Задание на практическую работу

В качестве задания на практическую работу студенту предлагается выполнить перечисленные ниже задания, с последующей защитой. Каждому студенту необходимо выполнить 4 лабораторные работы.

Для выполнения лабораторных работ необходимо выбрать исследуемую область. Исследуемая область выбирается студентом индивидуально, например:

- Учет товаров;
- Учет материальных ценностей;
- Учет основных средств;
- Управление персоналом;
- Расчеты с персоналом;
- Кадровое делопроизводство;
- Учет продаж;
- Управление заявками на ТО;
- Управление запасами;
- Формирование производственных заказов.

Все лабораторные работы можно выполнять как для одной исследуемой области, так и выбирать область для выполнения каждой работы.

Результаты работы необходимо оформить в виде отчета.

Студент, не выполнивший и не защитивший практическую работу, к тестированию или теоретическому экзамену не допускается.

Лабораторная работа 1

ПРОЕКТИРОВАНИЕ СИСТЕМЫ КЛАССИФИКАЦИИ И КОДИРОВАНИЯ

Цель работы: получить навыки проектирования систем классификации и кодирования различных видов информации, полученной в ходе проведения предпроектного обследования бизнес-процессов предприятия.

Задачи: для каждого элемента перечня данных предпроектного обследования сформулировать и поставить в соответствие свой уникальный код.

Теоретическая часть:

При любой классификации желательно, чтобы соблюдались следующие требования:

– полнота охвата объектов рассматриваемой области;

– однозначность реквизитов;

– возможность включения новых объектов.

При классификации широко используются понятия «классификационный признак» и «значение классификационного признака», которые позволяют установить сходство или различие объектов.

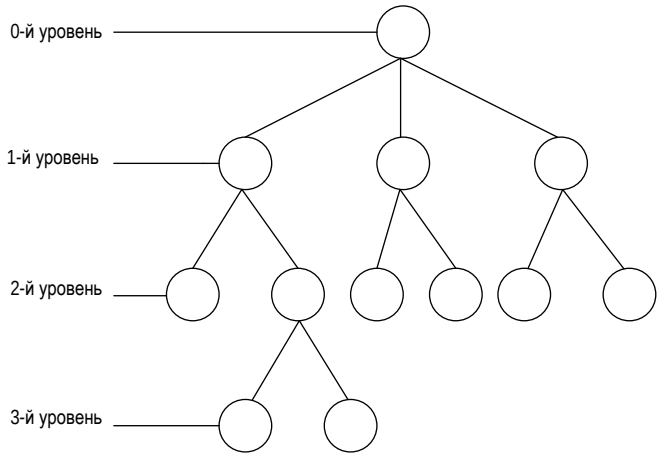
Разработаны три метода классификации объектов: иерархический, фасетный, дескрипторный, которые различаются разной стратегией применения классификационных признаков.

Иерархическая система классификации строится следующим образом:

– исходное множество элементов составляет 0-й уровень и делится в зависимости от выбранного классификационного признака на классы (группировки), которые образуют 1-й уровень;

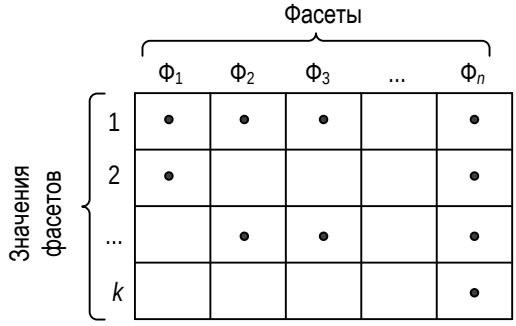
– каждый класс 1-го уровня в соответствии со своим, характерным для него классификационным признаком делится на подклассы, которые образуют 2-й уровень;

– каждый класс 2-го уровня аналогично делится на группы, которые образуют 3-й уровень и т. д.:



В иерархической системе классификации каждый объект на любом уровне должен быть отнесен к одному классу, который характеризуется конкретным значением выбранного классификационного признака. Количество уровней классификации, соответствующее числу признаков, выбранных в качестве основания деления, характеризует глубину классификации.

Фасетная система классификации в отличие от иерархической позволяет выбирать признаки классификации независимо как друг от друга, так и от семантического содержания классифицируемого объекта. Признаки классификации называются фасетами. Каждый фасет ($Ф_i$) содержит совокупность однородных значений данного классификационного признака. Причем значения в фасете могут располагаться в произвольном порядке, хотя предпочтительнее их упорядочение:



Достоинства фасетной системы классификации: возможность создания большой емкости классификации, т. е. использования большого числа признаков классификации и их значений для создания группировок; возможность простой модификации всей системы классификации без изменения структуры существующих группировок. Недостатком фасетной системы классификации является сложность ее построения, так как необходимо учитывать все многообразие классификационных признаков.

Для организации поиска информации, для ведения тезаурусов (словарей) эффективно используется **дескрипторная (описательная) система классификации**, язык которой приближается к естественному языку описания информационных объектов. Особенно широко она используется в библиотечной системе поиска. Суть дескрипторного метода классификации заключается в следующем:

– отбирается совокупность ключевых слов или словосочетаний, описывающих определенную предметную область или совокупность однородных объектов. Причем среди ключевых слов могут находиться синонимы;

– выбранные ключевые слова и словосочетания подвергаются нормализации, т. е. из совокупности синонимов выбирается один или несколько наиболее употребимых;

– создается словарь дескрипторов, т. е. словарь ключевых слов и словосочетаний, отобранных в результате процедуры нормализации.

По другой классификации методы кодирования делятся на последовательные, параллельные и серийно-порядковые.

Последовательный метод кодирования – образование кода классификационной группировки и (или) объекта классификации с использованием кодов последовательно расположенных подчиненных группировок, полученных при иерархическом методе классификации.

Преимуществом этого метода кодирования является простота, а недостатком – негибкая структура.

Параллельный метод кодирования – образование кода классификационной группировки и(или) объекта классификации с использованием кодов независимых группировок.

Этот метод кодирования вследствие его блочной структуры хорошо приспособлен для часто изменяющихся задач. К недостаткам следует отнести избыточность метода.

Порядковый метод кодирования – образование кода из чисел натурального ряда путем сквозной регистрации объектов. Данный метод обладает наибольшей полнотой и простотой для идентификации объектов. Но применение его в чистом виде неэффективно, так как на его основе трудно получить итоги по объектам со сходными признаками.

Серийно-порядковый метод кодирования – образование кода из чисел натурального ряда, закрепление отдельных серий или диапазонов этих чисел за объектами классификации с одинаковыми признаками. Например, с 1-го номера по 10-й закодирована одна группа объектов, с 11-го по 30-й – другая, с 31-го по 100-й – третья группа. Этот метод целесообразно применять для объектов, имеющих два или несколько порядковых признаков.

На практике методы кодирования в чистом виде применяются редко, в основном используются их различные комбинации. Выбор методов кодирования зависит от назначения классификатора и решаемых задач.

Описание хода работы

Для обобщения результатов предпроектного обследования составить отчет в соответствии с правилами организации системы классификации и кодирования. В отчете должны быть представлены следующие пункты.

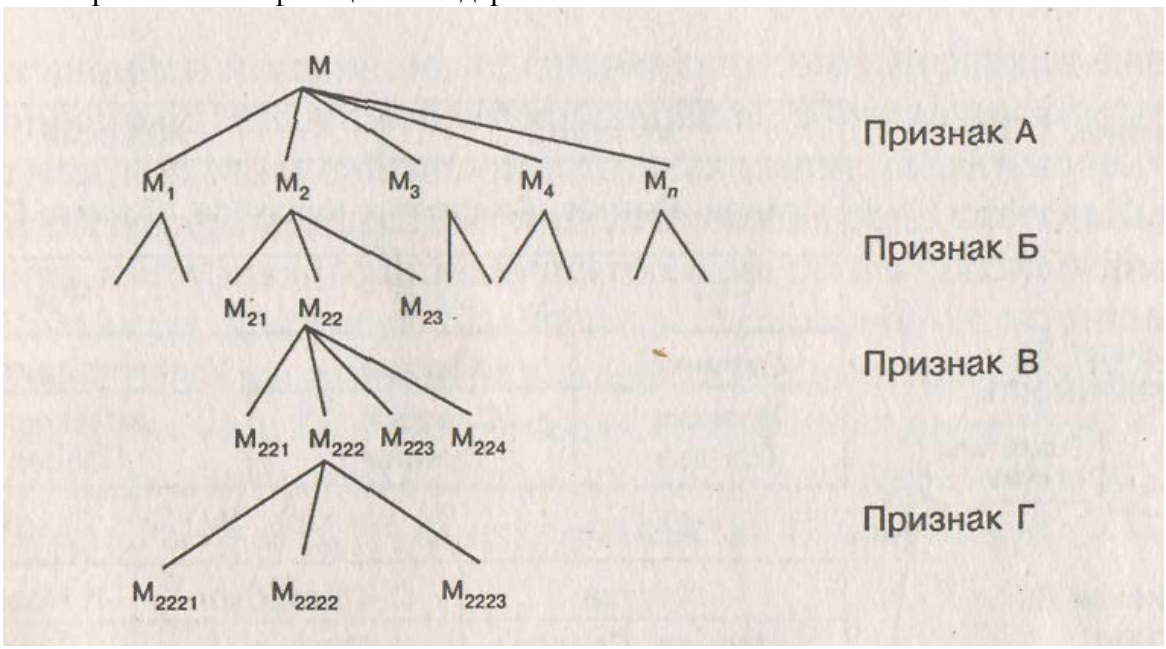
1 Построение иерархического классификатора. В данном разделе должен быть представлен детальный перечень структурных подразделений с указанием взаимодействия между ними с указанием степени подчиненности.

1.1 Для построения иерархического классификатора определить следующие:

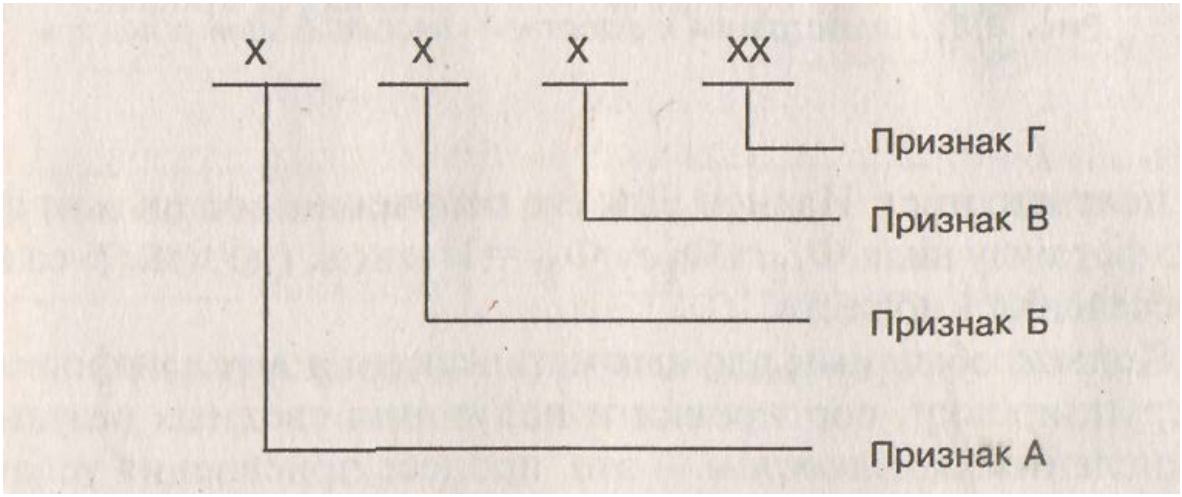
- Определить число признаков, указать их наименование и соподчиненность (например, А (факультет) включает Б (специальности), Б включает В (группы), В включает Г (номера студента в группе)).

- Определить число значений, принимаемых каждым признаком, и выбрать максимальное (например, А принимает максимальное значение 5, Б — 3, В — 4, Г — 25).

- Построить классификационное дерево.



1.2. Построить структуру кода по схеме



Результаты оформить в виде таблицы. В данном разделе должен быть представлен детальный перечень структурных подразделений с указанием взаимодействия между ними с указанием степени подчиненности (если возможно). Для каждого внутреннего подразделения указать перечень должностных лиц с указанием их подчиненности. Данные представить в форме таблиц:

Код структурного подразделения	Тип структурного подразделения	Наименование структурного подразделения
1	Факультет	МТС
1 1	Специальность	Специальность 1
1 2	Специальность	Специальность 2

Код должностного лица	Наименование должностного лица	Обозначение структурного подразделения

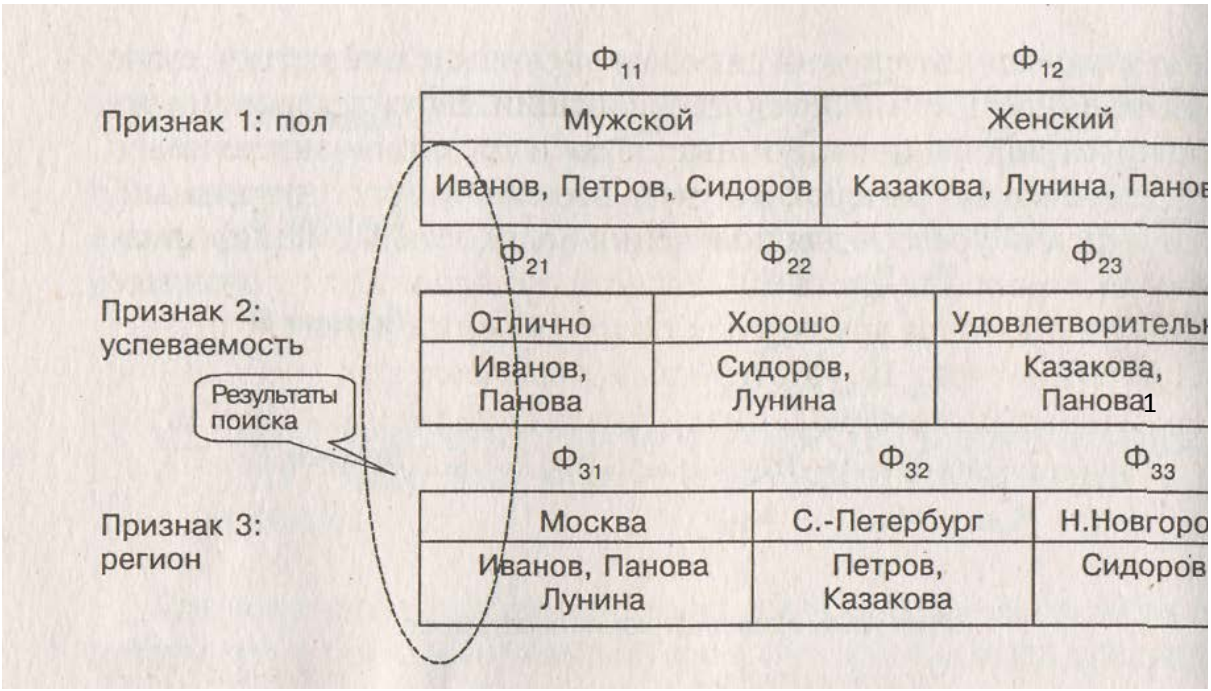
Иерархия подчиненности подразделений и должностных лиц задается порядком их следования в перечне.

Выполнить кодировку всех структурных подразделений и должностных лиц в соответствии с «Системой классификации и кодирования», используемой при обследовании объекта автоматизации.

2. Построение фасетного классификатора

Если между признаками нет иерархической зависимости, то имеет место одноуровневая многопризначная (фасетная) классификация. Она используется для такого деления объектов на классы, при котором ранг всех признаков одинаков. Классы-фасеты получают путем отнесения объектов в классы согласно значениям признаков одновременно.

Например, множество студентов можно разделить по трем признакам: пол, успеваемость и место проживания (регион). Получим независимые классы-фасеты



Полученные таким образом фасеты позволяют с помощью операций пересечения, объединения и др. получить ответы на Различные вопросы.

Например, на вопрос: «Какие студенты мужского пола, проживающие в Москве, учатся на отлично»? — будет получен ответ: Иванов. Для его получения составляют фасетную формулу вида Ф11 ?Ф21 ?Ф31 = Иванов.

После построения фасетного классификатора составить и решить 5 задач с составлением фасетной формулы.

3. Построение дескрипторной системы классификации

Выделить ключевые слова, характеризующие исследуемую предметную область, провести нормализацию и составить словарь дескрипторов.

Привести примеры связей между дескрипторами, которые позволяют расширить область поиска информации:

· синонимические, указывающие некоторую совокупность ключевых слов как синонимы;

· родовидовые, отражающие включение некоторого класса объектов в более представительный класс;

· ассоциативные, соединяющие дескрипторы, обладающие общими свойствами.

Содержание отчета:
- титульный лист
- постановка задачи
- выбор и описание предметной области
- построение иерархического классификатора
- построение фасетного классификатора (составить 5 задач, решаемых с помощью данного классификатора, составить фасетные формулы и решить задачи)
- построение дескрипторной системы классификации
- выводы (сравнительный анализ классификаторов, удобство применения, недостатки)

Лабораторная работа №2

Методология объектно-ориентированного моделирования

Цель работы:

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

Общие сведения:

UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

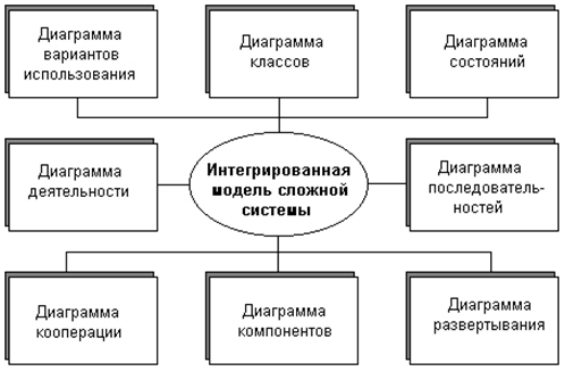


Рис. 1. Интегрированная модель сложной системы в нотации языка UML

1. **Диаграммы вариантов использования**

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. На языке UML вариант использования изображают следующим образом:

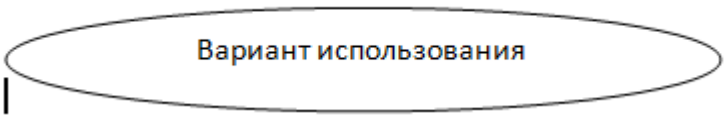


Рис.2. Вариант использования

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур:

Рис.3. Действующее лицо (актер)

Действующие лица делятся на три основных типа:

- пользователи;
- системы;
- другие системы, взаимодействующие с данной;

- время.

Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Связи между вариантами использования и действующими лицами

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

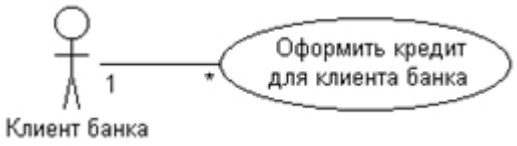


Рис.4. Пример связи коммуникации

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

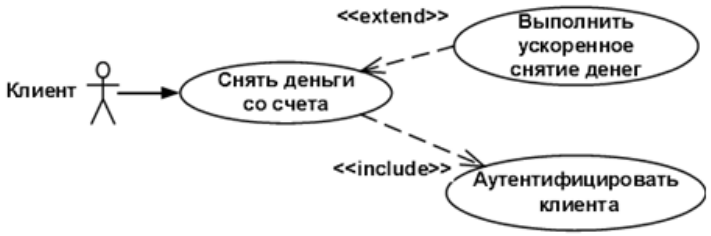


Рис.5. Пример связи включения и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты:

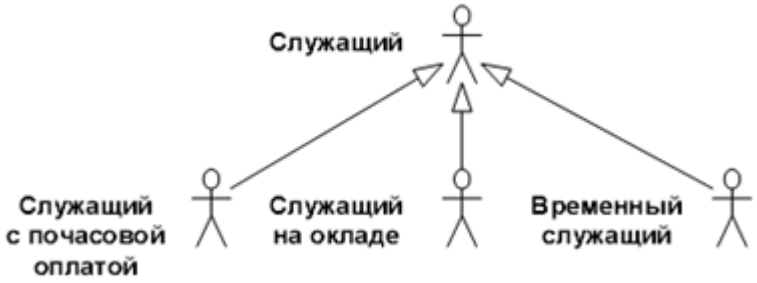
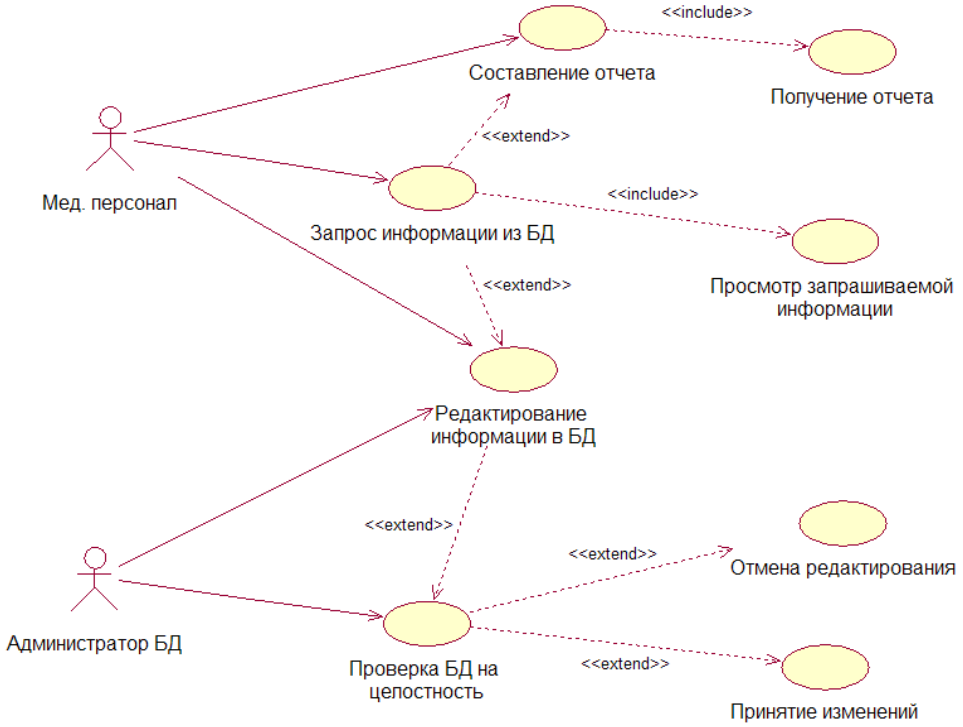
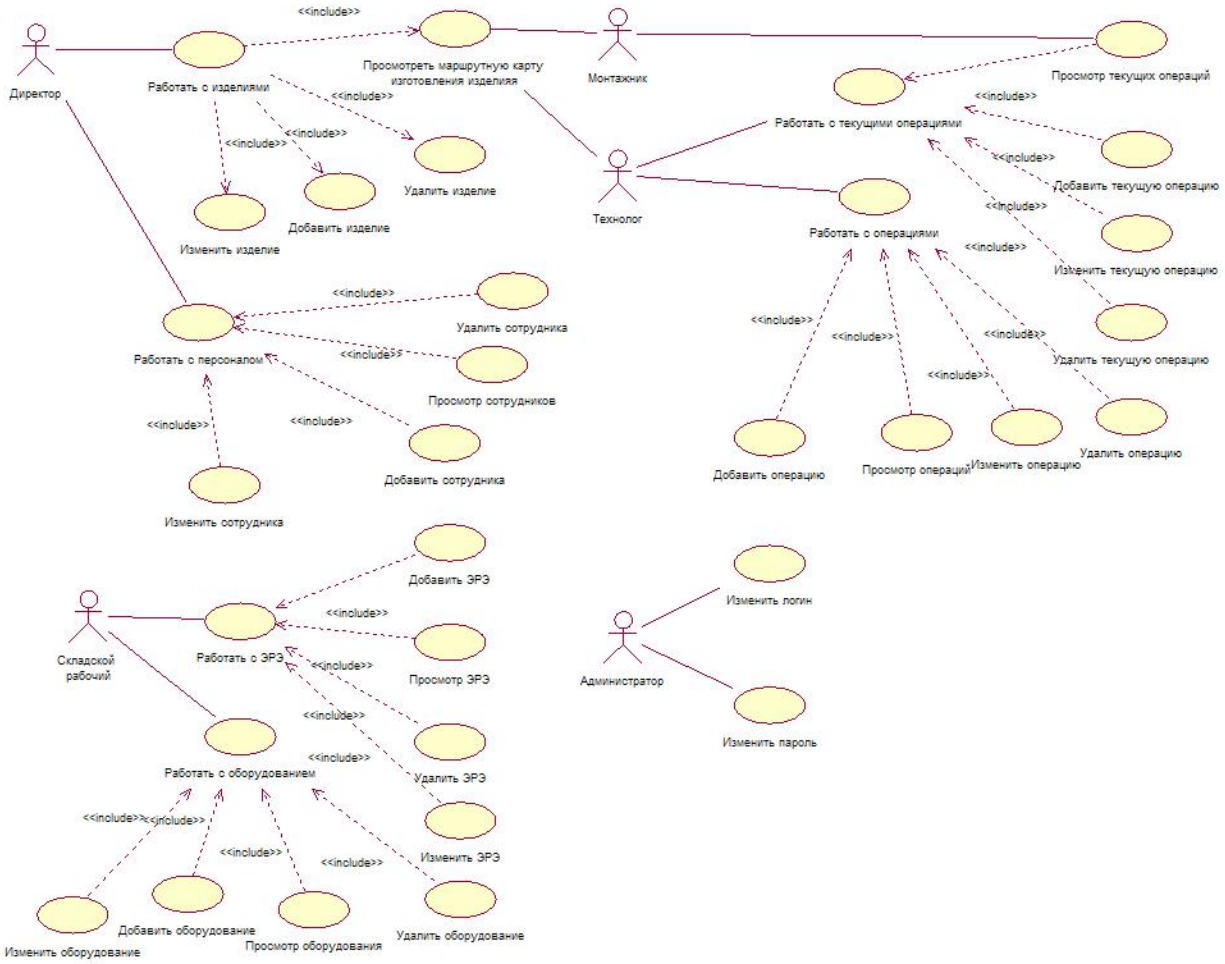


Рис.6. Пример связи обобщения

Примеры диаграмм вариантов использования:
Пример 1.



Пример 2.



2. Диаграмма последовательности.
   Диаграмма последовательности (sequence diagrams)

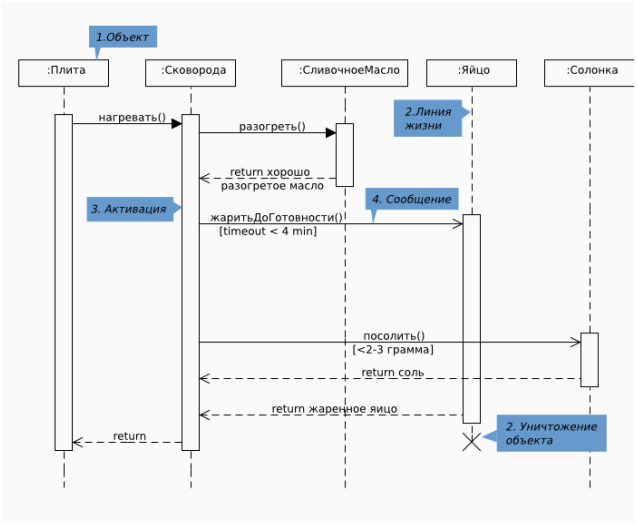
Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

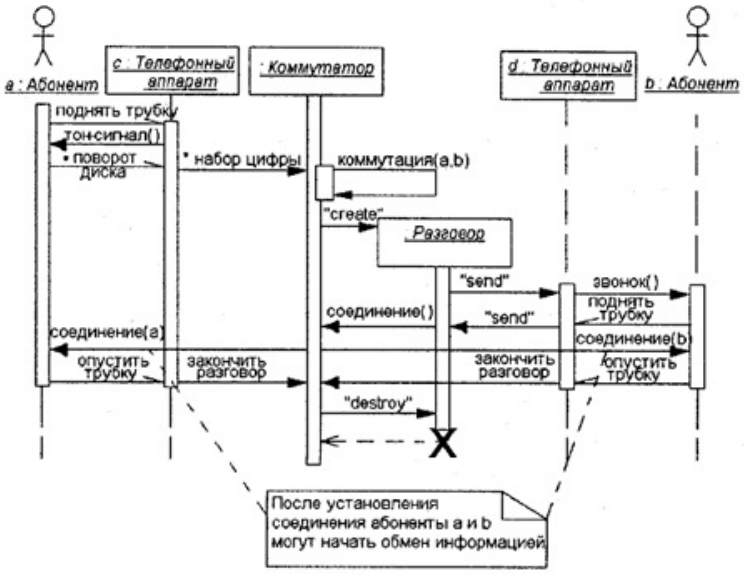
На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни. Примеры диаграммы последовательности:

Пример 1.



Пример 2.



Пример 3.

Содержание отчета:
- Титульный лист;
- Цель работы;
- Постановка задачи;
- Описание выбранной исследуемой области
- Диаграмму вариантов использования;
- Диаграммы последовательности для каждого варианта использования;
- Вывод.

Лабораторная работа №3

«Методология объектно-ориентированного моделирования (кооперативная диаграмма, диаграмма состояний)

1. Цель работы:

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML. Приобретения навыков проектирования кооперативных диаграмм и диаграмм состояний.

2. Теоретический материал.

Кооперативная диаграмма

Диаграмма кооперации (диаграмма коммуникации) – передает ту же информацию, что и диаграмма последовательности.

Основными символами в диаграммах кооперации являются прямоугольник, называемый классификатором роли, и линия, обозначающая сообщение и называемая связью.

Пример построения:

В качестве примера рассмотрим построение диаграммы кооперации для моделирования процесса телефонного разговора с использованием обычной телефонной сети. Напомним, что объектами в этом примере являются два абонента a и b, два телефонных аппарата c и d, коммутатор и сам разговор как объект моделирования.

На начальном этапе изобразим все объекты и связи между ними на диаграмме кооперации при помощи соответствующих обозначений (рис. 1).

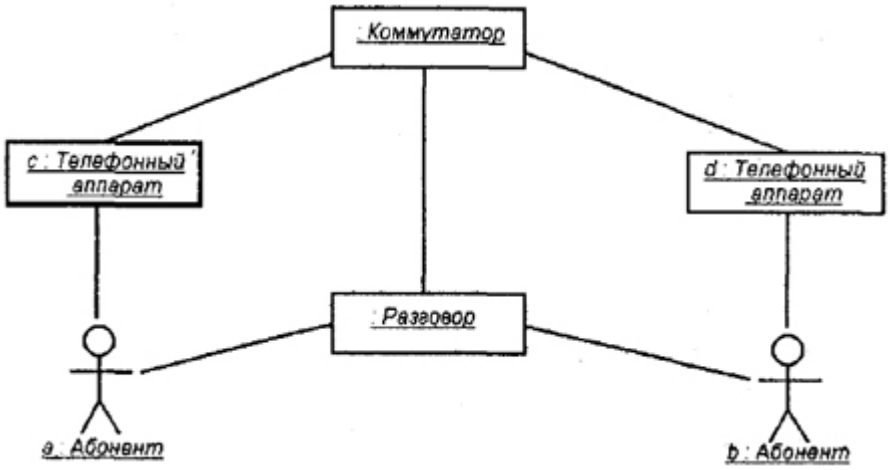


Рис. 1. Начальный фрагмент диаграммы кооперации для примера моделирования обычного телефонного разговора.

В последующем необходимо специфицировать все связи на этой диаграмме, указав на их концах необходимую информацию в форме ролей связей. Дополненный таким образом вариант диаграммы кооперации изображен ниже (рис. 2). Заметим, что для объекта «Разговор» указано помеченное значение {transient}, которое означает, что этот объект создается в процессе выполнения объемлющего процесса и уничтожается до его завершения. Напомним, что помеченные значения (tagged values) являются стандартными элементами языка UML.

Рис. 2. фрагмент диаграммы кооперации, дополненный стереотипами ролей связей, именами ассоциаций и помеченным значением объекта

На диаграмму кооперации необходимо нанести все сообщения, указав их порядок и семантические особенности. Окончательный фрагмент диаграммы кооперации изображен на рис. 3 и содержит модель кооперации только для начала разговора. Эта диаграмма может быть дополнена сообщениями, необходимыми для окончания разговора.

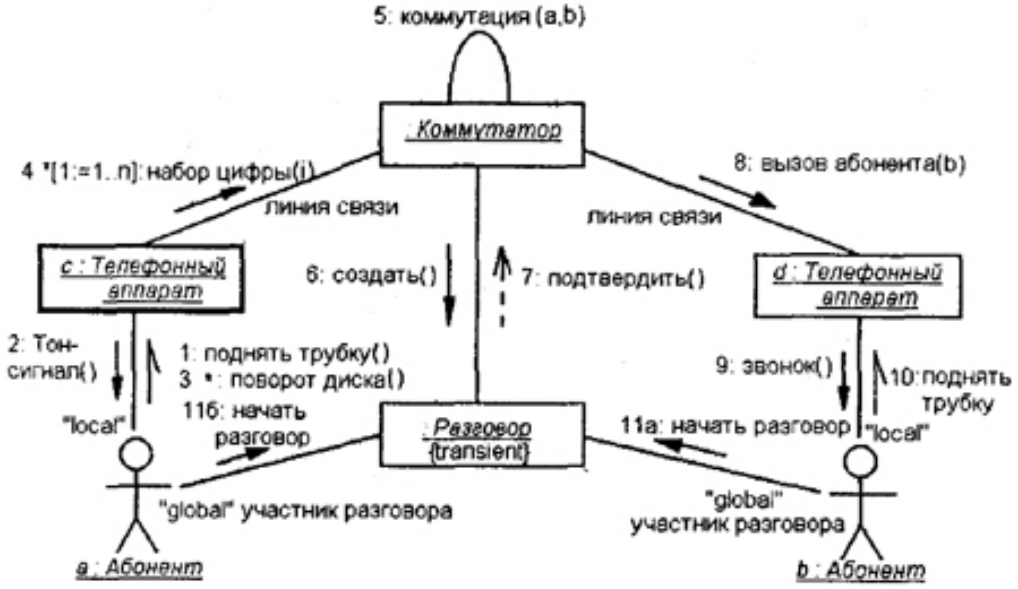
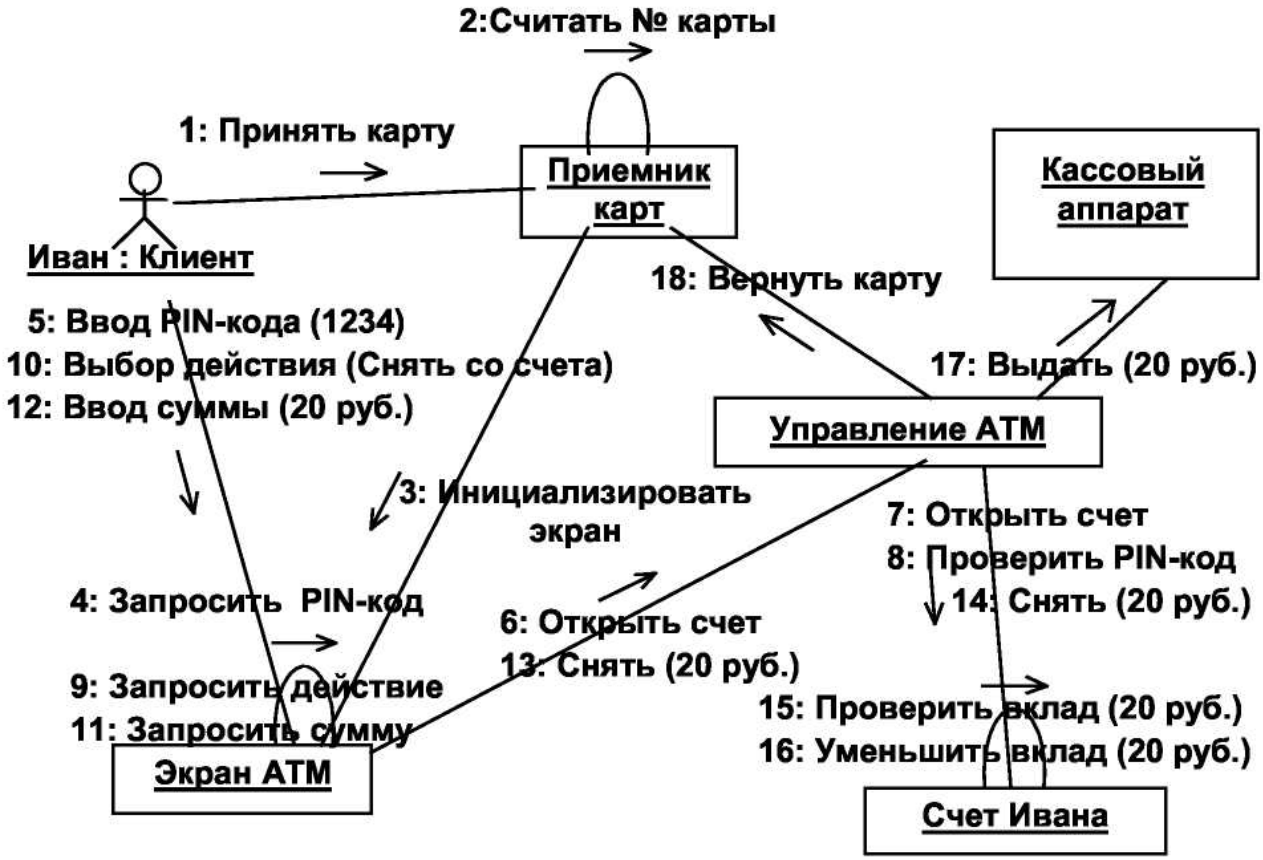


Рис. 3. Окончательный вариант диаграммы кооперации для моделирования телефонного разговора.

Диаграмма кооперации для примера с телефонным разговором не содержит ни временных особенностей передачи сообщений, ни особенностей жизненного цикла участвующих в данной кооперации объектов. Поэтому может быть принято решение о том, что она является избыточной при наличии построенной диаграммы последовательности. Этот факт не вызывает сомнений в тех случаях, когда структура взаимодействующих объектов является достаточно тривиальной.

Если же взаимодействующие объекты образуют между собой различные типы отношений-ассоциаций (композиция, агрегация), то диаграмма кооперации оказывается необходимым представлением модели на всех ее уровнях.

Пример диаграммы кооперации



Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На диаграмме имеются два специальных состояния – начальное (start) и

конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).

С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельность

Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входное действие

Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие

Выходное действие (exit action) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.

Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.

Поведение объекта во время деятельности, при входных и выходных действиях может включать отправку события другому объекту. В этом случае описанию деятельности, входного действия или выходного действия предшествует знак « ^ ».

Соответствующая строка на диаграмме выглядит как

Do: ^Цель.Событие (Аргументы)

Цель – это объект, получающий событие, Событие – это посылаемое

сообщение, а Аргументы являются параметрами посылаемого сообщения.

Деятельность может также выполняться в результате получения объектом некоторого события. При получении некоторого события выполняется определенная деятельность.

Переходом (Transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим.

Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.

У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

События

Событие (event) – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.

На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия

Ограждающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.

Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.

Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действие

Действием (action) является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия

показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

Событие или действие могут быть поведением внутри объекта, а могут представлять собой сообщение, посылаемое другому объекту. Если событие или действие посылается другому объекту, перед ним на диаграмме помещают знак « ^ ».

Пример диаграммы состояний представлен на рисунке 1.

Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

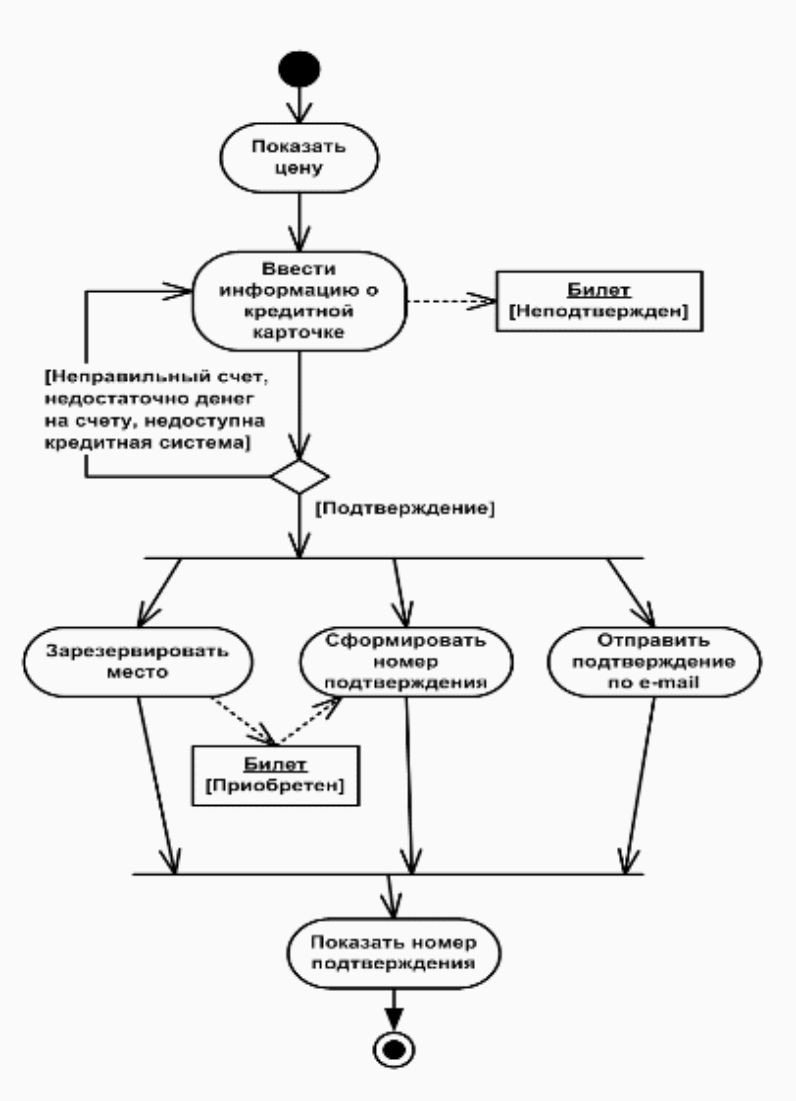
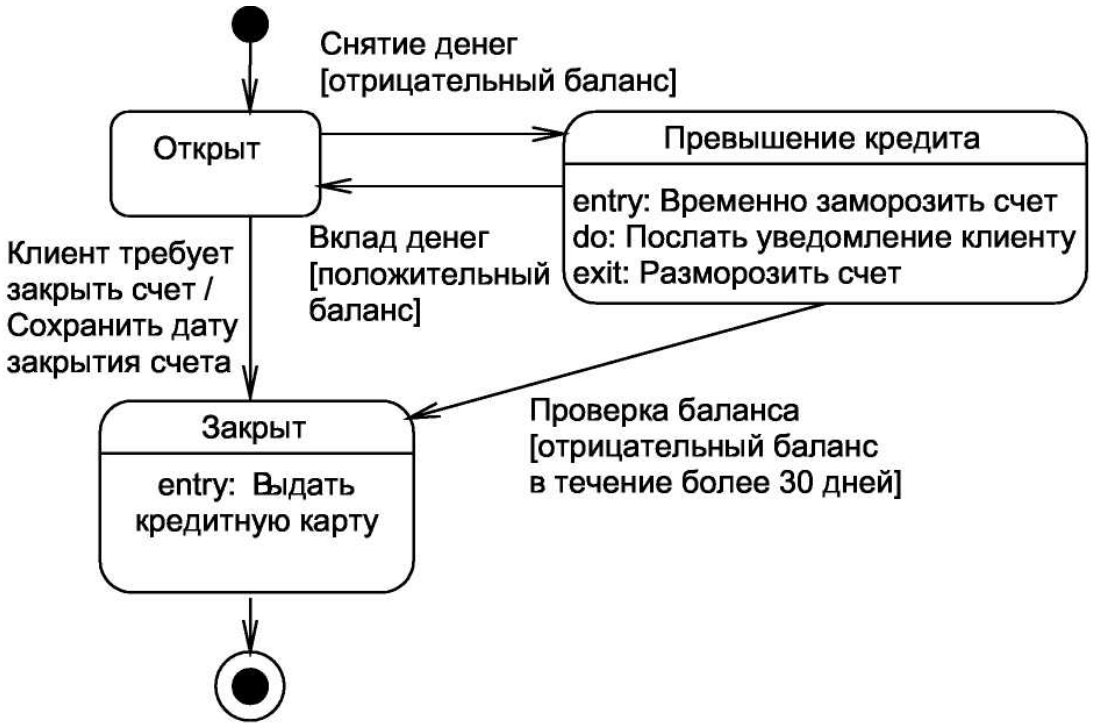
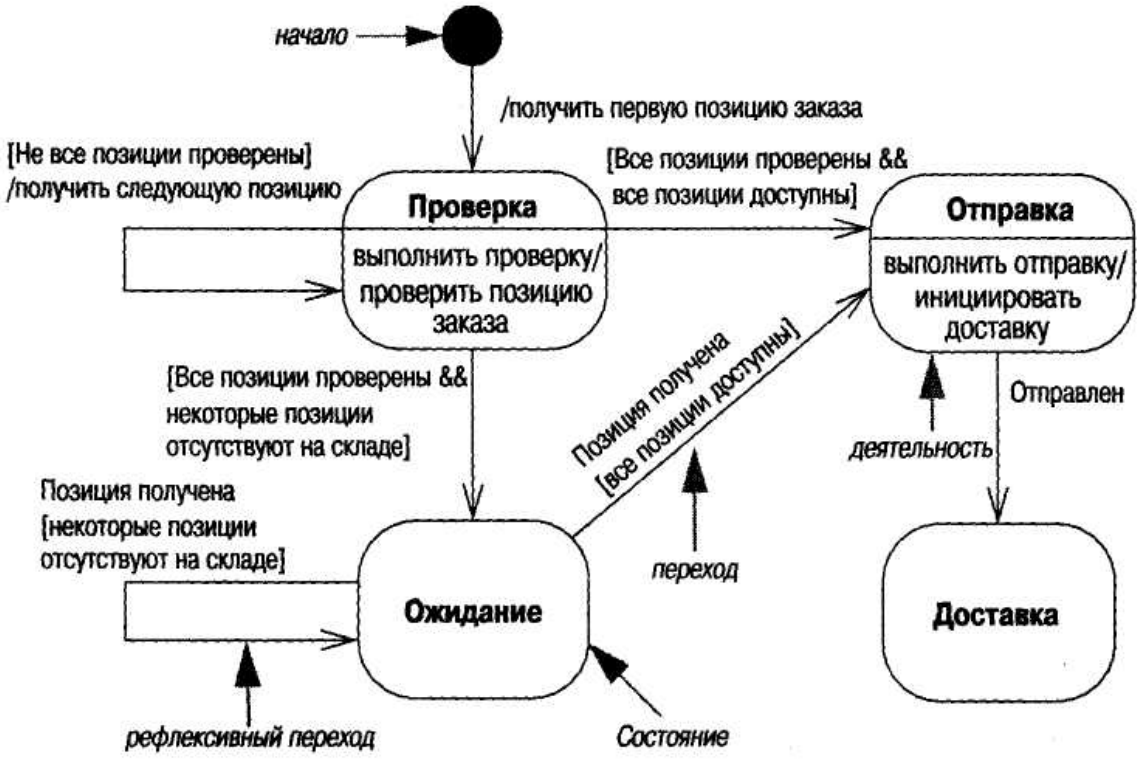


Рис. 1 Пример диаграммы состояний

Примеры диаграмм состояний:





Содержание отчета:

- Титульный лист;
- Цель работы;
- Постановка задачи;
- Описание выбранной исследуемой области;
- Одна диаграмма кооперации;
- Минимум 2 диаграммы состояния;
- Выводы.

Лабораторная работа №4
«Методология объектно-ориентированного моделирования (диаграмма классов)

1. Цель работы:

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

2. Задание.

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

3. Теоретический материал и пример выполнения работы.

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

-для моделирования словаря системы: предполагает принятие решения о том, какие абстракции являются частью системы, а какие - нет. С помощью диаграмм классов можно определить эти абстракции и их обязанности;

-для моделирования простых кооперaций. Кооперация - это сообщество классов, интерфейсов и других элементов, работающих совместно для обеспечения некоторого кооперативного поведения;

-для моделирования логической схемы базы данных.

Существуют три различные точки зрения на построение диаграмм классов или любой другой модели:

-концептуальная точка зрения - диаграммы классов служат для представления понятий изучаемой предметной области. Эти понятия будут соответствовать реализующим их классам, но прямое соответствие может отсутствовать. Концептуальная модель может иметь слабое отношение или вообще не иметь никакого отношения к реализующему ее программному обеспечению, поэтому ее можно рассматривать без привязки к какому-то языку программирования;

-точка зрения спецификации - рассматривается программная система, при этом рассматривается только ее интерфейсы, но не реализация;

-точка зрения реализации - классы диаграммы соответствуют реальным

классам программной системы.

Пример выполнения работы.

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

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

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

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

- самого заказа;

- клиента, который делает заказ;

- комплектующих изделий, которые входят в заказ.

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

Класс *Client:*

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

Класс *Order:*

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

Класс *OrderItem:*

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

Класс *ComponentPart:*

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

Результат создания классов-сущностей показан на рис. 1:

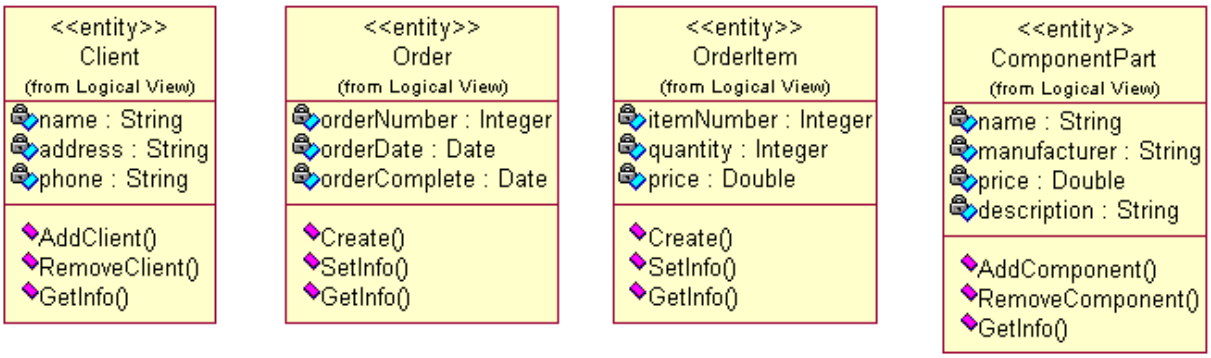


Рисунок 1. Созданные классы-сущности

Добавим отношения между классами (рис. 2):
- класс Client и Order - отношение ассоциации, поскольку данные два класса просто связаны друг с другом и никакие другие типы связей здесь применить нельзя. Один клиент может сделать несколько заказов, каждый заказ поступает только от одного клиента, поэтому кратность связи со стороны класса Client - 1, со стороны Order - 1..n;

- класс Order и OrderItem - отношение композиции, поскольку строка заказа является частью заказа, и без него существовать не может. В один заказ может входить несколько строк заказа, строка заказа относится только к одному заказу, поэтому кратность связи со стороны Order - 1, со стороны OrderItem - 1..n;

- класс OrderItem и ComponentPart - отношение агрегации, поскольку комплектующие изделия являются частями строки заказа, но и те, и другие, являются самостоятельными классами. Одно комплектующее изделие может входить во много строк заказа, в одну строку заказа входит только одно комплектующее изделие, поэтому кратность связи со стороны ComponentPart - 1, со стороны OrderItem - 1..n.

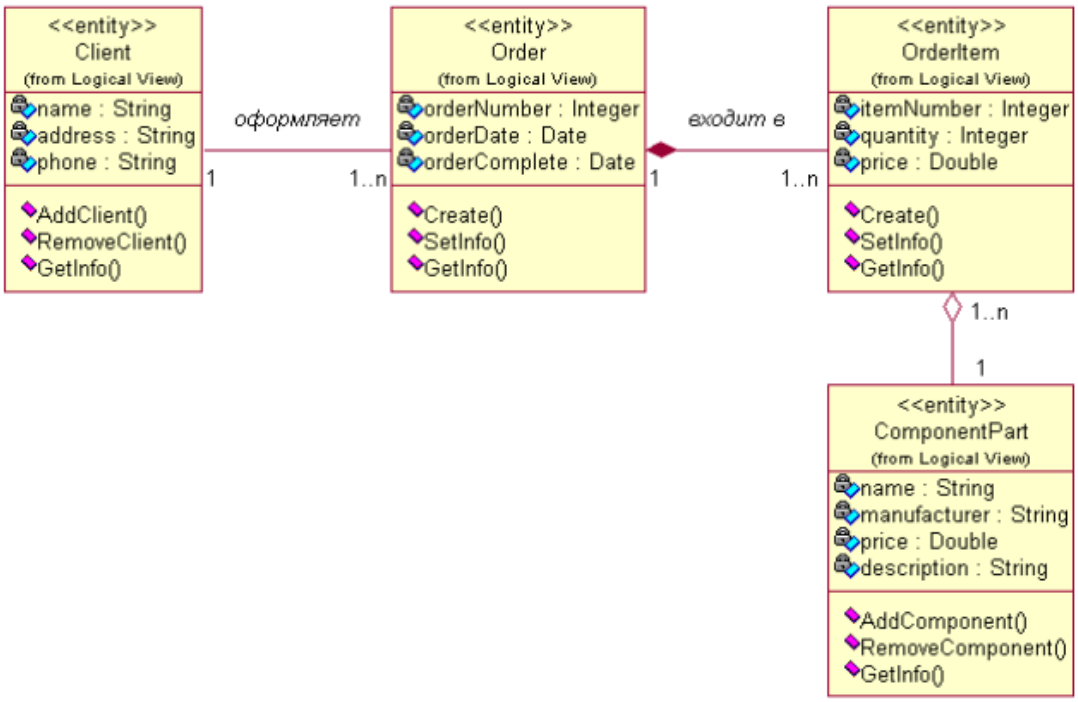


Рисунок 2. Классы-сущности и отношения между ними

Содержание отчета:

- Титульный лист;
- Цель работы;
- Постановка задачи;
- Описание каждого этапа выполнения работы;
- Итоговая диаграмма классов;
- Выводы.

Защита работы

Защита проводится в форме собеседования по теме работы (и/или в форме устного доклада с последующими ответами на вопросы слушателей и преподавателя), во время которого студент должен продемонстрировать понимание сути работы и ответить на вопросы преподавателя по существу выполненной работы.

Итоговый результат выставляется с учетом содержательности работы, качества оформления и защиты.