**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Информационные системы»

**ОТЧЁТ**

по индивидуальному графику обучения

на весенний семестр 2014/2015 учебного года

**Выполнили:**

студенты гр. ПИмд-11, ПИмд-12

Бексаева Е.А.,

Желепов А.С.,

Загайчук И.А.,

Клейн В.В.,

Смеречинский С.О.

**Проверил:**

К.т.н., доцент каф. ИВК

Кандаулов В.М.

Содержание

[Содержание 2](#_Toc438579366)

[1 Разработка математической модели для расчёта тепло-энергетических характеристик воды и водяного пара 3](#_Toc438579367)

[2 Анализ применения онтологии для кластеризации предметной области: “Объекты жилого фонда” для физических и технических характеристик 4](#_Toc438579368)

[3 Разработка мобильного игрового приложения средствами Android-SDK с применением шаблонов проектирования 5](#_Toc438579369)

[4 Разработка системы статистического, морфологического, кластерного и онтологического анализа текстов и данных для ЖКХ 6](#_Toc438579370)

[5 Системный анализ паспортов многоквартирных домов (МКД) для выделения физических и технических характеристик “Объектов жилого фонда” 7](#_Toc438579371)

[6 Список литературы 8](#_Toc438579372)

[Приложение А. Исходный код игрового мобильного приложения «Воздушный бой» 9](#_Toc438579373)

[Приложение Б. Исходный код системы статистического, морфологического, кластерного и онтологического анализа текстов и данных для ЖКХ 10](#_Toc438579374)

1. Разработка математической модели для расчёта тепло-энергетических характеристик воды и водяного пара
   1. Введение

Разрабатываемое решение предназначено для внедрения в автоматизированную систему коммерческого учёта энергоресурсов «АИС: Приборный учет» позволяющей вести комплексный мониторинг и анализ потребления энергоресурсов в разрезе жилого и нежилого фонда, объектов соцсферы. Система обеспечивает получение показаний приборов и их настроечных параметров с широкого ряда приборов учета, как с цифровыми, так и импульсными выходами. Это позволяет производить:

1. перерасчёт потреблённой энергии в рубли вплоть до каждой квартиры в соответствии законодательством;
2. объединение поквартирных и общедомовых показаний и сопоставление их с отдачей ресурсов РСО;
3. оценку качества поставляемых энергоресурсов и расчет снижения стоимости;
4. контроль за нормами ГОСТ и СанПиН;
5. графическую индикацию важных и критических событий;
6. анализ потерь в сетях МО.

Данное решение позволит точнее определить объёмы потребляемых ресурсов. Использование такого подхода будет выгодно как жителям многоквартирных домов, так и ресурсоснабжающим организациям. Жители домов сократят расходы на коммунальные услуги, а ресурсоснабжающие организации получат точные показатели энергопотребления, а соответственно смогут составить план закупок энергии с минимальными переплатами. Следовательно, необходимо исследовать возможность применения различных математических моделей для расчёта теплоэнергетических характеристик.

Цель работы: разработка математической модели для расчёта тепло-энергетических характеристик воды и водяного пара.

* 1. Теплоэнергетические характеристики воды и водяного пара

Тепловые явления можно описывать с помощью величин (макроскопических параметров), измеряемых такими приборами, как манометр и термометр. Эти приборы не реагируют на воздействие отдельных молекул. Теория тепловых процессов, в которой не учитывается молекулярное строение тел, называется термодинамикой. В термодинамике рассматриваются процессы с точки зрения превращения теплоты в другие виды энергии.

Первой научной теорией тепловых процессов была не молекулярно-кинетическая теория, а термодинамика.

Термодинамика возникла при изучении оптимальных условий использования теплоты для совершения работы. Это произошло в середине XIX в., задолго до того, как молекулярно-кинетическая теория получила всеобщее признание. Тогда же было доказано, что наряду с механической энергией макроскопические тела обладают ещё и энергией, заключённой внутри самих тел.

Сейчас в науке и технике при изучении тепловых явлений используется как термодинамика, так и молекулярно-кинетическая теория. В теоретической физике молекулярно-кинетическую теорию называют статистической механикой.

Термодинамика и статистическая механика изучают различными методами одни и те же явления и взаимно дополняют друг друга. Термодинамической системой называют совокупность взаимодействующих тел, обменивающихся энергией и веществом. Главное содержание термодинамики состоит в двух основных её законах, касающихся преобразования энергии. Эти законы установлены опытным путём. Они справедливы для всех веществ независимо от их внутреннего строения.

Внутренняя энергия тела (системы) — это сумма кинетической энергии хаотичного теплового движения молекул и потенциальной энергии их взаимодействия.

Механическая энергия тела (системы) как целого не входит во внутреннюю энергию. Например, внутренняя энергия газов в двух одинаковых сосудах при равных условиях одинакова независимо от движения сосудов и их расположения относительно друг друга.

Вычислить внутреннюю энергию тела (или её изменение), учитывая движение отдельных молекул и их положения относительно друг друга, практически невозможно из-за огромного числа молекул в макроскопических телах. Поэтому необходимо уметь определять значение внутренней энергии (или её изменение) в зависимости от макроскопических параметров, которые можно непосредственно измерить.

Вычислим внутреннюю энергию идеального одноатомного газа.

Согласно модели молекулы идеального газа не взаимодействуют друг с другом, следовательно, потенциальная энергия их взаимодействия равна нулю. Вся внутренняя энергия идеального газа определяется кинетической энергией беспорядочного движения его молекул.

Для вычисления внутренней энергии идеального одноатомного газа массой т нужно умножить среднюю кинетическую энергию одного атома на число атомов.

Внутренняя энергия идеального одноатомного газа прямо пропорциональна его абсолютной температуре. Она не зависит от объёма и других макроскопических параметров системы.

Изменение внутренней энергии идеального газа определяется температурами начального и конечного состояний газа и не зависит от процесса.

Если идеальный газ состоит из более сложных молекул, чем одноатомный, то его внутренняя энергия также пропорциональна абсолютной температуре, но коэффициент пропорциональности другой. Объясняется это тем, что сложные молекулы не только движутся поступательно, но ещё и вращаются и колеблются относительно своих положений равновесия. Внутренняя энергия таких газов равна сумме энергий поступательного, вращательного и колебательного движений молекул. Следовательно, внутренняя энергия многоатомного газа больше энергии одноатомного газа при той же температуре.

Зависимость внутренней энергии от макроскопических параметров. Мы установили, что внутренняя энергия идеального газа зависит от одного параметра — температуры.

У реальных газов, жидкостей и твёрдых тел средняя потенциальная энергия взаимодействия молекул не равна нулю. Правда, для газов она много меньше средней кинетической энергии молекул, но для твёрдых и жидких тел сравнима с ней.

Средняя потенциальная энергия взаимодействия молекул газа зависит от объёма вещества, так как при изменении объёма меняется среднее расстояние между молекулами. Следовательно, внутренняя энергия реального газа в термодинамике в общем случае зависит наряду с температурой Т и от объёма V.

Значения макроскопических параметров (температуры, объёма и др.) однозначно определяют состояние тел. Поэтому они определяют и внутреннюю энергию макроскопических тел.

* 1. Сбор данных для построения модели

Для построения модели, а соответственно и написания программы рассчитывающей отклонения в объёме воды и затраченной энергии, необходимо:

а) Получить набор исходных справочных данных удельного объёма и энтальпии;

б) Преобразовать полученные данные в формат позволяющий получить граничные значения на основании входных параметров;

в) Написать алгоритмы расчета показателей, для корректной работы модели;

г) Получить набор реальных показаний потребления воды на общедомовом счётчике горячей воды и рассчитать значение реально потреблённого объёма воды и энергии.

* + 1. ГСССД 187-99. Вода. Удельный объем и энтальпия

Таблица стандартных справочных данных разработана Московским энергетическим институтом (техническим университетом) под авторством д-р техн. наук Александровым А.А. и д-р техн. наук Григорьевым Б.А.. Рекомендованы к утверждению Российским национальный комитетом Международной ассоциации по свойствам воды и водяного пара.

Настоящие таблицы заменяют таблицы стандартных справочных данных ГСССД 98-86 "Вода. Удельный объем и энтальпия при температурах 0…800 °С и давлениях 0,001...1000 МПа". Замена таблиц обусловлена переходом на новую Международную температурную шкалу 1990 г. (МТШ-90) и тем, что в 1995 г. Международная ассоциация по свойствам воды и водяного пара (МАСВП) приняла новое уравнение состояния для воды и водяного пара, обобщающее новейшие экспериментальные данные и позволяющее производить расчет термодинамических свойств в более широкой области температур.

Это уравнение выражает зависимость удельной приведенной энергии Гельмгольца:

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

от приведенных температуры:

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

и плотности:

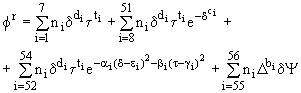
ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

и состоит из двух частей - относящихся к идеально газовому состоянию и описывающих реальную составляющую.

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

где

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа



ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

Значения коэффициентов уравнений приведены соответственно в табл.П1 и П2 (ГСССД 187-99. Вода. Удельный объем и энтальпия), а параметры приведения, используемые при вычислении термодинамических свойств, имеют следующие значения:

Приводимые в табл.1, 2 значения удельного объема

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

и энтальпии  получены для данного давления  и температуры из уравнения с помощью известных термодинамических соотношений

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

ГСССД 187-99 Вода. Удельный объем и энтальпия при температурах 0...1000 °С и давлениях 0,001...1000 МПа

Значения максимальных погрешностей табличных данных для удельного объема и энтальпии воды и водяного пара приведены в табл.3, 4, составленных на основе допусков, рекомендованных МАСВП в документе. Эти величины имеют характер оценки интервала указанного значения и не имеют статистического смысла.

Уравнение применимо для расчета термодинамических свойств воды и водяного пара в однофазных состояниях во всей области параметров, представленной в таблицах, за исключением малой области над критической точкой в интервале температур +5К при плотностях (1+0,05).

* + 1. Математическая модель

Математическая модель – [математическое](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) [представление](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F)) [реальности](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C), один из вариантов [модели](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_(%D0%BD%D0%B0%D1%83%D0%BA%D0%B0)), как системы, исследование которой позволяет получать [информацию](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) о некоторой другой системе. Процесс построения и изучения математических моделей называется математическим моделированием.

Все [естественные](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8) и [общественные](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8) науки, использующие математический аппарат, по сути, занимаются математическим моделированием: заменяют [объект исследования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) его математической [моделью](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) и затем изучают последнюю. Связь математической модели с [реальностью](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) осуществляется с помощью цепочки гипотез [идеализаций](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) и упрощений. С помощью математических методов описывается, как правило, идеальный объект, построенный на этапе содержательного моделирования.

* + 1. Получение набора реальных данных для проведения расчёта

Для проведения расчета теплоэнергетических характеристик воды, в качестве реального примера был взят многоквартирный жилой дом в городе Димитровграде.

Чтобы провести сравнительную характеристику, достаточно иметь объём данных за один отчётный период (месяц) с общедомового счётчика измерения тепловой энергии.

* 1. Реализация математической модели
     1. Преобразование исходных таблиц

Для работы со справочными значениями (таблицы 1.1 и 1.3), необходимо адаптировать данные и преобразовать их в таблицу MSSQL.

Таблица 1.1. Стандартные справочные значения удельного объёма V 10-3 м3/кг, воды и водяного пара

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P \ T** | **0** | **10** | **20** | **30** | **40** | **50** | **60** | **70** | **80** | **90** | **100** | **110** | **120** | **130** | **140** |
| **0,1** | 1,0002 | 1,0003 | 1,0018 | 1,0044 | 1,0078 | 1,0121 | 1,0171 | 1,0227 | 1,029 | 1,0359 | 1,696 | 1,745 | 1,793 | 1,841 | 1,889 |
| **0,5** | 1 | 1,0001 | 1,0016 | 1,0042 | 1,0077 | 1,0119 | 1,0169 | 1,0225 | 1,0288 | 1,0357 | 1,0433 | 1,0514 | 1,0602 | 1,0696 | 1,0797 |
| **1** | 0,9997 | 0,9999 | 1,0014 | 1,004 | 1,0075 | 1,0117 | 1,0167 | 1,0224 | 1,0286 | 1,0355 | 1,043 | 1,0512 | 1,0599 | 1,0693 | 1,0794 |
| **2,5** | 0,9989 | 0,9992 | 1,0007 | 1,0033 | 1,0068 | 1,011 | 1,016 | 1,0216 | 1,0279 | 1,0348 | 1,0422 | 1,0503 | 1,059 | 1,0684 | 1,0784 |
| **5** | 0,9977 | 0,998 | 0,9996 | 1,0022 | 1,0057 | 1,0099 | 1,0149 | 1,0207 | 1,0267 | 1,0338 | 1,0412 | 1,0493 | 1,0579 | 1,0672 | 1,0772 |

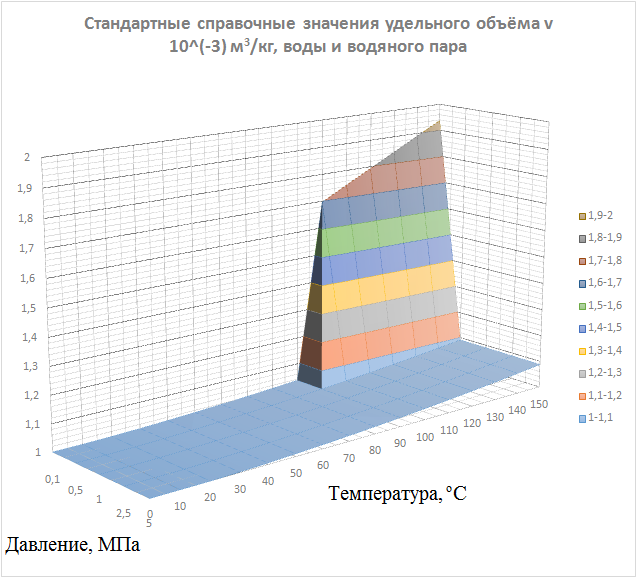


Рисунок 1.2. Стандартные справочные значения удельного объёма V 10-3 м3/кг, воды и водяного пара

Таблица 1.3. Стандартные справочные значения энтальпии h кДж/кг, воды и водяного пара

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P \ T** | **0** | **10** | **20** | **30** | **40** | **50** | **60** | **70** | **80** | **90** | **100** | **110** | **120** | **130** | **140** |
| **0,1** | 0,1 | 42,1 | 84 | 125,8 | 167,6 | 209,4 | 251,2 | 293,1 | 335 | 377 | 2675 | 2696,1 | 2716,6 | 2736,7 | 2756,7 |
| **0,5** | 0,47 | 42,5 | 84,4 | 126,2 | 168 | 209,8 | 251,6 | 293,4 | 335,3 | 377,3 | 419,4 | 461,6 | 504 | 546,5 | 589,3 |
| **1** | 0,98 | 43 | 84,9 | 126,7 | 168,4 | 210,2 | 252 | 293,8 | 335,7 | 377,7 | 419,8 | 462 | 504,3 | 546,9 | 589,6 |
| **2,5** | 2,5 | 44,5 | 86,3 | 128 | 169,7 | 211,5 | 253,2 | 295 | 336,9 | 378,8 | 420,9 | 463,1 | 505,4 | 547,9 | 590,6 |
| **5** | 5 | 46,9 | 88,6 | 130,3 | 172 | 213,6 | 255,3 | 297,1 | 338,9 | 380,8 | 422,8 | 464,9 | 507,2 | 549,6 | 592,2 |

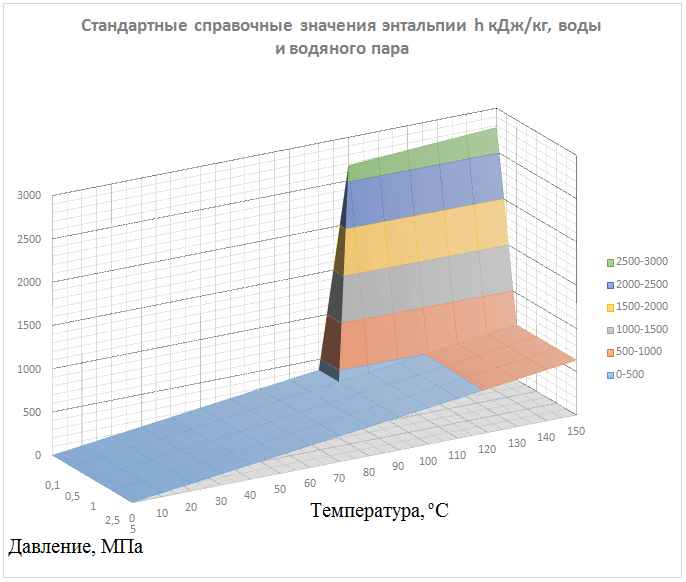


Рисунок 1.4. Стандартные справочные значения энтальпии h кДж/кг, воды и водяного пара

Для удобства обращения к данным, было создано 2 таблицы [Reference\_Water\_SpecificVolumeCalculation] – хранит стандартные справочные значения удельного объёма, [Reference\_Water \_SpecificEnthalpy] – хранит Стандартные справочные значения энтальпии.

Структура таблиц представляет собой формат хранения [ключ, значение], где используется составной ключ из характеристик температура-давление, а значением является удельный объём/энтальпия соответственно. Пример хранения данных представлен на рисунке 1.5.

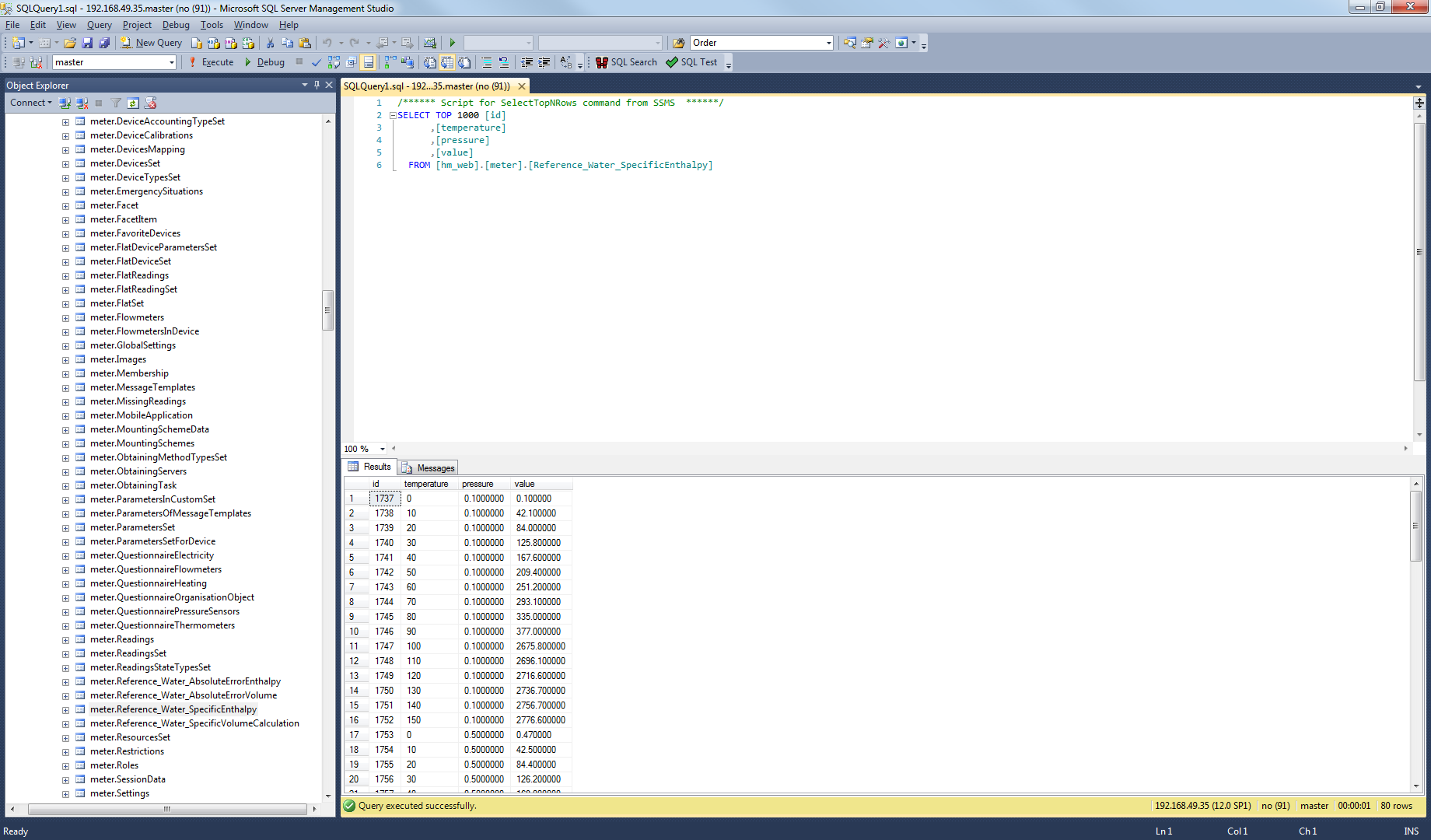


Рисунок 1.5. Пример хранения данных в таблице MSSQL

Подобный способ хранения данных позволяет быстро получать граничные стандартные справочные значения на языке T-SQL.

* + 1. Описание алгоритмов

Для того чтобы провести расчёты теплоэнергетических характеристик воды, нужно:

1. получить граничные значения температуры и давления
2. определить попадания в границы интервалов
3. построить изобару/изотерму соответственно
4. произвести расчёт отклонения

Если принять зависимость на ограниченном интервале линейной, то для нахождения теплоэнергетических характеристик потребуется только температура, давление и объём/масса (исходные). Листинги программ на языке T-SQL представлены ниже.

Листинг 1.6. Расчет объёма воды

ALTER PROCEDURE [meter].[Reference\_Water\_VolumeCalculation]

--Температура исходная

@temperature DECIMAL(18, 6),

--Давление исходное

@pressure DECIMAL(18, 6)

AS

BEGIN

SET NOCOUNT ON;

--Приведение давления к МПа

SET @pressure = @pressure \* 0.0980665

--Границы таблицы по давлению и температуры

DECLARE @MinPresure DECIMAL(18, 6)

SET @MinPresure = (SELECT MIN(t.pressure) FROM meter.Reference\_Water\_SpecificVolumeCalculation AS t)

DECLARE @MaxPresure DECIMAL(18, 6)

SET @MaxPresure = (SELECT MAX(t.pressure) FROM meter.Reference\_Water\_SpecificVolumeCalculation AS t)

DECLARE @MinTemperature DECIMAL(18, 6)

SET @MinTemperature = (SELECT MIN(t.temperature) FROM meter.Reference\_Water\_SpecificVolumeCalculation AS t)

DECLARE @MaxTemperature DECIMAL(18, 6)

SET @MaxTemperature = (SELECT MAX(t.temperature) FROM meter.Reference\_Water\_SpecificVolumeCalculation AS t)

--Верхнее ближайшее значение температуры

DECLARE @TempTemperature DECIMAL(18, 6)

--Верхнее ближайшее значение давления

DECLARE @TempPressure DECIMAL(18, 6)

--Нижнее ближайшее значение температуры

DECLARE @TempPrevTemperature DECIMAL(18, 6)

--Нижнее ближайшее значение давления

DECLARE @TempPrevPressure DECIMAL(18, 6)

--Коэффициенты К и В для уравнения прямой y=k\*x+b

DECLARE @K DECIMAL(18, 6)

DECLARE @B DECIMAL(18, 6)

--Прямые для определения значения в точке

DECLARE @Y\_1 DECIMAL(18, 6)

DECLARE @Y\_2 DECIMAL(18, 6)

--Проверка попадание в границы таблицы

IF ((@pressure >= @MinPresure) AND (@pressure <= @MaxPresure) AND (@temperature >= @MinTemperature) AND (@temperature <= @MaxTemperature))

BEGIN

--Определение Верхнего ближайшего значения температуры и давления

SET @TempTemperature = (SELECT TOP(1) temperature FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE @temperature <= temperature GROUP BY temperature ORDER BY temperature)

SET @TempPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE @pressure <= pressure GROUP BY pressure ORDER BY pressure)

--Проверка на несовпадение значения температуры с табличным

IF (@temperature <> @TempTemperature)

BEGIN

--Определение Нижнего ближайшего значения температуры

SET @TempPrevTemperature = (SELECT TOP(1) temperature FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature < @TempTemperature GROUP BY temperature ORDER BY temperature DESC)

--Проверка на несовпадение значения давления с табличным

IF (@pressure <> @TempPressure)

BEGIN

--Определение Нижнего ближайшего значения давления

SET @TempPrevPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE pressure < @TempPressure GROUP BY pressure ORDER BY pressure DESC)

--Вычисление уравнения 1ой прямой по t, при нижней границе давления

--K = (V2-V1)/(T2-T1), где V плотности согласно таблице

SET @K = ((SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPrevPressure) -

(SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempPrevTemperature AND pressure = @TempPrevPressure))/

(@TempTemperature - @TempPrevTemperature)

--В = V1-T1\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempPrevTemperature AND pressure = @TempPrevPressure) - @TempPrevTemperature \* @K

--Y1 = K\*T+B, где T - заданное значение температуры

SET @Y\_1 = @K \* @temperature + @B

--Вычисление уравнения 2ой прямой по t, при верхней границе давления

--В = V2-T2\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPressure) - @TempTemperature \* @K

--Y2 = K\*T+B, где T - заданное значение температуры

SET @Y\_2 = @K \* @temperature + @B

END

ELSE

BEGIN

--Вычисление уравнения прямой по t, при заданном давлении

--K = (V2-V1)/(T2-T1), где V плотности согласно таблице

SET @K = ((SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPressure) -

(SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempPrevTemperature AND pressure = @TempPressure))/

(@TempTemperature - @TempPrevTemperature)

--В = V1-T1\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempPrevTemperature AND pressure = @TempPressure) - @TempPrevTemperature \* @K

--Y1 = K\*T+B, где T - заданное значение температуры

SET @Y\_1 = @K \* @temperature + @B

END

END

ELSE

--При совпадении значения температуры с табличным

BEGIN

SET @TempPrevPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE pressure < @TempPressure GROUP BY pressure ORDER BY pressure DESC)

SET @Y\_1 = (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPrevPressure)

SET @Y\_2 = (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPressure)

END

--Проверка на несовпадение значения давления с табличным

IF (@pressure <> @TempPressure)

BEGIN

--K=(Y2-Y1)/(P2-P1)

SET @K = (@Y\_2 - @Y\_1)/(@TempPressure - @TempPrevPressure)

--B=Y1-T2\*K

SET @B = @Y\_1 - @TempPressure \* @K

--dV = K\*P+B, где T - заданное значение давления

SET @Y\_1 = (@K \* @pressure + @B)

END

ELSE

BEGIN

--Совпадение с табличными значениями => значение из таблицы

IF (@temperature = @TempTemperature)

SELECT (SELECT value FROM meter.Reference\_Water\_SpecificVolumeCalculation WHERE temperature = @TempTemperature AND pressure = @TempPressure) AS V

ELSE

--Иначе значение по Изобаре

SELECT @Y\_1 AS V

END

END

ELSE

BEGIN

--Выходы за границы таблицы

SET @Y\_1 = NULL

END

SELECT @Y\_1 AS V

END

Листинг 1.7. Расчет энтальпии

ALTER PROCEDURE [meter].[Reference\_Water\_Enthalpy]

--Температура исходная

@temperature DECIMAL(18, 6),

--Давление исходное

@pressure DECIMAL(18, 6)

AS

BEGIN

SET NOCOUNT ON;

--Приведение давления к МПа

SET @pressure = @pressure \* 0.0980665

--Границы таблицы по давлению и температуры

DECLARE @MinPresure DECIMAL(18, 6)

SET @MinPresure = (SELECT MIN(t.pressure) FROM meter.Reference\_Water\_SpecificEnthalpy AS t)

DECLARE @MaxPresure DECIMAL(18, 6)

SET @MaxPresure = (SELECT MAX(t.pressure) FROM meter.Reference\_Water\_SpecificEnthalpy AS t)

DECLARE @MinTemperature DECIMAL(18, 6)

SET @MinTemperature = (SELECT MIN(t.temperature) FROM meter.Reference\_Water\_SpecificEnthalpy AS t)

DECLARE @MaxTemperature DECIMAL(18, 6)

SET @MaxTemperature = (SELECT MAX(t.temperature) FROM meter.Reference\_Water\_SpecificEnthalpy AS t)

--Верхнее ближайшее значение температуры

DECLARE @TempTemperature DECIMAL(18, 6)

--Верхнее ближайшее значение давления

DECLARE @TempPressure DECIMAL(18, 6)

--Нижнее ближайшее значение температуры

DECLARE @TempPrevTemperature DECIMAL(18, 6)

--Нижнее ближайшее значение давления

DECLARE @TempPrevPressure DECIMAL(18, 6)

--Коэффициенты К и В для уравнения прямой y=k\*x+b

DECLARE @K DECIMAL(18, 6)

DECLARE @B DECIMAL(18, 6)

--Прямые для определения значения в точке

DECLARE @Y\_1 DECIMAL(18, 6)

DECLARE @Y\_2 DECIMAL(18, 6)

--Проверка попадание в границы таблицы

IF ((@pressure >= @MinPresure) AND (@pressure <= @MaxPresure) AND (@temperature >= @MinTemperature) AND (@temperature <= @MaxTemperature))

BEGIN

--Определение Верхнего ближайшего значения температуры и давления

SET @TempTemperature = (SELECT TOP(1) temperature FROM meter.Reference\_Water\_SpecificEnthalpy WHERE @temperature <= temperature GROUP BY temperature ORDER BY temperature)

SET @TempPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificEnthalpy WHERE @pressure <= pressure GROUP BY pressure ORDER BY pressure)

--Проверка на несовпадение значения температуры с табличным

IF (@temperature <> @TempTemperature)

BEGIN

--Определение Нижнего ближайшего значения температуры

SET @TempPrevTemperature = (SELECT TOP(1) temperature FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature < @TempTemperature GROUP BY temperature ORDER BY temperature DESC)

--Проверка на несовпадение значения давления с табличным

IF (@pressure <> @TempPressure)

BEGIN

--Определение Нижнего ближайшего значения давления

SET @TempPrevPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificEnthalpy WHERE pressure < @TempPressure GROUP BY pressure ORDER BY pressure DESC)

--Вычисление уравнения 1ой прямой по t, при нижней границе давления

--K = (V2-V1)/(T2-T1), где V плотности согласно таблице

SET @K = ((SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPrevPressure) -

(SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempPrevTemperature AND pressure = @TempPrevPressure))/

(@TempTemperature - @TempPrevTemperature)

--В = V1-T1\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempPrevTemperature AND pressure = @TempPrevPressure) - @TempPrevTemperature \* @K

--Y1 = K\*T+B, где T - заданное значение температуры

SET @Y\_1 = @K \* @temperature + @B

--Вычисление уравнения 2ой прямой по t, при верхней границе давления

--В = V2-T2\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPressure) - @TempTemperature \* @K

--Y2 = K\*T+B, где T - заданное значение температуры

SET @Y\_2 = @K \* @temperature + @B

END

ELSE

BEGIN

--Вычисление уравнения прямой по t, при заданном давлении

--K = (V2-V1)/(T2-T1), где V плотности согласно таблице

SET @K = ((SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPressure) -

(SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempPrevTemperature AND pressure = @TempPressure))/

(@TempTemperature - @TempPrevTemperature)

--В = V1-T1\*K

SET @B = (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempPrevTemperature AND pressure = @TempPressure) - @TempPrevTemperature \* @K

--Y1 = K\*T+B, где T - заданное значение температуры

SET @Y\_1 = @K \* @temperature + @B

END

END

ELSE

--При совпадении значения температуры с табличным

BEGIN

SET @TempPrevPressure = (SELECT TOP(1) pressure FROM meter.Reference\_Water\_SpecificEnthalpy WHERE pressure < @TempPressure GROUP BY pressure ORDER BY pressure DESC)

SET @Y\_1 = (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPrevPressure)

SET @Y\_2 = (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPressure)

END

--Проверка на несовпадение значения давления с табличным

IF (@pressure <> @TempPressure)

BEGIN

--K=(Y2-Y1)/(P2-P1)

SET @K = (@Y\_2 - @Y\_1)/(@TempPressure - @TempPrevPressure)

--B=Y1-T2\*K

SET @B = @Y\_1 - @TempPressure \* @K

--H = K\*P+B, где T - заданное значение энтальпии

SET @Y\_1 = (@K \* @pressure + @B)

END

ELSE

BEGIN

--Совпадение с табличными значениями => значение из таблицы

IF (@temperature = @TempTemperature)

SELECT (SELECT value FROM meter.Reference\_Water\_SpecificEnthalpy WHERE temperature = @TempTemperature AND pressure = @TempPressure) AS V

ELSE

--Иначе значение по Изобаре

SELECT @Y\_1 AS H

END

END

ELSE

BEGIN

--Выходы за границы таблицы

SELECT NULL

END

SELECT @Y\_1 AS H

END

1. Анализ применения онтологии для кластеризации предметной области: “Объекты жилого фонда” для физических и технических характеристик
2. Разработка мобильного игрового приложения средствами Android-SDK с применением шаблонов проектирования

**Аннотация:** В рамках статьи рассматривается цикл разработки игрового приложения для мобильной операционной системы Android. Приводится описание технологий и подходов, применяемых при разработке программного обеспечения.

**Ключевые слова:** мобильная платформа, игровое приложение, спрайт, обработка событий.

* 1. Введение

На сегодняшний день платформа Android является наиболее популярной для разработки интерактивных приложений, среди которых немалую часть занимают мобильные игры. Для их создания обычно применяются технологии игровых движков, готовых компонентов, обеспечивающих отображение и взаимодействие объектов на сцене.

Среди этих технологий наибольшей известностью пользуется платформа Unity 3D. Ее популярность объясняется возможностью ведения кроссплатформенной разработки, благодаря которой исходный код приложения может быть скомпилирован и запущен на мобильных устройствах, поддерживающих различные операционные системы (Android, IOS, Windows, Blackberry).

Однако в случае больших проектов (например, подразумевающих серверную часть) кроссплатформенность может сильно влиять на производительность приложения из-за того, что компоненты платформы для разработки игр созданы с учетом всех особенностей поддерживаемых операционных систем. В таком случае приложение обычно параллельно разрабатывается для нескольких платформ, чтобы все части игровой логики и графики были сделаны с помощью «нативной» технологии разработки программного обеспечения.

Объектно-ориентированный язык программирования Java в совокупности с Android SDK является «нативным» комплектом для разработки приложений под операционную систему Android. В качестве интеллектуальной среды разработки программного обеспечения чаще всего применяется Android Studio, позволяющая легко настроить SDK нужной версии и проводить отладку на эмуляторе мобильного устройства.

* 1. Особенности создания архитектуры игрового приложения

«Нативные» технологии не поддерживают ряд необходимых функций для создания игрового приложения, но, как правило, предоставляют достаточный набор инструментов для их пользовательской реализации.

Перед началом разработки с помощью технологии, не специализированной под разработку игр, всегда необходимо ответить на несколько вопросов:

1. каким образом будет организован процесс смены кадров (от англ. frame) приложения?
2. каким образом организовать архитектуру приложения в целом?
3. каким образом будет организовано отображение игровых элементов на экране?
4. каким образом будет построена система взаимодействия с пользователем?

Язык программирования Java позволяет определять и запускать несколько потоков в рамках одного приложения. Таким образом, процесс смены состояний элементов игрового мира, отображение и изменение позиций на экране, может быть вынесен в отдельный поток. Для получения эффекта равномерного изменения состояний объектов на экране приложения необходимо «регулировать» частоту выполнение этого потока. В листинге 3.1 приведен вариант реализации шаблона проектирования «Игровой цикл» с необходимыми комментариями:

Листинг 3.1. Реализация шаблона проектирования «Игровой цикл»

public class GameThread extends Thread {

   private SurfaceHolder surfaceHolder;

   private GamePanel gamePanel;

   private boolean executing;

   public static Canvas canvas;

   public GameThread(SurfaceHolder sh, GamePanel gp) {

    ...

   }

   @Override

   public void run() {

       long startTime, updateTime, waitTime,

       totalTime = 0, target = 1000/FPS; //"шаг" игрового цикла

       int frameCount = 0;

       //цикл смены кадров игрового приложения

       while(executing) {

           startTime = System.nanoTime(); //время начала итерации игрового цикла

           canvas = null;

           try {

               canvas = this.surfaceHolder.lockCanvas();

               synchronized (surfaceHolder) {

                   this.gamePanel.update(); //расчет состояний объектов

                   this.gamePanel.draw(canvas); //отображение объектов

               }

           }

           catch (Exception e) { }

           finally {

               if(canvas != null) {

                   try {

                       surfaceHolder.unlockCanvasAndPost(canvas);

                   }

                   catch (Exception e) { }

               }

           }

      updateTime= (System.nanoTime() - startTime)/1000000; //время обновления игрового мира

           waitTime = target - timeMillis; //время ожидания перед след. итерацией

           try {

               this.sleep(waitTime);

           }

           catch(Exception e) { }

       }

   }

}

С точки зрения архитектуры игровое приложение лучше всего подразделить на множество смысловых классов, каждый из которых отвечает за выполнение определенного набора функций приложения в целом, либо описывает поведение конкретного игрового объекта или его компонента.

Реализация архитектуры игрового приложения рассматривается на примере разработанной игры “Воздушный бой”, основная цель которой сбить как можно больше самолетов противника за боевой вылет. На рисунке 3.2 отображена архитектура игры в виде диаграммы классов.

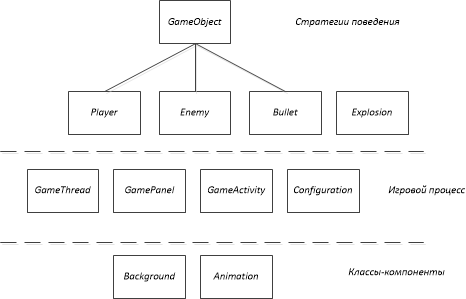


Рисунок 3.2. Архитектура разработанного игрового приложения

Была выбрана объектно-компонентная модель построения архитектуры. Представленные классы разделяются на три основные категории:

1. реализующие игровой процесс;
2. реализующие стратегии поведения отдельных объектов;
3. реализующие дополнительные компоненты игровых объектов.

Классы, реализующие игровой процесс - это классы приложения, в которых  описывающие основные механизмы: отображения объектов и необходимых расчетов. Этими классами являются:

1. GameThread, реализующий шаблон проектирования “Игровой цикл”;
2. Configuration, содержащий значения констант;
3. GamePanel, реализующий механизмы “прослушивания” событий, отображение пользовательского интерфейса и объектов на игровой сцене;
4. GameActivity, описывающий экран приложения.

Классы, описывающие игровые объекты - это классы, которые заключают необходимые свойства и стратегии поведения различных игровых элементов. К таким классам относятся:

1. GameObject, описывающий общие свойства игровых объектов и методы доступа к ним;
2. Player, реализующий поведение самолета игрока;
3. Enemy, реализующий поведение самолета противника;
4. Bullet, реализующий поведение снаряда, включая функцию-проверки на столкновение с другими объектами;
5. Explosion, реализующий отображение взрыва.

Классы-компоненты - это модули, дополняющие модель поведения игрового объекта. В рамках приложения был разработан единственный дополнительный класс-компонент Animation, позволяющий преобразовывать многокадровый спрайт игрового объекта в анимацию.

* 1. Роль шаблонов проектирования в архитектуре игрового приложения

Разработка компьютерной игры представляет собой сложную задачу из-за большого количества отдельных компонентов и связей между ними. Немаловажную роль при создании архитектуры приложения играют паттерны проектирования, проверенные временем решения, применяемые для создания основы программы.

Шаблоны, которые использовались в процессе разработки рассматриваемого в рамках статьи проекта можно разделить на две категории:

1. специализированные, применяются в основном при разработке игр;
2. классические, популярные решения распространенных проблем, возникающих на этапе проектирования программного обеспечения.

В процессе разработки игрового приложения применялись классические паттерны проектирования:

**Контроллер**. Единая точка управления отображением и выполнения приложения. Функции контроллера выполняет класс GamePanel. В листинге 3.3 приведены его основные методы.

Листинг 3.3. Основные методы класса GamePanel

public class GamePanel extends SurfaceView implements SurfaceHolder.Callback {

   //Игровой цикл

   private GameThread gameThread;

   //Отображение

   private Background bg; //фон

   public static Player player; //самолет игрока

   public static ArrayList<Enemy> enemies; //самолеты противников

   public static ArrayList<Explosion> explosions; //взрывы

   //Конструктор

   public GamePanel(Context context) { }

   //Метод, выполняющийся при закрытии окна приложения

   @Override

   public void surfaceDestroyed(SurfaceHolder holder) { }

   //Метод инициализации + запуск игрового потока

   @Override

   public void surfaceCreated(SurfaceHolder holder) { }

   //"Прослушивание" события касания экрана

   @Override

   public boolean onTouchEvent(MotionEvent e) { }

   //Отображение игрового проСранства <- ЛОШАДИ С ХУЯМИ

   @Override

   public void draw(Canvas canvas) { }

   //Расчет необходимых значений (позиции игровых объектов, их состояний) для "кадра" игры

public void update() { }

   //Отображение игрового интерфейса

   private void drawGameInterface(Canvas canvas) { }

}

**Состояние.** Шаблон позволяет изменять модель поведения объекта в зависимости от его текущего состояния. Данный паттерн был использован при реализации момента уничтожения самолета (игрока и противника). Т.о. в момент когда здоровье самолета меньше установленной единицы его спрайт заменяется анимацией взрыва. Реализация данного паттерна представлена в листинге 3.4.

Листинг 3.4. Реализация момента уничтожения самолета

//Запуск осуществляется в случае если на сцене есть самолеты противника

if(GamePanel.enemies.size() != 0) {

for (Enemy e : GamePanel.enemies) {

//снаряд игрока попал в самолет противника

if (b.checkHit(e)) {

e.getDamage(b.getDamage());

this.score += 5;

//если самолет противника уничтожен: начисляем очки игроку и "взрываем" самолет противника

if (e.getHealth() <= 0) {

GamePanel.explosions.add(GamePanel.createExplosion(e.getX(), e.getY(), e.getSpeed() \* (-1)));

GamePanel.enemies.remove(e);

this.score += 20;

}

bullets.remove(i);

break;

}

}

}

**LazyLoading.** Позволяет оптимизировать процесс управления ресурсами. В контексте разработанного приложения шаблон применяется для создания спрайтов противника/снарядов и удаления их, если они находятся за пределами игрового поля

* 1. Пользовательское управление игровым процессом

Операционная система Android поддерживает множество датчиков для получения различных сведений об устройстве. При разработке механик пользовательского управления приложениями чаще всего применяются два встроенных датчика в мобильное устройство: гироскоп и акселерометр. С помощью них можно узнать углы наклона устройства в трех плоскостях XY, YZ, ZX. На рисунке 3.5 представлены все возможные положения мобильного устройства в пространстве.

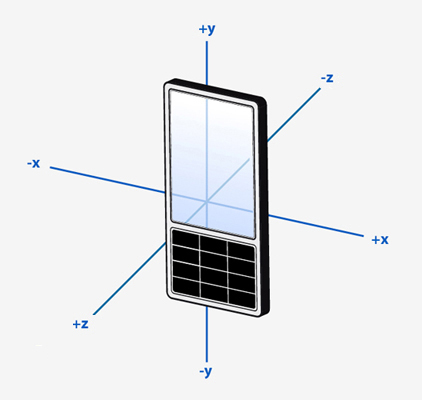


Рисунок 3.5. Положение мобильного устройства в пространстве

В процессе игры самолет игрока может набирать и опускаться до определенной высоты. Опытным путем было обнаружено, что наилучшей плоскостью для реализации этой механики управления с применением акселерометра является XY. В этом положении пользователь держит устройство перед собой и, совершая “рулевые” движения влево и вправо, может набирать высоту, либо снижаться. На рисунках 3.6а и 3.6б представлены основные положения экрана мобильного устройства для управления игровым процессом.



Рисунок 3.6а. Движение самолета игрока вниз при наклоне экрана



Рисунок 3.6б. Движение самолета игрока вверх при наклоне экрана

Встроенный акселерометр очень чувствителен к любому изменению угла устройства в пространстве. Поэтому для более точного отслеживания пользовательского события наклона мобильного устройства необходимо устанавливать наименьшее пороговое значение угла наклона. В листинге 3.7 представлена реализация события изменения угла отклонения мобильного устройства.

Листинг 3.7. Отслеживание события изменения угла мобильного устройства

private long lastUpdate = 0;

public static float aY;

private float turnThreshold = 1.5f; //пороговое значение угла в радианах

@Override

public void onSensorChanged(SensorEvent e) {

Sensor sensor = e.sensor;

if(sensor.getType() == Sensor.TYPE\_ACCELEROMETER) {

long curTime = System.currentTimeMillis();

//значения акселерометра берутся каждый 10 миллисекунд

if((curTime - lastUpdate) > 10) {

lastUpdate = curTime;

aY = e.values[1];

//самолет летит вверх

if(aY < turnThreshold \* -1) {

UP = true;

PLAIN = false;

}

//летит вниз

else if (aY > turnThreshold) {

UP = false;

PLAIN = false;

}

//летит плавно

else

PLAIN = true;

}

}

}

* 1. Отображение игровых объектов

В двумерной игре игровой объект представляет собой спрайт, который может состоять из одного изображения, либо из нескольких в случае анимации. В рамках проекта был разработан класс анимации для игровых объектов, основные части которого представлена в листинге 3.8.

Листинг 3.8. Класс-компонент анимации для игрового объекта

public class Animation {

   private Bitmap[] frames; //кадры анимации

   private int currentFrame; //текущий кадр

   private long startTime; //время начало анимации

   private long delay; //задержка между кадрами анимации

   private boolean playedOnce; //одиночная/постоянная анимации

   ...

   //Расчет текущего кадра анимации

   public void update() {

       long elapsed = (System.nanoTime() - startTime)/1000000;

       if(elapsed > delay) {

           currentFrame++; //изменение отображаемого кадра анимации

           startTime = System.nanoTime();

       }

       if(currentFrame == frames.length) {

           currentFrame = 0;

           playedOnce = true;

       }

   }

}

Для создания эффекта плавного перехода между изображениями анимации устанавливается параметр задержки, обозначающий продолжительность отображения каждого отдельного кадра анимации игрового объекта.

Отображение конкретного кадра анимации реализовано на стороне игрового объекта с помощью стандартных средств Android SDK для работы с графикой. В листинге 3.9 представлен вариант реализации отображения игрового объекта в пространстве.

Листинг 3.9. Отображение игрового объекта на экране устройства

//Конструктор игрового объекта

public Player(Bitmap s, int w, int h, int numFrames, long delay) {

...

       setUpAnimation(numFrames, delay);

...

}

//Преобразование целого спрайта в коллекцию (создание анимации)

private void setUpAnimation(int numFrames, long d) {

Bitmap[] spriteFrames = new Bitmap[numFrames];

for(int i = 0; i < spriteFrames.length; i++) {

spriteFrames[i] = Bitmap.createBitmap(sprite, width \* i, 0, this.width, this.height);

}

animation.setFrames(spriteFrames);

animation.setDelay(d);

startTime = System.nanoTime();

}

//Отображение игрового объекта

public void draw(Canvas canvas) {

canvas.drawBitmap(animation.getImage(), x, y, null);

}

Процесс создания анимации из спрайта предполагает его разбиение на одинаковые части, каждая из которых сохраняется в коллекцию кадров анимации. Единственным условием является то, что все кадры должны иметь одинаковую размерность.

1. Разработка системы статистического, морфологического, кластерного и онтологического анализа текстов и данных для ЖКХ
   1. Техническое задание на создание системы

В данном разделе приводится техническое задание на разработку системы статистического, морфологического, кластерного и онтологического анализа данных для ЖКХ.

* + 1. Назначение и цели создания системы

Данная система предназначена для общего круга лиц и предусматривает использование ее любым пользователем, которому требуется провести комплексный анализ текстов.

Система создается для того, чтобы:

1. автоматизировать расчет статистических характеристик анализируемых текстов;
2. автоматизировать определение списка терминов текста и степень их значимости.
   * 1. Характеристика объекта автоматизации

Объектом автоматизации системы является автоматическое выделение терминов из текста по предметной области.

* + - 1. Общее описание

Данная система предназначается для автоматизации проведения анализа текстовых данных. Система должна включать в себя статистический, морфологический, кластерный анализы для определения набора терминов обрабатываемого текста. Статистический анализ теста должен проводиться как для взятых слов, так и для словосочетаний. Морфологический анализ должен быть организован с применением специального программного обеспечения “Mystem”. Кластерный анализ должен быть реализован на основе метода К-средних.

* + - 1. Структура и принципы функционирования

Структура разрабатываемой системы должна быть последовательной и должна состоять из следующих ключевых этапов:

1. выбор обрабатываемых файлов и последующая их загрузка в память;
2. проведение морфологического анализа загруженных текстов;
3. расчет статистические характеристики для отдельных слов и словосочетаний;
4. поиск терминов.
   * 1. Требования к функциям, выполняемым системой

В разделе приводиться описание требований к основным функциям, выполняемым системой статистического, морфологического, кластерного и онтологического анализа данных для ЖКХ.

* + - 1. Функция загрузки обрабатываемых данных

Загрузка данных для обработки в разрабатываемом программном обеспечении должна производиться из текстовых файлов. Данные этих файлов должны быть загружены полностью в память программы. В программном обеспечении должна быть реализована возможность как загрузки одного файла, так и параллельное чтение данных из файлов в нескольких потоках.

* + - 1. Функция морфологического анализа текста

Функция должна выделять слова из загруженных на обработку текстов. Исключать знаки препинания, пробелы и другие разделители. Функция должна приводить все слова в каноническую форму для повышения точности выполнения последующих функций программы. Для реализации данной функции является обязательным использование специального программного обеспечения “Mystem”. Приложение должно быть встроено в реализуемое программное обеспечение, а обмен данными построен на основе файлового взаимодействия.

* + - 1. Функция статистического анализа слов и словосочетаний

Назначение функции заключается в произведении расчетов статистических характеристик слов и словосочетаний. Этими характеристиками являются:

1. частота встречаемости слова/словосочетания в тексте;
2. статистическая значимость встречаемости слов словосочетания в тексте по методу Mutual Information;
3. степень взаимосвязи двух слов по методу T-Score;
4. степень значимости словосочетания в тексте по методу Log-Likelihood;
5. степень значимости словосочетания в тексте по методу TF×IDF.
   * + 1. Функция кластерного анализа текста и определения набора терминов

Функция осуществляет поиск слов/словосочетаний-терминов в тексте на основе данных, полученных в ходе статистического анализа.

* 1. Информационное обеспечение системы

В данном разделе рассматриваются структуры источников данных для системы статистического, морфологического, кластерного и онтологического анализа данных для ЖКХ. Отдельно приводится описание онтологии предметной области.

* + 1. Организация входных данных

Основными входными данными системы являются текстовые файлы формата .pdf, .txt. Файлы могут иметь произвольную структуру.

* + 1. Организация обработки данных

Обработка данных выполняется по порядку в соответствии с описанием функций программы в разделе техническое задание. Каждая функция выполняется только при успешном выполнении всех предыдущих. Перед вызовом каждой функции возможна предварительная настройка их параметров.

* + 1. Организация выдачи информации

Выдача промежуточных и конечного результатов работы системы осуществляется с помощью специальных элементов пользовательского интерфейса. Результаты, полученные в ходе вычислений, могут быть экспортированы в файл формата .json для последующего использования, например для визуализации данных.

* + 1. Онтология предметной области

Онтология дает возможность представить данные в виде упорядоченной иерархической структуры. В рамках разработанной системы для анализа слабоструктурированных данных, текстов, была создана онтология, которая отражает особенности предметной области.

На рисунке 4.1 представлена онтология, дающая общее представление об основных объектах сферы ЖКХ.

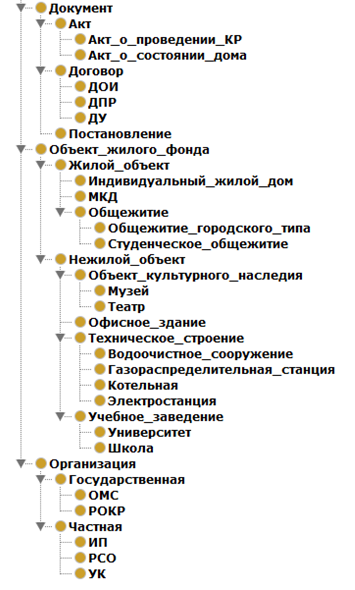


Рисунок 4.1. Структура онтологии

Онтология подразделяется на 3 группы классов:

1. объекты жилого фонда, многоквартирные дома, муниципальные учреждения и т.д.;
2. организации, производящие контроль и управление над объектами жилого фонда;
3. документы, которые описывают основные положения бизнес-процессов управления и контроля.

В таблице 4.2 приводится подробное описание представленной онтологии.

Таблица 4.2. Описание онтологии предметной области

| Название | Описание | Data property | Примечание |
| --- | --- | --- | --- |
| Документ | Класс «Документ» объединяет в себе все возможные документы касающихся состояния дома и управления им. | Дата начала действия: dateTime,  Дата окончания действия: dateTime,  Дата подписания: dateTime,  Номер: string | Содержит в себе подклассы: Акт и Договор. |
| Акт | Класс «Акт» предназначен для сбора информации об актах на состояние дома и проведения КР. | Положения: string,  Состояние дома: string | Является подклассом класса «Документ».  Содержит в себе подклассы: Акт о проведении КР и Акт о состоянии дома. |
| Акт о проведении  Капитального ремонта (КР) | Класс «Акт о проведении КР» предназначен для сбора информации о проведении в домах КР. | Дата проведения КР: dateTime | Является подклассом класса «Акт». |
| Акт о состоянии дома | Класс «Акт о состоянии дома» предназначен для сбора информации о состоянии дома. | - | Является подклассом класса «Акт». |
| Договор | Класс «Договор» предназначен для сбора информации о договорах на управление, поставку ресурсов и общего имущества. | Обязательства: string | Является подклассом класса «Документ».  Содержит в себе подклассы: Договор общего имущества, Договор поставки ресурсов и Договор управления. |
| Договор общего имущества (ДОИ) | Класс «Договор общего имущества» предназначен для сбора информации о договорах общего имущества. | Организация: string | Является подклассом класса «Договор». |
| Договор поставки ресурсов (ДПР) | Класс «Договор поставки ресурсов» предназначен для сбора информации о поставке ресурсов. | РСО: string | Является подклассом класса «Договор». |
| Договор управления (ДУ) | Класс «Договор управления» предназначен для сбора информации о договорах управления. | Управляющая организация: string | Является подклассом класса «Договор». |
| Постановление | Класс «Постановление» предназначен для сбора информации о постановлениях. | Закон: string | Является подклассом класса «Документ». |
| Объект жилого фонда | Класс «Объект жилого фонда» предназначен для сбора информации о жилых и нежилых объектах. | Аварийность: boolean,  Адрес: string,  Год постройки: dateTime,  Описание: string,  Площадь: decimal,  Этажность: int | Содержит в себе подклассы: Жилой объект и Нежилой объект |
| Жилой объект | Класс «Жилой объект» предназначен для сбора информации о индивидуальных жилых домах, многоквартирных домах и общежитие. | Жилая площадь: decimal,  Тип управления: string,  Число жителей: int | Является подклассом класса «Объект жилого фонда».  Содержит в себе подклассы: Индивидуальный жилой дом, Многоквартирный дом и Общежитие |
| Индивидуальный жилой дом | Класс «Индивидуаль-ный жилой дом» предназначен для сбора информации о индивидуальных жилых домах. | Наличие канализации: boolean,  Площадь участка: decimal | Является подклассом класса «Жилой объект». |
| Многоквартир-ный дом (МКД) | Класс «Многоквартирный дом» предназначен для сбора информации о многоквартирных домах | Количество квартир: int,  Количество лифтов: int,  Количество подъездов: int,  Наличие технического этажа: boolean | Является подклассом класса «Жилой объект». |
| Общежитие | Класс «Общежитие» предназначен для сбора информации о общежитиях городского типа и студенческих общежитиях | Количество комнат: int,  Наличие вахты: boolean | Является подклассом класса «Жилой объект». Содержит в себе подклассы: Общежитие городского типа и Студенческое общежитие |
| Общежитие городского типа | Класс «Общежитие городского типа» предназначен для сбора информации об общежитиях городского типа. | - | Является подклассом класса «Общежитие». |
| Студенческое общежитие | Класс «Студенческое общежитие» предназначен для сбора информации о студенческих общежитиях. | - | Является подклассом класса «Общежитие». |
| Нежилой объект | Класс «Нежилой объект» предназначен для сбора информации об объектах культурного наследия, офисных зданиях, технических строениях и учебных заведениях. | - | Является подклассом класса «Объект жилого фонда».  Содержит в себе подклассы: Объект культурного наследия, Офисное здание, Техническое строение и Учебное заведение |
| Объект культурного наследия | Класс «Объект культурного наследия» предназначен для сбора информации о музеях и театрах. | Максимальное число посетителей: int | Является подклассом класса «Нежилой объект».  Содержит в себе подклассы:  Музей и Театр |
| Музей | Класс «Музей» предназначен для сбора информации о музеях. | Количество выставочных помещений: int | Является подклассом класса «Объект культурного наследия». |
| Театр | Класс «Театр» предназначен для сбора информации о театрах. | Количество концертных залов: int | Является подклассом класса «Объект культурного наследия». |
| Офисное здание | Класс «Офисное здание» предназначен для сбора информации об офисных зданиях. | Количество организаций: int | Является подклассом класса «Объект жилого фонда». |
| Техническое строение | Класс «Техническое строение» предназначен для сбора информации о водоочистных сооружениях, газораспределительных станциях, котельных и электростанциях. | Вид ресурса: string,  Вид услуги: string,  Количество потребителей: int,  Производственная мощность: decimal | Является подклассом класса «Нежилой объект».  Содержит в себе подклассы:  Водоочистное сооружение, Газораспредели-тельная станция, Котельная и Электростанция |
| Водоочистное сооружение | Класс «Водоочистное сооружение» предназначен для сбора информации о водоочистных сооружениях. | Тип фильтрации: string | Является подклассом класса «Техническое строение». |
| Газораспреде-лительная станция | Класс «Газораспреде-лительная станция» предназначен для сбора информации о газораспределитель-ных станциях. | Диаметр труб: decimal,  Объем распределяемого газа: decimal | Является подклассом класса «Техническое строение». |
| Котельная | Класс «Котельная» предназначен для сбора информации о котельных. | Количество котлов: int,  Марка котла: string | Является подклассом класса «Техническое строение». |
| Электростанция | Класс «Электро-станция» предназначен для сбора информации о электростанциях. | Вид подстанции: string,  Напряжение: decimal | Является подклассом класса «Техническое строение». |
| Учебное заведение | Класс «Учебное заведение» предназначен для сбора информации о университетах и школах. | - | Является подклассом класса «Нежилой объект».  Содержит в себе подклассы:  Университет и Школа |
| Университет | Класс «Университет» предназначен для сбора информации о университетах. | - | Является подклассом класса «Учебное заведение». |
| Школа | Класс «Школа» предназначен для сбора информации о школах. | - | Является подклассом класса «Учебное заведение». |
| Организация | Класс «Организация» предназначен для сбора информации о государственных и частных организациях. | ИНН: string, Количество инженеров: int,  Количество работников: int,  Количество рабочих: int,  Контактные данные: string,  КПП: string,  Название: string,  ОГРН: string | Содержит в себе подклассы: Государственная и Частная. |
| Государственная | Класс «Государственная» предназначен для сбора информации о органах местного самоуправления и региональных операторах капитального ремонта. | Официальный представитель: string | Является подклассом класса «Организация».  Содержит в себе подклассы: Органы местного самоуправления и Региональный оператор капитального ремонта. |
| Органы местного самоуправления (ОМС) | Класс «Органы местного самоуправ-ления» предназначен для сбора информации о органах местного самоуправления. | - | Является подклассом класса «Государственная». |
| Региональный оператор капитального ремонта (РОКР) | Класс «Региональный оператор капитального ремонта» предназначен для сбора информации о региональных операторах капитального ремонта. | - | Является подклассом класса «Государственная». |
| Частная | Класс «Частная» предназначен для сбора информации об индивидуальных предпринимателях, ресурсоснабжающих организациях и управляющих компаниях. | Директор: string,  Одобренный подрядчик: boolean | Является подклассом класса «Организация».  Содержит в себе подклассы: Индивидуальный предпри-ниматель, Ресурсо-снабжающая организация и Управляющая компания. |
| Индивидуальный предпри-ниматель (ИП) | Класс «Индивидуаль-ный предприниматель» предназначен для сбора информации об индивидуальных предпринимателях. | - | Является подклассом класса «Частная». |
| Ресурсо-снабжающая организация (РСО) | Класс «Ресурсо-снабжающая организация» предназначен для сбора информации о ресурсоснабжающих организациях. | Тип обслуживания: string | Является подклассом класса «Частная». |
| Управляющая компания (УК) | Класс «Управляющая компания» предназначен для сбора информации об управляющих компаниях. | - | Является подклассом класса «Частная». |

Классы онтологии взаимосвязаны с помощью объектных слов, перечень которых представлен на рисунке 4.3.

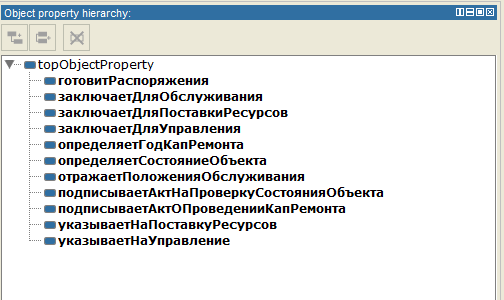


Рисунок 4.3. Объектные свойства онтологии

В таблице 4.4 представлено подробное описание объектных свойств онтологии.

Таблица 4.4. Описание объекных свойств онтологии

| Наименование | Описание |
| --- | --- |
| Готовит распоряжения | Свойство относится к государственной организации, а именно органам местного самоуправления. Указывает на подготовку ОМС распоряжения |
| Заключает для обслуживания | Свойство относится к частной организации, а именно индивидуальным предпринимателям. Указывает на подготовку договора общего имущества, заключенного для обслуживания объекта |
| Заключает для поставки ресурсов | Свойство относится к частной организации, а именно ресурсоснабжающим организациям. Указывает на заключение договора поставки энергетических ресурсов до объекта |
| Заключает для управления | Свойство относится к частной организации, а именно управляющим компаниям. Указывает на заключение договора управления объектом жилого фонда |
| Определяет год капитального ремонта | Свойство относится к документам, а именно актам о проведении капитального ремонта. Указывает на отношение договора к объекту жилого фонда |
| Определяет состояние объекта | Свойство относится к документам, а именно актам о состоянии дома. Указывает на отношение договора к объекту жилого фонда |
| Отражает положения обслуживания | Свойство относится к документам, а именно договорам общего имущества. Указывает на отношение договора к жилому объекту |
| Подписывает акт на поверку состояния объекта | Свойство относится к государственной организации, а именно региональному оператору капитального ремонта. Указывает на подписание акта поверки состояния объекта РОКР |
| Подписывает акт о проведении капитального ремонта | Свойство относится к государственной организации, а именно региональному оператору капитального ремонта. Указывает на подписание акта о проведении капитального ремонта РОКР |
| Указывает на поставку ресурсов | Свойство относится к договору на поставку ресурсов, а именно поставку ресурсов ресурсоснабжающими организациями. Указывает на заключение договора поставки энергетических ресурсов до объектов жилого фонда |
| Указывает на управление | Свойство относится к договору управления, а именно управления объектами жилого фонда. Указывает на заключение договора управления |

* 1. Алгоритмическое обеспечение системы

В разделе рассматриваются алгоритмы, разработанные в процессе создания системы статистического, морфологического, кластерного и онтологического анализа данных для ЖКХ. Реализация алгоритмов на ЯВУ C# представлена в приложении А.

* + 1. Алгоритм кластеризации

FCM (Fuzzy Classifier Means) - это алгоритм кластеризации, с помощью которого производится автоматическая классификация множества объектов, которые задаются векторами признаков в пространстве признаков. В случае рассматриваемой системы объектами является множество слов и словосочетаний. Их векторы признаков – наборы характеристик, получаемые в результате статистического анализа.

На схеме 4.5 представлена схема подпрограммы, которая реализует алгоритм кластеризации.



Схема 4.5. Блок-схема подпрограммы FCM-алгоритма

Входными данными для алгоритма являются:

1. количество итераций, определяющих точность выполнения алгоритма;
2. центры кластеров;
3. матрица, описывающая степень принадлежности каждого из объектов к центрам кластеров;
4. параметр нечеткости кластеризации. Рекомендуемое значение параметра выбирается в пределах ~1,5...2. При реализации алгоритма величина параметра была установлена в значение 1,6;
5. параметр сходимости алгоритма (уровень точности). Значение параметра установлено в 0.001.

На первом шаге алгоритма производится заполнение матрицы принадлежности объектов установленным центрам кластеров. Значения матрицы устанавливаются случайным образом, но с условием, что сумма значений вектора принадлежности отдельного объекта не превышает 1.0.

На втором шаге производится запуск основного цикла алгоритма кластеризации данных, регулируются значения центров кластеров по формуле:

, где – центр j-го кластера, – количество объектов, – значение i-Го объекта, – степень принадлежности объекта i кластеру j.

На третьем этапе происходит корректировка значений матрицы принадлежности по формуле:

, где – количество кластеров, – вектор центра j-го кластера, – вектор центра l-го кластера.

По завершении каждой итерации цикла алгоритма производятся проверки:

1. сравнения числа уже выполненных итераций с заранее предустановленным количеством;
2. сравнения параметра сходимости алгоритма и матричной нормы, которая должна быть больше для продолжения работы алгоритма.

При выполнении хотя бы одного из условий алгоритм считается завершенным.

Результом работы алгоритма является итоговая матрица принадлежности объектов к каждому из кластеров.

* 1. Программное обеспечение системы

В данном разделе пояснительной записки рассматривается структура разработанного программного обеспечения на уровне модулей и отдельных компонентов.

* + 1. Структура программного обеспечения

Приложение для проведения анализа текстов состоит из  четырех компонентов:

1. Core, ядро, содержащее в себе реализацию всех необходимых для проведения расчетов алгоритмов, параллельной обработки текстов, API для внешнего использования, среды выполнения программы морфологического анализа текстов mystem;
2. Helper, модуль, реализующий вспомогательные методы для работы с файловой системой;
3. FileHandler, реализующий функционал загрузки данных из файлов разных типов;
4. DesktopApplication, приложение для проведения анализа текстов для ОС Windows.
   * + 1. Модуль Core

Компонент Core, являющийся ядром разработанного программного обеспечения, включает в себя семь классов:

1. API.cs, модуль, реализующий внешний API, который используется при разработке пользовательского приложения;
2. ClasterAnalysis.cs, модуль, реализующий FCM-алгоритм кластеризации данных;
3. CommonClasses.cs, модуль, включающий описания всех используемых структур данных;
4. Configuration.cs, модуль, описывающий основные константы приложения (например, обозначения частей речи);
5. MorphologicalAnalysis.cs, модуль, реализующий методы для морфологического анализа заданного текста;
6. Multiprocessor.cs, модуль, обеспечивающий возможность параллельной обработки нескольких текстов;
7. MystemProvider.cs, модуль, реализующий среду для выполнения программы морфологического анализа текстов mystem
8. StatisticsAnalysis.cs, модуль, включающий реализацию алгоритмов для проведения статистического анализа

В таблице 4.6 представлено описание методов класса API.cs.

Таблица 4.6. Спецификация методов класса API.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public List<string> LoadFile(string path) | Преобразование файла в список строк его данных |
| 2. public List<FileData> LoadFilesMulticore(List<string> paths) | Многопоточное преобразование файлов в списки строк их данных |
| 3. public List<Lemm> HandleByMystem(List<string> fileLines) | Обработка mystem строк данных файла и получение списка лемм слов, содержащихся в тексте из файла |
| 4. public MystemData HandleByMystem(string filePath) | Обработка файла программой mystem и получение списка лемм слов, содержащихся в тексте из файла |
| 5. public List<MystemData> HandleByMystemMulticore(List<FileData> dataList) | Многопоточная обработка данных файлов и получение списков лемм слов |
| 6. public List<MystemData> HandleByMystemMulticore(List<string> filesPaths) | Многопоточная обработка нескольких файлов программой mystem |
| 7. public StatsAnalysisResult<string> ProvideWordsStatsAnalysis(List<Lemm> list) | Проведение статистического анализа на основе результатов, полученных в после обработки файла программой mystem. Результат работы функции - статистические характеристики слов в тексте. |
| 8. public StatsAnalysisResult<WordDigram> ProvideDigramsStatsAnalysis(MystemData data) | Проведение статистического анализа на основе результатов, полученных в после обработки файла программой mystem. Результат работы функции - статистические характеристики двусловий в тексте. |
| 9. public List<StatsAnalysisResult<string>> ProvideWordsStatsAnalysisMulticore(List<MystemData> list) | Проведение многопоточного статистического анализа для отдельных слов |
| 10. public List<StatsAnalysisResult<WordDigram>> ProvideDigramsStatsAnalysisMulticore(List<MystemData> list) | Проведение многопоточного статистического анализа для двусловий |
| 11. public MystemData ProvideMorphAnalysis(MystemData data, string[] excludedTypes) | Проведение морфологического анализа для одного текста |
| 12. public List<MystemData> ProvideMorphAnalysisMulticore(List<MystemData> list, string[] excludedTypes) | Проведение многопоточного морфологического анализа для данных нескольких текстов |
| 13. public double[,] GetDefaultClustersCenters<T>(Dictionary<T, double[]> statisticDictionary) | Поиск центров для проведения кластерного анализа на основе данных, полученных в результате выполнения кластерного анализа |
| 14. public ClasterAnalysisResult<T> ProvideClusterAnalysis<T>(ClasterSettings<T> settings, string name) | Проведение кластерного анализа слов/двусловий (зависит от типа входных данных) для данных одного текста |
| 15. public List<ClasterAnalysisResult<string>> ProvideWordClusterAnalysisMulticore(List<ClasterAnalysisData<string>> data) | Проведение многопоточного кластерного анализа слов для нескольких текстов |
| 16. public List<ClasterAnalysisResult<WordDigram>> ProvideWordDigramClusterAnalysisMulticore(List<ClasterAnalysisData<WordDigram>> data) | Проведение многопоточного кластерного анализа двусловий из нескольких текстов |
| 17. public Dictionary<T, double[]> GetDataReady<T>(StatsAnalysisResult<T> analysisResult) | Преобразование данных, полученных в результате статистического анализа в формат для проведения кластерного анализа |

В таблице 4.7 рассматривается спецификация методов класса ClasterAnalysis.cs

Таблица 4.7. Спецификация методов класса ClasterAnalysis.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. private List<T> FillDataMatrix(Dictionary<T, double[]> wordStats) | Заполнение матрицы данных из словаря |
| 2. private void InitializeMatrix() | Инициализация матрицы принадлежности |
| 3. private double CalculateEuclidLength(int i\_data, int j\_cl) | Получение Евклидова расстояния между векторами |
| 4. private double CalculateNewMemberDegreeItem(int i, int j) | Расчет нового значения матрицы принадлежности |
| 5. private void CalculateClusterCenters() | Функция перерасчета центров кластеров, согласно данным матрицы принадлежности на i-ой итерации выполнения кластерного анализа |
| 6. private double RecalculateMemberDegree() | Перерасчет матрицы принадлежности |
| 7. public Dictionary<T, double[]> Clasterize() | Реализация алгоритма кластеризации данных |

В таблице 4.8 рассматривается спецификация структур данных, определенных в классе  CommonClasses.cs.

Таблица4.8**.** Спецификация методов класса CommonClasses.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public class FileData | Класс, описывающий данные файла |
| 2. public class MystemData | Класс, применяющийся для сохранения данных, полученных в результате обработки файла программой mystem |
| 3. public class Lemm | Класс, описывающий лемму слова (начальную форму) |
| 4. public class Analysis | Класс, описывающий морфологические признаки слова, тип части речи |
| 5. public class WordDigram | Класс, описывающий двусловие |
| 6. public class StatsAnalysisResult<T> | Класс, предназначенный для описания результата статистического анализа слов/двусловий из заданного текста |
| 7. public class ClasterSettings<T> | Класс, описывающий входные параметры кластерного анализа |
| 8. public class ClasterAnalysisData<T> | Класс, описывающий результат выполнения кластерного анализа |

В таблице 4.9 рассматривается спецификация основных методов, класса  Multiprocessor.cs.

Таблица4.9. Спецификация методов класса Multiprocessor.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public void MultiprocessorMorphAnalysis(List<MystemData> list, string[] excludedTypes) | Метод, обеспечивающий проведение морфологического анализа на основе данных нескольких текстов в параллельном режиме |
| 2. public void MultiprocessorWordDigramClasterAnalysis(List<ClasterAnalysisData<WordDigram>> list) | Метод, обеспечивающий проведение кластерного анализа двусловий на основе данных нескольких текстов в параллельном режиме |
| 3. public void MultiprocessorWordClasterAnalysis(List<ClasterAnalysisData<string>> list) | Метод, обеспечивающий проведение кластерного анализа слов на основе данных нескольких текстов в параллельном режиме |
| 4. public void MultiprocessorWordStatsAnalysis(List<MystemData> list) | Метод, обеспечивающий проведение статистического анализа слов для данных нескольких текстов в параллельном режиме |
| 5. public void MultiprocessorDigramsStatsAnalysis(List<MystemData> list) | Метод, обеспечивающий проведение статистического анализа двусловий на основе данных нескольких текстов в параллельном режиме |
| 6. public void MultiprocessorMystemHandler(List<FileData> list) | Метод, обеспечивающий проведение параллельной обработки текстов программой mystem.exe |
| 7. public void MultiprocessorFileRead(List<string> paths) | Метод, обеспечивающий чтение данных из нескольких текстовых файлов (поддержка расширений .pdf, .txt) в параллельном режиме |

В таблице 4.10 представлены описания методов класса MystemProvider.cs.

Таблица 4.10. Спецификация методов класса MystemProvider.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public List<Lemm> LaunchMystem(List<string> lines, string flags = "-cgin --format json") | Метод, обеспечивающий запуск приложения mystem.exe с предустановленным режимом записи результата в файл с расширением .json. |
| 2. private List<Lemm> GetMystemResult(StreamReader srdr) | Преобразование данных json-файла, полученного в результате выполнения приложения mystem.exe, в список отдельных лемм |

В таблице 4.11 представлены описания методов класса StatiscticsAnalysis.cs.

Таблица 4.11. Спецификация методов класса StatisticsAnalysis.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public static Dictionary<string, double> GetFrequencyDictionary(List<string> words) | Получение частотного словаря слов текста |
| 2. public static Dictionary<string, double> GetTF(Dictionary<string, double> words, int count) | Нахождение абсолютной частоты встречаемости слова в тексте |
| 3. public static Dictionary<string, double> GetTF\_IDF(int ccSize, Dictionary<string, double> tf\_dictionary, int docNumber = 1) | Реализация алгоритма TFxIDF, предназначенного для поиска наиболее значимых слов |
| 4. public static Dictionary<WordDigram, double> GetDigramFrequenceDictionary(List<string> wordList) | Получение частотного словаря двусловий |
| 5. public static Dictionary<WordDigram, double> CalculateMutualInformation(Dictionary<WordDigram, double> frequencyDiagram, Dictionary<string, double> frequencyDictionary, int wordsCount) | Реализация алгоритма поиска наиболее значимых слвовосочетаний согласно методу Mutual Information |
| 6. public static Dictionary<WordDigram, double> CalculateTScore(Dictionary<WordDigram, double> frequencyDiagram, Dictionary<string, double> frequencyDictionary, int wordsCount) | Реализация алгоритма определения степени взаимосвязи двух слов по методу TScore |
| 7. public static Dictionary<WordDigram, double> CalculateLogLikelihood(Dictionary<WordDigram, double> frequencyDiagram) | Реализация алгоритма метода Log-Likelihood для выявления наиболее статистически значимых двусловий |
| 8. public static int GetNgramFrequence(List<string> wordList, List<string> ngramList) | Получение частотного словаря n-грамм для заданного текста |
| 9. public static Dictionary<T, double[]> MergeDictionaries<T>(this List<Dictionary<T, double>> dictionaries) | Преобразование данных, полученных в результате проведенных статистических методов, в единый словарь данных для кластерного анализа |
| 9. public static int GetWordsCount(this Dictionary<string, int> words) | Получение общего числа слов в тексте |
| 10. public static List<string> GetWords(this List<Lemm> list) | Получение списка лемм слов текста |

* + - 1. Модуль Helper.cs

Модуль Helper.cs содержит реализацию вспомогательных методов, которые используются для взаимодействия с файловой системой. В таблице 4.12 представлена спецификация методов модуля.

Таблица 4.12. Описание методов модуля Helper.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| 1. public static List<string> CheckFiles(List<string> paths) | Метод, обеспечивающий проверку на возможность дальнейшей обработки входного файла с заданным расширением |
| 2. public static void WriteFile(List<string> lines, string path) | Метод, обеспечивающий запись строк в файл |
| 3. public static bool DeleteFile(string path) | Метод, удаляющий файл по указанному пути |
| 4. public static FileHandler GetFileHandler(string path) | Метод, запускающий алгоритм загрузки в зависимости от расширения файла. Метод реализован с использованием паттерна проектирования “фабрика” |

* + - 1. Модуль FileHandler

Модуль FileHandler реализует алгоритмы обработки файлов. Данная подпрограмма состоит из трех классов:

1. FileHandler.cs, абстрактный класс, который описывает общие свойства и методы, необходимые для реализации обработчиков считывания данных из файлов с разными расширениями;
2. TxtHandler.cs, реализация чтения данных из файла, имеющего расширение .txt;
3. PdfHandler.cs, реализация чтения данных из файла, имеющего расширение .pdf.

В таблице 4.13 представлена спецификация методов класса FileHandler.cs

Таблица 4.13.  Спецификация методов абстрактного класса FileHandler.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| public abstract void ReadFile(out List<string> lines) | Абстрактный метод, в реализации которого необходимо описывать алгоритм чтения данных из файла с определенным расширением |

В  таблице 4.14 представлена спецификация методов класса PdfHandler.cs, реализующего чтение данных из файла с расширением .pdf.

Таблица 4.14. Спецификация методов класса PdfHandler.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| public override void ReadFile(out List<string> lines) | Метод, реализующий алгоритм загрузки данных из файла с расширением .pdf |

В  таблице 4.15 представлена спецификация методов класса TxtHandler.cs, реализующего чтение данных из текстового файла.

Таблица 4.15. Спецификация методов класса TxtHandler.cs

|  |  |
| --- | --- |
| Наименование | Назначение |
| public override void ReadFile(out List<string> lines) | Метод, реализующий алгоритм загрузки данных из текстового файла |

* + 1. Интерфейс пользователя с системой

Пользовательский интерфейс приложения состоит из шести форм, каждая из которых выполняет определенные функции.

При запуске приложения пользователю отображается форма, представленная на рисунке 4.16.

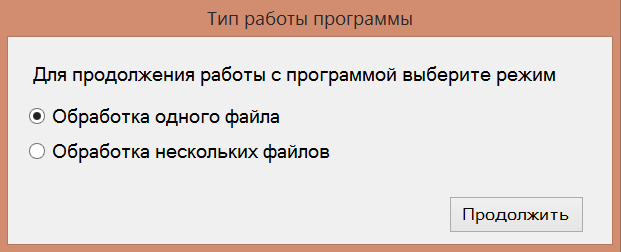


Рисунок 4.16. Выбор режима работы приложения

Пользователь может выбрать как вариант проведения анализа одного текста, так и нескольких. При этом необходимые вычисления будут проводиться в параллельном режиме

После определения режима работы приложения, пользователю необходимо выбрать файл (или группу файлов), который будет обработан программой. Выбор файла осуществляется с помощью диалогового окна, показанного на рисунке 17.

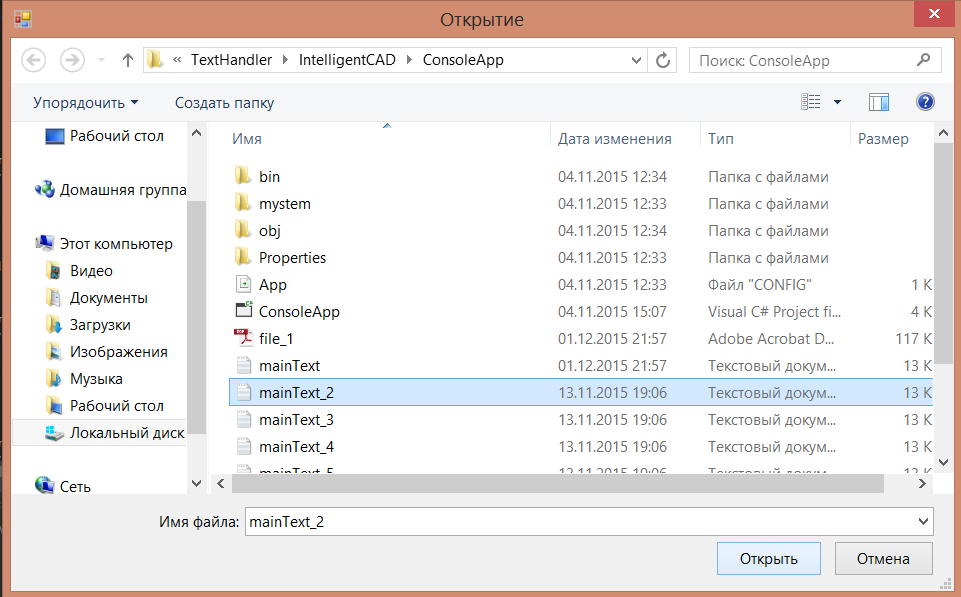


Рисунок 4.17. Процесс выбора файла для проведения анализа

Все выбранные файлы отображаются в списке в левой части окна приложения, как показано на рисунке 4.18.

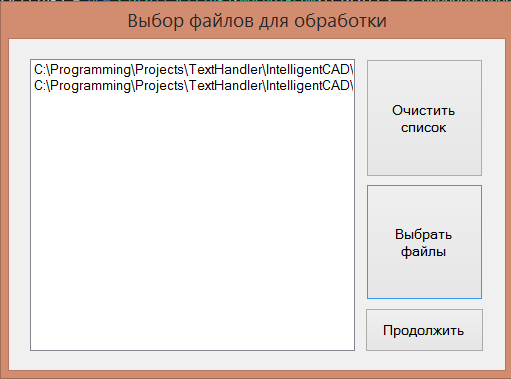


Рисунок 4.18. Отображение группы выбранных файлов

Также у пользователя есть возможность перезагрузить файлы, предварительно очистив список. Для этого нужно нажать на кнопку “Очистить список”, а затем выбрать новый файл (группу файлов) в диалоговом окне.

На следующем этапе производится обработка указанных файлов утилитой mystem.exe. При этом список лемм слов файла отображается в списке, как показано на рисунке 4.19.

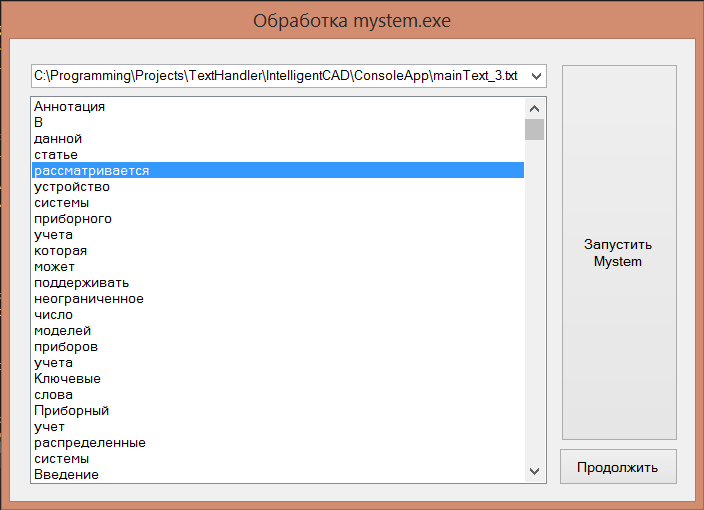


Рисунок 4.19. Отображение результата обработки файла приложением mystem.exe

Если обработка проводится для нескольких файлов, то отобразить результат для каждого из них можно, выбрав соответствующий файл в выпадающем меню в верхней части окна формы.

На следующем этапе пользователю предлагается удалить из дальнейшей обработки наименее значимые по своему смыслу части речи: союзы, предлоги и частицы. Интерфейс этого окна приложения представлен на рисунке 4.20.

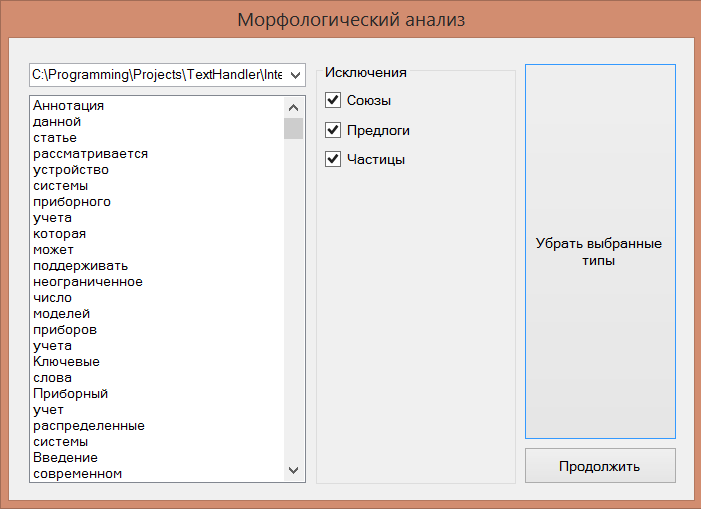


Рисунок 4.20. Удаление не значимых частей речи из списка лемм текста

Функционал следующей формы приложения позволяет провести статистический анализ двусловий текста (группы текстов). Анализ состоит из четырех характеристик:

1. частота встречаемости двусловий в тексте;
2. loglikelihood, определение наиболее стастически значимых двусловий;
3. mutual information, определение наиболее значимых двусловий;
4. TScore, степень взаимосвязи двух слов;

Отображение результатов статистического анализа представлено на рисунке 4.21.

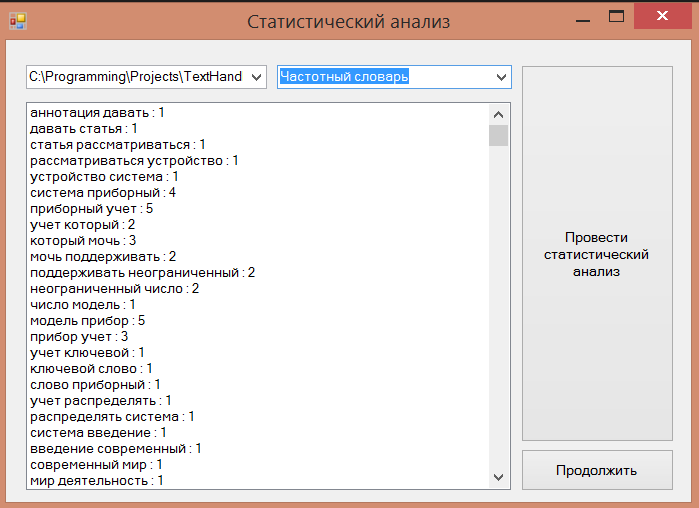


Рисунок 4.21. Окно приложения с результатами статистического анализа

Пользователю предоставляется возможность просмотра значений каждой из указанных характеристик, полученных в результате вычислений. Этот выбор можно произвести с помощью выпадающего меню в верхней части данного окна приложения.

На следующем этапе производится кластерный анализ словосочетаний текста. Результат выполнения кластерного анализа представлен на рисунке 4.22.

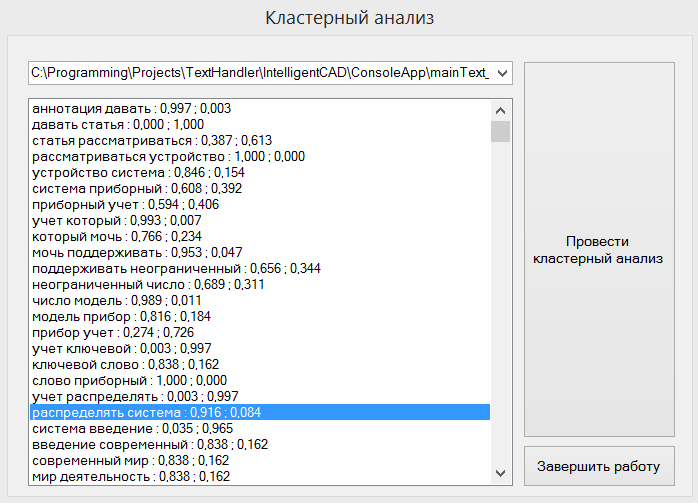


Рисунок 4.22. Форма отображения результатов кластерного анализа

* 1. Результаты проведенных исследований

В данном разделе пояснительной записки представлены результаты, полученные в ходе проведенных исследований, представлено обоснование выбора параметров центров для кластерного анализа.

* + 1. Область исследований

Первая задача, решаемая разработанной системой, заключается в сборе статистических данных о слабо структурированном наборе информации, тексте. В качестве предмета исследования была выбрана статья, описывающая систему приборного учета для сбора и хранения показаний со счетчиков, установленных в многоквартирных домах и других объектах сферы жилищно-коммунального хозяйства.

Система позволяет выполнять процедуру кластерного анализа на основании полученной статистической информации. Цель проведения такого анализа – выделение слов и словосочетаний, являющихся терминами из предметной области текста.

Для получения точных результатов перед проведением кластерного производится морфологический анализ. Он позволяет привести слова, составляющие текст, в каноническую форму. Также такой тип анализа позволяет определить части речи слов и удалить слова, имеющие наименьшую значимость в тексте (например, частицы, предлоги и союзы).

* + 1. Выбор центров кластеров

Целью проведения кластерного анализа является определение принадлежности слова/словосочетания к категориям термин/нетермин. Примером слова-термина является “прибор”, словосочетания “приборный учет” в рамках текста, на котором производились исследования.

Входными данными для кластерного анализа является массив значений статистических характеристик всех слов или словосочетаний. Кластеры определяются как набор с минимальными значениями, обозначающий “нижнюю границу”, и с максимальными – “верхнюю границу”.

Такой выбор центров позволяет четко обозначить области принадлежности словосочетаний/слов к центрам и повысить точность определения индекса значимости рассматриваемой единицы языка в тексте.

* + 1. Результат проведения кластерного анализа для словосочетаний

После проведения процедуры кластерного анализа словосочетаний в тексте был получен результат, представленный на рисунке 4.23.

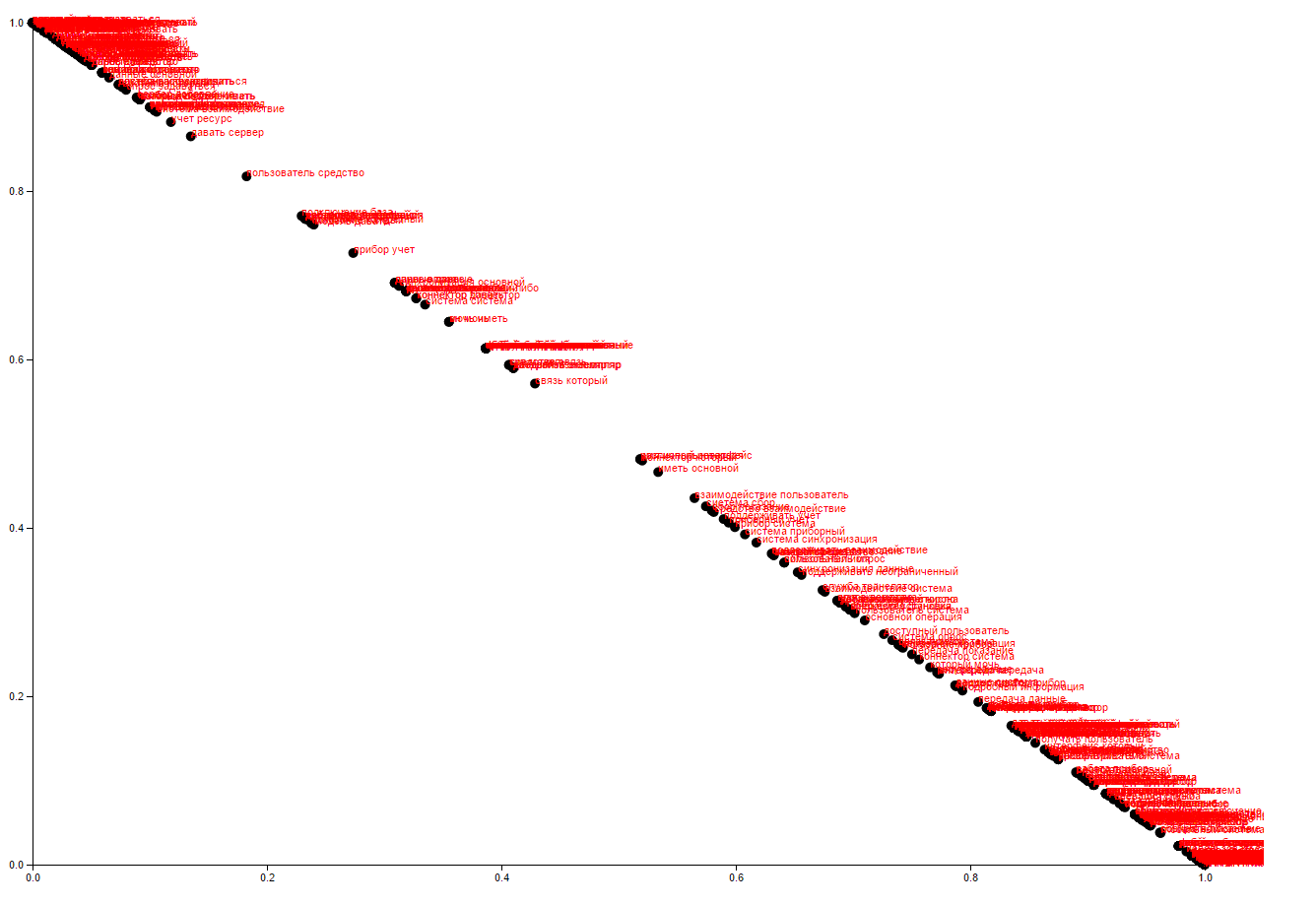


Рисунок 4.23. Визуализация результата проведенного кластерного анализа для словосочетаний

При подробном рассмотрении результатов можно увидеть, что словосочетания, имеющие наименьшую значимость, в своем большинстве входят в “нижний” или “верхний” кластеры. Значения принадлежности терминов к центрам обоих кластеров стремятся к равенству, что позволяет достаточно легко выделить такие словосочетания из текста.

* + 1. Результат проведения кластерного анализа для слов

Кластерный анализ позволяет точно разделить слова на категории термин/нетермин. На рисунке 4.24 представлен результат выполнения кластерного анализа для слов текста статьи о приборном учете.

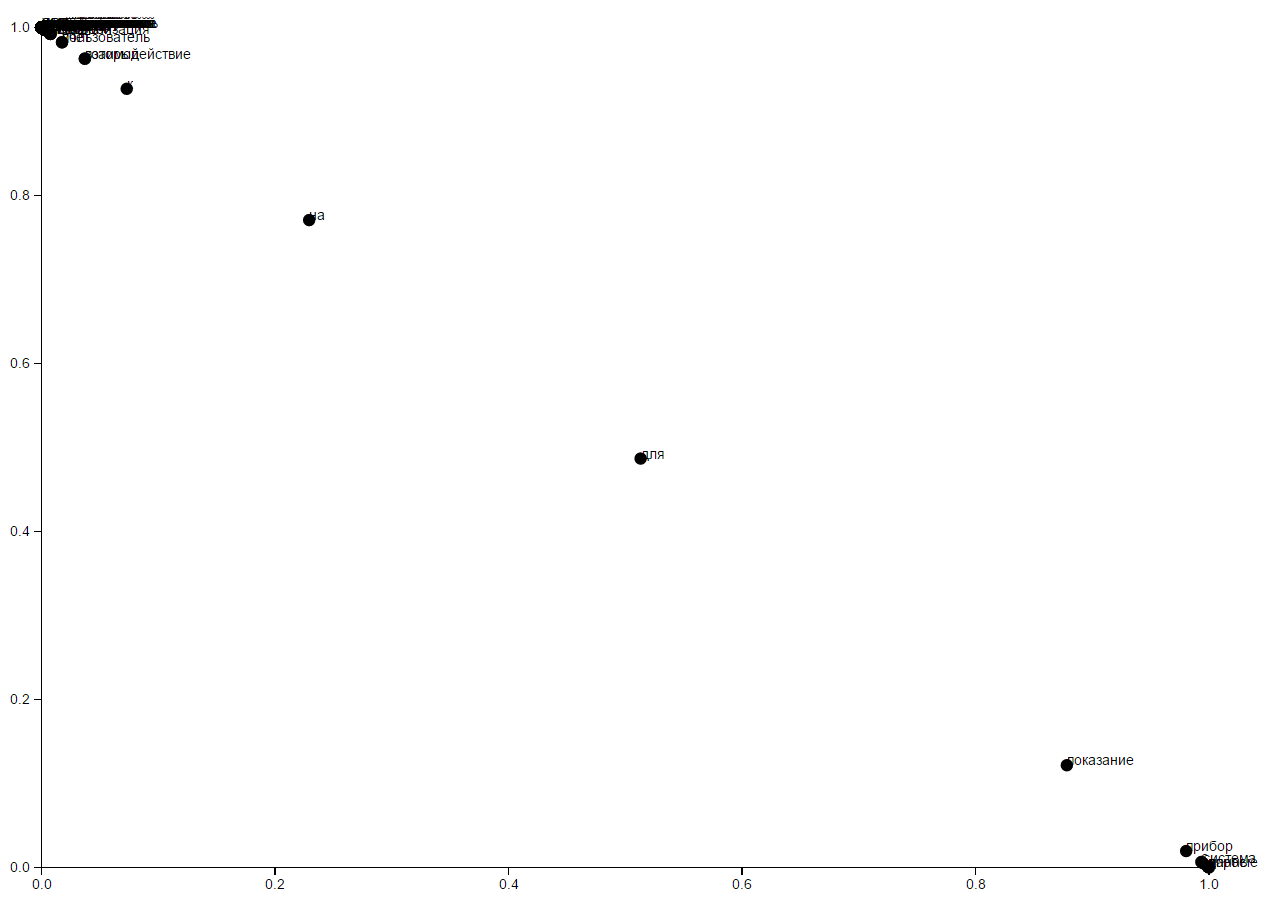


Рисунок 4.24. Визуализация результатов проведенного кластерного анализа для слов

Среди терминов можно видеть такие слова, как «система», «прибор», «с», «и», «данные», «показание», «опрос». Союзы и предлоги не являются терминами, поэтому они могут быть удалены из представленного списка путем вызова процедуры морфологического анализа.

1. Системный анализ паспортов многоквартирных домов (МКД) для выделения физических и технических характеристик “Объектов жилого фонда”
2. Список литературы

**Разработка математической модели для расчёта тепло-энергетических характеристик воды и водяного пара**

1. ГСССД 187-99. Вода. Удельный объем и энтальпия
2. Мякишев Г.Я. Физика 10 класс: учеб. для общеобразоват. организаций: базовый уровень / Г. Я. Мякишев, Б. Б. Буховцев, Н. А. Парфентьевой. – М. : Просвещение, 2014. – 416 с.
3. Советов Б. Я., Яковлев С. А., Моделирование систем: Учеб. для вузов — 3-е изд., перераб. и доп. — М.: Высш. шк., 2001. — 343 с.
4. Самарский А. А., Михайлов А. П. Математическое моделирование. Идеи. Методы. Примеры. — 2-е изд., испр. — М.: Физматлит, 2001.

**Анализ применения онтологии для кластеризации предметной области: “Объекты жилого фонда” для физических и технических характеристик**

**Разработка мобильного игрового приложения средствами Android-SDK с применением шаблонов проектирования**

1. Статья “Шаблон проектирования “Контроллер”. [Электронный ресурс] - Режим доступа: [<http://design-pattern.ru/patterns/application-controller.html>]
2. Статья “Шаблон проектирования “Состояние”. [Электронный ресурс] - Режим доступа: [<http://cpp-reference.ru/patterns/behavioral-patterns/state/>]
3. Статья “Шаблон проектирования Lazy Loading. [Электронный ресурс] - Режим доступа: [<http://design-pattern.ru/patterns/lazy-load.html>]
4. Статья “Using the Accelerometer on Android”. [Электронный ресурс] - Режим доступа: [<http://code.tutsplus.com/tutorials/using-the-accelerometer-on-android--mobile-22125>]
5. Статья “Собираем показания датчиков с Android смартфона”. [Интернет-ресурс] - Режим доступа: [<http://habrahabr.ru/post/137678/>]

**Разработка системы статистического, морфологического, кластерного и онтологического анализа текстов и данных для ЖКХ**

1. Гамма, Э. Приемы объектно-ориентированного программирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб. : Питер, 2001. – 344 с.
2. ГОСТ 19.701-90. Единая система конструкторской документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М. : Стандартинформ, 2010. – 8 с.
3. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М. : Стандартинформ, 1996. – 9 с.
4. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4 / Э. Троелсен. – 5-е изд. - М. : Вильямс, 2010. – 1392 с.
5. Фленов, М. Е. Библия C# / М.Е.Фленов. – 2-е изд. – СПб. : БХВ-Петербург, 2011. – 560 с.
6. Mukherjee, S. Thinking in LINQ / S. Mukherjee. – NY : Appress, 2015. – 259 pp.
7. Skeet, J. C# in depth. Third edition / J.Skeet. – Shelter Island, NY : Manning, 2014. – 614 pp.
8. Lee A., Powers E. Ontology-Aided Web Search Assistant / Lee A., Powers E. - [Электронный ресурс] - Режим доступа: [http://protege.stanford.edu/conference/2004/abstracts/Lee.pdf]
9. Dr. Singh M. Ontology Development and Query Retrieval using Protégé Tool / Singh M. - [Электронный ресурс] - Режим доступа: [http://www.mecs-press.org/ijisa/ijisa-v5-n9/IJISA-V5-N9-8.pdf]
10. Knublauch H., Musen M. A., Noy N. F. Creating Semantic Web (OWL) Ontologies with Protege / Knublauch H., Musen M. A., Noy N. F. - [Электронный ресурс] - Режим доступа: [http://iswc2003.semanticweb.org/pdf/Protege-OWL-Tutorial-ISWC03.pdf]

**Системный анализ паспортов многоквартирных домов (МКД) для выделения физических и технических характеристик “Объектов жилого фонда”**

Приложение А. Исходный код игрового мобильного приложения «Воздушный бой»

**Модуль Animation.java**

package com.example.alex.fighters;

import android.graphics.Bitmap;

/\*\*

\* Created by Alex on 13.09.2015.

\*/

public class Animation {

private Bitmap[] frames; //кадры анимации

private int currentFrame; //текущий кадр

private long startTime; //время начало анимации

private long delay; //задержка между кадрами анимации

private boolean playedOnce; //одиночная/постоянная анимации

public void Animation() { }

public void setFrames(Bitmap[] frames) {

this.frames = frames;

currentFrame = 0;

startTime = System.nanoTime();

}

public void setDelay(long d) {

this.delay = d;

}

public void setFrame(int i) {

this.currentFrame = i;

}

public int getFrame() {

return this.currentFrame;

}

public Bitmap getImage() {

return frames[currentFrame];

}

public boolean playedOnce() {

return playedOnce;

}

public void update() {

long elapsed = (System.nanoTime() - startTime)/1000000;

if(elapsed > delay) {

currentFrame++;

startTime = System.nanoTime();

}

if(currentFrame == frames.length) {

currentFrame = 0;

playedOnce = true;

}

}

public int getCurrentFrame() {

return currentFrame;

}

public int getFramesCount() {

return frames.length;

}

}

**Модуль Background.java**

package com.example.alex.fighters;

import android.graphics.Bitmap;

import android.graphics.Canvas;

/\*\*

\* Created by Alex on 13.09.2015.

\*/

public class Background {

private Bitmap bgImage;

private int x; //координата по Ox

private int y; //координата по Oy

private int dx; //скорость смещения

public Background(Bitmap bg) {

this.bgImage = bg;

this.dx = Player.SPEED;

}

//Расчет

public void update() {

this.x += dx;

if(x < -GamePanel.WIDTH) {

x = 0;

}

}

//Отрисовка background

public void draw(Canvas canvas) {

canvas.drawBitmap(bgImage, x, y, null);

if(x < 0) {

canvas.drawBitmap(bgImage, x + GamePanel.WIDTH, y, null);

}

}

}

**Модуль Bullet.java**

package com.example.alex.fighters;

import android.graphics.Canvas;

import android.graphics.Paint;

/\*\*

\* Created by Alex on 13.09.2015.

\*/

public class Bullet extends GameObject {

private int damage;

private int speed;

private int direction;

private Paint paint;

private Configuration.ObjectType targetType;

public Bullet(int dir, int x, int y, int s, int dmg, int r, int color, Configuration.ObjectType target) {

dy = 0;

direction = dir;

speed = s;

damage = dmg;

this.x = x;

this.y = y;

width = height = r;

type = Configuration.ObjectType.Missile;

targetType = target;

paint = new Paint();

paint.setColor(color);

paint.setStyle(Paint.Style.FILL);

}

public void draw(Canvas canvas) {

try {

canvas.drawCircle(x, y, width, paint);

}

catch (Exception e) { }

}

public void update() {

x -= (speed \* direction);

}

public boolean checkHit(GameObject obj) {

if(obj.type == targetType)

return obj.getRect().intersect(this.getRect());

return false;

}

public int getDamage() {

return damage;

}

}

**Модуль Configuration.java**

package com.example.alex.fighters;  
  
*/\*\*  
 \* Created by Alex on 13.09.2015.  
 \*/*public class Configuration {  
 public static int *BF\_HEIGHT* = 38;  
 public static int *BF\_WIDTH* = 90;  
 public static int *BF\_FRAMECOUNT* = 3;  
  
 public enum ObjectType { *Player*, *Enemy*, *Missile* }  
}

**Модуль Enemy.java**

package com.example.alex.fighters;  
  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
*/\*\*  
 \* Created by Alex on 13.09.2015.  
 \*/*public class Enemy extends GameObject {  
 private Bitmap sprite;  
 private int speed;  
 private int health;  
 private Animation animation = new Animation();  
 private Random rand = new Random(); //определяет направление движения по OY  
  
 private ArrayList<Bullet> bullets;  
 private long startShooting;  
 private long shootDelay = 1000;  
 private int shootSpeed;  
 private int shootStrength;  
 private Bullet b;  
  
 public Enemy(Bitmap s, int x, int y, int w, int h, int numFrames, long shD, int shSp, int shStr, int hlth) {  
 type = Configuration.ObjectType.*Enemy*;  
 this.x = x;  
 this.y = y;  
 width = w;  
 height = h;  
 sprite = s;  
 speed = 10;  
 dy = rand.nextInt(10) - 5;  
 setUpAnimation(numFrames, 100);  
 bullets = new ArrayList<Bullet>();  
 startShooting = System.*nanoTime*();  
 shootDelay = shD;  
 shootSpeed = shSp;  
 shootStrength = shStr;  
 health = hlth;  
 }  
  
 private void setUpAnimation(int numFrames, long d) {  
 Bitmap[] spriteFrames = new Bitmap[numFrames];  
  
 for(int i = 0; i < spriteFrames.length; i++) {  
 spriteFrames[i] = Bitmap.*createBitmap*(sprite, width \* i, 0, this.width, this.height);  
 }  
 animation.setFrames(spriteFrames);  
 animation.setDelay(d);  
 }  
  
 public void update() {  
 animation.update();  
  
 //стрельба  
 long fireTime = (System.*nanoTime*() - startShooting)/1000000;  
 if(fireTime > shootDelay) {  
 bullets.add(new Bullet(1, this.x, this.y + height/2, shootSpeed, shootStrength, 2, Color.*BLACK*, Configuration.ObjectType.*Player*));  
 startShooting = System.*nanoTime*();  
 }  
 for(int i = 0; i < bullets.size(); i++) {  
 b = bullets.get(i);  
 b.update();  
 if(b.getX() < -100) {  
 bullets.remove(i);  
 }  
 if(!GamePanel.*player*.getDisappear() && b.checkHit(GamePanel.*player*)) {  
 GamePanel.*player*.getDamage(b.getDamage());  
 bullets.remove(i);  
 if(GamePanel.*player*.getHealth() <= 0) {  
 GamePanel.*explosions*.add(GamePanel.*createExplosion*(GamePanel.*player*.getX(), GamePanel.*player*.getY(), 0));  
 GamePanel.*explosions*.get(GamePanel.*explosions*.size() - 1).setObjectType(Configuration.ObjectType.*Player*);  
 GamePanel.*player*.setDisappear(true);  
 //GamePanel.player.setLostGame(true);  
 break;  
 }  
 }  
 }  
 //перемещение в вертикальной плоскости  
 //не позволяем упасть самолету (иначе, что за ас там сидит)  
 if(y > GamePanel.*HEIGHT* - height - 10) {  
 y = GamePanel.*HEIGHT* - height - 10;  
 if(dy < 0)  
 dy \*= -1;  
 }  
 //не позволяем ему улететь  
 if(y < height + 10) {  
 y = height + 10;  
 if(dy > 0)  
 dy \*= -1;  
 }  
 y -= dy;  
  
 //Установка допустимого минимума/максимума  
 if (dy > 5)  
 dy = 5;  
 if(dy < -5)  
 dy = 5;  
  
 //Перемещение в горизонтальной плоскости  
 x -= speed;  
 }  
  
 public void draw(Canvas canvas) {  
 //на всякий случай  
 try {  
 canvas.drawBitmap(animation.getImage(), x, y, null);  
 }  
 catch (Exception e) { }  
  
 try {  
 for(Bullet m : bullets) {  
 m.draw(canvas);  
 }  
 }  
 catch (Exception e) { }  
 }  
  
 public void getDamage(int dmg) {  
 health -= dmg;  
 }  
  
 public int getHealth() {  
 return health;  
 }  
  
 public int getSpeed() {  
 return speed;  
 }  
  
}

**Модуль Explosion.java**

package com.example.alex.fighters;  
  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
  
*/\*\*  
 \* Created by Alex on 14.09.2015.  
 \*/*public class Explosion {  
 private int x;  
 private int y;  
 private int width;  
 private int height;  
 private Animation animation;  
 private Bitmap sprite;  
 private int dx;  
 private Configuration.ObjectType objectType;  
  
 public Explosion(int x, int y, int w, int h, Bitmap s, int numFrames, int dx) {  
 this.x = x;  
 this.y = y;  
 width = w;  
 height = h;  
 sprite = s;  
 this.dx = dx;  
 setUpAnimation(numFrames);  
 }  
  
 private void setUpAnimation(int numFrames) {  
 animation = new Animation();  
 int row = 0;  
 Bitmap[] frames = new Bitmap[numFrames];  
 for(int i = 0; i < frames.length; i++) {  
 if(i % 5 == 0 && i > 0)  
 row++;  
 frames[i] = Bitmap.*createBitmap*(sprite, (i - (5 \* row)) \* width, row \* height, width, height);  
 }  
 animation.setFrames(frames);  
 animation.setDelay(100);  
 }  
  
 public void draw(Canvas canvas) {  
 if(!animation.playedOnce()) {  
 canvas.drawBitmap(animation.getImage(), x, y, null);  
 }  
 }  
 public void update() {  
 if(!animation.playedOnce())  
 animation.update();  
 x += dx;  
 }  
  
 public int getHeight() {  
 return height;  
 }  
  
 public boolean isFinished() {  
 if(animation.getFramesCount() - 1 > animation.getCurrentFrame())  
 return false;  
 else  
 return true;  
 }  
  
 public void setX(int dx, int dir) {  
 this.x += (dx \* dir);  
 }  
  
 public Bitmap getBitmap() {  
 return this.sprite;  
 }  
  
 public void setObjectType(Configuration.ObjectType type) { objectType = type; }  
  
 public Configuration.ObjectType getObjectType() { return objectType; }  
}

**Модуль GameActivity.java**

package com.example.alex.fighters;  
  
import android.app.Activity;  
import android.content.Context;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.Window;  
import android.view.WindowManager;  
  
public class GameActivity extends Activity implements SensorEventListener {  
 private SensorManager senSensorManager;  
 private Sensor senAccelerometer;  
 private long lastUpdate = 0;  
 public static float *aX*, *aY*, *aZ*;  
 private float turnThreshold = 1.5f;  
  
 public static boolean *PLAIN*; //прямой полет  
 public static boolean *UP*; //true, самолет двигается вверх, иначе вниз  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*); // set to full screen  
 setContentView(new GamePanel(this));  
 senSensorManager = (SensorManager) getSystemService(Context.*SENSOR\_SERVICE*);  
 senAccelerometer = senSensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*);  
 senSensorManager.registerListener(this, senAccelerometer , SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
  
 @Override  
 public void onSensorChanged(SensorEvent e) {  
 Sensor sensor = e.sensor;  
 if(sensor.getType() == Sensor.*TYPE\_ACCELEROMETER*) {  
 long curTime = System.*currentTimeMillis*();  
 //берем значения акселерометра каждые 10 миллисекунд  
 if((curTime - lastUpdate) > 10) {  
 lastUpdate = curTime;  
 //определяем летит ли самолет вверх или вниз  
 *aX* = e.values[0];  
 *aY* = e.values[1];  
 *aZ* = e.values[2];  
  
 if(*aY* < turnThreshold \* -1) {  
 *UP* = true;  
 *PLAIN* = false;  
 }  
 else if (*aY* > turnThreshold) {  
 *UP* = false;  
 *PLAIN* = false;  
 }  
 else  
 *PLAIN* = true;  
 }  
 }  
 }  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) { }  
  
 protected void onPause() {  
 super.onPause();  
 senSensorManager.unregisterListener(this);  
 }  
  
 protected void onResume() {  
 super.onResume();  
 senSensorManager.registerListener(this, senAccelerometer, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*menu\_game*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
 if (id == R.id.*action\_settings*) {  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 public static String getAccelerometerString() {  
 return "aX = " + *aX* + " aY = " + *aY* + " aZ = " + *aZ*;  
 }  
}

**Модуль GameObject.java**

package com.example.alex.fighters;  
  
import android.graphics.Rect;  
  
*/\*\*  
 \* Created by Alex on 13.09.2015.  
 \*/*public class GameObject {  
 protected int x; //координата x  
 protected int y; //координата y  
 protected int dy; //перемещение по Oy  
 protected int width; //ширина объекта(спрайта)  
 protected int height; //высота объекта (спрайта)  
 protected Animation animation;  
 protected Configuration.ObjectType type;  
  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 public void setY(int y) {  
 this.y = y;  
 }  
  
 public int getX() {  
 return this.x;  
 }  
  
 public int getY() {  
 return this.y;  
 }  
  
 public int getHeight() {  
 return this.height;  
 }  
  
 public int getWidth() {  
 return this.width;  
 }  
  
 public Rect getRect() {  
 return new Rect(this.x + 10, this.y + 5, this.x + this.width - 10, this.y + this.height - 5); //для более реалистичного столкновения  
 }  
}

**Модуль GamePanel.java**

package com.example.alex.fighters;  
  
import android.content.Context;  
import android.content.pm.FeatureInfo;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Rect;  
import android.graphics.Typeface;  
import android.view.MotionEvent;  
import android.view.SurfaceHolder;  
import android.view.SurfaceView;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
*/\*\*  
 \* Created by Alex on 12.09.2015.  
 \*/*public class GamePanel extends SurfaceView implements SurfaceHolder.Callback {  
 private int pressDown = 0;  
 private int pressUp = 0;  
  
 //Расчет  
 private GameThread gameThread;  
 private boolean newGameCreated = false;  
 private boolean justEntered = true;  
  
 //Отображение  
 private Background bg;  
 public static Player *player*;  
 //Противники  
 public static ArrayList<Enemy> *enemies*;  
 public static ArrayList<Explosion> *explosions*;  
 private long enemyStartTime;  
 private Random rand;  
 private Enemy e;  
 public static Explosion *explosion*; //копируем и взрываем где-нибудь  
  
 //Интерфейс  
 private int bestResult = 0;  
  
 //Общие параметры  
 public static int *WIDTH* = 856;  
 public static int *HEIGHT* = 480;  
  
 public GamePanel(Context context) {  
 super(context);  
 getHolder().addCallback(this); //для вызова событий  
 setFocusable(true);  
 }  
  
 @Override  
 public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) { }  
  
 //Метод, срабатывающий при закрытии activity/приложения  
 @Override  
 public void surfaceDestroyed(SurfaceHolder holder) {  
 boolean retry = true;  
 int counter = 0;  
 while(retry && counter < 1000) {  
 counter++;  
 try {  
 gameThread.setExecuting(false);  
 gameThread.join();  
 retry = false;  
 gameThread = null;  
 }  
 catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 //Метод инициализации игрового пространства  
 @Override  
 public void surfaceCreated(SurfaceHolder holder) {  
 bg = new Background(BitmapFactory.*decodeResource*(getResources(), R.drawable.bg));  
 player = new Player(BitmapFactory.decodeResource(getResources(), R.drawable.spitfire\_sprite), 90, 38, 3, 100);  
 enemies = new ArrayList<Enemy>();  
 explosions = new ArrayList<Explosion>();  
 enemyStartTime = System.nanoTime();  
 rand = new Random();  
 explosion = new Explosion(0, 0, 100, 100, BitmapFactory.decodeResource(getResources(), R.drawable.explosion), 25, 0);  
 //Запуск игрового потока  
 gameThread = new GameThread(getHolder(), this);  
 gameThread.setExecuting(true);  
 gameThread.start();  
 }  
  
 //Срабатывает при касании пальцем экрана  
 @Override  
 public boolean onTouchEvent(MotionEvent e) {  
 if(e.getAction() == MotionEvent.*ACTION\_DOWN*) {  
 pressDown++;  
 if(!*player*.getPlaying() && !newGameCreated) {  
 resetGame();  
 *player*.setPlaying(true);  
 }  
 if(!*player*.getTriggerState()) {  
 *player*.setTriggerState(true);  
 if (!*player*.getDisappear()) {  
 *player*.shot();  
 }  
 }  
 return true;  
 }  
 if(e.getAction() == MotionEvent.*ACTION\_UP*) {  
 pressUp++;  
 if(*player*.getPlaying()) {  
 *player*.setTriggerState(false);  
 }  
 return true;  
 }  
 return super.onTouchEvent(e);  
 }  
  
 //Обеспечивает отрисовку игрового пространства  
 @Override  
 public void draw(Canvas canvas) {  
 super.draw(canvas);  
 final float scaleCoefficientX = getWidth()/(*WIDTH* \* 1.f);  
 final float scaleCoefficientY = getHeight()/(*HEIGHT* \* 1.f);  
  
 if(canvas != null) {  
 final int savedState = canvas.save();  
 canvas.scale(scaleCoefficientX, scaleCoefficientY);  
 //задний фон  
 bg.draw(canvas);  
 //игрок  
 if(!player.getDisappear())  
 player.draw(canvas);  
 //противники  
 for(Enemy e : enemies) {  
 e.draw(canvas);  
 }  
  
 //взорванные противники  
 for(Explosion exp : explosions) {  
 exp.draw(canvas);  
 }  
  
 //drawDebug(canvas, gameThread); //отладочная информация  
 drawText(canvas);  
 if(player.getPlaying())  
 drawGameInterface(canvas);  
 canvas.restoreToCount(savedState);  
 }  
 }  
  
 //Обеспечивает расчет всего необходимого для одного frame игры  
 public void update() {  
 //Рассчитываем в случае запущенной игры  
 if(player.getPlaying()) {  
 bg.update();  
 if(!player.getDisappear())  
 player.update();  
  
 //создание новых самолетов противника. Пока каждые 1.5 секунд  
 long enemyElapsedTime = (System.nanoTime() - enemyStartTime)/1000000;  
 if(enemyElapsedTime > 1500) {  
 enemies.add(new Enemy(BitmapFactory.decodeResource(getResources(), R.drawable.bf\_sprite),  
 WIDTH - 10,  
 rand.nextInt(HEIGHT - (Configuration.BF\_HEIGHT + 10)) + (Configuration.BF\_HEIGHT + 10),  
 Configuration.BF\_WIDTH,  
 Configuration.BF\_HEIGHT,  
 Configuration.BF\_FRAMECOUNT,  
 1000,  
 35,  
 10,  
 50  
 ));  
 enemyStartTime = System.nanoTime();  
 }  
 //update для противников  
 for(int i = 0; i < enemies.size(); i++) {  
 e = enemies.get(i);  
 e.update();  
 //столкновение c игроком  
 if (!player.getDisappear() && checkCollision(e.getRect(), player.getRect())) {  
 GamePanel.explosions.add(GamePanel.createExplosion(e.getX(), e.getY(), 0));  
 GamePanel.explosions.add(GamePanel.createExplosion(GamePanel.player.getX(), GamePanel.player.getY(), 0));  
 GamePanel.explosions.get(GamePanel.explosions.size() - 1).setObjectType(Configuration.ObjectType.Player);  
 enemies.remove(e);  
 GamePanel.player.setDisappear(true);  
 GamePanel.player.setHealth(0);  
 break;  
 }  
 //за экраном, то удаляем  
 if (e.getX() <= -300) {  
 enemies.remove(e);  
 }  
 }  
  
 //update для взрывов  
 for(int i = 0; i < explosions.size(); i++) {  
 explosions.get(i).update();  
 if(explosions.get(i).getObjectType() == Configuration.ObjectType.Player) {  
 if(explosions.get(i).isFinished()) {  
 player.setPlaying(false);  
 explosions.remove(i);  
 break;  
 }  
 }  
 if(explosions.get(i).isFinished())  
 explosions.remove(i);  
 }  
 }  
 else {  
 enemies.clear();  
 explosions.clear();  
 player.resetDY();  
 player.clearBullets();  
 newGameCreated = false;  
 }  
 }  
  
 private boolean checkCollision(Rect rect1, Rect rect2) {  
 return rect1.intersect(rect2);  
 }  
  
 private void drawText(Canvas canvas) {  
 Paint paint = new Paint();  
 paint.setColor(Color.BLACK);  
 paint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));  
  
 //Стартовый экран  
 if(justEntered) {  
 paint.setTextSize(100);  
 canvas.drawText("The Dogfight", 90, HEIGHT / 2 - 50, paint);  
 paint.setTextSize(30);  
 canvas.drawText("Tap the screen to start a new game", 110, HEIGHT / 2 + 20, paint);  
 paint.setTextSize(20);  
 canvas.drawText("Use accelerometer to soar up and down with tap for shooting", 110, HEIGHT/2 + 60, paint);  
 }  
 //В случае проигрыша  
 if(!justEntered && !newGameCreated) {  
 paint.setTextSize(100);  
 canvas.drawText("You lost", 100, HEIGHT / 2 - 50, paint);  
 paint.setTextSize(30);  
 if(player.getScore() > player.getBestScore())  
 canvas.drawText("New best: " + player.getScore() + " points", 110, HEIGHT / 2 + 30, paint);  
 else {  
 canvas.drawText("Your score: " + player.getScore() + " points", 110, HEIGHT / 2 + 20, paint);  
 canvas.drawText("Best score: " + player.getBestScore() + " points", 110, HEIGHT / 2 + 60, paint);  
 }  
 paint.setTextSize(20);  
 canvas.drawText("Tap to launch a new game", 110, HEIGHT / 2 + 90, paint);  
 }  
 }  
  
 private void drawGameInterface(Canvas canvas) {  
 Paint paint = new Paint();  
 paint.setColor(Color.BLACK);  
 paint.setTextSize(20);  
 paint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));  
 canvas.drawText(("Health: " + player.getHealth()), 10, 30, paint);  
 canvas.drawText(("Score: " + player.getScore()), 10, 60, paint);  
 }  
  
 //Вывод отладочной информации  
 private final void drawDebug(Canvas canvas, GameThread th) {  
 Paint paint = new Paint();  
 paint.setColor(Color.BLACK);  
 paint.setTextSize(20);  
 if(th != null) {  
 //canvas.drawText("FPS: " + th.getFPS(), 100, 100, paint);  
 canvas.drawText(GameActivity.aX + " - " + GameActivity.aY + " - " + GameActivity.aZ, 100, 100, paint);  
 }  
 }  
 private void resetGame() {  
 justEntered = false; //никогда больше не ставится в true  
 if(bestResult < player.getScore())  
 bestResult = player.getScore();  
 player.resetPlayer();  
 newGameCreated = true;  
 }  
  
 public static Explosion createExplosion(int x, int y, int dx) {  
 return new Explosion(x, y, 100, 100, explosion.getBitmap(), 25, dx);  
 }  
}

**Модуль GameThread.java**

package com.example.alex.fighters;  
  
import android.graphics.Canvas;  
import android.view.SurfaceHolder;  
  
*/\*\*  
 \* Created by Alex on 12.09.2015.  
 \*/*public class GameThread extends Thread {  
 private int FPS = 30;  
 private double averageFPS;  
 private SurfaceHolder surfaceHolder;  
 private GamePanel gamePanel;  
 private boolean executing;  
 private int i = 0;  
  
 public static Canvas *canvas*; //делаем его доступным для GamePanel  
  
 public GameThread(SurfaceHolder sh, GamePanel gp) {  
 this.gamePanel = gp;  
 this.surfaceHolder = sh;  
 }  
  
 @Override  
 public void run() {  
 long startTime, timeMillis, waitTime,  
 totalTime = 0, target = 1000/FPS; //"шаг" игрового цикла  
 int frameCount = 0;  
  
 while(executing) {  
 startTime = System.*nanoTime*();  
 *canvas* = null;  
  
 //блокировать Canvas для его редактирования  
 //синхронизация производится для рисования только одним потоком  
 try {  
 *canvas* = this.surfaceHolder.lockCanvas();  
 synchronized (surfaceHolder) {  
 this.gamePanel.update();  
 this.gamePanel.draw(*canvas*);  
 }  
 }  
 catch (Exception e) { }  
 finally {  
 if(*canvas* != null) {  
 try {  
 surfaceHolder.unlockCanvasAndPost(*canvas*);  
 }  
 catch (Exception e) { }  
 }  
 }  
  
 timeMillis = (System.*nanoTime*() - startTime)/1000000; //определение количества времени для одного update canvas  
 waitTime = target - timeMillis; //сколько времени нужно подождать перед след. итерацией цикла  
  
 try {  
 this.*sleep*(waitTime);  
 }  
 catch(Exception e) { }  
  
 totalTime = System.*nanoTime*() - startTime;  
 frameCount++;  
  
 if(frameCount == FPS) {  
 averageFPS = 1000/((totalTime/frameCount)/1000000) + i;  
 i++;  
 frameCount = 0;  
 totalTime = 0;  
 }  
 }  
 }  
  
 public void setExecuting(boolean flag) {  
 this.executing = flag;  
 }  
  
 public double getFPS() {  
 return this.averageFPS;  
 }  
}

**Модуль Player.java**

package com.example.alex.fighters;  
  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
  
import java.util.ArrayList;  
  
*/\*\*  
 \* Created by Alex on 13.09.2015.  
 \*/*public class Player extends GameObject {  
 private Bitmap sprite;  
 private boolean playing;  
 private int score;  
 private int bestScore;  
 private long startTime;  
 private boolean lostGame = false;  
 private int health = 200;  
  
 private ArrayList<Bullet> bullets;  
 private int shotSpeed = 20;  
 private int shotStrength = 100;  
 private boolean triggerPressed = false;  
 private Bullet b;  
  
 //Взрыв  
 private boolean disappear;  
  
 //Общие параметры игрока  
 public static int *SPEED* = -4;  
  
 public Player(Bitmap s, int w, int h, int numFrames, long d) {  
 score = 0;  
 //GameObject inheritance  
 x = 100; //начальная позиция по Ox  
 y = GamePanel.*HEIGHT* / 2; //начальная позиция по Oy  
 width = w;  
 height = h;  
 animation = new Animation();  
 sprite = s;  
 type = Configuration.ObjectType.*Player*;  
 bullets = new ArrayList<Bullet>();  
 setUpAnimation(numFrames, d);  
 disappear = true;  
 bestScore = 0;  
 }  
  
 private void setUpAnimation(int numFrames, long d) {  
 Bitmap[] spriteFrames = new Bitmap[numFrames];  
  
 for(int i = 0; i < spriteFrames.length; i++) {  
 spriteFrames[i] = Bitmap.*createBitmap*(sprite, width \* i, 0, this.width, this.height);  
 }  
 animation.setFrames(spriteFrames);  
 animation.setDelay(d);  
 startTime = System.*nanoTime*();  
 }  
  
 public void update() {  
 long elapsed = (System.*nanoTime*() - startTime) / 1000000; //конвертируем в миллисекунды  
 if(elapsed > 100) {  
 startTime = System.*nanoTime*();  
 score += 1;  
 }  
  
 animation.update();  
 //уперлись в небо  
 if(y <= 10)  
 y = 10;  
 //разбились --> ставим playing в false  
 if(y >= GamePanel.*HEIGHT* - (height - 10)) {  
 //lostGame = true;  
 //playing = false;  
 GamePanel.*explosions*.add(GamePanel.*createExplosion*(GamePanel.*player*.getX(), GamePanel.*player*.getY() - 40, 0));  
 GamePanel.*explosions*.get(GamePanel.*explosions*.size() - 1).setObjectType(Configuration.ObjectType.*Player*);  
 disappear = true;  
 health = 0;  
 }  
 //Вверх  
 if(GameActivity.*UP*)  
 dy -= 2;  
 //Вниз  
 else  
 dy += 2;  
  
 //Плавный полет  
 if(GameActivity.*PLAIN*)  
 dy = 1; //идеально плавного полета не бывает (постоянно клоним вниз)  
  
 y += dy;  
  
 //Установки max и min  
 if(dy > 5)  
 dy = 5;  
 if(dy < -5)  
 dy = -5;  
  
 //Стрельба  
 for(int i = 0; i < bullets.size(); i++) {  
 b = bullets.get(i);  
 b.update();  
 if(b.getX() > GamePanel.*WIDTH* + 100) {  
 bullets.remove(i);  
 }  
 //если есть хоть один противник начинаем проверять столкновение пули и самолета противника  
 if(GamePanel.*enemies*.size() != 0) {  
 for (Enemy e : GamePanel.*enemies*) {  
 if (b.checkHit(e)) {  
 e.getDamage(b.getDamage());  
 this.score += 5;  
 if (e.getHealth() <= 0) {  
 GamePanel.*explosions*.add(GamePanel.*createExplosion*(e.getX(), e.getY(), e.getSpeed() \* (-1)));  
 GamePanel.*enemies*.remove(e);  
 this.score += 20;  
 }  
 bullets.remove(i);  
 break;  
 }  
 }  
 }  
 }  
 }  
  
  
 public void draw(Canvas canvas) {  
 //Игрок  
 canvas.drawBitmap(animation.getImage(), x, y, null);  
 //Рисуем пули  
 for (Bullet b : bullets) {  
 b.draw(canvas);  
 }  
 }  
  
 public boolean getPlaying() {  
 return this.playing;  
 }  
  
 public void setPlaying(boolean f) {  
 playing = f;  
 }  
  
 public void addScrore(int s) {  
 score += s;  
 }  
  
 public int getScore() {  
 return this.score;  
 }  
  
 public void resetDY() {  
 dy = 0;  
 }  
  
 public void resetPlayer() {  
 if(score > bestScore)  
 bestScore = score;  
 score = 0;  
 dy = 0;  
 x = 100;  
 y = GamePanel.*HEIGHT*/2;  
 health = 200;  
 lostGame = false;  
 bullets.clear();  
 triggerPressed = false;  
 disappear = false;  
 }  
  
 public void getDamage(int dmg) {  
 if(health - dmg < 0)  
 health = 0;  
 else  
 health -= dmg;  
 }  
  
 public int getHealth() {  
 return health;  
 }  
  
 public void setHealth(int hlth) {  
 health = hlth;  
 }  
  
 public void shot() {  
 bullets.add(new Bullet(-1, x + width, y + height / 2, shotSpeed, shotStrength, 2, Color.*RED*, Configuration.ObjectType.*Enemy*));  
 }  
  
 public void setTriggerState(boolean f) {  
 triggerPressed = f;  
 }  
  
 public boolean getTriggerState() {  
 return triggerPressed;  
 }  
  
 public void clearBullets() {  
 bullets.clear();  
 }  
  
 public boolean getDisappear() { return disappear; }  
  
 public void setDisappear(boolean f) { disappear = f; }  
  
 public void setBestScore(int score) { bestScore = score; }  
  
 public int getBestScore() { return bestScore; }  
}

Приложение Б. Исходный код системы статистического, морфологического, кластерного и онтологического анализа текстов и данных для ЖКХ

**Модуль API.cs**

namespace Core

{

/// <summary>

/// API для подключения в приложение

/// </summary>

public class API

{

#region Работа с файлами (1-уровень)

/// <summary>

/// Загрузка одного файла

/// </summary>

/// <param name="path">Путь до файла</param>

/// <returns></returns>

public List<string> LoadFile(string path)

{

FileHandler fh = FileHelper.GetFileHandler(path);

if (fh == null)

return null;

List<string> lines;

fh.ReadFile(out lines);

return lines;

}

/// <summary>

/// Загрузка нескольких файлов (мультипроцессорная)

/// </summary>

/// <param name="paths">Пути до файлов</param>

public List<FileData> LoadFilesMulticore(List<string> paths)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorFileRead(paths);

return mps.FileCache;

}

#endregion

#region Mystem-api (2-уровень)

/// <summary>

/// Обработка строк файла программой mystem

/// </summary>

/// <param name="fileLines"></param>

/// <returns></returns>

public List<Lemm> HandleByMystem(List<string> fileLines)

{

MystemProvider mst = new MystemProvider(Guid.NewGuid().ToString());

return mst.LaunchMystem(fileLines);

}

/// <summary>

/// Обработка конкретного файла с помощью Mystem

/// </summary>

/// <param name="filePath">Путь до файла</param>

/// <returns></returns>

public MystemData HandleByMystem(string filePath)

{

var fileLines = LoadFile(filePath);

MystemProvider mst = new MystemProvider(Guid.NewGuid().ToString());

return new MystemData(filePath, mst.LaunchMystem(fileLines));

}

/// <summary>

/// Мультипроцуессорная обработка файлов программой mystem

/// </summary>

/// <param name="dataList"></param>

/// <returns></returns>

public List<MystemData> HandleByMystemMulticore(List<FileData> dataList)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorMystemHandler(dataList);

return mps.MystemCache;

}

/// <summary>

/// Обработка нескольких файлов с помощью Mystem

/// </summary>

/// <param name="filesPaths"></param>

/// <returns></returns>

public List<MystemData> HandleByMystemMulticore(List<string> filesPaths)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorFileRead(filesPaths);

mps.MultiprocessorMystemHandler(mps.FileCache);

return mps.MystemCache;

}

#endregion

#region Статистический анализ (3-уровень)

/// <summary>

/// Статистический анализ файла для отдельных слов

/// </summary>

/// <param name="list"></param>

/// <returns></returns>

public StatsAnalysisResult<string> ProvideWordsStatsAnalysis(List<Lemm> list)

{

List<string> words = list.GetWords();

StatsAnalysisResult<string> analysisResult = new StatsAnalysisResult<string>();

analysisResult.Frequency\_Dictionary = StatisticsAnalysis.GetFrequencyDictionary(words);

analysisResult.TF\_Dictionary = StatisticsAnalysis.GetTF(analysisResult.Frequency\_Dictionary, words.Count);

analysisResult.TF\_IDF\_Dictionary = StatisticsAnalysis.GetTF\_IDF(Configuration.CCSize, analysisResult.TF\_Dictionary);

return analysisResult;

}

/// <summary>

/// Статистический анализ для биграмм

/// </summary>

/// <param name="list"></param>

/// <returns></returns>

public StatsAnalysisResult<WordDigram> ProvideDigramsStatsAnalysis(MystemData data)

{

List<string> words = data.List.GetWords();

StatsAnalysisResult<WordDigram> analysisResult = new StatsAnalysisResult<WordDigram>();

var wordsFrequency = StatisticsAnalysis.GetFrequencyDictionary(words);

analysisResult.Name = data.Name;

analysisResult.Frequency\_Dictionary = StatisticsAnalysis.GetDigramFrequenceDictionary(words);

analysisResult.MutualInformation\_Dictionary = StatisticsAnalysis.CalculateMutualInformation(analysisResult.Frequency\_Dictionary, wordsFrequency, words.Count);

analysisResult.TScore\_Dictionary = StatisticsAnalysis.CalculateTScore(analysisResult.Frequency\_Dictionary, wordsFrequency, words.Count);

analysisResult.LogLikelihood\_Dictionary = StatisticsAnalysis.CalculateLogLikelihood(analysisResult.Frequency\_Dictionary);

return analysisResult;

}

/// <summary>

/// Мультипроцессорный статистический анализ для слов

/// </summary>

/// <param name="list"></param>

/// <returns></returns>

public List<StatsAnalysisResult<string>> ProvideWordsStatsAnalysisMulticore(List<MystemData> list)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorWordStatsAnalysis(list);

return mps.WordsStatsAnalysisCache;

}

/// <summary>

/// Мультипроцессорный статистический анализ для биграмм

/// </summary>

/// <param name="list"></param>

/// <returns></returns>

public List<StatsAnalysisResult<WordDigram>> ProvideDigramsStatsAnalysisMulticore(List<MystemData> list)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorDigramsStatsAnalysis(list);

return mps.DigramsStatsAnalysisCache;

}

#endregion

#region Морфологический анализ (4-уровень)

/// <summary>

/// Морфологический анализ (Исключение)

/// </summary>

/// <param name="list"></param>

/// <param name="excludedTypes"></param>

/// <returns></returns>

public MystemData ProvideMorphAnalysis(MystemData data, string[] excludedTypes)

{

return MorphologicalAnalysis.ExcludeWordsByType(data, excludedTypes);

}

/// <summary>

/// Мультипроцессорный анализ для Лемм

/// </summary>

/// <param name="list"></param>

/// <param name="excludedTypes"></param>

/// <returns></returns>

public List<MystemData> ProvideMorphAnalysisMulticore(List<MystemData> list, string[] excludedTypes)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorMorphAnalysis(list, excludedTypes);

return mps.MultiMorphAnalysisCache;

}

#endregion

#region Кластерный анализ (5-уровень)

public double[,] GetDefaultClustersCenters<T>(Dictionary<T, double[]> statisticDictionary)

{

var statisticsElements = statisticDictionary.Values;

var length = statisticsElements.Max(el => el.Length);

if (length != statisticsElements.Min(el => el.Length))

throw new Exception("Differ statistics array lengths!");

if (length < 1)

throw new Exception("Statistics is empty!");

var orderStaticstics = statisticsElements.OrderBy(el => el[0]);

for (var i = 1; i < length; i++)

{

var i1 = i;

orderStaticstics = orderStaticstics.ThenBy(el => el[i1]);

}

var first = orderStaticstics.First();

var last = orderStaticstics.Last();

var result = new double[2, length];

for(var i=0;i<length;i++)

{

result[0, i] = first[i];

result[1, i] = last[i];

}

return result;

}

/// <summary>

/// Производит кластерный анализ для одного текста

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="settings"></param>

/// <returns></returns>

public ClasterAnalysisResult<T> ProvideClusterAnalysis<T>(ClasterSettings<T> settings, string name)

{

ClasterAnalysis<T> ca = new ClasterAnalysis<T>(settings);

var result = ca.Clasterize();

return new ClasterAnalysisResult<T>(name, result);

}

/// <summary>

/// Многопроцессорная обработка для кластерного анализа слов

/// </summary>

/// <param name="data"></param>

/// <returns></returns>

public List<ClasterAnalysisResult<string>> ProvideWordClusterAnalysisMulticore(List<ClasterAnalysisData<string>> data)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorWordClasterAnalysis(data);

return mps.MultiWordsClusterAnalysisCache;

}

/// <summary>

/// Много процессорная обработка для кластерного анализа биграмм

/// </summary>

/// <param name="data"></param>

/// <returns></returns>

public List<ClasterAnalysisResult<WordDigram>> ProvideWordDigramClusterAnalysisMulticore(List<ClasterAnalysisData<WordDigram>> data)

{

Multiprocessor mps = new Multiprocessor();

mps.MultiprocessorWordDigramClasterAnalysis(data);

return mps.MultiWordDigramsClusterAnalysisCache;

}

/// <summary>

/// Методы сбора данных для кластерного анализа

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="analysisResult"></param>

/// <returns></returns>

public Dictionary<T, double[]> GetDataReady<T>(StatsAnalysisResult<T> analysisResult)

{

List<Dictionary<T, double>> data = new List<Dictionary<T, double>>();

if (analysisResult.Frequency\_Dictionary != null)

data.Add(analysisResult.Frequency\_Dictionary);

if (analysisResult.LogLikelihood\_Dictionary != null)

data.Add(analysisResult.LogLikelihood\_Dictionary);

if (analysisResult.MutualInformation\_Dictionary != null)

data.Add(analysisResult.MutualInformation\_Dictionary);

if (analysisResult.TF\_Dictionary != null)

data.Add(analysisResult.TF\_Dictionary);

if (analysisResult.TF\_IDF\_Dictionary != null)

data.Add(analysisResult.TF\_IDF\_Dictionary);

if (analysisResult.TScore\_Dictionary != null)

data.Add(analysisResult.TScore\_Dictionary);

return data.MergeDictionaries();

}

#endregion

}

}

**Модуль ClasterAnalysis.cs**

namespace Core

{

/// <summary>

/// Класс, реализующий FCM-алгоритм кластеризации

/// </summary>

/// <typeparam name="T">Тип ключа для словаря данных</typeparam>

public class ClasterAnalysis<T>

{

private int clustersCount;

private int dataVectorsCount;

private int dimensionsCount;

private int iterationCount;

private double[,] memberDegree;

private double epsilon;

private double fuzziness;

private double[,] data;

private double[,] clusterCenters;

private List<T> wordsDigrams;

#region Конструкторы

public ClasterAnalysis(ClasterSettings<T> settings)

{

//основные параметры

epsilon = settings.Epsilon;

fuzziness = settings.Fuzziness;

clusterCenters = settings.ClusterCenters;

iterationCount = settings.IterationCount;

//данные

memberDegree = new double[settings.Data.Count, clusterCenters.GetLength(0)];

data = new double[settings.Data.Count, settings.Data.First().Value.Length];

wordsDigrams = FillDataMatrix(settings.Data);

InitializeMatrix();

clustersCount = clusterCenters.GetLength(0);

dataVectorsCount = data.GetLength(0);

dimensionsCount = data.GetLength(1);

}

#endregion

/// <summary>

/// Сохранение данных

/// </summary>

/// <param name="wordStats"></param>

private List<T> FillDataMatrix(Dictionary<T, double[]> wordStats)

{

wordsDigrams = wordStats.Keys.ToList();

int n = 0;

foreach (var item in wordStats)

{

double[] stats = item.Value;

for (int j = 0; j < stats.Length; j++)

{

data[n, j] = stats[j];

}

n++;

}

return wordsDigrams;

}

/// <summary>

/// Инициализация матрицы принадлежности, с условием, что сумма всех значений не превышает 1.0

/// </summary>

private void InitializeMatrix()

{

double s;

int r = 100, rval;

Random rnd = new Random();

for (int i = 0; i < memberDegree.GetLength(0); i++)

{

s = 0.0f;

r = 100;

for (int j = 1; j < memberDegree.GetLength(1); j++)

{

rval = rnd.Next(1, r);

r -= rval;

memberDegree[i, j] = rval / 100.0f;

s += memberDegree[i, j];

}

memberDegree[i, 0] = 1.0 - s;

}

}

/// <summary>

/// Получение Евклидова расстояния

/// </summary>

/// <returns></returns>

private double CalculateEuclidLength(int i\_data, int j\_cl)

{

double sum = 0f;

for (int d = 0; d < dimensionsCount; d++)

{

sum += Math.Pow(data[i\_data, d] - clusterCenters[j\_cl, d], 2);

}

return Math.Sqrt(sum);

}

/// <summary>

/// Расчет нового значения матрицы принадлежности

/// </summary>

/// <param name="i"></param>

/// <param name="j"></param>

/// <returns></returns>

private double CalculateNewMemberDegreeItem(int i, int j)

{

double power = 2 / (fuzziness - 1), sum = 0.0f;

for (int k = 0; k < clustersCount; k++)

{

sum += Math.Pow(CalculateEuclidLength(i, j) / CalculateEuclidLength(i, k), power);

}

return 1.0f / sum;

}

/// <summary>

/// Перерасчет центров кластеров

/// SUM(k = 1, K)(M\_jk)^q \* x\_k

/// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/// SUM(k = 1, K)(M\_jk)^q

/// </summary>

private void CalculateClusterCenters()

{

double numerator, denominator;

double[,] M\_jk = new double[dataVectorsCount, clustersCount];

//расчет M\_jk

for (int i = 0; i < dataVectorsCount; i++)

{

for (int j = 0; j < clustersCount; j++)

{

M\_jk[i, j] = Math.Pow(memberDegree[i, j], fuzziness);

}

}

//перерасчет центров кластеров

for (int j = 0; j < clustersCount; j++)

{

for (int k = 0; k < dimensionsCount; k++)

{

numerator = 0f;

denominator = 0f;

for (int i = 0; i < dataVectorsCount; i++)

{

numerator += M\_jk[i, j] \* data[i, k];

denominator += M\_jk[i, j];

}

clusterCenters[j, k] = numerator / denominator;

}

}

}

/// <summary>

/// Перерасчет матрицы принадлежности

/// 1

/// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/// |x\_i - c\_j|

/// SUM(k = 1, C) (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) ^ (2/(q - 1))

/// |x\_i - x\_k|

/// </summary>

/// <returns></returns>

private double RecalculateMemberDegree()

{

double maxDiff = 0.0f, diff = 0.0f, mdValue;

for (int j = 0; j < clustersCount; j++)

{

for (int i = 0; i < dataVectorsCount; i++)

{

mdValue = CalculateNewMemberDegreeItem(i, j);

diff = Math.Abs(mdValue - memberDegree[i, j]);

//diff = mdValue - memberDegree[i, j];

if (diff > maxDiff)

maxDiff = diff;

memberDegree[i, j] = mdValue;

}

}

return maxDiff;

}

public Dictionary<T, double[]> Clasterize()

{

double maxDiff = 0.0f;

int i = 0;

do

{

CalculateClusterCenters();

maxDiff = RecalculateMemberDegree();

i++;

Console.WriteLine(maxDiff);

}

while (maxDiff > epsilon && i < iterationCount);

//формирование выходного словаря

var dict = new Dictionary<T, double[]>();

int n = 0;

foreach (var key in wordsDigrams)

{

double[] analysis = new double[clustersCount];

for (i = 0; i < clustersCount; i++)

{

analysis[i] = memberDegree[n, i];

}

dict.Add(key, analysis);

n++;

}

return dict;

}

}

}

**Модуль CommonClasses.cs**

namespace Core

{

#region Дополнительные классы для мультипроцессорной обработки

/// <summary>

/// Данные файла

/// </summary>

public class FileData

{

public List<string> List { get; private set; }

public string Name { get; private set; }

public FileData(string name, List<string> list)

{

Name = name;

List = list;

}

}

/// <summary>

/// Данные Mystem

/// </summary>

public class MystemData

{

public List<Lemm> List { get; set; }

public string Name { get; private set; }

public MystemData(string name, List<Lemm> list)

{

Name = name;

List = list;

}

}

#endregion

#region Лемма

[DataContract]

public class Lemm

{

[DataMember(Name = "text")]

public string text { get; set; } //слово в тексте

[DataMember(Name = "analysis")]

public Analysis[] analysis { get; set; } //анализ

public override string ToString()

{

return text;

}

}

[DataContract]

public class Analysis

{

[DataMember(Name = "lex")]

public string lex { get; set; }

[DataMember(Name = "gr")]

public string gr { get; set; }

public string wordType

{

get

{

return GetWordType(gr);

}

}

/// <summary>

/// Получение части речи

/// </summary>

/// <param name="gr"></param>

/// <returns></returns>

private string GetWordType(string gr)

{

return gr.Split(new[] { ',', '=' }, System.StringSplitOptions.RemoveEmptyEntries)[0];

}

}

#endregion

#region Биграмма

/// <summary>

/// Класс, описывающий биграмму

/// </summary>

public class WordDigram

{

private const char separator = ' ';

public string FirstWord { get; private set; }

public string SecondWord { get; private set; }

public WordDigram(string firstWord, string secondWord)

{

FirstWord = firstWord;

SecondWord = secondWord;

}

public override string ToString()

{

return string.Concat(FirstWord, separator, SecondWord);

}

public override bool Equals(object obj)

{

var dObj = obj as WordDigram;

if (dObj == null)

return base.Equals(obj);

return dObj.FirstWord == this.FirstWord && dObj.SecondWord == SecondWord;

}

public override int GetHashCode()

{

return FirstWord.GetHashCode() ^ SecondWord.GetHashCode();

}

}

#endregion

#region Анализы

/// <summary>

/// Хранение данных для статистического анализа

/// </summary>

/// <typeparam name="T">Биграмма, слово (WordDigram, string)</typeparam>

public class StatsAnalysisResult<T>

{

public string Name { get; set; }

public Dictionary<T, double> Frequency\_Dictionary { get; set; }

public Dictionary<T, double> TF\_Dictionary { get; set; }

public Dictionary<T, double> TF\_IDF\_Dictionary { get; set; }

public Dictionary<T, double> MutualInformation\_Dictionary { get; set; }

public Dictionary<T, double> TScore\_Dictionary { get; set; }

public Dictionary<T, double> LogLikelihood\_Dictionary { get; set; }

public StatsAnalysisResult(string name = "")

{

Name = name;

}

}

#endregion

#region Кластерный анализ

/// <summary>

/// Класс, содержащий настройки проводимого кластерного анализа (для UI)

/// </summary>

public class ClasterSettings<T>

{

public double[,] ClusterCenters { get; private set; }

public double Epsilon { get; private set; }

public double Fuzziness { get; private set; }

public int IterationCount { get; private set; }

public Dictionary<T, double[]> Data { get; private set; }

public ClasterSettings(double[,] cts, double e, double f, int ic, Dictionary<T, double[]> data)

{

ClusterCenters = cts;

Epsilon = e;

Fuzziness = f;

IterationCount = ic;

Data = data;

}

}

/// <summary>

/// Класс, описывающий набор параметров (Settings) для проведения кластерного анализа с указанием имени файла (Name)

/// </summary>

/// <typeparam name="T"></typeparam>

public class ClasterAnalysisData<T>

{

public string Name { get; private set; }

public ClasterSettings<T> Settings { get; private set; }

public ClasterAnalysisData(string name, ClasterSettings<T> settings)

{

Name = name;

Settings = settings;

}

}

/// <summary>

/// Класс, описывающий результат выполнения кластерного анализа

/// </summary>

/// <typeparam name="T"></typeparam>

public class ClasterAnalysisResult<T>

{

public string Name { get; private set; }

public Dictionary<T, double[]> Result { get; private set; }

public ClasterAnalysisResult(string name, Dictionary<T, double[]> result)

{

Name = name;

Result = result;

}

}

#endregion

}

**Модуль Configuration.cs**

namespace Core

{

public static class Configuration

{

public static int CCSize = 100000;

public static class WordType

{

//Типы частей речи

/// <summary>

/// Прилагательное

/// </summary>

public const string adjective = "A";

/// <summary>

/// Наречие

/// </summary>

public const string adverb = "ADV";

/// <summary>

/// Наречие-местоимение

/// </summary>

public const string adverb\_pronoun = "ADVPRO";

/// <summary>

/// Прилагательное-числительное

/// </summary>

public const string adjective\_numeral = "ANUM";

/// <summary>

/// Прилагательное-местоимение

/// </summary>

public const string adjective\_pronoun = "APRO";

/// <summary>

/// Сложное слово

/// </summary>

public const string composite = "COM";

/// <summary>

/// Союз

/// </summary>

public const string conjunction = "CONJ";

/// <summary>

/// Междометие

/// </summary>

public const string interjection = "INTJ";

/// <summary>

/// ЧИслительное

/// </summary>

public const string numeral = "NUM";

/// <summary>

/// Частица

/// </summary>

public const string particle = "PART";

/// <summary>

/// Предлог

/// </summary>

public const string preposition = "PR";

/// <summary>

/// Существительное

/// </summary>

public const string noun = "S";

/// <summary>

/// Существительное-местоимение

/// </summary>

public const string noun\_pronoun = "SPRO";

/// <summary>

/// Глагол

/// </summary>

public static string verb = "V";

}

}

}

**Модуль MorphologicalAnalysis.cs**

namespace Core

{

public static class MorphologicalAnalysis

{

/// <summary>

/// Получение лемм по типу речи

/// </summary>

/// <param name="lemms"></param>

/// <param name="type"></param>

/// <returns></returns>

public static List<Lemm> GetWordsByType(List<Lemm> lemms, string type)

{

return lemms.Where(i => i.analysis.Length > 0 && i.analysis[0].wordType == type).ToList();

}

/// <summary>

/// Исключение лемм по типу речи

/// </summary>

/// <param name="lemms"></param>

/// <param name="type"></param>

/// <returns></returns>

public static List<Lemm> ExcludeWordsByType(List<Lemm> lemms, params string[] types)

{

return lemms.Where(i => i.analysis.Length > 0 && !i.analysis.Any(j => types.Contains(j.wordType))).ToList();

}

/// <summary>

/// Исключение лемм по типу речи

/// </summary>

/// <param name="lemms"></param>

/// <param name="type"></param>

/// <returns></returns>

public static MystemData ExcludeWordsByType(MystemData mstData, params string[] types)

{

mstData.List = mstData.List.Where(i => i.analysis.Length > 0 && !i.analysis.Any(j => types.Contains(j.wordType))).ToList();

return mstData;

}

}

}

**Модуль Multiprocessor.cs**

namespace Core

{

public class Multiprocessor

{

private Thread[] threads; //потоки

private List<FileData> multiFileCache; //кэш с данными

private List<MystemData> multiMystemCache; //кэш с данными mystem

private List<StatsAnalysisResult<string>> multiWordsStatsAnalysisCache; //кэш для хранения результата по статистическому анализу для слов

private List<StatsAnalysisResult<WordDigram>> multiDigramsStatsAnalysisCache; //кэш для хранения результата по статистическому анализу для биграмм

private List<ClasterAnalysisResult<string>> multiWordsClusterAnalysisCache; //кэш для хранения результата кластерного анализа слов

private List<ClasterAnalysisResult<WordDigram>> multiDigramsClusterAnalysisCache; //кэш для хранения результата кластерного анализа биграмм

private List<MystemData> multiMorphAnalysisCache; //кэш для хранения результата морфологического анализа лемм

public List<FileData> FileCache

{

get { return multiFileCache; }

private set { multiFileCache = value; }

}

public List<MystemData> MystemCache

{

get { return multiMystemCache; }

private set { multiMystemCache = value; }

}

public List<StatsAnalysisResult<string>> WordsStatsAnalysisCache

{

get { return multiWordsStatsAnalysisCache; }

private set { multiWordsStatsAnalysisCache = value; }

}

public List<StatsAnalysisResult<WordDigram>> DigramsStatsAnalysisCache

{

get { return multiDigramsStatsAnalysisCache; }

private set { multiDigramsStatsAnalysisCache = value; }

}

public List<ClasterAnalysisResult<string>> MultiWordsClusterAnalysisCache

{

get { return multiWordsClusterAnalysisCache; }

private set { multiWordsClusterAnalysisCache = value; }

}

public List<ClasterAnalysisResult<WordDigram>> MultiWordDigramsClusterAnalysisCache

{

get { return multiDigramsClusterAnalysisCache; }

private set { multiDigramsClusterAnalysisCache = value; }

}

public List<MystemData> MultiMorphAnalysisCache

{

get { return multiMorphAnalysisCache; }

private set { multiMorphAnalysisCache = value; }

}

public Multiprocessor()

{

multiFileCache = new List<FileData>();

multiMystemCache = new List<MystemData>();

multiWordsStatsAnalysisCache = new List<StatsAnalysisResult<string>>();

multiDigramsStatsAnalysisCache = new List<StatsAnalysisResult<WordDigram>>();

multiWordsClusterAnalysisCache = new List<ClasterAnalysisResult<string>>();

multiDigramsClusterAnalysisCache = new List<ClasterAnalysisResult<WordDigram>>();

multiMorphAnalysisCache = new List<MystemData>();

}

private void \_cleanCache<T>(List<T> cache)

{

cache.Clear();

}

#region functions for parallel computing

private void \_readFile(string path)

{

FileHandler fh = FileHelper.GetFileHandler(path);

if (fh == null)

return;

List<string> lines;

fh.ReadFile(out lines);

multiFileCache.Add(new FileData(path, lines));

}

private void \_runMystem(FileData data, string index)

{

MystemProvider mp = new MystemProvider(index);

List<Lemm> list = mp.LaunchMystem(data.List);

multiMystemCache.Add(new MystemData(data.Name, list));

}

private void \_provideWordsStatsAnalysis(MystemData data)

{

List<string> words = data.List.GetWords();

StatsAnalysisResult<string> analysisResult = new StatsAnalysisResult<string>();

analysisResult.Frequency\_Dictionary = StatisticsAnalysis.GetFrequencyDictionary(words);

analysisResult.TF\_Dictionary = StatisticsAnalysis.GetTF(analysisResult.Frequency\_Dictionary, words.Count);

analysisResult.TF\_IDF\_Dictionary = StatisticsAnalysis.GetTF\_IDF(Configuration.CCSize, analysisResult.TF\_Dictionary);

multiWordsStatsAnalysisCache.Add(analysisResult);

}

private void \_provideDigramsStatsAnalysis(MystemData data)

{

List<string> words = data.List.GetWords();

StatsAnalysisResult<WordDigram> analysisResult = new StatsAnalysisResult<WordDigram>();

var wordsFrequency = StatisticsAnalysis.GetFrequencyDictionary(words);

analysisResult.Name = data.Name;

analysisResult.Frequency\_Dictionary = StatisticsAnalysis.GetDigramFrequenceDictionary(words);

analysisResult.MutualInformation\_Dictionary = StatisticsAnalysis.CalculateMutualInformation(analysisResult.Frequency\_Dictionary, wordsFrequency, words.Count);

analysisResult.TScore\_Dictionary = StatisticsAnalysis.CalculateTScore(analysisResult.Frequency\_Dictionary, wordsFrequency, words.Count);

analysisResult.LogLikelihood\_Dictionary = StatisticsAnalysis.CalculateLogLikelihood(analysisResult.Frequency\_Dictionary);

multiDigramsStatsAnalysisCache.Add(analysisResult);

}

private void \_provideWordClusterAnalysis(ClasterAnalysisData<string> data)

{

ClasterAnalysis<string> ca = new ClasterAnalysis<string>(data.Settings);

multiWordsClusterAnalysisCache.Add(new ClasterAnalysisResult<string>(data.Name, ca.Clasterize()));

}

private void \_provideDigramClusterAnalysis(ClasterAnalysisData<WordDigram> data)

{

ClasterAnalysis<WordDigram> ca = new ClasterAnalysis<WordDigram>(data.Settings);

multiDigramsClusterAnalysisCache.Add(new ClasterAnalysisResult<WordDigram>(data.Name, ca.Clasterize()));

}

private void \_provideMorphAnalysis(MystemData data, string[] excludedTypes)

{

var result = MorphologicalAnalysis.ExcludeWordsByType(data, excludedTypes);

multiMorphAnalysisCache.Add(result);

}

#endregion

public void MultiprocessorMorphAnalysis(List<MystemData> list, string[] excludedTypes)

{

\_cleanCache(multiMorphAnalysisCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

MystemData data = list[i];

threads[i] = new Thread(() => \_provideMorphAnalysis(data, excludedTypes));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorWordDigramClasterAnalysis(List<ClasterAnalysisData<WordDigram>> list)

{

\_cleanCache(multiWordsClusterAnalysisCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

ClasterAnalysisData<WordDigram> data = list[i];

threads[i] = new Thread(() => \_provideDigramClusterAnalysis(data));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorWordClasterAnalysis(List<ClasterAnalysisData<string>> list)

{

\_cleanCache(multiWordsClusterAnalysisCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

ClasterAnalysisData<string> data = list[i];

threads[i] = new Thread(() => \_provideWordClusterAnalysis(data));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorWordStatsAnalysis(List<MystemData> list)

{

\_cleanCache(multiWordsStatsAnalysisCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

MystemData data = list[i];

threads[i] = new Thread(() => \_provideWordsStatsAnalysis(data));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorDigramsStatsAnalysis(List<MystemData> list)

{

\_cleanCache(multiWordsStatsAnalysisCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

MystemData data = list[i];

threads[i] = new Thread(() => \_provideDigramsStatsAnalysis(data));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorMystemHandler(List<FileData> list)

{

\_cleanCache(multiMystemCache);

if (list != null && list.Count > 0)

{

threads = new Thread[list.Count];

for (int i = 0; i < threads.Length; i++)

{

FileData data = list[i];

threads[i] = new Thread(() => \_runMystem(data, Guid.NewGuid().ToString()));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

public void MultiprocessorFileRead(List<string> paths)

{

\_cleanCache(multiFileCache);

paths = FileHelper.CheckFiles(paths);

if (paths != null && paths.Count > 0)

{

threads = new Thread[paths.Count];

for (int i = 0; i < threads.Length; i++)

{

string path = paths[i];

threads[i] = new Thread(new ThreadStart(() => \_readFile(path)));

threads[i].Start();

}

foreach (Thread th in threads)

{

th.Join();

}

threads = null;

}

}

}

}

**Модуль MystemProvider.cs**

namespace Core

{

/// <summary>

/// Класс, обеспечивающий запуск mystem.exe

/// </summary>

public class MystemProvider

{

private string mystemPath;

private string index;

private string inputFileName;

private string outputFileName;

public MystemProvider(string index, string mystemPath = @"mystem\mystem.exe")

{

this.mystemPath = mystemPath;

this.index = index;

inputFileName = "tmp\_input\_" + index + ".txt";

outputFileName = "tmp\_output\_" + index + ".json";

}

/// <summary>

/// Парсинг json-файла результата выполнения mystem

/// </summary>

/// <param name="srdr"></param>

/// <returns></returns>

private List<Lemm> GetMystemResult(StreamReader srdr)

{

List<Lemm> lemms = new List<Lemm>();

DataContractJsonSerializer ser = new DataContractJsonSerializer(typeof(Lemm));

string line = srdr.ReadLine();

while (!srdr.EndOfStream)

{

Lemm obj = (Lemm)ser.ReadObject(new MemoryStream(Encoding.UTF8.GetBytes(line)));

if (obj.analysis != null)

lemms.Add(obj);

line = srdr.ReadLine();

}

srdr.Close();

return lemms;

}

/// <summary>

/// Запуск mystem.exe

/// </summary>

public List<Lemm> LaunchMystem(List<string> lines, string flags = "-cgin --format json")

{

Console.WriteLine(inputFileName);

FileHelper.WriteFile(lines, inputFileName);

Process process = new Process()

{

StartInfo = new ProcessStartInfo()

{

Arguments = String.Format("{0} {1} {2}", flags, inputFileName, outputFileName),

FileName = mystemPath,

UseShellExecute = false,

CreateNoWindow = true

}

};

process.Start();

process.WaitForExit();

List<Lemm> lemms = GetMystemResult(new StreamReader(outputFileName));

FileHelper.DeleteFile(inputFileName);

FileHelper.DeleteFile(outputFileName);

return lemms;

}

}

}

**Модуль StatisticsAnalysis.cs**

namespace CoreLib

{

/// <summary>

/// Статистический анализ текстов

/// </summary>

public static class StatisticsAnalysis

{

#region Однословия

/// <summary>

/// Получение частотного словаря (слова)

/// </summary>

/// <param name="words"></param>

/// <returns></returns>

public static Dictionary<string, double> GetFrequencyDictionary(List<string> words)

{

return

(

from i in words

group i by i into grp

select new { word = grp.Key, count = (double) grp.Count() }

)

.ToDictionary(i => i.word, i => i.count);

}

/// <summary>

/// Метод TF (Абсолютная частота встречаемости слова в тексте)

/// </summary>

/// <param name="words"></param>

/// <param name="count"></param>

/// <returns></returns>

public static Dictionary<string, double> GetTF(Dictionary<string, double> words, int count)

{

return words.ToDictionary(i => i.Key, i => ((double)i.Value / count));

}

/// <summary>

/// Метод TFxIDF (наиболее статистически значимые однословия)

/// </summary>

/// <param name="ccSize"></param>

/// <param name="tf\_dictionary"></param>

/// <param name="docNumber"></param>

/// <returns></returns>

public static Dictionary<string, double> GetTF\_IDF(int ccSize, Dictionary<string, double> tf\_dictionary, int docNumber = 1)

{

return tf\_dictionary.ToDictionary(i => i.Key, i => i.Value \* Math.Log((double)(ccSize - docNumber) / docNumber));

}

#endregion

#region Биграммы (Digrams)

/// <summary>

/// Получение частотного словаря биграмм

/// </summary>

/// <param name="wordList"></param>

/// <returns></returns>

public static Dictionary<WordDigram, double> GetDigramFrequenceDictionary(List<string> wordList)

{

var result = new Dictionary<WordDigram, double>();

List<string> digram;

for (var i = 0; i < wordList.Count - 1; i++)

{

digram = wordList.Skip(i).Take(2).ToList();

var digramKey = new WordDigram(digram[0], digram[1]);

if (!result.ContainsKey(digramKey))

result.Add(digramKey, GetNgramFrequence(wordList, digram));

}

return result;

}

/// <summary>

/// Метод Mutual Information (поиск наиболее статистически значимых двусловий)

/// </summary>

/// <param name="frequencyDiagram"></param>

/// <param name="frequencyDictionary"></param>

/// <param name="wordsCount"></param>

/// <returns></returns>

public static Dictionary<WordDigram, double> CalculateMutualInformation(Dictionary<WordDigram, double> frequencyDiagram,

Dictionary<string, double> frequencyDictionary, int wordsCount)

{

var result = new Dictionary<WordDigram, double>();

foreach (var pair in frequencyDiagram)

{

var key = pair.Key;

//частота первого слова

var fx = frequencyDictionary[key.FirstWord];

//частота второго слова

var fy = frequencyDictionary[key.SecondWord];

result.Add(key, Math.Log(

(pair.Value \* wordsCount) / fx \* fy

, 2));

}

return result;

}

/// <summary>

/// Метод TScore ()

/// </summary>

/// <param name="frequencyDiagram"></param>

/// <param name="frequencyDictionary"></param>

/// <param name="wordsCount"></param>

/// <returns></returns>

public static Dictionary<WordDigram, double> CalculateTScore(Dictionary<WordDigram, double> frequencyDiagram,

Dictionary<string, double> frequencyDictionary, int wordsCount)

{

var result = new Dictionary<WordDigram, double>();

foreach (var pair in frequencyDiagram)

{

var key = pair.Key;

//частота первого слова

var fx = frequencyDictionary[key.FirstWord];

//частота второго слова

var fy = frequencyDictionary[key.SecondWord];

result.Add(key,

((pair.Value - (fx \* fy / (double)wordsCount)) / (pair.Value \* pair.Value))

);

}

return result;

}

/// <summary>

/// Метод Log-Likelihood (выявление наиболее статистически значимых двусловий)

/// </summary>

/// <param name="frequencyDiagram"></param>

/// <returns></returns>

public static Dictionary<WordDigram, double> CalculateLogLikelihood(Dictionary<WordDigram, double> frequencyDiagram)

{

var result = new Dictionary<WordDigram, double>();

foreach (var pair in frequencyDiagram)

{

var a = pair.Value;

var b = frequencyDiagram

.Where(el => el.Key.FirstWord == pair.Key.FirstWord && el.Key.SecondWord != pair.Key.SecondWord)

.Sum(el => el.Value);

var c = frequencyDiagram

.Where(el => el.Key.FirstWord != pair.Key.FirstWord && el.Key.SecondWord == pair.Key.SecondWord)

.Sum(el => el.Value);

var d = frequencyDiagram

.Where(el => el.Key.FirstWord != pair.Key.FirstWord && el.Key.SecondWord != pair.Key.SecondWord)

.Sum(el => el.Value);

var value = a \* Math.Log(a + 1)

+ b \* Math.Log(b + 1)

+ c \* Math.Log(c + 1)

+ d \* Math.Log(d + 1)

- (a + b) \* Math.Log(a + b + 1)

- (a + c) \* Math.Log(a + c + 1)

- (b + d) \* Math.Log(b + d + 1)

- (c + d) \* Math.Log(c + d + 1)

+ (a + b + c + d) \* Math.Log(a + b + c + d + 1);

result.Add(pair.Key, value);

}

return result;

}

#endregion

#region N-граммы

/// <summary>

/// Частота N-граммы в предложении

/// </summary>

/// <param name="wordList"></param>

/// <param name="ngramList"></param>

/// <returns></returns>

public static int GetNgramFrequence(List<string> wordList, List<string> ngramList)

{

if (wordList == null || ngramList == null)

return 0;

if (wordList.Count < ngramList.Count || ngramList.Count <= 1)

return 0;

var result = 0;

//Возможна также реализация через IndexOf.

var startNgramWord = ngramList[0];

for (var i = 0; i < wordList.Count - ngramList.Count + 1; i++)

{

//если не первое слово энГраммы не совпадает с текущим в списке слов, катимся дальше

if (wordList[i] != startNgramWord)

continue;

var isEqual = true;

//начинаем цикл со 2 слова, т.к. первое проверили на предыдущем шаге

for (var j = 1; j < ngramList.Count; j++)

{

if (wordList[i + j] != ngramList[j])

{

isEqual = false;

continue;

}

}

if (isEqual) result++;

}

return result;

}

#endregion

}

/// <summary>

/// Extension-методы для статистического анализа

/// </summary>

public static class StatisticsAnalysisExtensions

{

/// <summary>

/// Возвращает словарь со всеми проведенными исследованиями

/// </summary>

/// <param name="dictionaries"></param>

/// <returns></returns>

public static Dictionary<T, double[]> MergeDictionaries<T>(this List<Dictionary<T, double>> dictionaries)

{

Dictionary<T, double[]> mergedDictionary = new Dictionary<T, double[]>();

foreach (var key in dictionaries[0].Keys)

{

List<double> metrics = new List<double>();

foreach (var dic in dictionaries)

{

double val;

if (dic.TryGetValue(key, out val))

metrics.Add(dic[key]);

else

metrics.Add(0.0f);

}

mergedDictionary.Add(key, metrics.ToArray());

}

return mergedDictionary;

}

/// <summary>

/// Общее количество слов

/// </summary>

/// <param name="words"></param>

/// <returns></returns>

public static int GetWordsCount(this Dictionary<string, int> words)

{

return words.Sum(i => i.Value);

}

/// <summary>

/// Получить список слов на основе List<Lemm>

/// </summary>

/// <param name="list">Данные mystem</param>

/// <returns></returns>

public static List<string> GetWords(this List<Lemm> list)

{

return list.Select(el =>

{

if (el.analysis.Length == 0)

return el.text;

return el.analysis[0].lex;

}).ToList();

}

}

}