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 **«Расчёт начисления заработной платы за дни отпуска»**

|  |  |
| --- | --- |
| Выполнил: | студент группы 10702221  Барковец Р. С. |
| Руководитель: | доц. Сидорик В.В. |

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту (работе)

по дисциплине «Программирование на языке Java»

на тему **«Расчёт начисления заработной платы за дни отпуска»**

|  |  |
| --- | --- |
| Выполнил: | студент группы 10702221  Барковец Р. С. |
| Руководитель: | доц. Сидорик В.В. |

Минск 2023

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc152586914)

[1. Постановка и описание задачи 5](#_Toc152586915)

[2. Теоретический раздел 6](#_Toc152586916)

[2.1 Язык программирования Java 6](#_Toc152586917)

[2.2 Библиотека Swing 8](#_Toc152586918)

[2.3 Расчёт отпускных выплат для РБ в 2023 г. 10](#_Toc152586919)

[3. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ 11](#_Toc152586920)

[4. Практический раздел 12](#_Toc152586921)

[4.1 Построение UML диаграммы классов 12](#_Toc152586922)

[4.2 Создание окон приложения 13](#_Toc152586923)

[4.3 Скриншоты с ошибками 19](#_Toc152586924)

[5. Результаты работы 21](#_Toc152586925)

[5. Описание структуры проекта 22](#_Toc152586926)

[Список использованной литературы 23](#_Toc152586927)

[Приложение А 24](#_Toc152586928)

# ВВЕДЕНИЕ

Почти каждый работающий человек рано или поздно задумывается об отпуске. И никто не упускает возможность его использовать, особенно желанный всеми ежегодный. Ведь за то, что человек отдыхает, ему еще и платят!

Но законодательство любой страны начисляет отпускные выплаты не так, как думают от этом люди. Нужно учесть достаточно много факторов, чтобы правильно рассчитать выплаты.

Но не всем хочется идти в бухгалтерию и спрашивать, что и как. На сегодняшний день гораздо проще найти программу, которая автоматически по заданным параметрам рассчитает всё что надо.

И в своём проекте я также создал похожий калькулятор, чтобы удобно рассчитывать отпускные выплаты людям.

# Постановка и описание задачи

**Постановка задачи**: разработать приложение для расчёта начисления заработной платы за дни отпуска. Сформировать документацию проекта с помощью утилиты javadoc и предоставить её в виде html-файлов. Приложение собрать в файл расширения .jar, исполняемый на различных устройствах.

**Описании задачи:** пустьимеется некий сотрудник, уходящий в отпуск, и желающий узнать, сколько же он получит перед уходом. В бухгалтерии сказали, что выдадут ему отпускной отчёт со всей информацией. В личном деле сотрудника была найдена вся необходимая информация.

**Исходные данные (информация из бухгалтерии)**: фамилия, имя, отчество, количество несовершеннолетних и очно обучающихся совершеннолетний, заработная плата за месяцы работы.

**Цель**: помочь бухгалтерии и разработать приложение для автоматизации их работы. Рассчитать отпускные выплаты уходящему в отпуск сотруднику. Создать отчёт с информацией.

# Теоретический раздел

## Язык программирования Java

На сегодняшний день Java является одним из самых популярных языков программирования. Со временем стало понятно, что он является в некотором роде революционным языком.

Современные приложения должны быть безопасны, высокопроизводительны, работать в распределенной среде, быть нейтральны к архитектуре. Все эти факторы привели к необходимости нового взгляда на сам процесс создания и распределения приложений на множестве машин различной архитектуры. Требования к переносимости заставили отказаться от традиционного способа создания и доставки бинарных файлов, содержащих машинные коды и, следовательно, привязанных к определенной платформе. Созданная компанией Sun Microsystems система разработки Java удовлетворяет всем этим требованиям. Java – объектно-ориентированный язык, удобный и надёжный в эксплуатации благодаря таким своим достоинствам, как многозадачность, поддержка протоколов Internet и многоплатформенность. Java – это интерпретируемый язык, и каждая Java-программа компилируется для гипотетической машины, называемой Виртуальная Машина Java. Результатом такой компиляции является байт-код Java, который в свою очередь может выполняться на любой операционной системе при условии наличия там системы времени выполнения Java, которая интерпретирует байт-код в реальный машинный код конкретной системы.

Однако, такая универсальность данной технологии рождает недостаток – требовательность к ресурсам компьютера. Так как Java-программы не содержат машинного кода и при их запуске включается в работу система времени выполнения Java, их производительность заметно ниже, чем у обычных программ, составленных, например, на языке программирования C++. Данный недостаток становится с течением времени всё менее ощутим, в следствии роста вычислительной мощности компьютерных систем.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Библиотеки классов Java значительно упрощают разработку приложений, предоставляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач, а не таких, как, например, организация динамических массивов, взаимодействие с операционной системой или реализация элементов пользовательского интерфейса.

В сферах применяют Java? Да во всех. Это и веб-разработка, и мобильные приложения на Android, где в свое время было очень интересно разрабатывать игры, и десктоп приложения, и гейм-дев сфера, где на этом языке написано много библиотек и непосредственно игр на чистом языке.

## Библиотека Swing

Java Swing - это набор библиотек и классов, предоставляемых Java для создания графического пользовательского интерфейса (GUI). Он предоставляет разработчикам широкие возможности для создания интерактивных и эстетически привлекательных приложений.

Swing является частью Java Foundation Classes (JFC) и предоставляет компоненты, такие как кнопки, текстовые поля, метки, таблицы и многое другое для создания пользовательского интерфейса. Он также поддерживает различные компоненты для создания меню, диалоговых окон и панелей инструментов.

Одной из главных особенностей Java Swing является его платформенная независимость. Это означает, что приложения, созданные с использованием Swing, могут работать на разных операционных системах, таких как Windows, macOS и Linux, без необходимости внесения изменений в код.

Swing также обладает мощными возможностями для настройки внешнего вида и поведения компонентов. Разработчики могут управлять цветами, шрифтами, размерами и другими атрибутами компонентов, чтобы создать уникальный и согласованный дизайн пользовательского интерфейса.

В Swing также имеется поддержка событий и обработчиков, что позволяет реагировать на действия пользователя, такие как нажатие кнопок или ввод текста. Это позволяет создавать интерактивные приложения, которые могут взаимодействовать с пользователем.

Java Swing также предоставляет возможности для создания сложных компонентов, таких как деревья, вкладки и панели переключателей, которые облегчают организацию и навигацию по большим объемам данных.

Несмотря на то, что Java Swing является старым технологическим решением, оно до сих пор активно используется разработчиками для создания приложений с графическим интерфейсом. Оно предоставляет надежную и гибкую платформу для разработки приложений, которые могут быть запущены на разных платформах.

В заключение, Java Swing — это мощный инструмент для создания графического пользовательского интерфейса в Java. Он предоставляет разработчикам широкие возможности для создания интерактивных и эстетически привлекательных приложений, которые могут работать на разных операционных системах.

## Расчёт отпускных выплат для РБ в 2023 г.

Любой работник может самостоятельно рассчитать отпускные, зная алгоритм. Для точного определения суммы требуются 2 показателя:

Среднее значение заработной платы сотрудника за весь рабочий год. Обязательно учитывают нюансы – введенные надбавки, полученные работником премиальные.

Пример, как правильно рассчитать отпускные работнику в Беларуси:

Сотрудник компании отработал должный промежуток, равный 12 календарным месяцам. В соответствии с законом человек имеет право на предоставление отпуска продолжительностью 24 дня.

Для точного подсчета необходимо выяснить размер дневного заработка. Чтобы его узнать, нужно разделить среднюю величину заработной платы на среднестатистическое количество рабочих дней в месяце.

Следующий этап – умножение числа дней отпуска на показатель среднего дневного заработка.

Как начисляются отпускные, когда срок работы составляет меньше календарного года:

В первую очередь необходимо определить число отпускных дней. Работнику положен отпуск в 24 дня при отработке полного года. Если человек работал 190 дней, рассчитывают продолжительность заработанного отдыха по формуле: 190:29,6.

Получается, работник трудился 6 месяцев и 12,4 дня. Остаток меньше 15, поэтому при расчете его игнорируют, к исчислению принимают только срок в 6 месяцев. Это предусматривает оплачиваемый отпуск продолжительностью 12 дней.

Далее применяется схема расчета с определением среднестатистического дневного заработка.

# МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Из предыдущего пункта мы узнали, как рассчитывать отпускные на словах. А теперь нужно представить это в виде формул.

Для начала нужно высчитать средний дневной заработок. Формула:

,

И тогда формула для расчёта отпускных:

,

Но есть нюансы. Считать нужно не просто по заработной плате, но еще и использовать поправочные коэффициенты, которые слегка коррелируют заработную плату каждый месяц. Формула:

,

И умножаем текущую месячную з. п. на этот коэффициент.

Но ведь отпускные – это та же самая заработная плата. И она тоже облагается налогами. Тогда «на руки» выдается сумма, рассчитанная по следующей формуле:

*,*

где z = 1, если отпускные меньше 944 рублей, а x – кол-во детей, попадающих в категории либо несовершеннолетние, либо совершеннолетние обучающиеся очно.

В добавок идут по 1% для профсоюза (если работник состоит в нем) и 1% для пенсионного взноса.

Итог:

.

Таким образом рассчитываются отпускные.

Блок-схему алгоритма можно найти в приложении Б

# Практический раздел

## 4.1 Построение UML диаграммы классов

Унифицированный язык моделирования (UML) – это язык для специфицирования, визуализации, конструирования и документирования программных систем, а также бизнес-моделей и прочих не программных систем. При решении сложных задач заблаговременное планирование и моделирование значительно упрощают программирование.

Далее представлены UML-диаграммы проекта по частям.

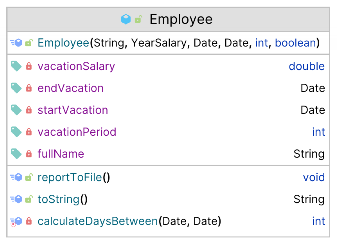
****

Рисунок 1 – UML-диаграмма основных классов (годовая зарплата и рабочий)

## 4.2 Создание окон приложения

После запуска программы перед пользователем открывается стартовое окно. В стартовом окне отображена общая информация о месте и человеке, причастных к созданию программы. Пользователь может выйти из программы, нажав кнопку «Выйти» или продолжить работу, нажав кнопку «Далее» (рис. 2). При бездействии пользователя в течение 60 секунд, программа автоматически закрывается.

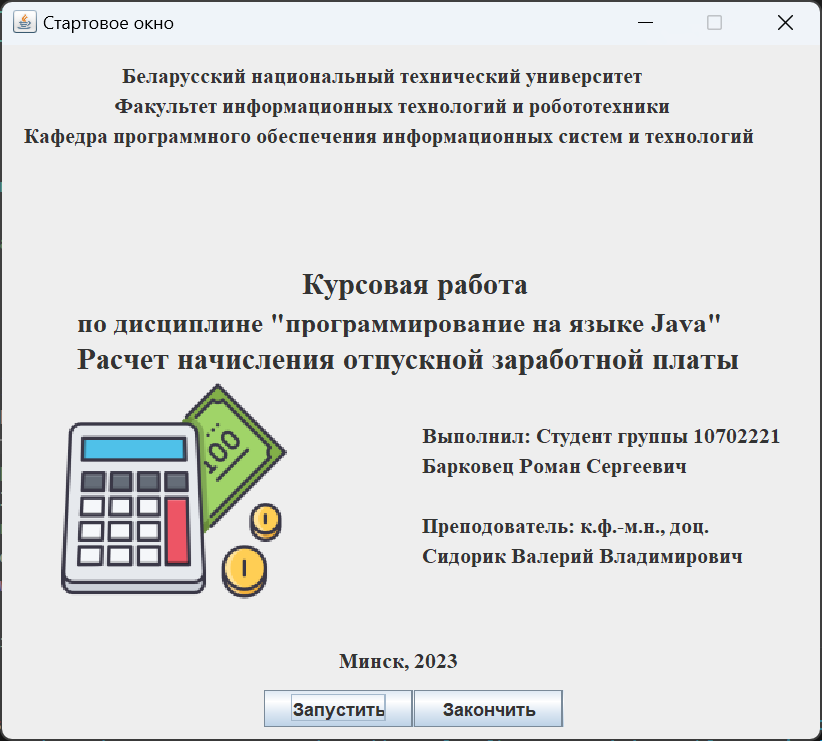


Рисунок 2 – Изображение стартового окна

При нажатии на кнопку «Запустить» в основном окне перед пользователем открывается рабочее окно, содержащее различные поля для ввода данных

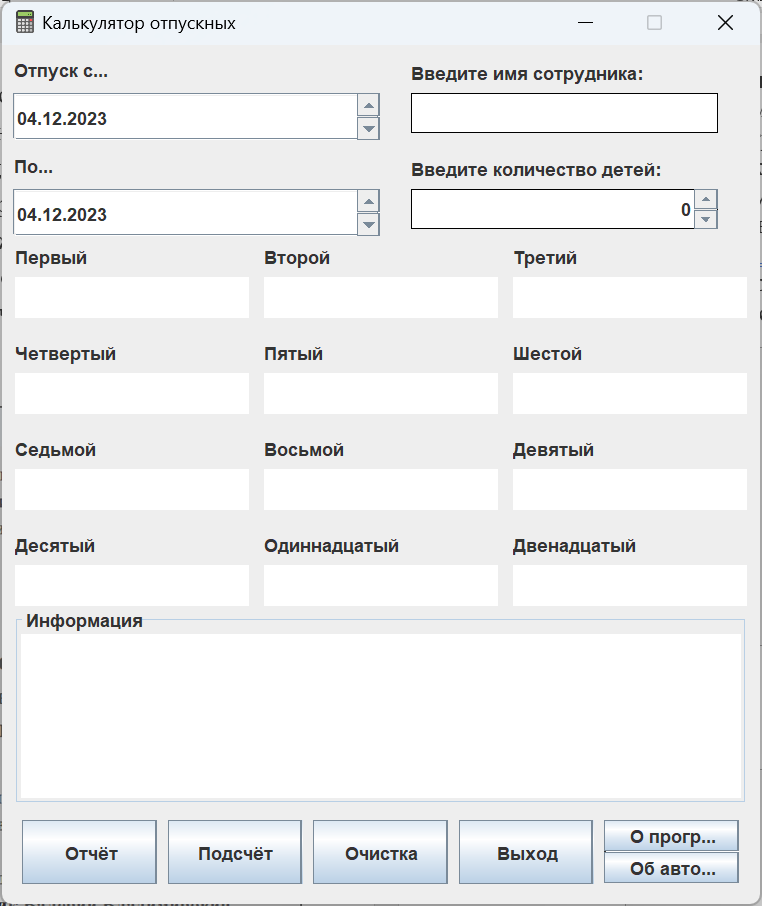


Рисунок 3 – Рабочая область

При правильном вводе данных и нажатии кнопки «Подсчёт» программа воспользуется введенными данными и выведет в окно «Информация» краткие данные о человеке.

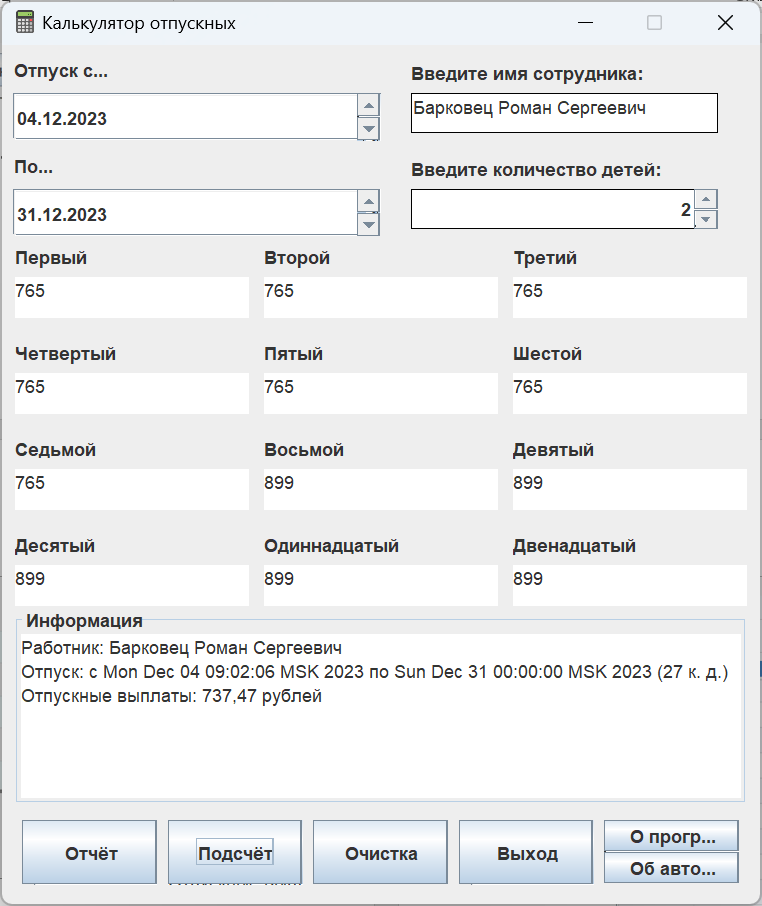


Рисунок 4 – Рабочая область с данными

Если же пользователь вдруг захотел распечатать отчёт, то ему нужно его где-то сохранить. А еще раньше – создать, ведь без предварительных вычисление программа не будет работать.

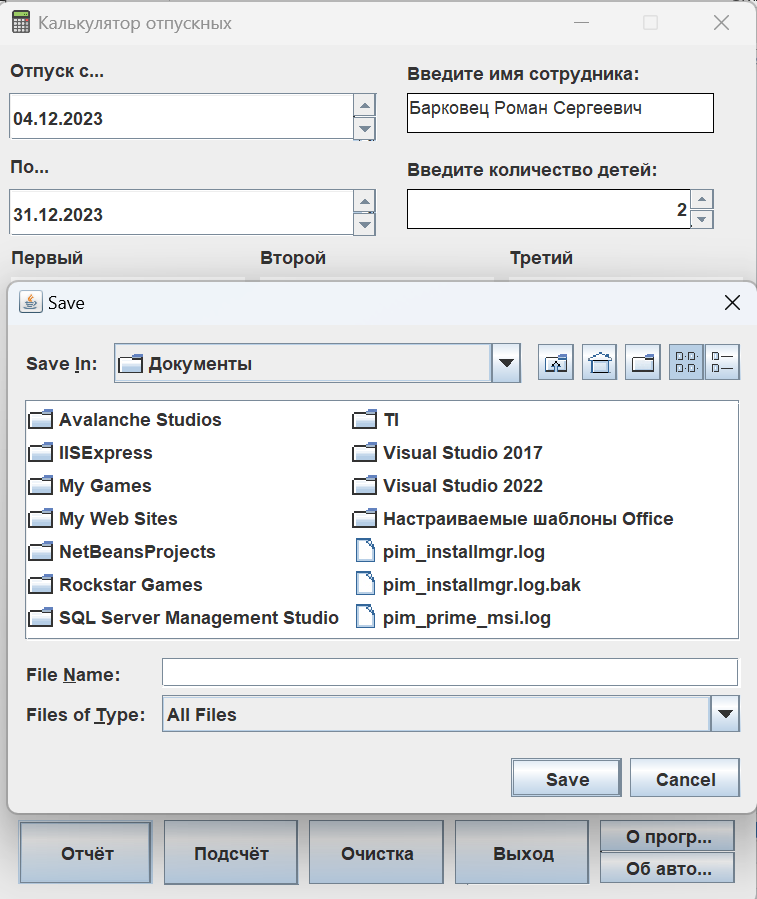
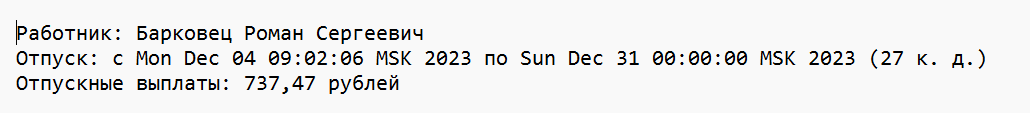


Рисунок 5 – Выбор местоположения будущего отчёта

И как это выглядит в файле формата .txt:

Рисунок 6 – Отчет в текстовом файле

Когда пользователь захочет взять нового человека, то он захочет быстро удалить старую информацию. Для этого есть кнопка «Очистка», которая удалит всю информацию из доступных полей, кроме даты:

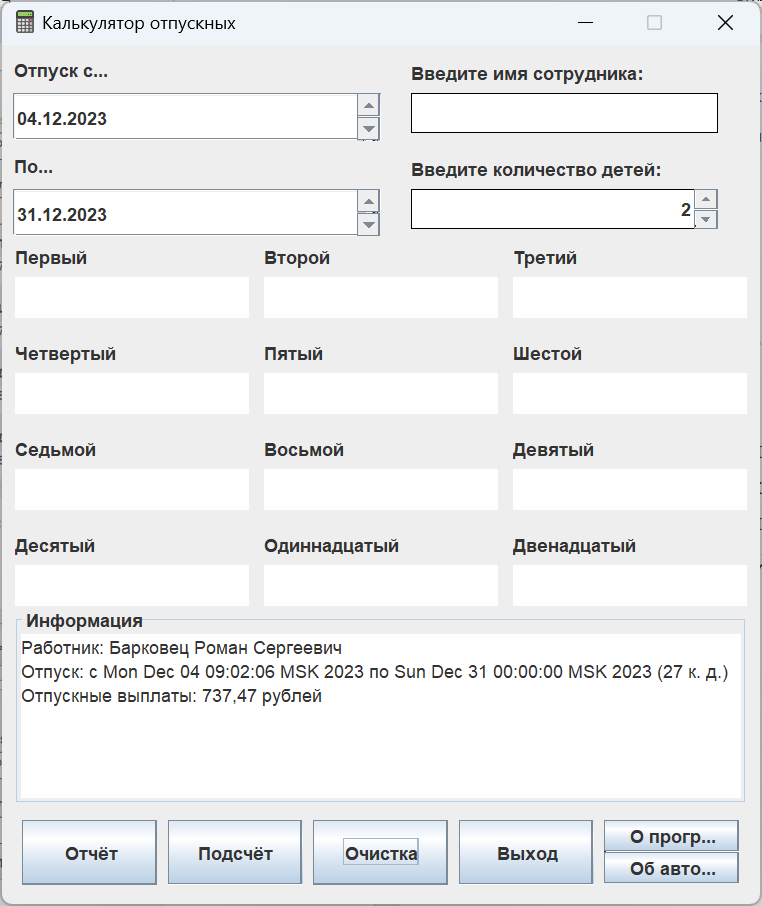


Рисунок 7 – Очистка полей

Окно «Об авторе» содержит основную информацию об авторе проекта: номер группы, имя, фамилию, отчество, фотографию автора, а также его электронную почту.

Это окно не перекрывает доступ к основному окну:

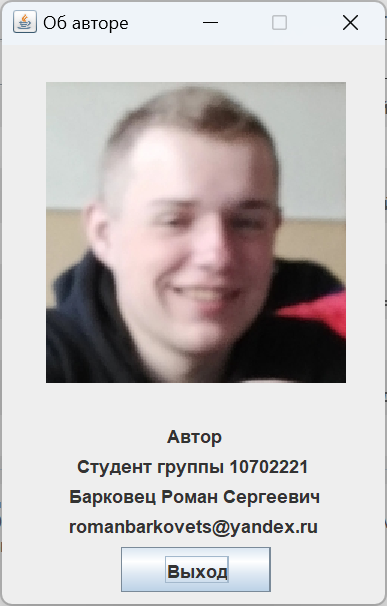


Рисунок 8 – Окно «Об авторе»

Окно «О программе» содержит основную информацию о возможностях программы и версию приложения.



Рисунок 9 – Изображение окна «О программе»

## 4.3 Скриншоты с ошибками

Если пользователь ввел строку, которая не может конвертироваться в число, то в информационном окне появляется информация об ошибке.

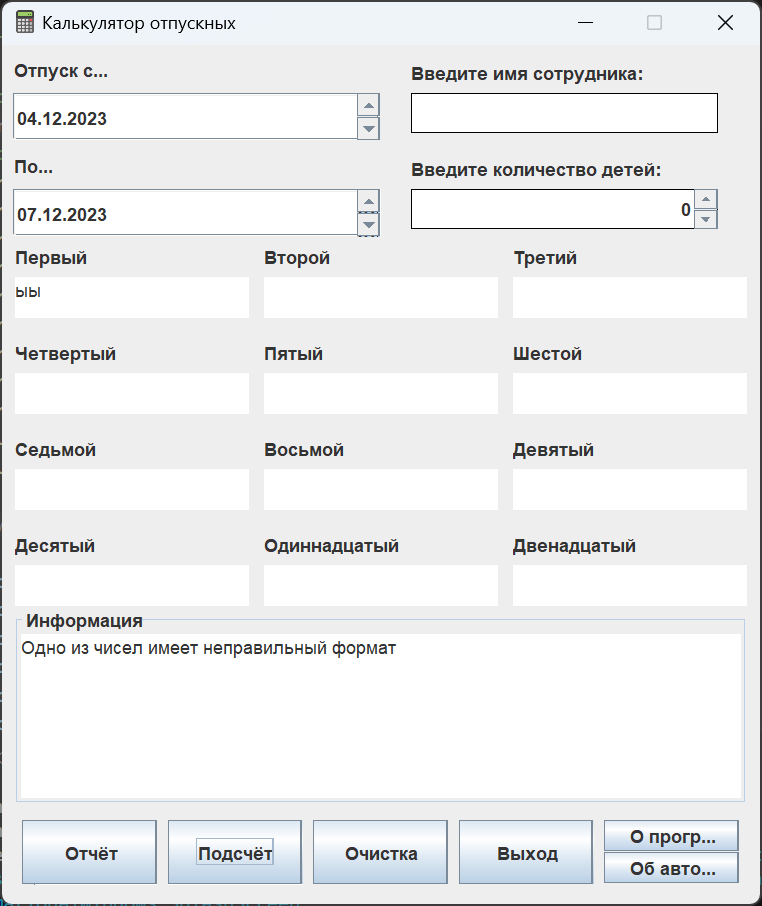


Рисунок 10 – Неправильный формат числа

Если пользователь ввел дату начала, которая больше даты конца, то в информационном окне появляется информация об ошибке.

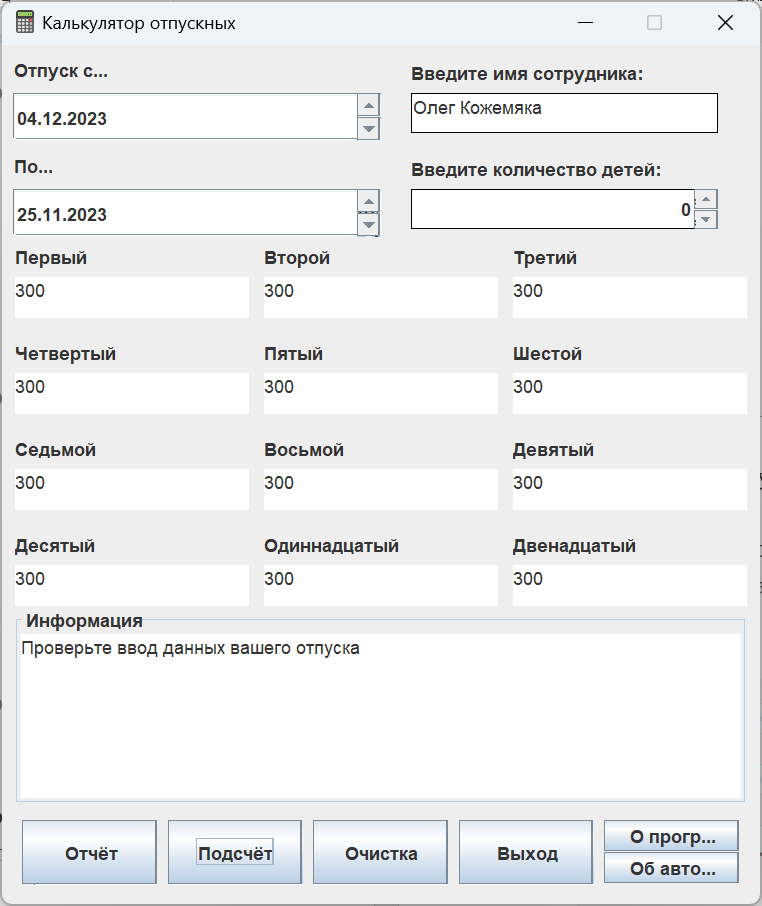


Рисунок 11 – Неправильно введены даты отпуска

# 5. Результаты работы

После правильного ввода данных необходимо нажать на кнопку «Подсчёт» и в соответствующее поле выведется информация.

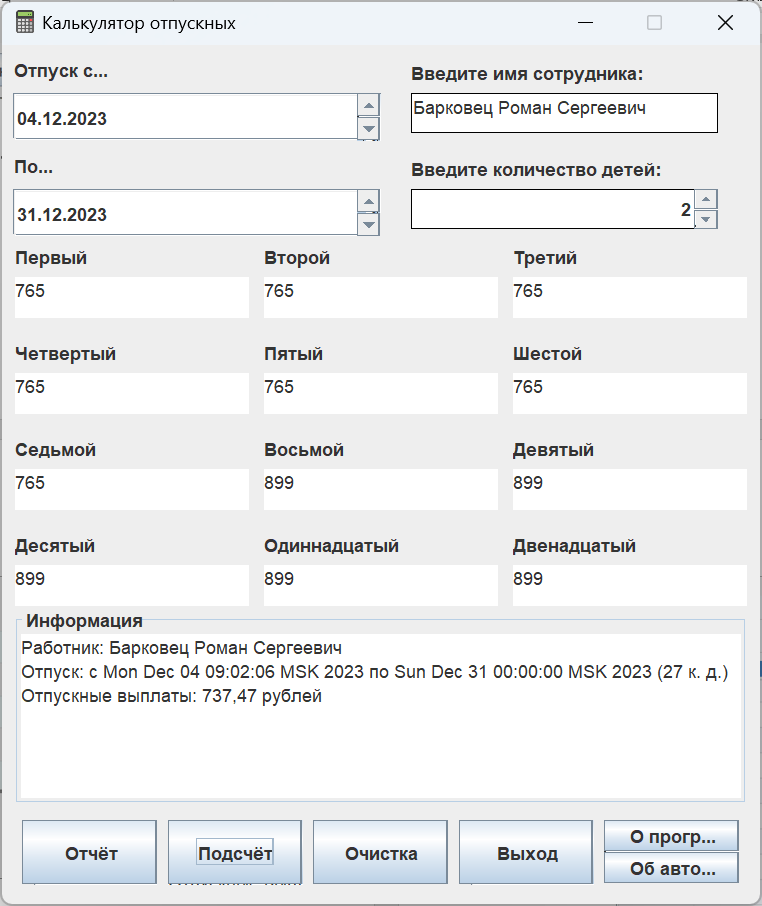


Рисунок 12 – Корректный ввод и вывод

Потом можно нажать кнопку распечатать и расчёты запишутся в файл формата .txt.

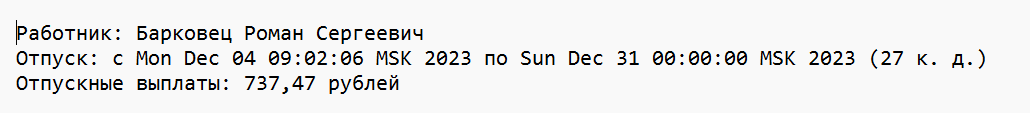


Рисунок 13 – Содержимое файла после записи

# Описание структуры проекта

Структура проекта была спроектирована в соответствии с требованиями к курсовому проекту.

В корневом каталоге app находятся:

1. zip – папка, в которой находится весь проект, упакованный в один архив zip;
2. doc – папка, в которой находится пояснительная записка в формате docx;
3. src – папка со всеми классами проекта;
4. javadoc – папка со сгенерированным JavaDoc документом.
5. dist – папка с jar-файлом проекта.
6. lib – папка с необходимыми библиотеками для реализации логики приложения (для работы с SQLite, Word, Excel).

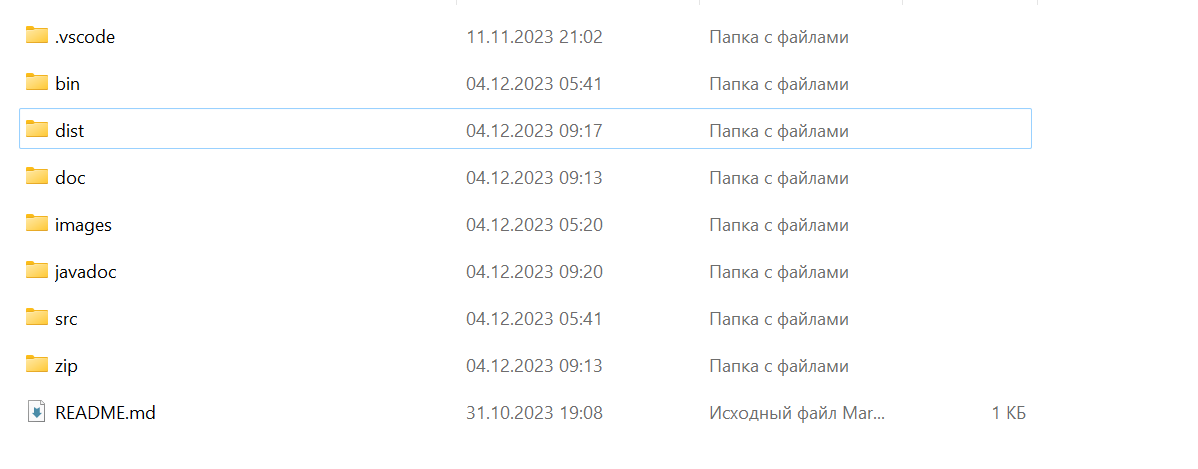


Рисунок 14 – Структура проекта

# Список использованной литературы

1. Horstmann, Cay S., 1959–Big Java : compatible with Java 5, 6 and 7 / Cay Horstmann. - 4th ed [Электронный ресурс] – Режим доступа: <https://pdfroom.com/books/big-java/qlgyyYDPgMG>
2. Блох, Джошуа Б70 Java: эффективное программирование, 3-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2019. [Электронный источник] – Режим доступа: <https://coderbooks.ru/books/java/java_ehffektivnoe_programmirovanie_3_izd_bloh_2019/>
3. Гуськова, Ольга Ивановна. Г968 Объектно ориентированное программирование в Java : учебное пособие / О. И. Гуськова. – Москва : МПГУ, 2018. [Электронный источник] – Режим доступа: <https://coderbooks.ru/books/java/obektno-orientirovannoe-programmirovanie-v-java/>
4. Шилдт, Герберт. Java. Полное руководство, 12-е изд. : Пер. с англ. - СПб. “Диалектика”; 2023. - 1344 с.: ил. - Парал. тит. Англ [Электронный ресурс] – Режим доступа: <https://coderbooks.ru/books/java/java_polnoe_rukovodstvo_12_izd_schildt_2023/>

# Приложение А

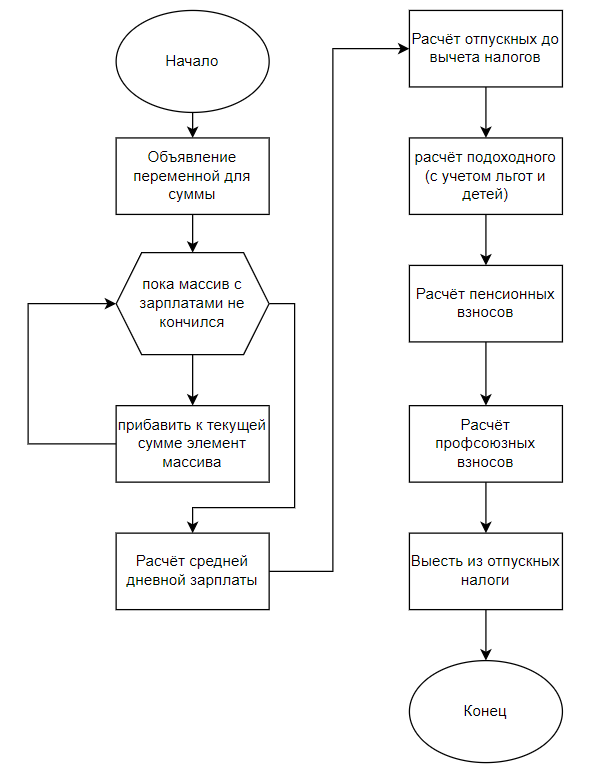
**

Рисунок 15 – Блок-схема расчета отпускных выплат

**ПРИЛОЖЕНИЕ Б**

Файл InputCheckerTool.java

package features;

import java.util.ArrayList;

import java.util.Date;

import windows.SecondaryWindows.DatePanel;

/\*\*

 \* Класс, содержащий инструменты для проверки входящих данных

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class InputCheckerTool {

    /\*\*

     \* проверяет список на наличие пустых элементов

     \*

     \* @param list - массив для проверки

     \* @return true, если в списке есть пустые элементы, иначе false

     \*/

    public static boolean isAnyEmpty(ArrayList<String> list) {

        for (int i = 0; i < list.size(); i++) {

            if (list.get(i).isEmpty())

                return true;

        }

        return false;

    }

    /\*\*

     \* Проверяет, являются ли элементы списка числами типа double или пустыми

     \* значениями.

     \*

     \* @param list список для проверки

     \* @return true, если все элементы списка являются числами типа double или

     \*         пустыми значениями, иначе false

     \*/

    public static boolean isDoubleOrEmpty(ArrayList<String> list) {

        for (int i = 0; i < list.size(); i++) {

            if (list.get(i).isEmpty() || list.get(i).isBlank() || list.get(i) == null) {

                continue;

            }

            if (!tryParseDouble(list.get(i))) {

                return false;

            }

            if (Double.parseDouble(list.get(i)) < 0) {

                return false;

            }

        }

        return true;

    }

    /\*\*

     \* Пытается преобразовать строку в число типа double.

     \*

     \* @param value строка для преобразования

     \* @return true, если строка может быть преобразована в число типа double, иначе

     \*         false

     \*/

    public static boolean tryParseDouble(String value) {

        try {

            Double.parseDouble(value);

            return true;

        } catch (NumberFormatException e) {

            return false;

        }

    }

    /\*\*

     \* Проверяет корректность введенной даты.

     \*

     \* @param panelFrom панель для выбора начальной даты

     \* @param panelTo   панель для выбора конечной даты

     \* @return true, если введенная дата корректна, иначе false

     \*/

    public static boolean isCorrectDateInput(DatePanel panelFrom, DatePanel panelTo) {

        Date dateFrom = (Date) panelFrom.getSpinner().getValue();

        Date dateTo = (Date) panelTo.getSpinner().getValue();

        if (dateFrom == null || dateTo == null) {

            return false;

        }

        if (dateFrom.toString().isEmpty() || dateTo.toString().isEmpty()) {

            return false;

        }

        if (dateFrom.compareTo(dateTo) > -1) {

            return false;

        }

        return true;

    }

}

Файл CalculateButtonListener.java

package listeners;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Date;

import features.InputCheckerTool;

import models.Employee;

import models.YearSalary;

import windows.MainWindows.InformationPanel;

import windows.MainWindows.PeriodPanel;

import windows.MainWindows.YearSalaryPanel;

import windows.SecondaryWindows.DatePanel;

import windows.SecondaryWindows.FullNameInputPanel;

import windows.SecondaryWindows.KidsCounter;

/\*\*

 \* Слушатель для кнопки "Рассчитать".

 \* Проверяет все значение на возможность конвертации их в тип тип Double

 \* Проверяет правильность введенной даты отпуска

 \* Если все правильно, то возвращает результат

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class CalculateButtonListener implements ActionListener {

    /\*\*

     \* Информационная панель, на которую выводится отчет либо предупреждение об

     \* ошибке

     \*/

    public static InformationPanel infoPanel;

    /\*\*

     \* Панель, содержащая ячейки для ввода зароботной платы

     \*/

    public static YearSalaryPanel ysPanel;

    /\*\*

     \* Панель содержащая поля для ввода даты

     \*/

    public static PeriodPanel periodPanel;

    /\*\*

     \* Панель содержащая поля для ввода имени

     \*/

    public static FullNameInputPanel nameInputPanel;

    /\*\*

     \* Для ввода количества детей

     \*/

    public static KidsCounter kidsCounter;

    /\*\*

     \* Панель для выбора даты начала отпуска

     \*/

    public static DatePanel datePanelFrom;

    /\*\*

     \* Панель для ввода даты конца отпуска

     \*/

    public static DatePanel datePanelTo;

    /\*\*

     \* Обработчик события нажатия на кнопку.

     \* Проверяет входящие данные и выводит результат проверки

     \* Успешная проверка выводит отчет о работнике

     \* @param e событие ActionEvent

     \*/

    @Override

    public void actionPerformed(ActionEvent e) {

        ArrayList<String> strSalaries = ysPanel.getMonthSalaries();

        if (InputCheckerTool.isDoubleOrEmpty(strSalaries)) {

            ArrayList<Double> salaries = new ArrayList<Double>(strSalaries.size());

            for (int i = 0; i < strSalaries.size(); i++) {

                try {

                    salaries.add(Double.parseDouble(strSalaries.get(i)));

                } catch (NumberFormatException ex) {

                    ysPanel.setMonthSalary(i, "");

                    salaries.add(.00000000000000001);

                }

            }

            if (InputCheckerTool.isCorrectDateInput(datePanelFrom, datePanelTo)) {

                Employee emp = new Employee(nameInputPanel.getName(), new YearSalary(salaries),

                        (Date) datePanelFrom.getSpinner().getValue(), (Date) datePanelTo.getSpinner().getValue(),

                        kidsCounter.getKidsCount(),

                        false);

                infoPanel.clearAllInfo();

                infoPanel.addInfo(emp.toString());

            } else {

                infoPanel.clearAllInfo();

                infoPanel.addInfo("Проверьте ввод данных вашего отпуска");

            }

        } else {

            infoPanel.clearAllInfo();

            infoPanel.addInfo("Одно из чисел имеет неправильный формат");

        }

    }

}

ФайлClearButtonListener.java

package listeners;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import windows.MainWindows.YearSalaryPanel;

import windows.SecondaryWindows.FullNameInputPanel;

/\*\*

 \* Класс-слушатель для кнопки "Очистить"

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class ClearButtonListener implements ActionListener {

    /\*\*

     \* панель содержащая ячейки зарплаты

     \*/

    public static YearSalaryPanel ysPanel;

    /\*\*

     \* Панель для ввода имени

     \*/

    public static FullNameInputPanel fniPanel;

    /\*\*

     \* Обрабатывает нажатие на кнопку.

     \* Очищает поле работника и ячейки зарплаты

     \*/

    @Override

    public void actionPerformed(ActionEvent e) {

        for (int i = 0; i < ysPanel.getMonthSalaries().size(); i++) {

            ysPanel.setMonthSalary(i, "");

        }

        fniPanel.setName();

    }

}

Файл SaveButton.java

package listeners;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

import models.Employee;

/\*\*

 \* Класс, содержащий инструменты для сохранения данных в файл формата .txt

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class SaveButtonListener implements ActionListener {

    /\*\*

     \* Объект класса Employee, требующийся для создания отчёта для записи в файл

     \*/

    public static Employee emp;

    /\*\*

     \* Вызывает метод объекта класса Employee, который записывает информацию о себе в файл

     \* Не записывает, если предварительно не создать объект и информацию о немЫ

     \* @param ActionEvent представляет собой объект, вызвавший событие

     \*/

    @Override

    public void actionPerformed(ActionEvent e) {

        try {

            emp.reportToFile();

        } catch (NullPointerException ex) {

            JOptionPane.showMessageDialog(null, "Для начала необходимо посчитать отпускные", "Внимание",

                    JOptionPane.INFORMATION\_MESSAGE);

        }

    }

}

Файл Employee.java

package models;

import java.util.Date;

import javax.swing.JFileChooser;

import listeners.SaveButtonListener;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDate;

import java.time.ZoneId;

/\*\*

 \* Класс представляющий работника для которого нуно рассчитать отпускные выплаты

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class Employee {

    /\*\*

     \* ФИО работника

     \*/

    private String fullName;

    /\*\*

     \* Начала отпуска

     \*/

    private Date startVacation;

    /\*\*

     \* Конец отпуска

     \*/

    private Date endVacation;

    /\*\*

     \* Отпускные выплаты

     \*/

    private double vacationSalary;

    /\*\*

     \* Количество отпускных дней

     \*/

    private int vacationPeriod;

    /\*\*

     \* Конструктор класса, который генерирует полную информацию об объекте

     \* Инициализирует поле слушателя кнопки "Сохранить" для сохранения информации об объекте

     \* @param fullName - ФИО работника

     \* @param yearSalary - ArrayList типа Double, содержащий зарплату за каждый месяц

     \* @param start - Начало отпуска

     \* @param end - Конец отпуска

     \* @param kidsCount - Количество детей работника

     \* @param isInTradeUnion - Состоит ли работник в профсоюзе

     \*/

    public Employee(String fullName, YearSalary yearSalary, Date start, Date end, int kidsCount,

            boolean isInTradeUnion) {

        this.fullName = fullName;

        this.startVacation = start;

        this.endVacation = end;

        this.vacationPeriod = calculateDaysBetween(startVacation, endVacation);

        this.vacationSalary = yearSalary.calculateVacationSalary(kidsCount, isInTradeUnion, vacationPeriod);

        SaveButtonListener.emp = this;

    }

    /\*\*

     \* Рассчитывает количество дней между двумя датами

     \* @param date1 - Ранняя дата

     \* @param date2 - Поздняя дата

     \* @return количество дней между @param date1 и @param date2

     \*/

    private static int calculateDaysBetween(Date date1, Date date2) {

        LocalDate localDate1 = date1.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

        LocalDate localDate2 = date2.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

        return (int) java.time.temporal.ChronoUnit.DAYS.between(localDate1, localDate2);

    }

    /\*\*

     \* Вызывает JFileChooser и записывает в выбранную директорию инормацию о работнике

     \*/

    public void reportToFile() {

        JFileChooser fileChooser = new JFileChooser();

        int userSelection = fileChooser.showSaveDialog(null);

        if (userSelection == JFileChooser.APPROVE\_OPTION) {

            File fileToSave = fileChooser.getSelectedFile();

            try {

                FileWriter writer = new FileWriter(fileToSave + ".txt", false);

                writer.write(this.toString());

                writer.close();

                System.out.println("Текст успешно записан в файл " + fileToSave.getName());

            } catch (IOException e) {

                System.out

                        .println("Произошла ошибка при записи в файл " + fileToSave.getName() + ": " + e.getMessage());

            }

        }

    }

    /\*\*

     \* Перегруженный метод, который возвращает информацию об объекте

     \*

     \* @return ФИО, период отпуска и кол-во дней отпуска, отпускные выплаты для работника

     \*/

    @Override

    public String toString() {

        return "Работник: " + this.fullName + "\n" +

                "Отпуск: с " + this.startVacation + " по " + this.endVacation + " (" + vacationPeriod + " к. д.)" + "\n"

                +

                "Отпускные выплаты: " + String.format("%.2f", this.vacationSalary) + " рублей";

    }

}

Файл YearSalary.java

package models;

import java.util.ArrayList;

/\*\*

 \* Класс представляющий годовую зарплату

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class YearSalary {

    /\*\*

     \* Зарплаты за каждый месяц

     \*/

    private ArrayList<Double> salaries = new ArrayList<>(12);

    /\*\*

     \* Количество отработанных месяцев

     \*/

    private int workMonths; // count of work months in year (payments >0)

    /\*\*

     \* Максимальная зарплата за последний отработанный год

     \*/

    private double maxSalary;

    /\*\*

     \* Коэффициент "месяца"

     \*/

    private final double koef = 29.6;

    /\*\*

     \* Предел рассчитанной зарплаты, ниже которой

     \* предоставляется скидка для снижения подоходного налога

     \*/

    private final double benefitSalary = 944;

    /\*\*

     \* Скидка в рублях на одного несовершеннолетнего ребенка

     \* или студента очного отделения

     \*/

    private final double kidKoef = 46;

    /\*\*

     \* Процент отчислений в пенсионных фонд

     \*/

    private final double pensKoef = 0.01;

    /\*\*

     \* Процент отчислений в профсоюз

     \*/

    private final double tradeUnionKoef = 0.01;

    /\*\*

     \* Подоходный налог

     \*/

    private final double incomeTaxKoef = 0.13;

    /\*\*

     \* Конструктор, который инициализирует поле salaries,

     \* ищет максимальную зарплату,

     \* считает количество отработанных месяцев

     \* @param salaries

     \*/

    public YearSalary(ArrayList<Double> salaries) {

        this.salaries = salaries;

        this.workMonths = findWorkMonths();

        this.maxSalary = findMaxSalary();

    }

    /\*\*

     \* Расчитывает отпускную зарплату заданного кол-ва отпускных дней

     \* с учетом нахождения в профсоюзе и количества детей

     \* @param kidsCount количество несовершеннолетний детей или очно обущающихся детей-студентов

     \* @param isInTradeUnion является ли работник членом профсоюза

     \* @param vacationDaysCount количество отпускных дней

     \* @return количество отпускных дней

     \*/

    public double calculateVacationSalary(int kidsCount, boolean isInTradeUnion, int vacationDaysCount) {

        double vacationSalary = 0;

        double sum = 0;

        for (int i = 0; i < salaries.size(); i++) {

            if(salaries.get(i).isNaN()){

                continue;

            }

            sum += (this.maxSalary / salaries.get(i)) \* salaries.get(i);

        }

        double mid = sum / (workMonths \* koef); // средняя зп за день

        double preVacSal = mid \* vacationDaysCount; // безнаоговые отпускные

        double tax = (preVacSal - kidsCount \* kidKoef - 156 \* (preVacSal < benefitSalary ? 1 : 0)) \* incomeTaxKoef; // подоходный налог

        double pensTax = preVacSal \* pensKoef; //пенсионные отчисления

        double tradeUnionTax = preVacSal \* tradeUnionKoef; // профсоюзные отчисления

        vacationSalary = preVacSal - tax - pensTax - tradeUnionTax \* (isInTradeUnion == true ? 1 : 0); // отпускные

        if (Double.isNaN(vacationSalary)) //если числа НаН, то отпускные "не начислены"

            vacationSalary = 0.0;

        return vacationSalary;

    }

    /\*\*

     \* Ищет максимальную зарплату

     \* @return максмальное значение в списке

     \*/

    private double findMaxSalary() {

        double max = 0; // max of salaries

        for (int i = 0; i < salaries.size(); i++) {

            if (salaries.get(i) > max) {

                max = salaries.get(i);

            }

        }

        return max;

    }

    /\*\*

     \* считает количество отработанных месяцев

     \* @return количество отработанных месяцев

     \*/

    private int findWorkMonths() {

        int workMonths = 0;

        for (int i = 0; i < salaries.size(); i++) {

            if (salaries.get(i) > 0)

            {

                workMonths++;

            }

        }

        return workMonths;

    }

}

Файл About.java

package windows.InformationalWindows;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

 \* Класс, представляющий окно "О программе" с информацией о программе.

 \*/

public class About extends JFrame {

    /\*\*

     \* Конструктор класса About, инициализирующий окно "О программе".

     \*

     \* @author Barkovets R. S.

     \* @version 1.0

     \*/

    public About() {

        super("О программе");

        setSize(600, 300);

        setResizable(false);

        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); // Закрывать только текущее окно

        setLayout(null);

        setLocationRelativeTo(null);

        Font aboutProgramWindowFont\_1 = new Font("Times New Roman", Font.BOLD, 14);

        JLabel lblNameOfProgram = new JLabel("Расчет начисления заработной платы за дни отпуска");

        JLabel lblAboutProgramAbility = new JLabel("Программа позволяет:");

        JLabel lblAboutEaseEnteringData = new JLabel("1. Считать отпускные выплаты за рабочий период");

        JLabel lblAboutEaseContainData = new JLabel("2. Создавать персонализированный отчёт для работника");

        JLabel lblAboutEaseUploadData = new JLabel("3. Сохранять отчёт в файл");

        setFont(aboutProgramWindowFont\_1);

        lblNameOfProgram.setBounds(200, 30, 350, 20);

        lblAboutProgramAbility.setBounds(200, 70, 500, 20);

        lblAboutEaseEnteringData.setBounds(200, 85, 500, 20);

        lblAboutEaseContainData.setBounds(200, 100, 500, 20);

        lblAboutEaseUploadData.setBounds(200,115,500,20);

        JLabel lblIcon = new JLabel();

        ImageIcon img = new ImageIcon("C:\\Users\\roman\\VSCodeProjects\\VacationPaymentsCalculator\\images\\splash.png");

        lblIcon.setIcon(img);

        lblIcon.setBounds(20, 50, 150, 150);

        JButton btnBack = new JButton("Выход");

        btnBack.setBounds(480, 240, 100, 20);

        btnBack.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // При нажатии на "Назад" делаем окно "О программе" невидимым

                setVisible(false);

            }

        });

        add(lblNameOfProgram);

        add(lblAboutProgramAbility);

        add(lblAboutEaseEnteringData);

        add(lblAboutEaseContainData);

        add(lblAboutEaseUploadData);

        add(lblIcon);

        add(btnBack);

    }

    /\*\*

     \* Главный метод, запускающий окно "О программе".

     \*

     \* @param args Параметры командной строки (не используются).

     \*/

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                new About().setVisible(true);

            }

        });

    }

}

Файл Author.java

package windows.InformationalWindows;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

 \* Класс, представляющий окно с информацией об авторе.

 \*/

public class Author extends JFrame {

    /\*\*

     \* Конструктор класса Author, инициализирующий окно с информацией об авторе.

     \*

     \* @author Barkovets R. S.

     \* @version 1.0

     \*/

    public Author() {

        super("Об авторе");

        setSize(270, 410);

        setResizable(false);

        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); // Закрывать только текущее окно

        setLayout(null);

        setLocationRelativeTo(null);

        JLabel lblAuthorWindowData\_1 = new JLabel("Автор");

        lblAuthorWindowData\_1.setBounds(110, 250, 75, 20);

        JLabel lblAuthorWindowData\_2 = new JLabel("Студент группы 10702221");

        lblAuthorWindowData\_2.setBounds(50, 270, 200, 20);

        JLabel lblAuthorWindowData\_3 = new JLabel("Барковец Роман Сергеевич");

        lblAuthorWindowData\_3.setBounds(45, 290, 230, 20);

        JLabel lblAuthorWindowData\_4 = new JLabel("romanbarkovets@yandex.ru");

        lblAuthorWindowData\_4.setBounds(45, 310, 200, 20);

        JButton cancel = new JButton("Выход");

        add(lblAuthorWindowData\_1);

        add(lblAuthorWindowData\_2);

        add(lblAuthorWindowData\_3);

        add(lblAuthorWindowData\_4);

        cancel.setBounds(80, 335, 100, 30);

        cancel.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // При нажатии на "Назад" делаем окно автора невидимым

                setVisible(false);

            }

        });

        add(cancel);

        JLabel foto = new JLabel();

        ImageIcon image = new ImageIcon("C:\\Users\\roman\\VSCodeProjects\\VacationPaymentsCalculator\\images\\avatar.jpg");

        foto.setIcon(image);

        // Устанавливаем положение и размер метки с изображением

        foto.setBounds(30, 0, 200, 250);

        add(foto);

    }

    /\*\*

     \* Главный метод, запускающий приложение.

     \*

     \* @param args Параметры командной строки (не используются).

     \*/

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                new Author().setVisible(true);

            }

        });

    }

}

Файл MainWindow.java

package windows.InformationalWindows;

import java.awt.Dimension;

import javax.swing.BoxLayout;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import windows.MainWindows.ButtonsPanel;

import windows.MainWindows.InformationPanel;

import windows.MainWindows.PeriodPanel;

import windows.MainWindows.YearSalaryPanel;

/\*\*

 \* Рабочее окно приложения, которое содержит в себе всю важную логику

 \*/

public class MainWindow extends JFrame {

    public static MainWindow instance;

    public void closeWindow(){

        this.dispose();

    }

    // add constructor with all main windows

    public MainWindow() {

        CreateWindow(520, 610);

        add(new PeriodPanel());

        add(new YearSalaryPanel());

        add(new InformationPanel());

        add(new ButtonsPanel());

        setVisible(true);

    }

    // creates window for other panels

    private void CreateWindow(int width, int length) {

        setTitle("Калькулятор отпускных");

        setSize(new Dimension(width, length));

        setResizable(false); // prevent from remaining

        setLayout(new BoxLayout(this.getContentPane(), BoxLayout.Y\_AXIS));

        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setIconImage(new ImageIcon("C:\\Users\\roman\\VSCodeProjects\\VacationPaymentsCalculator\\images\\icon.png")

                .getImage());

        setVisible(true);

    }

    public static void main(String[] args) {

        MainWindow mw= new MainWindow();

        MainWindow.instance = mw;

    }

}

Файл SplashScreen.java

package windows.InformationalWindows;

import javax.swing.\*;

import javax.swing.JFrame;

import javax.swing.ImageIcon;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

 \* Класс, представляющий стартовое окно приложения.

 \*/

public class SplashScreen extends JFrame {

    /\*\*

     \* Конструктор класса Start, инициализирующий стартовое окно приложения.

     \*

     \* @author Barkovets R. S.

     \* @version 1.0

     \*/

    public SplashScreen() {

        super("Стартовое окно");

        this.setSize(560, 500);

        // Установка операции закрытия окна по умолчанию

        this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        // Запрет изменения размера окна пользователем

        this.setResizable(false);

        Timer timer = new Timer(60000, e -> dispose());

        timer.start();

        setLocationRelativeTo(null);

        JLabel startWindowLabelData\_1 = new JLabel("Беларусский национальный технический университет");

        JLabel startWindowLabelData\_2 = new JLabel("Факультет информационных технологий и робототехники");

        JLabel startWindowLabelData\_3 = new JLabel("Кафедра программного обеспечения информационных систем и технологий");

        JLabel startWindowLabelData\_4 = new JLabel("Курсовая работа");

        JLabel startWindowLabelData\_5 = new JLabel("по дисциплине \"программирование на языке Java\"");

        JLabel startWindowLabelData\_6 = new JLabel("Расчет начисления отпускной заработной платы");

        JLabel startWindowLabelData\_7 = new JLabel("Выполнил: Студент группы 10702221");

        JLabel startWindowLabelData\_8 = new JLabel("Барковец Роман Сергеевич");

        JLabel startWindowLabelData\_9 = new JLabel("Преподователь: к.ф.-м.н., доц.");

        JLabel startWindowLabelData\_10 = new JLabel("Сидорик Валерий Владимирович");

        JLabel startWindowLabelData\_11 = new JLabel("Минск, 2023");

        // Настройка шрифта для меток

        Font startWindowLabelFont\_1 = new Font("Times New Roman", Font.BOLD, 14);

        Font startWindowLabelFont\_2 = new Font("Times New Roman", Font.BOLD, 20);

        Font startWindowLabelFont\_3 = new Font("Times New Roman", Font.BOLD, 18);

        // Установка шрифта для меток

        startWindowLabelData\_1.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_2.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_3.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_4.setFont(startWindowLabelFont\_2);

        startWindowLabelData\_5.setFont(startWindowLabelFont\_3);

        startWindowLabelData\_6.setFont(startWindowLabelFont\_2);

        startWindowLabelData\_7.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_8.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_9.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_10.setFont(startWindowLabelFont\_1);

        startWindowLabelData\_11.setFont(startWindowLabelFont\_1);

        JButton start = new JButton("Запустить");

        JButton stop = new JButton("Закончить");

// Создание панели для компоновки компонентов

        JPanel startWindowPanel = new JPanel();

        // Установка менеджера компоновки в null, что позволяет явно задавать позиции компонентов

        startWindowPanel.setLayout(null);

        // Установка расположения меток

        startWindowLabelData\_1.setBounds(80, 10, 370, 20);

        startWindowLabelData\_2.setBounds(75, 30, 400, 20);

        startWindowLabelData\_3.setBounds(15, 50, 520, 20);

        startWindowLabelData\_4.setBounds(200, 150, 200, 20);

        startWindowLabelData\_5.setBounds(50, 175, 450, 20);

        startWindowLabelData\_6.setBounds(50, 200, 500, 20);

        startWindowLabelData\_7.setBounds(280, 250, 250, 20);

        startWindowLabelData\_8.setBounds(280, 270, 280, 20);

        startWindowLabelData\_9.setBounds(280, 310, 250, 20);

        startWindowLabelData\_10.setBounds(280, 330, 250, 20);

        startWindowLabelData\_11.setBounds(225, 400, 100, 20);

        start.setBounds(175, 430, 100, 25);

        stop.setBounds(275, 430, 100, 25);

        // Добавление меток в панель

        startWindowPanel.add(startWindowLabelData\_1);

        startWindowPanel.add(startWindowLabelData\_2);

        startWindowPanel.add(startWindowLabelData\_3);

        startWindowPanel.add(startWindowLabelData\_4);

        startWindowPanel.add(startWindowLabelData\_5);

        startWindowPanel.add(startWindowLabelData\_6);

        startWindowPanel.add(startWindowLabelData\_7);

        startWindowPanel.add(startWindowLabelData\_8);

        startWindowPanel.add(startWindowLabelData\_9);

        startWindowPanel.add(startWindowLabelData\_10);

        startWindowPanel.add(startWindowLabelData\_11);

        startWindowPanel.add(start);

        startWindowPanel.add(stop);

        this.add(startWindowPanel);

        this.setVisible(true);

        // Создание метки для отображения изображения

        JLabel icon = new JLabel();

        ImageIcon img = new ImageIcon("C:\\Users\\roman\\VSCodeProjects\\VacationPaymentsCalculator\\images\\splash.png");

        // Установка изображения для метки

        icon.setIcon(img);

        // Установка позиции и размеров метки на панели

        icon.setBounds(40, 222, 223, 150);

        startWindowPanel.add(icon);

        start.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Скрыть стартовое окно

                setVisible(false);

                // Создание и отображение главного окна

                MainWindow mainWindow = new MainWindow();

                mainWindow.setVisible(true);

            }

        });

        stop.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Закрыть стартовое окно

                dispose();

            }

        });

    }

    /\*\*

     \* Главный метод, запускающий приложение.

     \*

     \* @param args Параметры командной строки (не используются).

     \*/

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                SplashScreen app = new SplashScreen();

                app.setVisible(true);

            }

        });

    }

}

Файл ButtonsPanel.java

package windows.MainWindows;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JPanel;

import listeners.CalculateButtonListener;

import listeners.ClearButtonListener;

import listeners.SaveButtonListener;

import windows.InformationalWindows.MainWindow;

import windows.SecondaryWindows.DoubledButtonPanel;

/\*\*

 \* Класс, представляющий панель с рабочими кнопками

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class ButtonsPanel extends JPanel {

    /\*\*

     \* Кнопка для печати отчёта

     \*/

    private JButton jbReportToFile = new JButton("Отчёт");

    /\*\*

     \* Кнопка для подсчета

     \*/

    private JButton jbCalculate = new JButton("Подсчёт");

    /\*\*

     \* Кнопка для очистки данных

     \*/

    private JButton jbClear = new JButton("Очистка");

    /\*\*

     \* Кнопка выхода

     \*/

    private JButton jbExit = new JButton("Выход");

    /\*\*

     \* Двойная кнопка

     \*/

    private DoubledButtonPanel dbp = new DoubledButtonPanel();

    /\*\*

     \* стандартная ширина всей панели

    \*/

    private int width = 490;// 490

    /\*\*

     \* стандартная высота всей панели

     \*/

    private int height = 53;// 43

    /\*\*

     \* стандартная ширина любой кнопки

     \*/

    private int btWidth = 90; // 115

    /\*\*

     \* стандартная высота любой кнопки

     \*/

    private int btHeight = 43; // 43

    /\*\*

    \* Конструктор класса, добавляющий кнопки на панель,

    \* располагающий их последовательно,

    \* добавляющий слушателей нажатия на каждую кнопку

    \*/

    public ButtonsPanel() {

        setBorder(BorderFactory.createEmptyBorder(10, 0, 0, 0));

        setLayout(new FlowLayout(FlowLayout.CENTER, 7, 0));

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setButtonsSizes();

        add(jbReportToFile);

        add(jbCalculate);

        add(jbClear);

        add(jbExit);

        add(dbp);

        setVisible(true);

        jbReportToFile.addActionListener(new SaveButtonListener());

        jbCalculate.addActionListener(new CalculateButtonListener());

        jbClear.addActionListener(new ClearButtonListener());

        jbExit.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e){

                MainWindow.instance.closeWindow();

            }

        });

    }

    /\*\*

     \* задает всем кнопкам стандартные размеры

     \*/

    private void setButtonsSizes() {

        jbReportToFile.setPreferredSize(new Dimension(btWidth, btHeight));

        jbCalculate.setPreferredSize(new Dimension(btWidth, btHeight));

        jbClear.setPreferredSize(new Dimension(btWidth, btHeight));

        dbp.setPreferredSize(new Dimension(btWidth, btHeight));

        jbExit.setPreferredSize(new Dimension(btWidth, btHeight));

    }

}

Файл InformationPanel.java

package windows.MainWindows;

import java.awt.BorderLayout;

import java.awt.Dimension;

import javax.swing.BorderFactory;

import javax.swing.JPanel;

import listeners.CalculateButtonListener;

import windows.SecondaryWindows.InformationPlace;

/\*\*

 \* Класс, представляющий панель для отображения информации

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class InformationPanel extends JPanel {

    /\*\*

     \* Место для текста

     \*/

    private InformationPlace infoPlace = new InformationPlace();

    /\*\*

     \* стандартная ширина элемента

     \*/

    private int width = 490; // 490

    /\*\*

     \* стандартная высота элемента

     \*/

    private int height = 132; // 132

    /\*\*

     \* Конструктор добавляющий на панель текстовую область

     \* и инициализирующий поле CalculateButtonListener.infoPanel,

     \* чтобы иметь возможность редактировать ее содержимое

     \*/

    public InformationPanel() {

        // setBorder(BorderFactory.createLineBorder(java.awt.Color.BLACK, 1));

        setBorder(BorderFactory.createEmptyBorder(10, 0, 0, 0));

        setBorder(BorderFactory.createTitledBorder("Информация"));

        // setLayout(new FlowLayout(FlowLayout.CENTER, 10,10));

        setLayout(new BorderLayout());

        add(infoPlace, BorderLayout.CENTER);

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setVisible(true);

        CalculateButtonListener.infoPanel = this;

    }

    /\*\*

     \* Добавляет текст на панель

     \* @param text добавляется на панель

     \*/

    public void addInfo(String text) {

        infoPlace.addInfo(text);

    }

    /\*\*

     \* Полность очищает панель

     \*/

    public void clearAllInfo() {

        infoPlace.clearAllInfo();

    }

}

Файл PeriodPanel.java

package windows.MainWindows;

import java.awt.Dimension;

import java.awt.GridLayout;

import javax.swing.JPanel;

import listeners.CalculateButtonListener;

import windows.SecondaryWindows.DatePanel;

import windows.SecondaryWindows.FullNameInputPanel;

import windows.SecondaryWindows.KidsCounter;

/\*\*

 \* Класс, представляющий панель для содержания на ней дат, работника, счетчика детей

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class PeriodPanel extends JPanel {

    /\*\*

     \* Панель для ввода начала отпуска

     \*/

    private DatePanel dpFrom = new DatePanel("Отпуск с...");

    /\*\*

     \* панель для ввода конца отпуска

     \*/

    private DatePanel dpTo = new DatePanel("По...");

    /\*\*

     \* поле для ввода ФИО работника

     \*/

    private FullNameInputPanel nameInputPanel = new FullNameInputPanel("Введите имя сотрудника:");

    /\*\*

     \* поле для отображения кол-ва детей

     \*/

    private KidsCounter kidsCounterPanel = new KidsCounter("Введите количество детей: ");

    /\*\*

     \* стандартная ширина

     \*/

    private int width = 490; // 490

    /\*\*

    \* стандартная высота

    \*/

    private int height = 128; // 128

    /\*\*

     \* Конструктор, создающий панель фиксированных размеров

     \*/

    public PeriodPanel() {

        setLayout(new GridLayout(2, 2));

        setPreferredSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        add(dpFrom);

        add(nameInputPanel);

        add(dpTo);

        add(kidsCounterPanel);

        setVisible(true);

        CalculateButtonListener.datePanelFrom = dpFrom;

        CalculateButtonListener.datePanelTo = dpTo;

    }

}

Файл YearSalaryPanel.java

package windows.MainWindows;

import java.awt.Dimension;

import java.awt.GridLayout;

import java.util.ArrayList;

import javax.swing.JPanel;

import listeners.CalculateButtonListener;

import listeners.ClearButtonListener;

import windows.SecondaryWindows.MonthSalaryPanel;

/\*\*

 \* Класс представляющий панель для ввода зарплаты за 12 месяцев

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class YearSalaryPanel extends JPanel {

    /\*\*

     \* Количество месяцев в году

     \*/

    private final int MONTH\_COUNT = 12;

    /\*\*

     \* Список порядковых номеров месяцев

     \*/

    private final String[] MONTH\_NAMES = new String[] { "Первый", "Второй", "Третий", "Четвертый", "Пятый", "Шестой",

            "Седьмой",

            "Восьмой", "Девятый", "Десятый", "Одиннадцатый", "Двенадцатый", };

    /\*\*

     \* Зарплатные ячейки

     \*/

    private ArrayList<MonthSalaryPanel> salaryCells = new ArrayList<MonthSalaryPanel>(MONTH\_COUNT);

    /\*\*

     \* Стандартная ширина

     \*/

    private int width = 490; // 490

    /\*\*

     \* Стандартная высота

     \*/

    private int height = 247; // 247

    /\*\*

     \* Конструктор, создающий панель фиксированной величины

     \*/

    public YearSalaryPanel() {

        setLayout(new GridLayout(4, 3, 10, 10));

        for (int i = 0; i < MONTH\_COUNT; i++) {

            MonthSalaryPanel panel = new MonthSalaryPanel(MONTH\_NAMES[i]);

            salaryCells.add(panel);

            add(panel);

        }

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setVisible(true);

        CalculateButtonListener.ysPanel = this;

        ClearButtonListener.ysPanel = this;

    }

    /\*\*

     \* Получает значение каждой зарплатной панели

     \* @return массив строк, которые нужно распарсить в Double

     \*/

    public ArrayList<String> getMonthSalaries() {

        ArrayList<String> sals = new ArrayList<>(12);

        for (int i = 0; i < MONTH\_COUNT; i++) {

            sals.add(salaryCells.get(i).getTAreaValue());

        }

        return sals;

    }

    /\*\*

     \* Устанавливает месячную зарплату

     \* @param index для конкретного месяца

     \* @param text зароботная плата

     \*/

    public void setMonthSalary(int index, String text) {

        salaryCells.get(index).setTAreaValue(text);

    }

}

Файл DatePanel.java

package windows.SecondaryWindows;

import java.awt.Dimension;

import java.awt.GridLayout;

import javax.swing.\*;

import java.util.Date;

import java.util.Calendar;

/\*\*

 \* Класс представляющий маленькую панель для ввода даты

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class DatePanel extends JPanel {

    /\*\*

     \* Надпись на основным элементом

     \*/

    private JLabel jlCaption = new JLabel();

    /\*\*

     \* Спиннер для ввода даты

     \*/

    private JSpinner scpDate;

    /\*\*

     \* Стандартная ширина элемента

     \*/

    private int width = 259;

    /\*\*

     \* Стандартная высота элемента

     \*/

    private int height = 54;

    /\*\*

     \* Конструктор создающий маленькую панель для ввода како-либо даты

     \* @param caption надпись над полем

     \*/

    public DatePanel(String caption) {

        scpDate = new JSpinner(setSpinnerProperties());

        // Установка формата отображения даты

        JSpinner.DateEditor editor = new JSpinner.DateEditor(this.scpDate, "dd.MM.yyyy");

        this.scpDate.setEditor(editor);

        this.jlCaption.setText(caption);

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setLayout(new GridLayout(2, 0, 0, 0));

        add(jlCaption);

        add(scpDate);

        setVisible(true);

    }

    /\*\*

     \* Устанавливает вид JSpinner -> SpinnerDateModel

     \* @return

     \*/

    private SpinnerDateModel setSpinnerProperties() {

        SpinnerDateModel dateModel = new SpinnerDateModel(new Date(), null, null, Calendar.DAY\_OF\_MONTH);

        return dateModel;

    }

    /\*\*

     \* Получает поле для даты

     \* @return JSpinner, в котором находится дата

     \*/

    public JSpinner getSpinner() {

        return this.scpDate;

    }

}

Файл DoubledButtonPanel.java

package windows.SecondaryWindows;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import windows.InformationalWindows.About;

import windows.InformationalWindows.Author;

/\*\*

 \* Класс, реализующий панель с двумя тесно расположенными кнопками

 \*/

public class DoubledButtonPanel extends JPanel {

    /\*\*

     \* Кнопка для вызова окна "Об авторе"

     \*/

    private JButton jbAuthor = new JButton("Об авторе");

    /\*\*

     \* Кнопка для вызова окна "О программе"

     \*/

    private JButton jbAbout = new JButton("О программе");

    /\*\*

     \* стандартная ширина любой кнопки

     \*/

    private int width = 115; // 115

    /\*\*

     \* стандартная высота любой кнопки

     \*/

    private int height = 43; // 43

    /\*\*

     \* Коструктор создающий панель с двумя близко расположенными кнопками и привязывает их к событиям

     \*/

    public DoubledButtonPanel() {

        setLayout(new BoxLayout(this, BoxLayout.Y\_AXIS));

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        jbAbout.setPreferredSize(new Dimension(width, (height)/2));

        jbAbout.setMinimumSize(new Dimension(width, (height)/2));

        jbAbout.setMaximumSize(new Dimension(width, (height)/2));

        jbAuthor.setPreferredSize(new Dimension(width, (height)/2));

        jbAuthor.setMinimumSize(new Dimension(width, (height)/2));

        jbAuthor.setMaximumSize(new Dimension(width, (height)/2));

        add(jbAbout);

        add(jbAuthor);

        setVisible(true);

        jbAbout.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                About aboutProgramWindow= new About();

                aboutProgramWindow.setVisible(true);

            }

        });

        jbAuthor.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                Author aboutAuthorMenu = new Author();

                aboutAuthorMenu.setVisible(true);

            }

        });

    }

}

Файл FullNameInputPanel.java

package windows.SecondaryWindows;

import java.awt.Dimension;

import javax.swing.\*;

import listeners.CalculateButtonListener;

import listeners.ClearButtonListener;

import java.awt.GridLayout;

import java.awt.Color;

/\*\*

 \* Класс представляющий маленькую панель для ввода имени

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class FullNameInputPanel extends JPanel {

    /\*\*

     \* Надпись над основным элементом

     \*/

    private JLabel jlCaption = new JLabel();

    /\*\*

     \* Поля для ввода имени

     \*/

    private JTextArea jtaNameInput = new JTextArea();

    /\*\*

     \* Стандартная ширина

     \*/

    private int width = 259;

    /\*\*

     \* Стандартная высота

     \*/

    private int height = 54;

    /\*\*

     \* Создает маленькую панель для ввода имени

     \* @param caption надпись на панели

     \*/

    public FullNameInputPanel(String caption) {

        this.jlCaption.setText(caption);

        setBorder(BorderFactory.createEmptyBorder(5, 20, 5, 20));

        jtaNameInput.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setLayout(new GridLayout(2, 0, 0, 0));

        add(jlCaption);

        add(jtaNameInput);

        setVisible(true);

        CalculateButtonListener.nameInputPanel = this;

        ClearButtonListener.fniPanel = this;

    }

    /\*\*

     \* Получает имя работника из поля

     \*/

    public String getName() {

        return jtaNameInput.getText();

    }

    /\*\*

     \* Устанавливает имя работника

     \*/

    public void setName() {

        jtaNameInput.setText("");

    }

}

Файл InformationPlace.java

package windows.SecondaryWindows;

import java.awt.Dimension;

import javax.swing.JTextArea;

/\*\*

 \* Класс, представляющий панель для информации

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class InformationPlace extends JTextArea {

    /\*\*

     \* стандартная ширина

     \*/

    private int width = 490; // 490

    /\*\*

     \* стандартная высота

     \*/

    private int height = 120; // 120

    /\*\*

     \* создает место для непорседственного вода информации

     \*/

    public InformationPlace() {

        setEditable(false);

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setVisible(true);

    }

    /\*\*

     \* Добавляет текст на панель

     \* @param text для панели

     \*/

    public void addInfo(String text) {

        append(text + "\n");

    }

    /\*\*

     \* полностью очищает панель

     \*/

    public void clearAllInfo() {

        this.setText("");

    }

}

Файл KidsCounter.java

package windows.SecondaryWindows;

import javax.swing.\*;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import listeners.CalculateButtonListener;

import java.awt.\*;

/\*\*

 \* Класс, представляющий панель со счетчиком детей

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class KidsCounter extends JPanel {

    /\*\*

     \* Надпись над основным элементом

     \*/

    private JLabel jlCaption = new JLabel();

    /\*\*

     \* Счетчик детей

     \*/

    private JSpinner sjCounter;

    /\*\*

     \* стандартная ширина

     \*/

    private int width = 259; // 259

    /\*\*

     \* стандартная высота

     \*/

    private int height = 54;

    /\*\*

     \* Создает счетчик детей с надписью сверху

     \* @param caption надпись

     \*/

    public KidsCounter(String caption) {

        this.jlCaption.setText(caption);

        SpinnerNumberModel snm = new SpinnerNumberModel();

        sjCounter = new JSpinner(snm);

        sjCounter.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        setBorder(BorderFactory.createEmptyBorder(5, 20, 5, 20));

        sjCounter.addChangeListener(new ChangeListener() {

            @Override

            public void stateChanged(ChangeEvent e) {

                if ((int) sjCounter.getValue() < 0) {

                    sjCounter.setValue(0);

                    return;

                }

                if ((int) sjCounter.getValue() > 10) {

                    sjCounter.setValue(10);

                    return;

                }

            }

        });

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setLayout(new GridLayout(2, 0, 0, 0));

        add(jlCaption);

        add(sjCounter);

        setVisible(true);

        CalculateButtonListener.kidsCounter = this;

    }

    /\*\*

     \* Получает количество детей

     \* @return значение счетчика

     \*/

    public int getKidsCount() {

        return (int) sjCounter.getValue();

    }

}

Файл MonthSalaryPanel.java

package windows.SecondaryWindows;

import java.awt.Dimension;

import java.awt.GridLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextArea;

/\*\*

 \* Класс, представляющий панель для информации

 \*

 \* @author Barkovets R. S.

 \* @version 1.0

 \*/

public class MonthSalaryPanel extends JPanel {

    /\*\*

     \* Надпись над полем

     \*/

    private JLabel jlMonthName = new JLabel();

    /\*\*

     \* Место для ввода зарплаты за месяц

     \*/

    private JTextArea taSalaryInput = new JTextArea();

    /\*\*

     \* стандартная ширина

     \*/

    private int width = 150; // 150

    /\*\*

     \* стандартная высота

     \*/

    private int height = 47; // 47

    /\*\*

     \* Создает ячейку для ввода зарплаты за месяц

     \* @param name надпись сверху

     \*/

    public MonthSalaryPanel(String name) {

        jlMonthName.setText(name);

        // setBorder(BorderFactory.createLineBorder(java.awt.Color.BLACK, 1));

        setPreferredSize(new Dimension(width, height));

        setMinimumSize(new Dimension(width, height));

        setMaximumSize(new Dimension(width, height));

        setLayout(new GridLayout(2, 1));

        add(jlMonthName);

        add(taSalaryInput);

        setVisible(true);

    }

    /\*\*

     \* Получает значение зарплаты

     \* @return содержимоетекстового поля

     \*/

    public String getTAreaValue() {

        return this.taSalaryInput.getText();

    }

    /\*\*

     \* Меняет содержимое текстового поля

     \* @param text текст для изменения

     \*/

    public void setTAreaValue(String text) {

        this.taSalaryInput.setText(text);

    }

}