ФГАОУ ВО «Национальный исследовательский университет   
«Московский институт электронной техники»

Направление: Технические дисциплины

Выпускная квалификационная работа

**Разработка программного модуля автодополнения и описания слов**

**(шифр ПМ АОС)**

**Работу выполнил:**

Мадеев Константин Андреевич Студент 4 курса

ФГАОУ ВО «Национальный

исследовательский университет

«Московский институт

электронной техники»

**Научный руководитель:**

Кононова Александра Игоревна  
к.т.н., доцент

ФГАОУ ВО «Национальный

исследовательский университет

«Московский институт

электронной техники»

Москва, 2021

# АННОТАЦИЯ

Актуальность разработки обоснована тем, что во многих частях S-Terra KP Console существует необходимость вводить длинные названия ключевых слов в течение 3-5 сек, а при совершении ошибок значительно больше.

Объектом исследования является печатание слов на клавиатуре. Предметом исследования является скорость ввода слов на клавиатуре.

Цель работы: ускорение набора слов в S-Terra KP Console с помощью автодополнения с выводом их описания при наведении.

Задачи:

* исследование предметной области;
* сравнительный анализ существующих аналогов;
* выбор языка и среды программирования;
* разработка схемы данных ПМ АОС;
* разработка схем алгоритмов ПМ АОС;
* разработка пользовательского интерфейса ПМ АОС;
* программная реализация ПМ АОС;
* отладка и тестирование ПМ АОС;
* разработка руководства оператора ПМ АОС.

При разработке программного модуля используется уникальный метод сравнения введенных символов с началом слов из базы данных.

В результате выполнения выпускной квалификационной работы разработана рабочая версия программного модуля автодополнения и описания слов.

ПМ АОС предназначен для ускорения написания ключевых слов в программе S-Terra KP Console, программный модуль предоставляет возможность оперативно просматривать документацию. Разработка модуля существенно сократит время, необходимое для правильной настройки программы.

Содержание

[Перечень используемых сокращений 3](#_Toc74046087)

[Введение 4](#_Toc74046088)

[1. Исследовательский раздел 7](#_Toc74046089)

[1.1. Исследование предметной области 7](#_Toc74046090)

[1.2. Обзор существующих программных аналогов 17](#_Toc74046091)

[1.3. Цель и задачи разработки 21](#_Toc74046092)

[1.4. Концептуальная модель предметной области 22](#_Toc74046093)

[1.5. Схема данных ПМ АОС 26](#_Toc74046094)

[1.6. Требования к алгоритмам работы ПМ АОС 27](#_Toc74046095)

[Вывод по исследовательскому разделу 28](#_Toc74046096)

[2. Конструкторский раздел 29](#_Toc74046097)

[2.1. Выбор языка программирования 29](#_Toc74046098)

[2.2. Выбор среды программирования 34](#_Toc74046099)

[2.3. Реализация разработанных моделей и алгоритмов 44](#_Toc74046100)

[2.1. Разработка пользовательского интерфейса 49](#_Toc74046101)

[Вывод по конструкторскому разделу 50](#_Toc74046102)

[3. Технологический раздел 51](#_Toc74046103)

[3.1. Средства и методы отладки 51](#_Toc74046104)

[3.2. Отладка ПМ АОС 52](#_Toc74046105)

[3.3. Анализ методов и средств тестирования 55](#_Toc74046106)

[3.4. Тестирование ПМ АОС 59](#_Toc74046107)

[Вывод по технологическому разделу 64](#_Toc74046108)

[Заключение 65](#_Toc74046109)

[Список литературы 66](#_Toc74046110)

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ОПЕРАТОРА

ПРИЛОЖЕНИЕ 2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# Перечень используемых сокращений

ПМ – программный модуль

ПМ АОС – программный модуль автодополнения и описания слов

ПО – программное обеспечение

ПИ – пользовательский интерфейс

ИС – информационная система

ОС – операционная система

ЭВМ – электронно-вычислительная машина

СУБД – система управления базами данных

IDE – integrated development environment

# Введение

В эпоху бурного развития информационных технологий и быстрой цифровизации различных отраслей бизнеса скорость написания текста в различных текстовых редакторах и редакторах кода играет важнейшую роль для: быстрого создания различных документов, написания программного кода, настройки различных программ. Одним из средств, помогающих ускорить выполнение вышеперечисленных задач, является программа автодополнения слов.

В текстовых редакторах или средах разработки присутствуют функции автодополнения слов. Они позволяют значительно ускорить набор текста и программного кода, экономя время разработчика. Можно просто щелкнуть на нужное слово в появившемся окне с предложенными словами вместо того, чтобы вводить слово целиком. Вы также можете нажать стрелку вниз на клавиатуре, чтобы прокрутить список слов, а затем нажать Enter, чтобы выбрать одно из них.

Во многих частях S-Terra KP Console существует необходимость вводить длинные названия ключевых слов в течение 3-5 сек, а при совершении ошибок значительно больше. Это сильно замедляет процесс настройки программы для ее корректного функционирования.

Чтобы правильно использовать ключевые слова, необходимо обращаться к документации, заходя на официальный сайт компании С-Терра, и искать описание этого слова на странице. Данная операция длится больше 20 сек.

Для ускорения набора ключевых слов в программе S-Terra KP Console и просмотра их документации, принято решение разработать программный модуль, решающий данную задачу.

В данной выпускной квалификационной работе исследуется предметная область, проводится сравнительный анализ существующих аналогичных программных решений, происходит выбор программных средств для реализации ПМ АОС, разработка описываемого программного модуля. Также работа описывает процесс тестирования и отладки.

Выполнение выпускной квалификационной работы проходило в ООО «С-Терра СиЭсПи», специализирующейся на разработке и поставке средств сетевой информационной безопасности, а также программного обеспечения для данных средств, базирующегося на технологии IPsec VPN [51].

Разрабатываемый ПМ АОС имеет высокую практическую значимость для решения задачи по быстрому написанию слов и просмотру документации в программе S-Terra KP Console.

Цель разработки:

Упрощение набора слов в S-Terra KP Console с помощью автодополнения с выводом их описания при наведении.

Задачи:

* исследование предметной области;
* сравнительный анализ существующих аналогов;
* выбор языка и среды программирования;
* разработка схемы данных ПМ АОС;
* разработка схем алгоритмов ПМ АОС;
* разработка пользовательского интерфейса ПМ АОС;
* программная реализация ПМ АОС;
* отладка и тестирование ПМ АОС;
* разработка руководства оператора ПМ АОС.

Пояснительная записка состоит из введения, исследовательского, конструкторского, технологического разделов, заключения, списка литературы и приложений.

Исследовательский раздел содержит: исследование предметной области, актуальность изучаемой проблемы; обзор существующих программных аналогов; цель и задачи разработки; описание концептуальной модели предметной области; на основании концептуальной модели описываются наборы входных и выходных данных, а также форматы внутреннего представления данных в программе; на основании информационных потребностей пользователей определяются требования к алгоритмам работы программы.

В конструкторском разделе выбирается язык программирования и среда разработки. Приводятся описание алгоритма (способа) и методы, т.е. программные технологии решения поставленной задачи; реализация разработанных моделей и алгоритмов; описание процесса разработки пользовательского интерфейса.

Технологический раздел посвящен средствам и методам отладки, использовавшихся при разработке ПМ АОС, анализу методов и средств тестирования, описанию процесса составления тест-кейсов.

Приложение 1 содержит фрагмент кода ПМ АОС.

Приложение 2 содержит руководство оператора ПМ АОС.

Приложение 3 содержит техническое задание на разработку ПМ АОС.

Объем пояснительной записки 70 листов.

# Исследовательский раздел

## Исследование предметной области

Автодополнение – это функция в программе или ПО, с помощью которого приложение выводит выпадающее окно со списком слов, подходящих под комбинацию символов, введенную пользователем.

В мобильных ОС, таких как Android и iOS, данная функция интеллектуальным вводом текста.

В графических пользовательских интерфейсах пользователи обычно могут нажать клавишу табуляции, чтобы принять предложение, или клавишу со стрелкой вниз, чтобы принять одно из нескольких.

Автодополнение значительно упрощает взаимодействие человека с компьютером, выводя список слов, которые может ввести пользователь, основываясь на уже введенных в текстовое поле символах. Данная функция всего работает в программах с ограниченным количеством возможных слов (например, в консоли GNU/Linux, PowerShell), или в случаях, когда некоторые слова встречаются чаще, чем остальные, например, в электронной почте при наборе электронного ящика, или при написании текста, имеющем определенную структуру, как в исходном коде.

Первоначальная цель ПО для автодополнения слов заключалась в том, чтобы оказать помощь людям с ограниченными возможностями в печатании текста на компьютере [8], в том числе, уменьшив количество нажатий на клавиши, необходимых для завершения слова или предложения [9]. Скорость произношения речи у людей, которые применяют устройства для генерации слов, на 10% медленнее, чем у людей, использующих устную речь – что так же является причиной создания подобных программ [10]. Автодополнение чрезвычайно полезно для всех, кто много печатает, особенно для тех, кто использует длинные и трудные для написания слова, например, врачей.

Автодополнение работает таким образом, что, когда человек пишет один или несколько символов, программа выводит одно или несколько возможных слов для быстрого завершения написания. С помощью клавиш клавиатуры или мыши можно вывести слово, если оно есть в списке. Если слова нет в списке, то необходимо ввести еще символ, либо ввести слово целиком. Во время набора символов список слов изменяется в зависимости от набранных букв [12,13]. Другим вариантов автодополнения является «предугадывание» слов, которые с большой долей вероятности будут напечатаны вслед за написанным. Это делается на основе комбинаций использованных слов [13]. В таком случае используется языковое моделирование, при котором происходят вычисления наиболее вероятные слова [14]. Наряду с языковым моделированием используется частота употребления слов, благодаря которым строится предугадывание слов на устройствах дополнительного и альтернативного общения (ДАО). ПО для автодополнения слов практически всегда позволяет пользователю вводить свои собственные слова в словари для прогнозирования слов, но это может происходить и путем запоминания слов [15,25].

Существуют автономные инструменты, которые добавляют функцию автозаполнения к существующим приложениям. Подобное программное обеспечения проводит анализ напечатанных символов, а по результатам анализа выводит окно автодополнения с подходящими под набранное слово вариантами выбора. К примеру, существуют бесплатные программы: LetMeType или TypingAid, исходный код которых доступен любому, поскольку авторы обоих приложений прекратили разработку [46]. Бесплатная версия IntelliComplete [35] доступна только в течение 30 дней, дальше придется выложить внушительную сумму для ее использования, версии, работает только в операционной системе Windows. Многие программы автодополнения также можно использовать для создания сокращенного списка слова, которые используются чаще всего. Первоначальным программным обеспечением для автодополнения был Smartype [50]. Данная программа появилась в конце 1980-х годов, разрабатывается и в наше время. Первоначально программа разработана для людей, работающих с медицинскими терминами, но теперь она представляет из себя программу с открытым исходным кодом для автодополнения программного кода.

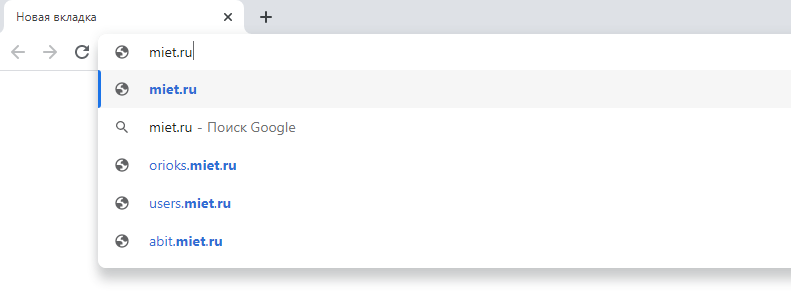
Автокоррекция, или автозамена, – это функция, одну шаблонную строку заменяет на другую шаблонную строку, но более длинную. Например, «adress» заменяется на «Moscow, Zelenograd, Ulitsa Yunosti». Данная функция может даже помогать исправлять опечатки: «Моква» заменяется на «Москва». Часть программного обеспечения (текстовые редакторы), могут определять контекст, который ввел пользователь, и на основе определенных слов, которые содержатся в специальном списке программы, способны вместо коротких шаблонных символов выводить необходимые пользователю длинные предложения.

Завершение контекста – это встроенная в текстовый редактор или специальную отдельную программу функция, которая завершает слова или всю фразу целиков, основываясь на некотором наборе обучающих данных. Полезность завершения контекста – это возможность более точно предсказывать слова или целиком строить фразы. Однако для такой программы требуется большое количество данных для обучения, в основном, из-за особенности завершения фраз. Большое набор обучающих данных доступен для редакторов популярных языков программирования и IDE, поскольку завершения контекста в них требуются гораздо чаще: именно поэтому он так популярен в вышеперечисленных программах. Подобные программы могут работать для всех программ, установленных в ОС, так и работать в конкретном приложении.

TED Notepad в июле 2006 года представил миру завершение строки, работающего по принципу завершения контекста. Это функция, при которой контекстом является сама строка, а набором обучающих данных является весь текстовый документ. Принцип работы заключается в завершении строки, которая начинается с повторяющейся фразы, до тех пор, пока одинаковых строки не различаются или предлагает список с возможными вариантами написания.

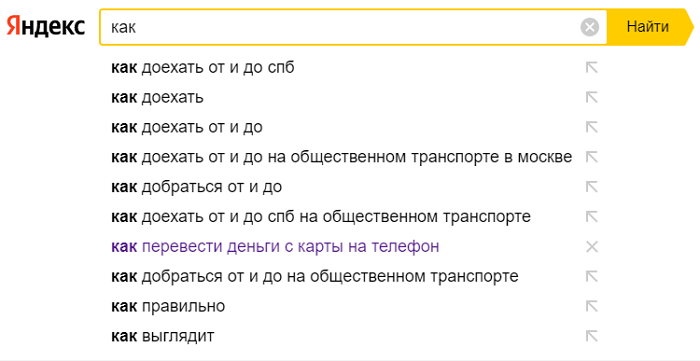
Использование программным обеспечением

Адресная строка браузера включает в себя автодополнение не только слов, но и целых предложений (рисунок 1.1). Поэтому, когда пользователь вводит текст в адресную строку (из которой можно осуществлять поиск), он видит предложения поиска, выводящиеся во время набора текста. Одновременно с вариантами поиска выводятся и URL-адреса, тем самым браузер, использует историю посещений и просмотров страниц в интернете, предлагая варианты для быстрой загрузки веб-страниц.

*Рисунок 1.1 – Автозаполнение в адресной строке Google Chrome*

В поисковых системах

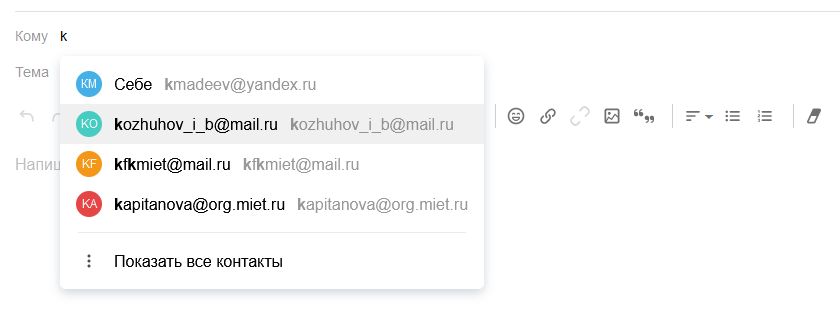
Если начать вводить текст в поисковой системе, например, Яндекса, пользователю сразу будет представлен список поисковых предложений для оперативного исполнения запроса (рисунок 1.2). Это называют инкрементным поиском. Он основан на алгоритмах сопоставления текста, которые устраняют ошибки ввода, например, алгоритм Левенштейна [16]. Далее необходимо выполнить поиск по базам данных популярных пользовательских запросов и поиск по большим индексам за очень короткое время для того, чтобы человек практически сразу увидел варианты автодополнения.



*Рисунок 1.2 – Автозаполнение в поисковой строке Яндекса*

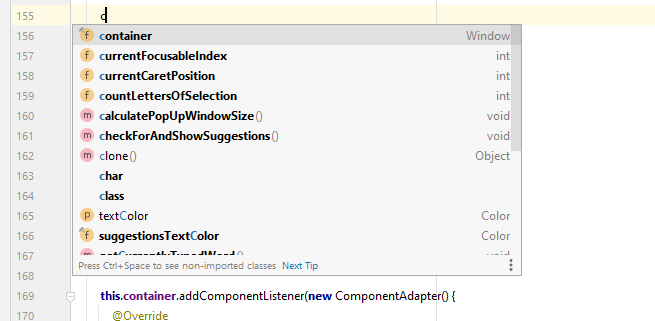
Список предложений формируется, в большинстве своем, на основе запросов всех пользователей, использующих данную поисковую систему. Но совершенствование алгоритмов автозаполнения не стоит на месте и может использовать другие источники информации. Например, поисковая система может выдавать запросы на основе текущего местоположения пользователя, беря в расчет дату и время, чтобы выдавать наиболее удовлетворяющие запросу результаты. Также браузеры собирают информацию об истории посещений и запросов пользователя, его личную информацию в виде логинов и паролей – так называемые cookie. При соответствующем разрешении от пользователя, поисковые системы могут использовать эти файлы для выдачи информации на основе личных интересов человека.

Для автодоплнения темы сообщения, адресов для отправки сообщения и автодополнения слов в программах электронной почты присутствует ПО для этих целей. Адреса электронной почты довольно часто бывают непонятными, труднозапоминаемыми и длинными, что значительно сказывается на удобстве их ввода. Автодополнение в таком случае довольно просто реализуется, поскольку относительно мало постоянных получателей писем. Например, Яндекс почта найдет адреса на основе уже отправленных писем (рисунок 1.3) или, наоборот, полученных.



*Рисунок. 1.3 – Автозаполнение адресата в электронной почте Яндекса*

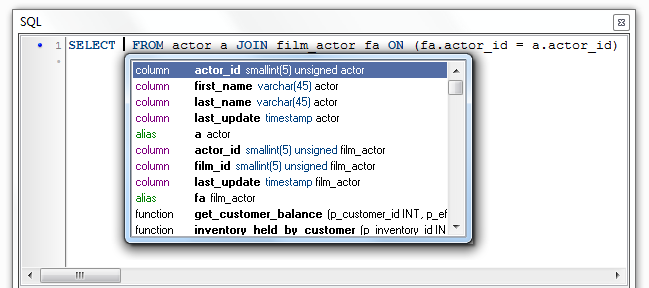
Из-за ограниченного количества слов языков программирования в средах разработки автодополнение является довольно упрощенным, если говорить лишь о завершении слов. Причиной этого, в основном, является ограничения самих языков программировния в виде текущего пространства имен или контекста. Например, в текущем пакете языка Java вы можете использовать protected классы, которые объявлены в этом пакете, за исключением публичных классов в других пакетах. Например, программа автодополнения кода IntelliSense от Microsoft может отображать окно со списком слов, которые может выбрать программист. Список обновляется по мере ввода новых символов или вовсе исчезает, если совпадений со словами в словаре не будет найдено. Данная функция особенно полезна в ООП, потому что прим написании экземпляра класса она уже точно знает, какие методы может использовать разработчик, а какие ему недоступны. К тому же, она способна выводить документацию для определенных методов или классов, облегчая доступ к информации об их применении. Эта особенность является большим преимуществом автодополнения для исходного кода, потому что оно позволяет программистам использовать говорящие имена переменных, применяя стиль написания CamelCase (или др.) – это делает исходный код намного читабельнее. Ввод больших слов, составленных из нескольких маленьких, таких как «CostomAttributeArgument», может быть трудозатратным, но автодополнение поможет написать слово целиком, в то время как разработчик напечатал пару символов.



*Рисунок 1.4 – Автодополнение кода в IntelliJ IDEA*

На рисунке 1.4. показано автодополнение кода: программист вводит некоторый код, и ПО автодополнения, анализируя введенные символы и понимая, что это название переменной, класса или метода, отображает список слов, подходящих под веденную строку. В IDE данная функция очень умная, поскольу сопоставляет связи классов и объектов с содержащимися в них методами и переменными, учитывая структуру проекта и модификаторы доступа. Если программист продолжит вводить, не выбрав какой-то вариант, то список будет обновляться до тех пор, пока есть совпадения.

При составлении запроса к базе данных редактор СУБД, обычно, имеет встроенную функцию автодополнения, которая способна определять динамически, что требуется ввести в данный момент пользователю: имя таблицы, столбца или оператора. Такое возможно благодаря пониманию контекста редактора запросов СУБД. При этом предоставляются таблицы той базы данных, к серверу которой подключился пользователь. Функция автодополнения, если пользователю нужно ввести таблицу, предоставляет список таблиц, доступных на сервере базы данных, к которой подключен пользователь. Если пользователю нужен столбец таблицы, автодополнения выводит список столбцов таблицы, указанной в операторе SQL. Популярная СУБД MySQL обеспечивает автодополнение в запросах.



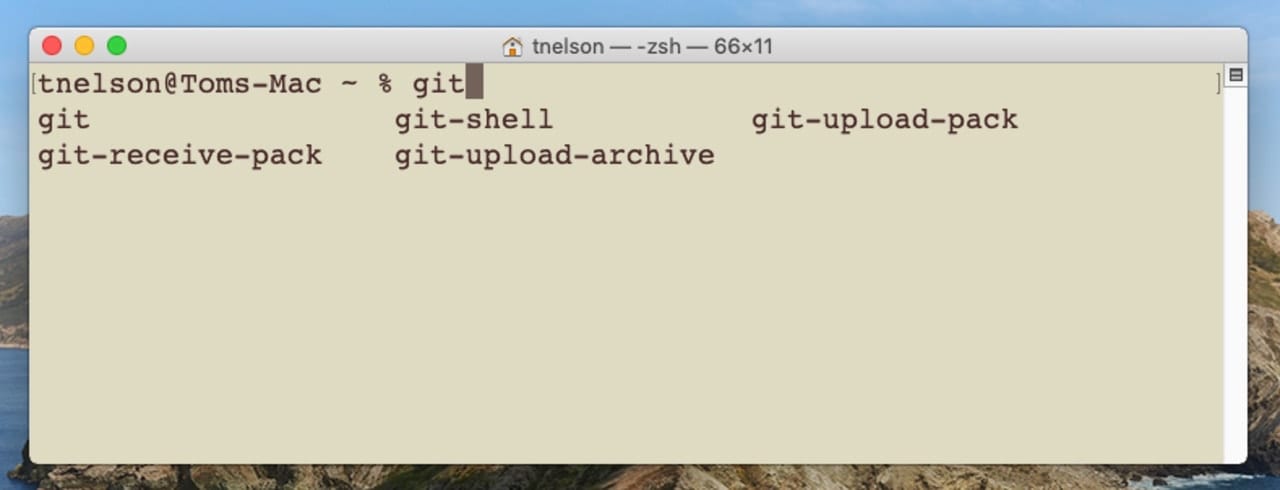
*Рисунок 1.5 – Автозаполнение имени столбца в MySQL*

Автодополнение в текстовых редакторах значительно сокращает время, необходимое для набора слов и предложений. База слов составляется из уже написанных слов в текстовом документе или из заранее определенного списка со словами. Например, офисные пакеты приложений для чтения файлов формата .doc и .docx: Apache OpenOffic и Microsoft Office имеют функцию автодополнения, однако их надо включать в параметрах программы и заниматься их отдельной настройкой. Редактор таблиц Microsoft Excel тоже включает в себя функцию автодополнения, которая работает на основе ранее введенных слов в верхние ячейки.



*Рисунок 1.6 – Автодополнение в текстовом редакторе Sublime Text 3*

В консоли ОС GNU/Linux или Windows: bash или PowerShell соответственно автодополнение работает как для функций комнадной строки, так и для различных имен файлов в текущей директории. Обычно используется клавиша Tab, которая выводи список всех возможных вариантов использования, если их много, и вставляет слово сразу, если их мало или его можно определить конкретно. Если в папке есть один файл, который начинается с «s», – это someFile – пользователь может ввести s и нажать Tab, чтобы автодополнение вставило все слово сразу. Если бы в той же области было другое имя файла или команда, начинающаяся с «s», пользователю пришлось бы напечатать больше символов, нажать клавишу Tab несколько раз, чтобы отобразилось нужное слово, или возможна ситуация, изображенная рисунке 1.7.



*Рисунок 1.7 – Автозаполнение в командной строке macOS*

Эффективность

Средняя длина слова в языке и длина слова в набранном тексте влияет на то, насколько эффективно автодополнение. Редакторы кода: IDE или универсальные редакторы, которые способны распознавать синтаксис различных языков программирования, например, Sublime Text – почти всегда имеют автодополнение слов в своем арсенале, поскольку они значительно упрощает набор ключевых слов языка и становится легче печатать различные имена переменных, классов и т.д. Очевидно, что в разных человекческих языках разная средняя длина слов. Из этого следует, что автодополнение может иметь разные степени полезности – от эффективного до неэффективного. К примеру, рассмотрим слово «футболист». На немецкий язык оно переводится как «fussballspieler», а на английский «footballer». Длина этих слов 15 и 10 символов соответственно. Этот пример показывает, что английский язык, по сравнению с немецким, менее эффективен для автодополнения. В исследовании «Notes on Compound Word Hyphenation in TEX» [11] показано, что, в среднем, длина слов для английского языка составляет 8,93; для чешского это число составляет 10,55; а для немецкого – 13,24. Стоит также учитывать, что в фузионных и агглютинативных языках, например, в немецком, слова можно соединять между собой, создавая более сложные и длинные слова. Люди, которым необходимо часто использовать длинные слова, например, врачи и химики, очевидно, нуждаются программном обеспечении для автодополнения слов.

Исследования показали, что ПО для предсказания слов действительно уменьшает количество необходимых действий и улучшает письменную продуктивность пользователя [8]. Однако из-за необходимости переводить взгляд с клавиатуры на экран увеличивается нагрузка на мозг, что в конечном счете может приводить к быстрому утомлению. Если уменьшить количество слов в появляющемся окне автодополнения, добавив вертикальную прокрутку, то когнитивную нагрузку, возникающую во время печатания, можно уменьшить. Вертикальное расположение слов сведет к минимуму перевод взгляда с клавиатуры на монитор. Многие разработчики ПО считают, что список автодополнения должен следовать за курсором вслед за набором текста, поскольку это также уменьшит количество движений глаз. В исследовании, в котором рассматривали скорость печатания у детей снезаращенными дужками позвонка, проведенном Tam, Reid, O'Keefe и Nauman (2002), показано, что детям было гораздо легче печатать если есть автодополнение, при этом совершалось меньше ошибок, а также выяснилось, что удобно, когда список отображается снизу слова. Другие исследования также показали, что польза автодополнения и легкость набора слов повышаются, когда список слов находится ниже строки, в которой происходит ввод. Программное обеспечение автодополнения слов существует в большом количестве вариантов. Данное ПО можно купить как дополнительное для других программ, таким как Microsoft Word. Например, к таким программам относятся WordQ + SpeakQ [48], Typing Assistant [47]). Или функция автодополнения будет поставляться вместе с текстовым редакотором. Программа от IntelliComplete распространяется как в платной, так и в бесплатной версии, но она работает только с тем ПО, для которого подходит. Letmetype и Typingaid – бесплатные программы, которые работают в любом текстовом редакторе. Впервые автодополнение применил Кристофер Лонге-Хиггинс в 1967 году, поскольку в своей книге он упоминает, что в его «Печатной машинке с программным управлением» [16]: «Такие слова, как «НАЧАТЬ» или «ПРОЦЕДУРА», или различные заранее определенные наборы символов, введенные человеком, автоматически допечатывались машинкой после того, как программист ввел несколько символов».

## Обзор существующих программных аналогов

В ходе предварительных исследований проведен анализ существующих решений с функционалом, требуемым от ПМ АОС. Помимо необходимых функицй, которые должна осуществлять ПМ АОС: выпадающий список, описание слова при наведении, управление выбором слова с помощью устройств ввода – но и дополнительные возможности, облегчающие использование автодополнения; также учитывались требования компании к программным технологиям, которые в обязательном порядке нужно использовать при создании ПМ АОС. Характеристики рассмотренных решений в сравнении с ПМ АОС представлены в таблице 1.1.

Таблица 1.1

Сравнительный анализ существующих решений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Решение  Критерий | ПМ АОС | Swingx[26] | Uberi Autocomplete[27] | GlazedList[28] | TypingAid[29] |
| Выпадающий список | Да | Да (с ограничениями) | Да | Да (с ограничениями) | Да |
| Подсветка слова при наведении | Да | Да | Да | Да | Да |
| Описание подсвеченного слова | Да | Нет | Нет | Нет | Нет |
| Выбор слов с помощью мыши и клавиатуры | Да | Да | Да | Да | Да |
| Решение на Java Swing | Да | Да | Нет | Да | Нет |

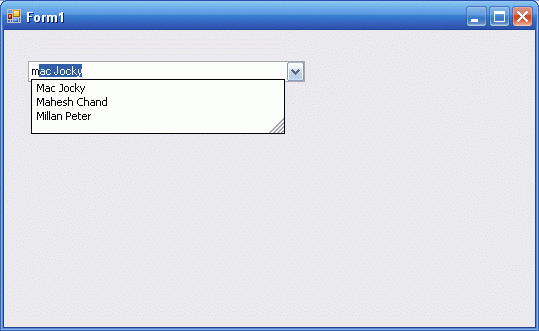
Продолжение таблицы 1.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Не является сторонней программой | Да | Да | Нет | Да | Нет |

Swingx

Swingx – библотека Java, котороя позволяет создавать многофункциональные приложения благодаря тому, что содержит улучшенные компоненты и расширения для стандартной библиотеки Java Swing.

Также содержит компоненты, с помощью которых можно реализовать автодополнение текста в программе (рисунок 1.8). Но без описания подсвеченного слова и с ограничениями по улучшению внешнего вида окна автодополнения.



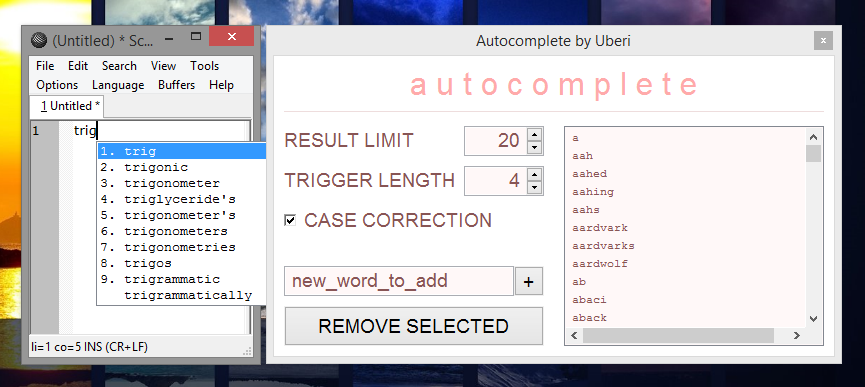
*Рисунок 1.8 – Пример работы автодополнения в библиотеке swingx*

Uberi Autocomplete

Uberi Autocomplete – программа автодополнения от пользователя github Uberi, которая показывает окно с предлагаемыми словами. Программа имеет диалоговое окно настроек. В нем можно настроить такие параметры, как максимальное количество результатов, отображаемых за раз, и количество символов, которые нужно ввести перед отображением предложений (рисунок 1.9).

Диалог также позволяет редактировать список слов, добавляя или удаляя слова. Все настройки автоматически сохраняются при выходе из программы.

Список слов содержится в специальном текстовом файле. Каждая строка представляет собой слово. Список можно редактировать с помощью текстового редактора.



*Рисунок 1.9 – Пример использования программы автодополнения от Uberi*

GlazedList

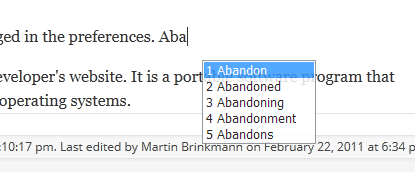
GlazedList – библиотека для создания приложений на основе Swing для представления данных, которые можно динамически сортировать и фильтровать по различным критериям. Автодополнение является вспомогательной функцией данной библиотеки (рисунок 1.10). Оно не может показывать описание элементов, и отсутствует возможность улучшать внешний вид компонентов.



*Рисунок 1.10 – Пример работы автодополнения в библиотеке GalzedList*

TypingAid

Typing Aid – это бесплатная переносимая программа для Windows, которая обеспечивает функцию автозаполнения текста на общесистемном уровне



*Рисунок 1.11 – Пример использования программы TypingAid*

Программа работает в фоновом режиме и совместима с большинством, если не всеми, приложениями и текстовыми формами, включая те, которые находятся в веб-браузере и текстовых редакторах (рисунок 1.11).

В программе есть несколько параметров совместимости, которые необходимо изменить, если автозаполнение не работает из коробки. Можно добавить заголовки для определенных окон и процессов в предпочтениях, чтобы помочь с этим в дальнейшем. Также вы можете отключить Typing Aid для выбранных заголовков окон или процессов. Последнее полезно, если вы заметили проблемы при использовании программы, если включен Typing Aid.

Слова взяты из wordlist.txt, который нужно поместить в директорию программы. Приложение не поставляется со списком слов, но вы можете загрузить несколько слов с сайта, на котором оно предлагается.

Кроме того, вы также можете создать свой собственный список слов с нуля. Программа поддерживает юникод и работает с акцентами. В нем есть режим обучения, позволяющий автоматически добавлять новые слова или фразы в свою базу данных.

Поскольку ни одно из существующих программных решений не удовлетворяет полностью установленным критериям, создание собственного программного модуля автодополнения и описания слов необходимо для программы S-Terra KP Console.

## Цель и задачи разработки

S-Terra KP Console – это программа, разрабатываемая компанией ООО «С-Терра СиЭсПи», предназначенная для конфигурации и настройки одной защищенной сети для VPN-устройств, разрабатываемых компанией С-Терра, а также для централизованного управления этими устройствами, со сбором всей необходимой информации, помогающей контролировать их работу.

Во время настройки программы для правильного функционирования необходимо часто совершать рутинные действия, в число которых входит частый набор длинных ключевых слов, написанных CamelCase методом: dot1dBaseBridgeAddress, cikeGlobalOutP2ExchgInvalids, cikeGlobalInDropPkts, cikeGlobalInP2ExchgInvalids, ciscoImageString и т.д. Самостоятельно печатать такие длинные и часто сложные слова, при этом не имея под рукой документации, которая поможет узнать их назначение, – многие слова похожи друг на друга – доставляет много неудобств и сильно замедляет процесс настройки программы. Для решения этой проблемы стал вопрос о разработке программного модуля для автодополнения и описания слов.

Таким образом, поставлена цель разработки:

Упрощение набора слов в программе S-Terra KP Console с помощью автодополнения с выводом их описания при наведении.

Задачи:

* исследование предметной области;
* сравнительный анализ существующих аналогов;
* выбор языка и среды программирования;
* разработка схемы данных ПМ АОС;
* разработка схемы алгоритма ПМ АОС;
* разработка пользовательского интерфейса ПМ АОС;
* программная реализация ПМ АОС;
* отладка и тестирование ПМ АОС;
* разработка руководства оператора АОС.

## Концептуальная модель предметной области

Концептуальная модель – это абстрактное описание будущей программной системы с помощью различных сущностей и связей между ними. Данная модель показывает описывает главные компоненты системы, раскрывая их функции, характеристики. Однако основной смысл данной модели в том, что она описывает связи между компонентами системы. Описывается модель не только с помощью различных диаграмм, но также графиков или таблиц. Концептуаьлная модель полезна для понимания программной системы не только программистами, но и другими специалистами, принимающих участие в создании системы, в том числе самими заказчиками. Приведем следующую классификацию методологии разработки ПО:

* структурный подход, подразумевает рассмотрение алгоритма как множество составляющих его функций, способах взаимодействия и передачи данных между ними.
* объектно-ориентированный подход [17], основан на рассмотрении поведения объектов внутри системы: работа системы исследуется через отношения объектов между собой, а внутреннее устройство программной системы описывается совокупностью всех объектов и связей между ними.

Обе методологии содержат в себе такие понятия как: нотации, методология и методы.

Методология – это совокупность различных методов, с помощью которых проводится исследование и разработка продукта. Методы концептуального проектирования информационных систем [7] основываются на системном использовании различных графических элементов (для связей и сущностей) и условий, при которых они используются – этот процесс называется нотацией моделирования.

Совокупность главных правил структурного подхода называется методологией структурного анализа и проектирования — SADT (Structured Analysis and Design Technique). После появления структурных методов проектирования программного обеспечения стал зарождаться новый вид управления разработкой, которая получила название бизнес-моделирование. Одной из основных концепций данной дисциплины является понятие бизнес-процесса. Это совокупность следующих друг за другом действий, с помощью которых преобразование данных выдает, необходимый пользователю результат.

Одни из самых популярных и эффективных методов структурного подхода к конструированию ПО точно так же эффективно помогают описывать и расширять бизнес-процессы. Например, известные методы концептуального проектирования DFD, IDEF1X, IDEF0, IDEF3 весьма успешно справляются с этой задачей. Данная особенность позволяет продемонстрировать особенности разрабатываемого ПО и показать необходимые для бизнес-процесса функции, что облегачет потенциальное принятие концептуальной модели заказчиками. Совокупность методов структурного и объектно-ориентированного подхода продемонстраирована в методологии бизнес-моделирования ARIS (Architecture of Integrated Information Systems), благодаря чему описание различных особенностей как разработки ПО, так и деятельности компании выходит на качественно новый уровень.

Из всех методов проектирования бизнес-процессов в ARIS-методологии сильно выделяется eEPC (Event Process Chain), поскольку в его инструментарий, помимо описания различных функций: данные, которые они получают и выдают; способах их взаимодействия и их роли в общей структуре проекта – позволяет также уделять внимание участникам бизнесс-процесса, различных материальных объектов и проч. По этой причине ARIS-методология с успехом используется во время концептуального проектирования при создании различных протоколов и регламентов применения ПО.

Чтобы применять объектно-ориентированный подход чаще всего используют средства языка графического моделирования – UML (Unified Modeling Language). Данный язык чрезвычайно эффективен для выполнения концептуальной модели разрабатываемого ПО, поскольку архитектор ПО может использовать инструменты для четкого представления структуры системы, описания связей между её компонентами и ее поведения, применяя несколько нотаций моделирования [18].

Всего в языке графического моделирования применяется около 15 разных диаграмм, которые с разной степенью абстракции описывают возможности, структуру, связи и поведение будущего программного обеспечения. Наиболее известными и часто используемыми видами диаграмм являются:

* классов;
* деятельности;
* компонентов и развертывания;
* последовательностей;
* состояний;
* вариантов использования.

Между этими видами диаграмм есть много общего, к примеру, разбиение предметной области на компоненты по степени влияния друг на друга и на функции, которые выполняет каждый из них. Однако все-таки между ними значительная разница в методологии концептуального проектирования программного обеспечения, благодаря которой вышеуказанные диаграммы выделяются в отдельные виды.

Концептуальную модель опишем с помощью диаграммы вариантов использования, поскольку она понятно и лаконично отображает задумку ПМ АОС. Ее смысл заключается в следующем: разрабатываемая система представляется в виде связей, которые выступают в роли вариантов использования, между внешними (акторами) и внутренними компонентами системы (сущностей). Актором называют такую сущность, которая взаимодействует с системой извне, применяя специальный интерфейс. Актором может быть как человек, так и аппаратная система, программное обеспечение или же это может быть какая-нибудь иная сущность, которая сможет так или иначе влиять на разрабатываемую систему. Вариант использования является совокупностью функций, на которые актор может использовать при взаимодействии с системой либо данные, которая система ему предоставит. Диаграмма вариантов использования может пояснять различными комментариями, которые помогут лучше понять работу моделируемой систему.



*Рисунок 1.12 – Диаграмма вариантов использования*

Диаграмма вариантов использования, изображенная на рисунке 1.12, отражает отношения между основными сценариями выполнения программы и участниками (ролями), принимающими участие в работе системы. Данная схема применяется для наглядной демонстрации основных вариантов использования разработанного модуля.

Диаграмма вариантов использования – это диаграмма, на которой представлены последовательные шаги, благодаря которым осуществляется взаимодействие действующих лиц между компонентами системы, а также взаимодействия между компонентами системы.

Предполагаемый алгоритм решения:

ПМ АОС прослушивает изменения в текстовом поле. При вводе символов ищет совпадения в БД. Выводит на экран те слова, начало которых совпадает с веденными символами.

## Схема данных ПМ АОС

Структура данных ПМ АОС показана на рисунке 1.13.

Входными данными во время начала работы программы являются слова из БД для последующего их сопоставления с вводимыми символами. Далее в текстовое поле пользователем вводятся данные с клавиатуры.

Введенные с клавиатуры данные сопоставляются с данными из БД. При совпадении данные из БД заносятся в список слов. По завершении сопоставления список слов (автодополнение) выводится на экран.

*Рисунок 1.13 – Схема данных ПМ АОС*

При отображении автодополнения можно ввести в текстовое поле другие данные, произойдет тот же алгоритм сопоставления слов, в процессе которого формируется новый список слов. В автодополнении отобразится обновленный список, или оно исчезнет.

В появившемся автодополнении можно выбрать слово, которое необходимо вставить, с помощью клавишей «вверх/вниз», чтобы подсветить нужное слово, и Enter, чтобы вставить подсвеченное слово в текстовое поле, либо с помощью двойного клика мыши. Один клик мыши подсвечивает слово.

Если слово подсвечено, загружаются данные из БД, чтобы вывести описание (документацию) на экран.

После вставки слова, автодополнение и описание исчезают.

Ввод данных прекращается при закрытии окна с текстовым полем.

## Требования к алгоритмам работы ПМ АОС

ПМ АОС должен обеспечивать выполнение следующих функций:

* вывод корректного списка со словами, начало которых совпадает с введенными символами, под текстовым полем;
* при отсутствии совпадений между введенной комбинацией символов и словами из БД список со словами не должен появляться
* при написании нескольких символов программа должна обновлять список со словами, в котором будут отображаться только те слова, которые подходят под введенную пользователем комбинацию символов;
* при удалении букв программа должна обновлять список слов, а при удалении всех написанных первоначальных символов должна убирать список слов;
* при написании какого-либо слова перед другим словом без пробелов должен выводиться список со словами;
* при написании какого-либо слова сразу после другого слова без пробела список со словами не должен выводиться;
* подсветка слова должна происходить с помощью клавиш клавиатуры вверх/вниз, а также можно подсветить слово щелкнув один раз ЛКМ;
* подсветка слова должно исчезать, если список со словами обновляется;
* вывод описания подсвеченного слова;
* при выделении написанных букв с помощью мыши или комбинации стрелок вправо/влево с клавишей shift описание должно исчезать;
* при пропадании или обновлении списка со словами описание должно исчезать;
* описание должно иметь динамический размер в зависимости от количества информации, содержащейся в нем;
* описание должно появляться с той стороны текстового поля, которая имеет достаточную площадь, чтобы вместить его;
* вставка выбранного слова вместо введенных символов;
* вставка слова должна осуществляться без каких-либо ошибок;
* введенные в текстовом поле буквы должны иметь цвет, отличающийся от цвета остальных букв слова в списке слов;
* комбинация клавиш ctrl+space должна вывести список со всеми словами, находящимися в БД.

## Вывод по исследовательскому разделу

В исследовательском разделе обоснована актуальность разработки ПМ АОС. Исследована предметная область и проведен анализ решений для вывода автодополнения. Рассмотрена практическая значимость ПМ АОС. Также определен состав входных и выходных данных ПМ АОС, состав выполняемых функций.

# Конструкторский раздел

## Выбор языка программирования

В таблице 2.1 представлены критерии, по которыми должен обладать язык программирования, с помощью которого будет написан ПМ АОС.

Опыт работы с языком программирования необходим, поскольку для качественного написания кода необходимо понимать внутреннее устройство языка: строение системы классов, работу методов, различных структур данных и многое другое.

Для создания графического интерфейса нужны различные инструменты, которые помогут решить эту задачу. Чтобы не искать надежные и подходящие библиотеки, язык должен иметь встроенную библиотеку для GUI.

Поддержка парадигмы ООП значительно упростит разработку ПМ АОС поскольку данный подход значительно популярнее остальных, его просто применять, к тому же вся программа S-Terra KP Console разработана с применением объектно-ориентированного подхода.

Язык должен быть кроссплатформенным, чтобы и ПМ АОС в версии программы S-Terra KP Console для GNU/Linux, и сама программа корректно работали.

Популярность языка способствует быстрому решению возникающих проблем, которые могут возникнуть в процессе разработки ПМ АОС.

Для быстрой и качественной разработки язык должен обладать: легким и понятным синтаксисом; сборщиком мусора, чтобы не возникало проблем с памятью; современными встроенными функциями, для оперативного написания программного кода; строгой типизацией, для исключения ошибок, связанных с типами.

Язык исходного кода, на котором написан графический интерфейс, основной критерий, поскольку написание графических элементов на языке, отличным от основного языка программы, может значительно затруднять разработку, а в дальнейшем, и поддержку приложения.

Таблица 2.1

Сравнительный анализ языков программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Язык  Критерий | Java[31] | C#[32] | C[33] | C++[34] | Python[52] |
| Опыт работы с языком программирования | + | - | - | - | - |
| Встроенная библиотека для GUI | + | + | - | - | + |
| Поддержка ООП | + | + | - | + | + |
| Поддержка в GNU/Linux | + | + | + | + | + |
| Развитое комьюнити | + | + | + | + | + |
| Удобство разработки | + | + | - | - | + |
| Язык исходного кода GUI S-Terra KP Console | + | - | - | - | - |

C – язык программирования, разработанный в 1969-1973гг. Деннисом Ритчи. Для его исполнения необходима компиляция, а также он является статически типизированным. Создавался, в первую очередь, для написания ОС GNU/Linux, но, из-за свойств и удобствами, которыми он обладал по сравнению с существовавшими на тот момент языками программирования, стал использоваться и на других операционных системах для самых различных целей, а на данный момент его поддерживают все операционные системы. Поскольку С транслируется сначала в ассемблер, то и применение он нашел на соответствующих платформа, в том числе и для создания обычных приложений. Язык C больше других языков повлиял на развитие программного обеспечения и благодаря ему появилось семейство С-подобных языков, которые являются очень популярными среди программистов.

Программы на C отличаются своей сложностью из-за отсутствия реализации многих повсеместно использующихся структур данных и синтаксической бедностью языка, что приводит к большому объему исходных текстов, а также необходимостью ручного управления памяти, что может привести к небезопасному поведению программы. С другой стороны, программы на C быстро компилируются и отличаются высокой скоростью работы по сравнению с любыми другими языками.

Последний значительный стандарт языка вышел 8 декабря 2011 года и назывался C11. С тех пор выходили только документы, которые исправляли неточности и ошибки в стандарте, но дополнительного функционала с 2011 года реализовано не было [21].

С++ – язык программирования, который является потомком языка С. Он, в отличие от своего предка, поддерживает большее количество парадигм программирования (включая те, что поддерживает язык C): объектно-ориентированное программирование, обобщенное программирование, функциональное программирование – благодаря чему гораздо более востребован у современных разработчиков, чем С. С++ создан Бьёрном Страуструпом, который выпустил ранние версии языка в 1980 году в качестве расширения языка C, однако далее этот язык быстро стал развиваться отдельно.

Особенностями языка являются высокая производительность, гибкость при разработке ПО, наличие множества разнообразных синтаксических конструкций, которые постоянно пополняются и обновляются. С++ является компилируемым и кроссплатформенным языком.

В настоящее время язык разрабатывается международным комитетом по

стандартизации, которая обсуждает, разрабатывает и принимает новые стандарты языка. Новые стандарты выходят раз в три года, последний появился в декабре 2020 года под названием C++20.

Язык сильно отличается от своего предшественника C и предоставляет множество различных высокоуровневых возможностей на уровне языка и богатые на функционал сторонние библиотеки вроде Boost.

С# – это объектно-ориентированный язык программирования. Он создан в период с 1998 по 2002 год командой инженеров Microsoft как язык программирования, предназначенного для разработки приложений для платформы.NET Framework от Microsoft, под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Является наследником лучших практик от С++, Delphi, Java и др., а многие недостатки этих языков или противоречивые свойства, разработчики Microsoft решили устранить. Синтаксис во многом похож на Java, также на многие особенности С++. Некоторые свойства C#:

* поддерживается перегрузка операторов;
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции;
* статистическая типизация;
* поддерживается полиморфизм.

Новейшая версия C# 9.0 вышла в 10 ноября 2020 года.

В данный момент Microsoft активно развивает платформу .NET, которая объединила в себе платформы .NET Core, Mono и .NET Framework и обеспечивает кроссплатформенность языка C#, из-за совместимости данной платформы со многими популярными ОС для ПК. Однако полноценной кроссплатформенности у него нет, например потому, что c помощью него нельзя разрабатывать ядра ОС.

Java – объектно-ориентированный, язык программирования, который начали разрабатывать в начале 90-х в компании Sun Microsystems, а официально представлен широкой общественности в 1995 году. Первоначальной целью разработки данного языка была возможность его применения в различных видах бытовой техники, однако его эффективность и кроссплатформенность стала очень привлекательной и для разработчиков приложений, что и сыграло значительную роль в его распространении.

Отличительной особенностью языка Java, является то, что он транслируется в байт-код, которые выполняется с помощью Java – виртуальной машине Java – которая передает процессору команды для исполнения. Благодаря этому и достигается кроссплатформенность Java: нужно всего лишь установить на устройство JVM, и оно сможет запускать программы на Java. У других языков, например, С или С++, стоит проблема компиляции для каждой, отдельно взятой, архитектуры, как для процессора, так и для операционной системы.

Некоторые особенности Java [38]:

* из-за грамотного подхода создателей к синтаксису языка Java, разработка надежного программного обеспечения значительно упростилась;
* код программы проверяется не только во время компиляции, но также анализируется во время её исполнения;
* отсутствие явных указателей значительно упрощает для работу с памятью. Java осуществляется при помощи ссылок, благодаря чему устраняется множество ошибок, которые могут появляться из-за указателей в языках C и С++.
* поскольку безопасность работы в распределенной среде является одним из основных проблем, то Java как раз имеет все преимущества для разработки приложений в этой области.

Python – это интерпретируемый, высокоуровневый язык программирования, который использует различные парадигмы программирования: объектно-ориентированное, функциональное, процедурное и др. Использует динамическую строгую типизацию для определения типов переменных. Данный язык является чрезвычайно популярным [53,54], поскольку он достаточно простой для изучения, у него удобный синтаксис, что влияет на читаемость кода, дальнейшую разработку и поддержку приложения. Обладает высокоуровневыми структурами данных и связыванием, что совокупности с динамической типизацией эти факторы влияют на скорость разработки приложений, позволяя быстро писать работающие в начале разработки ПО. Используется как сценарный языка для связи программных компонентов между собой. В нем присутствуют модули и пакеты, что влияет на качество кода разработчиков, поскольку ценится повторное использование кода. Популярный интерпретатор языка CPthon и большинство библиотек распространяются бесплатно, а на их основе могут создаваться коммерческие приложения.

Python – это язык программирования, который можно использовать при решении большиного объема задач. Он имеет свои преимущества и недостатки, а также сферы применения. В Интернете доступны качественные библиотеки для Python по различным предметным областям: средства обработки текстов и технологии Интернет, обработка изображений, инструменты для создания приложений, механизмы доступа к базам данных, пакеты для научных вычислений, библиотеки построения графического интерфейса и т.п. Кроме того, Python имеет достаточно простые средства для интеграции с языками C, C++ (и Java) как путем встраивания (embedding) интерпретатора в программы на этих языках, так и наоборот, посредством использования библиотек, написанных на этих языках, в Python-программах.

В результате сравнения представленных языков программирования выбран Java, как наиболее удовлетворяющий поставленным задачам для разработки ПМ АОС.

## Выбор среды программирования

Что такое IDE и его преимущества?

IDE – это интегрированная среда разработки – совокупность множества инструментов для существенного увеличения простоты, скорости, надежности и качества разработки ПО за счет автоматизации многих операций, которые совершали программисты ранее, для написания качественного кода:

* содержит в себе комплекс инструментов для работы с программным кодом: текстовый редактор, компилятор и отладчик, средства автоматизации сборки, система контроля версий, возможность устанавливать пользовательские плагины, которые улучшают функциональность самой IDE, например, различные инструменты для тестирования (модульное, интеграционне, тестирование производительности) и т.д.;
* в продвинутых IDE очень умная система понимания контекста и кода в целом, что значительно упрощает и ускоряет разработку. Этому также способствует функция автодополнения кода. В них также поддерживается полный цикл разработки ПО.

Среда разработки должна быть удобной для быстрого и комфортного создания программного обеспечения. В критерии удобства входят: лаконичный и красивый дизайн, встроенный анализатор, понимание контекста, автодополнение и преобразование кода и др.

Опыт работы в среде программирования нужен, чтобы не тратить время на обучение работе в IDE и на привыкание к её качественному использованию.

Поскольку графический интерфейс программы S-Terra KP Console создан с помощью использования библиотеки Java Swing, то для проектирования пользовательского интерфейса ПМ АОС, нужен встроенный в IDE Swing GUI designer.

В IDE должна осуществляться поддержка системы контроля версий CVS, хотя бы с помощью плагинов. Это необходимо, поскольку в компании используется именно эта система контроля версий.

Среда разработки должна быть кроссплатформенной, чтобы проверять работу ПМ АОС в версии программы S-Terra KP Console для GNU/Linux.

Для отладки ПМ АОС необходим удобный и надежный отладчик, чтобы быстро и безошибочно определять проблемные места в программе. Поскольку официальные разработчики IDE заботятся о качестве своего продукта, то встроенный отладчик, скорее всего, будет надежным.

В таблице 2.2 представлены результаты анализа по расписанным выше критериям.

Таблица 2.2

Сравнительный анализ сред разработки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Среда  Критерий | IntelliJ IDEA[40] | NetBeans[41] | Eclipse[42] | Oracle JDeveloper[43] | JCreator[44] |
| Удобство работы | + | - | - | - | - |
| Опыт работы | + | - | - | - | - |
| Встроенный Swing GUI designer | + | + | - | + | - |
| Поддержка CVS | + | + | + | + | + |
| Версия для GNU/Linux | + | + | + | + | + |
| Встроенный отладчик | + | + | + | + | + |

IntelliJ IDEA

IntelliJ IDEA – самая популярная IDE для разработки приложений с помощью Java. Её популярность легко обосновывается не только ее дружелюбным и красивым пользовательским интерфейсом, но также большой функциональностью, которая значительно облегчает разработку ПО. Данная IDE разработана и поддерживается компанией JetBrains и распространяется в бесплатной и платной версии.

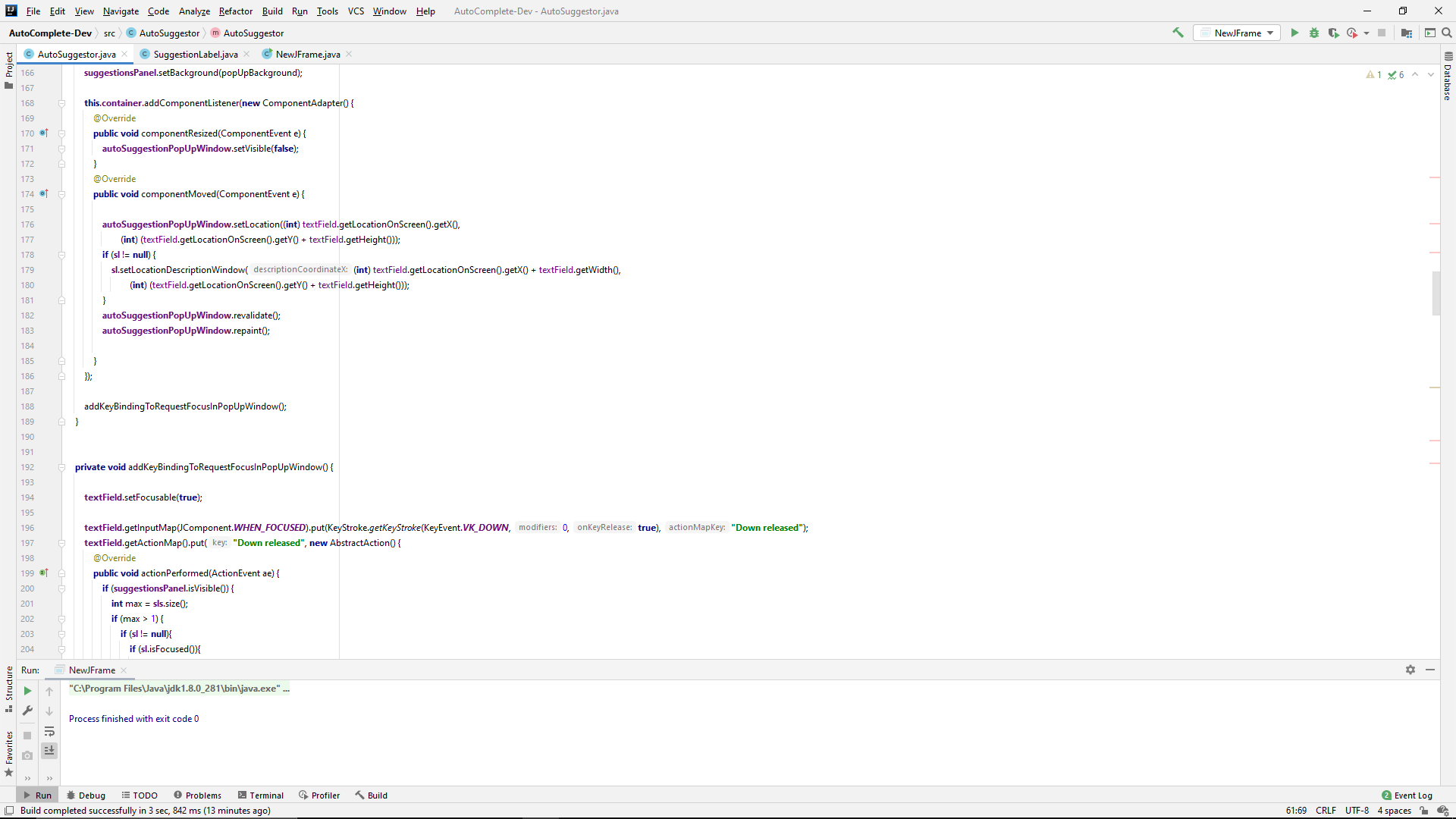
IntelliJ IDEA умеет глубоко понимать контекст программного кода, а также обладает очень умной функцией автодополнения. Она хорошо анализирует набираемый текст и выводит на экран список из возможных вариантов завершения слова, которые хочет написать программист.

Перечислим некоторые функции интеллектуального анализа кода, которыми обладает данная IDE:

* анализ ошибок в коде – эффективный инспектор кода сразу находит ошибки, которые совершил программист и, если возможно исправление силами IDE, то предлагает его через диалоговое окно;
* интеллектуальное завершение кода – IDE предлагает только те варианты названий классов, методов или переменных в автодополнении, которые подходят под текущее окружение (контекст). Благодаря данной функции программист гораздо меньше ошибается и быстрее пише;
* мониторинг дублирования – среда разработки, анализируя код, видит участки дублирующегося кода и соответствующим образом выделяет их, предлагая решение этой проблемы.

JetBrains придерживаются философии, что разработчика ничего не должно отвлекать от программирования. Поэтому, чтобы уменьшить количество факторов, мешающих программиста решать от стоящие перед ним задачи, создатели IDEA сделали очень минималистичный дизайн IDE: основную площадь экрана занимает текстовый редактор и совсем немного дополнительные инструменты, которые в данный момент не нужны разработчику, но к которым он с легкостью может обратиться, при необходимости (в то же время среда разработки обладает очень приятным внешним видом); большой упор сделан скорость работы IDE и быстрое использование функционала, который она предлагает.

* ориентация на пользователя – концепция минимума отвлекающих факторов привела к тому, что разработчику более необходимо отвлекаться на поиск документации благодаря всплывающим окнам при наведении, а также смотреть интересующую информацию об используемых классах;
* горячие клавиши – множество действий в среде IntelliJ IDEA можно выполнять, используя лишь горячие клавиши, а не перемещая постоянно руку с клавиатуры на мышь;
* встроенный отладчик – встроенный в IDE надежный отладчик, позволяет без лишних действий быстро находить и устранять проблемные места.



*Рисунок 2.1 – Внешний вид среды разработки IntelliJ IDEA*

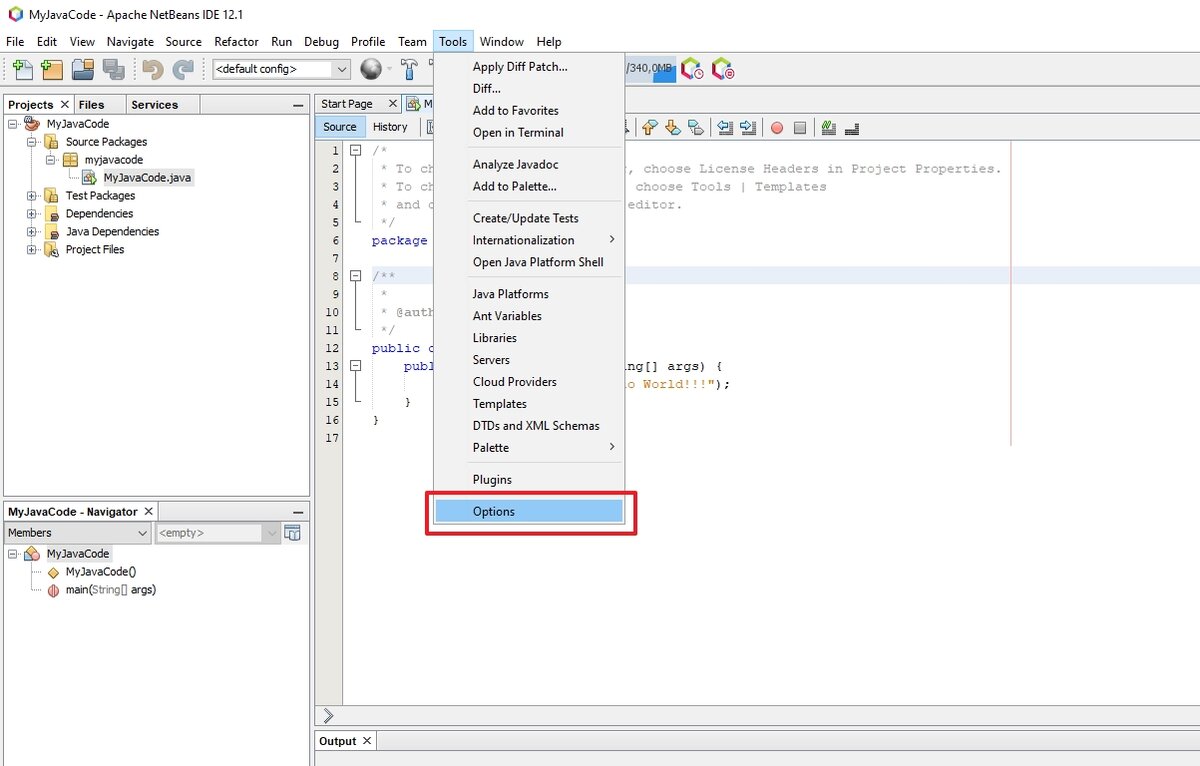
NetBeans

NetBeans – это среда разработки, созданная как студенческий проект, развивалась и поддерживалась Sun Microsystemsно поддерживается сообществом разработчиков под эгидой Apache Foundation.

Данная IDE может быть использована для разработки программного обеспечения на большом количестве языков программирования, в том числе популярные: Python, Java, С/С++, PHP, HTML5.

NetBeans включает в себя многие полезные функции, без которых современную IDE сложно представить: подсветка синтаксиса, автодополнение кода, понимание его контекста, отладка и профилирование. Поддерживается взаимодействие с базами данных. Благодаря плагинам, которые разрбатываются сообществом разработчиков NetBeans функционал и внешний вид данной среды разработки могут быть значительно улучшены.

В NetBeans можно разработывать приложения для платформ: Java EE, Java SE и Java ME и др. Java ME предназначена для разработки приложений для мобильных устройств.



*Рисунок 2.2 – Внешний вид среды разработки NetBeans*

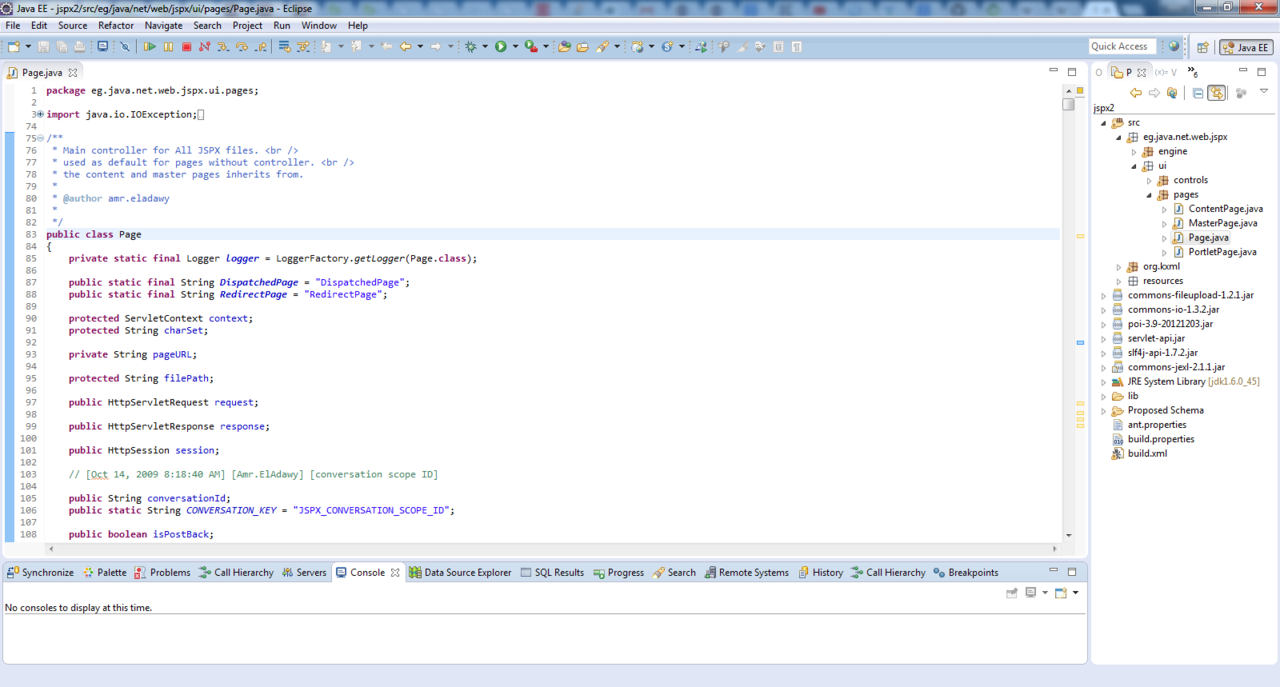
Можно выделить основные плюсы NetBeans:

* множество компонентов можно использовать сразу после установки данной среды разработки. Для работы с Java потребуется установить JDK необходимой версии, а при установке IDE можно выбрать необходимы языковые пакеты;
* бесплатная качественная IDE, существующая как open-source проект, поскольку Sun Microsystems опубликовала исходные коды IDE, а в данный момент среда разрабатывается сообществом;
* для разработки в команде имеет целый ряд инструментов, чтобы организовать эффективное взаимодействие между разработчиками. Это является весомым аргументов для малых компаний в пользу данной IDE, поскольку оная является бесплатной;
* имеет в своем арсенале мощный графический редактор Swing GUI designer, который значительно упрощает разработку пользовательского интерфейса с помощью библиотеки Swing благодаря оптимизации и автоматизации данного процесса;
* поддерживается большое количество языков программирования, что позволяет не устанавливать специальное программное обеспечения для написания программ на другом языке;
* поддержка Maven позволяет без лишних усилий добавлять в свой проект различные зависимости, не нарушая при этом работу приложения.

Eclipse

Eclipse Platform – среда разработки, которая распространяется полностью бесплатно и развивается сообществом Eclipse. Предоставляет как множество готовых инструментов «из-под коробки» для разработки программного обеспечения, так сотни плагинов, разработанных самими пользователями Eclipse, с помощью которых каждый разработчик может настроить IDE под свои нужды. Инструментарий в рамках Eclipse Platform является типовым в том смысле, что с его помощью проверяется применимость базовых возможностей платформы, иллюстрируются типовые ситуации их использования, поддерживаются разработка и сопровождение самой платформы Eclipse. Инструментарий является расширяемым, поскольку его функциональность доступна посредством документированных программных интерфейсов. Основными принципами разработки и развития Eclipse Platform, которым компания Eclipse Foundation следует до сих пор, являются:

* поддержка конструирования инструментов для разработки приложений;
* поддержка неограниченного числа поставщиков инструментальных средств, в том числе независимых разработчиков программного обеспечения;
* поддержка инструментов работы с произвольными типами контента, включая HTML, Java, C, JavaServer Pages, EJB, XML, GIF и др.;
* обеспечение «бесшовной» интеграции инструментов работы с различными типами контента разных производителей;
* поддержка сред разработки приложений с обычным и с графическим пользовательским интерфейсами;
* обеспечение возможности выполнять среду разработки на базе разных операционных систем (сегодня инструментарий на базе Eclipse реализован для GNU/Linux, HP-UX, IBM AIX, Sun Solaris, QNX, Mac OS X, Windows);
* учет в интересах развития платформы популярности языка Java при создании инструментария разработки.



*Рисунок 2.3 – Внешний вид среды разработки Eclipse*

Eclipse служит в первую очередь платформой для разработки расширений, чем он и завоевал популярность: любой разработчик может расширить Eclipse своими модулями. Уже существуют Java Development Tools (JDT), C/C++ Development Tools (CDT), разрабатываемые инженерами QNX совместно с IBM, и средства для языков Ada (GNATbench, Hibachi), COBOL, FORTRAN, PHP, X10 (X10DT) и пр. от различных разработчиков. Множество расширений дополняет среду Eclipse диспетчерами для работы с базами данных, серверами приложений и др.

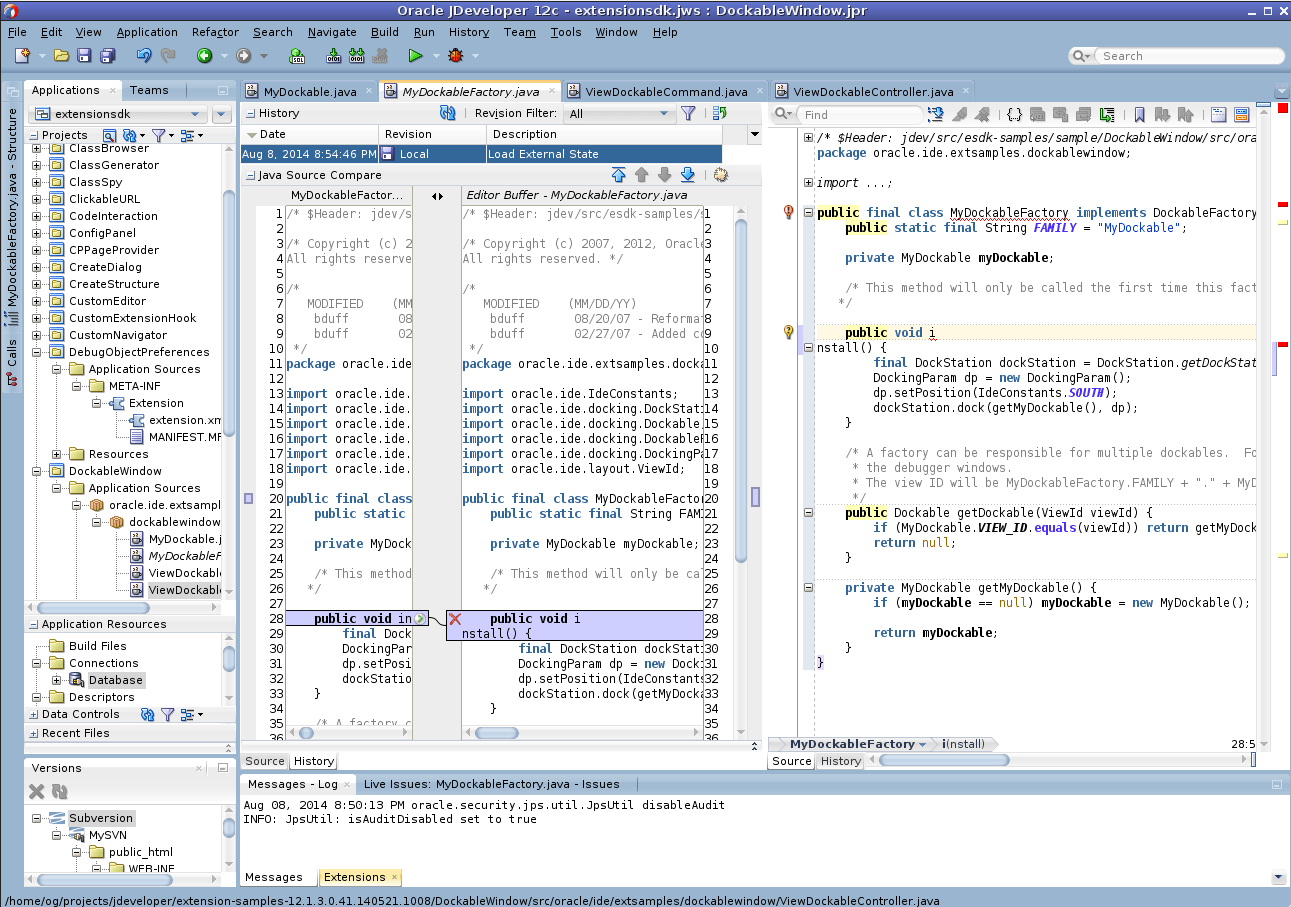
Eclipse JDT (Java Development Tools) - наиболее известный модуль, нацеленный на групповую разработку: среда интегрирована с системами управления версиями - CVS, GIT в основной поставке, для других систем (например, Subversion) существуют плагины. Также предлагает поддержку связи между IDE и системой управления задачами (ошибками). В основной поставке включена поддержка трекера ошибок Bugzilla, также имеется множество расширений для поддержки других трекеров (Trac, Jira и др.). В силу бесплатности и высокого качества, Eclipse во многих организациях является корпоративным стандартом для разработки приложений.

Eclipse написана на Java, потому является платформо-независимым продуктом, за исключением библиотеки SWT, которая разрабатывается для всех распространённых платформ (см. ниже). Библиотека SWT используется вместо стандартной для Java библиотеки Swing. Она полностью опирается на нижележащую платформу (операционную систему), что обеспечивает быстроту и натуральный внешний вид пользовательского интерфейса, но иногда вызывает на разных платформах проблемы совместимости и устойчивости приложений.

Oracle JDeveloper

JDeveloper — бесплатная Java IDE, предоставляемая корпорацией Oracle. Его можно использовать для разработки приложений на разных языках, кроме Java, таких как XML, HTML, SQL, PL / SQL, JavaScript, PHP и т.д. Некоторые из особенностей JDeveloper:

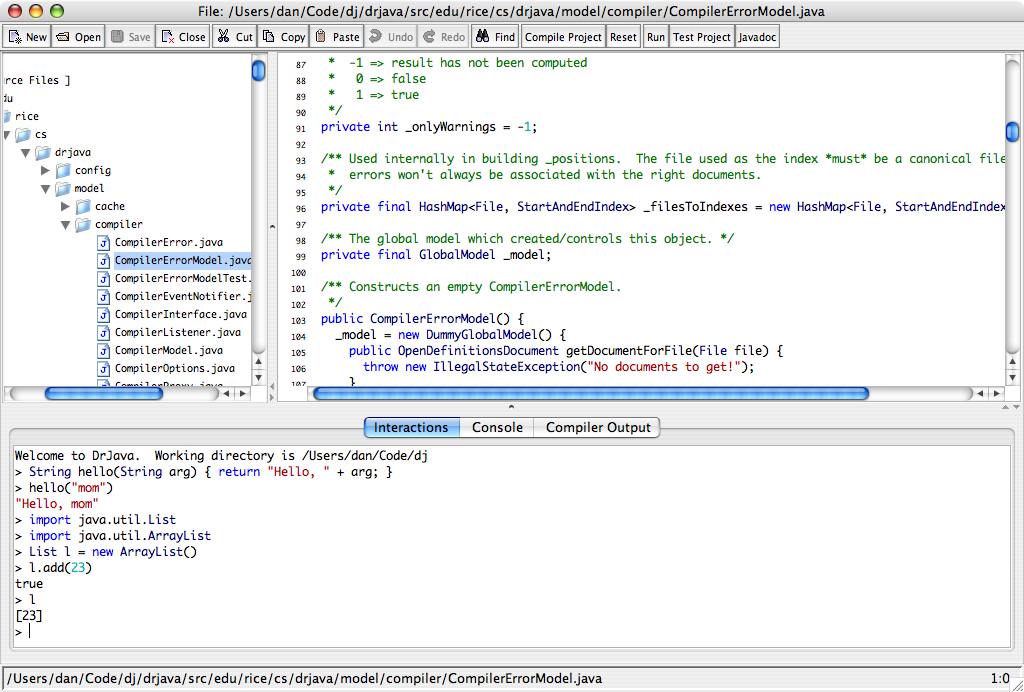
* Oracle JDeveloper объединяет функции разработки для Java, Интернета, мобильных устройств и веб-сервисов (REST & SOAP) и базу данных в единый инструмент разработки. У разных артефактов одно и то же структура проекта и опыт разработки, сокращая время обучения и упрощая процесс разработки составных приложений, использующих множество технологий. Кроме того, Oracle JDeveloper выступает в качестве среды разработки для различных компонентов. предложения Oracle Fusion Middleware, включая Oracle SOA Suite, Oracle WebCenter Портал и решения Oracle для бизнес-аналитики.
* Oracle JDeveloper охватывает полный жизненный цикл разработки от первоначального проектирования и анализа, на этапах кодирования и тестирования, вплоть до развертывания. Разработчики могут проектировать, генерировать и визуализировать свой код с помощью диаграмм UML, Java и баз данных. Продвинутое кодирование среда, а также декларативные и визуальные редакторы ускоряют разработку кода. Интегрированные функции тестирования, профилирования и аудита кода обеспечивают качество приложений. Интеграция инструменты управления версиями, сборки и развертывания позволяют упаковывать и развертывать приложения либо через IDE, либо через автоматизированные процессы.
* Oracle JDeveloper улучшает командную разработку за счет интеграции с множеством версий решения для управления, а также интеграция с Oracle Developer Cloud Service для полной контроль жизненного цикла и командное сотрудничество.



*Рисунок 2.4 – Внешний вид среды разработки Oracle JDeveloper*

JCreator

**JCreator** — интегрированная среда разработки, разработанная специально для платформы Java, **объединяющая в себе редактор, отладчик и компилятор**. Это одна из самых мощных и эффективных программ даже по сравнению с JDeveloper, JBuilder, Eclipse или Netbeans, так как она полостью написана на языке C++, поэтому потребляет меньше ресурсов.



*Рисунок 2.5 – Внешний вид среды разработки JCreator*

JCreator имеет три издания:

* Lite Edition (LE): [условно](https://en.wikipedia.org/wiki/Shareware)-бесплатная версия, которая стоит $35 после 30-дневной пробной версии.
* Pro Edition (Pro): [условно](https://en.wikipedia.org/wiki/Shareware)-бесплатная версия, которая стоит $89 после 30-дневной пробной версии.
* Lite-Pro Edition

JCreator доступен только в операционной системе Windows. Однако как LE, так и Pro версии JCreator работают на GNU/[Linux](https://en.wikipedia.org/wiki/Linux" \o "Линукс) (при использовании [Wine](https://en.wikipedia.org/wiki/Wine_(software)) ). До сих пор никаких версий GNU/Linux не планируется к немедленному выпуску, но новые компоненты будут построены для перекрестной совместимости в виду.

Набор функций версии Pro сопоставим с другими языковыми IDE по отношению к функциям управления проектами и редактирования, но не имеет расширенных функций, таких как автоматизированный [рефакторинг](https://en.wikipedia.org/wiki/Refactoring) , поддержка общих фреймворков и т. д., который можно найти в доминирующих Java IDE, таких как [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)) и [IntelliJ IDEA](https://en.wikipedia.org/wiki/IntelliJ_IDEA) . В бесплатной версии файла также отсутствуют некоторые функции , такие как [завершение кода](https://en.wikipedia.org/wiki/Code_completion), которые включены в другие бесплатные IDE. В отличие от доминирующих Java IDE сегодня, JCreator также не хватает уровня расширяемости через сторонние плагины, что является общим в популярных Java IDE.

В результате сравнительного анализа в качестве среды разработки для создания ПМ АОС выбрана IntelliJ IDEA, как наиболее удовлетворяющая поставленным требованиям.

## Реализация разработанных моделей и алгоритмов

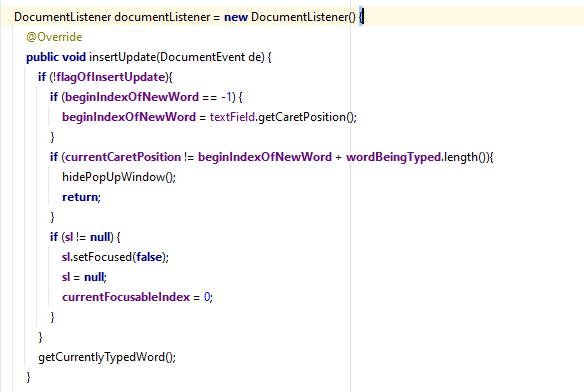
В результате декомпозиции алгоритма решения задачи, изображенного на рисунке 2.6, можно выделить следующие этапы: режим слушателя для регистрации события написания символов; проверка введенных символов на соответствие с началом слов из БД; запись слова в список вывода при соответствии; отображение автодополнения при наличии слов в списке; режим слушателя для регистрации нажатия клавиш вверх/вниз при появившемся автодополнении; показ описания слова, если оно подсвечено; режим слушателя для регистрации двойного клика ЛКМ по любому слову в автодополнении или регистрации нажатия клавиши Enter, если какое-либо слово в автодополнении подсвечено; вставка слова из списка вместо введенных символов.



*Рисунок 2.6 – Схема алгоритма ПМ АОС*

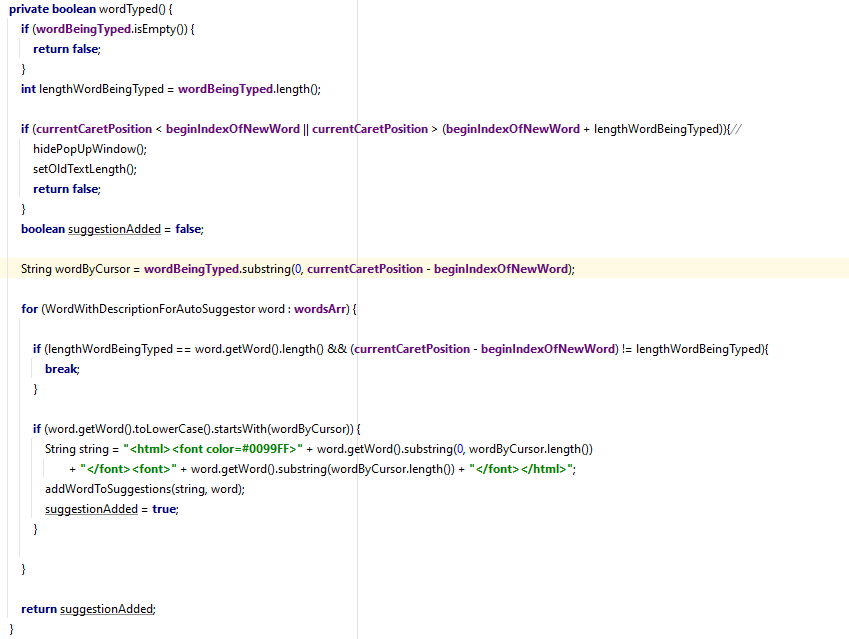
Запущенная программа работает в режиме слушателя, который реагирует на изменения в текстовом поле. Если введены символы, то начинается цикл проверки введенных символов на соответствие ключевых слов с их началом или всем словом целиком.

*Листинг 2.1 – Код слушателя введения новых символов*

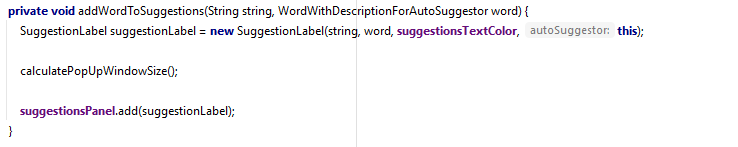


Если есть совпадения введенных символов с ключевыми словами, то они заносятся в список автодополнения.

*Листинг 2.2 – Метод для поиска совпадений*

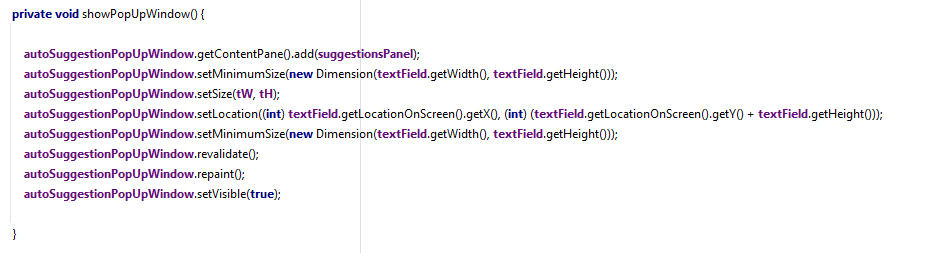


*Листинг 2.3 – Метод занесения совпадающих слов в список автодополнения*



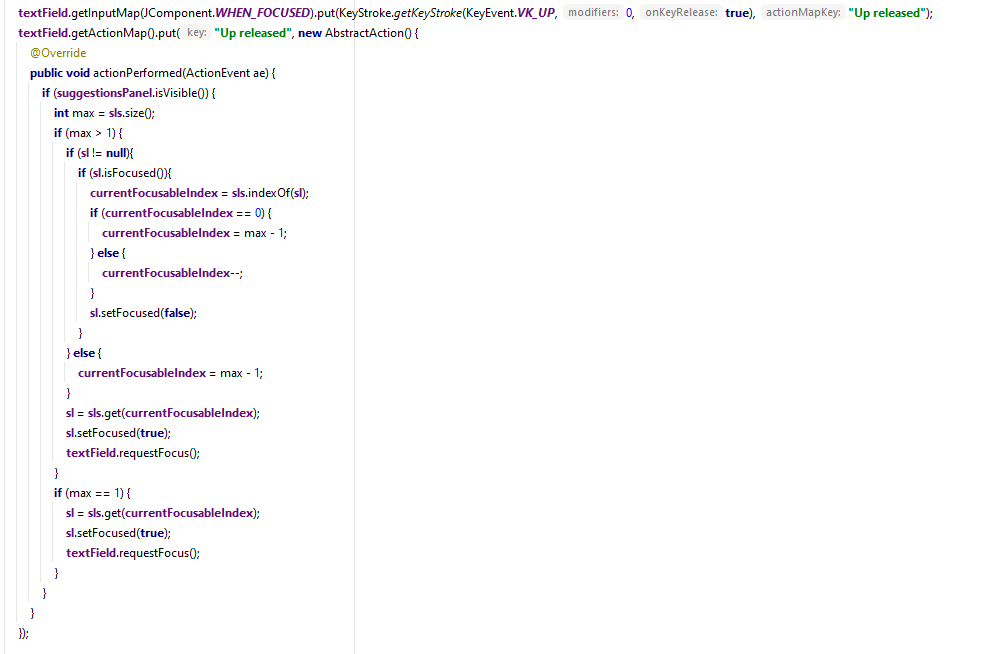
Список автодополнения отображает слова, если в цикле в него были занесены слова.

*Листинг 2.4 – Метод отображения списка слов в автодополнении*

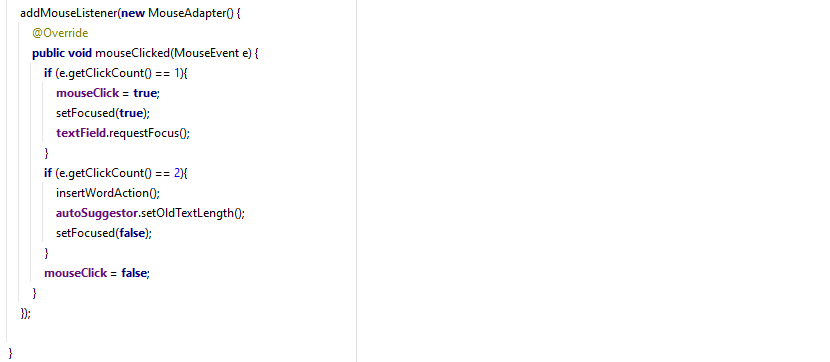


При отображаемом автодополнении фиксируются события нажатия клавиш вверх/вниз или нажатие ЛКМ.

*Листинг 2.5 – Обработка события нажатия клавиши вверх*



*Листинг 2.6 – Обработка события нажатия ЛКМ*



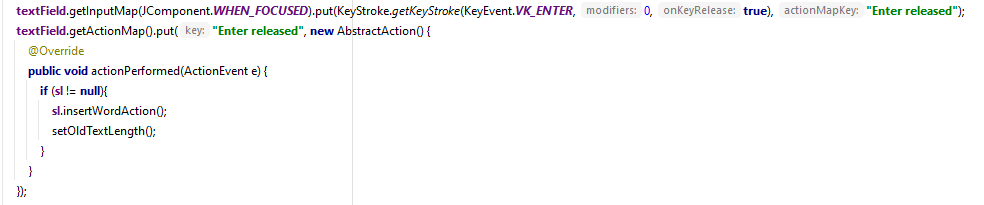
Если такое событие произошло, то слово, для которого произошло это событие начинает подсвечиваться, а рядом со списком автодополнения дополнительно выводится документация с назначением слова.

*Листинг 2.7 – Метод отображения описания слова*

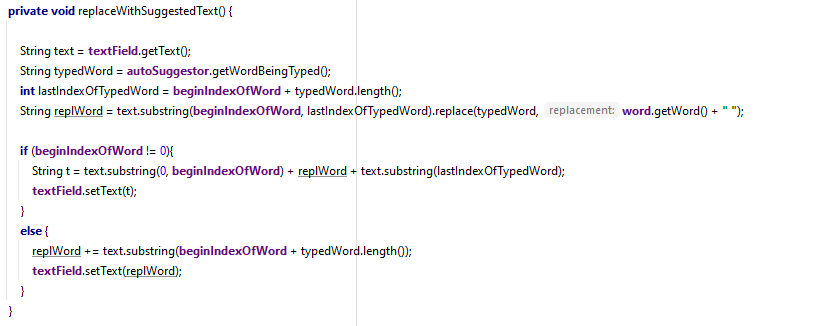


Также при отображаемом автодополнении фиксируются события нажатия клавиши Enter и двойного нажатия клавиши ЛКМ. Если какое-либо слово подсвечено и нажата клавиша Enter, или произошло двойное нажатие клавиши ЛКМ на какое-либо слово (не важно подсвеченное или нет), то слово вставляется вместо введенных символов.

*Листинг 2.8 – Обработка события нажатия клавиши Enter*



*Листинг 2.9 – Метод вставки подсвеченного слова вместо введенных символов*



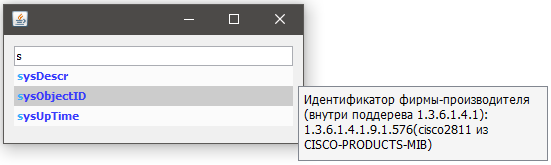
## Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность физиологических, технических, лексических и программных решений, примененных для обеспечения взаимодействия человека с компьютером. Основной задачей ПИ является обеспечение непосредственного взаимодействия пользователя и программного обеспечения, установленного на ЭВМ.

За основу внешнего вида ПМ АОС взяты графические решения в средах разработки IntelliJ IDEA и NetBeans.

Пользователь взаимодействует с программой через графический интерфейс, вводя символы в текстовое поле.

Если автодополнение отобразилось, пользователь может выбрать из предоставленного списка любое слово для вставки в текстовое поле с помощью двойного клика ЛКМ или используя стрелки «вверх/вниз» на клавиатуре, выделить необходимое слово и вставить его в текстовое поле, нажав Enter. При этом цвет фона подсвеченного слова и цвет начальных символов слов в списке, совпадающих с веденными символами в текстовом поле, изменятся.



*Рисунок 2.6 – Внешний вид автодополнения с описанием*

## Вывод по конструкторскому разделу

В конструкторском разделе рассмотрены различные языки программирования и среды разработки, приведены их плюсы и минусы, чтобы выбрать наиболее подходящие для решения поставленной задачи.

Описан алгоритм работы ПМ АОС, а также разработан пользовательский интерфейс.

# Технологический раздел

## Средства и методы отладки

Отладка – это процесс, проводящийся для исправления уже известных ошибок и локализации еще не обнаруженных при тестировании ошибок [23].

Локализацией – это мероприятия по обнаружению местоположения неверно работающего участка кода программы, который вызывает неверную работу всего или приложения. После нахождения такого участка, выясняется причина его неверной работы: для этого ищут методы, переменные или фрагменты программного кода, которые обрабатывают или хранят ошибочную для работы приложения информацию. Ошибки могут быть трудно локализуемыми или, наоборот, легко локализуемыми.

Классификация ошибки, в зависимости от этапа обработки кода, во время которого они возникают [44]:

* синтаксические ошибки – ошибки, которые сообщаются при компилировании программы. Они обозначают наличие синтаксических ошибок в программе, поскольку язык программирования, обычно, строго формализован и не допускает применение каких-либо необъявленных переменных, или переменных объявленных вне зоны видимости в текущем участке кода, или неверно написанных ключевых слов и идентификаторов языка. Компилятор при обнаружении таких ошибок сообщает их местоположение и описание;
* ошибки компоновки – ошибки, из-за которых компоновщик не в состоянии объединить модули программы между собой;
* ошибки выполнения – ошибки, возникшие во время работы программы: зависание или неверная работа приложения, системные сообщения от операционной системы и т.д.

Поскольку ошибки могут появляться по самым разным причинам, то и способов обнаружения ошибок существует довольно много. Выделяют несколько способов для определения наличия ошибок:

* при помощи системы прерываний процессора, благодаря которым выводятся ошибки о неверном применении машинных команд, их ошибочному результату/статусу;
* зависание компьютера;
* при помощи средств операционной системы, которая следит за корректностью подаваемых пользователем командами. Если команда не может быть выполнена по какой-либо причине, выводится ошибка;
* неверный результат работы программы.

Методы отладки условно разделяются статические и динамические:

* статические методы – методы, для использования которых не требуется запуск программного кода. Например, просмотр программистом написанного кода для того, чтобы найти проверить написанный алгоритм, является статическим методом, точно так же таким методом является компиляция, поскольку она не запускает код программы на исполнение. Главным недостатком этих методов является большие временные затраты, которые требуются для их исполнения несмотря на то, что они практически не требуют ресурсов компьютера.
* динамические методы – методы, которые выполняются с применением ресурсов компьютера: запуск кода для выполнения, использование точек останова, трассировка и т.д. В большинстве случаев такие методы используют при написании кода в среде разработки, поскольку в них встроены специальные инструменты для отладки. Динамические методы требуют в разы меньше времени, чтобы проводить отладку приложения, из-за своей наглядности, последовательности и простоте.

## Отладка ПМ АОС

При разработке ПМ АОС использовались динамические виды отладки:

* отладка с использованием точек останова;
* трассировка;
* логирование.

Отладка с использованием точек останова подразумевает отметку строк кода специальным образом, чтобы при запуске отладчика выполнение программы остановилось, когда будет достигнута точка останова. При достижении точки выполнение программы останавливается и управление передается отладчику среды программирования***.*** Точки устанавливаются в тех местах исходного кода программы, где нужно посмотреть состояние переменных или стек вызова функций. Возможность указывать места остановки программы существенно ускоряет процесс отладки, так как позволяет пропускать уже отлаженные участки кода. Пример точки останова показан на Рисунок 3.1.

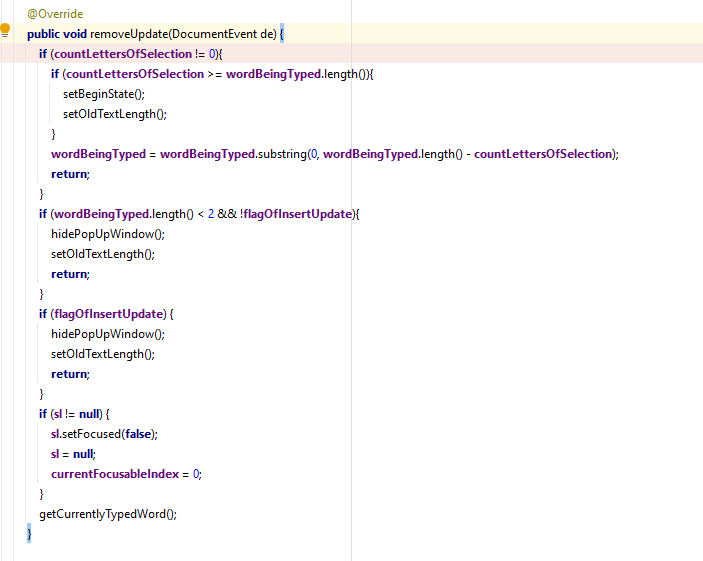


Рисунок 3.1 –Точка останова на 83 строчке кода

Отладка методом трассировки подразумевает построчное выполнение программы: происходит последовательное выполнение каждого оператора. Данный метод позволяет проанализировать шаг за шагом изменение состояния переменных и отследить причины их изменения, тем самым можно быстро понять, почему программа работает неправильно.

Использование трассировки при помощи встроенного отладчика IntelliJ IDEA показано на рисунке 3.2.

При использовании встроенного отладчика при построчном выполнении кода можно использовать следующие команды:

* step Into – программа выполнит текущую строку с заходом в подпрограмму;
* step Over – выполнение текущей строки без захода в подпрограмму;
* force Step Into – программа выполнит текущую строчку кода и курсор перейдет на следующую игнорируя все возможные ограничения;
* step Out – выход из подпрограммы;
* resume program – переход к следующей точке останова.

Также можно остановить отладку, перезапустить отладку, во время выполнения отладки можно переместиться к курсору и т.д.

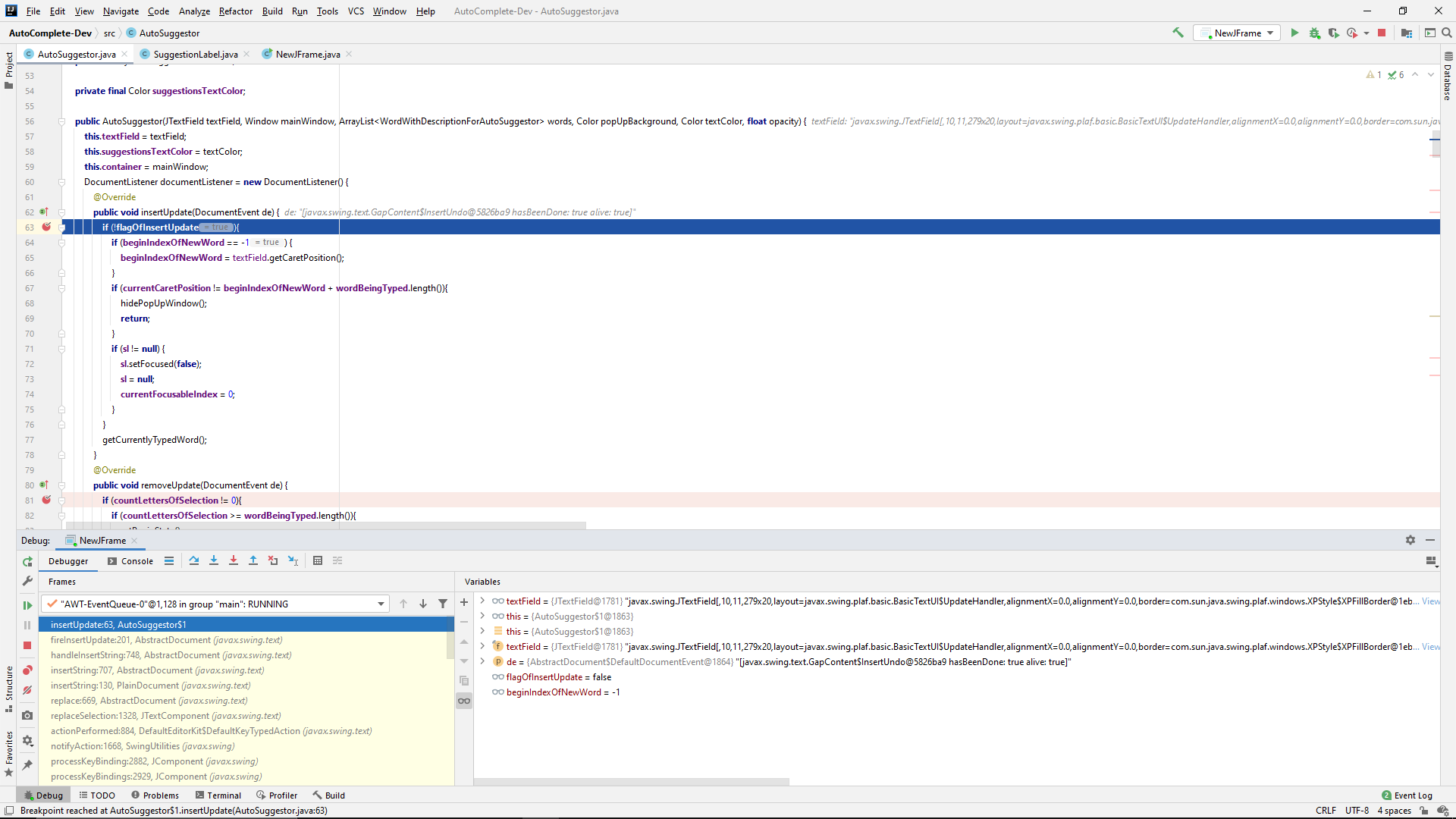


Рисунок 3.2 –Трассировка с помощью встроенного отладчика IntelliJ IDEA

Отладка с помощью логирования подразумевает вывод информации в консоль или в файл в процессе выполнения программы. Отладочной информацией могут служить следующие данные: входные и выходные аргументы функций, промежуточные значения локальных и глобальных переменных, состояние стека и памяти, какие-либо внешние источники входных данных.

Инструменты для отладки, представленные в IntelliJ IDEA, и вышеперечисленные методы отладки позволили существенно сократить время разработки ПМ АОС, а также повысить его качество.

## Анализ методов и средств тестирования

Тестирование программного обеспечения – это метод проверки соответствия фактического программного продукта ожидаемым требованиям, который также необходим, чтобы убедиться, что продукт не содержит дефектов. Подразумевает выполнение предварительно определенных тест-кейсов с использованием ручных или автоматизированных инструментов для оценки одного или нескольких интересующих свойств. Целью тестирования ПО является выявление ошибок, пробелов или отсутствующих требований, заданных на этапе проектирования продукта.

Тест или тест-кейс – это искусственно созданная ситуация, имеющая определенные входные параметры и начальные условия, а также описание ожидаемого результата, соответствующего некоторому требованию. Тест-кейс состоит из следующих атрибутов:

* номер – числовое значение, позволяющие однозначно идентифицировать тест-кейс. Используется при формировании отчетов по результатам тестирования для ссылок на тесты;
* название – краткое описание сути тест-кейса;
* предварительные шаги – описание последовательности действий, которые должны быть выполнены перед началом проверки. Допускается отсутствие данной секции;
* шаги – последовательность действий, которую необходимо выполнить для осуществления самой проверки;
* ожидаемый результат – результат, который должен быть получен на предыдущем шаге для успешного прохождения теста;

Тестирование можно классифицировать по самым разным критериям:

* по статусу работы приложения:
  1. Статическое тестирование – тестирование, при котором проверяется теоретическая работоспособность программы без запуска кода. Во время такого тестирования проверяется архитектура приложения, структура данных, алгоритм программы; проверяется правильность конфигурирования баз данных и их «связывания» с исходным кодом; анализу подвергается также сам код программы на предмет наличия неверного использования переменных, обращения к функциям и т.п.; оценивается визуальная составляющая приложения (пользовательский интерфейс); проверяются различные настройки среды разработки и программного окружения;
  2. Динамическое тестирование – тестирование с запуском кода на исполнение. Запускаться на исполнение может как код всего приложения целиком (системное тестирование), так и код нескольких взаимосвязанных частей (интеграционное тестирование), отдельных частей (модульное или компонентное тестирование) и даже отдельные участки кода.
* по доступу к коду и архитектуре приложения:
  1. Метод белого ящика – тестирование, проводимое с доступом к исходному коду программы и его внутреннему строению. При этом тестировщик хорошо понимает код и логику работы программы для проведения качественного тестирования;
  2. Метод чёрного ящика – метод тестирования, при котором проверяется работа приложения только с помощью пользовательского или программного интерфейса. При этом не обязательно, чтобы у тестировщика не было доступа к исходному коду программы и его структуре;
  3. Метод серого ящика – метод тестирования, при котором используются как метод черного ящика, так и метод белого ящика, потому что у тестировщика нет доступа к как-либо части исходного кода программы, но есть доступ к другой части.
* по степени автоматизации:
  1. Ручное тестирование – тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации;
  2. Автоматизированное тестирование – совокупность различных способов и средств тестирования, при котором возможно проверять работоспособность приложения без человека. При этом автоматизацией тестирования занимается человек, и он же занимается составлением тест-кейсов, настройкой программы тестирования, подготовкой специальных данных, оценкой тестирования.
* по уровню детализации приложения:
  1. Модульное (Unit) тестирование – тестирование, предназначенное, чтобы проверить отдельные модули исходного кода, которые могут выступать в роли отдельной программы, и результат которых практически не зависит от других модулей программы. В данном виде тестирования проверяются классы, методы классов, функции – все, что может выполняться вне зависимости от других частей программы;
  2. Интеграционное тестирование – тестирование, которое проверяет часть модулей программы как единое целое – их взаимодействие между собой. Однако перед проведением этого тестирования необходимо провести модульное тестирование, чтобы быть уверенным в работоспособности отдельных частей кода;
  3. Системное тестирование – тестирование, которое проводится для тестирования всех компонентов программы, протестированных с помощью двух предыдущих методов, и собранных в единый программный модуль или программу. Необходимо, в первую очередь, для выявления ошибок между отдельными частями кода, а также для проверки работоспособности приложения в различных условиях эксплуатации: разные ОС, программное окружение и т.д. При этом используются другие методы тестирования для полноценной проверки качества приложения.
* по степени важности тестируемых функций:
  1. Дымовое тестирование – основное тестирование, которое будет проверять только важнейшую функциональность, без которой дальнейшее использование программного обеспечения не имеет под собой никаких оснований;
  2. Тестирование критического пути – вид тестирования, который проверяет функционирование приложения в штатном режиме, имитируя ситуации, возникающие при обычной деятельности пользователя;
  3. Расширенное тестирование – вид тестирования, предназначенный для проверки всех зафиксированных в техническом задании требований. В данном случае ведется сортировка требований по степени важности, и, по возможности, также будет проводиться тестирование самых незначительных по приоритету требований.
* по принципам работы с приложением:
  1. Позитивное тестирование – тестирование, при котором все манипуляции с приложением проводятся только по предполагаемым случаям использования без попытки специально «сломать» программу. Предназначено ради проверки работоспособности приложения при абсолютно верных действиях со стороны пользователя;
  2. Негативное тестирование – тестирование, предназначенное для проверки устойчивости приложения в случаях, когда пользователь совершает операции, способные привести к ошибке в работе программы.

## Тестирование ПМ АОС

Для тестирования ПМ АОС выбран метод «черного ящика», как наиболее подходящий в связи со спецификой разрабатываемого ПМ, с использованием негативных и позитивных сценариев. Тест-кейсы составлялись на основе технического задания. Сначала проводилось позитивное тестирование, а затем негативное. Все тест-кейсы позитивного и негативного сценариев записывались в таблицы позитивного и негативного тестирования соответственно.

Примеры позитивных тест-кейсов представлены в таблице 3.1.

Таблица 3.1

Позитивные тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название тест-кейса | Описание | Ожидаемый результат | Вывод |
| 1 | Вывод корректного списка | 1. Запуск программы 2. Ввод символов в текстовое поле, с которых начинается как минимум 2 слова | Вывод автодополнения для введенных символов с корректным списком с слов | Тест пройден |
| 2 | Обновление списка при добавлении символов | 1. Запуск программы 2. Ввод тех символов, для которых отобразится список с минимум 2 словами 3. Добавление символов, совпадающих с символами первого слова из списка | Автодополнении отображает только те слова, начало которых совпадает с введенными символами | Тест пройден |

Продолжение таблицы 3.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Обновление списка при удалении символов | 1. Запуск программы 2. Ввод комбинации символов, для которой отобразится список минимум с 2 словами 3. Продолжение комбинации символов, пока количество слов в списке не уменьшится, либо автодополнение не пропадет вовсе 4. Удаление введенных символов по одному | Автодополнение обновляется с добавлением в него совпадающих слов, либо появляется снова, если оно не было отображено | Тест пройден |
| 4 | Вставка скопированных символов | 1. Запуск программы 2. Копирование комбинации символов, с которой начинается хотя бы 1 слово в БД 3. Вставка скопированной комбинации с помощью Ctrl+V или через ПКМ-Вставить | Вывод автодополнения как минимум с 1 словом в списке, которое начинается со вставленной комбинации символов | Тест пройден |
| 5 | Подсвечивание слова при нажатии клавиши «вниз» | 1. Запуск программы 2. Ввод комбинации символов, с которой начинается как минимум 1 слово 3. При появившемся автодополнении нажимаем клавишу вниз | В автодополнении цвет фона первого слова изменился, фон остальных слов (если в списке больше 1 слова) не изменился | Тест пройден |

Продолжение таблицы 3.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Подсвечивание слова при нажатии клавиши «вверх» | 1. Запуск программы 2. Ввод комбинации символов, с которой начинается как минимум 2 слова 3. При появившемся автодополнении нажимаем клавишу «вверх» | В автодополнении цвет фона последнего слова изменился, фон остальных слов не изменился | Тест пройден |

Примеры негативных тест-кейсов представлены в таблице 3.2.

Таблица 3.2

Негативные тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название тест-кейса | Описание | Ожидаемый результат | Вывод |
| 1 | Вставка скопированных символов | 1. Запуск программы 2. Копирование комбинации символов, с которой начинается хотя бы 1 слово в БД 3. Вставка скопированной комбинации с помощью Ctrl+V или через ПКМ-Вставить | Вывод автодополнения как минимум с 1 словом в списке, которое начинается со вставленной комбинации символов | Тест пройден |

Продолжение таблицы 3.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Написание слова при смещенном курсоре | 1. Запуск программы 2. Написание комбинации символов, с которой начинается как минимум 2 слова 3. Перемещение курсора клавишей «вправо» или ПКМ как минимум на 1 символ 4. Продолжение комбинации символов, совпадающих с каким-либо словом | Исчезновение автодополнения, без ошибок в консоли или зависаний программы | Тест провален:  Возникает ошибка в консоли |
| 4 | Проверка ввода при подсвечивании слова с помощью ЛКМ | 1. Запуск программы 2. Написание комбинации символов, с которой начинается как минимум 2 слова 3. Нажимаем на какое-либо слово в списке с помощью ЛКМ 4. Пытаемся продолжить писать комбинацию символов | В текстовом поле продолжается ввод символов | Тест провален:  пропадает фокус с текстового поля при нажатии на слово ЛКМ |

Продолжение таблицы 3.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Ввод какого-либо слова целиком + пробел | 1. Запуск программы 2. Ввод комбинации символов, с которой начинается как минимум 1 слово 3. Вводим слово целиком, которое появилось в автодополнении и нажимаем пробел 4. Снова вводим комбинацию символов, с которой начинается как минимум 1 слово | Появление автодополнения во время выполнения 4 пункта | Тест провален: автодополнение при вводе символов другого слова после пробела не появляется |
| 6 | Проверка на подсветку 2-х слов | 1. Запуск программы 2. Ввод комбинации символов, с которой начинается как минимум 2 слова 3. При появившемся автодополнении нажимаем клавишу «вверх», затем нажимаем клавишу «вниз» | В автодополнении сначала изменился цвет фона последнего слова, цвет фона остальных слов не изменился, затем изменился цвет фона первого слова, а цвет фона последнего слова стал белым | Тест пройден |

В таблицах представлены некоторые из тест-кейсов, проводившихся во время тестирования. Все проваленные тесты локализировались после тестирования, а затем комплексно устранялась причина их возникновения.

После устранения ошибок, снова проводилось тестирование, если тест-кейсы оказывались пройденными, то это означало, что отладка прошла успешно. Если какй-либо тест-кейс оказывался проваленным, цикл локализации ошибки и последующего устранения повторялся.

Все проваленные тест-кейсы, которые использовались для проверки работоспособности исправлены. По итогу тестирования и отладки все тест-кейсы успешно пройдены.

## Вывод по технологическому разделу

В технологическом разделе проведен анализ средств и методов отладки, частично описан встроенный отладчик IntelliJ IDEA, описаны использовавшиеся методы для отладки ПМ АОС.

Проведен анализ методов и средств тестирования. Продемонстрирован процесс составления тест-кейсов.

# Заключение

В результате выполнения выпускной квалификационной работы разработана рабочая версия программного модуля автодополнения и описания слов.

Поставленные цели и задачи выполнены. В рамках выпускной квалификационной работы решены следующие задачи:

* исследована предметная область;
* проведен сравнительный анализ существующих аналогов;
* выбраны язык и среда программирования;
* разработана схема данных ПМ АОС;
* разработана схема алгоритмов ПМ АОС;
* разработан пользовательский интерфейс ПМ АОС;
* осуществлена программная реализация ПМ АОС;
* проведены отладка и тестирование ПМ АОС;
* разработано руководство оператора ПМ АОС.

ПМ АОС предназначен для ускорения написания ключевых слов в программе S-Terra KP Console, Программный модуль предоставляет возможность оперативно просматривать документацию. Разработка модуля существенно сократит время, необходимое для правильной настройки программы.

# Список литературы

1. Л.Г. Гагарина, Р.А. Касимов, Д.Г. Коваленко, Е.Л. Федотова, Чжо Зо Е, Б.В. Черников Методические указания по подготовке выпускной квалификационной работы по направлению подготовки бакалавров 09.03.04 «Программная инженерия»/ Под ред. док. тех. наук Б.В.Черникова. – М.: МИЭТ, 2016
2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. Введ. - М., Стандартинформ, 2005 - 24 с.
3. ГОСТ Р 7.0.12-2011. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. Введ. – М.: Стандартинформ, 2011 – 27 с.
4. ГОСТ Р 7.0.100-2018. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. Введ – М.: Стандартинформ, 2018 – 65 с.
5. ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. Введ – М.: Стандартинформ, 2017 – 27 с.
6. ГОСТ 19.505-79. Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению. Введ – М.: Стандартинформ, 2010 – 3 с.
7. Гагарина Л.Г., Киселёв Д.В., Федотова Е.Л. Разработка и эксплуатация автоматизированных информационных систем: учеб. Пособие / Под ред. проф. Л.Г. Гагариной. - М.: ИД «ФОРУМ»: ИНФРА-М, 2012.
8. Tam, Cynthia Evaluating the Benefits of Displaying Word Prediction Lists on a Personal Digital Assistant at the Keyboard Leve / Cynthia Tam, David Wells. — Текст : непосредственный // Assistive Technology. — 2009. — Volume 21, Issue 3. — С. 105–114.
9. Anson, Denis The Effects of Word Completion and Word Prediction on Typing Rates Using On-Screen Keyboards / Denis Anson, Penni Moist, Mary Przywara, Heather Wells, Heather Saylor, Hantz Maxime — Текст : непосредственный // Assistive Technology. — 2006. — Volume 18. — С. 146-154.
10. Trnka, K., Yarrington, J.M. & McCoy, K.F The Effects of Word Prediction on Communication Rate for AAC / Trnka Keith, Yarrington Debra, McCaw John — Текст : непосредственный // Proceedings of NAACL HLT 2007. — Rochester : Companion Volume, 2007. — С. 173-176.
11. Sojka, Petr Notes on Compound Word Hyphenation in TEX / Petr Sojka. — Текст : непосредственный // TUGboat. — 1995. —Volume 16, No.3 - Proceedings of the 1995 Annual Meeting. — С. 290-296.
12. Beukelman, D. R. Augmentative and Alternative Communication: Supporting Children and Adults with Complex Communication Needs / D. R. Beukelman, P. Mirenda. — 3rd Ed. — Baltimore : Brookes Publishing, 2008. — 77 c. — Текст : непосредственный.
13. Witten, I. H. The reactive keyboard / I. H. Witten, J. J. Darragh. — 3rd Ed. — Cambridge : Cambridge University Press, 1992. — 43–44 c. — Текст : непосредственный.
14. Jelinek, F. Self-Organized Language Modeling for Speech Recognition / F. Jelinek. — Текст : непосредственный // Readings in Speech Recognition. — San Mateo : Morgan Kaufmann Publishers, Inc., 1990. — С. 450.
15. Oster, Jan Communication, defamation and liability of intermediaries / Jan Oster. — Текст : непосредственный // Legal Studies. — 2015. — № Volume 35, Issue 2. — С. 348-368.
16. Longuet-Higgins, H. C. The adaptive memorization of sequences / H. C. Longuet-Higgins, A. Ortony. — Текст : непосредственный // Machine Intelligence Volume 3. — Edinburgh : Edinburgh University Press, 1968. — С. 281-325.
17. Бадд, Тимоти Объектно-ориентированное программирование / Тимоти Бадд. — Санкт-Петербург : Питер, 1997. — 464 c. — Текст : непосредственный.
18. Г. Буч, Д. Рамбо, А. Джекобсон. Язык UML. Руководство пользователя. 2-е издание. 2007 - 496 с.
19. Программирование на языке Си. Учебное пособие по дисциплине «Информатика» / Моск. гос. инт электроники и математики; Сост. В.Г. Кулаков. М., 2010, 44 с.
20. Столяров А.В. Введение в язык Си++: Учебное пособие. – 5-е изд., испр. и доп. – Москва: МАКС Пресс, 2020. – 156 с.
21. Кестер У. Аналогово-цифровое преобразование: Под ред. У. Кестера М.: Техносфера, 2014. 1016 c.
22. ЛещенкоА.В. Обзор и практическое применение алгоритмов нечеткого поиска // Сборник научных трудов НГТУ. – 2018. – No 3–4 (93). – С. 59–69
23. Петров, А. М. Применение отладки при разработке программного обеспечения / А. М. Петров. — Текст : непосредственный
24. Куликов, С. Тестирование программного обеспечения. Базовый курс / С. Куликов. — 3-е изд. — : , 2021. — 298 c. — Текст : непосредственный.
25. McCulloch, Gretchen Autocomplete Presents the Best Version of You / Gretchen McCulloch. — Текст : электронный // wired : [сайт]. — URL: https://www.wired.com/story/autocomplete-presents-the-best-version-of-you/
26. Package org.jdesktop.swingx . — Текст : электронный // pirlwww.lpl.arizona.edu : [сайт]. — URL: https://pirlwww.lpl.arizona.edu/resources/guide/software/SwingX/org/jdesktop/swingx/package-summary.html
27. Autocomplete by Uberi. — Текст : электронный // github : [сайт]. — URL: https://github.com/Uberi/Autocomplete
28. Glazed Lists - List transformations in Java. — Текст : электронный // glazedlists : [сайт]. — URL: http://www.glazedlists.com/Home
29. TypingAid. — Текст : электронный // github : [сайт]. — URL: https://github.com/ManiacDC/TypingAid
30. Вендров, А. М. Современные технологии создания программного обеспечения. Обзор / А. М. Вендров. — Текст : электронный // citforum : [сайт]. — URL: http://citforum.ru/programming/application/program/1.shtml
31. Java Documentation. — Текст : электронный // oracle : [сайт]. — URL: https://docs.oracle.com/en/java/
32. C# documentation. — Текст : электронный // microsoft : [сайт]. — URL: https://docs.microsoft.com/en-us/dotnet/csharp/
33. Язык программирования C. Лекция 1: Аннотация и введение. — Текст : электронный // intuit : [сайт]. — URL: https://intuit.ru/studies/courses/97/97/lecture/2859
34. История языка C++ . — Текст : электронный // blogspot : [сайт]. — URL: http://cpp-cpp.blogspot.com/2014/01/c.html
35. История языка программирования C#. — Текст : электронный // interestprograms : [сайт]. — URL: http://www.interestprograms.ru/article-history-csharp
36. Язык программирования C#: краткая история, возможности и перспективы. — Текст : электронный // timeweb : [сайт]. — URL: https://timeweb.com/ru/community/articles/chto-takoe-csharp
37. Java. — Текст : электронный // progopedia : [сайт]. — URL: http://progopedia.ru/language/java/
38. История появления Java. — Текст : электронный // interface : [сайт]. — URL: http://www.interface.ru/home.asp?artId=1549
39. Язык программирования Python. Лекция 1: Введение в программирование на языке Python. — Текст : электронный // intuit : [сайт]. — URL: https://intuit.ru/studies/courses/49/49/lecture/27058
40. Why IntelliJ IDEA. — Текст : электронный // jetbrains : [сайт]. — URL: https://www.jetbrains.com/idea/
41. Apache NetBeans 12.4. — Текст : электронный // apache : [сайт]. — URL: http://netbeans.apache.org/kb/docs/contributing.html
42. Eclipse documentation - Current Release. — Текст : электронный // eclipse : [сайт]. — URL: https://help.eclipse.org/2021-03/index.jsp
43. Productive Java-based Application Development. — Текст : электронный // oracle : [сайт] — URL: https://www.oracle.com/application-development/technologies/jdeveloper.html
44. JCreator. — Текст : электронный // jcreator : [сайт]. — URL: http://www.jcreator.com/about.htm
45. Introduction. — Текст : электронный // intellicomplete : [сайт]. — URL: http://www.intellicomplete.com/
46. A free (gpl) Software utilities program for Windows. — Текст : электронный // softonic : [сайт]. — URL: https://letmetype.en.softonic.com/
47. Typing Assistant - New generation of word prediction software. — Текст : электронный // prlog : [сайт]. — URL: https://www.prlog.org/10519217-typing-assistant-new-generation-of-word-prediction-software.html
48. What is Word Q Speak Q. — Текст : электронный // lgfl : [сайт]. — URL: http://wordqspeakq.lgfl.net/About/What
49. Отладка программного обеспечения. Классификация ошибок, методы отладки программного обеспечения: ручного тестирования, индукции, дедукции, обратного прослеживания — Текст : электронный // skarlupka : [сайт]. — URL: https://skarlupka.ru/articles.php?id=33
50. Smartype. — Текст : электронный // mparticle : [сайт]. — URL: https://www.mparticle.com/platform/detail/smartype
51. About S-Terra CSP. — Текст : электронный // s-terra : [сайт]. — URL: https://www.s-terra.ru/company/about-s-terra-csp/
52. Python 3.9.5 documentation. — Текст : электронный // python : [сайт]. — URL: https://www.python.org/3/
53. TIOBE Index for June 2021. — Текст : электронный // tiobe : [сайт]. — URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 07.06.2021).
54. Interactive: The Top Programming Languages. — Текст : электронный // ieee : [сайт]. — URL: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020 (дата обращения: 07.06.2021).

ПРИЛОЖЕНИЕ 1

ПРОГРАММНЫЙ МОДУЛЬ АВТОДОПОЛНЕНИЯ И ОПИСАНИЯ СЛОВ (ПМ АОС)

РУКОВОДСТВО ОПЕРАТОРА

АННОТАЦИЯ

В данном программном документе приведено руководство оператора по использованию ПМ АОС, предназначенного для ускорения набора слов в S-Terra KP console с помощью автодополнения с выводом их написания при наведении.

В разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы.

В разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск и выполнение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

В разделе «Сообщения оператору» приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.505-79).

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 4](#_Toc73689662)

[1. НАЗНАЧЕНИЕ ПРОГРАММЫ 5](#_Toc73689663)

[1.1. Функциональное назначение 5](#_Toc73689664)

[1.2. Эксплуатационное назначение 5](#_Toc73689665)

[2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ 6](#_Toc73689666)

[2.1. Требования к персоналу (пользователю) 6](#_Toc73689667)

[2.2. Требования к составу и параметрам технических средств 6](#_Toc73689668)

[3. ВЫПОЛНЕНИЕ ПРОГРАММЫ 6](#_Toc73689669)

[3.1. Установка и запуск программы 7](#_Toc73689670)

[3.2. Выполнение программы 7](#_Toc73689671)

[3.3. Экранные формы 7](#_Toc73689672)

[4. СООБЩЕНИЯ ОПЕРАТОРУ 8](#_Toc73689673)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ПМ – программный модуль

ПМ АОС – мобильное приложение для цифровой системы управления производством

ПО – программное обеспечение

БД – база данных

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1. Функциональное назначение

Функциональным назначением ПМ АОС является вывод списка слов в автодополнения и оперативное отображение документации при наведении на слово

1.2. Эксплуатационное назначение

ПМ АОС должен использоваться в программе S-Terra KP console. Конечными пользователями ПМ АОС являются пользователи программы S-Terra KP console.

# 1.3. Основные функциональные модули

Рассмотрим назначение основных функциональных модулей, на которые можно разбить ПМ АОС.

Модуль «AutoSuggestor» – класс, который является основой приложения. Он предназначен для анализирования введенного пользователем текста и сравнения его со словами из БД, формирования списка слов для окна автодополнения, отслеживания событий нажатия клавиш на клавиатуре, для подсветки или вставки определенного слова.

Модуль «WordsWithDescriptionForAutoSuggestor» – класс, который предназначен для хранения слова с относящейся к нему документацией. При сравнении введенных символов со словами из БД используется информация из этого класса, экземпляры которого содержатся в массиве для сравнения.

Модуль «SuggestionLabel» – это класс, который содержит слово, занесенное в список слов автодополнения: экземпляр класса заносится в массив слов, предназначенных для вывода на экран. Предназначен для: хранения слова и его документации, вывода описания слова при фокусе на него, вставки слова при двойном клике мыши, замены введенных символов в текстовом поле выбранным словом.

1. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1. Требования к персоналу (пользователю)

К пользователю не предъявляется никаких требований для использования ПМ АОС.

2.2. Требования к составу и параметрам технических средств

В состав технических средств должны входить сервер с базой данных производства под управлением СУБД PostgreSQL, JDK не ниже 8 версии, персональный компьютер под управлением MS Windows Server.

*Таблица 1*

Минимальный состав технических средств и их технические характеристики

|  |  |
| --- | --- |
| Процессор | 1.4 ГГц, x64 |
| Оперативная память | 512 Мб (2 Гб для варианта «Сервер с рабочим столом») |
| Операционная система | Windows Server 2012 (64-bit),  Windows Server 2012R2 (64-bit),  Windows Server 2016 (64-bit),  Windows Server 2019 (64-bit) |
| Свободное место на диске | 32 Гб |
| Разрешение экрана | 800\*600 |
| Устройства ввода/вывода | Мышь, клавиатура, монитор |

*Таблица 2*

Рекомендуемый состав технических средств и их технические характеристики

|  |  |
| --- | --- |
| Процессор | 3.1 ГГц, x64 |
| ОЗУ | 16 ГБ |
| Операционная система | Windows Server 2012 (64-bit),  Windows Server 2012R2 (64-bit),  Windows Server 2016 (64-bit),  Windows Server 2019 (64-bit) |
| Свободное место на диске | 60 Гб |
| Разрешение экрана | 1024\*768 или более высокое |
| Устройства ввода/вывода | Мышь, клавиатура, монитор |

1. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Установка и запуск программы

ПМ АОС не требует отдельной установки, поскольку данный ПМ будет установлен во время установки программы S-Terra KP Console.

Для запуска программы требуется только запустить программу S-Terra KP Console.

3.2. Выполнение программы

Выполнение ПМ АОС проходит в несколько этапов:

1. Ввод символов в текстовое поле
2. При отображении автодополнения при нажатии клавиш-стрелок будет подсвечено одно из слов в списке. Слово также подсвечивается при одном клике ЛКМ
3. При подсвеченном слове можно нажать Enter, и это слово вставится вместо введенных символов в текстовое поле. Также слово можно вставить при двойном клике ЛКМ.
4. Если при отображенном автодополнении продолжить ввод символов, то список автодополнения будет обновляться в соответствии с введенными символами или исчезнет вовсе, если совпадений не будет найдено.

3.3. Экранные формы

Пользовательский интерфейс показан на рисунках 3.1 – 3.3. На рисунке 3.1 показано окно с пустым текстовым полем. На рисунке 3.2 показано, что при введенных в текстовом поле символах отображается автодополнение, в котором не подсвечены никакие слова. На рисунке 3.3 отображено автодополнение с подсвеченным словом и отображенной документацией для него.

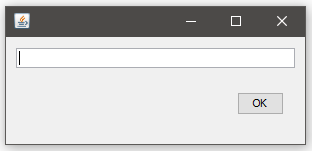


Рисунок 3.1 – окно с текстовым полем

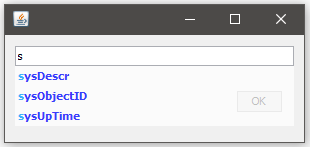


Рисунок 3.2 – окно с автодополнением

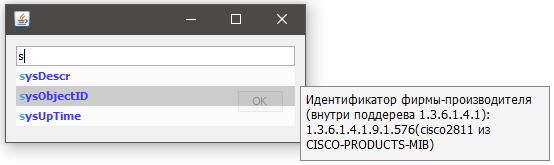


Рисунок 3.3 – окно с автодополнением и подсвеченным словом

4. СООБЩЕНИЯ ОПЕРАТОРУ

Сообщения для оператора ПМ АОС отсутствуют. Сообщения могут появиться только вне ПМ АОС, в программе S-Terra KP Console.

ПРИЛОЖЕНИЕ 2

Утверждаю

Зав. кафедрой ИПОВС  
НИУ МИЭТ  
д.т.н, проф. / Л.Г. Гагарина /

« » 20 г.

Техническое задание на разработку

«Программного модуля автодополнения и описания слов»

(шифр ПМ АОС)

Направление подготовки – 09.03.04

Квалификация – бакалавр

Руководитель выпускной работы:

к. т. н., доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кононова А.И./

Исполнитель:

студент гр. ПИН–41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Мадеев К.А./

Москва 2021

Введение

В текстовых редакторах или средах разработки используются программы автодополнения текста. Они позволяют значительно ускорить набор текста и программного кода.

В программе S-Terra KP Console необходимо часто вводить дилнные названия ключевых слов. А чтобы использовать их правильно необходимо открывать документацию, которая содержит в себе описания этих слов. В совокупности данные действия сильно замедляют процесс настройки программы.

Для ускорения набора ключевых слов в программе S-Terra KP Console, было принято решение разработать программный модуль, решающий данную задачу.

1. Основания для разработки
   1. Основание для разработки:

* задание на выпускную работу;
* решение директора института СПИНТех;
  1. Наименование и шифр работы:

Разработка программного модуля автодополнения и описания слов

Программному модулю, разрабатываемому по-настоящему ТЗ присваивается шифр «ПМ АОС» (далее – ПМ АОС).

* 1. Исполнитель:

Исполнителем является студент группы ПИН-41 НИУ «МИЭТ» Мадеев Константин Андреевич

1. Назначение разработки

ПМ необходим для упрощения использования программы при наборе текста

и помощи в выборе метрики в программе S-Terra KP Console.

1. Технические требования
   1. Требования к функциональным характеристикам
      1. Состав выполняемых функций

ПМ АОМ должен обеспечивать выполнение следующих функций:

* + Отображение выпадающего списка, показывающего варианты слов, начинающихся с введенных пользователем символов
  + Выбор вариантов слов в выпадающем списке при помощи устройств ввода: клавиатуры и мыши
  + Подсветку варианта выбора, если использовании клавиатура для перемещения по вариантам выбора
  + Отображение описания подсвеченного слова
    1. Организация входных и выходных данных

Входные данные

К входным данным относятся:

* Словарь, содержащий названия метрик
* Символы, введенные пользователем

Выходные данные

К выходным данным относятся:

* Выпадающий список автодополнения с вариантами слов
* Вставка в поле ввода выбранного слова вместо введенных пользователем символов
  1. Требования к надежности

Работа ПМ АОС не должна приводить к зависанию, замедлению, аварийному завершению основной программы.

* 1. Условия эксплуатации и требования к составу и параметрам технических средств

ПМ АОС не требует навыков работы с ПК.

*Таблица 1*

Минимальный состав технических средств и их технические характеристики

|  |  |
| --- | --- |
| Процессор | 1.4 ГГц, x64 |
| Оперативная память | 512 Мб (2 Гб для варианта «Сервер с рабочим столом») |
| Операционная система | Windows Server 2012 (64-bit),  Windows Server 2012R2 (64-bit),  Windows Server 2016 (64-bit),  Windows Server 2019 (64-bit) |
| Свободное место на диске | 32 Гб |
| Разрешение экрана | 800\*600 |
| Устройства ввода/вывода | Мышь, клавиатура, монитор |

*Таблица 2*

Рекомендуемый состав технических средств и их технические характеристики

|  |  |
| --- | --- |
| Процессор | 3.1 ГГц, x64 |
| ОЗУ | 16 ГБ |
| Операционная система | Windows Server 2012 (64-bit),  Windows Server 2012R2 (64-bit),  Windows Server 2016 (64-bit),  Windows Server 2019 (64-bit) |
| Свободное место на диске | 60 Гб |
| Разрешение экрана | 1024\*768 или более высокое |
| Устройства ввода/вывода | Мышь, клавиатура, монитор |

* 1. Требования к информационной и программной совместимости

ПМ должен корректно работать под управлением ОС Windows Server 2012 и выше;

Среда программирования – на усмотрение разработчика;

Язык программирования – Java;

Библиотека для GUI – Swing.

* 1. Требования к транспортировке и хранению

Не предъявляются

* 1. Специальные требования

Не предъявляются

1. Требования к программной документации

Вместе с ПП должны быть разработаны следующие документы Единой Системы Программной Документации (ЕСПД):

* Текст программы (ГОСТ 19.401-78);
* Руководство оператора (ГОСТ 19.505-79).

1. Стадии и этапы разработки

С 10.02.2021 по 05.06.2021 должны быть проведены следующие работы:

*Таблица 3*

Стадии и этапы разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование работы | Дата начала | Дата окончания | Форма отчетности |
| 1 | Постановка задачи | 10.02.2021 | 14.02.2021 | Эскиз слайда |
| 2 | Проведение предварительных НИР | 14.02.2021 | 15.02.2021 | Сравнительная таблица |
| 3 | Разработка Т3 | 15.02.2021 | 03.03.2021 | Утвержденное ТЗ |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Разработка структуры данных ПМ АОС | 04.03.2021 | 08.03.2021 | Схема данных |
| 5 | Разработка общего алгоритма работы ПМ АОС | 09.03.2021 | 18.03.2021 | Схема алгоритма |
| 6 | Уточнение методов решения | 19.03.2021 | 19.03.2021 | Текстовый документ |
| 7 | Проведение НИР | 20.03.2021 | 31.03.2021 | Сравнительные таблицы |
| 8 | Уточнение схемы данных ПМ АОС | 01.04.2021 | 04.04.2021 | Уточненная схема данных |
| 9 | Уточнение схемы алгоритма ПМ АОС | 27.03.2021 | 09.04.2021 | Алгоритмы модулей |
| 10 | Программная реализация алгоритма ПМ АОС | 10.04.2021 | 07.05.2021 | Программный код ПМ АОС |
| 11 | Разработка программных документов | 08.05.2021 | 10.05.2021 | Текстовые документы |
| 12 | Разработка программы и методика испытаний | 11.05.2021 | 21.05.2021 | Тест-кейс |
| 13 | Проведение испытаний | 22.05.2021 | 25.05.2021 | Результаты испытаний |
| 14 | Корректировка программ и документов по результатам испытаний | 25.05.2021 | 30.05.2021 | Скорректированные документы |
| 15 | Передача программы и документов для сопровождения | 01.05.2021 | 05.06.2021 | Рабочие документы |

1. Порядок контроля и приемки

Во время контроля и приёмки осуществляется проверка ПМ заявленному функционалу на ЭВМ заказчика.

На испытании предъявляются:

* Описание ПМ;
* Исходный код ПМ;
* Разработанный ПМ.