объявление множества публичных функций и обязательств. Он определяет контракт, и любой объект, реализующий этот интерфейс, должен соблюдать этот контракт. Обязательства, связанные с интерфейсом в виде различных видов ограничений (до и после выполнения) или спецификаций протокола, могут накладывать ограничения на взаимодействие через интерфейс.

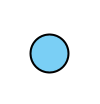


Рисунок 51 – Обозначение интерфейса

Поскольку интерфейс только декларируется, он не имеет экземпляров. Вместо этого, интерфейс реализуется классификаторами, которые могут существовать как объекты. Важно отметить, что один классификатор может реализовывать несколько интерфейсов, а также один интерфейс может быть реализован множеством классификаторов.

## Связь

Связь – это ассоциация, описывающая возможность устанавливать связь между экземплярами связанных типов.

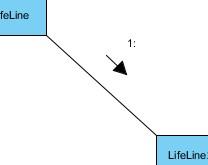


Рисунок 52 – Обозначение связи

|  |  |
| --- | --- |
| *Свойство* | *Пояснение* |
| Имя | Имя связи |
| Источник | Источник связи |
| Приёмник | Объект связи |
| Описание | Спецификация того, как вычислить, получить или построить экземпляр. |
| Классификаторы | Классификатор или классификаторы из представленного экземпляра. Если  несколько классификаторов указаны, экземпляр классифицируется по каждому из них. |
| Документация | Описание связи |

## 

## Примечание

Примечание позволяет присоединять комментарии к элементам. Комментарий не несет никакой смысловой нагрузки, но может содержать информацию, полезную для разработчика.

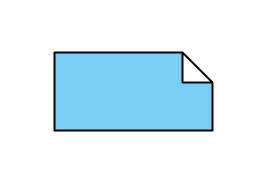


Рисунок 53 – Обозначение примечания

## 

## 

## Реализация

Реализация - специализированное отношение между двумя наборами элементов модели, где один набор представляет собой поставщика, а другой - клиента. Реализация может использоваться для моделирования пошагового уточнения, оптимизации, преобразований, шаблонов, синтеза модели и т.д.

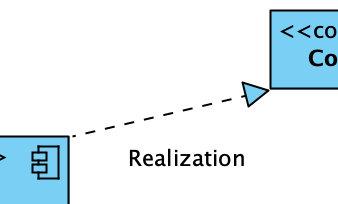


Рисунок 54 – Обозначение реализации

## **Практическая часть**

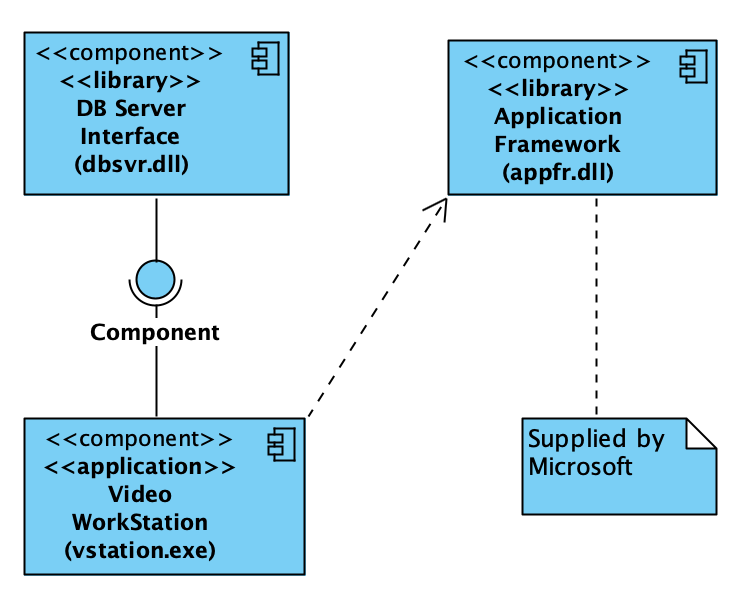


Рисунок 55 – Пример диаграммы компонентов

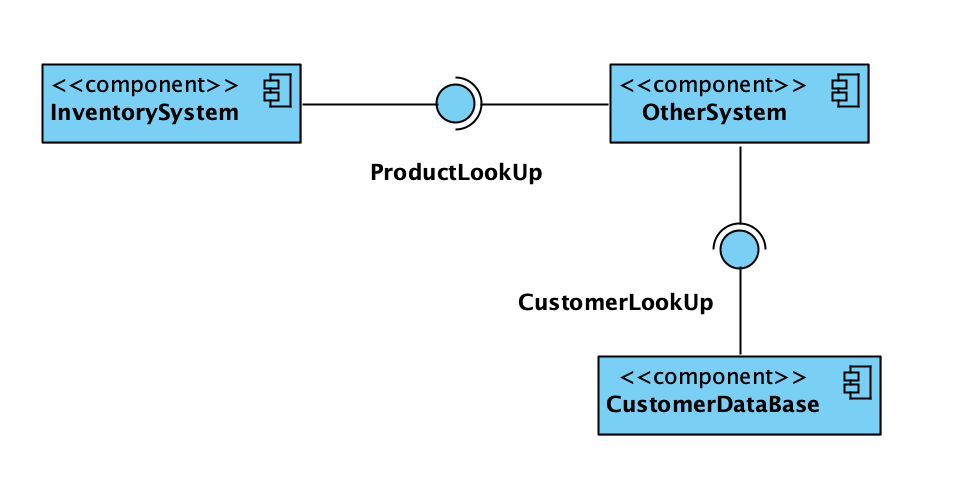


Рисунок 56 – Пример обозначения требуемых и реализуемых интерфейсов

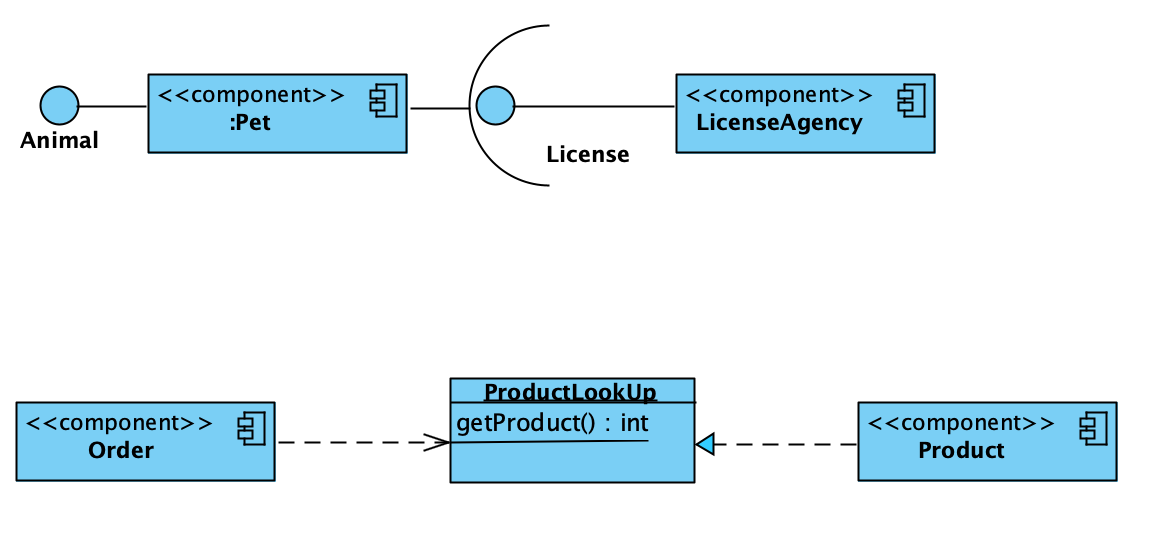


Рисунок 57 – Подходы к обозначению отношений между интерфейсами и

компонентами

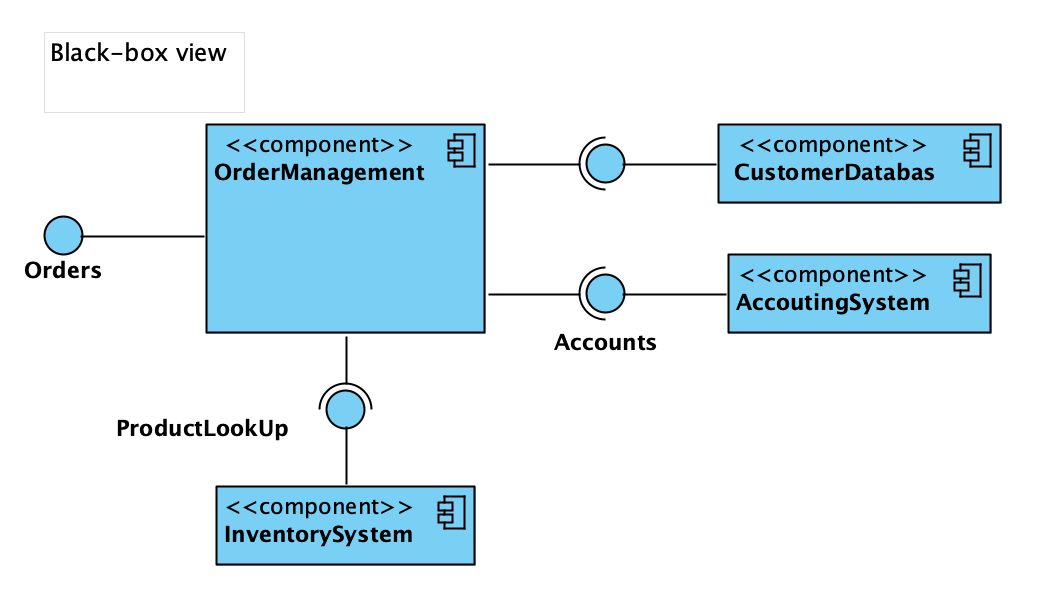


Рисунок 58 – Вид на компоненты снаружи

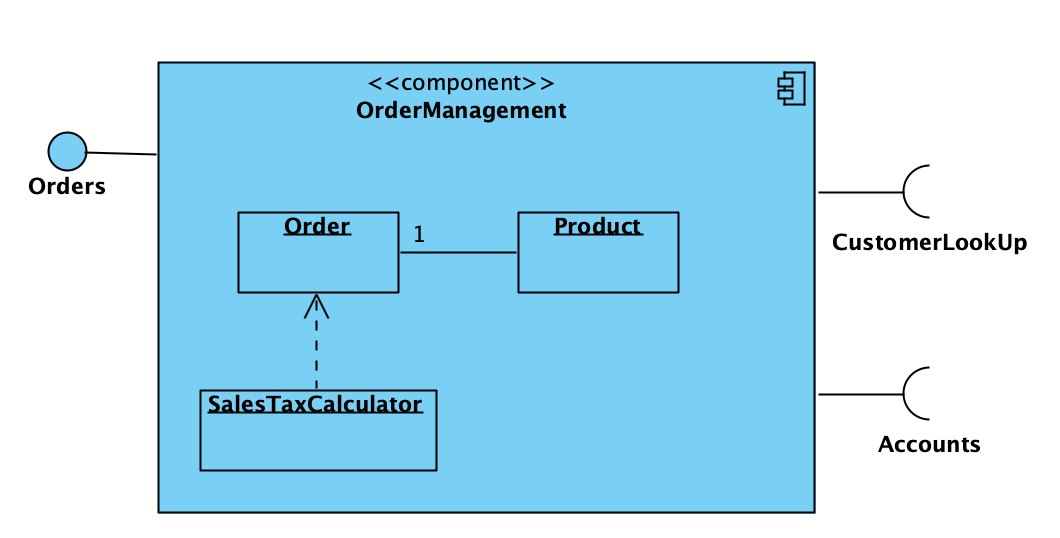


Рисунок 59 – Вид на компонент изнутри

|  |  |  |
| --- | --- | --- |
| Английское название стереотипа | Русское название  стереотипа | Описание |
| Application | Приложение | Пользовательский интерфейс |
| Datastore | Хранилище данных | Место хранения данных |
| Document | Документ | Электронный или печатный документ |
| Entity | Сущность | Представляет некоторую бизнес концепцию. Не содержит операций, только атрибуты |
| Executable | Исполняемый | Компонент может выполняться на узле |
| File | Файл | Файл данных |
| Infrastructure | Инфраструктура | Специальный компонент внутри системы |
| Library | Библиотека | DLL |
| Process | Процесс | Репозиторий (реализация операций над сущностями) |
| Realization | Реализация | Реализует другой компонент, его описание |
| Service | Сервис | Удовлетворяет какой-либо функциональный запрос и, как правило, не сохраняется |
| Source code | Исходный код | Java или C++ файл |
| Specification | Описание | Содержит интерфейсы, но не имеет |
| Subsystem | Подсистема | Часть большой системы. Может содержать несколько маленьких компонентов внутри себя, рассматривается как кусок большой системы. | |
| Table | Таблица | Таблица базы данных, которая хранит данные. |
| Web service | Веб-сервис |  |
| XML DTD |  | Описание схемы xml-документа. |

## **Содержание отчета**

1. Титульный лист;
2. Цель работы;
3. Выполненное индивидуальное задание. Список индивидуальных заданий приводится в конце методического пособия;
4. Выводы

# Лабораторная работа №8

Цель: приобрести навыки построения диаграммы пакетов, познакомится с основными элементами диаграммы, изучить их назначение и обозначение.

Диаграмма пакетов показывает расположение и организацию элементов модели в центре масштабного проекта. Диаграмма пакетов одновременно показать структуру и зависимости между подсистемами или модулями.

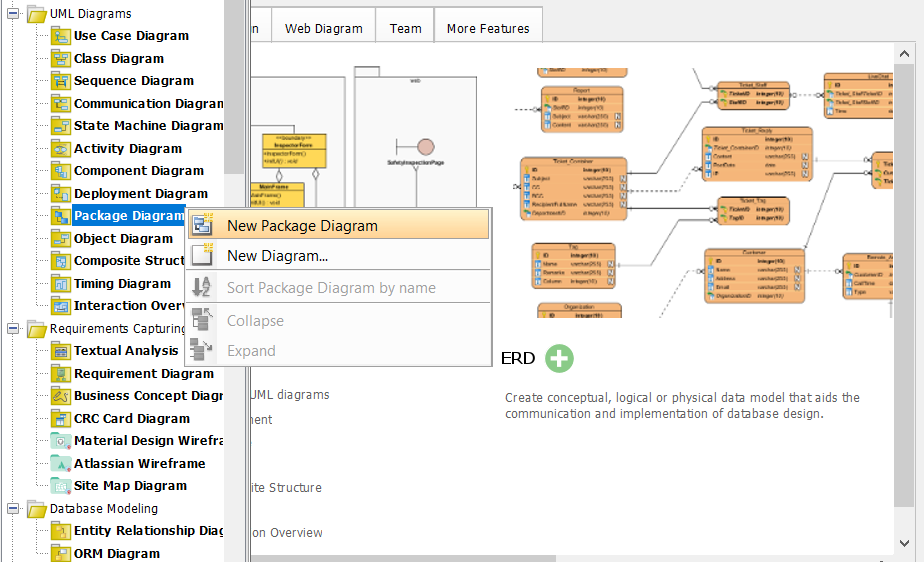


Рисунок 60 – Создание диаграммы пакетов

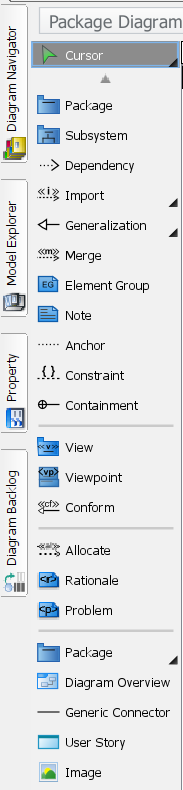


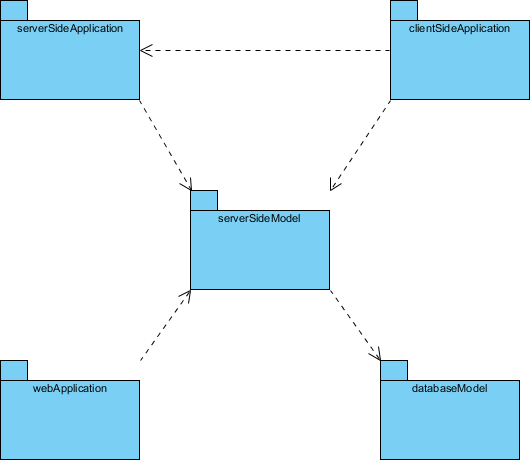
Рисунок 61 – Меню добавления новых элементов на диаграмму

Рисунок 62 – Диаграмма пакетов

Для создания диаграммы пакетов используется следующая нотация:

|  |  |
| --- | --- |
| Иконка | Описание |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Access_ico.png | Доступ |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Dependency_ico.png | Зависимость |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Import_ico.png | Импорт |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Note_ico.png | Примечание |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Realization_ico.png | Реализация |

|  |  |
| --- | --- |
| Иконка | Описание |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Constraint_ico.png | Ограничение |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Generalization_ico.png | Обобщение |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Merge_ico.png | Слияние |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Package_ico.png | Пакет |
| http://www.visual-paradigm.com/VPGallery/img/diagrams/Package/Subsystem_ico.png | Подсистема |

Теоретическая часть

## *Доступ*

Импорт элемента определяется как направленное отношение между импортируемым пространством имён и упаковываемого элемента. Название упакованного элемента или псевдоним должно быть добавлено в импортируемое пространство имён.

Импортируемый элемент показан с помощью пунктирной стрелки, с открытым концом из импортируемого пространства имён к импортируемому элементу.

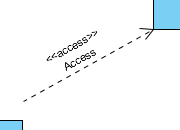


Рисунок 63 – Обозначение доступа

## *Импорт*

Импортирование пакета определяется как направленное отношение, которое идентифицирует атрибуты пакета, которые будут импортированы в пространство имён.

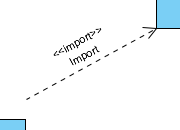


Рисунок 64 – Обозначение импорта

*Объединение*

Объединение пакета является направленным отношением между двумя пакетами, которое показывает, что содержимое пакетов должно быть объединено. Это очень напоминает обобщение, когда исходный элемент добавляет характеристики целевого элемента к себе, что приводит к появлению новых характеристик в исходном элементе. Таким образом, исходный элемент содержит все характеристики.

Этот механизм должен использоваться, когда разные пакеты имеют одинаковые имена и предназначены для представления одной и той же концепции. Чаще всего он используется, чтобы определить разные уровни, которые используются для разных целей, т.е. на каждой следующей итерации объединения, пакет дополняется новыми характеристиками, начиная от базовой своей реализации и далее.

Концептуально, слияние пакета можно рассматривать как операцию создания нового пакета, которая берет содержимое двух пакетов и производит новый пакет.

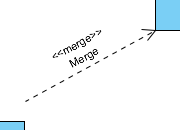


Рисунок 65 – Обозначение слияния

## *Пакет*

Пакет – это элемент, который используется для группировки элементов и предоставляет пространство имён для сгруппированных элементов. Пакет может содержать в себе другие пакеты.

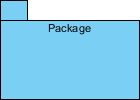


Рисунок 66 – Обозначение пакета

## *Подсистема*

Подсистема представляет границу физической системы.

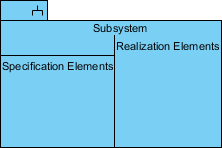


Рисунок 67 – Обозначение подсистемы

## Практическая часть

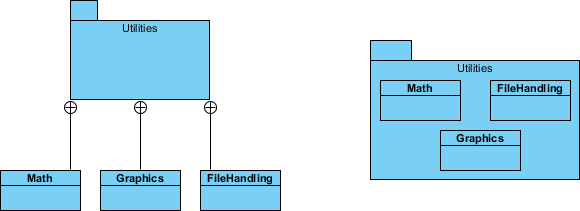


Рисунок 68 – Обозначение содержания пакета

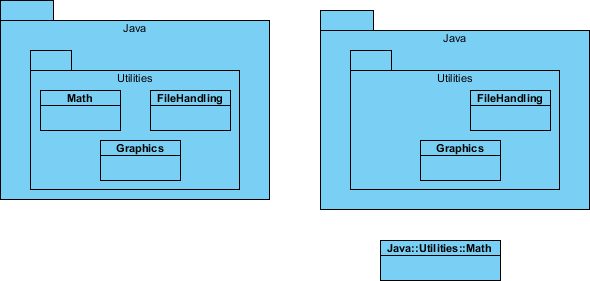


Рисунок 69 – Обозначение класса вне пакета с указанием всех пространств имён

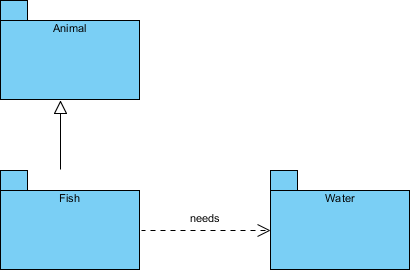


Рисунок 70 – Обозначение зависимости и обобщения

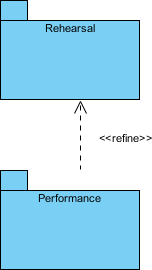


Рисунок 71 – Обозначение улучшения характеристик пакета относительно другого пакета

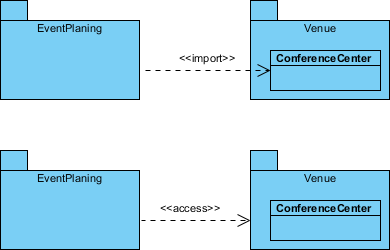


Рисунок 72 – Доступ и импорт пакета

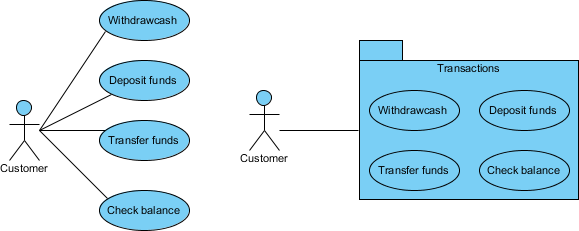


Рисунок 73 – Использование пакетов с прецедентами

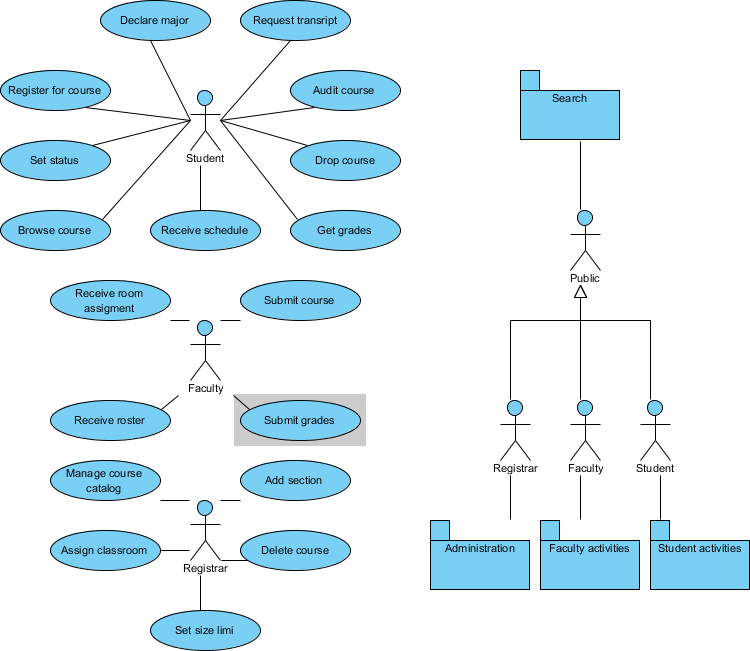


Рисунок 74 – Использование пакетов вместе с прецедентами

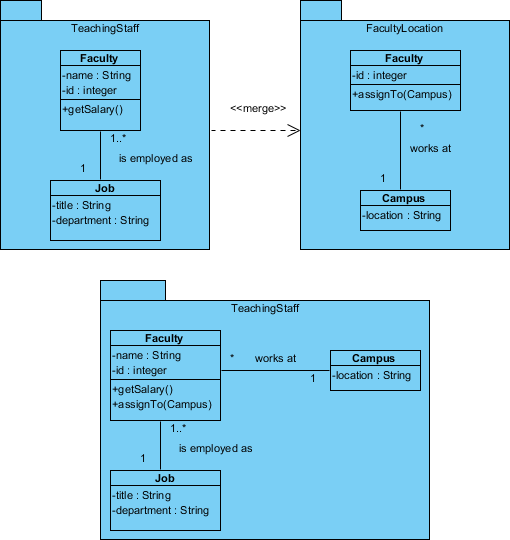


Рисунок 75 – Слияние пакетов

Содержание отчета

1. Титульный лист;
2. Цель работы;
3. Выполненное индивидуальное задание. Список индивидуальных заданий приводится в конце методического пособия;
4. Выводы.

# Приложение 1. Титульный лист

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра Сетевых информационных технологий и сервисов (СИТиС)

Отчёт по лабораторной работе №1

По дисциплине «Методы и средства проектирования информационных систем и технологий»

Выполнили: студ. гр. БСТ2100

Бригада №0

Проверил: ст.пр. кафедры СИТиС:

Рахмани Д.Д.

Москва, 2024 г.