|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | | | |
|  | |  |  |
| МИНОБРНАУКИ РОССИИ | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"Московский государственный университет информационных технологий, радиотехники и**  **электроники"**  **МИРЭА** | | | |
| Институт информационных технологий | | | |
| Кафедра информатики и информационных систем | | | |

|  |  |
| --- | --- |
| **КУРСОВАЯ РАБОТА** | |
| **по дисциплине** | |
| **«Объектно-ориентированное программирование»** | |
| **Тема курсовой работы: «Компьютерная модель конвертера физических и математических величин на базе открытого интерфейса социальной сети «ВКонтакте»».** | |
| Студент группы ИСБО-01-13 | *Коношенко Ю.А.* |
| Руководитель курсовой работы | *Хлебников А. А.* |
| Рецензент | *Хлебников А. А.* |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2015 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2015 г. |  |

Москва 2015

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  | | | | | |
| МИНОБРНАУКИ РОССИИ | | | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего профессионального образования  **"Московский государственный информационных технологий,**  **радиотехники и электроники"**  **МИРЭА** | | | | | |
| Институт информационных технологий (ИТ) | | | | | |
| Кафедра информатики и информационных систем (ИИС) | | | | | |
|  | | | **Утверждаю** | | |
|  | | |  | | |
|  | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2015 г. | | |
| **ЗАДАНИЕ** | | | | | |
| **на выполнение курсовой работы** | | | | | |
| **по** **дисциплине** «Объектно-ориентированное программирование» | | | | | |

Студент Коношенко Юлия Анатольевна Группа ИСБО-01-13

1. **Тема** «Компьютерная модель конвертера физических и математических величин на базе открытого интерфейса социальной сети «ВКонтакте»»
2. **Исходные данные:**

Документация API социальной сети «ВКонтакте»;

JDK 1.86;

Сторонние библиотеки: okhttp, gson, и другие.

* 1. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

Разработать модуль расширения на базе Java Extensible Applications;

Разработать окружение тестирования;

Разработать базовый класс для первичной обработки сообщений;

Разработать класс для конвертации физических и математических величин.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Срок представления к защите курсовой работы:** **до** «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г. | | | |
|  | | | |
| Задание на курсовую работу выдал | «\_\_\_»\_\_\_\_\_\_\_ 201\_\_г. |  | *Хлебников А. А.* |
| Задание на курсовую работу получил | «\_\_\_»\_\_\_\_\_\_\_ 201\_\_г. |  | *Коношенко Ю.А.* |

УДК 004.9

ББК 32.972

*Руководитель и рецензент:* Хлебников Андрей Александрович

*Студент:* Коношенко Юлия Анатольевна

*Тема:* **«**Компьютерная модель конвертера физических и математических величин на базе открытого интерфейса социальной сети «ВКонтакте»».

В ходе реализации задания на курсовую работу осуществляется исследование вопросов разработки программного обеспечения на языке программирования Java, изучение документации для работы с открытыми программными интерфейсами, написание программного кода.

Было создано соответствующее теме работы программное обеспечение и разработана документация работы, включающая в себя: утвержденное задание на курсовую работу; расчетно-пояснительную записку (РПЗ); рецензия (первичная) руководителя работы с итоговой отметкой о допуске работы к защите.

During the implementation of the task for course work done study on software development in the programming language Java, the study of the documentation for open APIs, writing software code.

It was established to work theme software and documentation developed by the work, which includes approved task for course work; settlement and explanatory note (RPGs); Reviews (primary) of the head with a final mark for admission to the work of the defense.

**Ключевые слова:** Java, JDK, ВКонтакте, конвертация.

Координаты автора:

МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

Кафедра Информатики и Информационных систем (ИИС)

**Реферат**

Данная курсовая работа содержит 24 страницы, 2 иллюстрации и 10 использованных источников.

Целью данной курсовой работы является разработка программного обеспечения на языке программирования Java.

Основными задачами явились: изучение документации Java, API «ВКонтакте», сторонних библиотек и протоколов, изучение среды разработки, создание программных модулей и классов, написание программного кода, реализующего необходимые задачи.

This term paper contains 23 pages, the 1 illustration and the 10 sources used.

The aim of this coursework is to develop software in the Java programming language.

The main tasks included: review of documentation Java API Vkontakte, third-party libraries and protocols, the study of development environment, software modules and classes, writing program code that implements the required task.

Оглавление

[**Введение** 6](#_Toc438738818)

[**Глава 1. Теоретическая часть** 7](#_Toc438738819)

[**1.1.** **Составление технического задания по реализации курсовой работы**……………………………....……………………………………………..7](#_Toc438738820)

[**1.2.** **Краткое описание международной системы единиц (СИ)** 7](#_Toc438738821)

[**1.3.** **Описание открытого программного интерфейса «ВКонтакте»** 10](#_Toc438738822)

[**1.4.** **Сетевая модель OSI** 11](#_Toc438738823)

[**1.5.** **Описание протокола HTTP** 12](#_Toc438738824)

[**1.6.** **Сетевой протокол TCP/IP** 16](#_Toc438738825)

[**Глава 2. Практическая часть** 18](#_Toc438738826)

[**2.1. Разработка архитектуры курсовой работы** 18](#_Toc438738827)

[**2.2. Краткое описание IntelliJ IDEA** 19](#_Toc438738828)

[**2.3. Сборка проекта и тестирование программы** 20](#_Toc438738829)

[**Заключение** 23](#_Toc438738830)

[**Список используемой литературы** 23](#_Toc438738831)

# **Введение**

Согласно теме курсовой работы, необходимо разработать компьютерную модель конвертера физических и математических величин на базе открытого интерфейса социальной сети «ВКонтакте»».

Курсовая работа выполнена на языке программирования Java (JDK 1.86), с применением ряда сторонних библиотек и модулей. В качестве среды разработке использована программа IntelliJ IDEA. Для контроля версий файлов применяется система контроля версий Git.

В результате выполнения курсовой работы ожидается получение готовой к работе программы, соответствующей теме работы.

При разработке программного обеспечения с подобным функционалом неизбежно обращение к сторонним сервисам. В данной работе используются сайт «ВКонтакте».

Обмен данными между локальным компьютером, на котором выполняется программа, и удаленным сервером, ведется при помощи API. API (интерфейс программирования приложений, интерфейс прикладного программирования) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована. Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

По такому принципу построены протоколы передачи данных по Интернет. Стандартный стек протоколов (сетевая модель OSI) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам HTTP и IMAP). Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню. Понятие протокола близко по смыслу к понятию API. И то, и другое является абстракцией функциональности, только в первом случае речь идёт о передаче данных, а во втором — о взаимодействии приложений.

Следует заметить, что в рамках данной курсовой работы используется веб-сервис. Его API реализуется при помощи набора HTTP-запросов, а также через определение структуры HTTP-ответов, для выражения которых используют XML или JSON форматы.

Для реализации курсовой работы используется Java Extensible Applications, т.е. модульная система построения приложений. Каждая часть программы отделена от других по функциональному признаку и осуществляет взаимодействие с другими при помощи интерфейсов.

**Глава 1. Теоретическая часть**

* 1. **Составление технического задания по реализации курсовой работы**

На начальном этапе работы, перед непосредственно написанием кода программы, необходимо определить архитектуру, необходимые библиотеки, взаимодействие между классами, модели взаимодействия между всеми компонентами программы друг с другом, а также с внешними сервисами.

Данная программа состоит из трех основных частей:

* Базовый модуль. Его работа заключается во взаимодействии с серверами социальной сети «ВКонтакте» с помощью API. Также этот модуль включает в себя осуществление таких функций, как авторизация с помощью логина и пароля, поддержка соединения, отправка запросов к внешним серверам и получение ответа, разбор этого ответа. Также включает в себя реализацию множества дополнительных функций.
* Модуль конвертации. Данный модуль осуществляет работу с сообщением, в частности заменяет встроенной функцией все падежи и виды написания конкретной величины на единый, создает и заполняет коллекцию, взаимодействует с конвертируемыми величинами.
* Модуль запуска, реализующий запуск на выполнение и загрузку других модули.

В результате созданный цикл, имея в наличии такие данные, как начальная величина, конвертирует эту величину. Вся полученная информация формируется в сообщение и отправляет пользователю.

* 1. **Краткое описание международной системы единиц (СИ)**

В то время как развивалась и широко распространялась техника в различных странах, появилась необходимость введения и использования легко просто воссоздаваемых единиц измерения, которые по возможности могли служить как можно дольше. Такая система измерения долго разрабатывалась и была реализована с применением тех природных взаимосвязей, с помощью которых стремились осуществить взаимосвязь единиц измерения.

Наиболее удобными были те единицы измерения, при которых беспрепятственно и самостоятельно друг от друга устанавливались единицы измерения для довольно небольшого количества единиц измерения, а другие величины устанавливались на основе уже известных закономерностей, существующих между этими единицами измерения.

Международная система единиц (СИ) — система единиц, основанная на Международной системе величин, вместе с наименованиями и обозначениями, а также набором приставок и их наименованиями и обозначениями вместе с правилами их применения, принятая Генеральной конференцией по мерам и весам (CGPM).

Развитием единой метрической системы мер является Международный эталон метра, использовавшийся СИ с 1889 по 1960 год. СИ была создана французскими учёными и впервые широко распространена после Великой французской революции. Сначала единицы измерения выбирались независимо друг от друга, что вызывало сложности с переводом из одной единицы в другую. Более того, в разных местах применялись совершенно разные единицы измерения, иногда даже с одинаковыми названиями. С появлением СИ эти неудобства должны были исчезнуть, а манипуляции с единицами измерения стать удобными.

В 1799 году во Франции были созданы два эталона — для единицы длины - метра и для единицы массы - килограмма.

В 1874 году была разработана система СГС на основе трех единиц: сантиметр, грамм и секунда и десятичные приставки начиная с микро заканчивая мега.

В 1875 году была подписана Метрическая конвенция представителями семнадцати государств, включая Россию, США, Германию, Италию и Францию, согласно которой были созданы Международный комитет мер и весов и Международное бюро мер и весов, а также учтен постоянный созыв Генеральных конференций по мерам и весам. Также началась разработка международных эталонов метра и килограмма.

В 1889 году была принята система единиц МКС, в основе которой лежат метр, килограмм и секунда. Позднее были установлены базовые единицы для физических величин в сфере оптики и электричества.

В соответствии с обращением Международного союза чистой и прикладной физики и французского правительства с предложениями о международной стандартизации единиц в 1948 году ГКМВ возложила на Международному комитету мер и весов обязанность разработать основу для создания единой системы единиц измерения для всеобщего пользования государствами участниками Метрической конвенции. В результате решения в 1954 году основными единицами разрабатываемой системы стали считать метр, килограмм, секунда, ампер, градус Кельвина, кандела.

В 1960 году метр, килограмм, секунда, ампер, градус Кельвина и кандела окончательно были принят как стандарт, который получил название «Международная система единиц», и установила международное сокращённое наименование этой системы «SI».

С 1963 года ГОСТом 9867-61 «Международная система единиц» СИ была установлена в СССР в качестве предпочтительной во всех областях науки и техники, народного хозяйства, при преподавании.

В последствии было принято новое определение единицы термодинамической температуры и дано ей имя «кельвин» и обозначение «К» (до этого единица называлась «градус Кельвина» и обозначалась «°K»).

В 1967—1968 годах было сформировано новое определение секунды.

В 1971 году ГКМВ внесла изменения в СИ, добавив в перечень базовых единиц единицу количества вещества - моль.

В 1979 году ГКМВ приняла новое определение канделы.

В 1983 году ГКМВ дала новое определение метра.

В 2009 году Правительство Российской Федерации утвердило Положение о единицах величин, допускаемых к использованию в Российской Федерации, а в 2015 году внесло в него правки, касающиеся срока действия некоторых внесистемных единиц.

На сегодняшний день система измерений является основной системой измерения единиц, принятой большей частью стран мира и почти всегда применяемой в области техники, даже в тех странах, которые в обычной жизни используются традиционные единицы (например, США).

Система измерений (СИ) включает в себя семь главных единиц физических величин, а также производные единицы и соответствующие приставки. По мимо этого СИ характеризует принятые сокращённые обозначения единиц и правила записи производных единиц. Основными единицами СИ являются килограмм, метр, секунда, ампер, кельвин, моль и кандела. Также считается, что эти единицы имеют самостоятельную размерность, другими словами ни одна из основных единиц не может быть получена из других.

Производные единицы получаются из ключевых с помощью математических действий, например, умножение и деление. Некоторым из производных единиц в СИ присвоены собственные наименования, например, единице радиан.

Приставки, как правило, используются перед наименованиями единиц. Они существуют для того, чтобы показать, что единицу нужно умножить или разделить на определённое целое число, степень числа 10. Например, приставка «кило» означает умножение на 1000 (километр = 1000 метров). Приставки СИ называют также десятичными приставками.

* 1. **Описание открытого программного интерфейса «ВКонтакте»**

API (application programming interface, или интерфейс программирования приложений, интерфейс прикладного программирования) — это перечень готовых классов, процедур, функций, структур и констант, предлагаемых приложением пользователю библиотекой или сервисом, для использования во внешних программных продуктах при написании всевозможных приложений.

Для разработчика API сайта “ВКонтакте” является отличным инструментом, который дает возможность использовать огромное количество функций этой социальной сети при разработке своих программ. В основном использованы могут быть функции подключения к странице пользователя, получения информации о пользователе, получения доступа к списку друзей пользователя, взаимодействия с группами, получения информации о группах, взаимодействия с изображениями, видео, музыкой пользователя, передачи сообщений пользователям и группам, работы со стеной и отправления записей на стену.

На начальном этапе работы с API необходимо зарегистрировать приложение на соответствующей странице сайта «ВКонтакте». В результате чего приложению будет присвоен его числовой код, с помощью которого будет осуществляться вся дальнейшая работа с API.

Любое Standalone-приложение требует авторизации пользователя, чтобы получить доступ к API ВКонтакте. Клиентом может быть любое компьютерное или мобильное приложение, имеющее возможность управлять веб-браузером. В рамках данной курсовой работы осуществляется исполнение запросов веб-браузера посредством JDK библиотек.  
 Процесс идентификации приложения состоит из трех шагов:

* открытие окна браузера для идентификации пользователя на сайте ВКонтакте.
* осуществление пользователем доступа к своим данным.
* передача в приложение ключа access\_token для доступа к API.

При клиентской авторизации ключ приложению выдается ключ доступа к API. После прохождения пользователем процедуры авторизации дальнейшая работа с API осуществляется с помощью специального ключа. Затем браузер пользователя будет перенаправлен по адресу REDIRECT\_URI, указанному при открытии диалога авторизации. При этом ключ доступа к API и другие параметры будут передана в URL-фрагменте ссылки.

Вместе с ключом также будет указано время его жизни.

Если срок использования ключа истек, то необходимо повторно пройти все перечисленные шаги, но на этот раз пользователь не будет разрешать доступ повторно. При смене пользователем логина или пароля, или удалении приложения в настройках доступа также необходимо требовать ключ.

Кроме того, среди возвращаемых параметров будет указан идентификатор пользователя, авторизовавшегося в социальной сети.

* 1. **Сетевая модель OSI**

Сетевая модель OSI (Open Systems Interconnection Basic Reference Model) — это теоритическая сетевая модель для взаимодействия и проектирования сетевых протоколов. Представляет уровневый подход к сети. Каждый уровень отвечает за свою часть процесса взаимодействия. Структура такого вида делает совместную работу сетевого оборудования и программного обеспечения гораздо проще и удобнее.

Уровни работы сетевой модели:

* Физический уровень. Он обеспечивает передачу необработанных битов по каналу связи. При разработке сети необходимо быть уверенным, что при передачи единицы одной стороны, другая сторона принимает такую же единицу.
* Уровень передачи данных. Уровень сетевой модели OSI, предназначенный для передачи данных узлам находящимся в том же сегменте локальной сети. Основная задача уровня передачи данных — быть способным передавать «сырые» данные физического уровня по надёжной линии связи, свободной от не обнаруженных ошибок с точки зрения вышестоящего сетевого уровня.
* Сетевой уровень. Сетевой уровень занимается управлением операциями в подсети. Важнейшим моментом здесь является определение маршрутов пересылки пакетов от источника к пункту назначения. Маршруты могут быть жестко заданы в виде таблиц и редко меняться. Кроме того, они могут задаваться в начале каждого соединения, на пример терминальной сессии. Наконец, они могут быть в высокой степени динамическими, то есть вычисляемыми заново для каждого пакета с учётом текущей загруженности сети.
* Транспортный уровень. Основная задача транспортного уровня — принять данные от сеансового уровня, разбить их на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению. Кроме того, всё это должно быть сделано эффективно и таким образом, чтобы изолировать более высокие уровни от каких-либо изменений в аппаратной технологии. Транспортный уровень также определяет тип сервиса, предоставляемого сеансовому уровню и, в конечном счёте, пользователям сети.
* Сеансовый уровень. Сеансовый уровень позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом. Кроме обычной доставки данных, которую обеспечивает также и транспортный уровень, сеанс может предоставлять еще и расширенный набор услуг, полезный для некоторых приложений. Сеанс может применяться для регистрации пользователя в системе разделения времени или для переноса файла с одного компьютера на другой.
* Уровень представления. Отличие от более низких уровней, задача которых - достоверная передача бит и байтов, уровень представления занимается по большей части синтаксисом и семантикой передаваемой информации. Типичный пример служб уровня представления - кодирование передаваемых данных определенным стандартным образом.
* Прикладной уровень. Прикладной уровень содержит набор популярных протоколов.



Рис.1 Сетевая модель OSI

* 1. **Описание протокола HTTP**

HTTP (Hypertext Transfer Protokol) - протокол прикладного уровня, предназначен для распределения и управления информационными системами, реализующими механизм гипертекстовых ссылок. Он является основным объектно-ориентированным протоколом, который может решать задачи управления обменом между серверами и объектами распределенных систем, используя их методы запросов. Основным направлением развития HTTP является определение типа и способов представления данных; применение систем, независимых от способа передачи данных.

HTTP используется Word-Wide Web начиная с 1990 года. Первая версия НТТР - НТТР/0.9 - являлась простым протоколом передачи данных через Internet. В версии НТТР/1.0 добавлена возможность передачи сообщений в формате MIME, содержащем различную информацию о переданных данных и изменениях в семантике запрос/ответ. Однако, НТТР/1.0 полностью не удовлетворял требованиям открытой системы, надежного соединения и другим инструкциям, которые обеспечивают защиту вызванных приложений.

Рассматриваемая версия НТТР - НТТР/1.1 полностью совместима с НТТР/1.0, но содержит более строгие требования для обеспечения совместимости с различными приложениями. Данный протокол позволяет расширять набор методов, которые определяют цель запросов. НТТР/1.1 разработан в соответствии с требованиями поддержки универсальных указателей идентификатора URI (Universal Resurse Identifier), ресурсов URL (Universal Resurse Location) и имени URN (Universal Resurse Name), для определения ресурса, к которому обратилось приложение. Сообщения передаются службами электронной почты (Internet Mail) и службами стандарта MIME (Multipurpose Internet Mail Extensions), разработанного с целью пересылки по электронной почте любых типов данных.

НТТР также используется как основной протокол для соединения агента пользователя и межсетевого шлюза с такими протоколами Интернета как SMTP, NNTP, FTP, Gopher и WAIS, как протокол, позволяющий организовать гипер-доступ к ресурсам, доступным из различных приложений и облегчающий применение агентов пользователей.

Протокол НТТР базируется на основе парадигмы запрос/ответ. Клиент посылает запросы серверу с указанием метода запроса, URI, версии протокола; сообщения передаются в соответствии со спецификацией MIME, содержащей информацию пользователя и поля, необходимые для установления соединения с сервером. Сервер, получив запрос, передает номер порта соединения, версию протокола соединения, сообщение об успешном соединении или ошибке при установлении сеанса, данные в соответствии со спецификацией MIME и служебные поля.

Большинство соединений НТТР инициализируется агентом пользователя и состоит из запроса на доступ к ресурсам необходимого сервера. Более сложные ситуации возникают, когда в цепочке запрос/ответ присутствует процесс-посредник. Выделяют три формы процессов-посредников: заместитель, шлюз и туннель. Процесс-заместитель - это передающий агент, который принимает запрос для URI, перезаписывает все части сообщения и передает преобразованный запрос серверу, определяемому по параметрам URI. Шлюз (межсетевой) - это принимающий агент, выступающий в роли отдельного сетевого уровня, который, если необходимо, может передать запрос вышестоящим службам протоколов. Туннель - это коммутатор между двумя точками соединения, который не изменяет семантику сеанса; туннель используется, когда необходимо передать информацию через посредника в том случае, когда посредник не может определить тип передаваемой информации. Сообщения, направляемые в соответствии с цепочками запрос/ответ, должны пройти через четыре различных соединения.

В Интернете сеансы НТТР базируются на соединениях TCP/IP. По умолчанию используется порт 80, но возможно использование и других портов. Это не исключает возможности использовать НТТР как протокол прикладного уровня для других протоколов Интернета или протоколов других сетей. НТТР предполагает наличие транспортного протокола; любой протокол, который удовлетворяет требованиям транспортного уровня, может использоваться для организации сеансов НТТР. Однако, функционирование НТТР/1.1 предполагает обеспечение устойчивого соединения. Как пользователь, так и сервер должны быть способны корректно обработать ситуации преждевременного завершения сеанса, истечения времени тайм-аута или ошибки программы. В любом случае прекращение сеанса одним или обоими абонентами всегда сопровождается уничтожением запросов, независимо от их статуса.

HTTP-запрос состоит из заголовка запроса и тела запроса, разделенных пустой строкой. Тело запроса может отсутствовать.

Заголовок запроса состоит из главной (первой) строки запроса и последующих строк, уточняющих запрос в главной строке. Последующие строки также могут отсутствовать.

Запрос в главной строке состоит из трех частей, разделенных пробелами:

Методы:

* GET - запрос документа. Наиболее часто употребляемый метод; в HTTP/0.9, говорят, он был единственным.
* HEAD - запрос заголовка документа. Отличается от GET тем, что выдается только заголовок запроса с информацией о документе. Сам документ не выдается.
* POST - этот метод применяется для передачи данных CGI-скриптам. Сами данные следуют в последующих строках запроса в виде параметров.
* PUT - разместить документ на сервере. Насколько я знаю, используется редко. Запрос с этим методом имеет тело, в котором передается сам документ.

Ресурс - это путь к определенному файлу на сервере, который клиент хочет получить (или разместить - для метода PUT). Если ресурс - просто какой-либо файл для считывания, сервер должен по этому запросу выдать его в теле ответа. Если же это путь к какому-либо CGI-скрипту, то сервер запускает скрипт и возвращает результат его выполнения. Кстати, благодаря такой унификации ресурсов для клиента практически безразлично, что он представляет собой на сервере.

Версия протокола-версия протокола HTTP, с которой работает клиентская программа.

Таким образом, простейший HTTP-запрос может выглядеть следующим образом: GET / HTTP/1.0.Здесь запрашивается корневой файл из корневой директории web-сервера.

Таким образом задаются параметры запроса. Это является необязательным, все строки после главной строки запроса могут отсутствовать; в этом случае сервер принимает их значение по умолчанию или по результатам предыдущего запроса (при работе в режиме Keep-Alive).

Некоторые наиболее употребительные параметры HTTP-запроса:

* Connection (соединение)- может принимать значения Keep-Alive и close. Keep-Alive ("оставить в живых") означает, что после выдачи данного документа соединение с сервером не разрывается, и можно выдавать еще запросы. Большинство браузеров работают именно в режиме Keep-Alive, так как он позволяет за одно соединение с сервером "скачать" html-страницу и рисунки к ней. Будучи однажды установленным, режим Keep-Alive сохраняется до первой ошибки или до явного указания в очередном запросе Connection: close.
* close ("закрыть") - соединение закрывается после ответа на данный запрос.
* User-Agent - значением является "кодовое обозначение" браузера, например:
* Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)
* Accept - список поддерживаемых браузером типов содержимого в порядке их предпочтения данным браузером, например для моего IE5:
* Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, \*/\*

Это, очевидно, нужно для случая, когда сервер может выдавать один и тот же документ в разных форматах. Значение этого параметра используется в основном CGI-скриптами для формирования ответа, адаптированного для данного браузера.

* Referer - URL, с которого перешли на этот ресурс.
* Host - имя хоста, с которого запрашивается ресурс. Полезно, если на сервере имеется несколько виртуальных серверов под одним IP-адресом. В этом случае имя виртуального сервера определяется по этому полю.
* Accept-Language - поддерживаемый язык. Имеет значение для сервера, который может выдавать один и тот же документ в разных языковых версиях.

Формат ответа очень похож на формат запроса: он также имеет заголовок и тело, разделенное пустой строкой. Заголовок также состоит из основной строки и строк параметров, но формат основной строки отличается от таковой в заголовке запроса. Основная строка запроса состоит из 3-х полей, разделенных пробелами. Версия протокола - аналогичен соответствующему параметру запроса. Код ошибки - кодовое обозначение "успешности" выполнения запроса. Код 200 означает "все нормально" (OK). Словесное описание ошибки - "расшифровка" предыдущего кода. Например для 200 это OK, для 500 - Internal Server Error. Наиболее употребительные параметры http-ответа:

* Connection - аналогичен соответствующему параметру запроса.

Если сервер не поддерживает Keep-Alive (есть и такие), то значение Connection в ответе всегда close.

* Content-Type ("тип содержимого") - содержит обозначение типа содержимого ответа. В зависимости от значения Content-Type браузер воспринимает ответ как HTML-страницу, картинку gif или jpeg, как файл, который надо сохранить на диске, или как что-либо еще и предпринимает соответствующие действия. Значение Content-Type для браузера аналогично значению расширения файла для Windows. Некоторые типы содержимого:
* text/html - текст в формате HTML (веб-страница);
* text/plain - простой текст (аналогичен "блокнотовскому");
* image/jpeg - картинка в формате JPEG;
* image/gif - то же, в формате GIF;
* application/octet-stream - поток "октетов" (т.е. просто байт) для записи на диск.

# **Сетевой протокол TCP/IP**

Взаимодействие между компьютерами в интернете осуществляется посредством сетевых протоколов, представляющих собой согласованный набор определенных правил, в соответствии с которыми разные устройства передачи данных обмениваются информацией. Существуют протоколы для форматов данных, скорости передачи, для контроля ошибок и другие виды протоколов. В глобальном межсетевом взаимодействии чаще всего используется протокол TCP-IP. Что же это за технология? Название протокола интернета TCP-IP произошло от двух сетевых протоколов: TCP и IP. Конечно, этими двумя протоколами построение сетей не ограничивается, но они являются базовыми в том, что касается именно организации передачи данных. Фактически, TCP-IP есть набор протоколов, позволяющих индивидуальным сетям объединяться для образования глобальной сети.

Протокол TCP-IP, описание которого невозможно обозначить только определениями IP и TCP, включает в себя также протоколы UDP, SMTP, ICMP, FTP, telnet, и не только. Эти и другие протоколы TCP-IP обеспечивают наиболее полноценную работу сети Интернет. Ниже приведем развернутую характеристику каждому протоколу, входящему в общее понятие TCP-IP. • Интернет-протокол (IP) отвечает за непосредственную передачу информации в сети. Информация делится на части (другими словами, пакеты) и передается получателю от отправителя. Для точной адресации нужно задать точный адрес или координаты получателя. Такие адреса состоят из четырех байт, которые отделены друг от друга точками. Адрес каждого компьютера уникален.

Однако использования одного лишь IP-протокола может быть недостаточно для корректной передачи данных, так как объем большей части пересылаемой информации более 1500 символов, что уже не вписывается в один пакет, а некоторые пакеты могут быть потеряны в процессе передачи или присланы не в том порядке, что требуется. • Протокол управления передачей (TCP) используется на более высоком уровне, чем предыдущий. Основываясь на способности IP-протокола переносить информацию от одного узла другому, TCP-протокол позволяет пересылать большие объемы информации. TCP отвечает также за разделение передаваемой информации на отдельные части - пакеты - и правильное восстановление данных из пакетов, полученных после передачи. При этом данный протокол автоматически повторяет передачу пакетов, которые содержат ошибки. Управление организацией передачи данных в больших объемах может осуществляться с помощью ряда протоколов, имеющих специальное функциональное назначение. В частности, существуют следующие виды TCP-протоколов. 1. FTP (File Transfer Protocol) организует перенос файлов и используется для передачи информации между двумя узлами Internet с использованием TCP-соединений в виде бинарного или же простого текстового файла, как поименованной области в памяти компьютера. При этом не имеет никакого значения, где данные узлы расположены и как соединяются между собой. 2. Протокол пользовательских дейтаграмм, или User Datagram Protocol, не зависит от подключений, он передает данные пакетами, которые называют UDP-дейтаграммами. Однако этот протокол не так надежен, как TCP, потому что отравитель не получает данных о том, был ли принят пакет в действительности.

3. ICMP (Internet Control Message Protocol) существует для того, чтобы передавать сообщения об ошибках, возникающих в процессе обмена данными в сети Internet. Однако при этом ICMP-протокол только лишь сообщает об ошибках, но не устраняет причины, которые привели к возникновению этих ошибок. 4. Telnet - сетевой протокол, который используется для реализации текстового интерфейса в сети с помощью транспорта TCP. 5. SMTP (Simple Mail Transfer Protocol) – это специальный протокол обмена электронными сообщениями, определяющий формат сообщений, которые пересылаются с одного компьютера, называемого SMTP-клиентом, на другой компьютер, на котором запущен SMTP-сервер. При этом данная пересылка может быть отложена на некоторое время до тех пор, пока не активируется работа как клиента, так и сервера. Схема передачи данных по протоколу TCP-IP 1. Протокол TCP разбивает весь объем данных на пакеты и нумерует их, упаковывая в TCP-конверты, что позволяет восстановить порядок получения частей информации. При помещении данных в такой конверт происходит вычисление контрольной суммы, которая записывается потом в TCP-заголовок. 2. Далее посредством протокола IP все пакеты передаются непосредственно получателю. 3. Затем с помощью протокола TCP происходит проверка того, все ли пакеты получены. Если во время приема вычисленная заново контрольная сумма не совпадает с указанной на конверте, это свидетельствует о том, что часть информации была утеряна или искажена при передаче, протокол TCP-IP заново запрашивает пересылку этого пакета. Также требуется подтверждение прихода данных от получателя. 4. После подтверждения получения всех пакетов протокол TCP упорядочивает их соответствующим образом и собирает заново в единое целое. Протоколом TCP используются повторные передачи данных, периоды ожидания (или таймауты), что обеспечивает надежность доставки информации. Пакеты могут передаваться в двух направлениях одновременно. Тем самым протокол TCP-IP снимает необходимость использования повторных передач и ожиданий для прикладных процессов (таких, как Telnet и FTP).

**Глава 2. Практическая часть**

**2.1. Разработка архитектуры курсовой работы**

Первая часть приложения представляет собой базовый модуль, содержащий различные классы, которые в свою очередь реализуют интерфейсы, необходимые для работы с API «ВКонтакте».

Базовый модуль осуществляет выполнение таких функций, как:

* считывание из файла информации для идентификации
* авторизация с помощью логина и пароля
* установка и поддержка HTTP-соединения;
* отправка запросов к внешним серверам и получение ответа, анализ сообщений от сервера.

Вторая часть приложения – модуль конвертации, устроенный так, что осуществляет выполнение таких действий как:

* деление его на две части (до пробела и после) с помощью встроенного метода split.
* замена встроенной функцией все падежи и виды написания конкретной величины на единый
* открытие файл values и создание коллекции типа двумерный массив "ключ"-"значение".
* анализ коллекции.
* формирование результата в сообщение и отправление пользователю.

Третьей частью приложения является модуль запуска, осуществляющий исполнение кода сразу после запуска приложения на выполнение. Его задача – в удобном виде запустить необходимый функциональный модуль.

**2.2. Краткое описание IntelliJ IDEA**

IntelliJ IDEA — это интегрированная среда разработки программного обеспечения на многих языках программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия IntelliJ IDEA появилась в январе 2001 года и быстро приобрела популярность, как первая Java IDE с широким набором интегрированных инструментов для рефакторинга[3], которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя им сконцентрироваться на разработке функциональности, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса.

Среди прочих возможностей, IntelliJ IDEA хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby.

Начиная с версии 9.0, IntelliJ IDEA доступна в двух версиях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0. В ней реализована полная поддержка Java SE, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В версии Ultimate Edition реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

**2.3. Сборка проекта и тестирование программы**

После написания основного кода классов, необходимо выполнить их сборку, используя Maven.Apache Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML. Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.

Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов. Информация для сборки проекта, поддерживаемого Apache Maven, содержится в XML-файле с названием pom.xml. При запуске Maven проверяет, содержит ли конфигурационный файл все необходимые данные и все ли данные синтаксически правильно записаны.

Минимальная конфигурация включает версию конфигурационного файла, имя проекта, его автора и версию. С помощью pom.xml конфигурируются зависимости от других проектов, индивидуальные фазы процесса построения проекта (build process), список плагинов, реализующих порядок сборки.

Проект курсовой работы поделен на несколько модулей, или подпроектов, каждый со своим собственным POM. Далее написан корневой POM, через который все модули компилируются единой командой.

Maven использует принцип Maven-архетипов (англ. Archetypes). Архетип — это инструмент шаблонов, каждый из которых определён паттерном или моделью, по аналогии с который создаются производные.

Стандартная структура каталогов — одна из реализаций принципа архитипов в Maven. Следующая структура показывает важнейшие каталоги для проекта на Java:

Корневой каталог проекта: файл pom.xml и все дальнейшие подкаталоги:

* src: все исходные файлы
* src/main: исходные файлы собственно для продукта
* src/main/java: Java-исходный текст
* src/main/resources: другие файлы, которые используются при компиляции или исполнении, например Properties-файлы
* src/test: исходные файлы, необходимые для организации автоматического тестирования
* src/test/java: JUnit-тест-задания для автоматического тестирования
* target: все создаваемые в процессе работы Maven файлы
* target/classes: компилированные Java-классы

В файле pom.xml задаются зависимости, которые имеет управляемый с помощью Maven проект. Менеджер зависимостей основан на нескольких основных принципах:

* Репозитории. Maven ищет необходимые файлы в локальных каталогах или в локальном Maven-репозитории. Если зависимость не может быть локально разрешена, Maven подключается к указанному Maven-репозиторию в сети и копирует в локальный репозиторий. По умолчанию Maven использует Maven Central Repository, но разработчик может конфигурировать и другие публичные Maven-репозитории, такие, как Apache, Ibiblio, Codehaus или Java.Net.
* Транзитивные зависимости. Необходимые библиотеки подгружаются в проект автоматически. При разрешении конфликта версий используется принцип «ближайшей» зависимости, то есть выбирается зависимость, путь к которой через список зависимых проектов является наиболее коротким.
* Исключение зависимостей. Файл описания проекта предусматривает возможность исключить зависимость в случае обнаружения цикличности или отсутствия необходимости в определённой библиотеке.
* Поиск зависимостей. Поиск зависимостей (open-source-библиотек и модулей) ведётся по их координатам (groupId, artifactId и version). Эти координаты могут быть определены с помощью специальных поисковых машин, например, Maven search engine. Например, по поисковому признаку «pop3», поисковая машина предоставляет результат с groupId="com.sun.mail" и artifactId="pop3".
* Менеджеры репозиториев. Репозитории реализуются с помощью менеджеров репозиториев Maven (Maven Repository Manager), таких как Apache Archiva, Nexus (ранее Proximity), Artifactory, Codehaus Maven Proxy или Dead Simple Maven Proxy.

После того, как создана правильная структура каталогов и файлов, корректно написаны файлы pom.xml и настроена среда разработки, проект может быть собран, классы могут быть преобразованы в байт-код, и стартовый класс может быть выполнен. Все это происходит при помощи одной команды «Run» в IDE IntelliJ IDEA. После запуска проекта, загружается необходимый модуль. Выполняется аутентификация «ВКонтакте», устанавливается Long Poll соединение. Когда эти действия выполнены, можно проверить работоспособность программы, работая с ней под видом обычного пользователя. Для этого достаточно зайти на сайт «ВКонтакте» через любой веб-браузер и написать на аккаунт, под которым авторизована программа, сообщение в установленном формате.

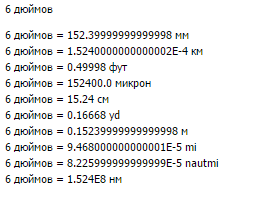


Рис.2 Пример работы программы

**Заключение**

Выполненная программная разработка соответствует заданию на курсовую работу. Была изучена документация открытых программных интерфейсов «ВКонтакте», JDK 1.86, ряда используемых сторонних библиотек и протоколов. Были разработаны несколько модулей проекта и все необходимые классы. Проведена сборка программы и проверка её работоспособности.

**Список используемой литературы**

1. Информационный портал GFC Group of freedom search// HTTP. Описание протокола URL: <http://www.gfs-team.ru/articles/read/31> (дата обращения - 24.12.2015)
2. Электронная энциклопедия Википедия// HTTP URL: <https://ru.wikipedia.org/wiki/HTTP> (дата обращения - 24.12.2015)
3. Электронная Энциклопедия Википедия// Open Systems Interconnection URL: <https://ru.wikipedia.org/wiki/Open_Systems_> (дата обращения - 24.12.2015)
4. Сообщество Habrahabr// Правильный подход к использованию API Вконтакте URL: <http://habrahabr.ru/post/221949/> (дата обращения - 24.12.2015)
5. Информационный портал OpenNET// Протоколы сетевого взаимодействия TCP/IP URL: <http://www.opennet.ru/docs/RUS/tcpip/#c2_arch> (дата обращения - 24.12.2015)
6. Электронная энциклопедия Википедия// IntelliJ IDEA URL: <https://ru.wikipedia.org/wiki/IntelliJ_IDEA> (дата обращения - 24.12.2015)
7. «ВКонтакте»// Документация платформы. URL: https://vk.com/dev (дата обращения – 24.12.2015)
8. Информационный портал FB.ru// Что такое протокол TCP-IP URL: <http://fb.ru/article/37185/chto-takoe-protokol-tcp-ip> (дата обращения – 24.12.2015)
9. Информационный портал Web of trust// API URL: <https://www.mywot.com/wiki/API> (дата обращения – 24.12.2015)
10. Информационный портал//TCP/IP URL: <http://gruzdoff.ru/wiki/TCP/IP> (дата обращения – 24.12.2015)