

|  |
| --- |
| **Министерство науки и ВЫСШЕГО ОБРАЗОВАНИЯ Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Уральский государственный экономический университет»**  **(УрГЭУ)** |

**КУРСОВАЯ РАБОТА**

по дисциплине **«Объектно-ориентированное программирование»**

на тему: **Разработка** **программного приложения с использованием библиотеки пользовательского интерфейса JavaFX**

|  |  |  |
| --- | --- | --- |
| Институт непрерывного и дистанционного образования      Направление подготовки  *09.03.01 Информатика и вычислительная техника*    Направленность (профиль)  *Программное обеспечение автоматизированных систем*  Кафедра  *Кафедра информационных технологий и статистики* |  | Студент  *Светляков Михаил Иванович*  Группа *ИНО ОЗБ ПОАС-23*  Руководитель  *Панов Михаил Александрович, к.э.н., доцент кафедры информационных технологий и статистики* |

Екатеринбург

2024 г.

[ВВЕДЕНИЕ 4](#_Toc164889853)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc164889854)

[1.1. Описание предметной области 6](#_Toc164889855)

[1.2. Словарь предметной области 8](#_Toc164889856)

[1.3. Анализ существующих программных решений 12](#_Toc164889857)

[1.4. Техническое задание 16](#_Toc164889858)

[1.4.1. Функциональные требования 16](#_Toc164889859)

[1.4.2. Требование к экранным формам 16](#_Toc164889860)

[1.4.3. Модель данных 17](#_Toc164889861)

[1.4.4. Нефункциональные требования 18](#_Toc164889862)

[2. ПРОЕКТНАЯ ЧАСТЬ 20](#_Toc164889863)

[2.1. Проектирование и разработка классов 20](#_Toc164889864)

[2.2. Проектирование пользовательского интерфейса 22](#_Toc164889865)

[2.3. Описание разработанных алгоритмов и программных модулей 23](#_Toc164889866)

[2.3.1. Иерархическая структура приложения 23](#_Toc164889867)

[2.3.2. Алгоритмы 23](#_Toc164889868)

[2.3.3. Описание основных программных модулей 25](#_Toc164889869)

[2.4. Тестирование программного комплекса 26](#_Toc164889870)

[2.4.1. Описание методики тестирования 26](#_Toc164889871)

[2.4.2. Результаты тестов и анализ 26](#_Toc164889872)

[2.5. Руководство пользователя 32](#_Toc164889873)

[ЗАКЛЮЧЕНИЕ 33](#_Toc164889874)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 34](#_Toc164889875)

[ПРИЛОЖЕНИЕ A 37](#_Toc164889876)

[ПРИЛОЖЕНИЕ Б 38](#_Toc164889877)

ВВЕДЕНИЕ

Цель курсовой работы: получить опыт использования библиотеки пользовательского интерфейса JavaFX в связке с языком программирования Java в процессе разработке приложения для конвертации валют.

Актуальность темы: тема получения реального опыта работы с тем или иным инструментом для начинающих специалистов всегда стоит остро. Создание приложения с использованием желаемого инструмента является одним из лучших способов получения опыта.

Разработка приложения для конвертации валют позволяет применить и развивать навыки в области программирования, а также изучить и применить новые технологии и алгоритмы для обеспечения точности и скорости конвертации, что дает разработчику опыт работы с используемыми инструментами, который поможет при прохождении собеседования на работу.

Приложения для расчетов отношений курсов валют, при использовании их как службы или сервисов, предоставляющие программный интерфейс для взаимодействия, позволяют внедрять их функционал в другие приложения, расширяя клиентскую базу не только кончеными пользователями оригинального приложения, но и юридическими лицами, которые использую функционал этого приложения в других своих продуктах.

Задачи:

* Исследовать и анализировать существующие решения в области разработки программного обеспечения.
* Определить требования и спецификации для программы.
* Разработать алгоритмы и логику работы для нового приложения.
* Спроектировать и реализовать пользовательский интерфейс для нового программного приложения.
* Провести тестирование программного продукта для обеспечения корректной работы программы.

Объект исследования: библиотека пользовательского интерфейса JavaFX.

Предмет исследования: программа для конвертации валют.

Краткое изложение структуры курсовой работы: первая часть работы отводится изучению предметной области, определению требований к разрабатываемой программе. Вторая часть работы отводится созданию и реализации алгоритмов, интерфейса приложения, предназначенного для конвертации валют.

Задачи:

* Исследовать и проанализировать существующие решения в сфере программного обеспечения.
* Определить требования для программного продукта.
* Разработать алгоритмы и логику работы для создаваемого программного приложения.
* Спроектировать и реализовать пользовательский интерфейс для нового программного приложения.
* Провести тестирование программного продукта для обеспечения его корректной работы.

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Описание предметной области

Перед началом разработки и написанием кода важно изучить предметную область разрабатываемого приложения. Под предметной областью предполагается создание программы, направленной на решение поставленных задач в определенной сфере деятельности.

В данной курсовой работе, объектом исследования является библиотека пользовательского интерфейса JavaFX. JavaFX — это набор графических и мультимедийных пакетов, который позволяет разработчикам создавать, тестировать, отлаживать и развертывать богатые клиентские приложения, работающие на различных платформах.

Библиотека JavaFX представляет собой общедоступный интерфейс программирования приложений Java (API), доступный для разработчиков.

Особенности использования JavaFX:

* JavaFX, библиотека для Java, которая предоставляет разработчикам возможность использовать все преимущества Java.
* Текущие приложения, разработанные на Swing, могут быть модернизированы, используя возможности JavaFX.
* FXML. FXML представляет собой язык разметки на основе XML и служит для создания пользовательского интерфейса UI приложения. Этот подход   
  позволяет писать приложение в декларативном стиле, описывая то, что нужно сделать, а не каким путем это нужно реализовать.
* Scene Builder. Отдельное приложение, предлагающее инструмент для написания и создания прототипов пользовательского интерфейса в FXML посредством размещения графических элементов на полотне и настройке базового поведения этих элементов. Может быть интегрирован напрямую в среду разработки IntelliJ IDEA.

JavaFX определенно можно считать инструментом разработки приложений под разные платформы, поддерживаемые Java Runtime.

Область Применения:

Хоть многие процессы многие процессы перешли из традиционных компьютерных систем в облачные и веб-решения, приложения для персональных компьютеров не теряют актуальности. Такие решения обладают более высокой эффективностью и надежностью, а в некоторых сферах просто незаменимы.

Некоторые сферы применения десктопных приложений:

* Компьютерные игры.
* Интегрированные среды разработки (IDE).
* Редакторы мультимедиа.

## Словарь предметной области

Словарь предметной области представляет — это набор терминов, понятий, сущностей и процессов, свойственных для конкретной предметной области. Используется для исчерпывающего описания элементов предметной области, что обязательно при разработке программного проекта, где требуется четкое понимание особенностей выбранной предметной области.

Java – объектно-ориентированный язык программирования.

Объектно-ориентированное программирование (ООП) – одна из парадигм организации программного кода, при которой исходный код описывает объекты и их поведение, а не прямое манипулирование исходными данными.

Объект – один из терминов ООП; определенная сущность, для которой описаны параметры и поведение, с публичным доступом, подразумевая, что к их можно использовать с наружи объекта, или приватными, обозначая, что эти параметры и методы для внутреннего использования объектом.

Класс – это модель для создания объектов определённого типа; описывает параметры и методы объектов. В ООП класс служит инструментом реализации абстрактных типов данных, определяя одновременно как интерфейс, так и реализацию для всех своих экземпляров (т.е. объектов).

Интерфейс – это соглашение, которое определяет набор методов, которые должны быть реализованы в классе, который реализует данный интерфейс.

Абстракция – это принцип, суть которого подразумевает сокрытие деталей реализации чего-либо, представляя сложный объект, как более простой с набором тех параметров, которые нам интересны в рамках задачи.

Инкапсуляция – принцип ООП, гласящий о сокрытии состояния и методов реализации внутренней логики класса или объекта. Согласно этому принципу, сущность предоставляет наружу только то, что нужно, сохраняя внутренние инварианты сущности.

Полиморфизм – принцип ООП, за счет которого классы потомки могут переопределять поведение, полученное от родительского класса.

Наследование – принцип ООП, согласно которому классы потомки перенимают параметры и методы родительского класса.

Модификаторы доступа – это зарезервированные слова в языке программирования, размещение которых в определенных метах кода указывает категорию доступности членов класса и методов, декларирую, как могут быть использованы те или иные члены класса.

Public – члены класса, объявленные как public, доступны из любого места программы, где видимо определение класса. Сущности с данным модификатором могут быть доступны извне класса, а также из производных классов.

Private – члены класса, объявленные как private, доступны только внутри класса, который их объявляет. Сущности с данным модификатором не могут быть доступны извне класса или из производных классов.

Protected – члены класса, объявленные как protected, доступны внутри класса, который их объявляет, а также в его производных классах. Сущности с данным модификатором не могут быть доступны извне класса, но могут быть доступны из производных классов.

Scene Builder – внешний инструмент для создания пользовательского интерфейса, редактируя файл FXML путем перетаскивания графических элементов на полотно.

(FXML) Fork eXtensible Markup Language – это формат описания положения графических элементов на сцене, основанный на XML; используется JavaFX.

Контейнеры – это панели компоновки в JavaFX, которые используются для организации положения элементов управления в приложении.

Stage – класс в JavaFX. Представляет окно приложения и служит контейнером для всех компонентов интерфейса для данного окна.

VBox – является компонентом-контейнером, который располагает все свои дочерние узлы (компоненты) в вертикальном порядке, друг за другом.

BorderPane – контейнер, который позволяет размещать компоненты в пяти различных областях: Top, Left, Right, Center и Bottom.

Элементы управления – графические элементы управления приложением в JavaFX; представляют собой разнообразные компоненты, служащие для создания пользовательского интерфейса. Эти элементы управления являются объектами и используются для взаимодействия с пользователем, обработки событий и отображения данных.

Кнопки и метки (Buttons and Labels) – кнопки (Button) используются для выполнения действий при нажатии, а метки (Label) для отображения текста.

Текстовые элементы управления – включают TextField для однострочного ввода текста и TextArea для многострочного ввода.

ListView – Отображает список элементов, которые могут быть выбраны.

## Анализ существующих программных решений

Изучение и анализ существующих программных решений в программировании подразумевает оценку различных программных продуктов, инструментов и технологий, которые применяются в разработке программного обеспечения. В процессе анализа рассматривается функциональность, производительность, надежность, удобство использования и другие аспекты программных продуктов для выявления наиболее подходящих альтернатив для конкретной задачи и требований проекта. Является одним из важнейших этапов в процессе разработки ПО, определяя наиболее подходящие инструменты и технологии для реализации требований, повышая скорость разработки, качество конечного продукта, его эффективность и надежность.

Abstract Window Toolkit (AWT)– это первоначальный набор инструментов для создания графических интерфейсов на языке программирования Java, предшествующий библиотеке Swing. Представляющий часть Java Foundation Classes (JFC), этот набор предоставляет базовый API для создания графических интерфейсов в Java-приложениях. AWT также используется в определенных профилях Java ME, например, в Connected Device Configuration, подчеркивая его важную роль в поддержке Java на мобильных устройствах.

Преимущества AWT:

* Интеграция с нативными элементами ОС
* Мощная система обработки событий
* Инструменты для работы с графикой и изображениями
* Менеджеры компоновки

Недостатки AWT:

* Тяжеловесность компонентов
* Зависимость от ОС
* Ограниченность интерфейса

После успешного запуска AWT, компания Sun Microsystems представила новаторский набор графических компонентов под названием Swing. Эти компоненты полностью разработаны на языке Java, что позволяет их использование на разных платформах. Визуализация осуществляется с использованием 2D-графики, открывая широкие возможности для создания привлекательных визуальных интерфейсов.

Преимущества Swing:

* Кроссплатформенность: позволяет разрабатывать приложения, которые работают на различных операционных системах.
* Глубокая интеграция с Java: использует возможности Java для работы с объектами и коллекциями.
* Расширенный набор компонентов: предоставляет широкий выбор компонентов для создания пользовательского интерфейса.

Недостатки Swing:

* Производительность: замечалась недостаточная производительность при работе с большими объемами данных или продвинутой графикой по сравнению с другими решениями.
* Сложности в стилизации: использование нативных компонентов ОС может усложнить стилизацию.
* Обновления и поддержка: развитие и поддержка могут быть менее активными по сравнению с другими фреймворками.

Standard Widget Toolkit (SWT), разработанный в компании IBM, был выпущен на рынок еще до достижения пика популярности Swing, его основной целью было продвижение интегрированной среды разработки Eclipse. В отличие от AWT, SWT использует нативные компоненты операционной системы, что обеспечивает лучшую адаптацию к разным платформам, требующим свои собственные версии Jar-библиотек. Это позволяет достичь более глубокой интеграции с операционной системой, но в то же время увеличивает сложность разработки, так как требуется предоставление отдельных реализаций для каждой платформы.

Преимущества SWT:

* Эффективность: SWT использует нативные элементы управления, что повышает производительность по сравнению с Swing.
* Нативный вид: Приложения, разработанные на основе SWT, имеют более нативный вид на различных платформах, что улучшает опыт пользователя.
* Интеграция с Eclipse: Eclipse реализовал поддержку SWT в своем продукте, что упрощает разработку приложений, при использовании продукта Eclipse.

Недостатки SWT:

* Недостаточная поддержка определенных компонентов: SWT не реализует поддержку всех компонентов, существующих в Swing, что может уменьшает возможности программистов и замедляет разработку, заставляя создавать приложения с определенным типом интерфейса.
* Сложность в использовании: SWT может быть сложнее в использовании по сравнению с Swing, особенно для начинающих из-за более низкоуровневого подхода к созданию графического интерфейса пользователя.
* Недостаток активной поддержки: В отличие от Swing, который активно поддерживается и развивается в рамках проекта OpenJDK, SWT может не получать такого же уровня поддержки и обновлений.

Платформа JavaFX представляет собой инновационный инструмент в области создания программного обеспечения, обеспечивая эффективную отрисовку через использование графического конвейера, что значительно ускоряет процесс выполнения приложений. В этой платформе имеется обширный набор встроенных компонентов, включая специализированные элементы для визуализации данных, а также поддержка мультимедийного контента и анимации. Смена оформления графических элементов достигается за счет использования CSS, позволяя использовать инструмент, хорошо зарекомендовавший себя в Web сфере.

Достоинства JavaFX:

* Позволяет разработчикам писать понятный, управляемый код на Java, который удобно обновлять и отлаживать;
* Включает в себя все библиотеки, необходимые для разработки GUI, и поддерживает CSS, FXML и многопоточность;
* Обеспечивает разработчиков многочисленными готовыми инструментами и удобным JavaFX Scene Builder, визуальным инструментом для разработки пользовательских интерфейсов без необходимости написания кода.

Недостатки JavaFX:

* Не подходит для приложений с высокой степенью сложности; 1
* Ограниченность сообщества JavaFX.

Выбор между JavaFX, Swing, AWT и SWT зависит от конкретных требований проекта, таких как производительность, количество доступных компонентов интерфейса, удобство использования, соответствие современным стандартам Java и поддержка сообщества.

## Техническое задание

### Функциональные требования

Функциональные требования — это доскональное описание функциональности разрабатываемого приложения. Представляет собой основной набор требований к программному продукту, которые должны быть реализованы.

При разработке программы следует учитывать следующие функциональные требования:

* Создание интуитивно понятного интерфейса: Разработка интерфейса, который легко понять и ориентироваться, позволяет пользователям без особых усилий взаимодействовать с программой.
* Визуализация иконок: пользователь должен понимать, за что отвечает та или иная кнопка без указания текстовой подсказки, что позволит уменьшить визуальных раздражителей в окне приложения.
* Удобство взаимодействия: пользователь должен иметь возможность легко закрыть программу, свернуть её и переместить меню запущенной программы на любой участок экрана Windows.
* Обновление данных: для обеспечения актуальности данных, которые пользователь получает, необходимо обновлять информацию через сетевые запросы к REST API.

### Требование к экранным формам

Требования к экранным формам включают в себя стандарты и правила, которые должны соблюдаться при создании и использовании интерфейсов в различных сферах, таких как веб-разработка, мобильная разработка и разработка программного обеспечения. Экранные формы представляют собой пользовательский интерфейс, позволяющий пользователям взаимодействовать с программным обеспечением или веб-сайтом, вводя данные, выбирая опции и выполняя другие действия.

Для соответствия нефункциональным требованиям, программный продукт должен быть разработан с использованием единого подхода к оформлению графических элементов, и реализации ожидаемого функционала окна приложения с одинаковым расположением ключевых элементов управления и навигации. Элементы управления, такие как кнопки закрытия и сворачивания приложения, обычно располагаются в верхнем левом углу, в то время как основные элементы управления располагаются по центру.

Для обозначения данных используются не только текстовые метки, но и графические иконки, что помогает улучшить восприятие информации пользователем. Цветовая палитра, включая как информационные элементы, так и фон, должна быть выбрана так, чтобы создавать спокойную и непринужденную атмосферу, минимизируя усталость пользователя.

### Модель данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь.

В данной курсовой работе для получения данных с сервиса используется формат передачи данных XML

XML (Extensible Markup Language) — это текстовый формат обмена данными, который используется для хранения и передачи данных в структурированном виде. В отличие от JSON, XML является более сложным и менее гибким, но предоставляет более широкие возможности для описания структуры данных и поддерживает пространства имен (namespaces), что делает его более подходящим для сложных систем и приложений.

Основные элементы XML включают:

* Элементы: основные строительные блоки XML, которые представляют собой пары открывающий/закрывающий тег. Элементы могут содержать в себе другие элементы, организую древовидную структуру файла.
* Атрибуты: дополнительные данные, прикрепленные к элементам, которые обычно используются для хранения метаданных.
* Пространства имен: предоставляют способ группировки элементов и атрибутов, что позволяет избежать конфликтов имен в сложных XML-документах.
* CDATA: позволяет включить в XML текст, который не должен быть интерпретирован как XML.
* Комментарии: текст, который не влияет на содержимое XML, используется для добавления пояснений или аннотаций.

### Нефункциональные требования

Нефункциональные требования - субъективные характеристики системы, которые не связаны с её конкретными функциональными возможностями. Представляют собой совокупность таких параметров, как скорость отклика приложения, оформление, и другие параметры.

Нефункциональные требования к программе:

* Пользователь не должен испытывать отвращения при работе с разработанной программой.
* Работоспособность в случае отсутствия интернет-подключения или невозможности получения пакетов данных от удаленного сервиса, предоставляющего данные.
* Программа должна предоставлять данные о курсах валют в удобном для пользователя формате.
* Должен быть реализованы инструменты расчетов курсов валют относительно друг друга.
* Программа должна обеспечивать стабильную работу, минимизируя время простоя и обеспечивая доступность данных курсах валют.

1. ПРОЕКТНАЯ ЧАСТЬ

## Проектирование и разработка классов

Данное приложение состоит из множества классов, которые можно разделить по функциональности, которые они реализуют.

Краткое описание функционалов классов:

CalculatorsScene – сцена калькуляторов.

OfflineCalculatorScene – сцена для калькулятора, которому не нужно онлайн подключение.

OnlineCalculatorScene – сцена для калькулятора, который получает данные из API, предоставляемом Банком России.

AppHeaderScene – сцена для заголовочной части окна приложения. Содержит функционал для управления окном приложения и кнопку для открытия окна настроек.  
 CenterScene – центральный виджет приложения. Является контейнером для других сцен.

MainScene – основной виджет приложения, который передается Stage основного окна.  
 WindowControlScene – сцена, содержащая кнопки для сворачивания, разворачивании и закрытии текущего окна.  
 CurrencyData – класс-структура для хранения данных, полученных от API.

CurrencyFetcher – класс получения данных от API.

AppTheme – класс для смены темы приложения по вкусу пользователя.

ResizeHelper – вспомогательный класс для возможности изменения размеров окна при перетягивании границ окна.

SceneTools – вспомогательный класс для инкапсуляции однотипных действий при написании кода программы.

AppLabel – класс виджета, реализующий специализированный Text.

CurrenciesTable – класс для отображения таблицы отношений курсов валют.  
 CurrencyRow – класс-структура для хранения записи для CurrenciesTable.

IconButton – класс виджета, реализующий кнопку с иконкой.

CustomDateCell – класс, модифицирующий возможность выбора даты в MyCalendar.

MyCalendar – класс виджета, реализующий календарь для выбора даты.

SettingsWindow – класс, реализующий окно настроек.

App – основная точка входа в приложение.

GlobalConstants – класс-namespace для хранения глобальных констант.

## Проектирование пользовательского интерфейса

Основные этапы проектирования пользовательского интерфейса определяют правила, требования и другие параметры, предъявляемые к итоговому дизайну приложения. Данный этап содержит в себе этап описания требований к приложению, благодаря которым пользователь сможет эффективно использовать приложение или сервис.

Требования к внешнему виду пользовательского интерфейса:

* Предсказуемость поведения приложения для пользователя.
* Используйте стандартные элементы управления и значки, которые пользователи уже знают и ожидают от различных приложений, которыми пользователь пользовался ранее. Это ускоряет процесс адаптации и сокращает время, необходимое для изучения нового интерфейса.
* Наличие специализированных визуальных элементов, которые могут улучшить понимание данных на пользовательском интерфейсе.
* Наличие возможности смены темы приложения.

## Описание разработанных алгоритмов и программных модулей

### Иерархическая структура приложения

Данное приложение было разработано без использования FXML и SceneBuilder.

Большинство классов, которые отвечают за функционирование и положение функциональных элементов графического интерфейса в окне приложения, относятся друг к другу отношением «содержит», т.е. используют принцип агрегирования заместо наследования.   
 Конструирование иерархии контейнеров, содержащих графические элементы реализуется за счет методов, GetScene и GetNode, например:

Листинг 1. Пример реализации методов GetScene и GetNode:

public Scene GetScene() { return button.getScene(); }  
public Node GetNode() { return button; }

Внешние библиотеки, которые используются в приложении и требуются для сборки программы, находятся в папке:

«\src\main\java\libs»

Для сборки приложения нужно добавить файлы из этой папки в систему сборки проекта.

### Алгоритмы

Для получения данных от API, предоставляемого Банком России, используется данный метод:

Листинг 2. Реализация метода получения данных от API класса CurrencyFetcher:

static private void GetData(LocalDate date) {  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");  
 String dateString = date.format(formatter);  
  
 try {  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder = factory.newDocumentBuilder();  
 URL url = new URL("http://www.cbr.ru/scripts/XML\_daily.asp?date\_req=" + dateString);  
 Document document = builder.parse(url.openStream());  
  
 NodeList currencies = document.getElementsByTagName("Valute");  
 // Итерация по всем элементам  
 *AddCurrency*(date, currencies);  
  
 } catch (ParserConfigurationException | SAXException | IOException e) {  
 e.printStackTrace();  
 }  
}

Для расчета отношений курсов валют, доступных лотов к покупке и доступных к покупке единиц валюты отвечает данная часть кода:

Листинг 3. Часть кода, отвечающая за расчет отношений параметров валют метода RefillTable класса OnlineCalculatorScene:

double relation = currentCurrency.valueRate\_ / currency.getValue().valueRate\_;  
double rawAvailable = relation \* count;  
double available = (rawAvailable - (commission / 100) \* rawAvailable);  
double availableToBuy = available / currency.getValue().nominal\_;

### Описание основных программных модулей

Базовый программный модуль — это независимая, самостоятельная единица программного кода, содержащая определенный набор функций, классов, переменных и других элементов программы.

Они используются для организации кода и управления им, обеспечивая модульность и повторное использование кода.

Модули делят программу на логические блоки, каждый из которых выполняет определенную функцию, что упрощает понимание и поддержку кода.

Классы исходного кода программы можно поделить на несколько групп, которые находятся в следующих папках:

* «src/main/java/com/usue/svetlyakov/coursework/scenes/calculator»: классы, релизующие функциональность онлайн версии конвертера валют, и офлайн версию конвертера.
* «src/main/java/com/usue/svetlyakov/coursework/scenes/settings»: классы, реализующие функциональность окна настроек приложения.
* «src/main/java/com/usue/svetlyakov/coursework/tools/currency»: классы, реализующие функицонал получения и кеширования данных, полученных от API.
* «src/main/java/com/usue/svetlyakov/coursework/tools/theme»: классы, реализующие смену темы приложения.
* «src/main/java/com/usue/svetlyakov/coursework/widgets»: классы, реализующие единичные виджеты со своей уникальной функциональностью.

## Тестирование программного комплекса

### Описание методики тестирования

В приложении были применены следующие методы тестирования:

* Функциональное тестирование.
* Тестирование удобства пользователя.
* Тестирование отказоустойчивости.

Функциональное тестирование проверяет, соответствует ли программное обеспечение исходным функциональным требованиям, а тестирование пользовательского опыта оценивает простоту использования, легкость обучения и привлекательность продукта для пользователей.

Тесты оценивали приложение с точки зрения производительности, эффективности, точности, использования памяти и отзыва пользователя.

### Результаты тестов и анализ

После запуска приложения, пользователя встречает окно программы, изображенное на рисунке 1.

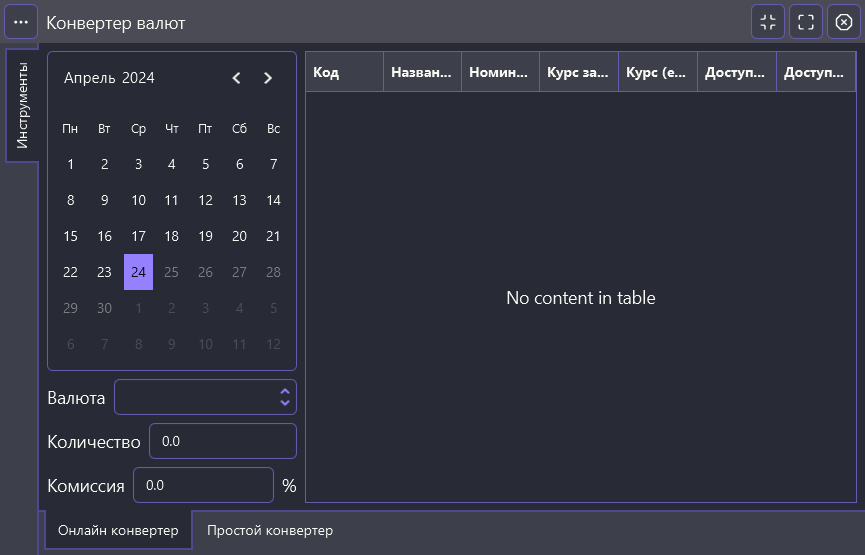


Рисунок 1 – вид открытого приложения[[1]](#footnote-1)

Пользователя встречает окно с онлайн конвертером валют. Для проверки работоспособности данного функционала нужно выбрать начальную валюту, чтобы отобразились остальные валюты.

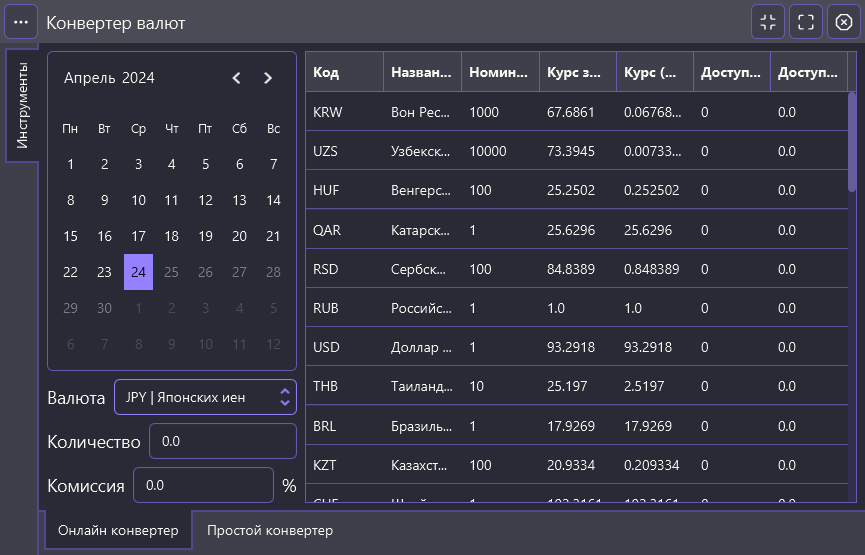


Рисунок 2 – вид приложения после выбора валют[[2]](#footnote-2)

Приложение получило данные из API Банка России, после чего заполнило таблицу данными.

При смене выбранной валюты, даты, количества валюты и процента комиссии, данные в таблице должны обновиться.

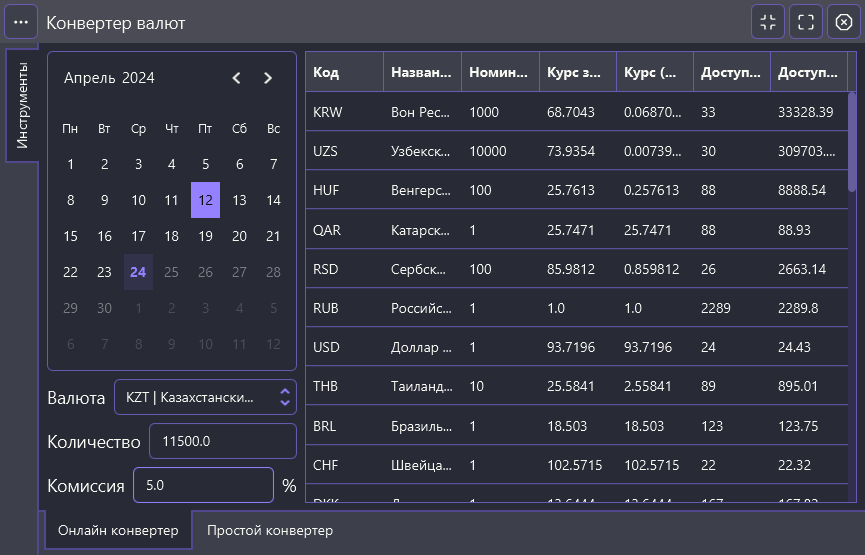


Рисунок 3 – вид приложения после смены параметров расчетов[[3]](#footnote-3)

Для офлайн версии конвертера валют реализован более простой вариант функционала.

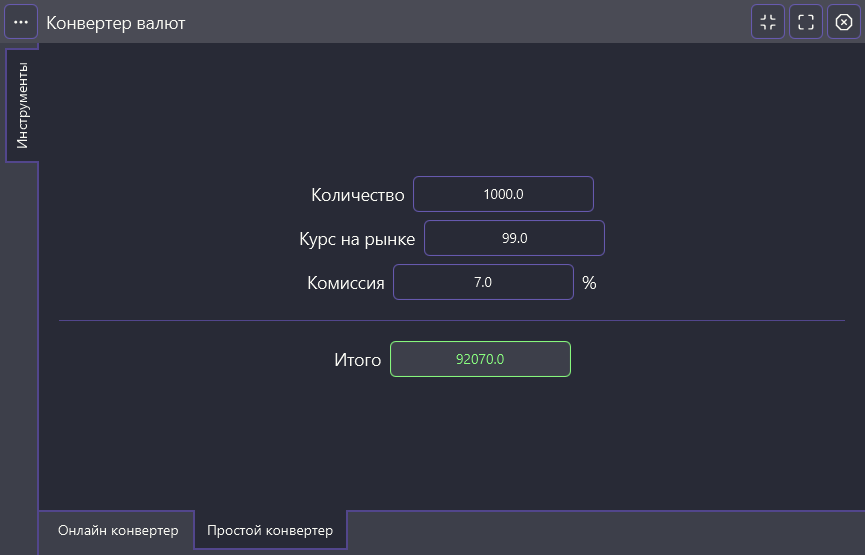


Рисунок 4 – вид приложения для простого конвертера валют[[4]](#footnote-4)

При нажатии на кнопку в верхнем левом углу приложения появляется окно настроек с возможностью смены темы приложения.

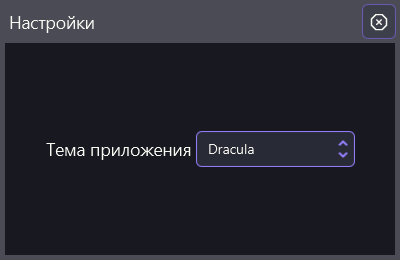


Рисунок 5 – вид окна настроек приложения[[5]](#footnote-5)

При смене темы приложения, оформление всей программы меняется в соответствии с выбранной темой.

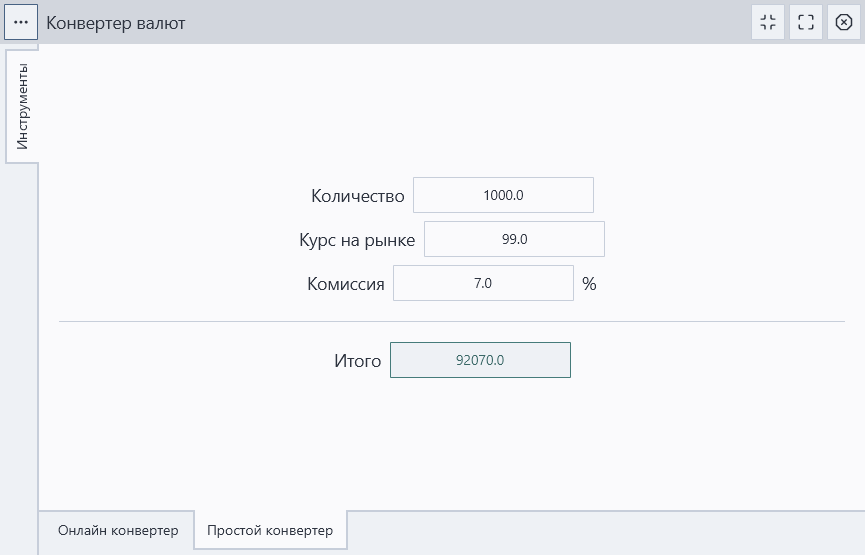


Рисунок 6 – вид приложения с измененной темой[[6]](#footnote-6)

## Руководство пользователя

Данное приложение позволяет рассчитать курсы валют как вручную, так и с использованием данных, полученных от API банка России.

Для использования онлайн версии конвертера валют нужно ввести количество единиц валюты, комиссию и выбрать начальную валюту. При смене параметров в приложении, программа сама пересчитает данные. Также данный инструмент позволяет выбрать дату, относительно которой будут проводится расчеты.

При использовании офлайн конвертера валют, вам нужно ввести параметры в поля. В случае, если параметры введены не корректно, интерфейс программы оповестит вас, обведя неверно заполненное поле красной рамкой.

Если пользователю захочется сменить тему приложения, он может открыть окно настроек приложения и сменить ее.

ЗАКЛЮЧЕНИЕ

На данный момент библиотека JavaFX выделяется своим удобством и богатым функционалом по сравнению с другими аналогичными библиотеками.

В примере данного приложения, можно сделать вывод, что JavaFX представляет собой удобный инструментарий для создания Java-приложений, что подтверждается непрекращающимся участием разработчиков в развитии данного инструментария и повсеместным использованием данной библиотеки для создания графических приложений.

Для достижения поставленной цели были выполнены следующие задачи: было проведено исследование существующих программных решений и установлено, что JavaFX имеет значительные преимущества перед альтернативами, включает в себя полный набор компонентов графического интерфейса, совместимость с CSS и возможность использования FXML для создания графический интерфейс.

Разработана спецификация и разработанное приложение полностью соответствует этим требованиям.

Он содержит все необходимые функции.

Нефункциональные требования к приложению соблюдены, программа разработана с использованием JavaFX, FXML, CSS, что подтверждает успешное выполнение поставленных задач и достижение целей.

В будущем программа, может быть доработана для удовлетворения измененных требований к программе.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

*Книги, статьи*

1. Хорстманн, Кей С. Биг Java: ранние объекты. — Москва: Питер, 2020. — 1200 с.
2. Жеребцов, А. С. Объектно-ориентированный анализ и программирование [Текст] : учебное пособие / А. С. Жеребцов, С. Ф. Молодецкая ; М-во образования и науки Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург: [Издательство УрГЭУ], 2014. - 126 с. Геккель, Брюс. Философия Java. — Санкт-Петербург: Питер, 2019. — 1152 с.Гонсалвес, Энтони Изучаем Java EE 7 / Энтони Гонсалвес. - М.: Питер, 2016. - 640 c.
3. Селдер, Рауль-Урма, Фурньо, Марио, Алан Майер. Java 8 и 9 в действии. — Москва: Манн, Иванов и Фербер, 2020. — 784 с.
4. Орельский, Владимир. Изучаем Java с нуля. — Москва: БХВ-Петербург, 2020. — 512 с.
5. Седжвик, Р. Алгоритмы на Java [Текст]: научное издание / Роберт Седжвик, Кевин Уэйн; [пер. с англ. А. А. Моргунова; под ред. Ю. Н. Артеменко]. - 4-е изд. - Москва: Вильямс, 2017. - 843 с
6. Прохоренок Н.А. JavaFX [Текст]: руководство для начинающи Прохоренок Н.А., 2020. - 687 с.

*Иностранная литература:*

1. Mastering JavaFX 10 Build Advanced and Visually Stunning Java Applications Sergey Grinev [Text]. - 2018. 218 p.
2. Ralph Lecessi Functional Interfaces in Java: Fundamentals and Examples [Text]. - 2019. 445 р.
3. Bloch, J. Effective Java: Programming Language Guide. Addison-Wesley Professional, 2018.
4. Josh Juneau Java EE 8 Recipes: A Problem-Solution Approach, Second Edition [Text]. - 2018. 298 p.

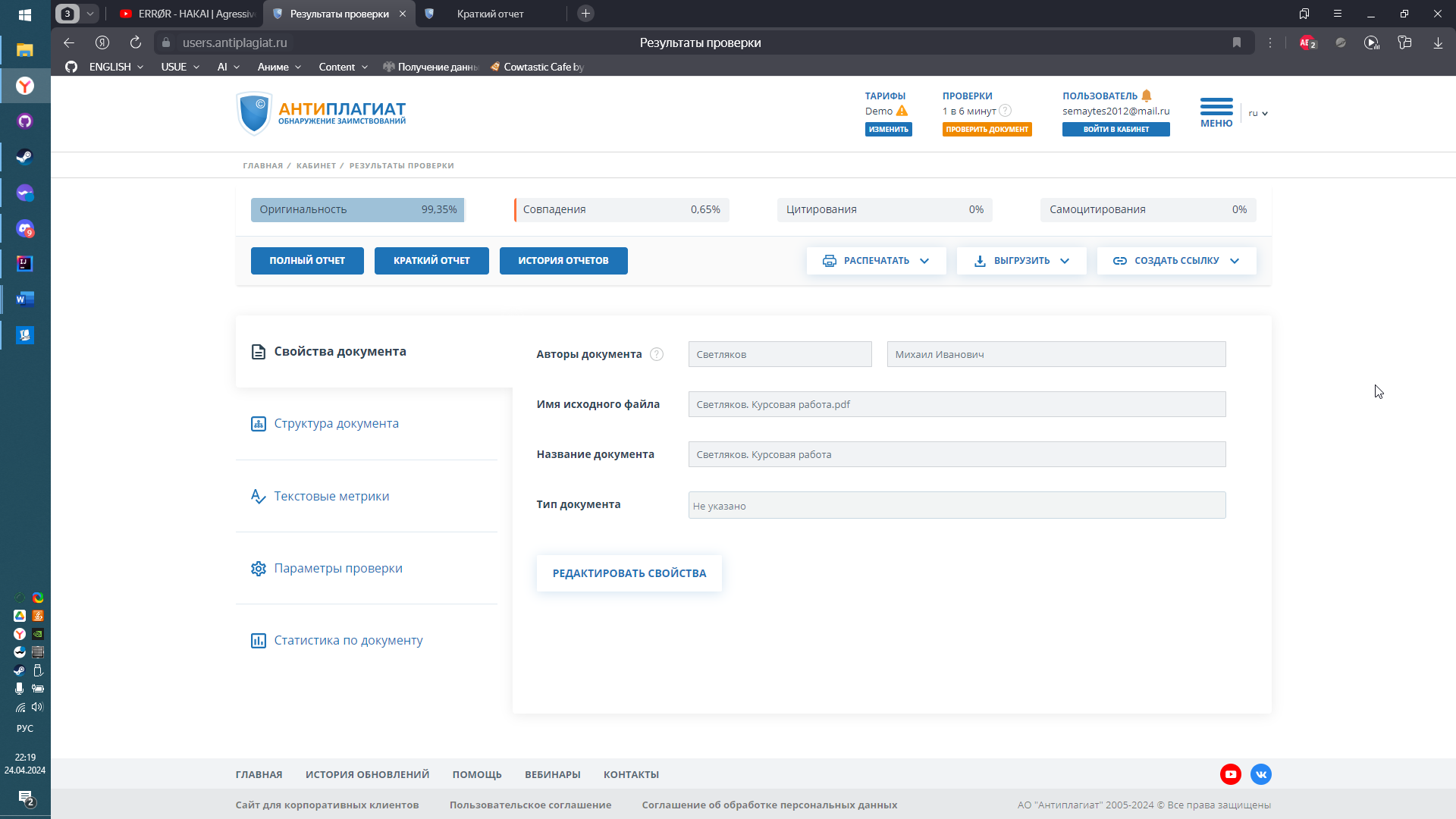
*Интернет-источники:*

1. Основных фреймворка для создания графических пользовательских интерфейсов (GUI): [Электронный ресурс] - Режим доступа: <https://www.geeksforgeeks.org/java-awt-vs-java-swing-vs-java-fx/>
2. Полезный ресурс для изучения Java: [Электронный ресурс] - Режим доступа: <https://habr.com/ru/articles/554274/>
3. Ресурс с иконками Freepik: [Электронный ресурс] - Режим доступа: [https://ru.](https://ru.freepik.com/icon/weather-news_648198)[freepik.com/icon/weather-news\_648198](https://ru.freepik.com/icon/weather-news_648198)
4. Официальный сайт JavaFX: [Электронный ресурс] - Режим доступа: <https://openjfx.io/>
5. Обучающие материалы по Java: [Электронный ресурс] - Режим доступа: <https://vertex-academy.com/tutorials/ru/list-java-primer/>
6. Подборкой полезных практических и обучающих материалов по Java Tproger : [Электронный ресурс] - Режим доступа: <https://tproger.ru/digest/java>
7. Официальная документация по Java от Oracle: [Электронный ресурс] - Режим доступа: <https://docs.oracle.com/en/java/>
8. Платформа для хранения и совместной работы над проектами на GitHub: [Электронный ресурс] - Режим доступа: <https://github.com/search?q=language%3Ajava&type=repositories>
9. Репозиторий проекта AtlantaFX: [Электронный ресурс] - Режим доступа: <https://github.com/mkpaz/atlantafx>
10. Документация по проекту AtlantaFX: [Электронный ресурс] - Режим доступа: <https://mkpaz.github.io/atlantafx/>
11. Страница на Wiki по JavaFX: [Электронный ресурс] - Режим доступа: <https://ru.wikipedia.org/wiki/JavaFX>

ПРИЛОЖЕНИЕ A

Ссылка на репозиторий с исходным кодом приложения: <https://github.com/Kimral/CurrencyConverter>

ПРИЛОЖЕНИЕ Б



1. Составлено автором [↑](#footnote-ref-1)
2. Составлено автором [↑](#footnote-ref-2)
3. Составлено автором [↑](#footnote-ref-3)
4. Составлено автором [↑](#footnote-ref-4)
5. Составлено автором [↑](#footnote-ref-5)
6. Составлено автором [↑](#footnote-ref-6)