Лабораторная работа № 2

Классификация, моделирование предметной области

Цели работы: 1. Познакомиться с процессом классификации на примере моделирования классов предметной области. Изучить базовые понятия ООП.
2. Познакомиться с диаграммами классов и взаимодействия UML.
3. Рассмотреть особенности моделирования предметной области в контексте унифицированного процесса (UP).

1. Некоторые теоретические сведения (понятия)

Объект - некая сущность реального мира или концептуальная сущность, либо – программная единица, состоящая из **атрибутов** (полей) и **методов** для их обработки. (характеризуется состоянием, поведением, и уникальностью). Состояние определяется атрибутами (память состояний объекта), поведение – методами.

Класс – шаблон для создания объектов или тип объектов (упрощенно).

Концептуальный класс - класс, используемый для моделирования предметной области задачи. Соответствует сущности реальноо мира, идеальному, техническому объекту. В отличие от программных классов (в модели проектирования или реализации) концептуальный класс обычно не содержит методов. Можно сказать, что концептуальный класс соотвеотствует сущности в модели "Сущность – связь" (ER – диаграммы), используемой при проектировании БД.

Программный класс – описание структурированного типа для создания объектов. Используется собственно для создания программ, при проектировании, отображается на диаграммах программных классов в моделях проектирования и реализации. Как правило, имеет методы.

Инкапсуляция - дословно "сокрытие". Можно толковать инкапсуляцию двояко : это, с одной стороны, абстракция "активных данных", то есть соединение в одном объекте и данных, и методов их обработки, с другой стороны – сокрытие данных (полей, атрибутов) с помощью методов доступа и других методов, с помощью которых и можно только менять атрибуты и таким образом состояние объекта. Единственный способ изменить состояние объекта – передача сообщений.

Наследование – установление отношения "родитель-потомок" между классами. Класс – наследник обладает всеми свойствами наследуемого класса, то есть списком атрибутов и методов, но как правило добавляет что-то свое. Наследник является наследуемым классом тоже, поэтому такой тип оношения называют отношение "is-a".

Полиморфизм - ("много форм" – дословно) – определяет различные формы реализации одноименного действия. С одной стороны, полиморфизм – это наличие в иерархии методов с одинаковой **сигнатурой**, которые по-разному выполняются разными наследниками (в т.ч. - простой статический полиморфизм), с другой – вызов метода для неявно определенного объекта (указывается тип базового класса, который во время выполнения может подменяться разными наследниками), то есть данное понятие связано с поздним связыванием.

Делегирование - передача сообщения известному объекту для выполнения некоторых действий, в частности, для реализации всех или части собственных обязанностей.

Отношение между классами - двунаправленная семантическая связь**.**

Виды отношений. Отношения делятся на две основные группы : отношения типа ассоциаций и отношения типа наследования ("is – a"). Отношения типа ассоциаций

позволяют отправлять сообщения посредством делегирования, а отношения типа наследования – по иерархии наследования (?! - правильно ли это ?!).

К отношениям типа ассоциаций относятся собственно ассоциация, агрегация, композиция и зависимость.

Ассоциация – наиболее общий тип оношений первого рода, связанных с передачей сообщений ассоциированным классам путем делегирования. Реализуется с помощью включения ссылок на ассоциированные классы в список атрибутов класса.

Агрегация – отношение типа “часть-целое” (“part of” - логическое включение)

Композиция – отношение типа “часть-целое” (“part of” - физическое включение)

Отношения агрегации и композиции в основном различаются по критерию времени жизни объектов и возможности их отдельного существования.

Зависимость – самый слабый тип отношений. Обычно связан с косвенной ссылкой (отсутствие ссылки в списке атрибутов класса, например, присутствует в списке параметров методов, создается внутри тела метода, возвращается при каком-то вызове и пр.)

К отношениям типа наследования относят собственно наследование и реализацию.

Наследование – второй тип отношений, связан с передачей сообщений путем передачи по иерархии.

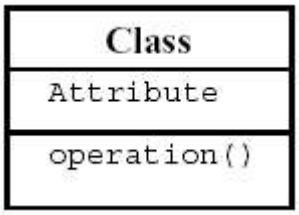
Реализация – наследование от абстрактного класса (интерфейса) с реализацией его неопределенных методов (чисто виртуальных функций в C++)**.**

Интерфейс - синоним **абстрактного класса**, то есть класса, у которого имеется по крайней мере один метод без реализации (то есть только **сигнатура**, или – совокупность имени метода, возвращаемого типа, количества, порядка и типов параметров). Таким образом, у абстрактного класса не может быть экземпляров, то есть объектов с типом данного класса. Экземпляры могут быть у наследников такого класса, реализующих недостающие методы. С другой стороны, допускаются ссылки на объекты с типом абстрактного базового класса и вызов методов абстрактного класса. Такая возможность и объясняет роль абстрактного класса, который выступает в качестве программного интерфейса : для одного и того же интерфейса допускается множество реализаций.

UP – Unified Process - унифицированный процесс, один из вариантов итеративного каскадного процесса разработки программ. Наиболее известной версией UP является RUP - Rational UP, разработанный компанией Rational Software и поддерживаемый в ее программных средствах, самым известным из которых является CASE – средство Rationa Rose (торговая марка Rational принадлежит в настоящее время компании IBM).

Диаграмма классов - диаграмма, на которой отображаются классы и их отношения. Является основной диаграммой UML. На диаграмме классов используются следующие основные обозначения :

Класс – прямоугольник, разделенный на три секции (раздела), в первом приводится имя класса, во втором – список атрибутов, в третьем – список методов.



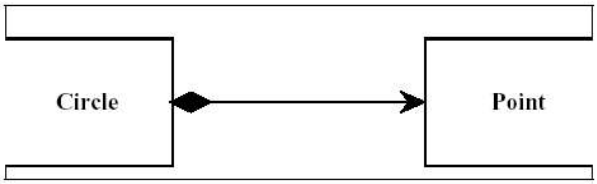
Допускается опускать раздел атрибутов, раздел методов или оба раздела. Для абстрактных классов указывается шаблон <<abstract>> или <<interface>> после имени класса, для абстрактных классов обычно также имя класса указывается *наклонным шрифтом*.

Для атрибутов может указываться тип и видимость (+ public, - private, # protected).

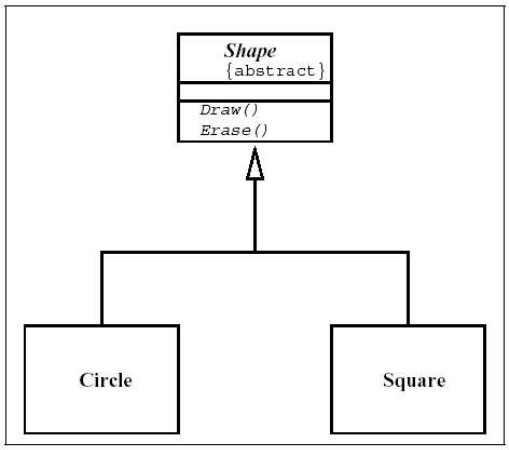
Для методов также могут указываться тип, список параметров и видимость.

Отношения :

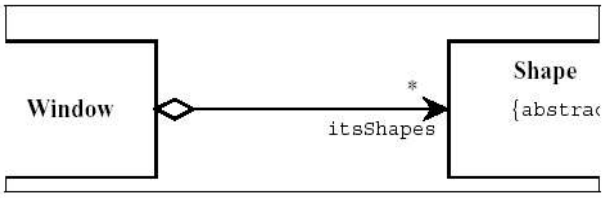
Композиция



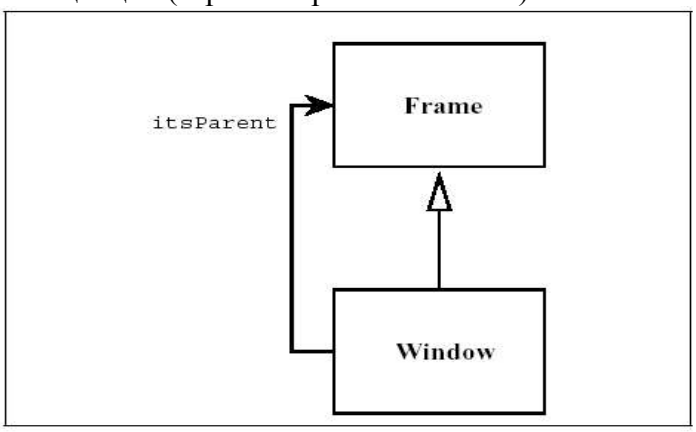
Наследование



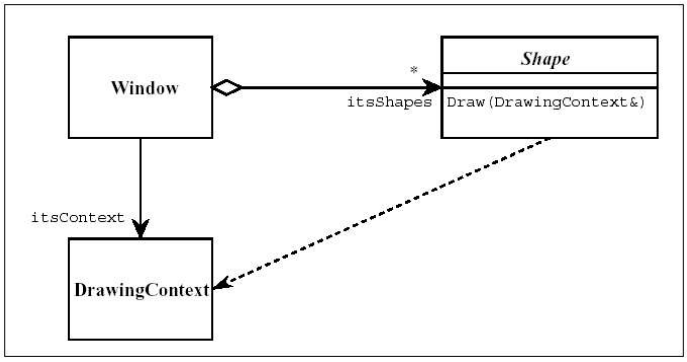
Агрегация



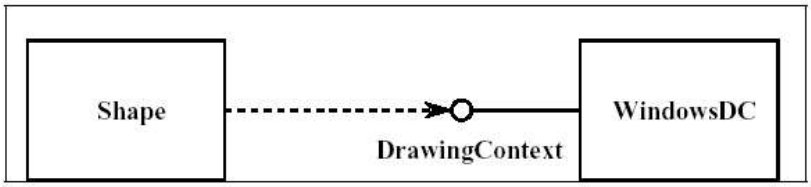
Ассоциация (стрелка с ролью itsParent)



Зависимость (пунктирная линия)



Реализация (здесь же показывается способ отображения интерфейса – кружок).



Другой вариант отображения реализации – пунктирная линия со стрелкой, как у наследования (при условии, что сам интерфейс отображается обычным прямоугольником).

При обозначении отношений типа ассоциаций дополнительно указывается направление отношений (с помощью стрелок), их наименование, роли участвующих в отношении классов, а также арности ролей (* - несколько, 0..* - ни одного или несоклько, 1..2 – один или два и пр.).

Диаграммы взаимодействия - два вида диаграмм UML (диаграмма последовательности и диаграмма кооперации), которые используются для спецификации динамики обмена сообщениями между объектами классов при решении определенных задач. Именно взаимодействие (кооперация) классов реализует прецеденты, для которых предназначена программа. Можно сказать, что диаграммы взаимодействия и классов являются основными в объектном проектировании.

Классификация - процедура выделения классов, их отношений и атрибутов из определенного неформального описания предметной области, например, из описания прецедентов. Сами классы выделяют, чаще всего, используя метод выделения существительных, либо на основании анализа часто встречающихся категорий классов.

Примеры категорий *классов* :

Физические или материальные объекты
Спецификации и описания
Места
Транзакции
Элементы транзакций
Роли людей
Контейнеры других объектов
Содержимое контейнеров
Компьютеры или внешние технические системы
Абстрактные понятия
Организации
События

Процессы
Правила и политики
Каталоги
Записи деятельности
Документы
Финансовые инструменты и службы

Примеры категорий *ассоциаций* :

A является частью B (логической /физической)
A содержится в B.
A является описанием B
A является элементом транзакции B.
A известен в B
A является членом B
A является подразделением B
A использует / управляет B
A взаимодействует с B
A следует за B
A является собственностью B
A связано с B

В качестве *атрибутов* классов обычно выделяют существительные, являющиеся атомарными характеристиками сущностей.

Модель классов предметной области – диаграмма концептуальных классов.

2. Примеры для изучения

2.1 Пример(ы) классификации и создания модели классов предметной области

2.1.1 Пример классификации и построения модели предметной области из книги К. Лармана (POS – система NextGen)

Эту информацию можно почерпнуть из описания прецедентов, выделяя существительные из текста описания, например :

Основной успешный сценарий (или основной процесс)

1. **Покупатель** подходит к **кассовому аппарату** POS-системы с выбранными **товарами**.
2. **Кассир** открывает новую **продажу**.
3. Кассир вводит идентификатор товара.
4. Система записывает наименование товара и выдает его **описание**, цену и общую стоимость. Цена вычисляется на основе набора правил.
   Кассир повторяет действия, описанные в пп. 3-4, для каждого наименования товара.
5. Система вычисляет общую стоимость покупки с налогом.
6. Кассир сообщает покупателю общую стоимость и предлагает оплатить покупку.
7. Покупатель оплачивает покупку, система обрабатывает **платеж.**
8. Система регистрирует продажу и отправляет информацию о ней внешней бухгалтерской системе (для обновления бухгалтерских документов и начисления комиссионных) и системе складского учета (для обновления данных).
9. Система выдает товарный чек.
10. Покупатель покидает магазин с чеком и товарами (если он что-то купил).

Получаем список кандидатов на роль концептуальных классов :

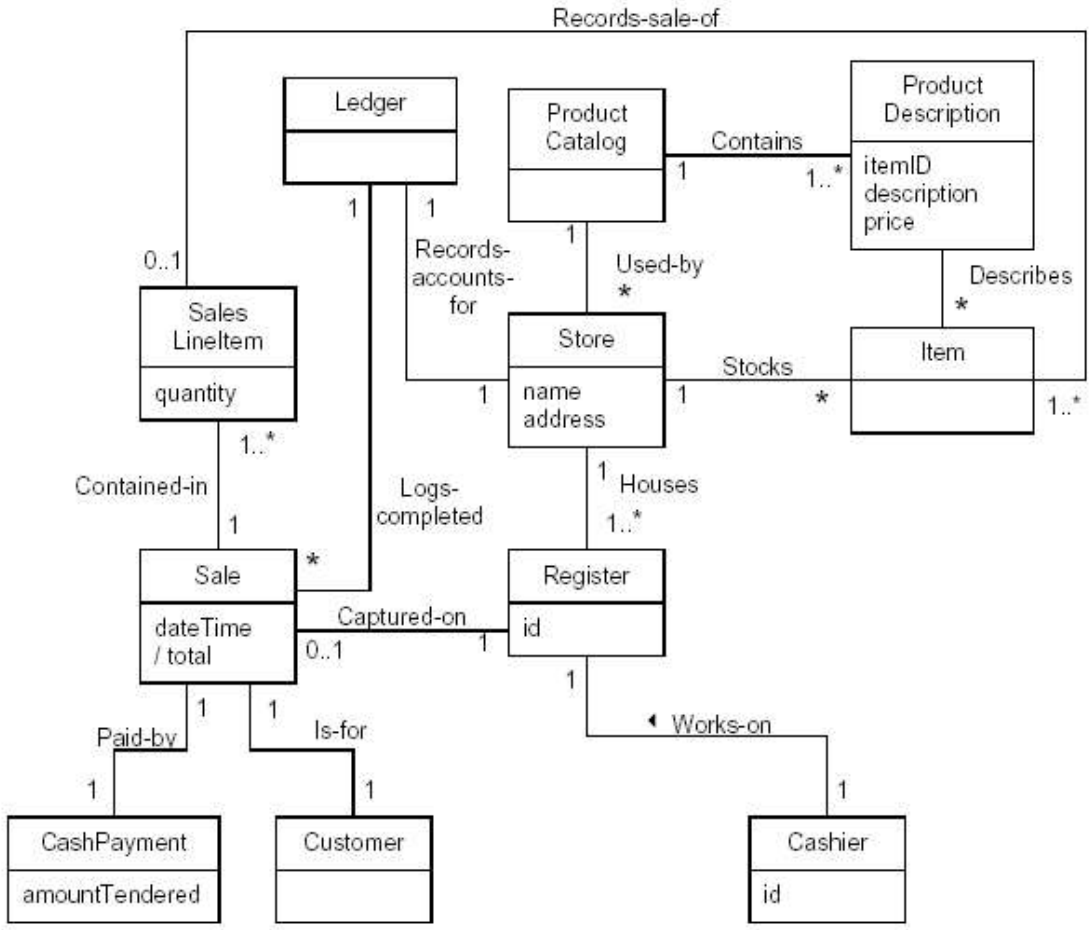
1. Покупатель
2. Кассовый аппарат
3. Товар
4. Кассир
5. Продажа
6. Описание товара
7. Платеж

В список не попали некоторые существительные, такие как POS-система, идентификатор, наименование и цена товара, стоимость, покупка, налог, бухгалтерская система, складская система, товарный чек, магазин и пр. Они не попали в список по разным причинам, в частности : POS-система это сущность, характеризующая всю создаваемую систему, бухгалтерская и складская системы являются вспомогательными исполнителями и внешними системами по оношению к NextGen, идентификатор, наименование и цена товара являются атрибутами товара, либо – описания товара, а не самостоятельными классами, покупка является синонимом продажи (то, что для покупателя покупка, для магазина – продажа), стоимость и налог являются рассчитываемыми величинами и относятся к продаже (их можно рассматривать как атрибуты, в программных классах – как методы; вполне возможно, что, допустим, налог трансформируется в будущем в класс, но пока мы его не выделяем), товарный чек это пример отчета, который составляется по данным других классов (Продажа и платеж). Магазин можно не включать, если он один в системе. В данном случае после дополнительного анализа, в том числе анализа прецедентов и категорий сущностей решено было добавить Магазин как дополнительный концептуальный класс, являющийся к тому же контейнером товаров и кассовых аппаратов. Кроме того, добавлен контейнеры Каталог описаний товаров и, наоборот, элемент контейнера Продажа – строка продажи. (Кроме того, решено было добавить в модель Кассовую книгу для регистрации всех продаж.)
Таким образом получены следующие концептуальные классы :

1) Item – товар,
2) SalesLineItem – строка товара в списке покупок,
3) Sale – Продажа (список покупок),
4) Payment – Платеж,
5) Store – Магазин (склад),
6) Register – касса (кассовый аппарат),
7) Product Description - Описание товара,
8) Product Catalog – Список описаний товаров – Каталог товаров,
9) Cashier – кассир,
10) Customer – покупатель,
11) Ledger – кассовая книга (?!)

После добавления важных ассоциаций и атрибутов (в результате анализа тех же прецедентов примерно по следующим правилам : ассоциации выделяются в результате анализа глагольных фраз и стандартных категорий ассоциаций (“является частью”, “включает”, “состоит из”, “создает” и пр.), в качестве атрибутов выбирают важные атомарные данные о сущности) получена диаграмма классов предметной области, представленая ниже.

Диаграмма отражает следующие ассоциации: Товар хранится на Складе (в Магазине), в Магазине имеется несколько Кассовых апапаратов, в Кассовом аппарате регистрируется Покупка, которая состоит из нескольких Строк товара, которые отражают сведения о покупке определенного количества Товаров. Покупка оплачивается Платежом. Наиболее важные атрибуты классов также представлены на диаграмме : название и адрес магазина, идентификатор кассира, идентификатор, название и цена товара, идентификатор кассы, количество единиц товара в строке покупки, дата и сумма покупки, сумма платежа и пр.



2.1.2 Пример постановки задачи, классификации и построения модели для утилиты анализа Log- файлов программы Tmeter (весьма и весьма условный !)

TMeterLA - TMeter Log Analyzer

Что должна делать программа (кратко)

Для заданного каталога (или маски) log-файлов выдается таблица, в которой для каждого файла (т.е. даты) указывается общее кол-во принятых и отправленных байт. В конце таблицы выдаются итоги по всей таблице. Интерфейс - в двух вариантах : командная строка, либо – MFC. В перспективе необходимо предусмотреть настройку на разные типы файлов и через разные библиотеки (например, через ADO - для ускорения работы).

В дальнейшем - ограничения по времени суток, по IP, по протоколам/портам, еще по чему-нибудь, построение графиков, удаленный доступ к LOG-файлам и т.д.

Модель прецедентов (вариантов использования) для проекта «TMeterLA» на начальном этапе

Выделяем три группы **Исполнителей** (Actors) :

1) **Основные исполнители** (их задачи выполняются с использованием системы).

В данном случае это :

1. Администратор сети.

Возможно, к основным исполнителям можно отнести также :

2. Автоматизированная система “TMeter” (поставщик данных для TMeterLA).

2) **Вспомогательные исполнители** обслуживают систему.
   3. Системный программист.

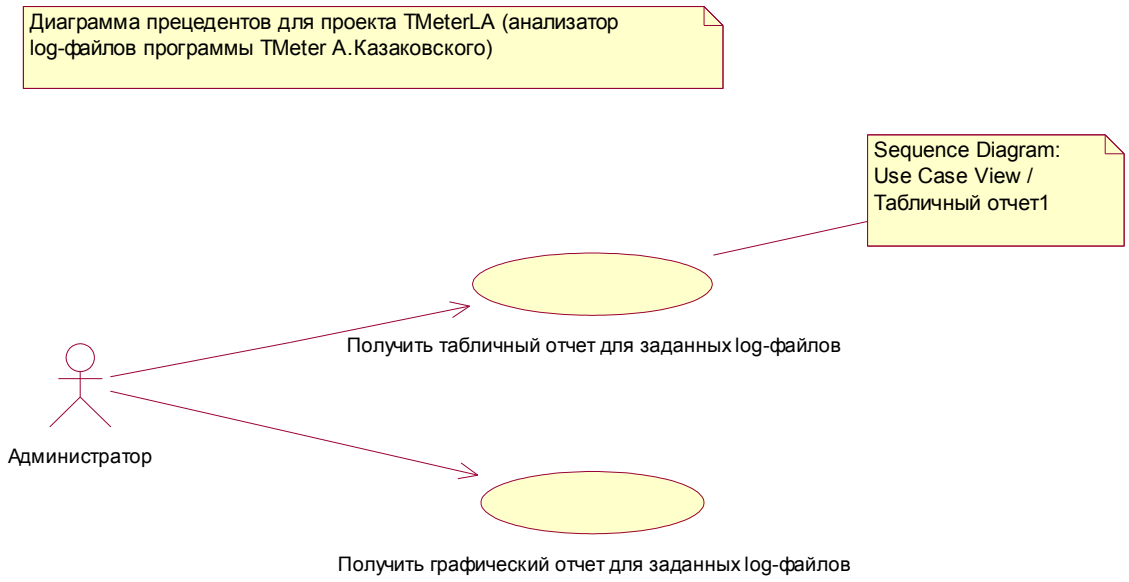
3) **«Закулисные» исполнители**. Таких пока нет.

У всех исполнителей есть какие-либо задачи.

Но если посмотреть внимательно, то к Исполнителям в принципе пока можно отнести только Администратора.

Список задач Администратора :

1. **Получить табличный отчет для заданных файлов.**
2. **Получить графический отчет для заданных файлов.**
3. **?**



Рассмотрим описание прецедента **Получить табличный отчет для заданных файлов** в сокращенном формате :

Получить табличный отчет для заданных файлов

Администратор указывает место расположения log-файлов, маску файлов в виде текста или выбирая из списка. При необходимости Администратор также указывает необходимые ограничения на выборку. Затем Администратор делает запрос. Система выдает затребованный отчет в виде таблицы на консоль / в файл / в окно (в зависимости от интерфейса).

Рассмотрим описание прецедента **Получить табличный отчет для заданных файлов** в подробном формате :

Получить табличный отчет для заданных файлов

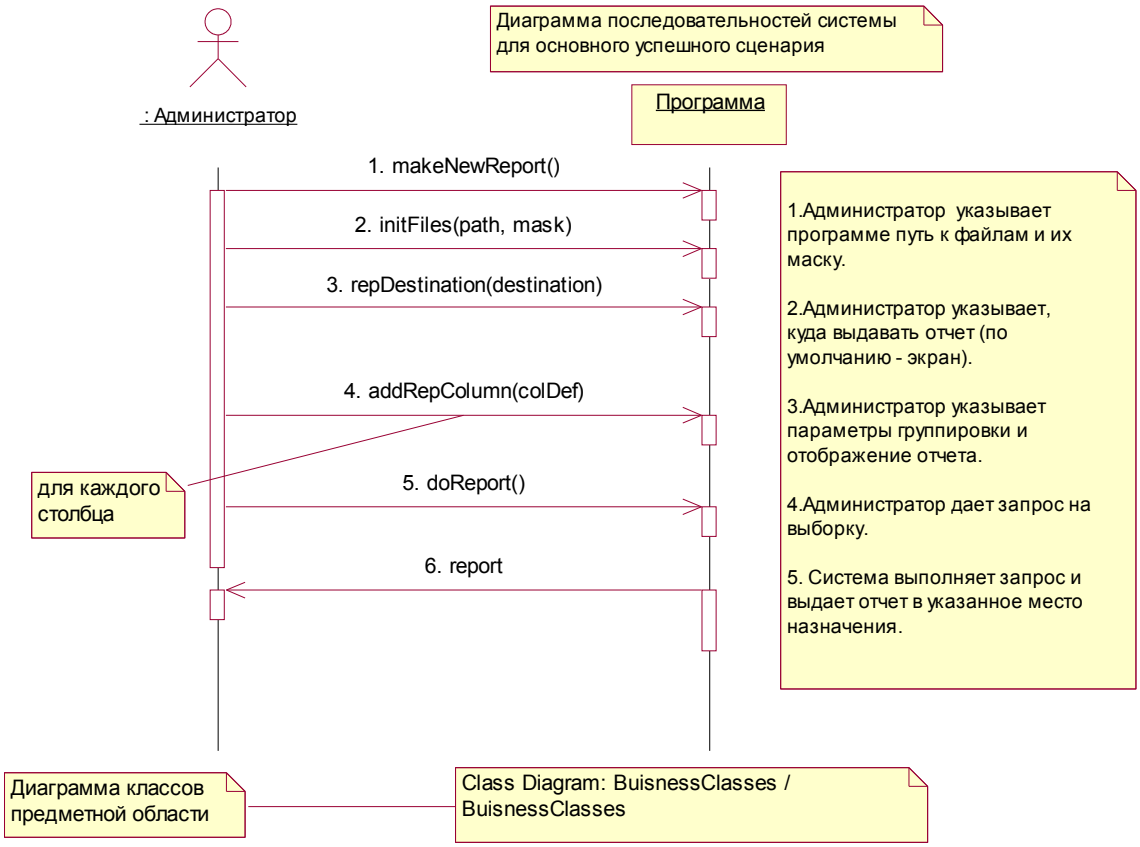
Исполнитель: Администратор

Предусловия: Администратор хочет запросить отчет. Где-то на доступном диске / в сети / на ресурсе имеются log-файлы программы TMeter.

Основной поток:

1. Администратор указывает программе путь к файлам и их маску текстом или выбирая из списка в диалоговом окне.
2. Администратор может указать ограничения на выборку в виде текста, либо – в окне.
3. Администратор может указать, куда выдавать отчет.

4. Администратор может указать прочие параметры.
5. Администратор дает запрос на выборку.
6. Система выполняет запрос и выдает отчет в указанное место.



Для информации : пример log-файла :

--- Time: 2003-04-15 00:00:03
TCP 194.87.225.87 2900 62.113.80.22 client 488 160
--- Time: 2003-04-15 00:00:21
TCP 194.87.225.87 2900 62.113.80.22 client 875 240
--- Time: 2003-04-15 09:29:10
TCP 212.248.18.149 80 62.113.80.22 client 44 445
--- Time: 2003-04-15 09:29:16
UDP 62.113.80.9 53 62.113.80.22 client 1061 308
TCP 195.131.4.161 80 62.113.80.22 client 500 490

--- Time: 2003-04-15 09:33:55
TCP 194.87.225.87 2900 62.113.80.22 client 94 57
ICMP 194.87.225.87 0 62.113.80.22 0 180 180
TCP 194.87.225.87 2900 62.113.80.22 4035 53 0
TCP 194.87.225.87 2900 62.113.80.22 4035 0 40
Realtime-коллектор пакетов полон. filterid=1, size=100

--- Time: 2003-04-14 08:39:51
UDP 62.113.80.21 53 62.113.80.22 client 327 133

TCP 212.248.18.149 80 62.113.80.22 client 30088 1313

Модель предметной области

Построим диаграмму классов предметной области. Для этого рассмотрим основной реализуемый прецедент и выделим из него классы-кандидаты, затем – добавим атрибуты и ассоциации.

1. Выделение классов

Выделим существительные в описании прецедента :

Основной поток:

1. **Администратор** указывает **программе путь к файлам** и их **маску** текстом или выбирая из **списка** в диалоговом окне.
2. Администратор может указать **ограничения на выборку** в виде текста, либо – в окне.
3. Администратор может указать, куда выдавать **отчет**.
4. Администратор может указать прочие **параметры**.
5. Администратор дает **запрос на выборку**.
6. **Система** выполняет **запрос** и выдает **отчет** в указанное **место**.

Выделяем список существительных :

1. ~~Администратор~~
2. Программа
3. ~~Путь к файлам~~
4. ~~Маска файлов~~
5. Список файлов
6. ~~Ограничения на выборку~~
7. Отчет
8. Параметры запроса
9. ~~Запрос~~

Анализируем список. Что можно исключить ? Администратора, так как он является исполнителем, то есть внешним по отношению к системе объектом. Программа и Система являются синонимами, поэтому можно оставить что-то одно, например Программа. Ограничения на выборку и Параметры запроса – суть одно и тоже ? (За исключением того, что Ограничения – на выбор файлов, а Параметры – внутри файлов ? Хотя можно это все обобщить и назвать Параметры запроса.) Путь к файлам и Маска файлов – это, скорее параметры. Запрос сам по себе, видимо, не нужен – его параметры могут храниться в списке, а хранить данные о запросах (когда, сколько и чего) не нужно пока. С другой стороны, запрос может хранить данные о требуемых **группах** (новое существительное !), параметрах в каждой группе и другие моменты, а также – и ссылку на отчет ? (то есть – атрибуты отчета, которые задаются в виде **колонок** и др. информации).

Отчет пока может оставаться, хотя не совсем ясна его роль в дальнейшем. Да и программа в общем как класс не очень-то нужна. С другой стороны, можно выделить отдельный класс Файл Log как элемент списка файлов.

Получаем список классов-кандидатов :

1) Запрос (Query)

2) Список файлов (FileList)
3) Файл “log” (LogFile)
4) Группа (Query) для отчета ?!
5) Параметр запроса (Parameter)
6) Отчет (Report)
7) Колонка (Column)

В результате анализа связей и атрибутов классов получена первая предварительная диаграмма классов предметной области (см. рис.).

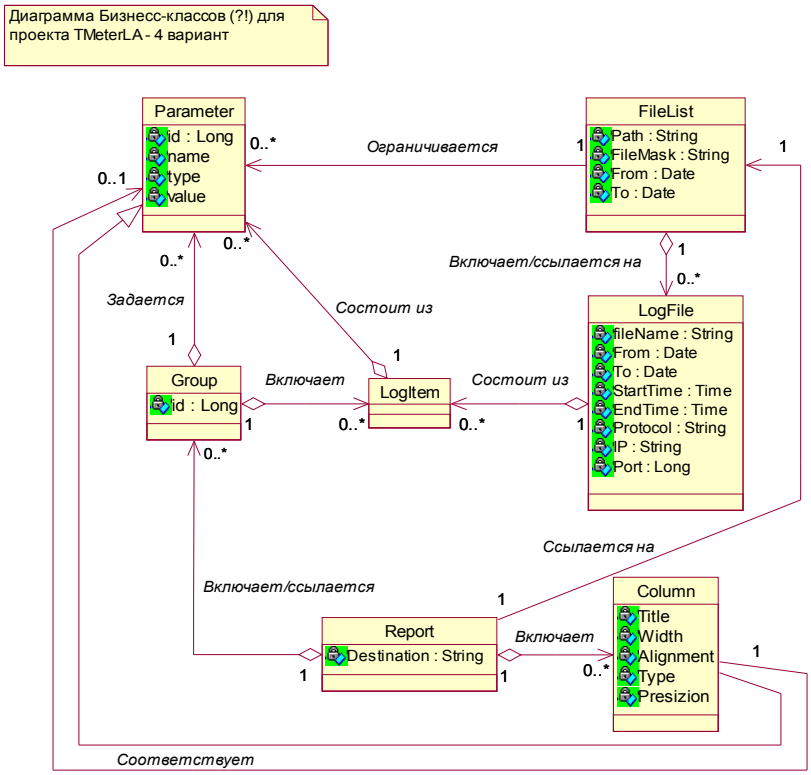
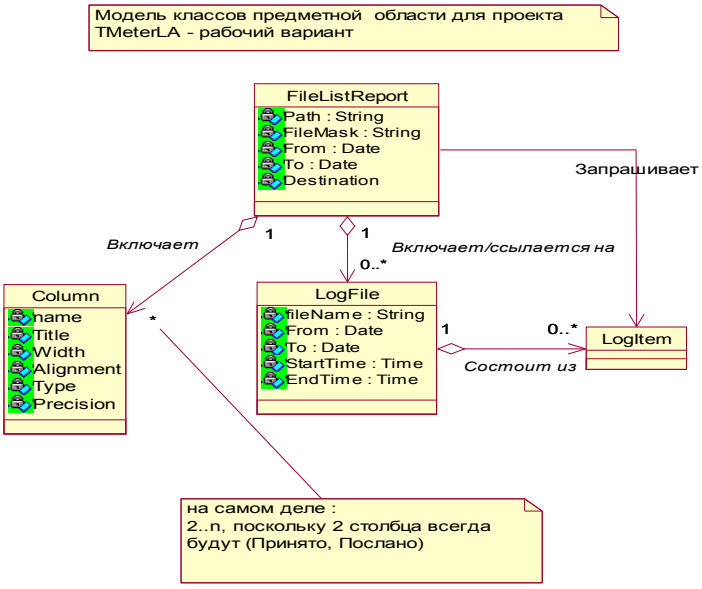


Диаграмма классов предметной области (рабочий вариант)

После совещания с другим разработчиком, принимая во внимание задачи проекта (сделать простую утилиту за минимальное время, поэкспериментировать с UML, C++, объектами и классами, проверить различные архитектурные решения для другого проекта) решено было упростить задачу, уменьшив возможности программы и ее гибкость, для того, чтобы сократить сроки реализации.

Из модели предметной области, таким образом, удалены классы Group, Parameter (его функции перенесены в класс Column), изменен список атрибутов, в результате получаем такую диаграмму классов :



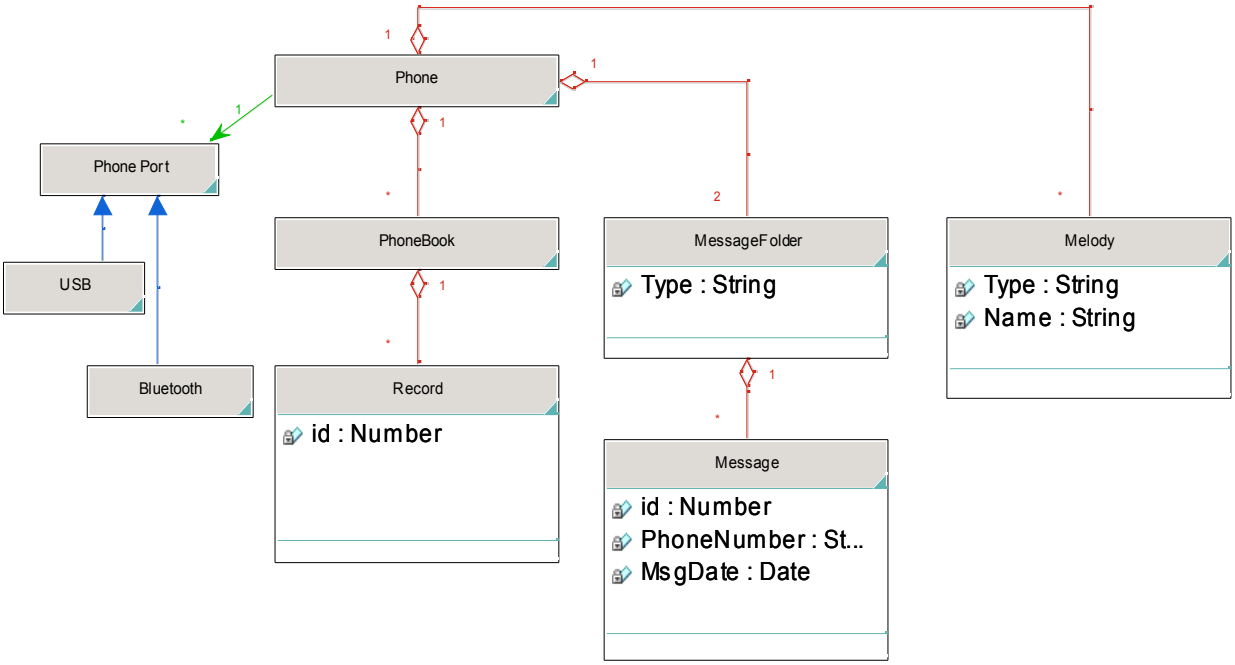
2.2 Примеры различных диаграмм классов и взаимодействия

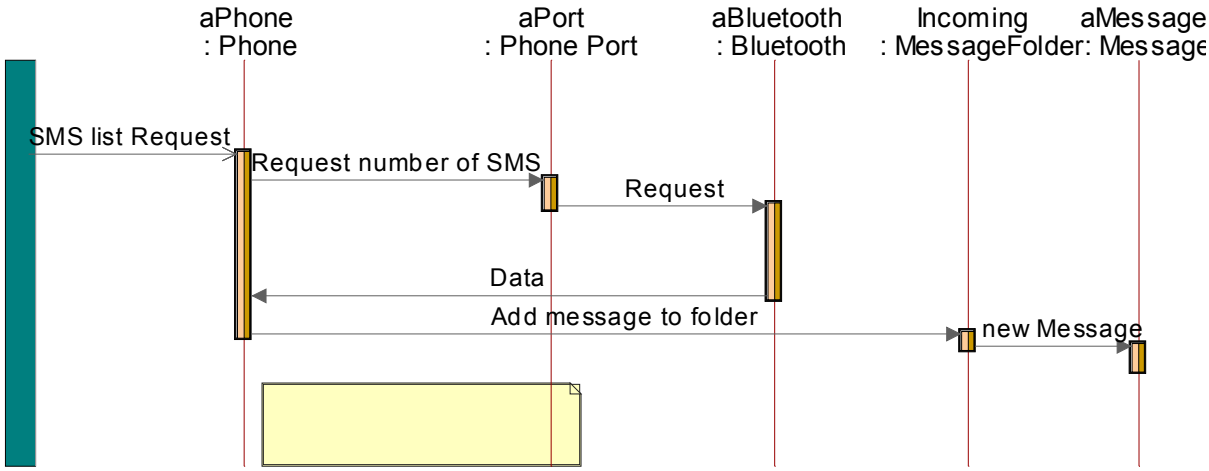
2.1 Пример диаграмм классов и последовательности для программы-броузера данных, хранящихся в мобильном телефоне

Основное назначение программы : обмен данными телефонной книги, архивами сообщений, мелодиями с мобильным телефоном.

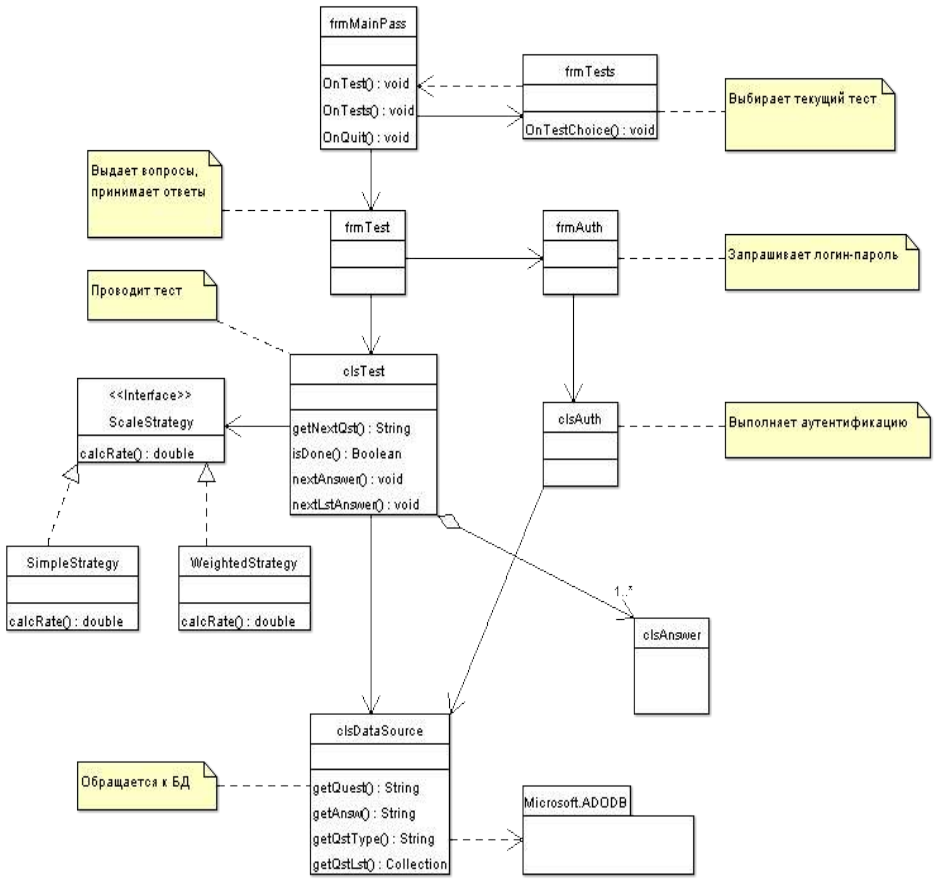
Ниже приводится пример диаграммы классов и диаграммы последовательности для задачи считывания входящих SMS с телефона. Обе диаграммы носят иллюстративный характер. Отсутствие в модели многих сущностей, связанных с телефоном (дисплей, клавиатура, настройки и др.) связано с назначением программы – она в большей степени предназначена для работы с представленными структурами данных.

Диаграммы построены с использованием ObjectiF.


Phone
Phone Port
USB
Bluetooth
PhoneBook
Record
id : Number
MessageFolder
Type : String
Message
id : Number
PhoneNumber : St...
MsgDate : Date
Melody
Type : String
Name : String
1
2
*


aPhone
: Phone
aPort
: Phone Port
aBluetooth
: Bluetooth
Incoming
: MessageFolder
aMessage
: Message
SMS list Request
Request number of SMS
Request
Data
Add message to folder
new Message

2.2 Для сравнения приведем диаграмму программных классов для системы тестирования знаний студентов (см. первую работу).



2.3 Также рассмотрим пример диаграммы взаимодействия (диаграммы последовательностей) для утилиты анализа лог-файлов :

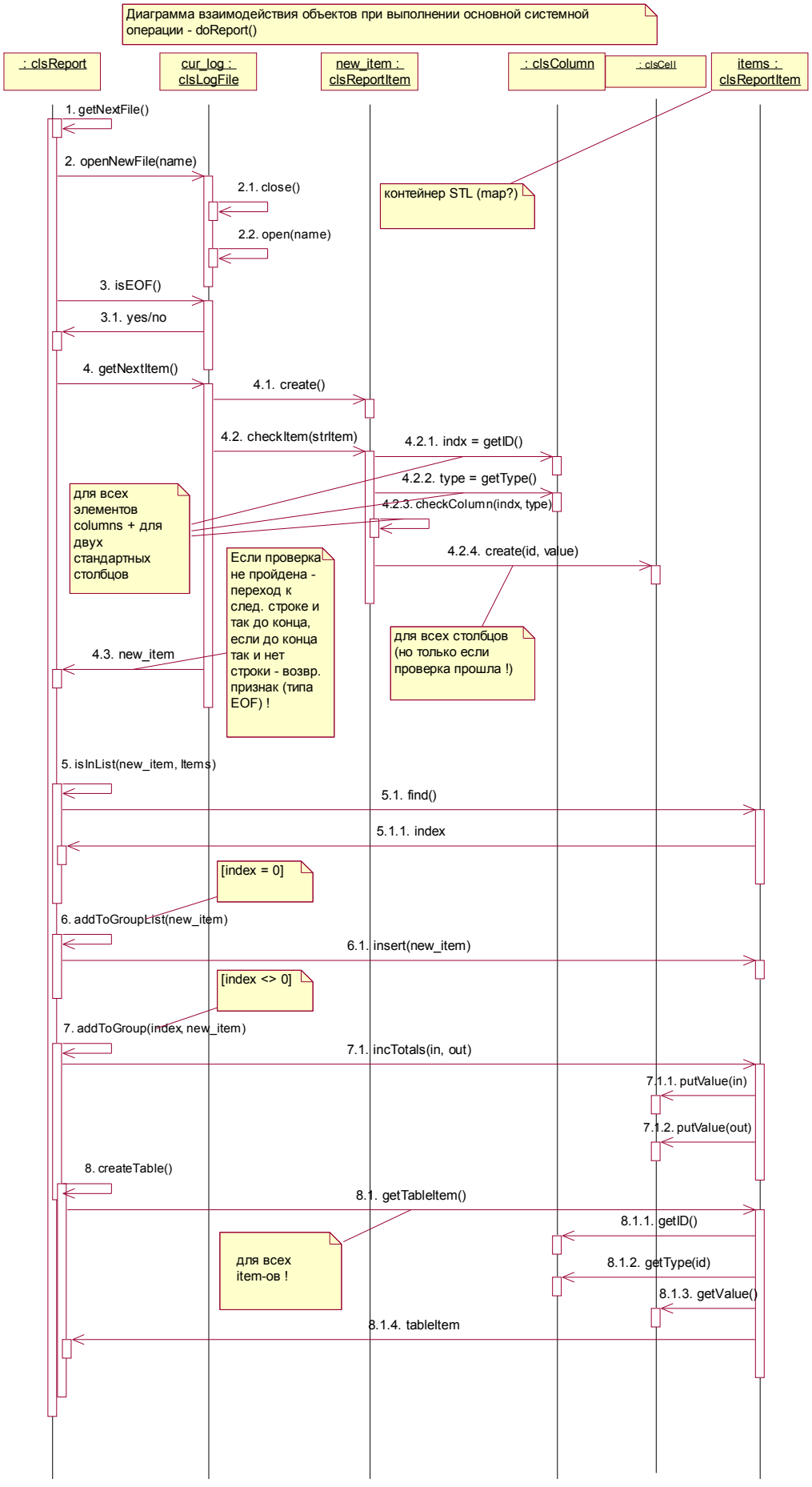


Диаграмма взаимодействия объектов при выполнении основной системной операции - doReport()
: clsReport
cur_log : clsLogFile
new_item : clsReportItem
: clsColumn
: clsCell
items : clsReportItem
1. getNextFile()
2. openNewFile(name)
2.1. close()
2.2. open(name)
контейнер STL (map?)
3. isEOF()
3.1. yes/no
4. getNextItem()
4.1. create()
4.2. checkItem(strItem)
4.2.1. indx = getID()
4.2.2. type = getType()
4.2.3. checkColumn(indx, type)
для всех элементов columns + для двух стандартных столбцов
Если проверка не пройдена - переход к след. строке и так до конца, если до конца так и нет строки - возвр. признак (типа EOF) !
4.2.4. create(id, value)
для всех столбцов (но только если проверка прошла !)
4.3. new_item
5. isInList(new_item, Items)
5.1. find()
5.1.1. index
[index = 0]
6. addToGroupList(new_item)
6.1. insert(new_item)
[index <> 0]
7. addToGroup(index, new_item)
7.1. incTotals(in, out)
7.1.1. putValue(in)
7.1.2. putValue(out)
8. createTable()
8.1. getTableItem()
8.1.1. getID()
для всех item-ов !
8.1.2. getType(id)
8.1.3. getValue()
8.1.4. tableItem

2.3 Пример по основной (сквозной) задаче, которая рассматривается в ходе практикума (расчет платежей по кредиту)

Рассмотрим сжатый формат прецедента (истории) **Расчет платежа по кредиту за указанный период и среднемесячного платежа**.

П1 Расчет платежа по кредиту за указанный период и среднемесячного платежа **Системе** сообщается **годовая процентная ставка**, **сумма кредита** и **количество месяцев** (срок), на который он берется. Система рассчитывает **сумму** начисляемых за все время пользования кредитом **процентов** с учетом ежемесячного погашения кредита равными долями и общую **сумму выплат**. Затем система расчитывает **среднемесячную сумму выплат**.

Формально можно рассмотреть в качестве кандидатов на классы предметной области все существительные в описании, однако практически все они таковыми не являются. Система не является таковой, поскольку это сама разработка, процентная ставка – это атрибут (атомарное значение), как и сумма кредита, срок кредита. Сумма процентов, сумма выплат, ежемесячная сумма выплат – это расчетные величины. Вполне возможно, что они на этапе реализации станут программными классами, а возможно, что и нет. В данном случае прецедент достаточно тривиален и вся модель может быть представлена одним единственным классом, который можно назвать Расчет Кредита (calcCredit).

С другой стороны, если мы рассмотрим прецеденты **Перерасчет кредита** (в т.ч. С учетом штрафов) и **Заключение договора** (которые включены в обозримые итерации проекта), то из них можно выделить ряд новых классов. Рассмотрим описание этих прецедентов.

П2 Заключение договора о кредите – **Клиент** обращается в кр. Отдел и сообщает сумму и срок кредита. **Менеджер** производит **расчет**, при этом задействуются прецеденты П1, П1.1, П1.2. Если Клиента устраивают **условия кредита**, он заполняет **анкету**, Менеджер анализирует ее, принимает решение о выдаче кредита (если да), затем вводит **параметры договора** в систему (возможно, при этом задействуется прецедент П10).

Отсюда можно выделить кандидатов на роль классов предметной области :

1) Клиент
2) Менеджер
3) Расчет кредита
4) ~~Анкета~~
5) Договор о кредите
6) ~~Кредит~~

В данном случае сущность Кредит в явном виде в постановке задачи не фигурирует, скорее его можно отождествить либо с Расчетом (когда задаются его параметры и определяются платежи), либо с Договором (когда Кредит связывается с Клиентом и с рассчитанным Кредитом). Вообще создавать или не создавать такой концептуальный класс это вопрос дискуссионный. Пока не будем включать такой класс в модель. В любом случае при необходимости можно вернуться к анализу данной модели и пересмотреть ее, если будет такая необходимость. Анкету мы пока не включаем, так как в обозримой перспективе анализ анкеты системой не будет разрабатываться.

П3-П5 **Перерасчет кредита с учетом штрафов**. **Клиент** обращается в кр. Организацию и хочет узнать параметры кредита (его **задолженность**) после погашения им сумм по **графику** с **опережением**, либо – с **отставанием** с учетом возможных **штрафов**. Клиент сообщает о **сроках** и **суммах платежей**, система производит **перерасчет** с учетом действующей системы штрафов (**политики штрафов**).

Проанализировав существительные можно отобрать дополнительные кандидаты в

классы предметной области :

~~7) График платежей~~

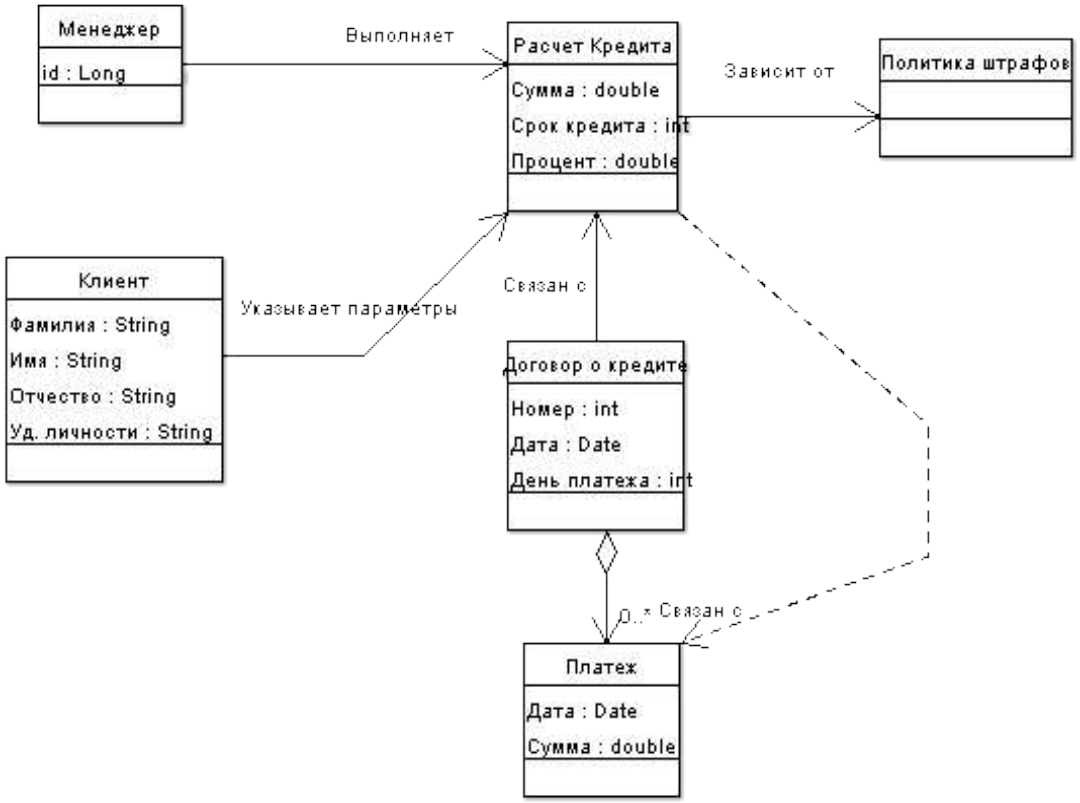
8) Платежи

9) Политика штрафов

Последний концептуальный класс требует дополнительного исследования, как, впрочем, и сам прецедент. Пока будем считать, что это некий алгоритм, зависящий от величины задолженности, ставки штрафов и сроков просрочки. Перерасчет можно не отделять от расчета, считая, что это один и тот же класс с дополнительными параметрами, учитывающими штрафы. Что касается графика платежей, то на начальном этапе его можно считать одним из атрибутов договора.

С точки зрения анализа категорий класов можно рассмотреть дополнительно классы агрегаты и агрегируемые ими классы. В данном случае Платежи относятся к классу-агрегату Договор о кредите. Больше вроде никаких агрегатов не требуется...

С учетом изложенного получаем следующую модель.



Заметим, что данная модель совсем не обязательно должна соответствовать модели реализации, то есть далеко не все концептуальные классы трансформируются затем в программные классы. Составляется данная модель в основном для лучшего изучения предметной области, для выбора имен для классов, для определения важных атрибутов, основных ассоциаций. При использовании подхода, характерного для UP, данная модель также используется для анализа основных системных задач, проектирования диаграмм взаимодействия, реализующих прецеденты, и более серьезно влияет на проект, чем в XP.

3. Порядок выполнения работы

Создание модели предметной области

1. Рассмотреть пример(ы) классификации с использованием прецедентов / постановки задачи.
2. Рассмотреть диаграмму(ы) классов, полученную в результате классификации.
3. Создать диаграмму классов из рассмотренных примеров в ArgoUML, ObjectiF, в других программных средствах для создания диаграмм UML (Poseidon, VP for UML, Jude и пр. Далее для краткости – ПО UML).
4. Рассмотреть другие виды диаграмм из примеров относящиеся к модели предметной области (диаграммы последовательности, активности и т.д.)
5. Создать рассмотренные диаграммы в ПО UML.
6. Провести классификацию и разработать модель классов предметной области на основе собственных прецедентов, реализованных в первой лабораторной работе (по своему варианту).

III Семестровая работа

10. Провести классификацию и разработать диаграмму классов предметной области и другие необходимые диаграммы.
11. Продолжить планирование

4. Варианты заданий

Вариант задания – как в первой лабораторной работе (необходимо провести классификацию на основе составленных прецедентов и выделенных актеров, составить примеры диаграмм последовательности, активности и др.)

1. Интернет – магазин.
2. Информационная система для учета кадров на небольшом предприятии
3. Информационная система для расчета зарплаты на небольшом предприятии
4. Утилита мониторинга сети (для администратора сети)
5. Обучающая программа с функцией контроля знаний (это основная функция !) (фактически кусок этого варианта рассмотрен выше в п. 2.7)
6. Биллинговая система для учета интернет-трафика.
7. Веб - сервер
8. Почтовый сервер
9. Текстовый редактор (потом – текстовый процессор ?!)
10. Программа моделирования цифровых схем
11. Информационная система складского учета (учет прихода и ухода товаров, категории, поиск и пр.)
12. АРМ менеджера компьютерной фирмы (не бухгалтерия !). - Быстрая прикидка конфигураций, выписка счетов, отслеживание счетов их оплат и пр.
13. АРМ начальника отдела техподдержки (учет ремонтов).
14. Графический редактор (векторная графика).
15. Графический редактор (растровая графика).