**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет Инфокоммуникационных сетей и систем

Кафедра Защищенных систем связи

Дисциплина Разработка защищенных сетевых приложений

**Отчет по курсовой работе**

Анализ успеваемости студентов в группе по отчетной ведомости

Студент:

Бунеску Г.П ИБС-23

*(Ф.И.О., № группы) (подпись)*

Преподаватель:

Асс.к. ЗСС Шариков П.И

*(уч. степень, уч. звание, Ф.И.О.) (подпись)*

# Содержание

[Содержание 2](#_Toc153712732)

[Введение 3](#_Toc153712733)

[Глава 1. Подготовка к написанию программы 4](#_Toc153712734)

[1.1 Выбор IDE среды. 4](#_Toc153712735)

[1.2 Пакеты и библиотеки в Java. 5](#_Toc153712736)

[Глава 2. Программа и взаимодействия с ней. 12](#_Toc153712737)

[2.1 Основной алгоритм работы. 17](#_Toc153712738)

[2.2 Описание переменных. 19](#_Toc153712739)

[2.3 Описание классов. 20](#_Toc153712740)

[2.4 Описание методов. 20](#_Toc153712741)

[2.5 Цель каждого метода и класса. 20](#_Toc153712742)

[2.6 Процесс работы программы. 21](#_Toc153712743)

[2.7 Принцип переиспользования. 25](#_Toc153712744)

[2.8 Обработка исключений. 26](#_Toc153712745)

[2.9 Создание Jar файла. 27](#_Toc153712746)

[2.10 Работа с UML. 29](#_Toc153712747)

[Глава 3. GitHub и его система. 31](#_Toc153712748)

[3.1 GitHub 31](#_Toc153712749)

[3.2 Загрузка кода на GitHub 33](#_Toc153712750)

[3.3 Система Git 34](#_Toc153712751)

[Глава 4 Заключение. 36](#_Toc153712752)

[Список использованных источников 37](#_Toc153712753)

# Введение

Написание программы — это не только увлекательное занятие, но и процесс, требующий полного внимания и систематического подхода. Каждая программа независимо от своего уровня сложности имеет свою цель. Одни программы требуют глубокого погружения в материал, чтобы каждая строчка кода работала слаженно с остальными, в то время как другие могут иметь более простые цели, например, вывод на экран всем знакомой фразы "Hello World".

Для написания программы, о которой идет речь в данной работе, а то есть: необходимо разобраться в библиотеке ApachePOI.

Приложение получает на вход путь до excel файла с таблицой успеваемости студентов по какому-либо предмету. В таблице 2 столбца: ФИО и оценка. Необходимо произвести парсинг Excel файла, проанализировать результаты группы и вывести их в другой excel-файл. Результаты оформить также в виде таблицы. В результатах: количество отличников, хорошистов, троешников. Средний балл группы по предмету. ФИО (в разных столбцах) тех, кто получил 5, 4, 3, не допущен к экзамену. В приложении учесть: обработку ошибок, максимальную оценку, графу не допущен, построение графика на основе статистики группы. Необходимо было овладеть синтаксисом языка программирования, в данном случае, "Java", а также обладать знаниями о различных библиотеках и пакетах для достижения поставленной цели. Весь процесс начался с тщательного изучения требований задачи и анализа возможных решений для эффективной реализации этих требований.

Итак, погружение в создание программы требует не только понимания основ языка программирования, но и умения выбирать оптимальные решения, исходя из поставленных задач. Весь этот процесс начинается с ясного определения требований и заканчивается разработкой работоспособного и эффективного программного решения.

Глава 1. Подготовка к написанию программы

## **1.1** Выбор IDE среды**.**

Стоит начать со среды, в которой будет написана та или иная программа. Есть большое количество IDE сред, которые могут предоставить нам удобный интерфейс и функции. Но остановимся на «IntelliJ IDEA».

IntelliJ IDEA — это одна из наиболее популярных интегрированных сред разработки (IDE) для языка Java. Вот несколько плюсов и минусов при использовании IntelliJ IDEA:

Плюсы IntelliJ IDEA:

* Предоставление широкого спектра инструментов для написания кода.
* Автодополнение.
* Быстрая навигация.
* Удобство работы с Maven и Gradle, о которых будет сказано ниже.
* Имеет встроенную поддержку систем управления зависимостями Maven и Gradle, что упрощает добавление библиотек в проекты.
* Мощные инструменты отладки.
* Поддержка тестирования: имеет инструменты для написания и запуска тестов, включая интеграцию с JUnit и другими тестовыми фреймворками.
* Интуитивноы понятный интерфейс и удобство использования: предлагает интуитивный интерфейс, который делает процесс разработки более продуктивным.

Минусы IntelliJ IDEA:

* Ресурсоемкость: иногда IntelliJ IDEA может потреблять больше ресурсов компьютера, особенно при работе с большими проектами.
* Платная версия: хотя есть бесплатная Community версия, полная Ultimate версия является платной.

Почему стоит использовать IntelliJ IDEA:

Продуктивность: IntelliJ IDEA обладает множеством инструментов, которые значительно повышают продуктивность разработчика, помогая быстрее писать, отлаживать и поддерживать код.

Широкая поддержка: Поддержка множества технологий и фреймворков делает его отличным выбором для разработки Java-приложений разной сложности. Активное сообщество и обновления: IntelliJ IDEA имеет активное сообщество пользователей и регулярные обновления, что обеспечивает поддержку новых функций и улучшение производительности. Выбор IDE часто зависит от личных предпочтений, но IntelliJ IDEA обладает широким набором функций, что делает его популярным среди Java-разработчиков. Версия Java SE на которой была написана программа – Java SE 16. Далее стоит перейти к библиотекам и способам их установки.

## **1.2 Пакеты и библиотеки в Java.**

В мире Java термины "пакеты" и "библиотеки" имеют ключевые различия, но оба они играют важную роль в организации и повторном использовании кода. В контексте Java, библиотека представляет собой собрание классов и методов, которые предоставляют определенную функциональность для использования в ваших Java-программах. Библиотека содержит уже написанный и протестированный код, который вы можете использовать для выполнения определенных задач без необходимости писать их с нуля. Для работы с программой для создания таблиц «Excel» потребуется использовать библиотеку «Apache POI». Данный сборник классов позволит читать, редактировать и создавать таблицы формата «XLSX». Так-же понадобится библиотека для построения графиков, так как Apache POI не умеет их создавать. Для этой задачи используется «JFreeChart». Данная библиотека позволяет создавать диаграммы разных видов. Все остальные задачи, поставленные программой, можно реализовать стандартными функциями, библиотеками и пакетами, которые имеются в языке программирования.

Стоит поподробнее затронуть тему установки библиотек. Их установка в Java обычно отличается от установки в других языках программирования, таких как Python или JavaScript. В Java используются специальные файлы JAR (Java Archive), которые содержат библиотеки и классы, необходимые для выполнения определенных функций.

Основные способы установки библиотек в Java:

1. Использование средств управления зависимостями

Maven

Maven — это мощный инструмент автоматизации сборки проектов, который облегчает управление зависимостями, сборку и развертывание программного обеспечения в Java. Он предоставляет структурированную систему управления проектами, обеспечивая повторное использование кода и стандартизацию процесса сборки.

Установка Maven

1. Загрузка Maven: Перейдите на официальный сайт Apache Maven и загрузите актуальную версию Maven.
2. Распаковка архива: после загрузки архива установки, распакуйте его в желаемую директорию на вашем компьютере.
3. Настройка переменных среды (опционально): для использования Maven из командной строки можно добавить путь к исполняемым файлам Maven в переменную среды PATH. Это позволит запускать Maven из любой директории в командной строке.

Проверка установки Maven

Чтобы убедиться, что Maven установлен правильно, нужно выполнить команду «mvn -version» в командной строке. Если Maven установлен корректно, то появится информация о версии Maven и JDK.

Основой проекта Maven является файл pom.xml (Project Object Model), который описывает структуру проекта, зависимости, настройки сборки и другую важную информацию о проекте. Для управления зависимостями(библиотеками), необходимыми для проекта, добавляется соответствующая информация в раздел <dependencies> файла pom.xml. Это позволяет Maven автоматически загружать требуемые библиотеки из центральных репозиториев.

Пример добавления зависимости в файл pom.xml:

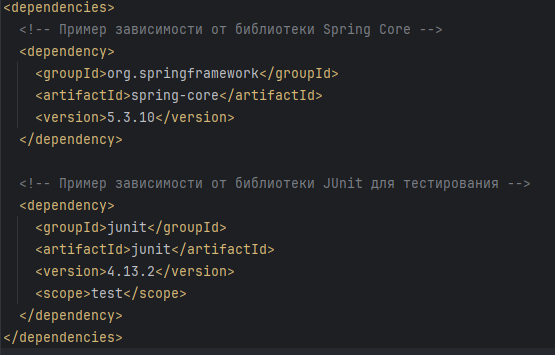


Рис. 1.2.1(Зависимость)

После добавления таких зависимостей Maven при выполнении сборки проекта автоматически загрузит указанные библиотеки из репозиториев и добавит их в ваш проект, что позволит использовать функционал этих библиотек в коде вашего приложения.

Gradle

Gradle - еще один инструмент управления зависимостями, используемый в Java. Добавление зависимостей в проект Gradle происходит в файле build.gradle:

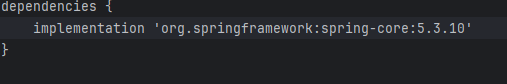


Рис 1.2.2(Зависимость с Graddle)

Этот код добавляет зависимость от библиотеки Spring Core версии 5.3.10 в ваш проект. Gradle автоматически загрузит эту библиотеку из соответствующего репозитория при сборке проекта.Начало формы

Выполнение команды gradle build загрузит библиотеки из репозитория Gradle.

Gradle устанавливаается практически идентично Maven.

2. Ручная установка

Также можно вручную загрузить JAR файлы и добавить их в проект:

Для этого нужно скачать JAR файлы, для этого можно использовать базу библиотек «Maven Central».

Добавьте их в проект: в большинстве IDE существует функциональность для добавления внешних библиотек. Вы можете добавить JAR файлы в папку проекта и указать их как внешние библиотеки. В «IntelliJ» достаточно указать путь к Jar файлам через вкладку «Project Structure» и «Libraries».

Для реализации моей программы использовался второй способ, так как он имеет довольно простой и понятный алгоритм, а так-же не требует никаких внешних установок, не считая скачивания самих библиотек.

Далее представлен список библиотек, которые были установлены:

JFreeChart:

Первые три набора предоставляют набор методов для проверки условий в тестах JUnit и других тестовых фреймворках:

* hamcrest-core-1.3.jar
* jcommon-1.0.23.jar
* jfreechart-1.0.19.jar Они предоставляют классы и методы для создания различных типов графиков (линейные, круговые, столбчатые и т. д.) в приложениях Java:

Ниже находятся библиотеки связанные с JFreeChart и предлагающие экспериментальные возможности, поддержку SWT (Standard Widget Toolkit), SVG (Scalable Vector Graphics) и так далее:

* jfreechart-1.0.19-experimental.jar
* jfreechart-1.0.19-swt.jar
* jfreechart-1.5.4.jar
* jfreesvg-2.0.jar

junit-4.11.jar (Они предоставляют функционал для написания и выполнения тестовых случаев в Java.)

Следующие две библиотеки связаны с Orson Charts и Orson PDF, которые предоставляют возможности для создания и работы с графиками и PDF файлами соответственно.

* orsoncharts-1.4-eval-nofx.jar
* orsonpdf-1.6-eval.jar
* servlet.jar (Предоставляет функциональность для разработки веб-приложений с использованием сервлетов)
* swtgraphics2d.jar (позволяет использовать возможности Java2D (например, рисование фигур, текста, изображений) внутри окон библиотеки SWT.)

ApachePOI:

Данный набор позволяет создавать, читать и изменять документы в форматах Office (например, файлы Excel, Word) с помощью Java:

* poi-4.0.0.jar
* poi-examples-4.0.0.jar
* poi-excelant-4.0.0.jar
* poi-ooxml-4.0.0.jar
* poi-ooxml-schemas-4.0.0.jar
* poi-ooxml-schemas-4.0.0.jar

Далее библиотеки являются вспомогательными для работы с форматами документов, обеспечивая дополнительные функции.

(кодирование/декодирование, логирование и другие вспомогательные инструменты для работы с файлами и данными в Java)

* commons-codec-1.11.jar
* commons-collections4-4.2.jar
* commons-logging-1.2.jar
* commons-math3-3.6.1.jar
* jaxb-api-2.3.0.jar
* jaxb-core-2.3.0.1.jar
* jaxb-impl-2.3.0.1.jar
* junit-4.12.jar
* log4j-1.2.17.jar

Не все библиотеки из данного списка были использованы, но было решено их оставить ведь данные Jar файлы шли все вместе в одной папке, что позволило их сразу вместе установить, а как таковой нагрузки из-за них не возникает.

Далее разберем что такое пакеты и для чего они нужны в программе:

Пакеты — это организационная единица, используемая для группировки связанных классов и интерфейсов. Они помогают структурировать код, делая его более читаемым и упорядоченным. Пакеты позволяют избежать конфликтов имен, так как классы в разных пакетах могут иметь одинаковые имена без конфликтов.

В Java пакеты используются для организации классов и интерфейсов в логически связанные группы. Они помогают структурировать код и делают его более читаемым, а также предотвращают конфликты имен. Каждый пакет содержит классы, которые относятся к определенной теме или области функциональности.

Пакеты в реализованной программе:

* org.apache.poi.ss.usermodel (Этот конкретный пакет содержит интерфейсы и классы, представляющие функциональность для работы с документами Excel в Apache POI.)
* org.apache.poi.xssf.usermodel (Здесь содержатся классы и интерфейсы, относящиеся к работе с файлами формата XLSX (Excel 2007 и более новые версии) в Apache POI.)
* org.jfree.chart (Здесь находятся основные классы и методы, предоставляющие возможности для создания различных типов графиков (линейные, круговые, столбчатые и другие) в JFreeChart.)
* java.io (Этот пакет предоставляет классы для работы с потоками ввода-вывода, файлами, сериализацией и другими операциями ввода-вывода в Java.)
* java.util (Это еще один стандартный пакет Java, содержащий различные утилитарные классы и интерфейсы для работы с коллекциями данных, датами, временем, случайными числами и другими структурами данных.)

# **Глава 2. Программа и взаимодействия с ней.**

Теперь, когда имеются все нужные библиотеки и пакеты, а также понимание того, что они делают, можно перейти к самой программе.

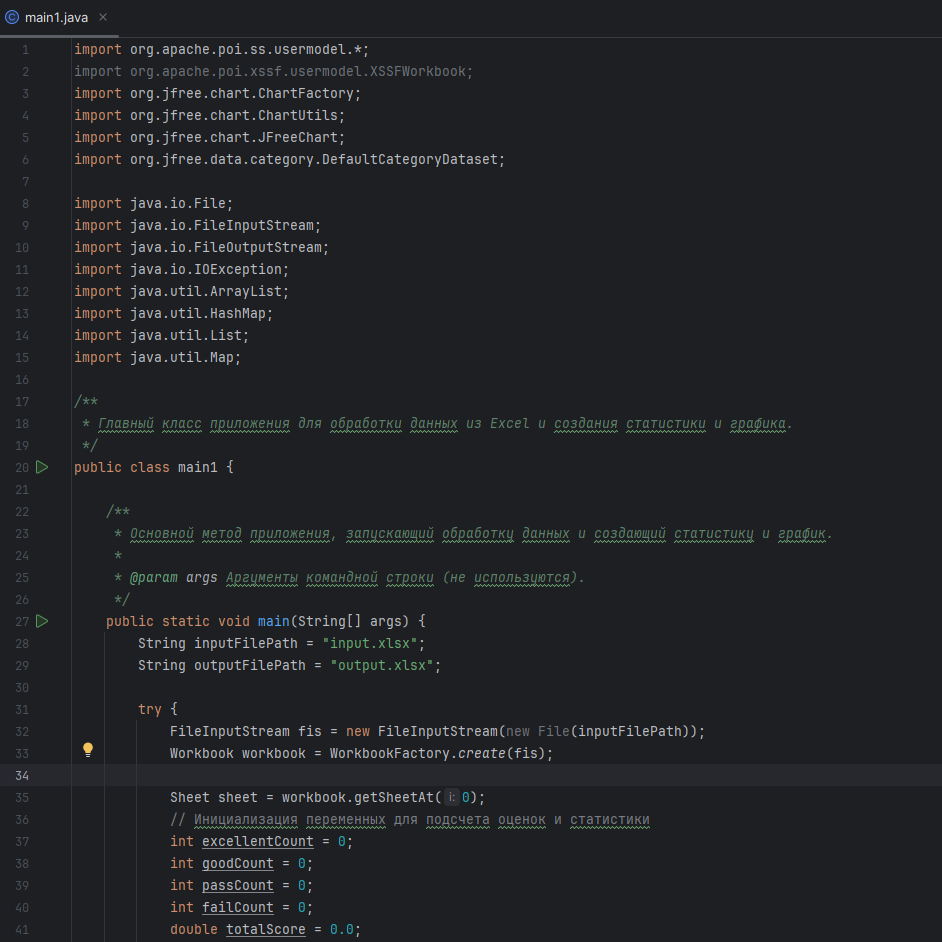


Рис. 2.1 (Программа)

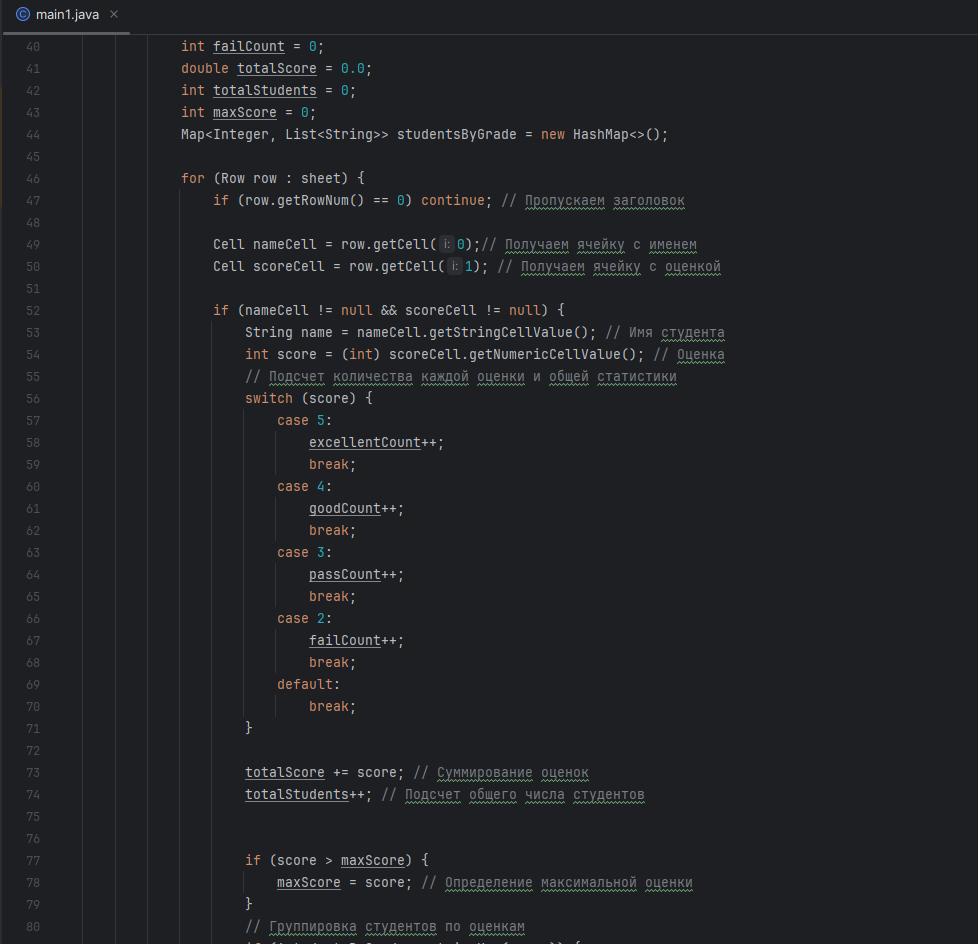


Рис. 2.2 (Программа)

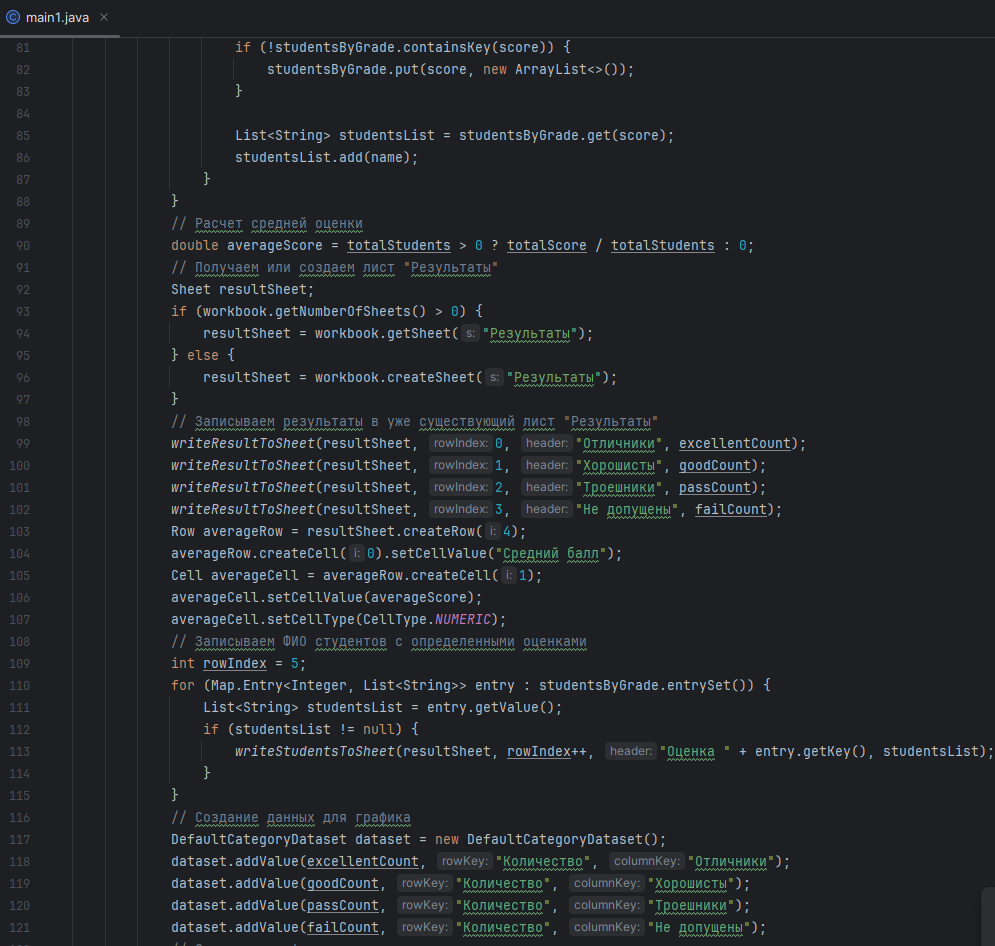


Рис. 2.3 (Программа)

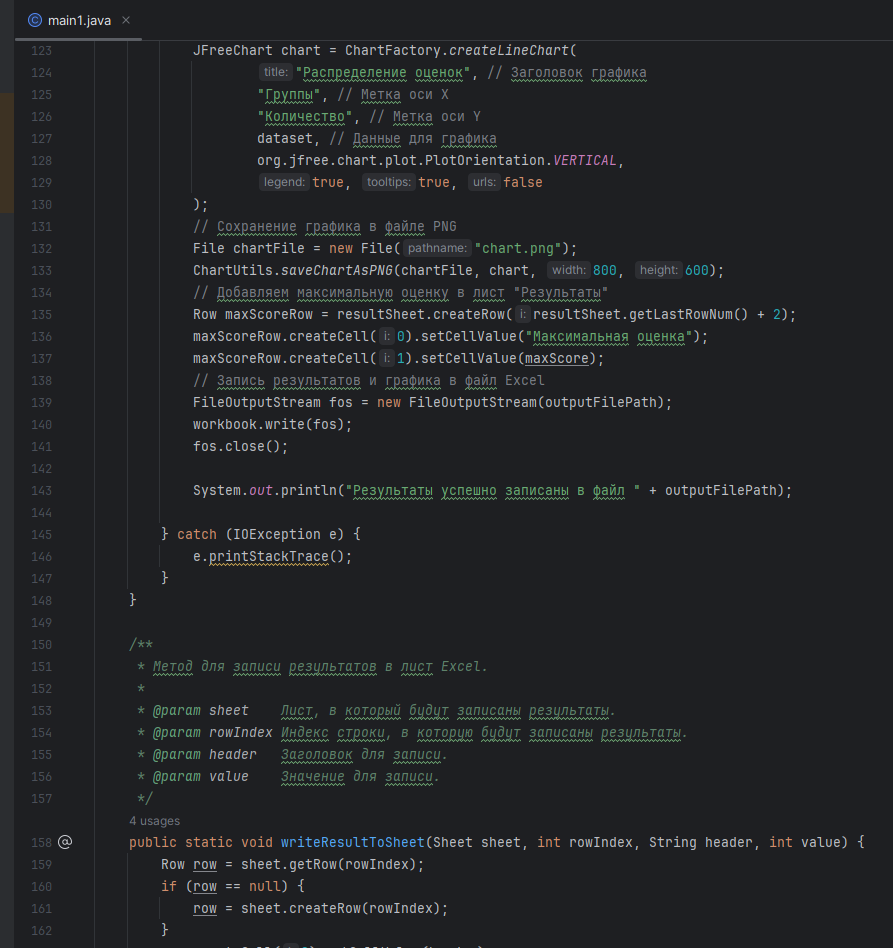


Рис. 2.4 (Программа)

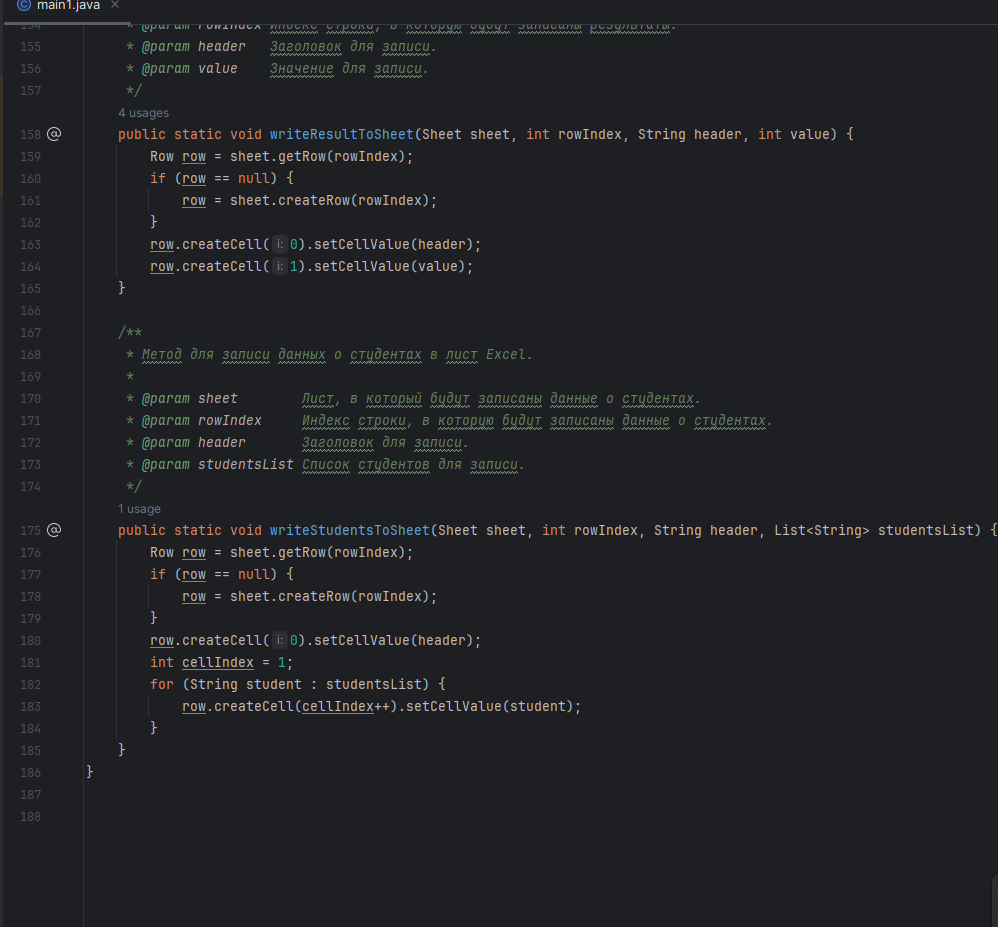


Рис. 2.5 (Программа)

## 2.1 Основной алгоритм работы.

Для начала стоит разобрать основной алгоритм работы программы.

1. Чтение данных из файла Excel

* FileInputStream fis = new FileInputStream(new File(inputFilePath)); (Создается поток чтения из файла inputFilePath.)
* Workbook workbook = WorkbookFactory.create(fis); (Создается объект Workbook из файла, содержащего всю книгу Excel.)
* Sheet sheet = workbook.getSheetAt(0); (Получение первого листа из книги.)

2. Анализ данных и подсчет статистики

* Цикл по строкам (студентам) листа Excel:
* Пропуск первой строки (предположительно, это заголовок).
* Извлечение ячеек с именем и оценкой каждого студента.
* Подсчет количества каждой оценки и общей статистики:
* Определение категорий студентов по оценкам (отличники, хорошисты и т.д.).
* Суммирование оценок для расчета среднего балла.
* Определение максимальной оценки.
* Группировка студентов по оценкам в Map<Integer, List<String>> studentsByGrade.

3. Создание листа Excel с результатами

Создание или получение листа "Результаты":

* Если в файле уже существует лист "Результаты", он получается для записи результатов. Иначе создается новый лист.

Запись статистики и среднего балла:

* Создание строк и ячеек для различных категорий студентов и среднего балла.

Запись данных о студентах по оценкам:

* Создание строк и ячеек для группировки студентов по оценкам.

4. Создание графика

Подготовка данных для графика:

* Создание DefaultCategoryDataset и добавление в него количества студентов по каждой оценке.

Создание графика:

* Использование ChartFactory для создания графика на основе подготовленных данных.
* Указание заголовка, меток осей и других параметров графика.

5. Запись результатов в новый файл Excel

Создание файла для вывода результатов:

* Создание новой рабочей книги Excel XSSFWorkbook.
* Запись результатов (статистика, средний балл, данные о студентах) в этот новый файл.

Сохранение графика в файле PNG:

* Сохранение созданного графика в файл chart.png.
* Добавление максимальной оценки в результаты:
* Создание строки и ячейки для максимальной оценки и ее запись.

Запись результатов в новый файл Excel:

* Запись всех результатов и графика в файл outputFilePath.

6.Дополнительные замечания

Исключения:

* Обработка возможных исключений, связанных с чтением/записью файлов.
* Это основной принцип программы, далее опишем все переменные, классы и методы в программе.

## 2.2 Описание переменных.

В основном коде переменные используются для различных целей:

* inputFilePath и outputFilePath: строки, представляющие пути к входному файлу Excel (inputFilePath) и файлу, в который будут записаны результаты (outputFilePath).
* fis: объект FileInputStream для чтения данных из входного файла.
* workbook: объект Workbook из библиотеки Apache POI, представляющий книгу Excel, в которой содержатся данные.
* sheet: объект Sheet, который представляет лист в книге Excel.
* excellentCount, goodCount, passCount, failCount: переменные для подсчета количества студентов с разными оценками.
* totalScore: общий балл, который используется для вычисления среднего балла.
* totalStudents: общее количество студентов.
* maxScore: максимальная оценка среди всех студентов.
* studentsByGrade: Map, где ключ - оценка, а значение - список студентов с этой оценкой.
* averageScore: средний балл всех студентов.
* resultSheet: лист, куда записываются результаты статистики.
* chartFile: файл, в который сохраняется график в формате PNG.
* fos: объект FileOutputStream для записи данных в выходной файл.
* dataset: объект DefaultCategoryDataset, который содержит данные для графика.
* chart: объект JFreeChart, который представляет собой график.

## 2.3 Описание классов.

Классы:

main1 (public class main1):

* main(String[] args): Это основной метод, который выполняет программу. Он обрабатывает данные из файла Excel, подсчитывает статистику оценок студентов и создает результаты в новом файле Excel.

## 2.4 Описание методов.

Методы:

* main (String[] args):

Основной метод, который выполняет программу. Он сначала открывает файл Excel, читает данные, анализирует каждую оценку студентов и подсчитывает количество каждой оценки. После этого он создает новый файл Excel, в который записывает статистику оценок, средний балл, список студентов, по каждой оценке, и график, отображающий распределение оценок.

* writeResultToSheet (Sheet sheet, int rowIndex, String header, int value):

Метод, который записывает результаты (название категории и значение) в указанный лист Excel в указанную строку. Например, это может быть запись количества студентов с оценкой "Отлично" в строку "Отличники".

* writeStudentsToSheet (Sheet sheet, int rowIndex, String header, List<String> studentsList):

Метод, который записывает данные о студентах (список имен) с определенной оценкой в указанный лист Excel в указанную строку. Например, это может

быть запись всех студентов с оценкой "Отлично" в строку "Оценка 5".

## 2.5 Цель каждого метода и класса.

Цель каждого класса/метода:

* main1 (public class main1):

main (String[] args): Этот метод управляет основным процессом программы, выполняя последовательность действий от чтения данных из файла до создания и записи результатов в новый файл.

* writeResultToSheet(Sheet sheet, int rowIndex, String header, int value):

Этот метод предназначен для записи сводной информации о количестве студентов, по каждой оценке, в указанный лист Excel.

* writeStudentsToSheet (Sheet sheet, int rowIndex, String header, List<String> studentsList):

Этот метод предназначен для записи списка студентов с определенной оценкой в указанный лист Excel.

Каждый из этих методов играет свою роль в обработке данных, подготовке результатов и их записи в файл Excel. Их цель - облегчить структурирование кода и повторное использование функциональности для работы с данными оценок студентов. Стоит начать с создания самого xlsx файла, назовем его input.xlsx.

## 2.6 Процесс работы программы.

Для работы программы нужно создать Excel таблицу, в которой будут нужные нам данные. Таблица состоит из двух столбцов, в которых. Таблица, выполненная в Excel, с двумя столбцами для студентов и их оценок, представляет собой набор данных, где каждая строка содержит информацию об ученике и его оценке.

Столбец "Студент": Этот столбец содержит имена учеников, каждое имя размещено в отдельной ячейке. Каждая строка таблицы представляет уникального ученика.

Столбец "Оценка": Этот столбец содержит оценки, соответствующие каждому ученику в столбце "Студент". Оценки также размещаются в отдельных ячейках под каждым учеником.

Эти данные будут обработаны программами для создания статистики по успеваемости класса, вычисления средней оценки, определения количества учеников с разными оценками и многое другое.

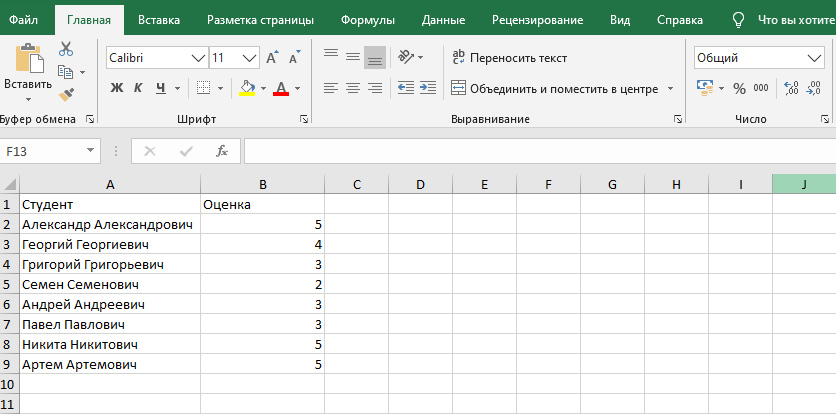


Рис. 2.6.1 (Исходный файл)

Далее запускаем программу, которая в свою очередь будет получать данные из созданной выше таблицы.

На выходе программы можно увидеть такой вывод в консоль, а так-же два новых созданных файла, input.xlsx и chart.png.

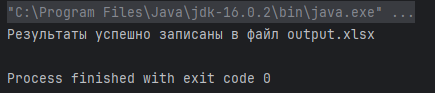


Рис 2.6.2 (Итог работы программы)

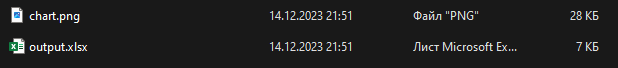


Рис 2.6.3(Созданные файлы)

Input.xlsx представляет из себя xlsx таблицу в которой дано: количество отличников, хорошистов, троешников. Средний балл группы по предмету. ФИО (в разных столбцах) тех, кто получил 5, 4, 3, не допущен к экзамену.

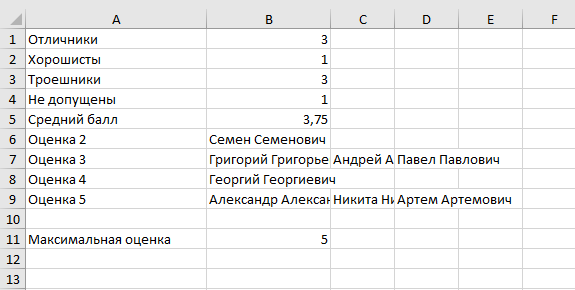


Рис. 2.6.4(Полученный Excel файл)

Файл «chart.png» представляет собой график, где показана статистика, показывающая какое количество людей с тройками, четверками и пятерками, а также не допущенные к предмету, то есть двоечники.

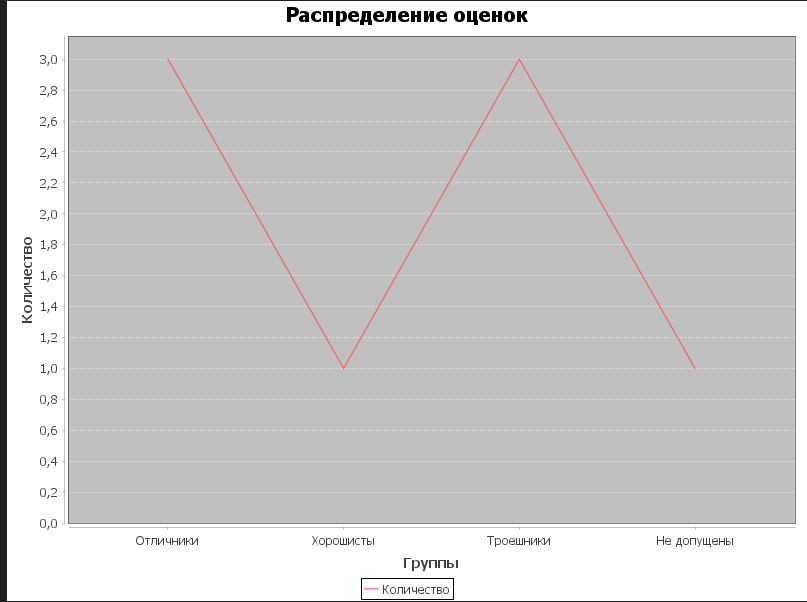


Рис. 2.6.5(График)

Программа выполняет все требуемые задачи и работает стабильно.

Плюсы программы:

1)Перезаписывает файл если файл с таким именем уже был создан.

2)Быстро работает, так как библиотеки и сами программа небольшие, не вызывая как таковой нагрузки.

3)Библиотеки и пакеты, используемые в программе, позволяют ее удобно редактировать или добавлять что-либо при надобности.

Но, как и у любого приложения, у него тоже есть свои минусы.

1) Если изображение или таблица Excel были открыты, то программа просто выдаст ошибку, не выполнив никаких операций, например создания новых файлов.

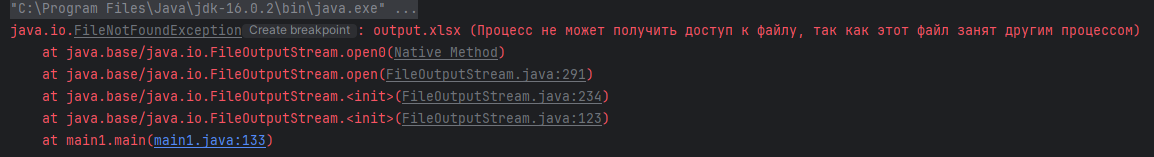


Рис. 2.6.6(Ошибка при запуске)

2)При создании Excel таблицы имена раскидываются по разным ячейкам, но ширина этих ячеек бывает мала для того, чтобы увидеть все имя. Данный недочет можно исправить с помощью «setColumnWidth». Но данный аспект приложения имеет скорее косметический эффект, поэтому некритичен.

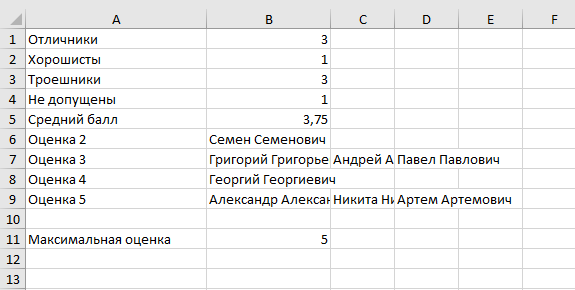


Рис. 2.6.7 (Проблема с шириной столбцов)

3)Данная программа требует установки библиотек, что дополнительно нагружает среду программирования, а также дополнительная затрата времени на поиск и установку библиотек.

Несмотря на все минусы, данная программа является неплохим решением для создания таблицы со статистикой успеваемости группы студентов.

## 2.7 Принцип переиспользования.

Принцип повторного использования кода — это довольно интересная и важная тема. Принцип переиспользования программы подразумевает создание такого кода, который может быть использован не только в текущем контексте, но и в других схожих ситуациях или проектах. Вот некоторые принципы, которые помогают обеспечить переиспользование кода:

* Модульность: разделите программу на небольшие модули или компоненты, каждый из которых отвечает за определенную функциональность. Это позволит вам использовать эти модули в других проектах или частях программы.
* Интерфейсы: определите четкие интерфейсы для ваших модулей, чтобы другие части программы могли легко взаимодействовать с ними. Использование интерфейсов позволяет заменять или расширять конкретную реализацию, не изменяя внешний код.
* Компоненты с настраиваемыми параметрами: Сделайте ваши компоненты настраиваемыми, чтобы их можно было легко настраивать и использовать в разных контекстах. Это может быть достигнуто с помощью параметров, конфигурационных файлов или инъекции зависимостей.
* Документация и тестирование: Пишите хорошую документацию к вашему коду, объясняющую, как его использовать. Также создавайте тесты, которые проверяют функциональность ваших компонентов, чтобы быть уверенными в их правильной работе при повторном использовании.
* Повторное использование существующих библиотек и фреймворков: Используйте сторонние библиотеки и фреймворки для решения типовых задач, вместо написания собственных решений. Это сократит время разработки и улучшит качество вашего кода.
* Паттерны проектирования: Применение известных паттернов проектирования, таких как Фабрика, Стратегия, Наблюдатель и другие, может помочь в создании гибких и переиспользуемых решений.

Переиспользование кода важно для создания эффективного и надежного программного обеспечения. Помните, что хорошо спроектированный, гибкий и тестируемый код упрощает его переиспользование и поддержку в долгосрочной перспективе.

В в программе есть элементы, способствующие переиспользованию кода:

* Методы и структуры данных: в программе используются методы для обработки данных из Excel, создания статистики и графиков. Эти методы могут быть переиспользованы для обработки аналогичных данных в других программах или сценариях.
* Использование сторонних библиотек: Вы используете библиотеки Apache POI и JFreeChart для работы с Excel и создания графиков. Использование таких библиотек позволяет не писать собственные решения для этих задач и переиспользовать их функциональность в других проектах.
* Функции для записи в Excel: имеются методы `writeResultToSheet` и `writeStudentsToSheet`, которые записывают результаты в лист Excel. Эти функции могут быть переиспользованы для записи различных данных в Excel в других проектах.
* Структуры данных: Использование Map для хранения студентов по оценкам позволяет удобно группировать данные. Это может быть полезным для обработки и анализа данных в других программах.

Хотя код содержит элементы, которые могут быть переиспользованы, можно дополнительно улучшить переиспользуемость, разделяя код на более модульные части, чтобы облегчить понимание и использование вашего кода другими разработчиками.

## 2.8 Обработка исключений.

В коде также присутствует обработка исключений с помощью блока try-catch. В вашем основном методе (main) есть обработка IOException, которая может возникнуть при чтении или записи файлов. Этот блок try-catch обертывает чтение данных из файла, работу с книгой Excel, создание графика, сохранение результатов в файл и другие операции, которые могут вызвать исключения ввода-вывода (IOException)

## 2.9 Создание Jar файла.

В IntelliJ IDEA создание JAR файла обычно выполняется через инструменты сборки проекта или с помощью функций экспорта проекта. Рассмотрим основные шаги:

Открытие проекта: открыть проект в IntelliJ IDEA.

Сборка проекта: зайти во вкладку "Build" в верхнем меню IntelliJ IDEA. Выбрать "Build Artifact". Если в проекте уже настроены артефакты сборки, выберите нужный артефакт для создания JAR файла. Если артефакты еще не настроены, нужно выбрать "Edit Artifacts", чтобы добавить новый артефакт.

Создание артефакта: Нажмите на плюсик в окне артефактов и выберите "JAR", затем настройте параметры JAR файла: укажите главный класс (если необходимо), пути к файлам для включения в JAR и другие опции сборки.

Собрать JAR: после того как настройки артефакта выполнены, нажмите "OK" и затем выберите "Build" -> "Build Artifacts". Выберите нужный артефакт для сборки JAR файла.

Поиск JAR файла: после сборки JAR файл будет доступен в директории проекта, обычно в папке out или target.

По итогу получается Jar файл с нужной нам программой.

Проверим работоспособность данной программы. Это можно сделать минимум двумя способами: запуск через «IntelliJ IDEA» или через запуск в консоли windows. Возьмем второй способ.

Для начала в поисковой строке системы вводим «cmd» и запускаем от имени администратора выше появившееся приложение.

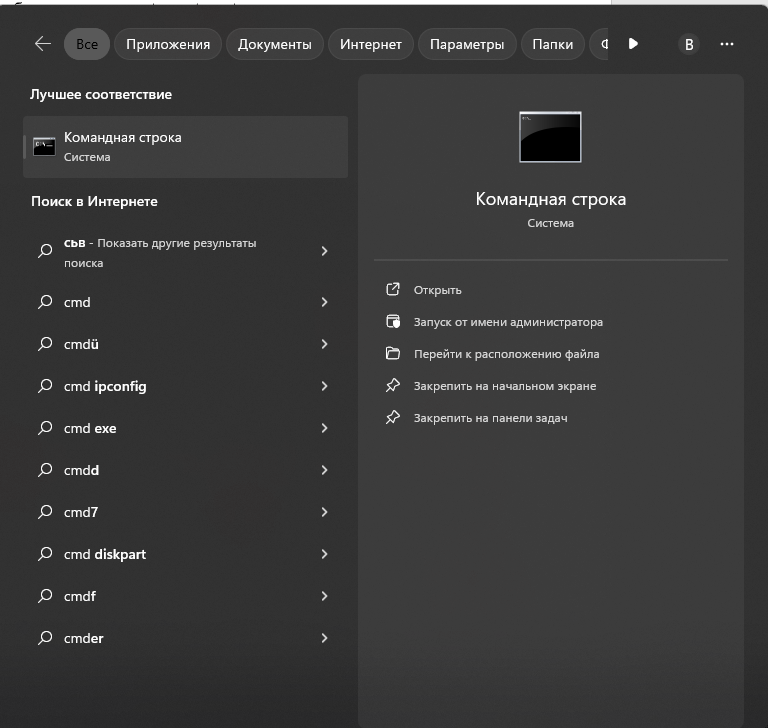


Рис. 2.9.1(Командная строка)

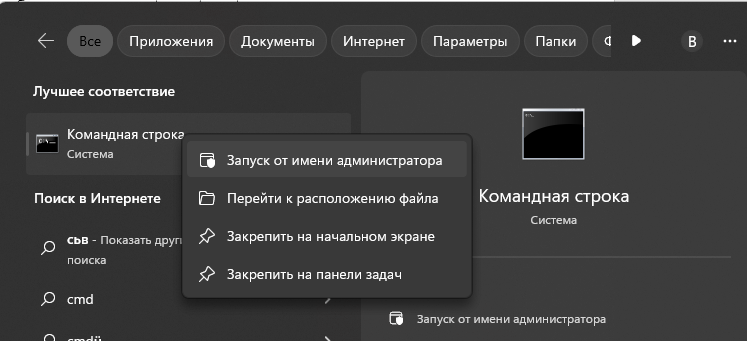


Рис 2.9.2(Открытие командной строки)

Далее переходим в нужный диск и папку, а точнее в диск и папку с программой.

Для этого вводим название диска и двоеточие, (В моем случае «D:»)

В следующей строке вводим «cd» и путь до папки с проектом (В моем случае это «cd \programs\untitled1»)

Находясь в папке с проектом, можно его запустить, важно учесть то, что файлы, которые программа берет на вход (если таковые имеются) требуется хранить в папке с проектом.

Для запуска программы нужно ввести «java -jar и название вашей программы». (в моем случае это java -jar untitled1.jar)

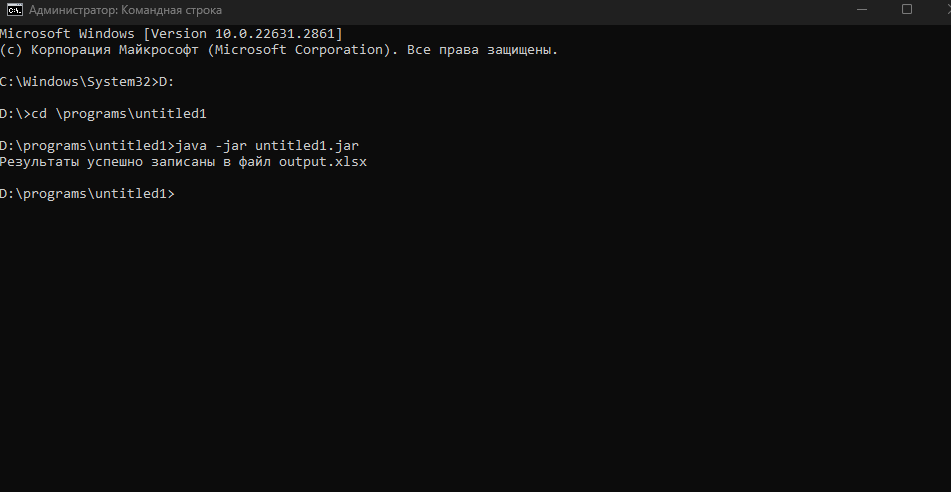


Рис. 2.9.3(Успешный запуск программы)

Можно заметить, что программа успешно выполнена, а если зайти в изменяемый файл Excel, то можно увидеть, что он обновился.

Далее перейдем к созданию UML диаграммы.

## 2.10 Работа с UML.

UML (Unified Modeling Language) – это универсальный язык моделирования, предназначенный для визуализации, проектирования и документирования системных архитектур и процессов разработки программного обеспечения.

UML предоставляет набор графических инструментов для описания различных аспектов системы:

* Диаграммы классов: показывают структуру системы через классы, их атрибуты, методы и отношения между классами.
* Диаграммы вариантов использования: описывают взаимодействие между системой и ее окружением (пользователи, варианты использования и т.д.).
* Диаграммы последовательностей: показывают последовательность взаимодействия между объектами в системе во времени.
* Диаграммы состояний: иллюстрируют различные состояния, в которых может находиться объект или система, и переходы между этими состояниями.
* Диаграммы активностей: позволяют моделировать последовательность действий или процессов в системе.
* Диаграммы компонентов и развертывания: описывают структуру компонентов программной системы и их взаимосвязи на уровне аппаратного обеспечения.
* Умение работать с UML-диаграммами позволяет разработчикам и архитекторам программного обеспечения лучше понимать систему, ее структуру и взаимодействие между компонентами. Это облегчает коммуникацию между членами команды, позволяет предсказывать проблемы в процессе разработки и оптимизировать архитектуру перед началом реализации.

В данном случае будет использоваться диаграмма классов в моей программе. Диаграмма будет построена с помощью иIntelliJ IDEA Ultimate, в ней есть специальный плагин позволяющий работать с UML. Если использовать IntelliJ Idea Ultimate, то у нас будет установлен плагин "UML Support". Он позволяет автоматически создавать диаграммы классов. Например, через «Ctrl+N» или пункт меню "Navigate" -> "Class" перейдём в класс ArrayList. Теперь, через контекстное меню по имени класса выберем "Diagram" -> "Show diagram popup". В результате мы получим диаграмму:

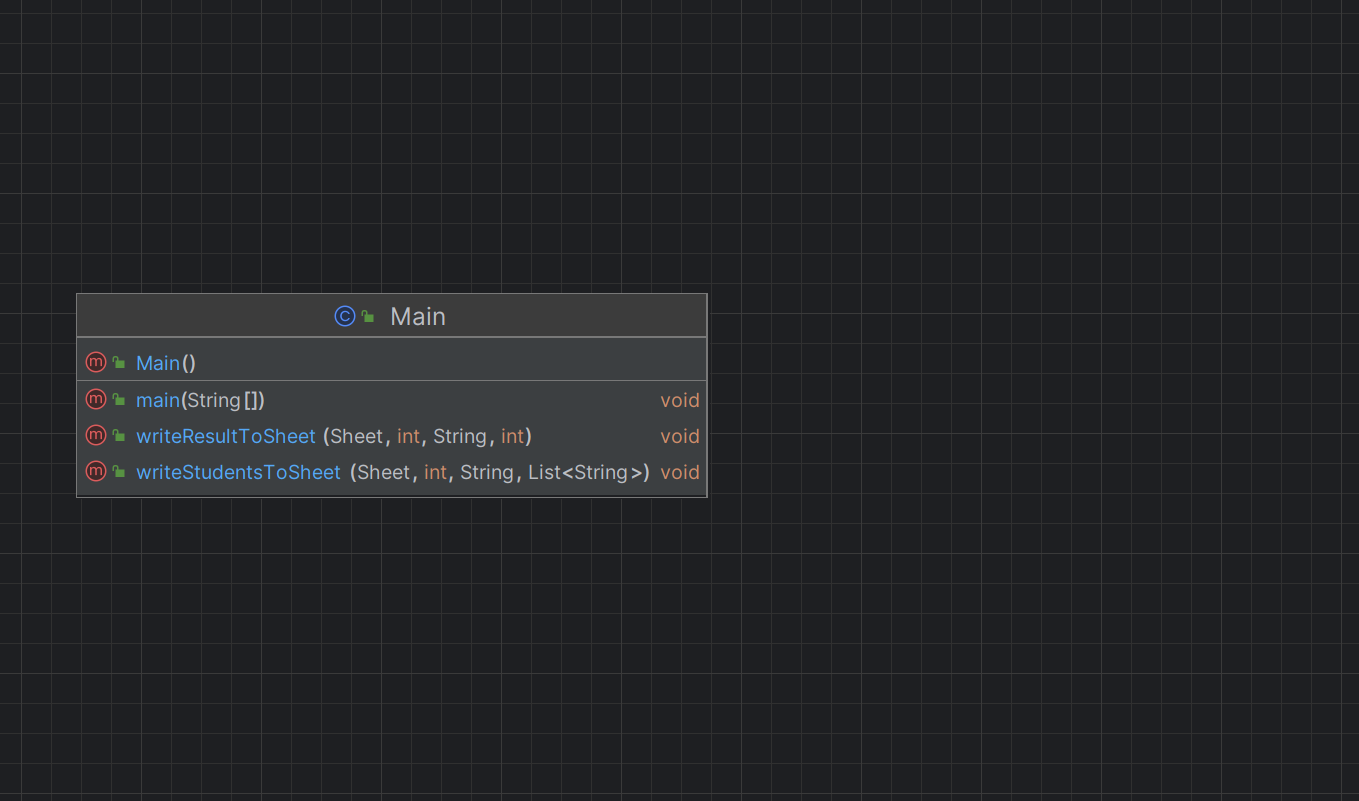


Рис. 2.10.1(UML диаграмма классов)

При большом количестве классов и методов диаграммы становятся очень полезными для понимания работы кода

# Глава 3. GitHub и его система.

## 3.1 GitHub

GitHub — это популярная платформа для хостинга репозиториев Git, предоставляющая широкий набор инструментов для разработки программного обеспечения. Вот некоторые плюсы и минусы GitHub:

Плюсы GitHub:

* Широко используемая платформа: GitHub широко признан и используется в различных открытых и коммерческих проектах, что обеспечивает прозрачность и доступность для коллаборации.
* Открытость и общедоступность: Большинство репозиториев на GitHub являются открытыми, что позволяет широкому кругу людей просматривать код, предлагать изменения и участвовать в проектах.
* Инструменты управления проектами: GitHub предоставляет возможности для создания задач, отслеживания ошибок, обсуждения изменений кода и другие инструменты для управления проектами.
* Коллаборация и форканье проектов: Пользователи могут форкать (создавать копии) проектов и предлагать свои изменения через пулл-реквесты, облегчая процесс сотрудничества и рецензирования кода.
* Интеграция с CI/CD: GitHub интегрируется с различными инструментами Continuous Integration (CI) и Continuous Deployment (CD), что позволяет автоматизировать процессы тестирования и развертывания.

Минусы GitHub:

* Ограничения в планах для хранения приватных репозиториев: Бесплатные аккаунты GitHub имеют ограничения по количеству приватных репозиториев и их размеру, что может быть недостаточно для коммерческих проектов.
* Зависимость от интернет-соединения: для работы с GitHub требуется постоянное интернет-соединение, что может быть неудобно в условиях ограниченной связи.
* Проблемы безопасности: поскольку многие репозитории открыты для просмотра, есть риск утечки конфиденциальной информации, если не управлять доступом к репозиториям должным образом.
* Ограниченные возможности в бесплатном плане: Некоторые продвинутые функции, такие как защищенные ветки и расширенные инструменты управления проектами, доступны только в платных версиях GitHub.
* Зависимость от одного провайдера: Использование одной платформы для хостинга репозиториев может привести к зависимости от GitHub, что в определенных случаях может быть рискованным.
* GitHub имеет множество преимуществ для разработчиков, но важно учитывать и недостатки, чтобы выбрать наиболее подходящую платформу в зависимости от конкретных требований проекта.

Репозиторий — это хранилище данных, которое используется для хранения файлов, кода, документации и другой информации, связанной с проектом. В контексте системы контроля версий, такой как Git или SVN, репозиторий представляет собой основное место, где хранится и управляется код программы или проекта.

Репозиторий может быть публичным (доступным для всех) или приватным (доступным только для определенных пользователей). Он обычно содержит:

* Исходный код: Файлы, которые составляют программу или проект.
* Конфигурационные файлы: например, файлы настроек для развертывания приложения, настройки среды разработки и другие параметры.
* Документация: Описание проекта, инструкции по установке, руководства пользователя и другая документация.
* История изменений: Репозиторий отслеживает изменения, сделанные в коде и других файлах, и сохраняет их историю. Это позволяет возвращаться к предыдущим версиям, сравнивать изменения и отслеживать, кто и когда вносил изменения.

Репозитории могут храниться локально на компьютере разработчика или на удаленном сервере, таком как GitHub, GitLab или Bitbucket, где они доступны для коллаборации и обмена между членами команды. Создание репозитория позволяет упорядочить и структурировать разработку проекта, обеспечивая централизованное управление и контроль за версиями кода и других файлов.

Далее подробно изучим загрузку кода на GitHub

## 3.2 Загрузка кода на GitHub

Пусть есть проект с исходным кодом, состоящим из одного файла untitled jar Загрузим его в репозиторий GitHub.

Переходим на сайт. Регистрируемся, и входим в аккаунт. На главной странице GitHub нажимаем кнопку "New" для создания нового репозитория. На открывшейся странице вводим название репозитория. Название должно отражать суть проекта.

Нажимаем кнопку "Create repository" для завершения создания нового репозитория.

Пустой репозиторий GitHub создан. Теперь загрузим сюда код нашего проекта. Сделать это можно разными способами. Рассмотрим один из них. Затем нажмите на кнопку “Upload files” или “Добавить файл”, которая находится ниже названия репозитория. Выберите файл на вашем компьютере, который нужно добавить, и нажмите на кнопку “Commit changes” или “Сохранить” чтобы сохранить изменения. Теперь файл будет добавлен в репозиторий на GitHub

Не нужно загружать в репозиторий производные файлы проекта. Это те файлы, которые генерируются тем или иным образом из исходного кода. Например, это файлы сборки (вроде Makefile), файлы редактора кода (IDE) и т.п.

## 3.3 Система Git

Git — это распределенная система контроля версий, используемая для управления изменениями в коде и других текстовых файлах в проектах разработки программного обеспечения. Она позволяет отслеживать изменения, возвращаться к предыдущим версиям, объединять изменения от разных разработчиков и управлять разветвлениями кода.

Основные концепции Git:

* Репозиторий: Место, где хранится история изменений. Может быть локальным или удаленным.
* Коммиты: Снимки состояния проекта на определенный момент времени.
* Ветвление и слияние: Создание новых линий разработки (ветвей) и их последующее объединение с основной веткой.
* Индекс (staging area): Промежуточная область, где отмечаются изменения перед фиксацией в коммит.
* Хеширование: Использование хеш-сумм для идентификации коммитов, файлов и изменений.

Основные команды Git:

* «git init»: Инициализация нового репозитория.
* «git clone»: Клонирование существующего репозитория.
* «git add»: Добавление изменений в индекс.
* «git commit»: Создание коммита с сохранением изменений.
* «git push»: Отправка локальных изменений в удаленный репозиторий.
* «git pull»: Получение изменений с удаленного репозитория и их объединение с локальными.
* «git branch»: Управление ветками.
* «git merge»: Объединение веток.
* «git diff»: Просмотр изменений между коммитами или файлами.

Git позволяет эффективно сотрудничать над проектами, отслеживать изменения и управлять версиями кода, что делает его популярным инструментом в разработке ПО.

# 

# Глава 4 Заключение.

Этот проект по работе с библиотекой Apache POI в Java отражает не только умение создания программ, способных читать и анализировать данные из Excel-файлов, но и подчеркивает значимость интеграции различных технологий для решения конкретных задач. Представленная программа не только выводит подробную статистику успеваемости студентов, но и является отличным примером того, как различные инструменты могут быть использованы для обработки и анализа данных.

Данный проект призван показать не только технические навыки работы с Excel в Java, но и внедрение эффективных решений для обработки информации. Особый интерес представляет исследование возможностей Apache POI, выявление его преимуществ, таких как гибкость и удобство в работе с данными Excel, что актуально для разработчиков, стремящихся к оптимизации процессов обработки данных.

Проект также подчеркивает значимость автоматизации анализа данных и создания отчетов, что важно не только для Java-разработчиков, но и для широкого круга специалистов, работающих с данными. Подобные исследования позволяют более глубоко понять возможности интеграции между языками программирования и прикладными программами, открывая новые горизонты в области аналитики данных и автоматизации рабочих процессов.

Язык Java является неотъемлемой частью современной программной индустрии благодаря своей универсальности и широкому спектру применений.

Для начинающих разработчиков Java представляет собой прекрасное введение в мир программирования. Его лаконичный синтаксис и простота понимания делают его доступным для изучения, позволяя новичкам быстро приступить к созданию простых приложений и плавно переходить к более сложным задачам.

Опытные специалисты также ценят Java за ее масштабируемость и надежность. Она позволяет разрабатывать крупные корпоративные системы, обрабатывать огромные объемы данных и эффективно взаимодействовать с другими технологиями.

# Список использованных источников

(Мой проект GitHub) <https://github.com/senkyo0/Kursovaya>

(Работа с POI) <https://www.geeksforgeeks.org/reading-writing-data-excel-file-using-apache-poi/>

(Работа с POI) <https://tproger.ru/translations/how-to-read-write-excel-file-java-poi-example>

(Работа с POI)<https://habr.com/ru/articles/56817/>

(Описание POI на «ходу») <https://yandex.ru/video/preview/15552245767471183086>

(Работа с POI) <https://javacodepoint.com/apache-poi-getting-started/>

(Установка библиотеки) <https://qaautomation.expert/2023/11/28/how-to-download-install-apache-poi/>

(Официальный сайт Apache POI) <https://poi.apache.org/>

(Работа с Maven)<https://javarush.com/groups/posts/2523-chastjh-4osnovih-maven>

(База Maven) <https://mvnrepository.com/>

(Подключение библиотек) <https://gb.ru/blog/podklyuchenie-bibliotek-v-java/>

(Работа с Maven) <https://habr.com/ru/articles/311108/>

(Теория по JFreeChart) <https://java-online.ru/jfreechart.xhtml>

(Работа с JFReeChart) <https://ninavasilok.ru/rabota-s-bibliotekoi-jfreechart-dlya-sozdaniya-grafikov-na-java/>

(Теория по JFreeChart)<https://russianblogs.com/article/706539408/>

(Сайт с IntelliJ) <https://www.jetbrains.com/ru-ru/idea/>

(Теория по IntelliJ) <https://javarush.com/groups/posts/intellij-idea>

(Руководство по пакетам) <https://habr.com/ru/articles/755654/>

(Подключение библиотек) <https://yandex.ru/video/preview/2775787676703238424>

(Теория про библиотеки и пакеты) <http://axis.bplaced.net/news/762>

(Подключение библиотек) <https://yandex.ru/video/preview/1612521600021535530>

(Сайт с Java) <https://www.oracle.com/java/>

(Официальный сайт GitHub) <https://github.com/>

(Теория по GitHub) <https://habr.com/ru/companies/yandex_praktikum/articles/700708/>

(Теория по GitHub) <https://skillbox.ru/media/code/chto-takoe-github-i-kak-im-polzovatsya/>

(Работа с Jar) <https://uchet-jkh.ru/i/kak-otkryt-jar-fail-s-pomoshhyu-java/>

(Запуск Jar из консоли) <https://stackoverflow.com/questions/5774970/run-jar-file-in-command-prompt>

(Jar файл)<https://studio-sr.ru/sozdanie-jar-faila-v-intellij-idea/>

<https://www.youtube.com/watch?v=3Xo6zSBgdgk&abchannel=ArturSpirin>