МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «Программирование на языке “Java”»

на тему «**Посещаемость лекционных занятий**»

**Студент**

**группы 10702122** Солдатов Н. В.

**Руководитель**  доц. Сидорик В. В.

**Сроки выполнения:** 05.09.2024 – 05.12.2025

**Дата сдачи:** 05. 12. 2024

Минск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc184084391)

[1. ПОСТАНОВКА И ОПИСАНИЕ ЗАДАЧИ 4](#_Toc184084392)

[2. ТЕОРЕТИЧЕКИЙ РАЗДЕЛ 5](#_Toc184084393)

[2.1 Язык программирования Java 5](#_Toc184084394)

[2.2 Среда разработки Intellij IDEA Community Edition 2024.2 6](#_Toc184084395)

[2.3 Платформа JavaSwing 7](#_Toc184084396)

[3. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ 11](#_Toc184084397)

[4. ПРАКТИЧЕСКИЙ РАЗДЕЛ 12](#_Toc184084398)

[4.1 Описание структуры проекта 12](#_Toc184084399)

[4.2 UML-диаграмма классов 13](#_Toc184084400)

[4.3 Использование Apache POI, Maven 13](#_Toc184084401)

[4.4 Реализация графических окон 16](#_Toc184084402)

[4.5 Стартовое окно 16](#_Toc184084403)

[4.6 Главное окно 17](#_Toc184084404)

[4.7 Окно карты студента 20](#_Toc184084405)

[4.8 Окно с информацией об авторе 20](#_Toc184084406)

[4.9 Окно с информацией о программе 21](#_Toc184084407)

[4.10 Утилита JavaDoc 22](#_Toc184084408)

[4.11 Сборка jar-файла 22](#_Toc184084409)

[5. РЕЗУЛЬТАТЫ РАБОТЫ 23](#_Toc184084410)

[ЗАКЛЮЧЕНИЕ 29](#_Toc184084411)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 31](#_Toc184084412)

[ПРИЛОЖЕНИЕ 1 32](#_Toc184084413)

# **ВВЕДЕНИЕ**

В современном образовательном процессе цифровые технологии занимают всё более важное место, способствуя автоматизации и повышению эффективности управления учебным процессом. Одной из актуальных задач является разработка программного обеспечения для учёта посещаемости лекционных занятий, что позволяет преподавателям и администрациям учебных заведений оперативно получать и анализировать данные.

Традиционные методы учёта посещаемости, такие как ручная регистрация студентов в журнале, имеют ряд недостатков: они трудоёмки, подвержены человеческому фактору и не всегда позволяют оперативно анализировать данные. Внедрение специализированных программных решений позволяет устранить эти проблемы, автоматизируя процесс учёта, сохраняя данные в цифровом виде и предоставляя инструменты для их анализа.

Целью данной курсовой работы является разработка программного обеспечения для учёта посещаемости лекционных занятий, включающего функционал регистрации студентов, отметки их присутствия, а также формирования аналитических отчётов. Программа должна быть удобной в использовании, надёжной и легко интегрироваться в существующий образовательный процесс.

В рамках работы будут исследованы современные подходы к разработке программного обеспечения, изучены требования к системе, разработан интерфейс программы, а также реализованы основные функции. Результаты работы могут быть использованы для автоматизации учёта посещаемости в учебных заведениях и повышения эффективности образовательного процесса.

# **ПОСТАНОВКА И ОПИСАНИЕ ЗАДАЧИ**

Тема курсовой работы: разработать приложение посещение лекционных занятий с возможностью визуализации и сохранением данных. Программный код документировать. Сформировать с помощью утилиты javadoc html-файл, документирующий проект. Приложение скомпоновать в единый jar-файл, выполняемый на различных компьютерах.

Цель программы: получение данных о группе студентов, визуальное отображение карты студента (обозначение присутствия) с возможностью сохранения данных о посещений лекции.

Исходные данные (информация о студенте): фамилия, имя, отчество, группа, даты посещений.

Выходные данные: список студентов с отмеченными дата присутствия и отсутствия на лекционных занятиях.

Технологии, используемые в работе:

1. Операционная система Windows 11;
2. Язык программирования Java;
3. Платформа JavaSwing, Apache POI;
4. Среда разработки Intellij IDEA Community Edition 2024.2.

# **ТЕОРЕТИЧЕКИЙ РАЗДЕЛ**

# **Язык программирования Java**

**Java** – это высокоуровневый объектно-ориентированный язык программирования, разработанный корпорацией Sun Microsystems (ныне Oracle) в 1995 году. Java изначально создавалась с целью обеспечить кроссплатформенность программного обеспечения, то есть возможность его работы на любой операционной системе, на которой установлена виртуальная машина Java (JVM).

Java имеет следующие основные особенности:

1. *Кроссплатформенность*  
   Благодаря концепции "Напиши один раз, запускай где угодно" (Write Once, Run Anywhere, WORA), программы на Java могут выполняться на любых устройствах с установленной JVM, будь то Windows, macOS, Linux или другие платформы.
2. *Объектно-ориентированный подход*  
   Java поддерживает основные принципы объектно-ориентированного программирования (ООП), включая инкапсуляцию, наследование и полиморфизм. Это делает код более модульным, читаемым и легким для сопровождения.
3. *Надежность и безопасность*  
   Встроенные механизмы обработки исключений, управление памятью через автоматический сборщик мусора (Garbage Collector) и встроенные средства защиты от распространенных ошибок делают Java одним из самых надежных языков программирования.
4. *Большой набор стандартных библиотек*Java предоставляет разработчикам обширный API для выполнения самых различных задач: от работы с файлами и сетями до создания графических интерфейсов и взаимодействия с базами данных.
5. *Активная поддержка и развитие*  
   Java поддерживается корпорацией Oracle и имеет большое сообщество разработчиков. Регулярные обновления и улучшения делают язык актуальным и современным.
6. *Применение* Java  
   Java активно используется для разработки:
   * Настольных приложений (с помощью Java Swing, JavaFX).
   * Веб-приложений (с использованием Spring, Hibernate).
   * Мобильных приложений (на платформе Android).
   * Сложных корпоративных систем и серверного программного обеспечения.
   * В рамках курсовой работы Java выбрана как язык программирования благодаря сочетанию ее кроссплатформенности, обширной библиотечной поддержки и удобству создания графических интерфейсов и работы с файлами.

# **Среда разработки Intellij IDEA Community Edition 2024.2**

**Intellij IDEA** — это интегрированная среда разработки (IDE), созданная компанией JetBrains и предназначенная для написания, отладки и тестирования приложений на различных языках программирования, включая Java. Версия Community Edition является бесплатной и поддерживает все основные функции для разработки на Java.

**Основные возможности Intellij IDEA:**

1. *Удобный пользовательский интерфейс*  
   IDEA предоставляет интуитивно понятный интерфейс, который облегчает навигацию по проекту и ускоряет процесс разработки.
2. *Автоматизация рутинных операций*  
   Среда поддерживает автодополнение кода, рефакторинг, автоматическое форматирование и управление зависимостями через системы сборки (например, Maven, Gradle).
3. *Мощные инструменты для отладки*  
   Встроенный отладчик позволяет находить ошибки в коде, устанавливать точки останова (breakpoints) и анализировать состояние программы в процессе выполнения.
4. *Поддержка плагинов и интеграций*  
   IDEA поддерживает плагины для работы с дополнительными технологиями, такими как Apache POI, JavaFX, Git и другие, что делает ее универсальной средой для различных задач.
5. *Интеграция с системами контроля версий*  
   Встроенная поддержка Git, SVN и других систем позволяет легко управлять версиями проекта и работать в команде.
6. *Инструменты анализа и улучшения кода*  
   Intellij IDEA автоматически анализирует код, предлагает исправления и рекомендации, что способствует улучшению качества программного обеспечения.
7. *Кроссплатформенность*  
   Intellij IDEA доступна для Windows, macOS и Linux, что соответствует принципу кроссплатформенности Java.
8. *Поддержка Java Swing и Apache POI*  
   IDEA предоставляет встроенные инструменты для разработки графических интерфейсов с использованием Java Swing, а также легко интегрируется с библиотекой Apache POI для работы с файлами Microsoft Excel.

**Преимущества выбора Intellij IDEA для курсовой работы:**

* + Удобство написания и отладки кода благодаря интеллектуальным подсказкам.
  + Возможность работы с большими проектами за счет встроенных инструментов навигации и управления зависимостями.
  + Легкость интеграции с дополнительными библиотеками и технологиями, такими как Apache POI.

Использование Intellij IDEA Community Edition в сочетании с языком Java обеспечивает быструю, удобную и эффективную разработку приложения для учета посещаемости лекционных занятий, а также позволяет сосредоточиться на реализации функционала, минимизируя сложности настройки среды разработки.

# **Платформа JavaSwing**

Java Swing — это графическая библиотека для разработки настольных приложений на языке программирования Java. Она является частью Java Foundation Classes (JFC) и предоставляет набор инструментов для создания графических пользовательских интерфейсов (GUI). Swing используется для создания как простых, так и сложных интерфейсов, которые могут работать на разных операционных системах, таких как Windows, macOS и Linux, благодаря платформонезависимой природе Java.

Swing представляет собой усовершенствованный набор компонентов и виджетов, с помощью которых разработчики могут создавать приложения с графическим интерфейсом. В отличие от старой библиотеки AWT (Abstract Window Toolkit), которая предоставляла базовые элементы интерфейса, Swing включает более гибкие и функциональные компоненты. Среди них можно выделить кнопки, текстовые поля, таблицы, панели, вкладки и другие элементы.

Архитектуру Java Swing можно рассматривать как набор компонентов, которые работают внутри контейнеров и могут взаимодействовать друг с другом через события. Основной концепцией является использование компонента **JComponent**, который является основным строительным блоком для большинства элементов интерфейса в Swing.

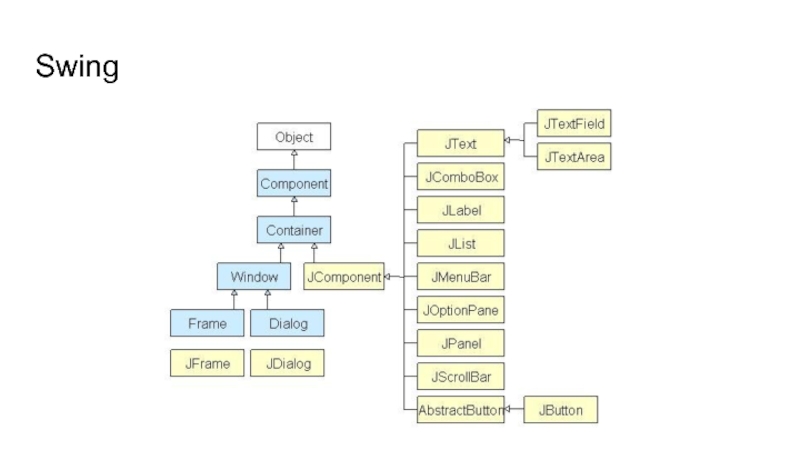


Рисунок 1 – Архитектура JavaSwing

В структуре Java Swing также существует несколько ключевых компонентов, которые играют аналогичную роль в создании графических интерфейсов. Основными компонентами являются JFrame, JPanel, и компоненты, такие как кнопки или текстовые поля. Эти элементы можно рассматривать как аналоги компонентов JavaFX, но с немного иной архитектурой и функциональностью.

1. **JFrame (основное окно приложения)**  
   Основой для создания графического интерфейса в Java Swing является класс JFrame. Это окно приложения, в котором размещаются все другие компоненты. Подобно объекту Stage в JavaFX, JFrame является контейнером для других элементов интерфейса, таких как кнопки, текстовые поля и панели. Его конкретная реализация зависит от платформы, на которой запускается приложение, и он будет представлять окно в случае настольных приложений. При запуске приложения JFrame инициализируется и отображается на экране.
2. **JPanel (контейнер для компонентов)**  
   Класс JPanel представляет собой контейнер для других компонентов и является аналогом сцены в JavaFX. Он используется для организации графических элементов внутри основного окна (JFrame). В отличие от JavaFX, где используется концепция графа сцены, в Java Swing компоненты добавляются непосредственно в контейнеры, такие как JPanel, с помощью метода add(). Панели могут быть использованы для группировки компонентов, таких как кнопки и текстовые поля, для упрощения компоновки и управления ими.
3. **JComponent (графические элементы)**  
   В Java Swing графические элементы (например, кнопки, метки, текстовые поля) представляют собой объекты, наследуемые от класса JComponent. Все эти компоненты могут быть добавлены в контейнеры, такие как JPanel или непосредственно в окно JFrame. Каждое из этих графических элементов может обрабатывать события, такие как нажатия кнопок, изменение текста или другие действия пользователя. В отличие от JavaFX, где компоненты добавляются в граф сцены, в Swing компоненты работают в более простом и прямолинейном стиле.
4. **Система компоновки (Layout Manager)**  
   В Java Swing управление размещением компонентов в контейнерах осуществляется с помощью менеджеров компоновки (Layout Managers). Эти менеджеры отвечают за автоматическое расположение компонентов в окне в зависимости от их размеров и предпочтений. Например, FlowLayout, BorderLayout, GridLayout и другие менеджеры компоновки могут использоваться для размещения компонентов в окне или панели. Это позволяет создать интерфейс, который автоматически адаптируется к размеру окна.

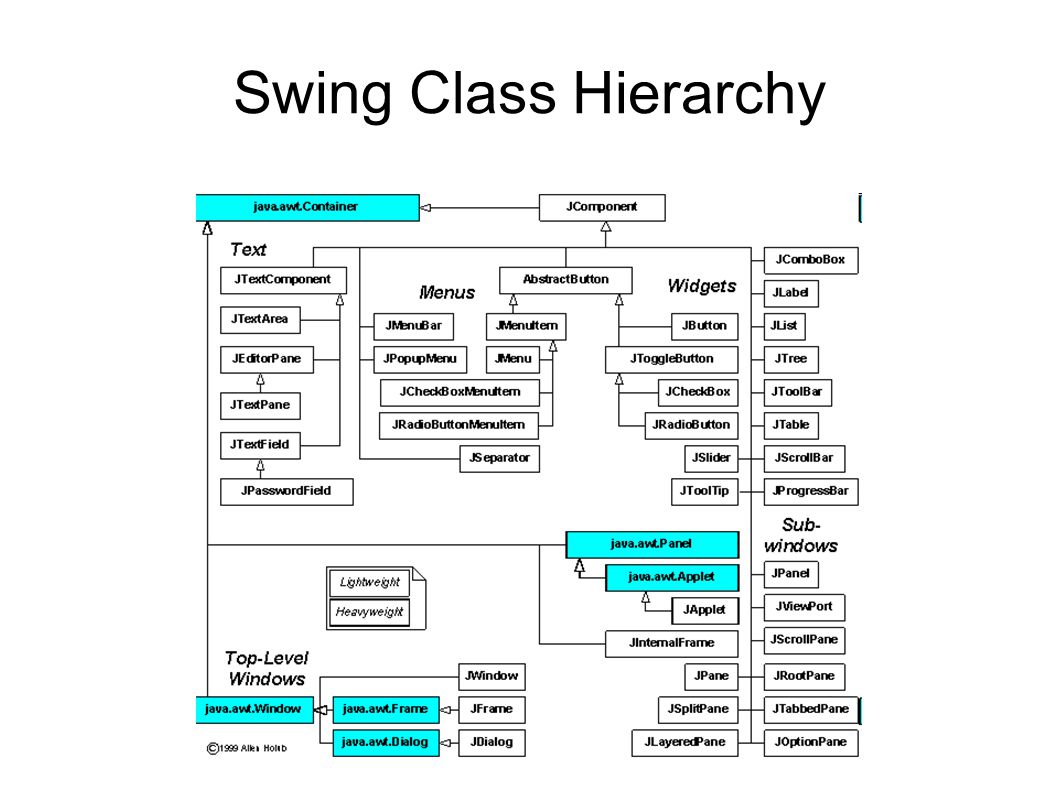


Рисунок 2 – Структура JavaSwing

# **МАТЕМАТИЧЕСКАЯ МОДЕЛЬ**

Приложение содержит:

1. *Окно входа в приложение.* В котором отображается начальная информация о проекте, реализовано начало работы с приложением и его завершение при необходимости либо при бездействии пользователя.
2. *Основное окно.* Окно для просмотра и взаимодействия с информацией, хранящейся в Excel файлах. В этом окне пользователь сможет внести данные о студенте, используя текстовые поля, добавив данные вручную, и отметить присутствие либо отсутствие студента на данной паре.
3. *Окно с информацией об авторе.* С актуальной информацией об авторе программы, контактные данные.
4. *Окно с информацией о программе.* С отображением информации о версии приложения, описание функционала программы, предоставляемой интерфейсом.

Для того, чтобы отметить студента на присутствие либо же отсутствие требуется: открыть карту студента, добавить посещение через ручной способ. Блок-схема алгоритма (рис. 2)

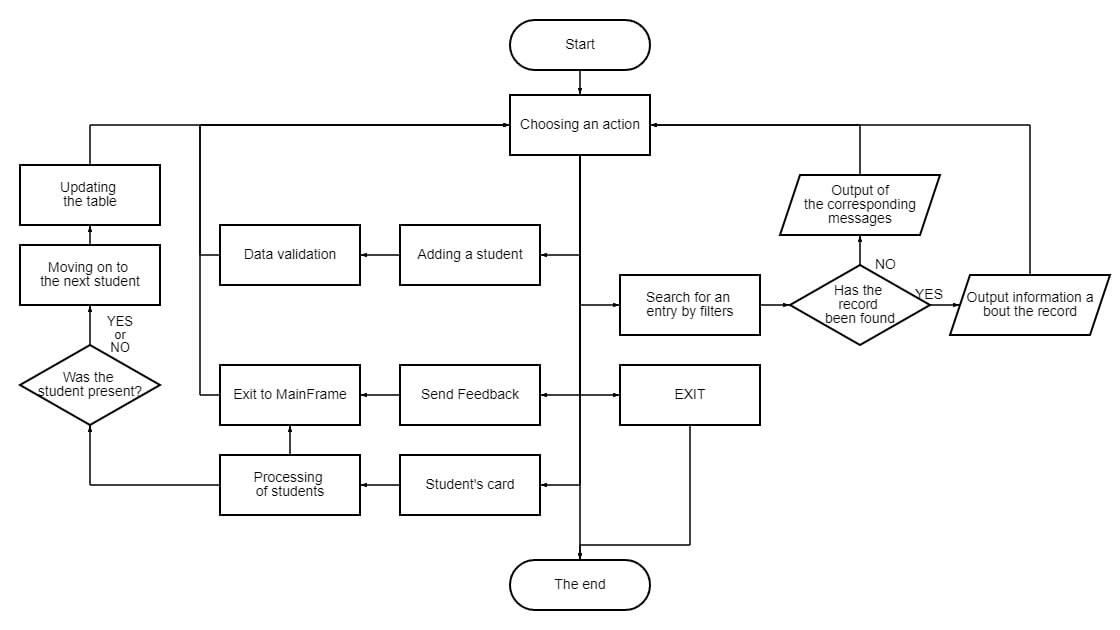


Рисунок 2 – Блок-схема алгоритма

# **ПРАКТИЧЕСКИЙ РАЗДЕЛ**

# **Описание структуры проекта**

Приложение «Посещаемость лекционных занятий» имеет следующую структуру (рис. 3).

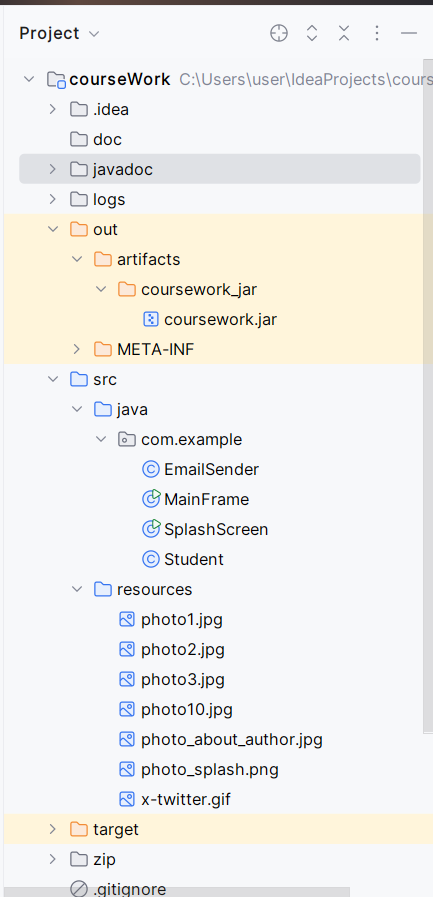
****

Рисунок 3 – Структура проекта

courseWork – папка со всеми файлами проекта;

src – папка с файлами источниками .java;

javadoc – папка с файлами JavaDoc;

coursework.jar – исполняемый файл всего проекта.

# **UML-диаграмма классов**

Унифицированный язык моделирования (UML) – это язык для специфицирования, визуализации, конструирования и документирования программных систем, а также бизнес-моделей и прочих не программных систем. UML представляет собой объединение инженерных приемов, которые ранее успешно использовались при моделировании больших и сложных систем. При решении сложных задач заблаговременное планирование и моделирование значительно упрощают программирование.

Ниже представлена UML диаграммы классов для приложения «Посещаемость лекционных занятий» (рис.4)

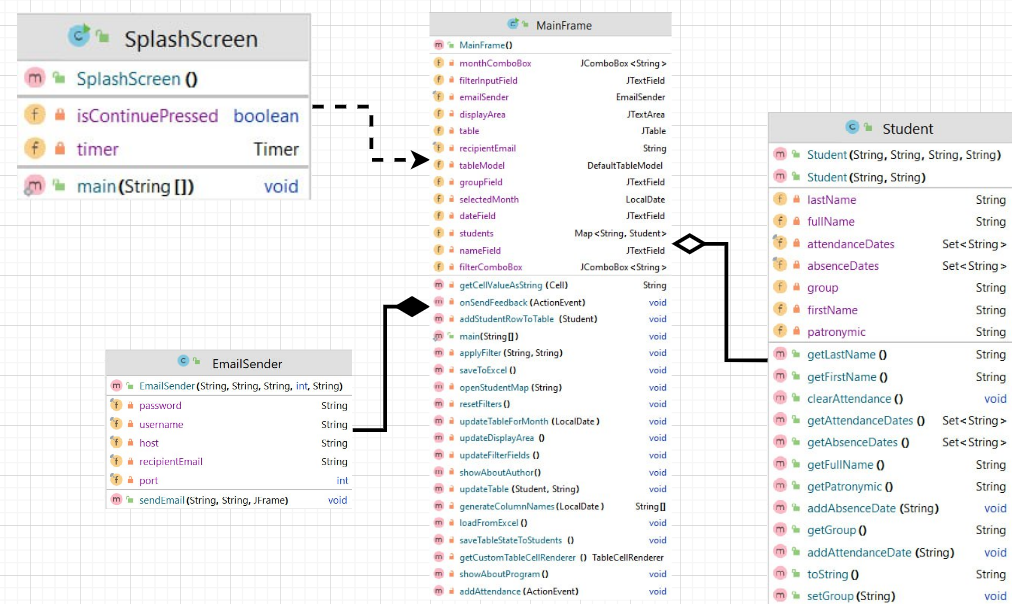
****

Рисунок 4 – UML-диаграмма классов

# **Использование Apache POI, Maven**

**Apache POI** — это проект с открытым исходным кодом, разработанный для работы с форматами файлов Microsoft Office. Он поддерживает форматы документов Word, Excel, PowerPoint, а также другие. Для работы с Excel файлы используются два основных компонента библиотеки:

* *HSSF* (Horrible Spreadsheet Format) — для работы с устаревшим форматом файлов Excel .xls (Excel 97-2003).
* *XSSF* (XML Spreadsheet Format) — для работы с более новым форматом файлов .xlsx (Excel 2007 и позже).

Apache POI позволяет выполнять следующие основные операции с файлами Excel:

1. *Чтение данных*: извлечение данных из ячеек Excel-таблиц, включая строки, числа и формулы.
2. *Запись данных:* создание новых таблиц и запись данных в ячейки.
3. *Форматирование ячеек:* настройка стилей ячеек, таких как шрифты, выравнивание текста, цвета фона и границы.
4. *Манипуляции с листами и строками:* создание новых листов, добавление и удаление строк и столбцов.
5. *Обработка формул:* использование встроенных функций Excel для выполнения вычислений в ячейках.
6. *Работа с метаданными:* возможность читать и изменять метаданные файлов, такие как автор и дата создания.

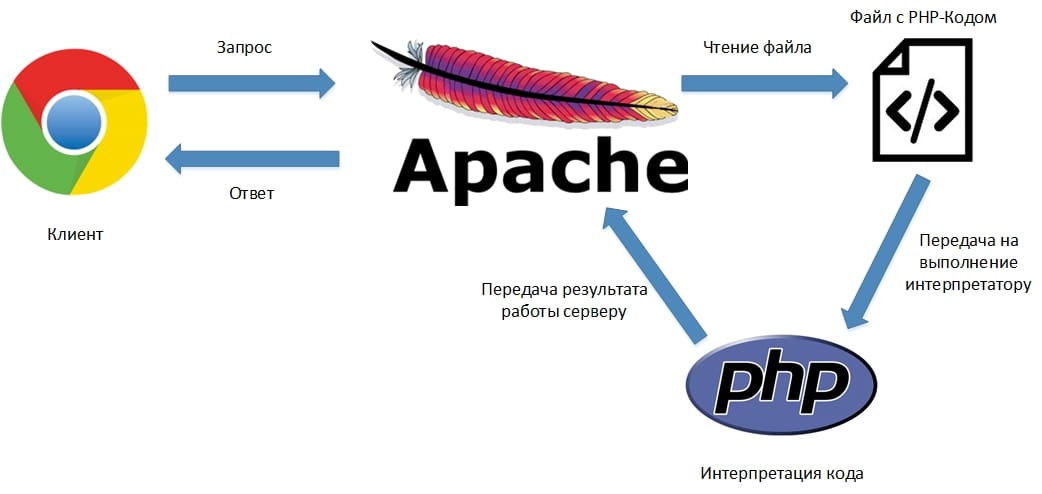


Рисунок 4.1 – Принцип работы Apache POI

**Maven** — это инструмент для управления проектами и сборки в Java, который упрощает процессы:

* Зависимости: автоматическое подключение библиотек (например, Apache POI) из централизованных репозиториев.
* Сборка: автоматическое выполнение различных этапов сборки проекта (например, компиляция, тестирование, упаковка).
* Публикация: автоматическое создание и развертывание приложения на сервере или в облаке.

Maven использует файл конфигурации pom.xml (Project Object Model), в котором описываются зависимости проекта, плагины и цели. Этот файл позволяет разработчику централизованно управлять всеми аспектами проекта.

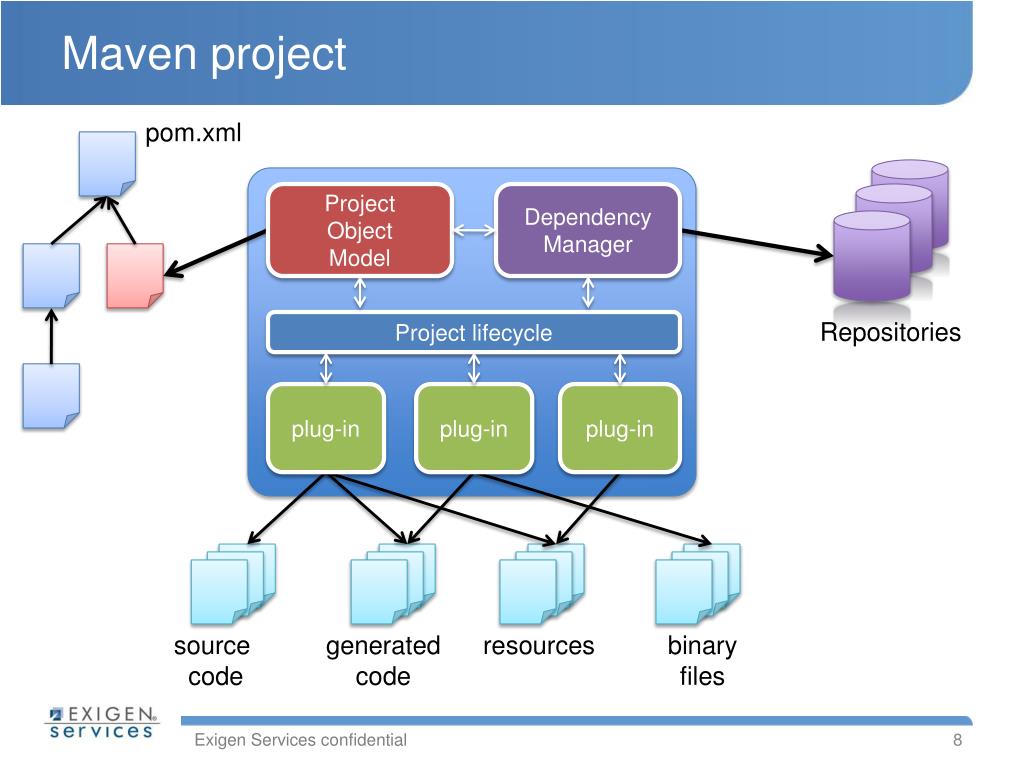


Рисунок 4.2 – Принцип работы Maven

**Использование Apache POI с Maven**

Одной из ключевых особенностей Maven является возможность управления зависимостями. Это означает, что для использования библиотеки Apache POI достаточно указать ее в конфигурационном файле pom.xml, и Maven автоматически загрузит нужные версии библиотек и настроит их для использования в проекте.

**Как подключить Apache POI в проект с использованием Maven:**

1. **Добавление зависимостей в pom.xml**: Для того чтобы использовать Apache POI в Java-проекте, необходимо добавить соответствующие зависимости в файл pom.xml. Maven скачает необходимые библиотеки и их зависимости из центрального репозитория Maven.
2. **Автоматическая загрузка зависимостей**: После того как зависимости указаны в pom.xml, Maven автоматически скачает все необходимые библиотеки при сборке проекта. Это упрощает процесс добавления новых библиотек и их версий в проект.
3. **Обновление зависимостей**: Maven также автоматически отслеживает версии зависимостей и их обновления. Разработчику достаточно изменить номер версии библиотеки в pom.xml, и при следующей сборке будет использована новая версия.

# **Реализация графических окон**

Проект представлен блок-схемой или, как ее по-другому называют, бизнес-логикой переключения окон проекта, взаимосвязанных между собой:

* Splash Screen – стартовое окно приложения,
* Main Window – основное окно для работы с приложением,
* About program – окно, предоставляющее информацию о программе
* About Author – окно, предоставляющее информацию об авторе (рис. 5):

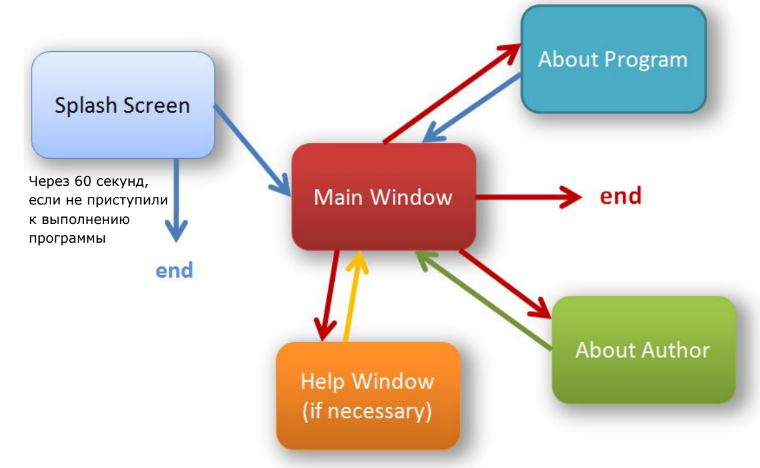


Рисунок 5 – Бизнес-логика переключения графических окон

# **Стартовое окно**

После запуска программы перед пользователем открывается стартовое окно. В стартовом окне отображена следующая информация: название учебного учреждения, кафедры, дисциплины, темы проекта, имени студента, выполнившего работу, и преподавателя, являющегося научным руководителем. Пользователь может выйти из программы, нажав кнопку «Выйти» или продолжить работу, нажав кнопку «Далее» (рис. 6)

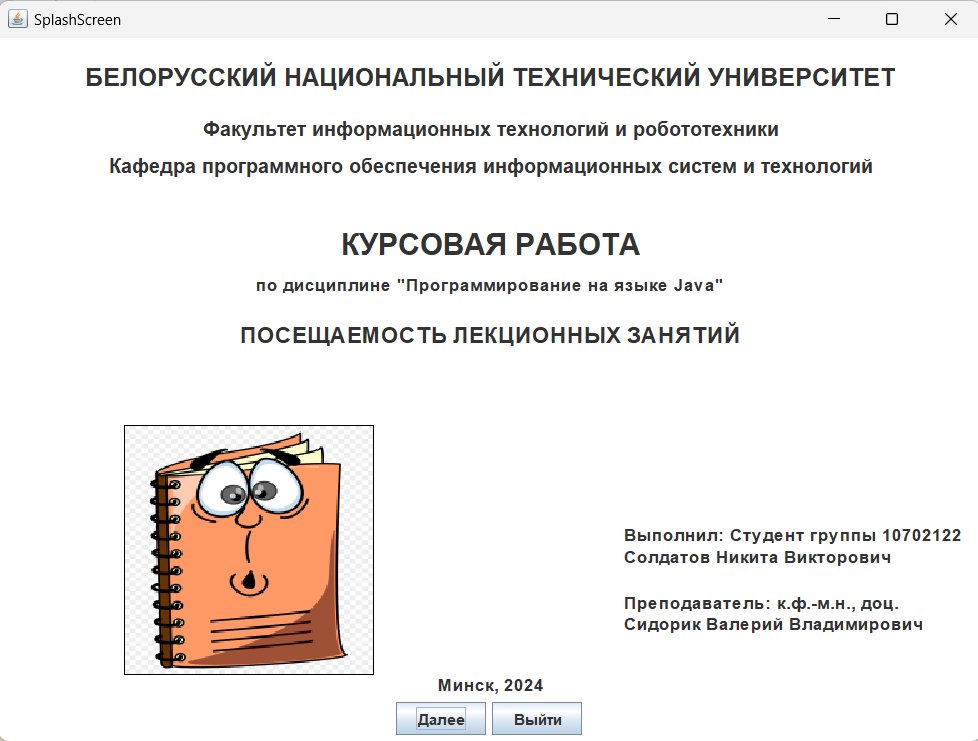
****

Рисунок 6 – Стартовое окно

# **Главное окно**

При нажатии на кнопку «Далее» в основном окне перед пользователем открывается рабочее окно, содержащее меню с пунктами «File», имеющего всплывающую вкладку «Exit», «Information», имеющее всплывающие вкладки «О программе», «Об авторе» (рис.7) и «Help», имеющее всплывающую вкладку Send Feedback.

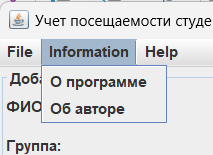


Рисунок 7 – Меню

С помощью элемента управления созданы различные кнопки, чтобы взаимодействовать с программой и списком студентов (предварительно загруженных) (рис.8)

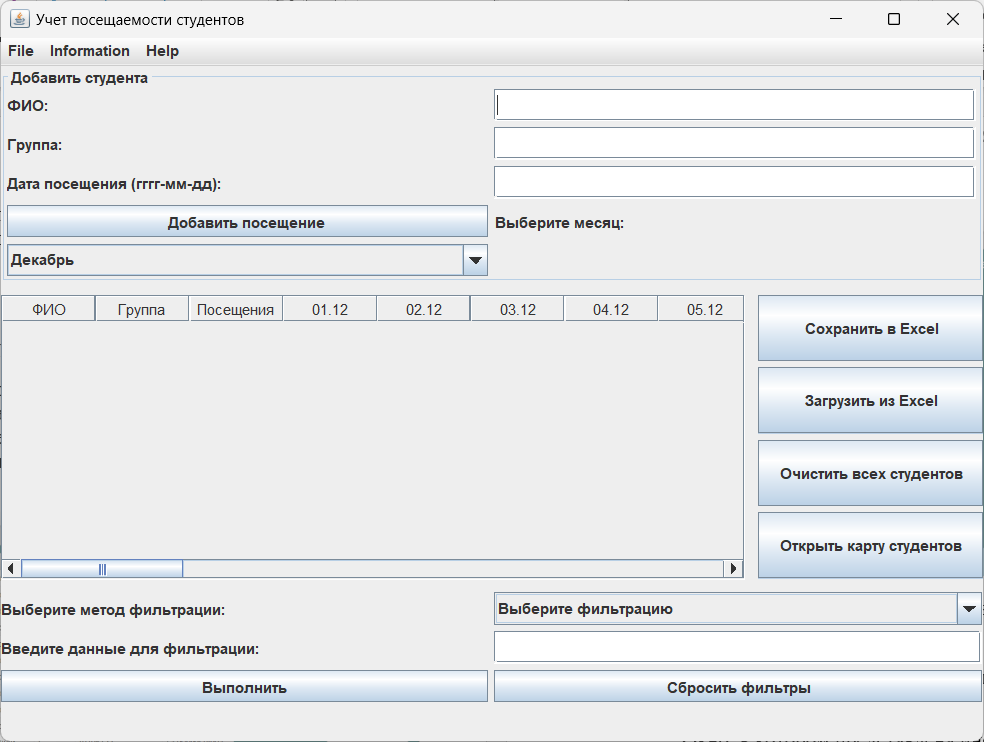


Рисунок 8 – Главное окно

С помощью кнопки «Добавить посещение» и соответствующих полей можно добавлять студентов, а также дат посещений ими занятий (рис.9).

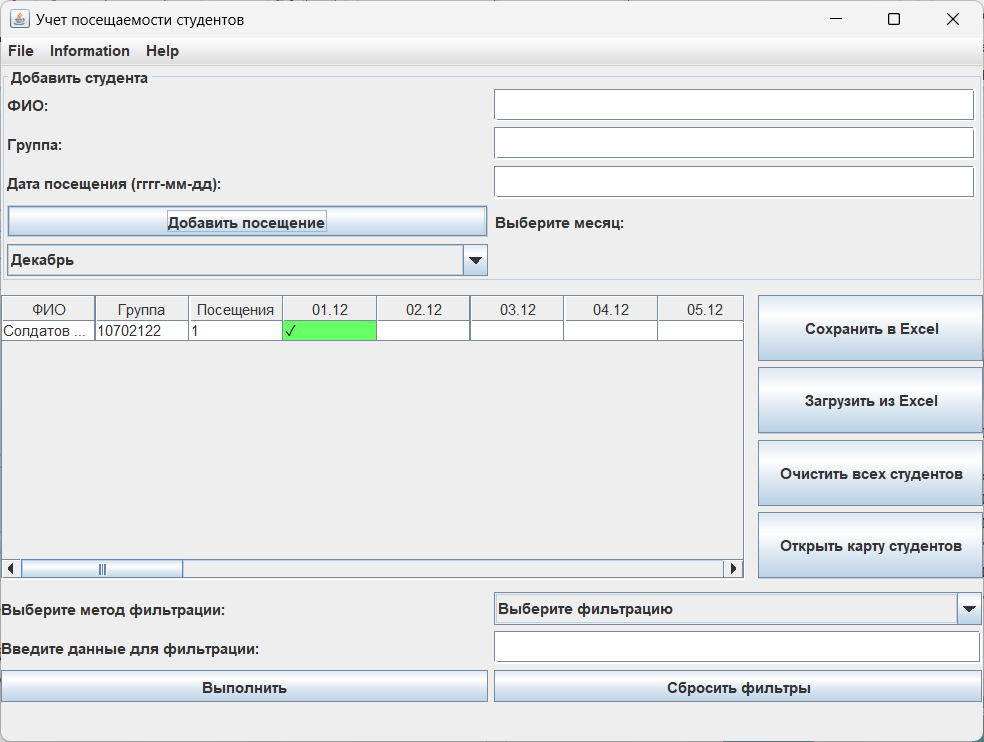


Рисунок 9 – Добавление студента

При нажатий на кнопку «Загрузить из Excel» предоставляется выбор Excel файла из любой директорий компьютера. После загрузки данных из Excel файлов можно как фильтровать данные так и добавлять (рис.10)

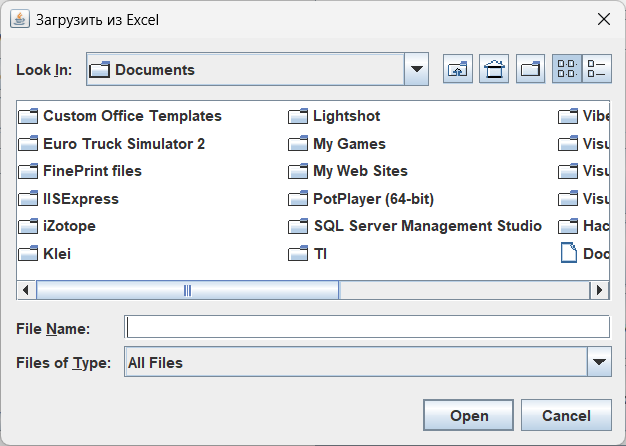


Рисунок 10 – Загрузка информаций из Excel

# **Окно карты студента**

Окно «Карта студента» с помощью которого можно отмечать присутствие либо же отсутствие студента по датам (рис.11).

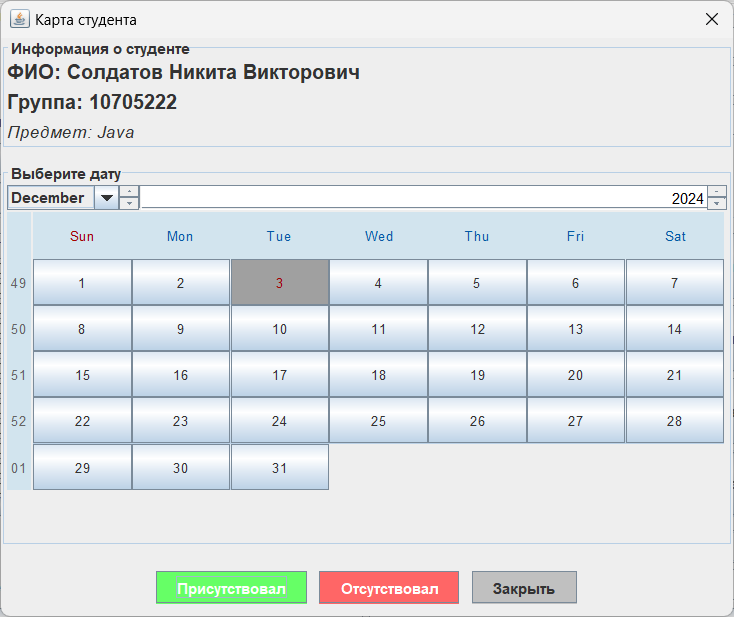


Рисунок 11 – Окно «Карта студента»

# **Окно с информацией об авторе**

Окно «Об авторе» (рис. 12) содержит основную информацию об авторе проекта: номер группы, имя, фамилию, отчество, фотографию автора, а также адрес его электронной почты.

Если данной информационное окно открыто, то пользователь не может взаимодействовать с другими окнами. Чтобы вернуться обратно к меню, нужно нажать на кнопку «Назад», расположенную в окне.

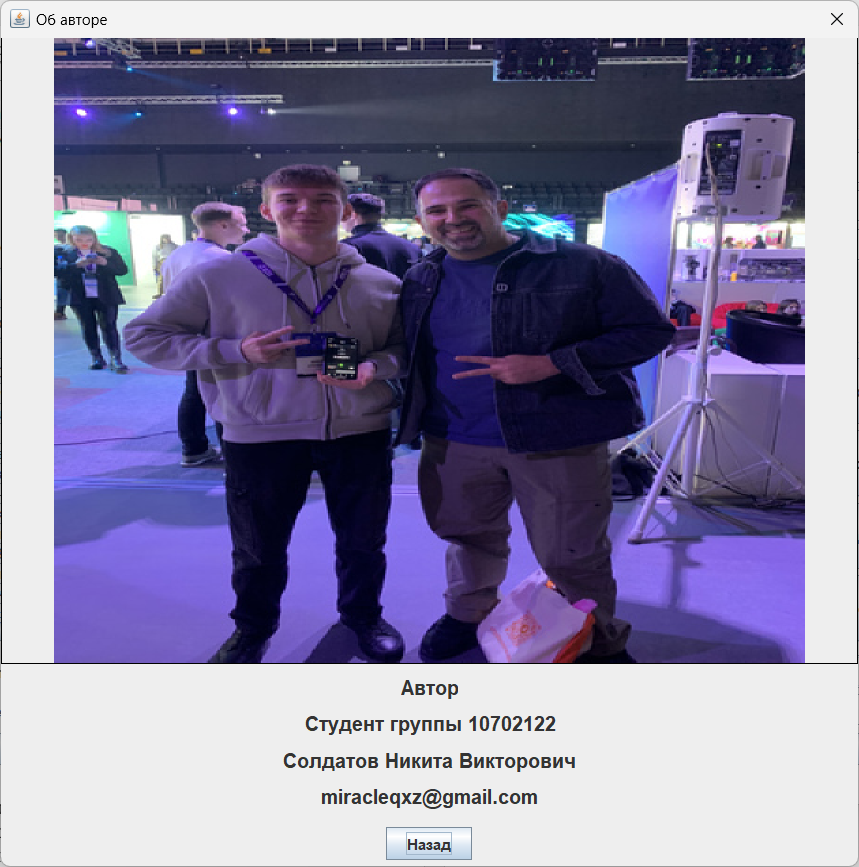


Рисунок 12 – Окно «Об авторе»

# **Окно с информацией о программе**

Окно «О программе» (рис. 13) содержит основную информацию о возможностях программы и версию приложения. Если данной информационное окно открыто, то пользователь не может взаимодействовать с другими окнами. Чтобы вернуться обратно к меню, нужно нажать на кнопку «Выход», расположенную в окне.

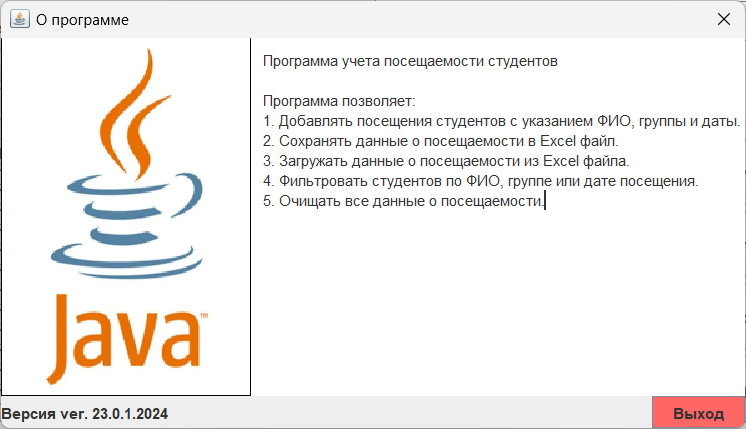


Рисунок 13 – Окно «О программе»

# **Утилита JavaDoc**

С помощью утилиты javadoc, входящей в состав JDK, комментарий документации можно извлекать и помещать в НТМL файл. Утилита javadoc позволяет вставлять HTML тэги и использовать специальные ярлыки (дескрипторы) документирования. НТМL тэги заголовков не используют, чтобы не нарушать стиль файла, сформированного утилитой (рис. 14)

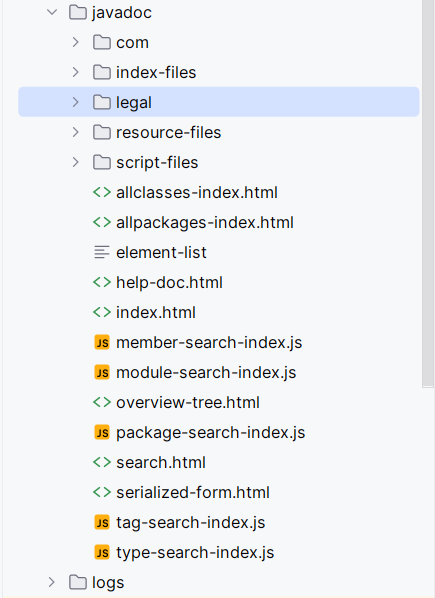
****

Рисунок 14 – Сгенерированные утилитой JavaDoc файлы

# **Сборка jar-файла**

Весь проект объединяем в единый jar-файл, который можно запускать на любом компьютере (рис.15).

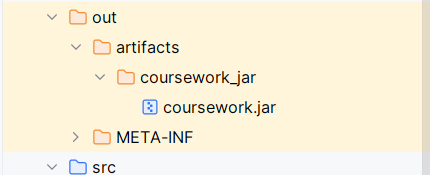


Рисунок 15 – Сгенерированный jar-файл

# **РЕЗУЛЬТАТЫ РАБОТЫ**

В основном окне доступны данные для ввода такие как ФИО, номер группы и дата посещения студента. В каждом поле предусмотрена валидация введенного текста (рис. 16).

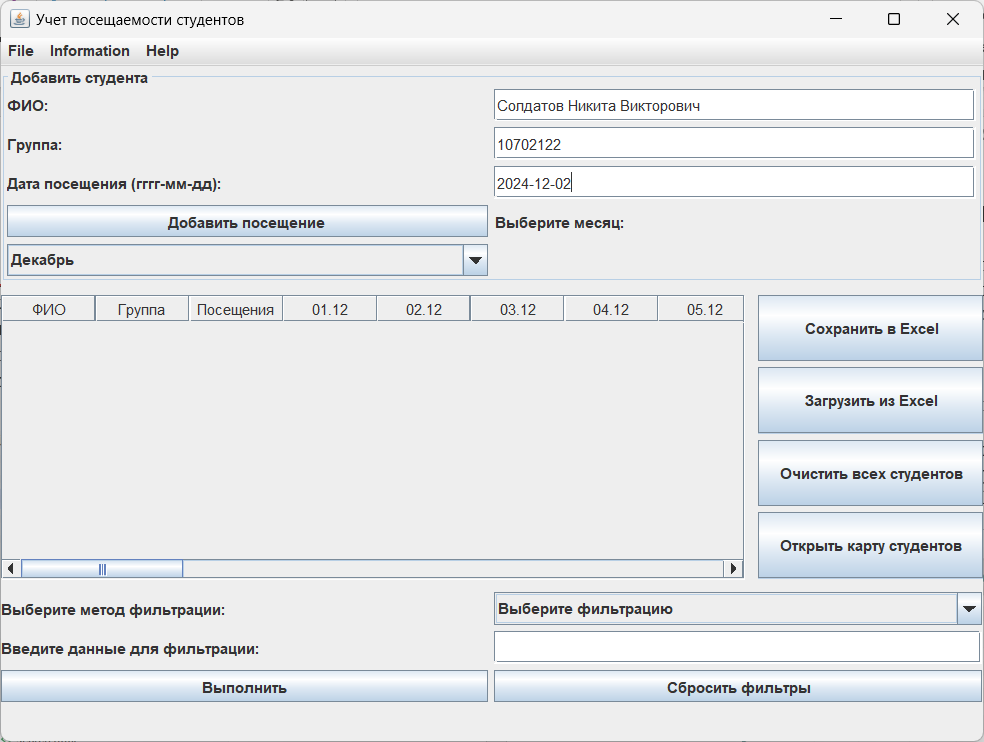


Рисунок 16 – Ввод данных студента

После нажатия на кнопку «Добавить», программа записывает данные о студенте в текстовую панель, а также обновляет список на экране (рис.17).

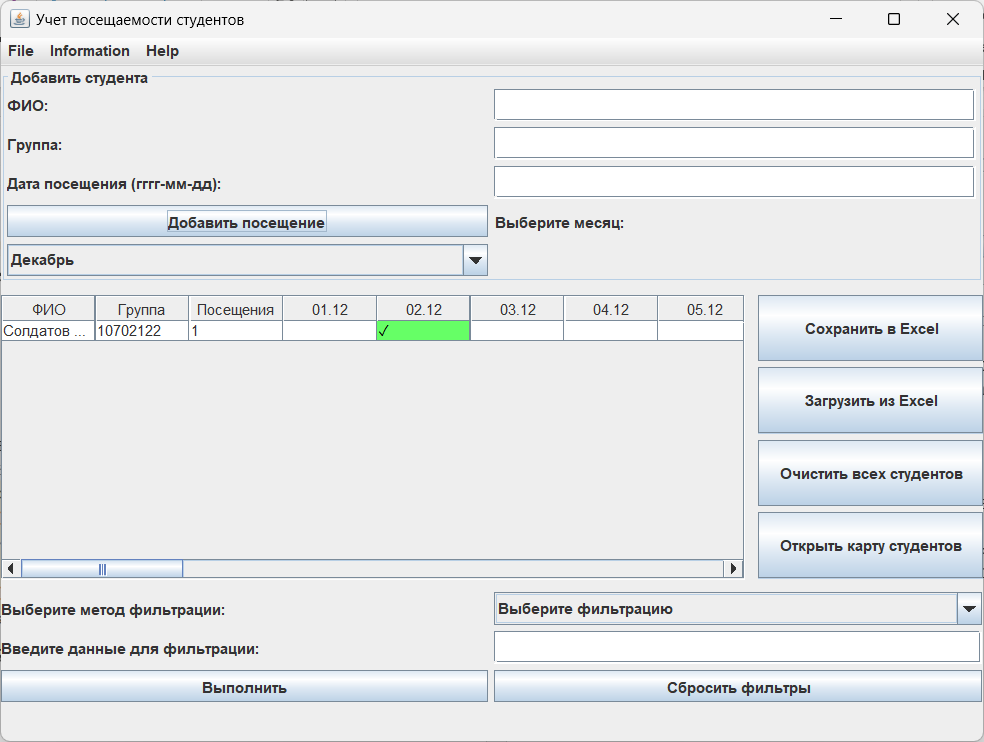


Рисунок 17 – Добавление студента в список

При вводе неверных данных появится сообщение об ошибке (рис.18)

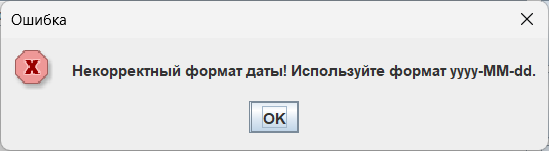


Рисунок 18 – Валидация ввода данных

При попытке добавить студента, не введя все его данные, появится сообщение об ошибке (рис.19)

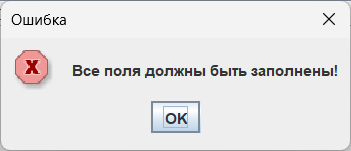


Рисунок 19– Попытка неполного ввода данных

При нажатии на кнопку «Загрузить из Excel» предоставляется выбор директории и файла с которого будет загружена информация о группе студентов(рис.20).

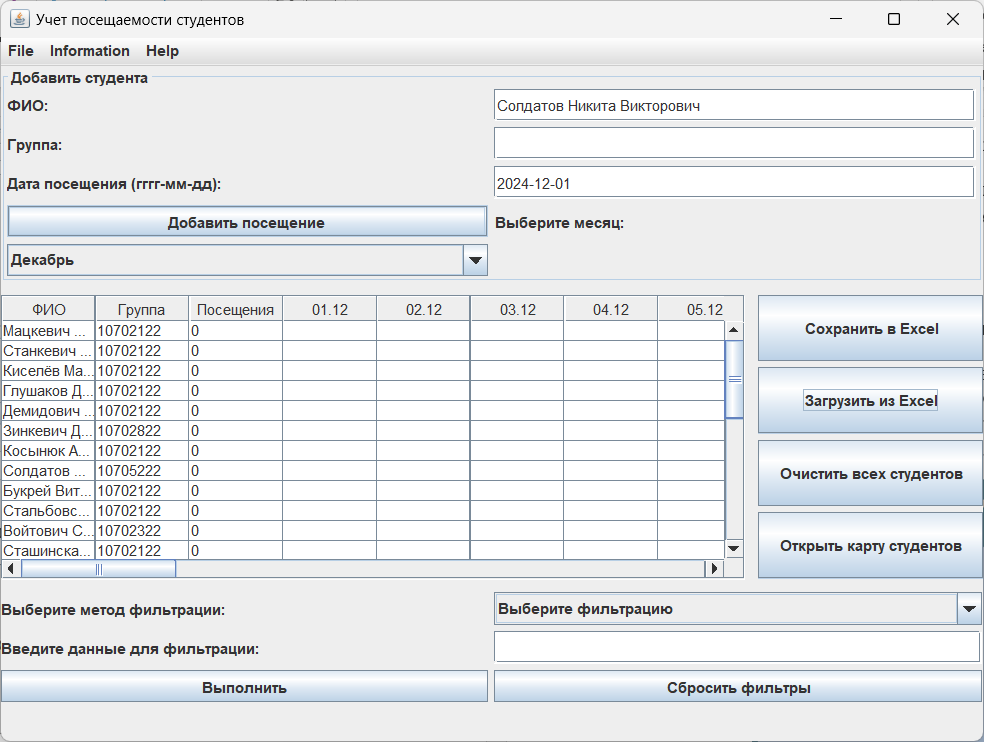


Рисунок 20 – Вывод списка группы студентов

Пользователь может с помощью карты студента отметить всю группу студентов (рис.21)

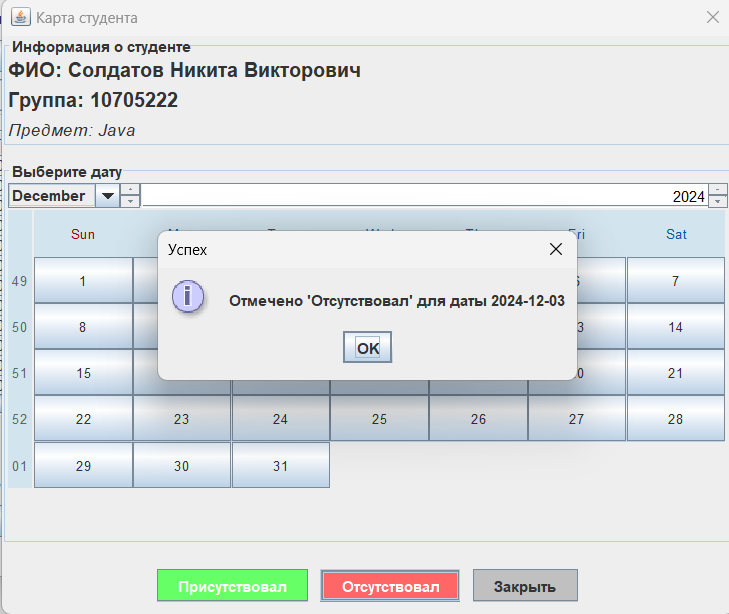


Рисунок 21 – Отметка студентов с помощью карты студента

Пользователь может фильтровать данные из таблицы с помощью разработанных фильтров «Фильтровать по дате», «Фильтровать по группе», «Фильтровать по ФИО», а также сброс при нажатии на кнопку «Сбросить фильтры» (рис.22-25)

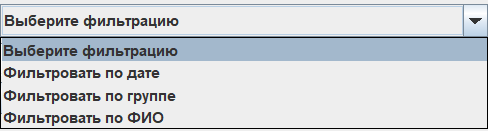


Рисунок 22 – Выборка фильтров таблицы студентов

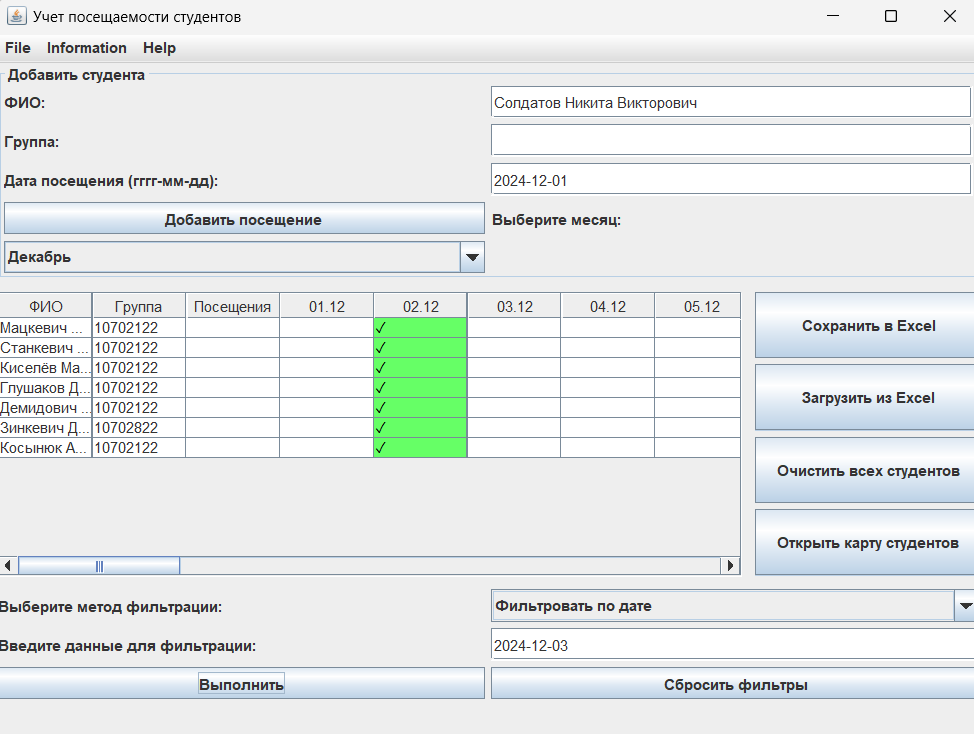


Рисунок 23 – Фильтровка по дате

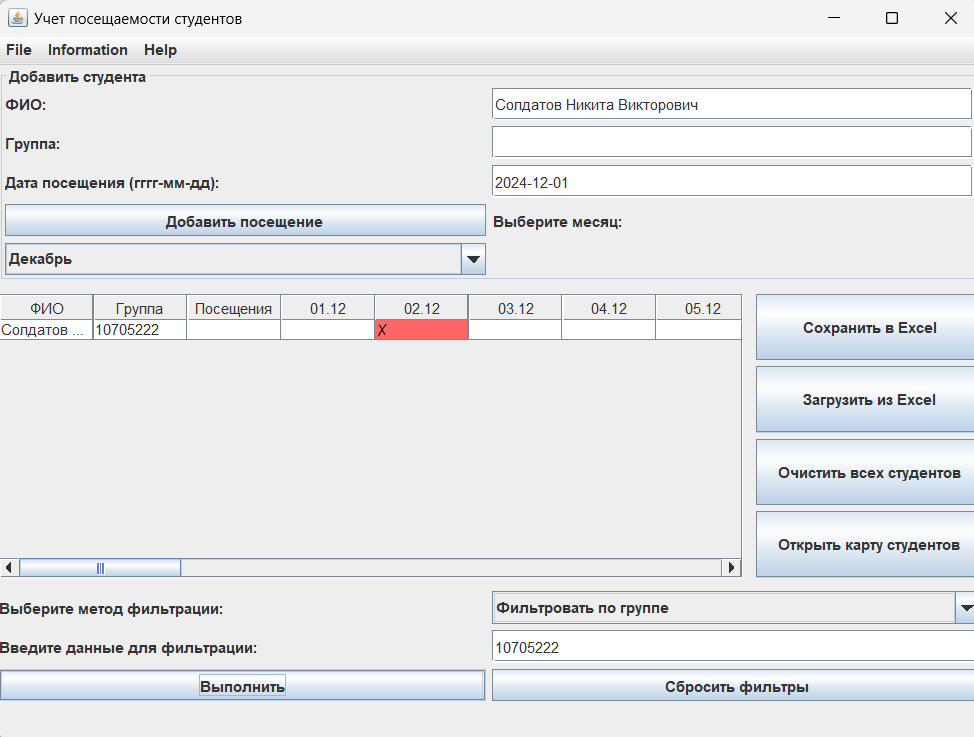


Рисунок 24 – Фильтровка по группе

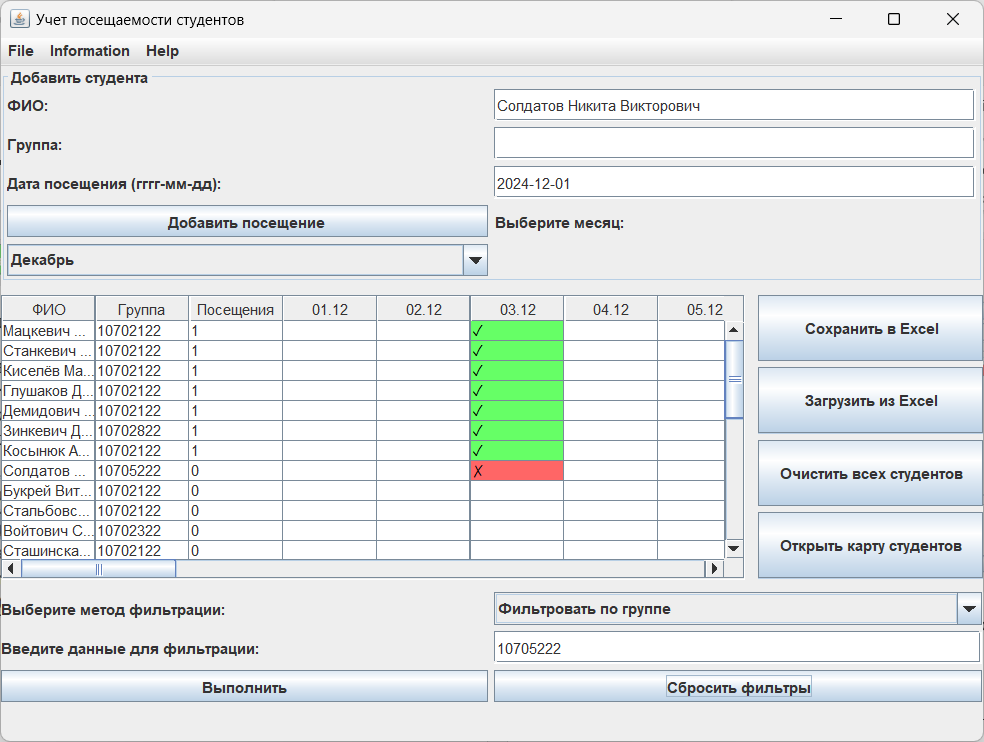


Рисунок 25 – Сброс фильтров

При нажатии на кнопку «Сохранить в Excel» пользователю дается возможность сохранить Excel файл с таблицей посещений лекционных занятий в определенную директорию, Excel файл создается автоматически (рис. 26)

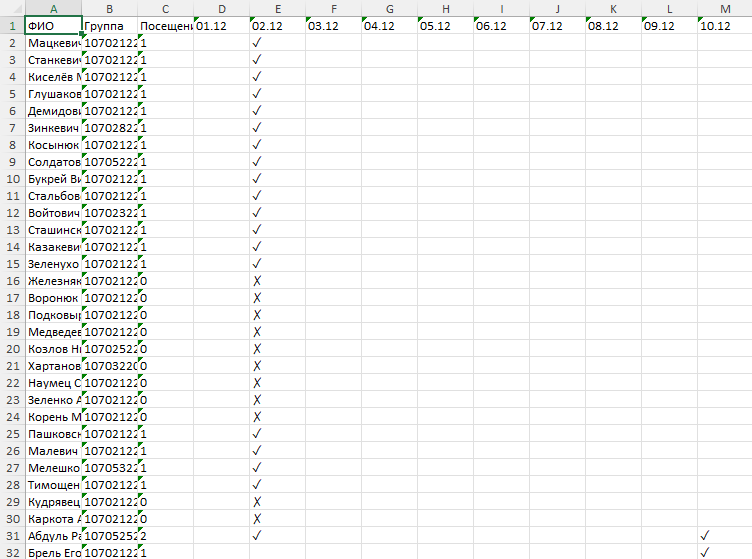


Рисунок 26 – Сохранение Excel файла

# **ЗАКЛЮЧЕНИЕ**

В рамках курсового проекта *"Посещение лекционных занятий"* была разработана программа на языке программирования Java с использованием графического интерфейса на базе *Swing*. Основная цель проекта заключалась в создании удобного инструмента для учета посещаемости студентов, позволяющего регистрировать, анализировать и сохранять данные.

В процессе работы были выполнены следующие задачи:

1. Разработан интуитивно понятный интерфейс с использованием *Java Swing*, обеспечивающий легкость ввода данных, их отображение и фильтрацию.
2. Реализован функционал для валидации вводимых данных:
   * Проверка корректности формата ФИО, номера группы и даты.
   * Исключение некорректных данных, что обеспечивает надежность работы программы.
3. Интегрирована поддержка работы с Excel-файлами с помощью библиотеки *Apache POI*:
   * Возможность импорта данных из Excel для загрузки ранее созданных записей.
   * Экспорт данных в Excel для последующего использования и анализа.
4. Для управления зависимостями, автоматизации сборки и упрощения подключения библиотек был использован инструмент *Maven*.

Программа предоставляет следующие ключевые возможности:

* Добавление и отображение информации о посещаемости студентов.
* Фильтрация данных по различным критериям, включая ФИО, номер группы и дату.
* Экспорт и импорт данных для работы с внешними таблицами.
* Отправка обратной связи по электронной почте через встроенный функционал.

Данный проект демонстрирует возможности *Java* для создания полноценных десктопных приложений, а также преимущества интеграции с популярными библиотеками и инструментами (такими как *Apache POI* и *Maven*).

В заключение можно отметить, что поставленные задачи успешно выполнены, а разработанная программа может быть полезна как в учебной, так и в профессиональной деятельности. Возможным направлением дальнейшего развития проекта является добавление следующих функций:

* Построение отчетов и графиков для анализа посещаемости.
* Интеграция с базами данных для хранения больших объемов данных.
* Реализация более сложных алгоритмов анализа и прогнозирования посещаемости.

Проект подтверждает актуальность использования технологий Java для создания эффективных и удобных решений, что делает его ценным опытом в профессиональном развитии разработчика.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Кей С.Хорстман – Java. Библиотека профессионала.
2. <https://drive.google.com/drive/shared-with-me> - Методические указания к оформлению курсовой работы, В.В. Сидорик.
3. Курс по JavaSwing.[Электронный ресурс] – Режим доступа: https://progoschool.ru/java/java-swing/, свободный.
4. <https://javadevblog.com/sozdaem-yspolnyaemyj-jar-v-intellij-idea.html> - Cоздание jar-файлов.
5. Документация языка программирования Java [Электронный ресурс] – Режим доступа: https://docs.oracle.com/en/java/, свободный.
6. Видео ресурс по JavaSwing [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=VsVw5ow8Ynw&list=PL786bPIlqEjTN-Sy8uMLQ6s85qNxP1hAp, свободный.

# **ПРИЛОЖЕНИЕ 1**

**MainFrame.java**

package com.example;  
  
  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableCellRenderer;  
import javax.swing.table.DefaultTableModel;  
import javax.swing.table.TableCellRenderer;  
import java.awt.Color;  
import java.awt.Font;  
import java.awt.event.ActionListener;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.time.LocalDate;  
import java.util.\*;  
import java.util.List;  
import com.toedter.calendar.JCalendar;  
  
  
  
*/\*\*  
 \* Главный класс приложения для учета посещаемости студентов.  
 \* Этот класс расширяет {@link JFrame} и реализует графический интерфейс  
 \* для добавления, отображения и фильтрации информации о студентах и их посещаемости.  
 \*  
 \* Включает функционал:  
 \* <ul>  
 \* <li>Добавление посещений студентов с указанием ФИО, группы и даты посещения;</li>  
 \* <li>Отображение таблицы студентов;</li>  
 \* <li>Фильтрация студентов по различным критериям (дата, группа, ФИО);</li>  
 \* <li>Экспорт и импорт данных из Excel;</li>  
 \* <li>Отправка обратной связи по email.</li>  
 \* </ul>  
 \*  
 \* @author Soldatov N. V.  
 \* @version 22.0.2  
 \*/*public class MainFrame extends JFrame {  
  
 */\*\*  
 \* Список студентов, где ключ - это ФИО студента, а значение - объект {@link Student}.  
 \*/* private Map<String, Student> students = new HashMap<>();  
  
 */\*\*  
 \* Область для отображения информации о текущем состоянии.  
 \*/* private JTextArea displayArea;  
  
 */\*\*  
 \* Поля для ввода информации о студенте.  
 \*/* private JTextField nameField, groupField, dateField;  
  
 */\*\*  
 \* Таблица для отображения списка студентов.  
 \*/* private JTable table;  
  
 */\*\*  
 \* Модель таблицы для отображения данных студентов.  
 \*/* private DefaultTableModel tableModel;  
  
 */\*\*  
 \* Комбинированное поле для выбора фильтра.  
 \*/* private JComboBox<String> filterComboBox;  
  
 */\*\*  
 \* Поле ввода для ввода данных фильтрации.  
 \*/* private JTextField filterInputField;  
  
 */\*\*  
 \* Отправитель email для обратной связи.  
 \*/* private final EmailSender emailSender;  
  
 */\*\*  
 \* Электронная почта получателя для обратной связи.  
 \*/* private final String recipientEmail = "miracleqxz@gmail.com";  
  
 */\*\*  
 \* Комбинированное поле для выбора месяца.  
 \*/* private JComboBox<String> monthComboBox;  
  
 */\*\*  
 \* Текущий выбранный месяц для отображения посещаемости студентов.  
 \*/* private LocalDate selectedMonth = LocalDate.*now*().withDayOfMonth(1);  
  
 */\*\*  
 \* Конструктор, инициализирующий интерфейс, компоненты и действия.  
 \*/* public MainFrame() {  
 setTitle("Учет посещаемости студентов");  
 setSize(800, 600); *// Уменьшенный размер окна* setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
  
 *// Панель ввода с более компактным GridLayout* JPanel inputPanel = new JPanel(new GridLayout(5, 2, 5, 5)); *// 4 строки, 2 столбца* inputPanel.setBorder(BorderFactory.*createTitledBorder*("Добавить студента"));  
  
 *// Поле ввода ФИО* inputPanel.add(new JLabel("ФИО:"));  
 nameField = new JTextField(15);  
 inputPanel.add(nameField);  
  
 *// Поле ввода группы* inputPanel.add(new JLabel("Группа:"));  
 groupField = new JTextField(15);  
 inputPanel.add(groupField);  
  
 *// Поле ввода даты* inputPanel.add(new JLabel("Дата посещения (гггг-мм-дд):"));  
 dateField = new JTextField(15);  
 inputPanel.add(dateField);  
  
 *// Кнопка добавления* JButton addButton = new JButton("Добавить посещение");  
 addButton.addActionListener(this::addAttendance);  
 inputPanel.add(addButton);  
  
 *// Выбор месяца* inputPanel.add(new JLabel("Выберите месяц:"));  
 String[] months = {"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};  
 monthComboBox = new JComboBox<>(months);  
 monthComboBox.setSelectedIndex(selectedMonth.getMonthValue() - 1);  
 monthComboBox.addActionListener(e -> {  
 selectedMonth = LocalDate.*now*().withMonth(monthComboBox.getSelectedIndex() + 1).withDayOfMonth(1);  
 updateTableForMonth(selectedMonth); *// Обновляем таблицу* });  
 inputPanel.add(monthComboBox);  
  
 *// Инициализация таблицы* String[] columnNames = generateColumnNames(selectedMonth);  
 tableModel = new DefaultTableModel(columnNames, 0);  
 table = new JTable(tableModel);  
 table.setAutoResizeMode(JTable.*AUTO\_RESIZE\_OFF*);  
 JScrollPane scrollPane = new JScrollPane(table);  
  
 *// Панель кнопок с уменьшенными кнопками* JPanel buttonPanel = new JPanel(new GridLayout(4, 1, 5, 5)); *// Изменено с 3 на 4 для нового места* JButton saveExcelButton = new JButton("Сохранить в Excel");  
 saveExcelButton.addActionListener(e -> saveToExcel());  
 buttonPanel.add(saveExcelButton);  
  
 JButton loadExcelButton = new JButton("Загрузить из Excel");  
 loadExcelButton.addActionListener(e -> loadFromExcel());  
 buttonPanel.add(loadExcelButton);  
  
 JButton clearButton = new JButton("Очистить всех студентов");  
 clearButton.addActionListener(e -> {  
 students.clear();  
 updateDisplayArea();  
 });  
 buttonPanel.add(clearButton);  
  
 *// Добавляем новую кнопку для открытия карты студентов* JButton openStudentMapButton = new JButton("Открыть карту студентов");  
 openStudentMapButton.addActionListener(e -> {  
 String subject = JOptionPane.*showInputDialog*(this, "Введите предмет:");  
 if (subject != null && !subject.trim().isEmpty()) {  
 openStudentMap(subject); *// Вызов ранее реализованного метода* } else {  
 JOptionPane.*showMessageDialog*(this, "Предмет не указан!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
 buttonPanel.add(openStudentMapButton);  
  
 *// Панель фильтрации* JPanel filterPanel = new JPanel(new GridLayout(4, 1, 5, 5));  
 String[] filterOptions = {"Выберите фильтрацию", "Фильтровать по дате", "Фильтровать по группе", "Фильтровать по ФИО"};  
 filterComboBox = new JComboBox<>(filterOptions);  
 filterComboBox.addActionListener(e -> updateFilterFields());  
 filterPanel.add(new JLabel("Выберите метод фильтрации:"));  
 filterPanel.add(filterComboBox);  
  
 filterInputField = new JTextField(15);  
 filterPanel.add(new JLabel("Введите данные для фильтрации:"));  
 filterPanel.add(filterInputField);  
  
 *// Кнопка выполнения фильтрации* JButton applyFilterButton = new JButton("Выполнить");  
 applyFilterButton.addActionListener(e -> {  
 String selectedFilter = (String) filterComboBox.getSelectedItem();  
 String input = filterInputField.getText().trim();  
 applyFilter(selectedFilter, input);  
 });  
 filterPanel.add(applyFilterButton);  
  
 *// Добавляем кнопку "Сбросить фильтры"* JButton resetFiltersButton = new JButton("Сбросить фильтры");  
 resetFiltersButton.addActionListener(e -> resetFilters());*// Возвращаем исходные данные* filterPanel.add(resetFiltersButton);  
  
 *// Основная панель* JPanel mainPanel = new JPanel(new BorderLayout(10, 10));  
 mainPanel.add(inputPanel, BorderLayout.*NORTH*);  
 mainPanel.add(scrollPane, BorderLayout.*CENTER*);  
 mainPanel.add(buttonPanel, BorderLayout.*EAST*);  
 mainPanel.add(filterPanel, BorderLayout.*SOUTH*);  
  
 *// Создаем меню* JMenuBar menuBar = new JMenuBar();  
 setJMenuBar(menuBar);  
  
 *// Меню "Файл" с всплывающим меню* JMenu fileMenu = new JMenu("File");  
 menuBar.add(fileMenu);  
  
 *// Добавляем пункт меню "Exit" в меню "File"* JMenuItem exitItem = new JMenuItem("Exit");  
 exitItem.addActionListener(e -> {  
 System.*exit*(0); *// Завершаем программу* });  
 fileMenu.add(exitItem);  
  
 *// Меню "Информация"* JMenu infoMenu = new JMenu("Information");  
 menuBar.add(infoMenu);  
  
 *// Пункт меню "О программе"* JMenuItem aboutProgramItem = new JMenuItem("О программе");  
 aboutProgramItem.addActionListener(e -> showAboutProgram());  
 infoMenu.add(aboutProgramItem);  
  
 *// Пункт меню "Об авторе"* JMenuItem aboutAuthorItem = new JMenuItem("Об авторе");  
 aboutAuthorItem.addActionListener(e -> showAboutAuthor());  
 infoMenu.add(aboutAuthorItem);  
  
 *// Меню "Помощь" (пока пустое)* JMenu helpMenu = new JMenu("Help");  
 menuBar.add(helpMenu);  
  
 *// Пункт меню для отправки письма* JMenuItem sendFeedbackMenuItem = new JMenuItem("Send Feedback");  
 emailSender = new EmailSender("nikitikk5@yandex.ru", "muvnquyaqebtolhk", "smtp.yandex.ru", 465, recipientEmail);  
 sendFeedbackMenuItem.addActionListener(this::onSendFeedback);  
 helpMenu.add(sendFeedbackMenuItem);  
  
 add(mainPanel);  
 updateDisplayArea();  
 }  
  
  
 */\*\*  
 \* Обрабатывает событие для отправки отзыва.  
 \* Создает и отображает окно для ввода письма с полями "Тема" и "Сообщение".  
 \* При отправке письма использует {@link EmailSender} для отправки.  
 \* Закрывает окно после отправки.  
 \*  
 \* @param e Событие, вызвавшее обработчик  
 \*/* private void onSendFeedback(ActionEvent e) {  
 *// Создаем и показываем окно для ввода письма* JFrame frame = new JFrame("Send Feedback");  
 frame.setSize(400, 300);  
 frame.setLayout(new BorderLayout(10, 10)); *// Добавляем отступы* frame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*); *// Закрытие только окна  
  
 // Поля ввода* JTextField subjectField = new JTextField();  
 JTextArea messageArea = new JTextArea();  
 JButton sendButton = new JButton("Отправить");  
  
 *// Панель ввода* JPanel inputPanel = new JPanel(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5); *// Отступы между элементами* gbc.fill = GridBagConstraints.*HORIZONTAL*;  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 inputPanel.add(new JLabel("Тема:"), gbc);  
  
 gbc.gridx = 1;  
 gbc.gridy = 0;  
 inputPanel.add(subjectField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 gbc.gridwidth = 2;  
 gbc.fill = GridBagConstraints.*BOTH*;  
 inputPanel.add(new JLabel("Сообщение:"), gbc);  
  
 gbc.gridy = 2;  
 gbc.weightx = 1.0;  
 gbc.weighty = 1.0;  
 JScrollPane scrollPane = new JScrollPane(messageArea);  
 inputPanel.add(scrollPane, gbc);  
  
 frame.add(inputPanel, BorderLayout.*CENTER*);  
 frame.add(sendButton, BorderLayout.*SOUTH*);  
  
 *// Кнопка отправки* sendButton.addActionListener((ActionEvent event) -> {  
 String subject = subjectField.getText();  
 String text = messageArea.getText();  
 emailSender.sendEmail(subject, text, this);  
 frame.dispose(); *// Закрываем окно после отправки* });  
  
 frame.setLocationRelativeTo(null); *// Центрируем окно* frame.setVisible(true);  
 }  
  
 */\*\*  
 \* Обновляет доступность полей ввода в зависимости от выбранного фильтра.  
 \* Включает соответствующие подсказки и очищает поле ввода, если это необходимо.  
 \*  
 \*/* private void updateFilterFields() {  
 String selectedFilter = (String) filterComboBox.getSelectedItem();  
 if ("Фильтровать по ФИО".equals(selectedFilter)) {  
 filterInputField.setEnabled(true);  
 filterInputField.setText("");  
 filterInputField.setToolTipText("Введите фамилию студента для фильтрации");  
 } else if ("Фильтровать по группе".equals(selectedFilter)) {  
 filterInputField.setEnabled(true);  
 filterInputField.setText("");  
 filterInputField.setToolTipText("Введите группу студента для фильтрации");  
 } else if ("Фильтровать по дате".equals(selectedFilter)) {  
 filterInputField.setEnabled(true);  
 filterInputField.setText("");  
 filterInputField.setToolTipText("Введите дату для фильтрации");  
 } else {  
 filterInputField.setEnabled(false);  
 filterInputField.setText("");  
 }  
 }  
  
 */\*\*  
 \* Добавляет посещение для выбранного студента.  
 \* Проверяет корректность ввода и формат даты.  
 \* После добавления обновляет таблицу с посещаемостью.  
 \*  
 \* @param e Событие, вызвавшее обработчик  
 \*/* private void addAttendance(ActionEvent e) {  
 try {  
 String fullName = nameField.getText().trim();  
 String group = groupField.getText().trim();  
 String date = dateField.getText().trim();  
  
 *// Проверка на пустые поля* if (fullName.isEmpty() || group.isEmpty() || date.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Все поля должны быть заполнены!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 *// Валидация ФИО (Фамилия Имя Отчество)* if (!fullName.matches("[А-Яа-яЁёA-Za-z]+\\s[А-Яа-яЁёA-Za-z]+\\s[А-Яа-яЁёA-Za-z]+")) {  
 JOptionPane.*showMessageDialog*(this, "ФИО должно быть в формате: Фамилия Имя Отчество, без лишних пробелов и специальных символов!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 *// Валидация номера группы (8 цифр)* if (!group.matches("\\d{8}")) {  
 JOptionPane.*showMessageDialog*(this, "Номер группы должен содержать ровно 8 цифр!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 *// Валидация даты* SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
 sdf.setLenient(false);  
  
 try {  
 Date parsedDate = sdf.parse(date);  
  
 Calendar cal = Calendar.*getInstance*();  
 cal.setTime(parsedDate);  
  
 int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*) + 1; *// Январь = 0* int currentYear = LocalDate.*now*().getYear();  
  
 *// Проверка года и месяца* if (year > currentYear) {  
 JOptionPane.*showMessageDialog*(this, "Год не может быть больше текущего!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
 if (month < 1 || month > 12) {  
 JOptionPane.*showMessageDialog*(this, "Месяц должен быть в диапазоне от 1 до 12!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
 } catch (ParseException ex) {  
 JOptionPane.*showMessageDialog*(this, "Некорректный формат даты! Используйте формат yyyy-MM-dd.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 *// Если все проверки пройдены, добавляем студента* Student student = students.getOrDefault(fullName, new Student(fullName, "", "", group));  
 student.addAttendanceDate(date);  
 students.put(fullName, student);  
  
 updateDisplayArea(); *// Обновляем таблицу после добавления* nameField.setText("");  
 groupField.setText("");  
 dateField.setText("");  
  
 } catch (Exception ex) {  
 JOptionPane.*showMessageDialog*(this, "Ошибка ввода данных: " + ex.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
  
  
 */\*\*  
 \* Обновляет таблицу с посещаемостью студентов.  
 \* Для каждого студента рассчитывается количество посещений в выбранный месяц.  
 \* Вставляются значения в таблицу, включая посещения и отсутствие.  
 \*/* private void updateDisplayArea() {  
 tableModel.setRowCount(0); *// Очистить текущую таблицу* for (Student student : students.values()) {  
 List<String> row = new ArrayList<>();  
 row.add(student.getFullName());  
 row.add(student.getGroup());  
  
 int attendanceCount = 0; *// Счётчик посещений* LocalDate startOfMonth = selectedMonth.withDayOfMonth(1);  
 LocalDate endOfMonth = selectedMonth.withDayOfMonth(selectedMonth.lengthOfMonth());  
  
 for (LocalDate date = startOfMonth; !date.isAfter(endOfMonth); date = date.plusDays(1)) {  
 String formattedDate = date.format(java.time.format.DateTimeFormatter.*ofPattern*("yyyy-MM-dd"));  
 if (student.getAttendanceDates().contains(formattedDate)) {  
 row.add("✓");  
 attendanceCount++; *// Увеличиваем счётчик посещений* } else if (student.getAbsenceDates().contains(formattedDate)) {  
 row.add("✗");  
 } else {  
 row.add("");  
 }  
 }  
  
 *// Добавляем количество посещений в строку* row.add(2, String.*valueOf*(attendanceCount)); *// Вставляем в колонку "Посещения"* tableModel.addRow(row.toArray());  
 }  
  
 *// Применение рендера после обновления данных* for (int i = 3; i < tableModel.getColumnCount(); i++) { *// Колонки с датами начинаются с 3* table.getColumnModel().getColumn(i).setCellRenderer(getCustomTableCellRenderer());  
 }  
 }  
  
 */\*\*  
 \* Применяет фильтр к данным в таблице на основе выбранного типа фильтра и введенного значения.  
 \* Сохраняет текущее состояние перед фильтрацией, очищает таблицу и добавляет только отфильтрованные данные.  
 \*  
 \* @param filterType Тип фильтра  
 \* @param input Ввод пользователя для фильтрации  
 \*/* private void applyFilter(String filterType, String input) {  
 saveTableStateToStudents(); *// Сохраняем текущее состояние перед фильтрацией* tableModel.setRowCount(0); *// Очищаем таблицу* for (Student student : students.values()) {  
 boolean matches = false;  
  
 if ("Фильтровать по дате".equals(filterType)) {  
 matches = student.getAttendanceDates().contains(input);  
 } else if ("Фильтровать по группе".equals(filterType)) {  
 matches = student.getGroup().equalsIgnoreCase(input);  
 } else if ("Фильтровать по ФИО".equals(filterType)) {  
 matches = student.getFullName().toLowerCase().contains(input.toLowerCase());  
 }  
  
 if (matches) {  
 addStudentRowToTable(student); *// Добавляем отфильтрованного студента* }  
 }  
 }  
  
 */\*\*  
 \* Сохраняет данные таблицы в Excel файл.  
 \* Пользователь выбирает место для сохранения файла, и данные сохраняются в формате XLSX.  
 \*/* private void saveToExcel() {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Сохранить в Excel");  
 if (fileChooser.showSaveDialog(this) == JFileChooser.*APPROVE\_OPTION*) {  
 String filePath = fileChooser.getSelectedFile().getAbsolutePath();  
 if (!filePath.endsWith(".xlsx")) {  
 filePath += ".xlsx";  
 }  
  
 try (Workbook workbook = new XSSFWorkbook();  
 FileOutputStream fileOut = new FileOutputStream(filePath)) {  
 Sheet sheet = workbook.createSheet("Attendance");  
  
 Row headerRow = sheet.createRow(0);  
 for (int i = 0; i < tableModel.getColumnCount(); i++) {  
 headerRow.createCell(i).setCellValue(tableModel.getColumnName(i));  
 }  
  
 for (int i = 0; i < tableModel.getRowCount(); i++) {  
 Row row = sheet.createRow(i + 1);  
 for (int j = 0; j < tableModel.getColumnCount(); j++) {  
 row.createCell(j).setCellValue(String.*valueOf*(tableModel.getValueAt(i, j)));  
 }  
 }  
  
 workbook.write(fileOut);  
 JOptionPane.*showMessageDialog*(this, "Данные успешно сохранены в Excel!", "Успех", JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (IOException e) {  
 JOptionPane.*showMessageDialog*(this, "Ошибка при сохранении файла: " + e.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
  
 */\*\*  
 \* Загружает данные студентов из файла Excel.  
 \* <p>  
 \* Этот метод предоставляет возможность выбрать файл Excel, содержащий информацию о студентах.  
 \* Данные из файла загружаются в текущий список студентов. Пропускаются строки с пустыми значениями  
 \* или некорректными ФИО. В случае ошибок при загрузке данных, пользователю показывается сообщение об ошибке.  
 \* </p>  
 \*/* private void loadFromExcel() {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("Загрузить из Excel");  
 if (fileChooser.showOpenDialog(this) == JFileChooser.*APPROVE\_OPTION*) {  
 try (FileInputStream fis = new FileInputStream(fileChooser.getSelectedFile());  
 Workbook workbook = WorkbookFactory.*create*(fis)) {  
  
 Sheet sheet = workbook.getSheetAt(0);  
 students.clear(); *// Очищаем текущий список студентов* Iterator<Row> rowIterator = sheet.rowIterator();  
 if (rowIterator.hasNext()) rowIterator.next(); *// Пропускаем заголовок* while (rowIterator.hasNext()) {  
 Row row = rowIterator.next();  
  
 String fullName = getCellValueAsString(row.getCell(0));  
 String group = getCellValueAsString(row.getCell(1));  
  
 if (fullName == null || fullName.isEmpty() || group == null || group.isEmpty()) {  
 continue; *// Пропуск пустых строк* }  
  
 *// Удаление лишних пробелов* fullName = fullName.trim();  
 group = group.trim();  
  
 *// Разделение ФИО на части* String[] nameParts = fullName.split("\\s+");  
 if (nameParts.length < 2) {  
 JOptionPane.*showMessageDialog*(this, "Некорректное ФИО: " + fullName, "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 continue; *// Пропуск некорректных строк* }  
  
 String lastName = nameParts[0];  
 String firstName = nameParts.length > 1 ? nameParts[1] : "";  
 String patronymic = nameParts.length > 2 ? nameParts[2] : "";  
  
 *// Создаем объект студента* Student student = new Student(lastName, firstName, patronymic, group);  
  
 *// Добавляем студента в список* students.put(fullName, student);  
 }  
  
 updateDisplayArea(); *// Обновляем таблицу* JOptionPane.*showMessageDialog*(this, "Данные успешно загружены!", "Успех", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(this, "Ошибка при загрузке файла: " + ex.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
  
  
 */\*\*  
 \* Получает значение ячейки Excel как строку.  
 \* <p>  
 \* Этот метод извлекает значение из ячейки Excel в виде строки, учитывая тип данных в ячейке.  
 \* Если ячейка содержит число, оно конвертируется в строку. Если ячейка содержит дату, дата будет  
 \* преобразована в строковое представление. Если ячейка пуста или имеет неподдерживаемый тип, метод возвращает пустую строку.  
 \* </p>  
 \*  
 \* @param cell ячейка Excel, из которой необходимо извлечь значение.  
 \* @return строковое представление значения ячейки.  
 \*/* private String getCellValueAsString(Cell cell) {  
 if (cell == null) {  
 return "";  
 }  
 switch (cell.getCellType()) {  
 case *STRING*:  
 return cell.getStringCellValue().trim();  
 case *NUMERIC*:  
 if (DateUtil.*isCellDateFormatted*(cell)) {  
 return cell.getDateCellValue().toString();  
 } else {  
 return String.*valueOf*((long) cell.getNumericCellValue()); *// Если число, то приводим к строке* }  
 case *BOOLEAN*:  
 return String.*valueOf*(cell.getBooleanCellValue());  
 case *FORMULA*:  
 try {  
 return cell.getStringCellValue();  
 } catch (IllegalStateException e) {  
 return String.*valueOf*(cell.getNumericCellValue());  
 }  
 default:  
 return "";  
 }  
 }  
  
 */\*\*  
 \* Сбрасывает фильтры и восстанавливает состояние таблицы.  
 \* <p>  
 \* Этот метод сохраняет текущее состояние таблицы в список студентов перед сбросом фильтров.  
 \* После сброса фильтров, отображение обновляется для отображения всех студентов.  
 \* </p>  
 \*/* private void resetFilters() {  
 saveTableStateToStudents(); *// Сохраняем текущее состояние перед сбросом фильтров* updateDisplayArea();  
 }  
  
  
 */\*\*  
 \* Отображает окно с информацией о программе.  
 \* <p>  
 \* Метод создает и отображает окно с информацией о программе, включая описание функциональности,  
 \* фото и текст. Также имеется кнопка "Выход" для закрытия окна.  
 \* </p>  
 \*/* private void showAboutProgram() {  
 JDialog aboutDialog = new JDialog(this, "О программе", true);  
 aboutDialog.setSize(610, 350); *// Размер окна, как на скриншоте* aboutDialog.setLayout(new BorderLayout());  
  
 *// Панель для размещения картинки и текста* JPanel contentPanel = new JPanel(new BorderLayout());  
  
 *// Фото* JLabel photoLabel = new JLabel();  
 photoLabel.setPreferredSize(new Dimension(200, 200));  
 photoLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  
 photoLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 photoLabel.setVerticalAlignment(JLabel.*CENTER*);  
  
 ImageIcon photoIcon = null;  
 try {  
 java.net.URL imgURL = getClass().getResource("/photo10.jpg"); *// Путь к изображению в ресурсах* if (imgURL != null) {  
 Image img = new ImageIcon(imgURL).getImage();  
 photoIcon = new ImageIcon(img.getScaledInstance(400, 300, Image.*SCALE\_SMOOTH*));  
 } else {  
 throw new NullPointerException("Файл изображения /photo10.jpg не найден.");  
 }  
 } catch (Exception ex) {  
 String errorMessage = "Ошибка загрузки изображения: " + ex.getMessage();  
 JOptionPane.*showMessageDialog*(this, errorMessage, "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 System.*err*.println(errorMessage);  
 }  
  
 if (photoIcon != null) {  
 photoLabel.setIcon(photoIcon);  
 }  
  
 contentPanel.add(photoLabel, BorderLayout.*WEST*);  
  
 *// Текстовая область* JTextArea aboutTextArea = new JTextArea();  
 aboutTextArea.setEditable(false);  
 aboutTextArea.setText(  
 "Программа учета посещаемости студентов\n\n" + *// Новое описание* "Программа позволяет:\n" +  
 "1. Добавлять посещения студентов с указанием ФИО, группы и даты.\n" +  
 "2. Сохранять данные о посещаемости в Excel файл.\n" +  
 "3. Загружать данные о посещаемости из Excel файла.\n" +  
 "4. Фильтровать студентов по ФИО, группе или дате посещения.\n" +  
 "5. Очищать все данные о посещаемости."  
 );  
 aboutTextArea.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10)); *// Отступы от краев* contentPanel.add(aboutTextArea, BorderLayout.*CENTER*);  
  
  
 aboutDialog.add(contentPanel, BorderLayout.*CENTER*);  
  
  
 *// Панель для версии и кнопки* JPanel bottomPanel = new JPanel(new BorderLayout());  
  
 JLabel versionLabel = new JLabel("Версия ver. 23.0.1.2024");  
 bottomPanel.add(versionLabel, BorderLayout.*WEST*);  
  
  
 JButton closeButton = new JButton("Выход");  
 closeButton.setBackground(new Color(255, 102, 102)); *// Розовый цвет кнопки* closeButton.addActionListener(e -> aboutDialog.dispose());  
 bottomPanel.add(closeButton, BorderLayout.*EAST*);  
  
  
 aboutDialog.add(bottomPanel, BorderLayout.*SOUTH*);  
  
  
 aboutDialog.setLocationRelativeTo(this);  
 aboutDialog.setVisible(true);  
 }  
  
 */\*\*  
 \* Отображает окно с информацией об авторе программы.  
 \* <p>  
 \* Этот метод создает окно с информацией об авторе, включая фото и контактные данные.  
 \* Также есть кнопка "Назад" для закрытия окна.  
 \* </p>  
 \*/* private void showAboutAuthor() {  
 JDialog authorDialog = new JDialog(this, "Об авторе", true);  
 authorDialog.setSize(700, 700);  
  
 JPanel mainPanel = new JPanel(new BorderLayout());  
  
 *// Фото* JLabel photoLabel = new JLabel();  
 photoLabel.setPreferredSize(new Dimension(200, 200));  
 photoLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  
 photoLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 photoLabel.setVerticalAlignment(JLabel.*CENTER*);  
  
 ImageIcon photoIcon = null;  
 try {  
 java.net.URL imgURL = getClass().getResource("/photo\_about\_author.jpg"); *// Путь к изображению в ресурсах* if (imgURL != null) {  
 Image img = new ImageIcon(imgURL).getImage();  
 photoIcon = new ImageIcon(img.getScaledInstance(600, 500, Image.*SCALE\_SMOOTH*));  
 } else {  
 throw new NullPointerException("Файл изображения /photo2.jpg не найден.");  
 }  
 } catch (Exception ex) {  
 String errorMessage = "Ошибка загрузки изображения: " + ex.getMessage();  
 JOptionPane.*showMessageDialog*(this, errorMessage, "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 System.*err*.println(errorMessage);  
 }  
  
 if (photoIcon != null) {  
 photoLabel.setIcon(photoIcon);  
 }  
  
 mainPanel.add(photoLabel, BorderLayout.*CENTER*); *// Добавляем фото в центр  
  
 // Текстовая информация об авторе* JPanel textPanel = new JPanel();  
 textPanel.setLayout(new BoxLayout(textPanel, BoxLayout.*Y\_AXIS*));  
 textPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10)); *// Отступы* JLabel authorLabel = new JLabel("Автор");  
 JLabel groupLabel = new JLabel("Студент группы 10702122");  
 JLabel nameLabel = new JLabel("Солдатов Никита Викторович");  
 JLabel emailLabel = new JLabel("miracleqxz@gmail.com");  
  
 *// Установка шрифта и выравнивания для всех меток* Font boldFont = new Font("Arial", Font.*BOLD*, 16);  
 authorLabel.setFont(boldFont);  
 groupLabel.setFont(boldFont);  
 nameLabel.setFont(boldFont);  
 emailLabel.setFont(boldFont);  
  
 *// Центрирование текста* authorLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 groupLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 nameLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 emailLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
  
 *// Добавляем метки в панель* textPanel.add(authorLabel);  
 textPanel.add(Box.*createVerticalStrut*(10)); *// Отступ между строками* textPanel.add(groupLabel);  
 textPanel.add(Box.*createVerticalStrut*(10));  
 textPanel.add(nameLabel);  
 textPanel.add(Box.*createVerticalStrut*(10));  
 textPanel.add(emailLabel);  
  
 mainPanel.add(textPanel, BorderLayout.*SOUTH*);  
  
 *// Кнопка "Назад"* JButton backButton = new JButton("Назад");  
 backButton.addActionListener(e -> authorDialog.dispose());  
 JPanel backButtonPanel = new JPanel(new FlowLayout(FlowLayout.*CENTER*));  
 backButtonPanel.add(backButton);  
  
 *// Добавляем все в диалог* authorDialog.add(mainPanel, BorderLayout.*CENTER*);  
 authorDialog.add(backButtonPanel, BorderLayout.*SOUTH*);  
  
 authorDialog.setLocationRelativeTo(this);  
 authorDialog.setVisible(true);  
 }  
  
 */\*\*  
 \* Открывает карту студента для выбора даты посещения и отметки присутствия.  
 \* <p>  
 \* Метод отображает окно с картой студента, где можно выбрать дату и отметить присутствие студента.  
 \* Также показывается информация о студенте, и пользователь может отмечать студентов как присутствующих  
 \* или отсутствующих на определенные даты.  
 \* </p>  
 \*  
 \* @param subject название предмета, который отображается в диалоговом окне.  
 \*/* private void openStudentMap(String subject) {  
 JDialog studentMapDialog = new JDialog(this, "Карта студента", true);  
 studentMapDialog.setSize(600, 500);  
 studentMapDialog.setLayout(new BorderLayout(10, 10));  
  
 *// Панель информации о студенте* JPanel infoPanel = new JPanel(new BorderLayout(10, 10));  
 infoPanel.setBorder(BorderFactory.*createTitledBorder*("Информация о студенте"));  
  
 JLabel nameLabel = new JLabel("ФИО: ");  
 nameLabel.setFont(new Font("Arial", Font.*BOLD*, 16));  
 JLabel groupLabel = new JLabel("Группа: ");  
 groupLabel.setFont(new Font("Arial", Font.*BOLD*, 16));  
 JLabel subjectLabel = new JLabel("Предмет: " + subject);  
 subjectLabel.setFont(new Font("Arial", Font.*ITALIC*, 14));  
  
 JPanel labelsPanel = new JPanel(new GridLayout(3, 1, 5, 5));  
 labelsPanel.add(nameLabel);  
 labelsPanel.add(groupLabel);  
 labelsPanel.add(subjectLabel);  
  
 infoPanel.add(labelsPanel, BorderLayout.*CENTER*);  
 studentMapDialog.add(infoPanel, BorderLayout.*NORTH*);  
  
 *// Календарь для выбора даты* JCalendar calendar = new JCalendar();  
 calendar.setBorder(BorderFactory.*createTitledBorder*("Выберите дату"));  
 studentMapDialog.add(calendar, BorderLayout.*CENTER*);  
  
 *// Панель кнопок* JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.*CENTER*, 10, 10));  
 JButton presentButton = new JButton("Присутствовал");  
 JButton absentButton = new JButton("Отсутствовал");  
 JButton closeButton = new JButton("Закрыть");  
  
 presentButton.setBackground(new Color(102, 255, 102));  
 presentButton.setForeground(Color.*WHITE*);  
 absentButton.setBackground(new Color(255, 102, 102));  
 absentButton.setForeground(Color.*WHITE*);  
 closeButton.setBackground(Color.*LIGHT\_GRAY*);  
  
 buttonPanel.add(presentButton);  
 buttonPanel.add(absentButton);  
 buttonPanel.add(closeButton);  
  
 studentMapDialog.add(buttonPanel, BorderLayout.*SOUTH*);  
  
 *// Список студентов* List<Student> studentList = new ArrayList<>(students.values());  
 if (studentList.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Список студентов пуст!", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 final int[] currentIndex = {0};  
  
 *// Метод для обновления информации о текущем студенте* Runnable updateStudentInfo = () -> {  
 if (currentIndex[0] < studentList.size()) {  
 Student currentStudent = studentList.get(currentIndex[0]);  
 nameLabel.setText("ФИО: " + currentStudent.getFullName());  
 groupLabel.setText("Группа: " + currentStudent.getGroup());  
 } else {  
 JOptionPane.*showMessageDialog*(studentMapDialog, "Все студенты обработаны!", "Завершено", JOptionPane.*INFORMATION\_MESSAGE*);  
 studentMapDialog.dispose();  
 }  
 };  
  
 updateStudentInfo.run();  
  
 *// Обработчики кнопок* ActionListener markAttendance = e -> {  
 if (currentIndex[0] < studentList.size()) {  
 Student currentStudent = studentList.get(currentIndex[0]);  
 String selectedDate = new SimpleDateFormat("yyyy-MM-dd").format(calendar.getDate());  
  
 if (e.getSource() == presentButton) {  
 currentStudent.addAttendanceDate(selectedDate);  
 JOptionPane.*showMessageDialog*(studentMapDialog, "Отмечено 'Присутствовал' для даты " + selectedDate, "Успех", JOptionPane.*INFORMATION\_MESSAGE*);  
 } else if (e.getSource() == absentButton) {  
 currentStudent.addAbsenceDate(selectedDate);  
 JOptionPane.*showMessageDialog*(studentMapDialog, "Отмечено 'Отсутствовал' для даты " + selectedDate, "Успех", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
  
 currentIndex[0]++;  
 updateStudentInfo.run();  
 updateDisplayArea(); *// Обновляем таблицу в главном окне* }  
 };  
  
 presentButton.addActionListener(markAttendance);  
 absentButton.addActionListener(markAttendance);  
  
 closeButton.addActionListener(e -> studentMapDialog.dispose());  
  
 studentMapDialog.setLocationRelativeTo(this);  
 studentMapDialog.setVisible(true);  
 }  
  
  
  
  
 *// Метод для обновления таблицы* private void updateTable(Student student, String mark) {  
 String fullName = student.getFullName();  
 int rowCount = tableModel.getRowCount();  
 LocalDate today = LocalDate.*now*();  
 String currentDateColumn = today.format(java.time.format.DateTimeFormatter.*ofPattern*("dd.MM"));  
  
 for (int row = 0; row < rowCount; row++) {  
 if (tableModel.getValueAt(row, 0).equals(fullName)) {  
 int colCount = tableModel.getColumnCount();  
 for (int col = 2; col < colCount; col++) {  
 if (tableModel.getColumnName(col).equals(currentDateColumn)) {  
 tableModel.setValueAt(mark, row, col);  
  
 *// Обновляем данные в объекте Student* if (mark.equals("✓")) {  
 student.addAttendanceDate(today.toString());  
 } else if (mark.equals("✗")) {  
 student.addAbsenceDate(today.toString());  
 }  
  
 return; *// Обновляем только одну строку* }  
 }  
 }  
 }  
 }  
  
  
  
  
*/\*  
 private void openStudentMap(String subject) {  
 JDialog studentMapDialog = new JDialog(this, "Карта студентов", true);  
 studentMapDialog.setSize(500, 400);  
 studentMapDialog.setLayout(new BorderLayout());  
  
 // Панель для отображения текущего студента  
 JPanel studentInfoPanel = new JPanel(new GridLayout(5, 1));  
 JLabel studentNameLabel = new JLabel("ФИО: ");  
 JLabel studentGroupLabel = new JLabel("Группа: ");  
 JLabel datesLabel = new JLabel("Даты посещений: ");  
 JLabel subjectLabel = new JLabel("Предмет: " + subject);  
  
 studentInfoPanel.add(studentNameLabel);  
 studentInfoPanel.add(studentGroupLabel);  
 studentInfoPanel.add(datesLabel);  
 studentInfoPanel.add(subjectLabel);  
  
 studentMapDialog.add(studentInfoPanel, BorderLayout.NORTH);  
  
 // Календарь для выбора даты  
 com.toedter.calendar.JCalendar calendar = new com.toedter.calendar.JCalendar();  
 studentMapDialog.add(calendar, BorderLayout.CENTER);  
  
 // Панель для кнопок управления  
 JPanel buttonPanel = new JPanel(new FlowLayout());  
 JButton presentButton = new JButton("Присутствовал");  
 JButton absentButton = new JButton("Отсутствовал");  
 JButton nextStudentButton = new JButton("Следующий студент");  
 buttonPanel.add(presentButton);  
 buttonPanel.add(absentButton);  
 buttonPanel.add(nextStudentButton);  
  
 studentMapDialog.add(buttonPanel, BorderLayout.SOUTH);  
  
 // Логика работы с текущими студентами  
 List<Student> studentList = new ArrayList<>(students.values());  
 int[] currentIndex = {0};  
  
 // Метод обновления информации о студенте  
 Runnable updateStudentInfo = () -> {  
 if (currentIndex[0] < studentList.size()) {  
 Student student = studentList.get(currentIndex[0]);  
 studentNameLabel.setText("ФИО: " + student.getFullName());  
 studentGroupLabel.setText("Группа: " + student.getGroup());  
 datesLabel.setText("Даты посещений: " + String.join(", ", student.getAttendanceDates()));  
 } else {  
 studentNameLabel.setText("ФИО: ");  
 studentGroupLabel.setText("Группа: ");  
 datesLabel.setText("Даты посещений: ");  
 }  
 };  
  
 updateStudentInfo.run(); // Инициализация информации о первом студенте  
  
 // Обработчик кнопки "Присутствовал"  
 presentButton.addActionListener(e -> {  
 if (currentIndex[0] < studentList.size()) {  
 Student student = studentList.get(currentIndex[0]);  
 Date selectedDate = calendar.getDate();  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
 String dateString = sdf.format(selectedDate);  
  
 student.addAttendanceDate(dateString); // Добавляем дату посещения  
 students.put(student.getLastName(), student);  
  
 updateDisplayArea(); // Обновляем главное окно  
 updateStudentInfo.run();  
 JOptionPane.showMessageDialog(studentMapDialog, "Дата " + dateString + " добавлена как 'Присутствовал'!", "Успех", JOptionPane.INFORMATION\_MESSAGE);  
 }  
 });  
  
 // Обработчик кнопки "Отсутствовал"  
 absentButton.addActionListener(e -> {  
 if (currentIndex[0] < studentList.size()) {  
 updateStudentInfo.run(); // Переход к следующему студенту  
 JOptionPane.showMessageDialog(studentMapDialog, "Студент отмечен как 'Отсутствовал'!", "Информация", JOptionPane.INFORMATION\_MESSAGE);  
 }  
 });  
  
 // Обработчик кнопки "Следующий студент"  
 nextStudentButton.addActionListener(e -> {  
 currentIndex[0]++;  
 if (currentIndex[0] < studentList.size()) {  
 updateStudentInfo.run();  
 } else {  
 JOptionPane.showMessageDialog(studentMapDialog, "Все студенты обработаны!", "Завершено", JOptionPane.INFORMATION\_MESSAGE);  
 studentMapDialog.dispose();  
 }  
 });  
  
 studentMapDialog.setLocationRelativeTo(this);  
 studentMapDialog.setVisible(true);  
 }  
\*/  
  
 /\*\*  
 \* Генерирует массив названий колонок для таблицы, включая ФИО, группу, посещения и даты для выбранного месяца.  
 \*  
 \* @param month месяц, для которого генерируются названия колонок  
 \* @return массив строк, представляющих названия колонок  
 \*/* private String[] generateColumnNames(LocalDate month) {  
 List<String> columnNames = new ArrayList<>();  
 columnNames.add("ФИО");  
 columnNames.add("Группа");  
 columnNames.add("Посещения"); *// Новая колонка для подсчёта посещений* LocalDate startOfMonth = month.withDayOfMonth(1);  
 LocalDate endOfMonth = month.withDayOfMonth(month.lengthOfMonth());  
  
 for (LocalDate date = startOfMonth; !date.isAfter(endOfMonth); date = date.plusDays(1)) {  
 columnNames.add(date.format(java.time.format.DateTimeFormatter.*ofPattern*("dd.MM")));  
 }  
 return columnNames.toArray(new String[0]);  
 }  
  
 */\*\*  
 \* Добавляет строку с информацией о студенте в таблицу, включая ФИО, группу и отметки о посещениях за каждый день месяца.  
 \*  
 \* @param student студент, чьи данные необходимо добавить в таблицу  
 \*/* private void addStudentRowToTable(Student student) {  
 List<String> row = new ArrayList<>();  
 row.add(student.getFullName());  
 row.add(student.getGroup());  
  
 LocalDate today = LocalDate.*now*();  
 LocalDate startOfMonth = today.withDayOfMonth(1);  
 LocalDate endOfMonth = today.withDayOfMonth(today.lengthOfMonth());  
  
 for (LocalDate date = startOfMonth; !date.isAfter(endOfMonth); date = date.plusDays(1)) {  
 String formattedDate = date.format(java.time.format.DateTimeFormatter.*ofPattern*("yyyy-MM-dd"));  
 if (student.getAttendanceDates().contains(formattedDate)) {  
 row.add("✓");  
 } else if (student.getAbsenceDates().contains(formattedDate)) {  
 row.add("✗");  
 } else {  
 row.add("");  
 }  
 }  
  
 tableModel.addRow(row.toArray());  
 }  
  
 */\*\*  
 \* Сохраняет состояние таблицы в объектах студентов, обновляя их даты посещений и отсутствий.  
 \* Использует фамилию студента для поиска в списке студентов.  
 \*/* private void saveTableStateToStudents() {  
 *// Сохраняем состояние таблицы в объекте Student* int rowCount = tableModel.getRowCount();  
 int colCount = tableModel.getColumnCount();  
  
 for (int row = 0; row < rowCount; row++) {  
 String fullName = tableModel.getValueAt(row, 0).toString();  
 Student student = students.get(fullName.split(" ")[0]); *// Используем фамилию как ключ* if (student != null) {  
 LocalDate today = LocalDate.*now*();  
 LocalDate startOfMonth = today.withDayOfMonth(1);  
  
 *// Очищаем старые данные и обновляем новые* student.clearAttendance();  
  
 for (int col = 2; col < colCount; col++) { *// Колонки с датами начинаются с 2* String date = startOfMonth.plusDays(col - 2).format(java.time.format.DateTimeFormatter.*ofPattern*("yyyy-MM-dd"));  
 String mark = tableModel.getValueAt(row, col).toString();  
  
 if ("✓".equals(mark)) {  
 student.addAttendanceDate(date);  
 } else if ("✗".equals(mark)) {  
 student.addAbsenceDate(date);  
 }  
 }  
 }  
 }  
 }  
  
 */\*\*  
 \* Обновляет таблицу для выбранного месяца, сохраняя текущие данные и изменяя заголовки колонок.  
 \*  
 \* @param month месяц, для которого обновляется таблица  
 \*/* private void updateTableForMonth(LocalDate month) {  
 saveTableStateToStudents(); *// Сохраняем текущие данные* tableModel.setColumnIdentifiers(generateColumnNames(month)); *// Обновляем заголовки колонок* updateDisplayArea();  
 }  
  
 */\*\*  
 \* Возвращает кастомный рендерер для ячеек таблицы, который окрашивает ячейки в зависимости от отметок о посещении.  
 \*  
 \* @return рендерер для ячеек таблицы  
 \*/* private TableCellRenderer getCustomTableCellRenderer() {  
 return new DefaultTableCellRenderer() {  
 @Override  
 public Component getTableCellRendererComponent(JTable table, Object value,  
 boolean isSelected, boolean hasFocus,  
 int row, int column) {  
 Component cell = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);  
  
 if (value != null) {  
 String cellValue = value.toString();  
 if ("✓".equals(cellValue)) {  
 cell.setBackground(new Color(102, 255, 102)); *// Зеленый цвет* cell.setForeground(Color.*BLACK*);  
 } else if ("✗".equals(cellValue)) {  
 cell.setBackground(new Color(255, 102, 102)); *// Красный цвет* cell.setForeground(Color.*BLACK*);  
 } else {  
 cell.setBackground(Color.*WHITE*); *// Обычный фон для пустых ячеек* cell.setForeground(Color.*BLACK*);  
 }  
 }  
 return cell;  
 }  
 };  
 }  
  
 */\*\*  
 \* Главный метод для запуска приложения, который создает и отображает окно.  
 \*  
 \* @param args аргументы командной строки (не используются)  
 \*/* public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 MainFrame frame = new MainFrame();  
 frame.setVisible(true);  
 });  
 }  
}

**SlashScreen.java**

package com.example;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
  
*/\*\*  
 \* Класс SplashScreen представляет собой окно загрузки (splash screen),  
 \* которое отображает информацию о проекте и предоставляет пользователю  
 \* возможность продолжить или выйти.  
 \*  
 \* @author Soldatov N. V.  
 \* @version 22.0.2  
 \*/*public class SplashScreen extends JFrame {  
  
 */\*\*  
 \* Поле, указывающее, была ли нажата кнопка продолжения.  
 \*/* private boolean isContinuePressed = false;  
  
 */\*\*  
 \* Таймер для отслеживания времени на экране.  
 \*/* private Timer timer;  
  
 */\*\*  
 \* Конструктор, создающий окно splash screen.  
 \* Конфигурирует все элементы UI и их взаимодействие.  
 \*/* public SplashScreen() {  
 *// Настройка заголовка окна* setTitle("SplashScreen");  
 setSize(800, 600);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
  
 *// Основная панель* JPanel mainPanel = new JPanel();  
 mainPanel.setLayout(new BorderLayout());  
 mainPanel.setBackground(Color.*WHITE*);  
  
 *// Панель с текстом* JPanel textPanel = new JPanel();  
 textPanel.setLayout(new BoxLayout(textPanel, BoxLayout.*Y\_AXIS*));  
 textPanel.setBackground(Color.*WHITE*);  
  
 *// Текстовые метки* JLabel universityLabel = new JLabel("БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ");  
 universityLabel.setFont(new Font("Arial", Font.*BOLD*, 20));  
 universityLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 JLabel facultyLabel = new JLabel("Факультет информационных технологий и робототехники");  
 facultyLabel.setFont(new Font("Arial", Font.*BOLD*, 16));  
 facultyLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 JLabel departmentLabel = new JLabel("Кафедра программного обеспечения информационных систем и технологий");  
 departmentLabel.setFont(new Font("Arial", Font.*BOLD*, 16));  
 departmentLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 JLabel courseLabel = new JLabel("КУРСОВАЯ РАБОТА");  
 courseLabel.setFont(new Font("Arial", Font.*BOLD*, 24));  
 courseLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 JLabel disciplineLabel = new JLabel("по дисциплине \"Программирование на языке Java\"");  
 disciplineLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 disciplineLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 JLabel topicLabel = new JLabel("ПОСЕЩАЕМОСТЬ ЛЕКЦИОННЫХ ЗАНЯТИЙ");  
 topicLabel.setFont(new Font("Arial", Font.*BOLD*, 18));  
 topicLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
  
 *// Добавление меток в текстовую панель* textPanel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
 textPanel.add(universityLabel);  
 textPanel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
 textPanel.add(facultyLabel);  
 textPanel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 textPanel.add(departmentLabel);  
 textPanel.add(Box.*createRigidArea*(new Dimension(0, 40)));  
 textPanel.add(courseLabel);  
 textPanel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 textPanel.add(disciplineLabel);  
 textPanel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
 textPanel.add(topicLabel);  
  
 *// Нижняя панель с фото, автором и преподавателем* JPanel bottomPanel = new JPanel(new BorderLayout());  
 bottomPanel.setBackground(Color.*WHITE*);  
  
 *// Фото* JLabel photoLabel = new JLabel();  
 photoLabel.setPreferredSize(new Dimension(200, 200));  
 photoLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  
 photoLabel.setHorizontalAlignment(JLabel.*CENTER*);  
 photoLabel.setVerticalAlignment(JLabel.*CENTER*);  
  
 *// Загрузка изображения* ImageIcon photoIcon = null;  
 try {  
 java.net.URL imgURL = getClass().getResource("/photo\_splash.png");  
 if (imgURL != null) {  
 Image img = new ImageIcon(imgURL).getImage();  
 photoIcon = new ImageIcon(img.getScaledInstance(200, 200, Image.*SCALE\_SMOOTH*));  
 } else {  
 throw new NullPointerException("Файл изображения /photo\_splash.png не найден.");  
 }  
 } catch (Exception ex) {  
 JOptionPane.*showMessageDialog*(this, "Ошибка загрузки изображения: " + ex.getMessage(),  
 "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 *// Установка изображения, если оно загружено* if (photoIcon != null) {  
 photoLabel.setIcon(photoIcon);  
 }  
  
 *// Панель для фото* JPanel photoContainer = new JPanel();  
 photoContainer.setLayout(new BorderLayout());  
 photoContainer.setBackground(Color.*WHITE*);  
 photoContainer.setBorder(BorderFactory.*createEmptyBorder*(0, 100, 0, 0));  
 photoContainer.add(photoLabel, BorderLayout.*CENTER*);  
  
 *// Панель с текстами (автор, преподаватель)* JPanel textAlignmentPanel = new JPanel();  
 GroupLayout layout = new GroupLayout(textAlignmentPanel);  
 textAlignmentPanel.setLayout(layout);  
 textAlignmentPanel.setBackground(Color.*WHITE*);  
  
 JLabel authorLabel = new JLabel("Выполнил: Студент группы 10702122");  
 JLabel fioLabel = new JLabel("Солдатов Никита Викторович");  
 JLabel supervisorLabel = new JLabel("Преподаватель: к.ф.-м.н., доц.");  
 JLabel supervisorNameLabel = new JLabel("Сидорик Валерий Владимирович");  
  
 *// Установка шрифтов для текста* authorLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 fioLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 supervisorLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 supervisorNameLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
  
 *// Компоновка текста* layout.setHorizontalGroup(  
 layout.createSequentialGroup()  
 .addGap(200)  
 .addGroup(layout.createParallelGroup(GroupLayout.Alignment.*LEADING*)  
 .addComponent(authorLabel)  
 .addComponent(fioLabel)  
 .addComponent(supervisorLabel)  
 .addComponent(supervisorNameLabel))  
 );  
  
 layout.setVerticalGroup(  
 layout.createSequentialGroup()  
 .addGap(80)  
 .addComponent(authorLabel)  
 .addComponent(fioLabel)  
 .addGap(20)  
 .addComponent(supervisorLabel)  
 .addComponent(supervisorNameLabel)  
 );  
  
 *// Город и год* JLabel cityLabel = new JLabel("Минск, 2024");  
 cityLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 cityLabel.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
 *// Компоновка нижней панели* bottomPanel.add(photoContainer, BorderLayout.*WEST*);  
 bottomPanel.add(textAlignmentPanel, BorderLayout.*CENTER*);  
 bottomPanel.add(cityLabel, BorderLayout.*SOUTH*);  
  
 *// Панель с кнопками* JPanel buttonPanel = new JPanel();  
 buttonPanel.setLayout(new FlowLayout(FlowLayout.*CENTER*));  
 buttonPanel.setBackground(Color.*WHITE*);  
  
 *// Кнопки "Выйти" и "Далее"* JButton exitButton = new JButton("Выйти");  
 JButton continueButton = new JButton("Далее");  
 buttonPanel.add(continueButton);  
 buttonPanel.add(exitButton);  
  
 *// Действия на кнопки* exitButton.addActionListener(e -> {  
 timer.stop();  
 System.*exit*(0);  
 });  
  
 continueButton.addActionListener(e -> {  
 timer.stop();  
 this.dispose();  
 new MainFrame().setVisible(true);  
 });  
  
 *// Таймер для автоматического закрытия* timer = new Timer(60000, (ActionEvent e) -> {  
 if (this.isVisible()) {  
 this.dispose();  
 System.*exit*(0);  
 }  
 });  
 timer.setRepeats(false);  
 timer.start();  
  
 *// Добавление всех панелей в главное окно* mainPanel.add(textPanel, BorderLayout.*CENTER*);  
 mainPanel.add(bottomPanel, BorderLayout.*SOUTH*);  
  
 setLayout(new BorderLayout());  
 add(mainPanel, BorderLayout.*CENTER*);  
 add(buttonPanel, BorderLayout.*SOUTH*);  
 }  
  
 */\*\*  
 \* Основной метод для запуска приложения и отображения splash screen.  
 \*  
 \* @param args Аргументы командной строки (не используются).  
 \*/* public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> new SplashScreen().setVisible(true));  
 }  
}

**EmailSender.java**

package com.example;  
  
import jakarta.mail.\*;  
import jakarta.mail.internet.\*;  
import javax.swing.\*;  
import java.io.File;  
import java.util.Properties;  
  
*/\*\*  
 \* Класс для отправки email-сообщений с использованием SMTP.  
 \* Предоставляет метод для отправки письма с заданным текстом и темой.  
 \*  
 \* @author Soldatov N. V.  
 \* @version 22.0.2  
 \*/*public class EmailSender {  
  
 private final String username;  
 private final String password;  
 private final String host;  
 private final int port;  
 private final String recipientEmail; *// Новое поле для адреса получателя  
  
 /\*\*  
 \* Конструктор для инициализации параметров подключения и получателя письма.  
 \*  
 \* @param username Имя пользователя для SMTP-сервера.  
 \* @param password Пароль для SMTP-сервера.  
 \* @param host Хост SMTP-сервера.  
 \* @param port Порт SMTP-сервера.  
 \* @param recipientEmail Адрес электронной почты получателя.  
 \*/* public EmailSender(String username, String password, String host, int port, String recipientEmail) {  
 this.username = username;  
 this.password = password;  
 this.host = host;  
 this.port = port;  
 this.recipientEmail = recipientEmail;  
 }  
  
 */\*\*  
 \* Метод для отправки email-сообщения с заданной темой и текстом.  
 \*  
 \* @param subject Тема письма.  
 \* @param text Текст письма.  
 \* @param parentFrame Окно-родитель, которое используется для отображения сообщений об успехе или ошибке.  
 \*/* public void sendEmail(String subject, String text, JFrame parentFrame) {  
 Properties props = new Properties();  
  
 *// Для SSL (порт 465):* props.put("mail.smtp.host", host);  
 props.put("mail.smtp.port", port);  
 props.put("mail.smtp.ssl.enable", "true");  
 props.put("mail.smtp.auth", "true");  
  
 *// Создание сессии для отправки email* Session session = Session.*getInstance*(props, new Authenticator() {  
 @Override  
 protected PasswordAuthentication getPasswordAuthentication() {  
 return new PasswordAuthentication(username, password);  
 }  
 });  
  
 try {  
 *// Создание email-сообщения* Message message = new MimeMessage(session);  
 message.setFrom(new InternetAddress(username));  
 message.setRecipients(Message.RecipientType.*TO*, InternetAddress.*parse*(recipientEmail));  
 message.setSubject(subject);  
 message.setText(text);  
  
 *// Отправка письма* Transport.*send*(message);  
  
 *// Информирование пользователя об успешной отправке письма* SwingUtilities.*invokeLater*(() -> {  
 JOptionPane.*showMessageDialog*(parentFrame, "Письмо успешно отправлено!", "Успех", JOptionPane.*INFORMATION\_MESSAGE*);  
 });  
  
 } catch (MessagingException e) {  
 *// В случае ошибки информируем пользователя* SwingUtilities.*invokeLater*(() -> {  
 JOptionPane.*showMessageDialog*(parentFrame, "Ошибка отправки: " + e.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  
 });  
 e.printStackTrace();  
 }  
 }  
}

**Student.java**

package com.example;  
  
import java.util.HashSet;  
import java.util.Set;  
  
*/\*\*  
 \* Класс, представляющий студента с данными о его посещаемости.  
 \* Хранит информацию о фамилии, имени, отчестве, группе, посещениях и отсутствиях.  
 \*  
 \* @author Soldatov N. V.  
 \* @version 22.0.2  
 \*/*public class Student {  
 private String lastName;  
 private String firstName;  
 private String patronymic; *// Отчество* private String group;  
 private String fullName;  
 private final Set<String> attendanceDates = new HashSet<>(); *// Даты посещений* private final Set<String> absenceDates = new HashSet<>(); *// Даты отсутствий  
  
 /\*\*  
 \* Конструктор для создания студента с указанием фамилии, имени, отчества и группы.  
 \* Полное имя студента будет сформировано автоматически.  
 \*  
 \* @param lastName Фамилия студента.  
 \* @param firstName Имя студента.  
 \* @param patronymic Отчество студента.  
 \* @param group Группа студента.  
 \*/* public Student(String lastName, String firstName, String patronymic, String group) {  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.patronymic = patronymic;  
 this.group = group;  
 this.fullName = getFullName(); *// Формирование полного имени* }  
  
 */\*\*  
 \* Конструктор для создания студента, указывая только полное имя и группу.  
 \*  
 \* @param fullName Полное имя студента.  
 \* @param group Группа студента.  
 \*/* public Student(String fullName, String group) {  
 this.fullName = fullName;  
 this.group = group;  
 }  
  
 */\*\*  
 \* Метод для очистки всех данных о посещениях и отсутствиях.  
 \*/* public void clearAttendance() {  
 attendanceDates.clear();  
 absenceDates.clear();  
 }  
  
 */\*\*  
 \* Получить полное имя студента.  
 \* Если оно уже задано, возвращает его. В противном случае собирает его из фамилии, имени и отчества.  
 \*  
 \* @return Полное имя студента.  
 \*/* public String getFullName() {  
 if (fullName != null && !fullName.isEmpty()) {  
 return fullName;  
 }  
 return (lastName == null ? "" : lastName) + " " +  
 (firstName == null ? "" : firstName) + " " +  
 (patronymic == null ? "" : patronymic).trim();  
 }  
  
 *// Методы доступа  
  
 /\*\*  
 \* Получить фамилию студента.  
 \*  
 \* @return Фамилия студента.  
 \*/* public String getLastName() {  
 return lastName;  
 }  
  
 */\*\*  
 \* Получить имя студента.  
 \*  
 \* @return Имя студента.  
 \*/* public String getFirstName() {  
 return firstName;  
 }  
  
 */\*\*  
 \* Получить отчество студента.  
 \*  
 \* @return Отчество студента.  
 \*/* public String getPatronymic() {  
 return patronymic;  
 }  
  
 */\*\*  
 \* Получить группу студента.  
 \*  
 \* @return Группа студента.  
 \*/* public String getGroup() {  
 return group;  
 }  
  
 */\*\*  
 \* Установить новую группу для студента.  
 \*  
 \* @param group Новая группа.  
 \*/* public void setGroup(String group) {  
 this.group = group;  
 }  
  
 */\*\*  
 \* Получить все даты посещений студента.  
 \*  
 \* @return Множество дат посещений.  
 \*/* public Set<String> getAttendanceDates() {  
 return attendanceDates;  
 }  
  
 */\*\*  
 \* Получить все даты отсутствий студента.  
 \*  
 \* @return Множество дат отсутствий.  
 \*/* public Set<String> getAbsenceDates() {  
 return absenceDates;  
 }  
  
 */\*\*  
 \* Добавить дату посещения студента.  
 \* Если студент был ранее отмечен как отсутствующий в этот день, дата будет удалена из списка отсутствий.  
 \*  
 \* @param date Дата посещения.  
 \*/* public void addAttendanceDate(String date) {  
 attendanceDates.add(date);  
 absenceDates.remove(date); *// Удалить из отсутствий, если есть* }  
  
 */\*\*  
 \* Добавить дату отсутствия студента.  
 \* Если студент был ранее отмечен как посещающий в этот день, дата будет удалена из списка посещений.  
 \*  
 \* @param date Дата отсутствия.  
 \*/* public void addAbsenceDate(String date) {  
 absenceDates.add(date);  
 attendanceDates.remove(date); *// Удалить из посещений, если есть* }  
  
 */\*\*  
 \* Переопределение метода toString для удобного вывода информации о студенте.  
 \* Возвращает строку с полным именем, группой, датами посещений и отсутствий.  
 \*  
 \* @return Строка с информацией о студенте.  
 \*/* @Override  
 public String toString() {  
 String attendanceStr = attendanceDates.isEmpty() ? "Нет посещений" : String.*join*(", ", attendanceDates);  
 String absenceStr = absenceDates.isEmpty() ? "Нет отсутствий" : String.*join*(", ", absenceDates);  
 return String.*format*("ФИО: %s, Группа: %s, Посещения: %s, Отсутствия: %s",  
 getFullName(), group, attendanceStr, absenceStr);  
 }  
}

**pom.xml**

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.example</groupId>  
 <artifactId>student-attendance</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 *<!-- Для работы с Excel -->* <dependency>  
 <groupId>org.apache.poi</groupId>  
 <artifactId>poi</artifactId>  
 <version>5.2.3</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.poi</groupId>  
 <artifactId>poi-ooxml</artifactId>  
 <version>5.2.3</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.xmlbeans</groupId>  
 <artifactId>xmlbeans</artifactId>  
 <version>5.1.1</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-collections4</artifactId>  
 <version>4.4</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-api</artifactId>  
 <version>2.17.1</version> *<!-- Или более поздняя, если доступна -->* </dependency>  
  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-core</artifactId>  
 <version>2.17.1</version> *<!-- Используйте ту же версию, что и для log4j-api -->* </dependency>  
  
 *<!-- Для тестирования (JUnit) -->* <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 *<!-- Для отправки email через Jakarta Mail  
 <dependency>  
 <groupId>jakarta.mail</groupId>  
 <artifactId>jakarta.mail-api</artifactId>  
 <version>2.1.0</version> &lt;!&ndash; Используйте актуальную версию &ndash;&gt;  
 </dependency>-->* <dependency>  
 <groupId>com.sun.mail</groupId>  
 <artifactId>jakarta.mail</artifactId>  
 <version>2.0.1</version> *<!-- Или более новую версию -->* </dependency>  
  
 <dependency>  
 <groupId>com.toedter</groupId>  
 <artifactId>jcalendar</artifactId>  
 <version>1.4</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.sun.activation</groupId>  
 <artifactId>jakarta.activation</artifactId>  
 <version>2.0.1</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <resources>  
 <resource>  
 <directory>src/main/resources</directory>  
 </resource>  
 </resources>  
  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>